101 Problema Programimi

Dashamir Hoxha

2019-05-29

Contents

1 Parathënie

Ky libër është për fillestarët në programim, qofshin nxënës të shkollës 9-vjeçare ose të shkollës së mesme, apo të tjerë të çdo moshe. Konceptet e programimit mësohen hap pas hapi, nëpërmjet zgjidhjes së ushtrimeve dhe problemave, të cilat fillojnë shumë të thjeshta dhe vijnë gradualisht duke u bërë më interesante.

Në fakt, synimi i këtij libri është që ti aftësojë lexuesit në zgjidhjen e problemave të programimit, dhe jo që tju mësojë ndonjë gjuhë të veçantë programimi. Gjuha e programimit mësohet vetvetiu si një mjet i nevojshëm për të shprehur zgjidhjen e problemit. Kështu që mësimi i veçorive të gjuhës së programimit nuk do të jetë shterues, do të mësojmë vetëm aq sa na duhet.

Po ashtu, edhe mësimi i teknikave algoritmike nuk do të jetë shterues, do të mësohen vetëm ato që na duhen për problemat përkatëse. Kurse vetë problemat mund të jenë shpeshherë sfiduese dhe të kërkojnë disa njohuri matematike dhe aftësi të forta llogjike për tu zgjidhur. Në fakt, shumica e këtyre problemave janë marrë prej olimpiadave të programimit që zhvillohen online, dhe shpresoj se do të jenë zbavitëse, por njëkohësisht edhe do ti ndihmojnë studentët që të aftësohen për të konkuruar në këto olimpiada. Ose të paktën do ti ndihmojnë në hapat e para në këtë drejtim.

Gjuha e përdorur për zgjidhjen e problemave është Python (ose më saktë Python3), si një gjuhë e thjeshtë për fillestarët, me aftësi shprehëse të fuqishme, e një niveli pak më të lartë se gjuhët e tjera të nivelit të lartë, por edhe që përdoret gjerësisht në praktikë, në çdo fushë të programimit, duke përfshirë aplikacionet dekstop, aplikacionet shkencore, aplikacionet web, inteligjencën artificiale, robotikën, etj. Në fakt, Python është gjuha që përdoret nga Google për motorrin e tij të famshëm të kërkimit, i cili sistemon informacionin e gjithë botës, për YouTube, etj. A doni më për Python?

Si një mjedis pune për fillestarët, unë sugjeroj që të instalohet LinuxMint Xfce në një makinë virtuale (VirtualBox), i cili e ka të instaluar vetvetiu Python3 dhe çdo gjë tjetër që mund të na duhet. Lidhjet dhe videot e mëposhtme mund tju ndihmojnë për ta bërë këtë:

- Download Linux Mint
- How to Install LinuxMint in VirtualBox
- How to Dual-Boot Windows10 and LinuxMint
- How to Create a Bootable USB for Installing LinuxMint

Edhe pse nuk është e nevojshme që të studjohet ndonjë tutorial ose manual për Python para se të fillohet ky libër, po i jap gjithsesi disa referenca për ata që dëshirojnë të thellohen më tepër në detajet e gjuhës Python, para ose pasi ta kenë lexuar këtë libër:

- learnpython.org
- Learn Python Programming
- Python Programming Tutorial
- How to Think Like a Computer Scientist

Na vaftë mbarë!

Versioni

Versioni më i fundit i këtij libri gjendet në http://p101.fs.al/ ose në http://dashohoxha.gitlab.io/101-problema-programimi/

Rregullimet ose ndryshimet mund të sugjerohen te https://gitlab.com/dashohoxha/101-problema-programimi

Licensa

Problemi 000

Kërkesa

Shkruani një program që merr emrin, mbiemrin, dhe qytetin ku jetoni, dhe nxjerr në ekran:

```
Ju jeni <emri> <mbiemri> nga <qyteti>.
```

Zgjidhja 1

```
# lexojmë nga tastiera emrin, mbiemrin dhe qytetin
print('Emri: ', end='')
emri = input()
print('Mbiemri: ', end='')
mbiemri = input()
```

```
print('Qyteti: ', end='')
qyteti = input()

# nxjerim në ekran fjalinë e kërkuar
print('Ju jeni', emri, mbiemri, 'nga', qyteti)
```

https://tinyurl.com/101-prog-000

Sqarime

Funksioni `print()` përdoret për të nxjerrë diçka (në ekran), kurse funksioni `input()` për të lexuar diçka (nga tastiera). Parametri `end=''` i tregon funksionit `print()` që në fund të mos printojë asgjë, sepse zakonisht ai printon edhe një *fund-rreshti* (*EOL* – End Of Line), i cili bën që kursori të kaloj në rreshtin më poshtë. Kur i japim funksionit `print()` disa parametra, ai i printon ato me nga një vend bosh në mes.

Hapni terminalin dhe shkruajeni programin brenda direktorisë `prog/p000`:

```
mkdir -p prog/p000
cd prog/p000/
nano p000.py
```

Komanda `mkdir` krijon një direktori të re (`make directory`), kurse komanda `cd` ndryshon direktorinë e punës (`change directory`).

Në editorin `nano` mund të përdorni kombinimin e tastave `Ctrl+O` për ta ruajtur programin, dhe `Ctrl+X` për të dalë. Në vend të editorit `nano` mund të përdorni edhe *Text Editor* ose ndonjë editor tjetër.

Zbatojeni këtë program nga terminali duke përdorur komandën:

```
python3 p000.py
```

Zgjidhja 2

```
# lexojmë emrin, mbiemrin dhe qyetin
emri = input()
mbiemri = input()
qyteti = input()

# nxjerim në ekran fjalinë e kërkuar
print('Ju jeni {} {} nga {}.'.format(emri, mbiemri, qyteti))
```

https://tinyurl.com/101-prog-000-2

Sqarime

Hapni terminalin dhe shkruajeni programin brenda direktorisë `prog/p000/`:

```
cd prog/p000/
nano p000.2.py
```

Krijoni edhe skedarin `input.txt` me një përmbajtje të ngjashme si kjo:

```
Ismail
Qemali
Vlora
```

Zbatojeni këtë program nga terminali duke përdorur komandën:

```
python3 p000.py < input.txt
```

Rezultati do jetë si ky:

```
Ju jeni Ismail Qemali nga Vlora.
```

Në këtë rast, me anë të shenjës `<` ne kemi *ridrejtuar* inputin e programit, në mënyrë që të lexojë nga skedari `input.txt` dhe jo nga tastiera. Pra te skedari `input.txt` ne mund të vendosim gjithçka që do ti jepnim programit nga tastiera, dhe programi do ta lexojë njëlloj sikur ti merrte nga tastiera. Funksioni `input()` lexon nga ky skedar një rresht të plotë deri në fund.

Këtej e tutje ne do përdorim këtë metodën me ridrejtim, sepse është më praktike, të lejon ta shkruash inputin njëherë dhe ta testosh programin shumë herë. Gjithashtu lejon që vetë problemi të përcaktojë një input shembull, bashkë me rezultatin përkatës të programit.

Detyra

Shkruani një program që kërkon muajin dhe vitin e lindjes, dhe nxjerr në ekran:

```
Ju keni lindur në muajin <muaji> të vitit <viti>.
```

2 Problemat 001-010

Problemi 001

Kërkesa

Bëni një program që merr 5 numra dhe nxjerr katrorin e tyre (fuqinë e dytë).

Shembull

```
$ cat input.txt
13
7
3
11
5

$ python3 p001.py < input.txt
169
49
9
121
25
```

Shënim: Komanda `cat` shfaq në ekran përmbajtjen e një skedari.

Zgjidhja 1

```
n = int(input())
print(n * n)
n = int(input())
print(n * n)
n = int(input())
print(n * n)
n = int(input())
```

```
print(n * n)
n = int(input())
print(n * n)
```

https://tinyurl.com/101-prog-001-1

Sqarime

Ne duam të lexojmë një numër, por funksioni `input()` kthen një *string* (varg me shkronja), kështu që përdorim funksionin `int()` për ta kthyer në *integer* (numër të plotë).

Zgjidhja 2

```
for i in range(5):
    n = int(input())
    print(n * n)
```

https://tinyurl.com/101-prog-001-2

Sqarime

Duke përdorur një cikël **for** që përsëritet 5 herë, shmangim përsëritjen në program të atyre dy rreshtave.

Vini re që `:` në fund të rreshtit `for...` tregojnë që më poshtë vjen një *bllok* me *instruksione* (komanda, ose rreshta programi). Të gjithë rreshtat e këtij blloku janë të spostuar në brendësi të rreshtit `for...` me të njëjtin numër hapësirash. Ky bllok përsëritet sa herë që ti thotë komanda `for...` (në rastin tonë, 5 herë).

Funksioni `range(5)` kthen një objekt që ka 5 vlera, siç tregon edhe kjo provë:

```
$ python3
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> quit()
$ _
```

Detyra

Bëni një program që merr 5 numra dhe nxjerr kubin e tyre (fuqinë e tretë).

Problemi 002

Kërkesa

Bëni një program që merr një listë me emra dhe mbiemra dhe i nxjerr në formatin "Mbiemer, Emer".

Shembull

```
$ cat input.txt
3
Ada Lovelace
Charles Babbage
Alan Turing

$ python3 p002.py < input.txt
Lovelace, Ada
Babbage, Charles
Turing, Alan
```

Numri që ndodhet në rreshtin e parë të `input.txt` tregon se sa rreshta me emra vijnë më poshtë.

Zgjidhja

```python
T = int(input())
for t in range(T):
    emri, mbiemri = input().split()
    print("{}, {}".format(mbiemri, emri))
```

https://tinyurl.com/101-prog-002

Sqarime

Funksioni `split()`, i cili thirret te një *varg shkronjash* (*string*), e ndan atë varg në nënvargje sipas vendeve bosh, dhe krijon me to një objekt të listueshëm. P.sh.:

```
$ python3
>>> list('ab cd ef'.split())
['ab', 'cd', 'ef']
>>> quit()
```

Ndryshoret `emri` dhe `mbiemri` marrin vlerë njëkohësisht, me një vlerëdhënie të shumëfishtë (dyfishtë), ku `emri` merr vlerën e parë të një objekti të listueshëm, kurse `mbiemri` merr vlerën e dytë. P.sh.:

```
$ python3
>>> x, y, z = [10, 20, 30]
>>> x
10
>>> y
20
>>> z
30
>>> quit()
```

Funksioni `format()`, i cili thirret te një *varg*, zëvendëson vendmbajtësit `{}` brenda vargut me vlerat përkatëse që i jepen funksionit si parametra.

Detyra

Bëni një program që merr një listë me çifte numrash dhe nxjerr shumën e tyre.

Shembull

```
$ cat input.txt
3
5 7
20 12
136 2

$ python3 p002.2.py < input.txt
12
32
138
```

Problemi 003

Kërkesa

Bëni një program që merr vlerat e tre këndeve (në gradë) dhe nxjerr "YES" nëse këto mund të jenë kënde të një trekëndëshi, ose "NO" nëse nuk mund të jenë.

Referenca: https://www.codechef.com/problems/FLOW013

Shembull

```
$ cat input.txt
3
40 40 100
45 45 90
180 1 1

$ python3 p003.py < input.txt
YES
YES
NO
```

Zgjidhja 1

```python
T = int(input())
for t in range(T):
    a, b, c = input().split()
    if int(a) + int(b) + int(c) == 180:
        print("YES")
    else:
        print("NO")
```

https://tinyurl.com/101-prog-003-1

Sqarime

Këndet e një trekëndëshi e kanë shumën 180 gradë.

Meqënëse numrat i lexojmë si *varg* (*string*), kemi përdorur funksionin `int()` për ti kthyer në numra (përndryshe nuk mund ti mbledhim).

Zgjidhja 2

```python
T = int(input())
for t in range(T):
    a, b, c = map(int, input().split())
    if a + b + c == 180:
        print("YES")
    else:
        print("NO")
```

https://tinyurl.com/101-prog-003-2

Sqarime

Funksioni `map()` që është përdorur këtu, zbaton funksionin `int()` te të gjithë elementet e objektit të listueshëm. Si përfundim, në ndryshoret `a`, `b` dhe `c` ruhen numra, dhe jo vargje.

Detyra

Bëni një program që merr 2 numra `a` dhe `b` dhe nxjerr:

- shenjën `<` nëse `a` është më i vogël se `b`
- shenjën `>` nëse `a` është më i madh se `b`
- shenjën `=` nëse `a` është i barabartë me `b`

Referenca: https://www.codechef.com/problems/CHOPRT

Shembull

```
$ cat input.txt
3
10 20
20 10
10 10

$ python3 p003.3.py < input.txt


=
```

Problemi 004

Kërkesa

Në një varg numrash $a_1, a_2, ..., a_n$ gjeni vlerën më të vogël të mundshme të shprehjes $a_i + a_j$, ku $1 \leq i < j \leq n$.

Referenca: https://www.codechef.com/problems/SMPAIR

Shembull

```
$ cat input.txt
1
4
5 1 3 4

$ python3 prog.py < input.txt
4
```

Zgjidhja

```
T = int(input())
for t in range(T):
    n = int(input())
    l = list(map(int, input().split()))
    l.sort()
    print(l[0] + l[1])
```

https://tinyurl.com/101-prog-004

Sqarime

Te lista `l` lexojmë vargun e numrave. Funksioni `sort()`, i cili zbatohet te lista, e rendit listën në rendin rritës. Meqënëse lista e numrave është e renditur, dy të parët janë më të vegjlit, kështu që shuma e tyre është shuma më e vogël e mundshme.

Vini re që numri i parë i listës e ka indeksin **0** (`l[0]`), numri i dytë e ka indeksin **1**, e kështu me rradhë. Pra numërimi i elemeneteve të një liste fillon nga **0**.

Detyra

Në një varg numrash $a_1, a_2, ..., a_n$ gjeni vlerën më të **madhe** të mundshme të shprehjes $a_i + a_j$, ku $1 \leq i < j \leq n$.

Shembull

```
$ cat input.txt
1
4
5 1 3 4
```

```
$ python3 prog.py < input.txt
9
```

Problemi 005

Kërkesa

Jepet një varg me numra natyrorë: $a_1, a_2, ..., a_n$. A është e mundur që duke zëvendësuar vetëm k prej tyre me numra natyrorë të tjerë (sipas dëshirës), të bëjmë që ky kusht të jetë i vërtetë: $a_1^2 + a_2^2 + ... + a_n^2 \leq a_1 + a_2 + ... + a_n$

Referenca: https://www.codechef.com/problems/CHFAR

Shembull

```
$ cat input.txt
2
3 2
1 2 5
5 3
7 4 9 1 5

$ python3 prog.py < input.txt
YES
NO
```

Kemi 2 rast testimi. Në rastin e parë kemi `n=3` dhe `k=2`, dhe në rreshtin e tretë kemi `a1=1, a2=2, a3=5`. Nqs zëvendësojmë `a2=1` dhe `a3=1` atere mosbarazimi plotësohet.

Në rastin e dytë nuk ka mundësi që mosbarazimin ta bëjmë të vërtetë vetëm me 3 zëvendësime.

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n, k = map(int, input().split())
    A = list(map(int, input().split()))
    nr = 0
    for a in A:
```

```
        if a > 1:
            nr += 1
    if nr > k:
        print('NO')
    else:
        print('YES')
```

https://tinyurl.com/101-prog-005-1

Sqarime

Mosbarazimi i dhënë mund të plotësohet vetëm nëse të gjithë numrat e vargut janë `1` (dhe në këtë rast është barazim).

Te lista `A` mbajmë vargun e numrave, dhe gjejmë sa prej elementeve të listës janë më të mëdhenj se `1`. Nëse ky numër është më i madh se `k` (numri i elementëve të vargut që mund të zëvendësojmë), atëherë përgjigja është `NO` (nuk mund ta bëjmë të vërtetë mosbarazimin). Përndryshe përgjigja është `YES`.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    n, k = map(int, input().split())
    A = list(map(int, input().split()))
    nr = len([ a for a in A if a > 1 ])
    if nr > k:
        print('NO')
    else:
        print('YES')
```

https://tinyurl.com/101-prog-005-2

Fillimisht krijojmë një listë të re me të gjithë elementët e listës `A` që janë më të mëdhenj se `1`, dhe pastaj gjejmë gjatësinë e kësaj liste me anë të funksionit `len()`.

Zgjidhja 3

```python
T = int(input())
for t in range(T):
    n, k = map(int, input().split())
    A = list(map(int, input().split()))
    nr = len([ a for a in A if a > 1 ])
    print('NO' if nr > k else 'YES')
```

https://tinyurl.com/101-prog-005-3

Zgjidhja 4

```python
for _ in range(int(input())):
    n, k = map(int, input().split())
    A = list(map(int, input().split()))
    print('NO' if k < n - A.count(1) else 'YES')
```

https://tinyurl.com/101-prog-005-4

Detyra

Albani mendon se numri 3 është me fat (nuk thonë kot 'e treta e vërteta'). Kështu që kur shikon një varg me numra ai numëron sa tresha ka në të. Bëni një program që e ndihmon Albanin të gjejë sa tresha ka në një varg me numra të dhënë. Ky program duhet të shfaqë numrin e treshave, por nëse janë 3 tresha duhet të shkruajë `SUPERFAT`.

Shembull

```
$ cat input.txt
3
5
2 1 3 5 3
2
7 11
4
3 9 3 3

$ python3 prog.py < input.txt
2
0
SUPERFAT
```

Problemi 006

Kërkesa

Tre shokë, le ti quajmë A, B dhe C, bëjnë pohimet e mëposhtme:

- A: Unë kam x lekë më shumë ose më pak se B
- B: Unë kam y lekë më shumë ose më pak se C
- C: Unë kam z lekë më shumë ose më pak se A

Bëni një program që merr numrat x, y, z dhe gjen nëse A, B dhe C mund të kenë sasi lekësh të tilla që të gjitha pohimet të jenë të vërteta.

Referenca: https://www.codechef.com/problems/THREEFR

Shembull

```
$ cat input.txt
2
1 2 1
1 1 1

$ python3 prog.py < input.txt
yes
no
```

Zgjidhja 1

```python
T = int(input())
for t in range(T):
    x, y, z = map(int, input().split())
    if x >= y and x >= z:
        print('YES' if x == y + z else 'NO')
    elif y >= x and y >= z:
        print('YES' if y == x + z else 'NO')
    else:
        print('YES' if z == x + y else 'NO')
```

https://tinyurl.com/101-prog-006-1

Sqarime

Që pohimet të jenë të vërteta, duhet që më i madhi prej numrave të jetë i barabartë me shumën e dy të tjerëve.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    x, y, z = map(int, input().split())
    l = [x, y, z]
    l.sort(reverse=True)
    print('YES' if l[0] == l[1] + l[2] else 'NO')
```

https://tinyurl.com/101-prog-006-2

Sqarime

Për të gjetur më të madhin, ne mund të krijojmë një listë me ta dhe ti rendisim në rendin zbritës.

Zgjidhja 3

```
T = int(input())
for t in range(T):
    x, y, z = map(int, input().split())
    if y > x: x,y=y,x
    if z > x: x,z=z,x
    print('YES' if x==y+z else 'NO')
```

https://tinyurl.com/101-prog-006-3

Sqarime

Mund tu ndërrojmë vendet vlerave, në mënyrë të tillë që te `x`-i të kemi gjithmonë vlerën më të madhe.

Zgjidhja 4

```python
for _ in range(int(input())):
    x, y, z = map(int, input().split())
    print('YES' if 2*max(x,y,z)==x+y+z else 'NO')
```

https://tinyurl.com/101-prog-006-4

Sqarime

Disa zgjidhje janë më të thjeshta dhe më të shkurtra se të tjerat.

Detyra

Në një restorant ka filluar punë një kamarier i ri. Meqënëse nuk është dhe aq i mirë në matematikë, ndonjëherë e kthen reston gabim. Shefi i jep atij një problem të thjeshtë: Sa bëjnë $a - b$? Përgjigja e tij është gabim. Dhe për çudi ka vetëm një shifër gabim. E merrni dot me mënd?

A mund të shkruani një program që bën të njëjtin gabim? Programi merr dy numra `a` dhe `b` dhe duhet të japë një diference të gabuar `a-b`. Përgjigja duhet të jetë një numër pozitiv që ka të njëjtin numër shifrash me diferencën e saktë, ku të gjitha shifrat janë njësoj me diferencën e saktë, me përjashtim të njërës që duhet të jetë e ndryshme. Zeroja nuk duhet të jetë si shifër e parë (nga e majta). Nëse ka përgjigje të ndryshme që i plotësojnë këto kushte, mjafton të jepni vetëm njërën.

Referenca: https://www.codechef.com/problems/CIELAB

Shembull

```
$ cat input.txt
3
5858 1234
239 230
4375 4375

$ python3 prog.py < input.txt
1624
8
1
```

Në rastin e parë, diferenca e saktë është `5858 - 1234 = 4624`, kështu që këto përgjigje do ishin të sakta: `2624`, `4324`, `4623`, `4604`, `4629`, kurse këto do ishin të gabuara: `0624`, `624`, `5858`, `4624`, `04624`.

Problemi 007

Kërkesa

Grada e çelikut përcaktohet në bazë to kushteve të mëposhtme:

- **i**. Fortësia është më e madhe se 50
- **ii**. Përmbajtja e karbonit është më pak se 0.7
- **iii**. Forca elastike është më e madhe se 5600

Grada është si më poshtë:

- **10** nëse plotësohen të treja kushtet
- **9** nëse plotësohen kushtet (i) dhe (ii)
- **8** nëse plotësohen kushtet (ii) dhe (iii)
- **7** nëse plotësohen kushtet (i) dhe (iii)
- **6** nëse plotësohet vetëm njëri nga kushtet
- **5** nëse asnjë nga kushtet nuk plotësohet

Bëni një program që merr vlerat e fortësisë, përmbajtjes së karbonit dhe forcës elastike, dhe nxjerr gradën e çelikut.

Referenca: https://www.codechef.com/problems/FLOW014

Shembull

```
$ cat input.txt
3
53 0.6 5602
45 0 4500
0 0 0

$ python3 prog.py < input.txt
10
6
6
```

Zgjidhja

```
T = int(input())
for t in range(T):
    hardness, carbon, tensile = map(float, input().split())
    k1 = hardness > 50
    k2 = carbon < 0.7
    k3 = tensile > 5600
    if k1 and k2 and k3:
        print(10)
    elif k1 and k2:
        print(9)
    elif k2 and k3:
        print(8)
    elif k1 and k3:
        print(7)
    elif k1 or k2 or k3:
        print(6)
    else:
        print(5)
```

https://tinyurl.com/101-prog-007

Detyra

Bëni një program që merr kodin e një anije (si një shkronjë) dhe shfaq llojin e saj (si emër të plotë). Kodet dhe llojet e anijeve janë si më poshtë:

Kodi	Lloji
B ose b	BattleShip
C ose c	Cruiser
D ose d	Destroyer
F ose f	Frigate

Referenca: https://www.codechef.com/problems/FLOW010

Shembull

```
$ cat input.txt
3
```

```
B
c
D

$ python3 prog.py < input.txt
BattleShip
Cruiser
Destroyer
```

Problemi 008

Kërkesa

Një olimpiadë programimi zhvillohet në disa faza. Nga faza e parë në të dytën kualifikohen `K` pjesëmarrësit që kanë marrë më shumë pikë. Por nqs janë disa pjesëmarrës që kanë numër të njëjtë pikësh me atë që është në vendin e K-të, edhe ata kualifikohen.

Bëni një program që merr numrin K dhe pikët e pjesëmarrësve, dhe nxjerr numrin e pjesëmarrësve që kualifikohen për në fazën tjetër.

Referenca: https://www.codechef.com/problems/QUALPREL

Shembull

```
$ cat input.txt
2
5 1
3 5 2 4 5
6 4
6 5 4 3 2 1

$ python3 prog.py < input.txt
2
4
```

Zgjidhja

```
T = int(input())
for t in range(T):
    n, k = map(int, input().split())
    piket = list(map(int, input().split()))
```

```
    piket.sort(reverse=True)
    while k < n and piket[k] == piket[k-1]:
        k += 1
    print(k)
```

https://tinyurl.com/101-prog-008

Sqarime

Së pari, pikët e konkurentëve duhen renditur në rendin zbritës (`piket.sort(reverse=True)`). Pastaj, `k` studentët e parë do kualifikohen pa diskutim, por gjithashtu edhe studentët pasardhës, nëse kanë të njëjtin numër pikësh me atë që është në vendin `k`.

Kushti `k < n` siguron që `k`-ja nuk do ta kapërcejë fundin e listës `piket`. Gjithashtu, gjuha Python (si shumë gjuhë të tjera) bën një vlerësim të shkurtuar të shprehjeve llogjike, që do të thotë se nëse kushti `A` i shprehjes `A and B` del `False`, atere nuk kalohet fare te vlerësimi i shprehjes `B`.

Në rastin tonë, nëse shpreja `piket[k] == piket[k-1]` do të vlerësohej kur `k==n`, do të nxirrte një gabim (sepse `piket[n]` ndodhet jashtë kufirit të listës). Por falë vlerësimit të shkurtuar kjo nuk ndodh.

Detyra

Një olimpiadë programimi është zhvilluar në një qytet në vitet `2010`, `2015`, `2016`, `2017`, `2019`. Bëni një program që merr disa vite dhe për secilin prej tyre nxjerr `HOSTED` nëse olimpiada është zhvilluar atë vit në qytet, ose `NOT HOSTED` nëse nuk është zhvilluar.

Referenca: https://www.codechef.com/problems/SNCKYEAR

Shembull

```
$ cat input.txt
2
2019
2018

$ python3 prog.py < input.txt
HOSTED
NOT HOSTED
```

Problemi 009

Kërkesa

Në një kompani shefi ka vënë dy roja që numërojnë sa herë hyjnë punonjësit në ndërtesë. Ndonjëherë rojat mund të dremisin pak, por për fat të mirë asnjëherë nuk dremisin në të njëjtën kohë, të paktën njëri qëndron zgjuar për të numëruar ata që hyjnë.

Tani shefi do të dijë sa herë ka hyrë një punonjës në ndërtesë. Ai pyeti rojat dhe i pari i tha numrin A, kurse i dyti i tha numrin B. Ndihmojeni shefin të llogarisë minimumin dhe maksimumin e mundshëm të herëve që ka hyrë punonjësi në ndërtesë.

Referenca: https://www.codechef.com/problems/REMISS

Shembull

```
$ cat input.txt
1
19 17

$ python3 prog.py < input.txt
19 36
```

Zgjidhja

```
T = int(input())
for t in range(T):
    a, b = map(int, input().split())
    print(max(a,b), a+b)
```

https://tinyurl.com/101-prog-009

Sqarime

Minimumi i mundshëm është kur hyrjet që ka numëruar njëri i ka numëruar gjithashtu edhe tjetri. Kështu që minimumi është numri më i madh prej të dyve.

Maksimumi i mundshëm është kur hyrjet i ka numuruar ose njëri ose tjetri, por jo të dy njëkohësisht. Kështu që maksimumi është shuma e dy numrave.

Funksioni `max()` kthen më të madhin e numrave që i jepen.

Detyra

Një pikë shitje bën zbritje 10% nëse sasia e mallit të blerë është mbi 1000 copë. Bëni një program që llogarit vlerën e blerjes, duke ditur sasinë dhe çmimin.

Referenca: https://www.codechef.com/problems/FLOW009

Shembull

```
$ cat input.txt
3
100 120
10 20
1200 20

$ python3 prog.py < input.txt
12000.000000
200.000000
21600.000000
```

Problemi 010

Kërkesa

Kosta e ka qejf numrin **4** sepse ka ca veti shumë interesante. P.sh.:

- Katra është numri më i vogël i përbërë.
- Grupi më i vogël jo-ciklik ka 4 elementë.
- Katra është numri maksimal i shkallës së ekuacioneve që mund të zgjidhen në rrënjë.
- Është një teoremë që thotë se çdo hartë mund të ngjyroset vetëm me 4 ngjyra të ndryshme, në mënyrë që çdo dy vënde kufitare të mos kenë të njëjtën ngjyrë.
- Një teoremë e Lagrange-it thotë se çdo numër natyror mund të shprehet si shumë e të shumtën 4 katrorë numrash (numra në fuqi të dytë).
- E plot të tjera.

I mahnitur nga fuqitë e këtij numri, Kosta e ka zakon që numëron sa katra janë në çdo numër. Për ta ndihmuar Kostën, bëni një program që numëron sa shifra **4** ka në një numër të dhënë.

Referenca: https://www.codechef.com/problems/LUCKFOUR

Shembull

```
$ cat input.txt
5
447474
228
6664
40
81

$ python3 prog.py < input.txt
4
0
1
1
0
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    number = list(input())
    k = 0
    for s in number:
        if s == '4': k += 1
    print(k)
```

https://tinyurl.com/101-prog-010-1

Sqarime

Nqs funksionit `list()` i japim si parametër një varg shkronjash, na kthen një listë me të gjitha shkronjat përbërëse të këtij vargu.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    number = list(input())
    print(number.count('4'))
```

https://tinyurl.com/101-prog-010-2

Zgjidhja 3

```
T = int(input())
for t in range(T):
    number = input()
    k = 0
    for i in range(len(number)):
        if number[i] == '4': k += 1
    print(k)
```

https://tinyurl.com/101-prog-010-3

Zgjidhja 4

```
for _ in range(int(input())):
    print(input().count('4'))
```

https://tinyurl.com/101-prog-010-4

Detyra

Jepet një numër natyror. Gjeni shumën e shifrës së parë dhe të fundit të tij.

Referenca: https://www.codechef.com/problems/FLOW004

Shembull

```
$ cat input.txt
3
1234
124894
242323

$ python3 prog.py < input.txt
5
5
5
```

3 Problemat 011-020

Problemi 011

Kërkesa

Bëni një program që merr kodin e një anije (si një shkronjë) dhe shfaq llojin e saj (si emër të plotë). Kodet dhe llojet e anijeve janë si më poshtë:

Kodi	Lloji
B ose b	BattleShip
C ose c	Cruiser
D ose d	Destroyer
F ose f	Frigate

Referenca: https://www.codechef.com/problems/FLOW010

Shembull

```
$ cat input.txt
3
B
c
D

$ python3 prog.py < input.txt
BattleShip
Cruiser
Destroyer
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
```

```
    k = input()
    if k == 'b' or k == 'B':
        print('BattleShip')
    elif k == 'c' or k == 'C':
        print('Cruiser')
    elif k == 'd' or k == 'D':
        print('Destroyer')
    else:
        print('Frigate')
```

Zgjidhja 2

```
ships = {
    'b': 'BattleShip',
    'c': 'Cruiser',
    'd': 'Destroyer',
    'f': 'Frigate'
}
for _ in range(int(input())):
    print(ships[input().lower()])
```

Sqarime

Funksioni `lower()` që aplikohet te një varg, e kthen këtë varg në shkronja të vogla. Kurse `ships` është një strukturë që quhet *fjalor* (*dictionary*). Provoni këto komanda në promptin e Python:

```
$ python3
>>> fruits = { 'a': 'Apple', 'b': 'Banana' }
>>> fruits['b']
'Banana'
>>> fruits.get('a')
'Apple'
>>> fruits['o'] = 'Orange'
>>> fruits.get('o')
'Orange'
>>> fruits['x']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'x'
```

```
>>> fruits.get('x')
>>> fruits.get('x', False)
False
>>>
```

Detyra

Kur pohojmë zakonisht e tundim kokën nga lart poshtë. Le ta shënojmë me `Y`, për "yes". Kur mohojmë e tundim kokën nga e majta në të djathtë, rreth boshtit vertikal. Le ta shënojmë me `N` për "no".

Por indianët zakonisht përdorin një shenjë tjetër për të pohuar, duke e tundur kokën nga e majta në të djathtë rreth boshtit para-mbrapa. Le ta shënojmë këtë me `I`.

Një djalë vëzhgoi disa persona në një stacion treni dhe mbajti shënim shënjat që bënin me kokë (duke përdorur shkronjat `Y`, `N` dhe `I`). Bëni një program që shfaq nëse personi në fjalë është indian, jo indian, ose nuk mund ta themi me siguri.

Referenca: https://www.codechef.com/problems/HEADBOB

Shembull

```
$ cat input.txt
3
5
NNNYY
6
NNINNI
4
NNNN

$ python3 prog.py < input.txt
NOT INDIAN
INDIAN
NOT SURE
```

Problemi 012

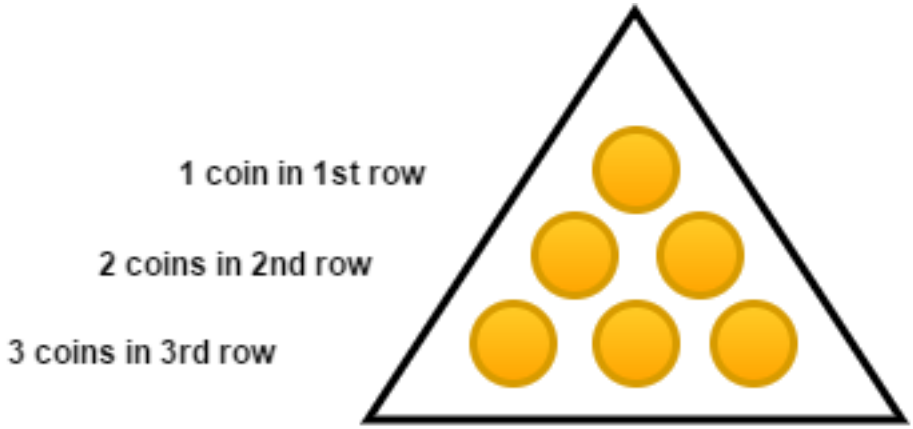
Kërkesa

Nqs kemi N monedha dhe i vendosim në formë trekëndëshi, në mënyrë të tillë që:

- në rreshtin e parë vendosim 1 monedhë

- në rreshtin e dytë vendosim 2 monedha
- në rreshtin e tretë vendosim 3 monedha
- e kështu me rradhë (siç tregohet në figurë)

Sa është lartësia e trekëndëshit më të madh që mund të formohet?



A Traingle with height = 3 requires 6 coins

Referenca: https://www.codechef.com/problems/TRICOIN

Shembull

```
$ cat input.txt
3
3
5
7

$ python3 prog.py < input.txt
2
2
3
```

- Testi 1: Nuk mund të formojmë një trekëndësh me lartësi >2, sepse kjo kërkon të paktën 6 monedha.
- Testi 2: Nuk mund të formojmë një trekëndësh me lartësi >2, sepse kjo kërkon të paktën 6 monedha.
- Testi 3: Nuk mund të formojmë një trekëndësh me lartësi >3, sepse kjo kërkon të paktën 10 monedha.

Zgjidhja 1

```
T = int(input())
for t in range(T):
     n = int(input())
     l = 0    # lartesia
     s = 0    # sasia e monedhave
     while True:
         s += l + 1
         if s > n:
             break
         else:
             l += 1
     print(l)
```

Sqarime

Fillimisht shënojmë me `0` **l**artësinë e trekëndëshit dhe **s**asinë e monedhave që duhet për ta ndërtuar. Pastaj fillojmë ta rrisim lartësinë me nga 1, duke llogaritur edhe sasinë e monedhave që na duhen për ta ndërtuar. Këtë punë e bëjmë deri sa sasia e monedhave mos ta kalojë numrin `n` që na është dhënë.

Zgjidhja 2

```
for _ in range(int(input())):
     n = int(input())
     l = s = 0
     while s <= n:
         l += 1
         s += l
     print(l - 1)
```

Fillojmë ta rrisim lartësinë me nga 1, duke llogaritur edhe sasinë e monedhave që na duhen. Këtë punë e bëjmë deri sa sasia e duhur e monedhave ta kalojë numrin `n` që na është dhënë. Atere lartësia e mundshme e trekëndëshit është vlera e `l`-së që sapo kaluam (ku sasia e dhënë e monedhave ishte e mjaftueshme për ta ndërtuar).

Detyra

Shkruani një program që llogarit shumën e shifrave të një numri të dhënë.

Referenca: https://www.codechef.com/problems/FLOW006

Shembull

```
$ cat input.txt
3
12345
31203
2123

$ python3 prog.py < input.txt
15
9
8
```

Problemi 013

Kërkesa

Ju e dini që $n! = 1 * 2 * 3 * ... * n$. Le të përkufizojmë funksionin $z(n)$ si numrin e zerove që ka në fund $n!$. Vini re që ky është një funksion jo-zbritës, d.m.th. për $n_1 < n_2$ kemi $z(n_1) \leq z(n_2)$. Kjo ndodh sepse kur e shumëzojmë një numër që ka zero në fund me një tjetër, numri i zerove që janë në fund vetëm mund të rritet.

Bëni një program që gjen $z(n)$ (d.m.th. sa zero ka në fund $n!$).

Referenca: https://www.codechef.com/problems/FCTRL

Shembull

```
$ cat input.txt
6
3
60
100
1024
23456
8735373

$ python3 prog.py < input.txt
0
14
24
```

```
253
5861
2183837
```

Zgjidhja 1

```python
T = int(input())
for t in range(T):
    n = int(input())
    f = 1  # faktoriali
    for i in range(1, n+1):
        f *= i
    z = 0  # numri i zerove të faktorialit
    while f % 10 == 0:
        z += 1
        f //= 10
    print(z)
```

Sqarime

Fillimisht gjejmë faktorialin e numrit të dhënë, pastaj gjejmë sa zero ka në fund.

Kjo është zgjidhje e thjeshtë për tu ndërtuar por jo edhe aq efiçente (e shpejtë). Është e mundur të gjendet numri i zerove pa e gjetur më parë vetë faktorialin.

Zgjidhja 2

```python
for _ in range(int(input())):
    n = int(input())
    z = 0
    p = 5    # fuqite e 5-es
    while p < n:
        z += n // p
        p *= 5
    print(z)
```

Sqarime

Nqs faktorialin e shprehim si prodhim numrash të thjeshtë, atere është e qartë se numri i zerove në fund të tij është i barabartë me fuqinë e numrit 5, sepse çdo zero në fund të

numrit vjen si rezultat i prodhimit `5*2` (dysha në prodhimin faktorial kemi më shumë se pesa, sepse çdo numër çift kontribon në prodhim të paktën me një dysh).

Zgjidhja e problemit bazohet në faktin që çdo shumëfish i 5-ës kontribon **1** pesë në prodhim, çdo shumëfish i 25-ës kontribon **2** pesa, çdo shumëfish i 5^3 kontribon **3** pesa, e kështu me rradhë.

Për një sqarim më të detajuar të zgjidhjes shikoni edhe këtë diskutim: https://discuss.codechef.com/questions/58730/fctrl-editorial

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    z = 0
    while n > 0:
        n //= 5
        z += n
    print(z)
```

Sqarime

Ideja është e njëjtë me zgjidhjen e dytë, por algoritmi pak më ndryshe.

Gjejmë sa shumëfisha të 5-ës kemi deri te n-ja, dhe këtë ia shtojmë z-së. Pastaj gjejmë sa shumëfisha të 25-ës dhe ia shtojmë z-së, e kështu me rradhë.

Detyra

Në një varg shkronjash të dhënë, përcaktoni nëse njëra prej shkronjave përsëritet po aq sa përsëriten të gjitha shkronjat e tjera sëbashku.

Referenca: https://www.codechef.com/problems/LCH15JAB

Shembull

```
$ cat input.txt
4
acab
zzqzqq
abc
kklkwwww
```

```
$ python3 prog.py < input.txt
YES
YES
NO
YES
```

Problemi 014

Kërkesa

Në një varg me numra natyrorë $a_1, a_2, ..., a_n$ gjeni diferencën më të vogël të mundshme midis 2 prej tyre. Dmth gjeni vlerën minimale të $|a_i - a_j|$ për çdo $1 \leq i < j \leq n$.

Referenca: https://www.codechef.com/problems/HORSES

Shembull

```
$ cat input.txt
1
5
4 9 1 32 13

$ python3 prog.py < input.txt
3
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n = int(input())
    l = list(map(int, input().split()))
    dmin = abs(l[0] - l[1])
    i = 0
    while i <= n-2:
        j = i + 1
        while j <= n-1:
            d = abs(l[i] - l[j])
            if d < dmin:
                dmin = d
            j += 1
```

```
        i += 1
    print(dmin)
```

Sqarime

Marrim të gjitha kombinimet e mundshme të `i` dhe `j`, gjejmë diferencat midis tyre, dhe ndërkohë gjejmë edhe më të voglën e diferencave.

P.sh. për numrin a_1 bëjmë diferencën me $a_2, a_3, ..., a_n$, për numrin a_2 bëjmë diferencën me $a_3, ..., a_n$, e kështu me rradhë.

Kjo zgjidhje nuk është dhe aq efiçente sepse për n numra duhet të bëjmë përafërsisht n^2 veprime (zbritje, krahasime, etj.)

Zgjidhja më poshtë është më e shpejtë.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    n = int(input())
    l = list(map(int, input().split()))
    l.sort(reverse=True)
    dmin = l[0] - l[1]
    i = 0
    while i <= n-2:
        d = l[i] - l[i+1]
        if d < dmin:
            dmin = d
        i += 1
    print(dmin)
```

Sqarime

Fillimisht i rendisim numrat në rendin zbritës. Pastaj, në vend që të bëjmë diferencën e numrit a_i me të gjithë numrat pasardhës, mjafton që të bëjmë diferencën e tij vetëm me numrin a_{i+1}, dhe dihet që diferenca me numrat e tjerë pasardhës nuk është më e vogël se kjo (sepse janë më të vegjël se a_{i+1}).

Detyra

Kemi një varg me numra ku të gjithë numrat përsëriten një numër çift herësh, me përjashtim të njërit i cili përsëritet një numër tek herësh. Të gjendet ky numër.

Referenca: https://www.codechef.com/problems/MISSP

Shembull

```
$ cat input.txt
2
3
1
2
1
5
1
1
2
2
3

$ python3 prog.py < input.txt
2
3
```

Janë 2 raste testimi, i pari ka 3 numra (njëri nën tjetrin), dhe i dyti ka 5 numra.

Problemi 015

Kërkesa

Gjeni nëse një numër i dhënë është i thjeshtë ose jo (pra nuk ka plotpjesëtues të tjerë përveç 1-shit dhe vetes).

Referenca: https://www.codechef.com/problems/PRB01

Shembull

```
$ cat input.txt
5
23
```

```
13
20
1000
99991

$ python3 prog.py < input.txt
yes
yes
no
no
yes
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    N = int(input())
    i = 2
    while i < N:
        if N % i == 0:
            print("no")
            break
        i += 1
    if i == N:
        print("yes")
```

Sqarime

Kontrollojmë me rradhë numrat nga `2` në `N-1`, dhe nëse gjendet ndonjë që e plotpjesëton (mbetja e pjesëtimit është 0), printojmë "no" dhe ndërpresim ciklin (me `break`). Nëse kemi shkuar deri në fund të kontrollit pa e ndërprerë ciklin, i bie që është numër i thjeshtë, kështu që printojmë "yes".

Zgjidhja 2

```
T = int(input())
for t in range(T):
    N = int(input())
    for i in range(2, N):
        if N % i == 0:
```

```
            print("no")
            break
    else:
        print("yes")
```

Sqarime

E njëjta llogjikë si më sipër, por e realizuar me një cikël `for` (në vend të `while`). Vini re që `else`-i i `for`-it zbatohet vetëm kur cikli shkon deri në fund (pa u ndërprerë nga `break`).

Zgjidhja 3

```
T = int(input())
for t in range(T):
    N = int(input())
    i = 2
    while i*i < N:
        if N % i == 0:
            print("no")
            break
        i += 1
    if i*i >= N:
        print("yes")
```

Sqarime

Mjafton që të kontrollojmë numrat nga `2` deri te rrënja katrore e `N`.

Zgjidhja 4

```
def is_prime(n):
    i = 2
    while i*i < n:
        if n % i == 0:
            return "no"
        i += 1
    return "yes"
```

```
for _ in range(int(input())):
    print(is_prime(int(input())))
```

Sqarime

Përdorimi i një funksioni në këtë rast e thjeshton llogjikën e programit.

Detyra

Bëni një program që gjen rrënjën katrore natyrore (të përafërt) të një numri natyror të dhënë.

Referenca: https://www.codechef.com/problems/FSQRT

Shembull

```
$ cat input.txt
3
10
5
10000

$ python3 prog.py < input.txt
3
2
100
```

Problemi 016

Kërkesa

Shkruani një program që llogarit shumën e shifrave të një numri të dhënë.

Referenca: https://www.codechef.com/problems/FLOW006

Shembull

```
$ cat input.txt
3
12345
```

```
31203
2123

$ python3 prog.py < input.txt
15
9
8
```

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    s = 0
    while n > 0:
        s += n % 10
        n = n // 10
    print(s)
```

Zgjidhja 2

```
for _ in range(int(input())):
    n = input()
    s = 0
    for i in range(len(n)):
        s += int(n[i])
    print(s)
```

Zgjidhja 3

```
for _ in range(int(input())):
    n = input()
    s = 0
    for d in n: s += int(d)
    print(s)
```

Detyra

Bëni një program që e shkruan mbrapsht një numër të dhënë natyror.

Referenca: https://www.codechef.com/problems/FLOW007

Shembull

```
$ cat input.txt
4
12345
31203
2123
2300

$ python3 prog.py < input.txt
54321
30213
3212
32
```

Problemi 017

Kërkesa

Në një varg shkronjash të dhënë, përcaktoni nëse njëra prej shkronjave përsëritet po aq sa përsëriten të gjitha shkronjat e tjera sëbashku.

Referenca: https://www.codechef.com/problems/LCH15JAB

Shembull

```
$ cat input.txt
4
acab
zzqzqq
abc
kklkwwww

$ python3 prog.py < input.txt
YES
YES
NO
YES
```

Zgjidhja 1

```
for _ in range(int(input())):
    char_list = list(input())

    # find the maximum frequence
    char_list.sort()
    max_freq = 0
    freq = 0
    prev_char = None
    for ch in char_list:
        if ch == prev_char:
            freq += 1
        else:
            if freq > max_freq:
                max_freq = freq
            prev_char = ch
            freq = 1
    if freq > max_freq:
        max_freq = freq

    print('YES' if 2*max_freq==len(char_list) else 'NO')
```

Sqarime

Fillimisht gjejmë numrin e përsëritjeve të shkronjës që ka numrin më të madh të përsëritjeve, dhe pastaj kontrollojmë nëse ky numër është sa gjysma e numrit të të gjitha shkronjave.

Për të gjetur numrin më të madh të përsëritjeve, fillimisht krijojmë një listë që ka të gjitha shkronjat, dhe pastaj e rendisim në mënyrë që shkronjat e njëjta të jenë në rradhë njëra pas tjetrës, dhe kështu të mund ti numërojmë kollaj. Pastaj fillojmë të numërojmë secilën shkronjë dhe ruajmë atë numër përsëritjesh që ka vlerën më të madhe.

Zgjidhja 2

```
for _ in range(int(input())):
    chars = input()
    frequences = {}    # dictionary of char frequences
    for ch in chars:
```

```
        frequences[ch] = frequences.get(ch, 0) + 1:
    print('YES' if 2*max(frequences.values())==len(chars) else 'NO')
```

Sqarime

Krijojmë një *fjalor* (*dictionary*) që për çdo shkronjë të vargut ruan përsëritjet e kësaj shkronje. Pastaj është kollaj të marrim një listë të të gjitha vlerave të këtij fjalori dhe të gjejmë vlerën më të madhe prej tyre.

Zgjidhja 3

```
for _ in range(int(input())):
    chars = input()
    max_freq = max([chars.count(c) for c in set(chars)])
    print('YES' if 2*max_freq==len(chars) else 'NO')
```

Sqarime

Funksioni `set(chars)` kthen një *bashkësi* të shkronjave të veçanta (pa përsëritje) që ndodhen në vargun `chars`. Për çdo shkronjë `c` të kësaj bashkësie, `chars.count(c)` numëron përsëritjet e kësaj shkronje në vargun `chars`. Kurse kllapat katrore `[ . . . ]` bëjnë që gjithë këto vlera të krjojnë një listë të përsëritjeve të shronjave të veçanta. Funksioni `max()` gjen vlerën më të madhe të këtyre përsëritjeve.

Detyra

Kemi një varg me numra ku të gjithë numrat përsëriten një numër çift herësh, me përjashtim të njërit i cili përsëritet një numër tek herësh. Të gjendet ky numër.

Referenca: https://www.codechef.com/problems/MISSP

Shembull

```
$ cat input.txt
2
3
1
2
1
5
```

```
1
1
2
2
3

$ python3 prog.py < input.txt
2
3
```

Janë 2 raste testimi, i pari ka 3 numra (njëri nën tjetrin), dhe i dyti ka 5 numra.

Problemi 018

Kërkesa

Një shkallë që të çon në katin e dytë ka n hapa, dhe lartësia e secilit hap nga toka është h_i. Ada do të ngjitet në katin e dytë, por ajo mund të ngjitet nga lartësia h_i në lartësinë h_f vetëm nëse $h_f - h_i \leq k$. Për ta ndihmuar Adën duhet të ndërtojmë disa hapa të ndërmjetëm midis hapave ekzistues (ose para hapit të parë).

Bëni një program që gjen numrin më të vogël të hapave ndihmës që duhen ndërtuar, që Ada të mund të ngjitet në katin e dytë.

Referenca: https://www.codechef.com/problems/ADASTAIR

Shembull

```
$ cat input.txt
1
4 3
2 4 8 16

$ python3 prog.py < input.txt
3
```

Jepen numrat n dhe k, dhe pastaj jepet lartësia e secilit prej hapave.

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n, k = list(map(int, input().split()))
    h = list(map(int, input().split()))
    hapat = 0
    for i in range(n):
        d = h[0] if i==0 else h[i] - h[i-1]
        while d > k:
            d -= k
            hapat += 1
    print(hapat)
```

Sqarime

Gjejmë diferencën midis dy hapave, dhe nqs është më e madhe se `k`, ndërtojmë një hap ndihmës me lartësi `k` (dhe kështu diferenca zvogëlohet me `k`).

Gjatë gjetjes së diferencës hapin e parë e kemi trajtuar në mënyrë të veçantë sepse nuk kemi hap tjetër para tij.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    n, k = list(map(int, input().split()))
    h = list(map(int, input().split()))
    hapat = 0
    h = [0] + h
    for i in range(1, n+1):
        d = h[i] - h[i-1]
        while d > k:
            d -= k
            hapat += 1
    print(hapat)
```

Sqarime

Kemi shtuar një `0` në krye të listës së hapave, dhe `i`-në e marrim nga `1` deri në `n`, në mënyrë që të shmangim `if`-n për gjetjen e diferencës.

Zgjidhja 3

```
for _ in range(int(input())):
    n, k = list(map(int, input().split()))
    h = [0] + list(map(int, input().split()))
    hapat = 0
    for i in range(1, n+1):
        hapat += (h[i] - h[i-1]) // k
    print(hapat)
```

Sqarime

I rrisim `hapat` me aq herë sa hyn `k`-ja te diferenca midis hapave ekzistues.

Detyra

Një sistem monetar ka prerje (ose monedha) me vlerë: 1, 2, 5, 10, 50, 100. Bëni një program që gjen numrin më të vogël të monedhave që e kanë shumën sa një vlerë e dhënë N.

Referenca: https://www.codechef.com/problems/FLOW005

Shembull

```
$ cat input.txt
3
1200
500
242

$ python3 prog.py < input.txt
12
5
7
```

Problemi 019

Kërkesa

Bëni një program që e shkruan mbrapsht një numër të dhënë natyror.

Referenca: https://www.codechef.com/problems/FLOW007

Shembull

```
$ cat input.txt
4
12345
31203
2123
2300

$ python3 prog.py < input.txt
54321
30213
3212
32
```

Zgjidhja 1

```python
T = int(input())
for t in range(T):
    n = int(input())
    l = []
    while n > 0:
        l.append(n % 10)
        n //= 10
    n1 = 0
    for d in l:
        n1 *= 10
        n1 += d
    print(n1)
```

Sqarime

Fillimisht nxjerrim të gjitha shifrat e numrit, duke marrë mbetjen e pjesëtimit me 10, dhe duke e vazhduar këtë punë me herësin e pjesëtimit me 10. Këto shifra i ruajmë në një listë.

Pastaj ndërtojmë numrin e ri duke shumëzuar me 10 dhe duke shtuar shifrat, por sipas rradhës së kundërt.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    l = list(map(int, list(input())))
    l.reverse()
    n1 = 0
    for d in l:
        n1 *= 10
        n1 += d
    print(n1)
```

Zgjidhja 3

```
for _ in range(int(input())):
    l = list(input())
    l.reverse()
    print(int(''.join(l)))
```

Sqarime

Funksioni `list(input())` krijon një listë me shkronjat e vargut të lexuar.

Funksioni `join(l)` që thirret te vargu `''`, i bashkon (prapangjit) këtij vargu të gjitha vargjet që ndodhen në listën `l`.

Para se ta printojmë duhet ta kthejmë në `int`, në mënyrë që të eliminojmë ndonjë zero fillestare që mund të ketë.

Detyra

Në një lojë të rregullt tenisi, lojtari që shërben ndërrohet çdo 2 pikë, pavarësisht se kush i shënon. Chef dhe Cook po e luajnë lojën pak më ndryshe. Ata vendosën që lojtari i shërbimit të ndërrohet çdo k pikë, pavarësisht se kush i shënon.

Në fillim të lojës, rradhën e shërbimit e ka gjithmonë Chef-i. Nqs dimë që deri tani Chef ka shënuar p_1 pikë dhe Cook ka shënuar p_2 pikë, kush e ka tani rradhën e shërbimit?

Referenca: https://www.codechef.com/problems/CHSERVE

Shembull

```
$ cat input.txt
3
1 3 2
0 3 2
34 55 2

$ python3 prog.py < input.txt
CHEF
COOK
CHEF
```

Tre numrat që jepen janë p_1, p_2 dhe k.

Problemi 020

Kërkesa

Kemi një varg me numra ku të gjithë numrat përsëriten një numër çift herësh, me përjashtim të njërit i cili përsëritet një numër tek herësh. Të gjendet ky numër.

Referenca: https://www.codechef.com/problems/MISSP

Shembull

```
$ cat input.txt
2
3
1
2
1
5
1
1
2
2
3

$ python3 prog.py < input.txt
2
3
```

Janë 2 raste testimi, i pari ka 3 numra (njëri nën tjetrin), dhe i dyti ka 5 numra.

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n = int(input())
    l = []
    for i in range(n):
        l.append(int(input()))
    l.sort()
    l.append(None)
    i = 0
    while i < n:
        if l[i] != l[i+1]:
            break
        i += 2
    print(l[i])
```

Sqarime

Krijojmë një listë dhe i fusim në të të gjithë numrat. Pastaj listën e rendisim. Në këtë mënyrë, të gjithë numrat e barabartë i kemi përkrah njëri-tjetrit dhe mund ti kalojmë dy e nga dy. Aty ku dy numrat e rradhës nuk janë të barabartë, i bie që të jetë numri që përsëritet një numër tek herësh. Për të kapur edhe rastin kur ky numër ndodhet në fund të listës (së renditur), kemi shtuar në list edhe një element që është i ndryshëm nga gjithë numrat e tjerë: `None`.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    n = int(input())
    l = []
    for i in range(n):
        a = input()
        if a in l:
            l.remove(a)
        else:
```

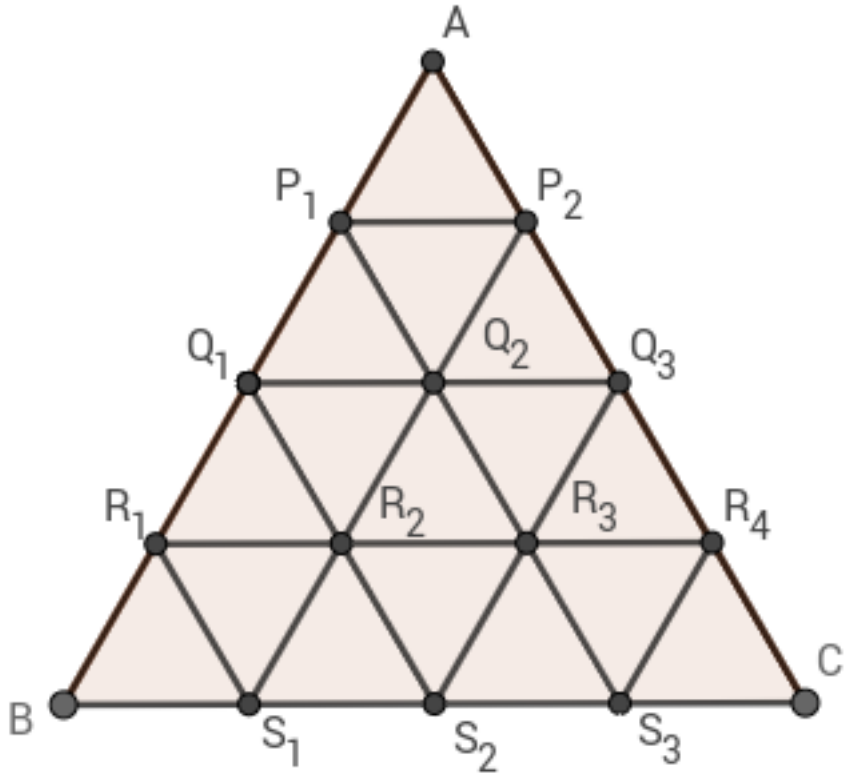
```
            l.append(a)
    print(l[0])
```

Sqarime

Krijojmë një listë bosh dhe çdo numër që lexojmë ose e shtojmë në listë nëse nuk është në të, ose e heqim nga lista nëse është në të. Në këtë mënyrë, në fund të këtij procesi do na ketë mbetur në listë vetëm ai numër që përsëritet një numër tek herësh.

Detyra

Në brinjët e një trekëndëshi barabrinjës vendosim disa pika të barazlarguara nga njëratjetra, dhe i bashkojmë me vija, si në figurë:



Le të quajmë të rregullt vetëm trekëndëshat që e kanë majën lart dhe bazën poshtë. Nqs ndarjet e brinjëve i quajmë 1 njësi, dhe trekëndëshi i madh e ka brinjën L njësi, sa trekëndësha të rregullt me brinjë K njësi (ku $1 \leq K \leq N$) ndodhen në figurë?

Referenca: https://www.codechef.com/problems/ZUBTRCNT

Shembull

```
$ cat input.txt
2
4 3
4 4

$ python3 prog.py < input.txt
Case 1: 3
Case 2: 1
```

Rezultati duhet të shfaqet në formën `Case i:` dhe pastaj përgjigja.

4 Problemat 021-030

Problemi 021

Kërkesa

Në një shumëkëndësh që ka N brinjë, këndet formojnë një progresion aritmetik, ku këndi më i vogël e ka madhësinë A gradë. Gjeni madhësinë e këndit K ($1 \leq K \leq N$) dhe shpreheni atë si një raport dy numrash natyrorë X dhe Y që janë reciprokisht të thjeshtë (pjesëtuesi më i madh i përbashkët është 1).

Referenca: https://www.codechef.com/problems/PCJ18C

Shembull

```
$ cat input.txt
1
3 30 2

$ python3 prog.py < input.txt
60 1
```

Tre numrat që jepen në input janë N, A dhe K. Në rastin tonë kemi një trekëndësh, ku këndi më i vogël është 30 gradë, dhe na kërkohet vlera e këndit të dytë.

Zgjidhja 1

```python
def pmp(a, b):
    '''
    Gjej pjesetuesin me te madh te perbashket te dy numrave te dhene.
    '''
    while a > 0:
        a, b = b % a, a
    return b

T = int(input())
```

```
for t in range(T):
    N, A, K = map(int, input().split())
    X = A*N*(N - 1) + 2*(K - 1)*(180*(N - 2) - N*A)
    Y = N*(N - 1)
    p = pmp(X, Y)
    print(X // p, Y // p)
```

Sqarime

Vërtetohet kollaj se shuma e këndeve të një N-këndëshi është $180 * (N - 2)$. Meqënëse këndet formojnë progresion aritmetik, këtë shumë mund ta shkruajmë edhe: $A + (A + x) + (A + 2 * x) + ... + (A + (N - 1) * x) = N * A + \frac{x*N*(N-1)}{2}$. Duke barazuar këto dy vlera gjejmë: $x = \frac{2*(180*(N-2)-N*A)}{N*(N-1)}$. Atere vlera e këndit të K-të është: $A + (K - 1) * x = \frac{A*N*(N-1)+2*(K-1)[180*(N-2)-N*A]}{N*(N-1)}$ Kështu që përgjigja është $X = A * N * (N - 1) + 2 * (K - 1)[180 * (N - 2) - N * A]$ dhe $Y = N * (N - 1)$.

Vetëm se këto vlera duhet ti thjeshtojmë duke i pjesëtuar me pjesëtuesin më të madh të përbashkët, të cilin e gjejmë me anë të funksionit `pmp()`. Ky funksion bazohet në faktin që `pmp(a, b)` është i njëjtë me `pmp(b - a, a)`, dhe si rrjedhim është i njëjtë edhe me `pmp(b % a, a)`.

Zgjidhja 2

```
import math

T = int(input())
for t in range(T):
    N, A, K = map(int, input().split())
    X = A*N*(N - 1) + 2*(K - 1)*(180*(N - 2) - N*A)
    Y = N*(N - 1)
    p = math.gcd(X, Y)
    print(X // p, Y // p)
```

Sqarime

Në këtë rast kemi përdorur funksionin e gatshëm `math.gcd()` (greatest common denominator), për të gjetur pjesëtuesin më të madh të përbashkët.

Detyra

Çufo ka një recetë me përbërësit që duhen për një gatim. Por sipas kësaj recete do të gatuhej shumë më tepër ushqim nga çi duhet Çufos. Kështu që ai do që ti pakësojë përbërësit, por duke ruajtur përpjesëtimin (raportin) e tyre, dhe duke qënë numra të plotë.

Reference: https://www.codechef.com/problems/RECIPE

Shembull

```
$ cat input.txt
3
2 4 4
3 2 3 4
4 3 15 9 6

$ python3 prog.py < input.txt
1 1
2 3 4
1 5 3 2
```

Për çdo rast testimi, numri i parë tregon numrin e përbërësve, dhe më pas vijnë sasitë përkatëse të përbërësve.

Problemi 022

Kërkesa

Në një lojë të rregullt tenisi, lojtari që shërben ndërrohet çdo 2 pikë, pavarësisht se kush i shënon. Chef dhe Cook po e luajnë lojën pak më ndryshe. Ata vendosën që lojtari i shërbimit të ndërrohet çdo k pikë, pavarësisht se kush i shënon.

Në fillim të lojës, rradhën e shërbimit e ka gjithmonë Chef-i. Nqs dimë që deri tani Chef ka shënuar p_1 pikë dhe Cook ka shënuar p_2 pikë, kush e ka tani rradhën e shërbimit?

Referenca: https://www.codechef.com/problems/CHSERVE

Shembull

```
$ cat input.txt
3
1 3 2
```

```
0 3 2
34 55 2

$ python3 prog.py < input.txt
CHEF
COOK
CHEF
```

Tre numrat që jepen janë p_1, p_2 dhe k.

Zgjidhja

```
T = int(input())
for t in range(T):
    p1, p2, k = map(int, input().split())
    piket = p1 + p2 + 1
    seti = piket // k
    if piket % k > 0:
        seti += 1
    if seti % 2 == 0:
        print('COOK')
    else:
        print('CHEF')
```

Sqarime

Nqs një grup prej k pikësh të njëpasnjëshme (ku shërben i njëjti lojtar) i quajmë një set, atere gjejmë në fillim se për cilin set po luhet kjo pikë. Nqs seti është tek, atere shërbimin e ka lojtari i parë, përndryshe e ka lojtari i dytë.

Detyra

Një qenush gjeti një arkë me N monedha. Ai nuk e hap dot vetë, por mund të lehë që tu tërheq vëmendjen njerëzve që ndodhen aty pranë. Ai e di që njerëzit do marrin secili një sasi të barabartë monedhash, dhe të tjerat do ti lënë në shesh, kështu që ai mund ti marrë.

Qenushi mund ta ndryshojë fortësinë e të lehurit, në mënyrë që ti tërheqë vëmendjen 1 personi, ose 2 personave, e kështu me rradhë, deri në K persona. Sa persona duhet të thërrasë, në mënyrë që në fund ti ngelen sa më shumë monedha?

Referenca: https://www.codechef.com/problems/GDOG

Shembull

```
$ cat input.txt
2
5 2
11 3

$ python3 prog.py < input.txt
1
2
```

Problemi 023

Kërkesa

Çufoja ka një recetë me përbërësit që duhen për një gatim. Por sipas kësaj recete do të gatuhej shumë më tepër ushqim nga ç'i duhet Çufos. Kështu që ai do që ti pakësojë përbërësit, por duke ruajtur përpjesëtimin (raportin) e tyre, dhe duke qënë numra të plotë.

Reference: https://www.codechef.com/problems/RECIPE

Shembull

```
$ cat input.txt
3
2 4 4
3 2 3 4
4 3 15 9 6

$ python3 prog.py < input.txt
1 1
2 3 4
1 5 3 2
```

Për çdo rast testimi, numri i parë tregon numrin e përbërësve, dhe më pas vijnë sasitë përkatëse të përbërësve.

Zgjidhja 1

```python
import math

for _ in range(int(input())):
    perberesit = list(map(int, input().split()))

    # hiq numrin e pare nga lista
    perberesit = perberesit[1:]

    # gjej pjesetuesim me te madh te perbashket
    pmp = perberesit[0]
    for p in perberesit:
        pmp = math.gcd(pmp, p)

    # shfaq pergjigjen
    for p in perberesit[:-1]:
        print(p // pmp, end=' ')
    print(perberesit[-1] // pmp)
```

Sqarime

Për të zvogëluar vlerat e përbërësve duke ruajtur përpjesëtimet dhe duke qënë numra të plotë, duhet të gjejmë pjesëtuesin më të madh të përbashkët të të gjithë numrave, dhe ti pjesëtojmë me të. PMP të të gjithë numrave mund ta gjejmë duke përdorur këtë veti: $pmp(a_1, a_2, a_3) = pmp(pmp(a_1, a_2), a_3)$

Kemi përdorur gjithashtu edhe disa mjete të gjuhës Python që përdoren për listat. Bëni këto prova:

```
$ python3
>>> p = [0, 1, 2, 3, 4]
>>> p
[0, 1, 2, 3, 4]
>>> p[1:4]
[1, 2, 3]
>>> p[1:]
[1, 2, 3, 4]
>>> p[:-1]
[0, 1, 2, 3]
>>> p[-1]
4
>>>
```

Zgjidhja 2

```
from math import gcd
from functools import reduce

for _ in range(int(input())):
    perberesit = list(map(int, input().split()))

    # hiq numrin e pare nga lista
    perberesit = perberesit[1:]

    # gjej pjesetuesim me te madh te perbashket
    pmp = reduce(gcd, perberesit)

    # shfaq pergjigjen
    for p in perberesit[:-1]:
        print(p // pmp, end=' ')
    print(perberesit[-1] // pmp)
```

Funksioni `reduce()` është i ngjashëm me funksionin `map()` në atë që zbaton një funksion të dhënë te vlerat e një liste të dhënë, por ndërsa funksioni `map()` kthen një listë, funksioni `reduce()` kthen një vlerë të vetme. Shiko edhe: http://book.pythontips.com/en/latest/map_filter.html#reduce

Detyra

Një fermer do të shesë tokën e tij, e cila është në formë drejtkëndëshi me përmasa M dhe N. Meqënëse copat e tokës në formë katrore kanë çmim më të lartë, ai do që ta ndajë tokën e tij në copa katrore të barabarta. Sa është numri më i vogël i katrorëve të barabartë që mund të krijohen në tokën e tij?

Referenca: https://www.codechef.com/problems/RECTSQ

Shembull

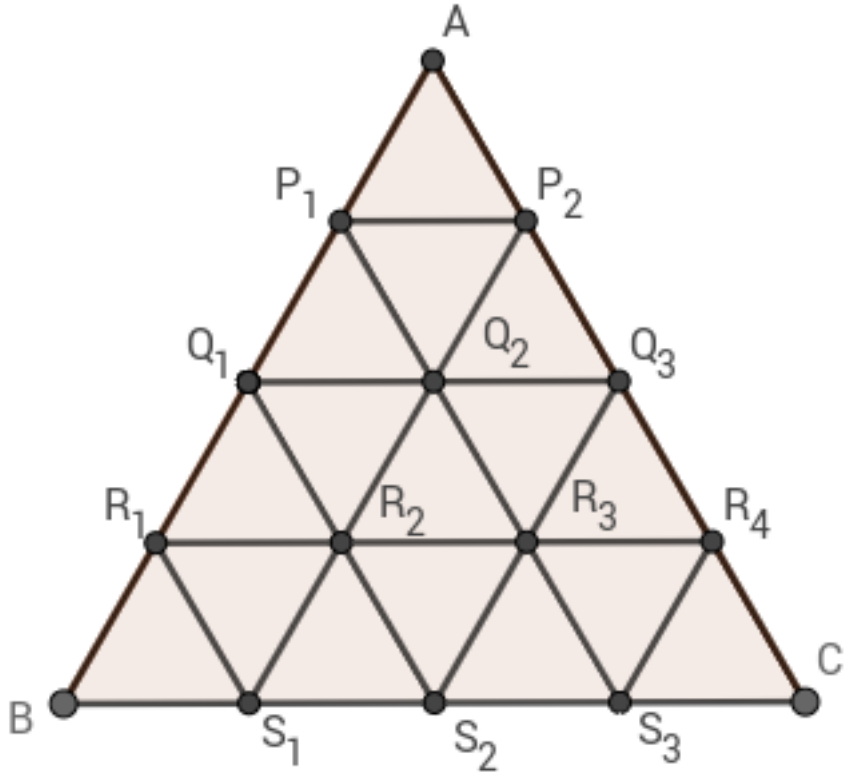
```
$ cat input.txt
2
10 15
4 6

$ python3 prog.py < input.txt
6
6
```

Problemi 024

Kërkesa

Në brinjët e një trekëndëshi barabrinjës vendosim disa pika të barazlarguara nga njëratjetra, dhe i bashkojmë me vija, si në figurë:



Le të quajmë të rregullt vetëm trekëndëshat që e kanë majën lart dhe bazën poshtë. Nqs ndarjet e brinjëve i quajmë 1 njësi, dhe trekëndëshi i madh e ka brinjën L njësi, sa trekëndësha të rregullt me brinjë K njësi (ku $1 \leq K \leq N$) ndodhen në figurë?

Referenca: https://www.codechef.com/problems/ZUBTRCNT

Shembull

```
$ cat input.txt
2
4 3
4 4
```

```
$ python3 prog.py < input.txt
Case 1: 3
Case 2: 1
```

Rezultati duhet të shfaqet në formën `Case i:` dhe pastaj përgjigja.

Zgjidhja 1

```python
T = int(input())
for t in range(T):
    l, k = map(int, input().split())
    nr = 1
    i = 1
    while k < l:
        i += 1
        nr += i
        k += 1
    print('Case {}: {}'.format(t+1, nr))
```

Sqarime

Sado që të jetë K-ja, kemi të paktën një trekëndësh të rregullt me brinjë K njësi (i cili e ka kulmin në pikën A). Kështu që fillimisht shënojmë `nr = 1`. Pastaj, duke e ulur kulmin e trekëndëshit një nivel më poshtë, do kemi 2 trekëndësha të tjerë (me kulme në pikat P_1 dhe P_2). Në nivelin e tretë do kemi 3 trekëndësha të tjerë (me kulme në Q_1, Q_2 dhe Q_3), e kështu me rradhë. Gjatësia K e brinjës së trekëndëshit të rregullt na kufizon sa nivele mund të ulemi.

Zgjidhja 2

```python
T = int(input())
for t in range(T):
    l, k = map(int, input().split())
    nr = 1
    for i in range(l - k):
        nr += i + 2
    print('Case {}: {}'.format(t+1, nr))
```

Detyra

Mamaja i ka blerë Cufit dy shporta me fruta, njëra ka N mollë dhe tjera ka M portokalle. Cufit i pëlqejnë edhe mollët edhe portokallet dhe do që ti ketë në sasi të barabartë. Çdo kokërr mollë ose portokalle kushton 1 monedhë, dhe Cufi ka K monedha në xhep. Me këto monedha ai mund të blejë mollë ose portokalle dhe të përpiqet ta bëjë ndryshimin midis tyre sa më të vogël.

Bëni një program që të gjejë ndryshimin më të vogël të mundshëm që mund të arrijë Cufi midis mollëve dhe portokalleve.

Referenca: https://www.codechef.com/problems/FRUITS

Shembull

```
$ cat input.txt
3
3 4 1
5 2 1
3 4 3

$ python3 prog.py < input.txt
0
2
0
```

Jepen numrat N, M dhe K. Në rastin e parë Cufi mund të blejë një mollë dhe ti bëjë të barabarta. Në rastin e dytë mund të blejë vetëm një portokalle dhe diferenca bëhet 2. Në rasting e tretë mund të blejë 2 mollë dhe një portokalle, duke e bërë diferencën 0.

Problemi 025

Kërkesa

Një sistem monetar ka prerje (ose monedha) me vlerë: 1, 2, 5, 10, 50, 100. Bëni një program që gjen numrin më të vogël të monedhave që e kanë shumën sa një vlerë e dhënë N.

Referenca: https://www.codechef.com/problems/FLOW005

Shembull

```
$ cat input.txt
```

```
3
1200
500
242

$ python3 prog.py < input.txt
12
5
7
```

Zgjidhja

```
prerjet = [100, 50, 10, 5, 2, 1]

for _ in range(int(input())):
    N = int(input())
    nr = 0
    for p in prerjet:
         nr += N // p
         N %= p
    print(nr)
```

Sqarime

Në listën `prerjet` mbajmë vlerat e prerjeve të renditura në rendin zbritës. Për një numër të dhënë `N`, gjejmë sa herë hyn prerja më e madhe në të, pastaj sa herë hyn prerja pasardhëse në vlerën e mbetur, e kështu me rradhë.

Detyra

Çufoja ka bërë N copa keku të vogla dhe tani po mendon si ti paketojë. Çdo paketë duhet të ketë numër të njëjtë kekësh. Çufoja duhet të zgjedhë një numër të plotë A, nga 1 në N, dhe duhet të vendosë fiks A copa keku në çdo paketë. Pasi ka plotësuar aq paketa sa të jetë e mundur, copat e kekut që kanë mbetur pa paketuar Çufoja mund ti hajë vetë. Çufos i pëlqejnë shumë këta dreq kekë. Sa duhet ta zgjedhë numrin A të paketimit që ti mbeten sa më shumë copa keku për të ngrënë?

Bëni një program që merr numrin N dhe gjen numrin A, ku $2 \leq N \leq 100000000(10^8)$ Ky program duhet ta kthejë përgjigjen në më pak se 1 sek.

Referenca: https://www.codechef.com/problems/MUFFINS3

Shembull

```
$ cat input.txt
2
2
5

$ python3 prog.py < input.txt
2
3
```

Problemi 026

Kërkesa

Për një varg shkronjash S shënojmë me C bashkësinë e shkronjave që shfaqen në të të paktën 1 herë. Një përkëmbim (rradhitje) të elementeve të bashkësisë C e shënojmë $(c_1, c_2, c_3...)$. Le të jetë $f(c)$ numri i herëve që përsëritet shkronja c në vargun S.

Nëse ndonjë përkëmbim i elementeve të bashkësisë C kënaq kushtin $f(c_i) = f(c_{i-1}) + f(c_{i-2})$ për çdo $i \geq 3$ vargu S quhet **varg dinamik**.

Bëni një program që gjen nëse një varg i dhënë është dinamik.

Vini re që nëse numri i shkronjave të veçanta që shfaqen në varg është më pak se 3, p.sh. nëse $|C| < 3$, atere vargu është gjithmonë dinamik.

Referenca: https://www.codechef.com/problems/CLFIBD

Shembull

```
$ cat input.txt
3
aaaabccc
aabbcc
ppppmmnnoooopp

$ python3 prog.py < input.txt
Dynamic
Not
Dynamic
```

Zgjidhja 1

```
def is_dynamic(S):
    # count the frequency of each char of the string
    freq = {}
    for c in S:
        freq[c] = freq.get(c, 0) + 1

    if len(freq) < 3:
        return 'Dynamic'

    f = list(freq.values())
    f.sort()
    for i in range(2, len(f)):
        if f[i] != f[i-1] + f[i-2]:
            return 'Not'

    return 'Dynamic'

for _ in range(int(input())):
    print(is_dynamic(input()))
```

Zgjidhja 2

```
def is_dynamic(S):
    # count the frequency of each char of the string
    f = [S.count(c) for c in set(S)]

    if len(f) < 3:
        return 'Dynamic'

    f.sort()
    for i in range(2, len(f)):
        if f[i] != f[i-1] + f[i-2]:
            return 'Not'

    return 'Dynamic'

for _ in range(int(input())):
    print(is_dynamic(input()))
```

Detyra

Një drejtkëndësh me përmasa A dhe B duam ta ndajmë në copa katrore (jo detyrimisht të barabarta). Sa është numri më i vogël i katrorëve në të cilët mund të ndahet ky drejtkëndësh?

Shembull

```
$ cat input.txt
2
10 15
4 7

$ python3 prog.py < input.txt
3
5
```

Problemi 027

Kërkesa

Shënojmë `shuma(N)` një funksion që gjen në mënyrë efiçente shumën e numrave nga **1** në **N**. Quajmë `shumshuma(D, N)` një funksion që veprimin `shuma(N)` e zbaton **D** herë: herën e parë te **N** dhe herët e tjera te rezultati i veprimit të mëparshëm.

Për shembull, nëse `D = 2` dhe `N = 3`, atere `shumshuma(2, 3)` është e barabartë me `shuma(shuma(3)) = shuma(1 + 2 + 3) = shuma(6) = 21`

Bëni një program që gjen vlerën e `shumshuma(D, N)` për vlerat e dhëna të **D** dhe **N**.

Referenca: https://www.codechef.com/problems/PPSUM

Shembull

```
$ cat input.txt
2
1 4
2 3

$ python3 prog.py < input.txt
10
21
```

Në rastin e parë kemi `shumshuma(1, 4) = shuma(4) = 1 + 2 + 3 + 4 = 10`. Rasti i dytë është siç është sqaruar te kërkesa.

Zgjidhja 1

```
def shuma(n):
    return n * (n + 1) // 2

def shumshuma(d, n):
    if d == 1:
        return shuma(n)
    else:
        return shumshuma(d - 1, shuma(n))

for _ in range(int(input())):
    d, n = map(int, input().split())
    print(shumshuma(d, n))
```

Sqarime

Zgjidhja bazohet në faktin që `shumshuma(d, n) == shumshuma(d-1, shuma(n))`, për `d > 1`. Kurse kur `d == 1`, shumën e numrave nga `1` në `n` mund ta llogarisim me formulë.

Zgjidhja 2

```
def shuma(n):
    return n * (n + 1) // 2

for _ in range(int(input())):
    d, n = map(int, input().split())
    while d > 0:
        n = shuma(n)
        d -= 1
    print(n)
```

Detyra

Sa katrorë me madhësi **2x2** mund të futen në një trekëndësh këndrejt dybrinjëshëm, ku kateti është me gjatësi **B**, dhe brinjët e katrorëve janë paralel me katetet?

Referenca: https://www.codechef.com/problems/TRISQ

Shembull

```
$ cat input.txt
11
1
2
3
4
5
6
7
8
9
10
11

$ python3 prog.py < input.txt
0
0
0
1
1
3
3
6
6
10
10
```

Problemi 028

Kërkesa

Cufi ka gjetur dy fleta letre shumë të vjetra. Secila prej tyre fillimisht ka pasur shkronja latine, por meqënëse fletat janë shumë të vjetra, disa prej shkronjave janë bërë të palexueshme. Megjithatë numri i shkronjave në secilën fletë është i njëjtë.

Cufi do të donte të vlerësonte ndryshimin midis këtyre vargjeve. Le ta quajmë vargun e parë **S1** dhe vargun e dytë **S2**. Shkronjat e palexueshme janë shënuar me një pikëpyetje (`?`). Ndryshimi midis dy vargjeve përkufizohet si numri i pozicioneve **i** të tilla që shkronja `S1[i]` është e ndryshme nga shkronja `S2[i]`.

Ndihmojeni Cufin duke bërë një program që gjen ndryshimin më të vogël dhe më të madh midis dy vargjeve, nëse i zëvendësojmë të gjitha shkronjat e palexueshme me shkronja

të tjera.

Referenca: https://www.codechef.com/problems/CHEFSTLT

Shembull

```
$ cat input.txt
3
a?c
??b
???a
???a
?abac
aba?w

$ python3 prog.py < input.txt
1 3
0 3
3 5
```

1. Në rastin e parë mund ti zëvendësojmë pikëpyetjet në mënyrë të tillë që `S1 = 'abc'` dhe `S2 = 'abb'`. Atere S1 dhe S2 do ndryshojnë vetëm në një pozicion. Por mund ti bëjmë edhe `S1 = 'abc'` dhe `S2 = 'bab'`, dhe në këtë rast ndryshimi midis tyre është 3.
2. Po ti bëjmë zëvendësimet të tilla që `S1 = 'dcba'` dhe `S2 = 'dcba'`, atere ndryshimi është **0**. Por mund ti bëjmë zëvendësimet edhe të tilla që `S1 = 'aaaa'` dhe `S2 = 'dcba'`, dhe ndryshimi midis tyre do ishte **3**.
3. Po ti bëjmë zëvendësimet të tilla që `S1 = 'aabac'` dhe `S2 = 'abaaw'`, ndryshimi është **3**. Po ti bëjmë zëvendësimet të tilla që `S1 = 'xabac'` dhe `S2 = 'abayw'`, ndryshimi është **5**.

Zgjidhja

```python
for _ in range(int(input())):
    s1 = input()
    s2 = input()
    dmin = 0    # diferenca minimale
    dmax = 0    # diferenca maksimale
    for i in range(len(s1)):
        if s1[i] == '?' or s2[i] == '?':
            dmax += 1
```

```
        elif s1[i] != s2[i]:
            dmin += 1
            dmax += 1
    print(dmin, dmax)
```

Sqarime

Kur njëra nga shkronjat (ose të dyja) është `?`, ato mund të kenë qenë të barabarta ose mund të kenë qenë të ndryshme. Kështu që diferenca minimale nuk rritet, kurse diferenca maksimale rritet me 1.

Përndryshe (të dyja janë të ndryshme nga `?`), nëse shkronjat janë të ndryshme, rritet edhe diferenca minimale edhe ajo maksimale.

Detyra

Në një varg shkronjash, shënja `?` mund të zëvendësohet me çdo lloj shkronje tjetër. Gjeni nëse dy vargje të dhëna, me gjatësi të njëjtë, janë të përputhshëm me njëri-tjetrin (dmth nëse mund të bëhen të barabartë duke zëvendësuar shenjat `?` me shkronja).

Referenca: https://www.codechef.com/problems/TWOSTR

Shembull

```
$ cat input.txt
2
s?or?
sco??
stor?
sco??

$ python3 prog.py < input.txt
Yes
No
```

Në rastin e parë, fjala `score` përputhet me të dyja vargjet e dhëna. Kurse në rastin e dytë nuk është e mundur të përputhen.

Problemi 029

Kërkesa

Në fund të ditës, para se kuzhina e restorantit të mbyllet, është një listë me **n** punë (nga **1** në **n**) që duhen bërë. Kuzhinierët kanë bërë disa prej tyre dhe janë larguar, dhe tani ka mbetur vetëm shefi dhe ndihmësi i tij. Ata i marin punët e mbetura me rradhë: të parën shefi, të dytën ndihmësi, të tretën shefi, e kështu me rradhë (një po një jo).

Bëni një program që nxjerr punët që bën shefi dhe ato që bën ndihmësi, kur jepet numri **n** i punëve, numri **m** i punëve që janë bërë deri tani, dhe numrat e punëve që janë bërë deri tani.

Referenca: https://www.codechef.com/problems/CLEANUP

Shembull

```
$ cat input.txt
3
6 3
2 4 1
3 2
3 2
8 2
3 8

$ python3 prog.py < input.txt
3 6
5
1

1 4 6
2 5 7
```

1. Në rastin e parë janë `6` punë dhe janë bërë punët `2 4 1`. Kanë mbetur pa bërë punët `3 5 6`, kështu që shefi bën punët `3 6` kurse ndihmësi bën punën `5`.
2. Në rastin e dytë janë `3` punë dhe janë bërë punët `3 2`. Ka mbetur pa bërë vetëm puna `1`, kështu që këtë e bën shefi kurse ndihmësi nuk bën asgjë (rreshti bosh).
3. Në rastin e tretë janë `8` punë dhe janë bërë punët `3 8`. Kanë mbetur pa bërë punët `1 2 4 5 6 7`, kështu që shefi bën punët `1 4 6`, kurse ndihmësi bën punët `2 5 7`.

Zgjidhja 1

```
def print_punet(punet):
    if len(punet) == 0:
        print()
    else:
        for p in punet[:-1]:
            print(p, end=' ')
        print(punet[-1])

for _ in range(int(input())):
    n, m = map(int, input().split())
    punet = [1] + [0]*n    # liste me 1 njesh dhe n zero
    punet_e_bera = list(map(int, input().split()))
    for p in punet_e_bera:
        punet[p] = 1
    punet1 = []    # punet e shefit
    punet2 = []    # punet e ndihmesit
    rradha = 'shefi'
    for p in range(1, n+1):
        if punet[p] == 0:
            if rradha == 'shefi':
                punet1 += [p]
                rradha = 'ndihmesi'
            else:
                punet2 += [p]
                rradha = 'shefi'
    print_punet(punet1)
    print_punet(punet2)
```

Sqarime

Mbajmë një listë me të gjitha punët, ku punët e pabëra shënohen me 0, punët e bëra me 1, kurse indeksi në listë është numri i punës. Pastaj punët e pabëra i ndajmë në dy lista të tjera.

Zgjidhja 2

```
for _ in range(int(input())):
    n, m = map(int, input().split())
```

```
    punet = []    # te gjitha punet
    for i in range(n):
        punet += [i+1]
    punet_e_bera = list(map(int, input().split()))
    for p in punet_e_bera:
        punet[p-1] = -1
    punet1 = []    # punet e shefit
    punet2 = []    # punet e ndihmesit
    rradha = 'shefi'
    for p in punet:
        if p != -1:
            if rradha == 'shefi':
                punet1 += [p]
                rradha = 'ndihmesi'
            else:
                punet2 += [p]
                rradha = 'shefi'
    print(*punet1)
    print(*punet2)
```

Sqarime

E ngjashme me zgjidhjen e parë, por punët i ruajmë në listë sipas numrit të tyre, kurse punët e bëra i shënojmë me `-1`.

Kur themi `print(*list)`, i tregojmë funksionit `print()` që të printojë gjithë elementët e listës, jo vetë listën.

Zgjidhja 3

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    punet = []    # te gjitha punet
    for i in range(n):
        punet += [i+1]
    punet_e_bera = list(map(int, input().split()))
    for p in punet_e_bera:
        punet.remove(p)
    punet1 = []    # punet e shefit
    punet2 = []    # punet e ndihmesit
    rradha = 'shefi'
```

```
    for p in punet:
        if rradha == 'shefi':
            punet1 += [p]
            rradha = 'ndihmesi'
        else:
            punet2 += [p]
            rradha = 'shefi'
    print(*punet1)
    print(*punet2)
```

Sqarime

E ngjashme me zgjidhjen e dytë, por punët e bëra, në vend që ti shënojmë me `-1`, i fshijmë fare nga lista (me funksionin `remove()`). Punët që mbeten i ndajmë një andej një këtej.

Zgjidhja 4

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    punet = []    # te gjitha punet
    for i in range(n): punet += [i+1]
    punet_e_bera = list(map(int, input().split()))
    for p in punet_e_bera: punet.remove(p)
    punet1 = [punet[i] for i in range(len(punet)) if i % 2 == 0]
    punet2 = [punet[i] for i in range(len(punet)) if i % 2 == 1]
    print(*punet1)
    print(*punet2)
```

Sqarime

E ngjashme me zgjidhjen e tretë, vetëm se kodi për ndarjen e punëve të mbetura është pak më kompakt.

Detyra

Në një rrugë ndodhen 100 shtëpi dhe dyqane, me numra nga 1 në 100. Në M prej tyre banojnë policë të veprimit të shpejtë. Në rast se në ndonjë nga shtëpitë ose dyqanet ndodh ndonjë vjedhje ose incident, pronari mund të shtypë butonin e alarmit dhe policët

më të afërt vrapojnë për në vendngjarje. Nëse arrijnë brenda një kohe të caktuar, të quajtur koha kritike, mund ta kapin hajdutin me presh në dorë.

Çdo polic mund të arrijë K shtëpi (ose dyqane) brenda kohës kritike, në të dyja anët e shtëpisë së tij. Nëse te një shtëpi nuk mund të arrijë asnjë polic brenda kohës kritike, thuhet që kjo shtëpi nuk ka mbrojtje të mjaftueshme. Nëse mund të arrijnë disa policë (më shumë se një), thuhet që ka mbrojtje të shumëfishtë.

Na jepet numri K dhe vendbanimi i çdo polici. Bëni një program që gjen sa shtëpi nuk kanë mbrojtje të mjaftueshme dhe sa kanë mbrojtje të shumëfishtë.

Referenca: https://www.codechef.com/problems/COPS

Shembull

```
$ cat input.txt
3
4 56
12 52 56 8
2 20
21 75
2 40
10 51

$ python3 prog.py < input.txt
0 100
18 0
9 40
```

Problemi 030

Kërkesa

Një përkëmbim i numrave nga **1** deri në **n** është një renditje e këtyre numrave, kështu që mënyra më e natyrshme për të paraqitur një përkëmbim është që të rreshtosh numrat sipas kësaj rradhe. P.sh. një përkëmbim i numrave nga 1 deri në 5 mund të jetë kështu: `2 3 4 5 1`.

Por është edhe një mënyrë tjetër për të paraqitur një përkëmbim, duke krijuar një listë me numra, ku numri në pozicionin **i** tregon pozicionin e numrit **i** në përkëmbim. Le ta quajmë këtë **përkëmbim i anasjelltë**. Përkëmbimi i anasjelltë për vargun e mësipërm është: `5 1 2 3 4`.

Një **përkëmbim i dykuptimtë** është ai i cili nuk mund të dallohet prej përkëmbimit të tij të anasjelltë. P.sh. përkëmbimi `1 4 3 2` është i dykuptimtë sepse është i njëjtë me përkëmbimin e tij të anasjelltë.

Bëni një program i cili gjen nëse një përkëmbim i dhënë është i dykuptimtë ose jo.

Referenca: https://www.codechef.com/problems/PERMUT2

Shembull

```
$ cat input.txt
3
1 4 3 2
2 3 4 5 1
1

$ python3 prog.py < input.txt
ambiguous
not ambiguous
ambiguous
```

Zgjidhja 1

```python
for _ in range(int(input())):
    P = list(map(int, input().split()))
    for i in range(len(P)):
        if P[P[i]-1] != i+1:
            print('not ambiguous')
            break
    else:
        print('ambiguous')
```

Sqarime

Meqënëse përkëmbimet janë nga **1** deri në **n**, kurse treguesit (indekset) e listës janë nga **0** në **n-1**, duhet të kemi kujdes që ti zbresim dhe ti shtojmë 1 treguesit sipas nevojës.

Zgjidhja 2

```
for _ in range(int(input())):
    P = list(map(int, input().split()))
    P = [0] + P
    for i in range(len(P)):
        if P[P[i]] != i:
            print('not ambiguous')
            break
    else:
        print('ambiguous')
```

Sqarime

Nqs një përkëmbimi të dykuptimtë i shtojmë një **0** përpara dhe e fillojmë treguasin nga **0**, ai mbetet përsëri një përkëmbim i dykuptimtë. E njëjta gjë ndodh edhe me një përkëmbim jo të dykuptimtë. Kështu që përkëmbimit të dhënë i shtojmë një zero përpara, dhe treguesit të listës nuk është nevoja ti shtojmë ose ti zbresim **1**.

Detyra

Në një listë me këngë, secila prej këngëve e ka gjatësinë një numër të plotë minutash, dhe secila këngë e ka gjatësinë të ndryshme nga të tjerat. Kënga juaj e preferuar ndodhet në pozicionin **K**. Më pas ju i rendisni këngët sipas gjatësisë së tyre. Në cilin pozicion ndodhet tani kënga juaj e preferuar?

Referenca: https://www.codechef.com/problems/JOHNY

Shembull

```
$ cat input.txt
3
4
1 3 4 2
2
5
1 2 3 9 4
5
5
1 2 3 9 4
1

$ python3 prog.py < input.txt
3
```

```
4
1
```

1. Në rastin e parë kemi: `N = 4`, `L = [1, 3, 4, 2]`, `K = 2`. Kënga e preferuar është **3**. Pas renditjes kemi `L = [1, 2, 3, 4]`, kështu që përgjigja është **3**.
2. Në rastin e dytë: `N = 5`, `L = [1, 2, 3, 9, 4]`, `K = 5`. Kënga e preferuar është **4**. Pas renditjes do jetë në pozicionin **4**.
3. Në rastin e tretë: `N = 5`, `L = [1, 2, 3, 9, 4]`, `K = 1`. Kënga e preferuar është **1**. Pas renditjes do jetë prapë në pozicionin **1**.

5 Problemat 031-040

Problemi 031

Kërkesa

Në një varg shkronjash, shënja `?` mund të zëvendësohet me çdo lloj shkronje tjetër. Gjeni nëse dy vargje të dhëna, me gjatësi të njëjtë, janë të përputhshëm me njëri-tjetrin (dmth nëse mund të bëhen të barabartë duke zëvendësuar shenjat `?` me shkronja).

Referenca: https://www.codechef.com/problems/TWOSTR

Shembull

```
$ cat input.txt
2
s?or?
sco??
stor?
sco??

$ python3 prog.py < input.txt
Yes
No
```

Në rastin e parë, fjala `score` përputhet me të dyja vargjet e dhëna. Kurse në rastin e dytë nuk është e mundur të përputhen.

Zgjidhja 1

```
for _ in range(int(input())):
    s1 = input()
    s2 = input()
    for i in range(len(s1)):
        if s1[i] != '?' and s2[i] != '?' and s1[i] != s2[i]:
            print('No')
```

```
            break
    else:
        print('Yes')
```

Zgjidhja 2

```
def perputhet(s1, s2):
    for i in range(len(s1)):
        if s1[i] != '?' and s2[i] != '?' and s1[i] != s2[i]:
            return 'No'
    return 'Yes'

for _ in range(int(input())):
    s1 = input()
    s2 = input()
    print(perputhet(s1, s2))
```

Detyra

Vogëlushja Meli i ka qejf balonat me ngjyra dhe mami i ka blerë për ditëlindje disa balona me ngjyra të kuqe dhe blu. Mirëpo Meli do të dëshironte ti kshte të gjitha balonat me të njëjtën ngjyrë, kështu që mori kolorat dhe ju fut punës për ti ngjyrosur.

Bëni një program që gjen sa është numri më i vogël i balonave që duhen ngjyrosur, për ti bërë të gjitha me të njëjtën ngjyrë.

Referenca: https://www.codechef.com/problems/CHN09

Shembull

```
$ cat input.txt
3
ab
bb
baaba

$ python3 prog.py < input.txt
1
0
2
```

Balonat e kuqe i kemi shënuar me **a**, blutë me **b**. Në rastin e parë mjafton të ngjyrosim një balonë të kuqe me blu, ose një blu me të kuqe. Në rastin e dytë të gjitha balonat kanë të njëjtën ngjyrë. Kurse në rastin e tretë mjafton të ngjyrosim dy balonat blu me të kuqe.

Problemi 032

Kërkesa

Shefi dhe 2 punonjës të tij janë në lëvizje në terren dhe komunikojnë me radiomarrëse. Radiomarrëset e tyre mund të komunikojnë e shumta deri në një largësi R, por kanë aftësinë të ndërlidhen edhe nëse midis tyre ndodhet një radiomarrëse e tretë me largësi jo më të madhe se R nga secila prej tyre.

Bëni një program që llogarit nëse shefi dhe punonjësit mund të komunikojnë me njëritjetrin, nëse i jepen largësia R dhe koordinatat e secilës radiomarrëse.

Referenca: https://www.codechef.com/problems/COMM3

Shembull

```
$ cat input.txt
3
1
0 1
0 0
1 0
2
0 1
0 0
1 0
2
0 0
0 2
2 1

$ python3 prog.py < input.txt
yes
yes
no
```

Zgjidhja

```
import math

def largesia(x1, y1, x2, y2):
    '''
    Gjej largësinë midis pikave (x1, y1) dhe (x2, y2).
    '''
    return math.sqrt((x1 - x2)**2 + (y1 - y2)**2)

for _ in range(int(input())):
    R = int(input())
    x1, y1 = map(int, input().split())
    x2, y2 = map(int, input().split())
    x3, y3 = map(int, input().split())
    d1 = largesia(x1, y1, x2, y2)
    d2 = largesia(x1, y1, x3, y3)
    d3 = largesia(x2, y2, x3, y3)
    k1 = d1 <= R
    k2 = d2 <= R
    k3 = d3 <= R
    if (k1 and k2) or (k1 and k3) or (k2 and k3):
        print('yes')
    else:
        print('no')
```

Sqarime

Gjejmë largësitë midis radiomarrëseve. Që të mund të komunikojnë të treja, duhet që të paktën dy prej largësive të jenë jo më të mëdhaja se R.

Detyra

Vogëlushja Meli i ka qejf balonat me ngjyra dhe mami i ka blerë për ditëlindje disa balona me ngjyra të ndryshme. Mirëpo Meli do të dëshironte ti kshte të gjitha balonat me të njëjtën ngjyrë, kështu që mori kolorat dhe ju fut punës për ti ngjyrosur.

Bëni një program që gjen sa është numri më i vogël i balonave që duhen ngjyrosur, për ti bërë të gjitha me të njëjtën ngjyrë.

Shembull

```
$ cat input.txt
3
abc
bb
baabacd

$ python3 prog.py < input.txt
2
0
4
```

Çdo ngjyrë shënohet me një shkronjë të veçantë. Në rastin e parë mjafton të ngjyrosim balonat **b** dhe **c** me ngjyrën **a**. Në rastin e dytë të gjitha balonat kanë të njëjtën ngjyrë. Kurse në rastin e tretë balonat me ngjyrat **b**, **c** dhe **d** duhet ti ngjyrosim me ngjyrën **a**.

Problemi 033

Kërkesa

Në një rrugë ndodhen 100 shtëpi dhe dyqane, me numra nga 1 në 100. Në M prej tyre banojnë policë të veprimit të shpejtë. Në rast se në ndonjë nga shtëpitë ose dyqanet ndodh ndonjë vjedhje ose incident, pronari mund të shtypë butonin e alarmit dhe policët më të afërt vrapojnë për në vendngjarje. Nëse arrijnë brenda një kohe të caktuar, të quajtur koha kritike, mund ta kapin hajdutin me presh në dorë.

Çdo polic mund të arrijë K shtëpi (ose dyqane) brenda kohës kritike, në të dyja anët e shtëpisë së tij. Nëse te një shtëpi nuk mund të arrijë asnjë polic brenda kohës kritike, thuhet që kjo shtëpi nuk ka mbrojtje të mjaftueshme. Nëse mund të arrijnë disa policë (më shumë se një), thuhet që ka mbrojtje të shumëfishtë.

Na jepet numri K dhe vendbanimi i çdo polici. Bëni një program që gjen sa shtëpi nuk kanë mbrojtje të mjaftueshme dhe sa kanë mbrojtje të shumëfishtë.

Referenca: https://www.codechef.com/problems/COPS

Shembull

```
$ cat input.txt
3
4 56
12 52 56 8
```

```
2 20
21 75
2 40
10 51

$ python3 prog.py < input.txt
0 100
18 0
9 40
```

Zgjidhja

```python
for _ in range(int(input())):
    n, k = map(int, input().split())
    policat = list(map(int, input().split()))
    mbrojtja = [0 for i in range(101)]
    mbrojtja[0] = 1
    for p in policat:
        mbrojtja[p] += 1
        for i in range(1, k + 1):
            if p - i >= 1:
                mbrojtja[p - i] += 1
            if p + i <= 100:
                mbrojtja[p + i] += 1
    nr1 = 0
    nr2 = 0
    for m in mbrojtja:
        if m == 0:
            nr1 += 1
        if m > 1:
            nr2 += 1
    print(nr1, nr2)
```

Sqarime

Në listën `mbrojtja` mbajmë shënim për çdo shtëpi se nga sa polica mund të mbrohet (fillimisht 0). Pastaj për çdo polic të dhënë shtojmë 1 te secila shtëpi që mund të mbrojë (shtëpinë e vet dhe `k` në të majtë e `k` në të djathtë). Përfundimisht numërojmë shtëpitë me mbrojtje 0 dhe ato me mbrojtje më shumë se 1.

Detyra

Colit i pëlqen të luaj me numrat dhe sot po bën një lojë të tillë:

Në një listë me numra A zgjedh dy numra të njëpasnjëshëm. Më të madhin prej tyre e heq nga lista, dhe kështu gjatësia e listës zvogëlohet me 1. Kostoja e këtij veprimi është sa më i vogli prej tyre. Këtë veprim e përsërit deri sa në listë të ketë mbetur vetëm 1 numër, dhe gjen shumën e kostove për të gjitha veprimet.

Sa është vlera më e vogël e kësaj shume?

Referenca: https://www.codechef.com/problems/MNMX

Shembull

```
$ cat input.txt
2
2
3 4
3
4 2 5

$ python3 prog.py < input.txt
3
4
```

Në rastin e parë mund të kryejmë vetëm një veprim. Ky veprim fshin numrin 4 nga lista dhe e ka koston 3 (që është edhe kostoja e përgjithshme).

Në rastin e dytë mund të zgjdhim në fillim dyshen `4 2`, ku fshihet numri `4` dhe kostoja e veprimit është **2**, pastaj mund të zgjedhim dyshen `2 5`, ku fshihet numri `5` dhe kostoja e veprimit është **2**. Kostoja e përgjithshme është **4**.

Problemi 034

Kërkesa

Një sistem për manaxhimin e skedarëve të një projekti mban një listë të skedarëve të injoruar dhe një listë të skedarëve të gjurmuar (të ndjekur). Gjeni sa skedarë janë edhe të injoruar edhe të gjurmuar, dhe sa janë as të injoruar dhe as të gjurmuar.

Referenca: https://www.codechef.com/problems/VCS

Shembull

```
$ cat input.txt
2
7 4 6
1 4 6 7
1 2 3 4 6 7
4 2 2
1 4
3 4

$ python3 prog.py < input.txt
4 1
1 1
```

Kemi 2 raste testimi. Në rastin e parë kemi `N=7` numri total i skedarëve (të etiketuar nga 1 deri në 7), `M=4` numri i skedarëve të injoruar, dhe `K=6` numri i skedarëve të gjurmuar. Më poshtë kemi `1 4 6 7` lista e skedarëve të injoruar, dhe `1 2 3 4 6 7` lista e skedarëve të gjurmuar. Po ti krahasojmë, shohim se skedarët `1 4 6 7` (pra **4** skedarë) janë edhe të injoruar edhe të gjurmuar, kurse skedari `5` (pra **1** skedar) nuk është as i injoruar as i gjurmuar.

Në rastin e dytë kemi gjithsej 4 skedarë, 2 të injoruar dhe 2 të gjurmuar. Nga këta vetëm skedari `4` (pra **1** skedar) është edhe i injoruar edhe i gjurmuar, kurse skedari `2` (pra **1** skedar) nuk është as i injoruar dhe as i gjurmuar.

Zgjidhja 1

```python
for _ in range(int(input())):
    N, M, K = map(int, input().split())
    l1 = input().split()
    l2 = input().split()
    l1.sort()
    l2.sort()
    bashkimi = []
    prerja = []
    i = 0
    j = 0
    while i < M and j < K:
        if l1[i] == l2[j]:
            prerja.append(l1[i])
            bashkimi.append(l1[i])
```

```
            i += 1
            j += 1
        elif l1[i] < l2[j]:
            bashkimi.append(l1[i])
            i += 1
        else:
            bashkimi.append(l2[j])
            j += 1
    while i < M:
        bashkimi.append(l1[i])
        i += 1
    while j < K:
        bashkimi.append(l2[j])
        j += 1
    print(len(prerja), N - len(bashkimi))
```

Sqarime

Për të gjetur ata skedarë që janë edhe të injoruar edhe të gjurmuar, duhet të gjejmë prerjen e dy listave të dhëna.

Për të gjetur ata skedarë që nuk janë as të injoruar dhe as të gjurmuar, fillimisht gjejmë ata që janë të injoruar OSE të gjurmuar (bashkimin e dy listave të dhëna).

Zgjidhja 2

```
for _ in range(int(input())):
    N, M, K = map(int, input().split())
    s1 = set(input().split())
    s2 = set(input().split())
    print(len(s1.intersection(s2)), N - len(s1.union(s2)))
```

Sqarime

Duke përdorur strukturën **set** të python-it dhe veprimet e saj (`intersection()`, `union()`) zgjidhja bëhet shumë e thjeshtë.

Bëni këto prova:

```
$ python3
>>> s1 = {2, 3, 4, 5}
```

```
>>> s1
{2, 3, 4, 5}
>>> s2 = {4, 5, 5, 6, 7}
>>> s2
{4, 5, 6, 7}
>>> s1.intersection(s2)
{4, 5}
>>> s1.union(s2)
{2, 3, 4, 5, 6, 7}
>>>
```

Detyra

Në një lojë kemi **N** monedha të numëruara nga **1** në **N**, dhe fillimisht të gjitha monedhat janë nga e njëjta anë (**H** ose **T**). Loja luhet me **N** raunde, dhe në çdo round **k**, lojtari kthen **k** monedhat e para (nga **1** deri në **k**) në anën tjetër. Duam të gjejmë se sa monedha do jenë në një anë të caktuar (**H** ose **T**) në fund të lojës.

Referenca: https://www.codechef.com/problems/CONFLIP

Shembull

```
$ cat input.txt
2
1 5 1
1 5 2

$ python3 prog.py < input.txt
2
3
```

Kemi 2 raste testimi. Në rastin e parë kemi `I=1`, `N=5`, `Q=1`, që do të thotë se fillimisht monedhat janë të gjitha në anën **H**, kemi **5** monedha dhe **5** rounde, dhe në fund të raundit të 5-të duam të dimë se sa monedha janë në anën **H**.

Në rastin e dytë është e njëjta gjë, vetëm se duam të dimë se sa monedha janë në anën **T** (`Q=2`).

Gjendja e monedhave pas çdo raundi do jetë e tillë: 1. **T** H H H H 2. **H T** H H H 3. **T H T** H H 4. **H T H T** H 5. **T H T H T**

Kështu që përgjigja për rastin e parë është **2**, kurse për rastin e dytë është **3**.

Problemi 035

Kërkesa

Në një reaktor bërthamor janë K dhoma njëra pas tjetrës, të etiketuara nga 0 në K-1. Grimcat vazhdimisht mblidhen në dhomën 0. Por nëse bëhen më shumë se N grimca në një dhomë, ndodh një reaksion i cli bën që një nga grimcat të kalojë në dhomën pasardhëse, kurse të tjerat zhduken. E njëjta gjë vazhdon edhe në dhomat pasardhëse, deri sa asnjë nga dhomat të mos ketë më shumë se N grimca. Pasi të gjitha këto reaksione kanë përfunduar, vjen grimca tjetër në dhomën 0, e kështu me rradhë.

Nëse numri total i grimcave të bombarduara është A, sa është numri i grimcave në secilën dhomë në fund të bombardimit?

Referenca: https://www.codechef.com/problems/NUKES

Shembull

```
$ cat input.txt
3
3 1 3
11 2 2
100 13 3

$ python3 prog.py < input.txt
1 1 0
2 0
2 7 0
```

Në rastin e parë, pasi bombardohet grimca e parë gjendja e dhomave është `1 0 0`. Pasi bombardohet grimca e dytë, te dhoma e parë bëhen 2, kështu që një grimcë kalon te dhoma djathtas, kurse të tjerat zhduken: `0 1 0`. Pas grimcës së tretë gjendja është: `1 1 0`.

Zgjidhja 1

```
for _ in range(int(input())):
    A, N, K = map(int, input().split())
    R = [0] * K
    for i in range(A):
        R[0] += 1
        for j in range(K-1):
```

```
            if R[j] > N:
                R[j+1] += 1
                R[j] = 0
            else:
                break
        if R[-1] > N:
            R[-1] = 0
        #print(R)    # debug
    print(' '.join(map(str, R)))
```

Sqarime

Thjesht simulojmë dhomat e reaktorit, bombardimin e A grimcave, dhe reaksionin sa herë që bëhen më shumë se N grimca në një dhomë. Në fund shfaqim rezultatin që kemi në R.

Zgjidhja 2

```
for _ in range(int(input())):
    A, N, K = map(int, input().split())
    R = [0] * K
    N += 1
    i = 0
    while A > 0 and i < K:
        R[i] = A % N
        A //= N
        i += 1
    print(' '.join(map(str, R)))
```

Sqarime

Kjo zgjidhje është edhe më e thjeshtë edhe më efiçente (më e shpejtë) se e para, sepse këtu duhen K hapa (ose veprime) për të gjetur rezultatin, kurse te e para duhen A hapa (që mund të jetë edhe numër shumë i madh).

Kjo zgjidhje bazohet në faktin që mënyra se si zhvillohen reaksionet është e ngjashme me paraqitjen e numrit A në bazën N+1, kështu që ne mund ta gjejmë rezultatin duke përdorur mbetjen dhe herësin e pjesëtimit me N+1.

Detyra

Gjuhët e harruara janë gjuhë që janë përdorur dikur gjerësisht por sot nuk përdoren më. Megjithatë disa prej fjalëve të tyre mund të jenë ende në përdorim në gjuhët e sotme.

You keni gjetur në internet N fjalë të një gjuhe të harruar. Gjithashtu keni edhe K fjali që përdoren në gjuhët e sotme. Detyra juaj është që të përcaktoni për secilën prej N fjalëve nëse gjendet në ndonjërën nga këto K fjalitë ose jo.

Referenca: https://www.codechef.com/problems/FRGTNLNG

Shembull

```
$ cat input.txt
2
3 2
piygu ezyfo rzotm
1 piygu
6 tefwz tefwz piygu ezyfo tefwz piygu
4 1
kssdy tjzhy ljzym kegqz
4 kegqz kegqz kegqz vxvyj

$ python3 prog.py < input.txt
YES YES NO
NO NO NO YES
```

Kemi 2 raste testimi. Në rastin e parë, kemi `N=3` fjalë të gjuhës së harruar dhe `K=2` fjali të gjuhëve të sotme. Pastaj vijnë 2 fjalitë, ku e para ka 1 fjalë dhe e dyta ka 6 fjalë. Dy fjalët e para ndodhen në këtë fjali, kurse e treta jo.

Problemi 036

Kërkesa

Një varg shkronjash quhet palindorm nëse duke u lexuar nga fillimi në fund dhe nga fundi në fillim është njësoj. Për shembull emri ANA nëse lexohet mbrapsht do të jetë përsëri e njëjta fjalë. Bëni një program që kontrollon nëse një varg i dhënë është palindrom.

Referenca: https://www.codechef.com/problems/PALL01

Shembull

```
$ cat input.txt
5
6666
123321
kapak
ana
palindrom

$ python3 prog.py < input.txt
Yes
Yes
Yes
Yes
No
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    S = input()
    n = len(S)
    for i in range(n):
        if S[i] != S[n-i-1]:
            print('no')
            break
    else:
        print('yes')
```

Sqarime

Për çdo shkronjë kontrollojmë nëse është e njëjtë me shkronjën korresponduese nga fundi.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    S = input()
    n = len(S)
```

```
    i = 0
    while i < n:
        if S[i] != S[n-i-1]:
            print('no')
            break
        i += 1
    else:
        print('yes')
```

Sqarime

Zgjidhje e njëjtë me të parën, vetëm se në vend të for-it përdorim ciklin while.

Zgjidhja 3

```
T = int(input())
for t in range(T):
    S = input()
    n = len(S)
    i = 0
    j = n-1
    while  i < j:
        if S[i] != S[j]:
            print('no')
            break
        i += 1
        j -= 1
    else:
        print('yes')
```

Sqarime

Në këtë zgjidhje përdorim 2 indekse, një që niset nga fillimi dhe një që niset nga fundi. Cikli ndërpritet kur të dy indekset takohen në mes të vargut.

Zgjidhja 4

```python
for _ in range(int(input())):
    S = input()
    l1 = list(S)
    l2 = list(S)
    l2.reverse()
    print('Yes') if l1 == l2 else print('No')
```

Sqarime

Në këtë zgjidhje përdorim funksionin `reverse()` të listës që e kthen një listë në renditjen e kundërt.

Detyra

Në kohët e lashta, komandanti i një ushtrie ishte supersticioz. Ai e quante një ushtar "me fat" nëse numri i armëve që kishte ushtari ishte çift dhe "pa fat" në të kundërt. Ushtrinë e quante "gati për betejë" nëse numri i ushtarëve me fat ishte më i madh se ai i ushtarëve pa fat. Në të kundërt e quante "jo gati".

Nëse ju jepet numri i armëve që ka çdo ushtar, bëni një program që gjen nëse ushtria është gati për betejë ose jo.

Referenca: https://www.codechef.com/problems/AMR15A

Shembull

```
$ cat input.txt
5
1
1
1
2
4
11 12 13 14
3
2 3 4
5
1 2 3 4 5

$ python3 prog.py < input.txt
NOT READY
READY FOR BATTLE
```

```
NOT READY
READY FOR BATTLE
NOT READY
```

Problemi 037

Kërkesa

Bëni një program i cili gjen nëse një listë me numra përfshihet si nënlistë në një listë tjetër.

Shembull

```
$ cat input.txt
4
6
1 2 3 4 5 6
3
2 3 4
6
22 5 6 33 1 4
2
4 15
6
22 5 6 33 1 4
2
1 4
6
22 5 6 33 1 4
2
4 1

$ python3 prog.py < input.txt
Yes
No
Yes
No
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n1 = int(input())
    L1 = list(map(int, input().split()))
    n2 = int(input())
    L2 = list(map(int, input().split()))
    i = 0
    while i < n1-n2+1:
        found = True
        j = 0
        while j < n2:
            if L1[i+j] != L2[j]:
                found = False
                break
            j += 1
        if found:
            break
        i += 1

    if found:
        print("Yes")
    else:
        print("No")
```

Sqarime

Për çdo pozicion në listën e parë kontrollojmë nëse ndodhet aty lista e dytë. Nëse e gjejmë diku, e mbyllim ciklin sepse nuk është nevoja të kërkojmë më tej.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    n1 = int(input())
    L1 = list(map(int, input().split()))
    n2 = int(input())
    L2 = list(map(int, input().split()))
    for i in range(n1-n2+1):
```

```
        for j in range(n2):
            if L1[i+j] != L2[j]:
                break
        else:
            print("Yes")
            break
    else:
        print('No')
```

Sqarime

E ngjashme me zgjidhjen e parë, por e ndërtuar me `for`, që e bën programin më kompakt.

Detyra

Bëni një program i cili gjen nëse numrat e një liste përmbahen në një listë tjetër.

Referenca: https://www.codechef.com/problems/CHEFSQ

Shembull

```
$ cat input.txt
4
6
1 2 3 4 5 6
3
2 3 4
6
22 5 6 33 1 4
2
4 15
6
22 5 6 33 1 4
2
1 4
6
22 5 6 33 1 4
2
4 1

$ python3 prog.py < input.txt
Yes
```

```
No
Yes
Yes
```

Problemi 038

Kërkesa

Në një listë me numra natyrorë, gjeni numrin e nënlistave për të cilat shuma dhe prodhimi i elementëve është i barabartë.

Referenca: https://www.codechef.com/problems/CHEFARRP

Shembull

```
$ cat input.txt
3
3
1 3 2
4
4 1 2 1
6
1 2 2 2 2 1

$ python3 prog.py < input.txt
4
5
9
```

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    L = list(map(int, input().split()))
    nr = 0
    for i in range(n):
        s = 0
        p = 1
        for j in range(i, n):
            s += L[j]
            p *= L[j]
```

```
            if s == p:
                nr += 1
    print(nr)
```

Sqarime

Për të gjetur të gjitha listat përdorim dy treguesa (indekse) `i` dhe `j`, ku i-ja lëviz nga elementi i parë i listës deri tek i fundit, kurse j-ja lëviz nga i-ja deri në fund të listës. Për secilën nënlist (ose segment) që fillon në `i` dhe mbaron në `j`, llogarisim shumën dhe prodhimin e numrave përbërës, por këtë e bëjmë në mënyrë dinamike (duke shtuar një element të ri në listë dhe duke përtërirë shumën dhe prodhimin e listës së mëparshme). Sa herë që shuma dhe prodhimi bëhen të barabartë, shtojmë `1` te përgjigja.

Detyra

Kërkesa

Roboti Bani ka humbur rrugën. Ai ndodhet në kordinatat (x_1, y_1) në një plan 2 dimensional, kurse shtëpia e tij ndodhet në kordinatat (x_2, y_2). Bëni një program që ta ndihmojë duke i treguar drejtimin në të cilin duhet të lëvizë për të arritur në shtëpi. Nëse i tregojmë një drejtim, roboti do të vazhdojë të lëvizë në atë drejtim deri sa të arrijë në shtëpi. Janë katër drejtime që mund tia japim si komandë: **majtas**, **djathtas**, **lartë**, **poshtë**. Nëse asnjë nga këto drejtime nuk e çon robotin në shtëpi, programi duhet të nxjerrë: **më vjen keq**.

Referenca: https://www.codechef.com/problems/ICPC16A

Shembull

```
$ cat input.txt
5
0 0 1 0
0 0 0 1
0 0 1 1
2 0 0 0
0 2 0 0

$ python3 prog.py < input.txt
right
up
sad
left
down
```

Problemi 039

Kërkesa

Na jepet një numër binar (i cili ka vetëm shifrat **0** dhe **1**). Bëni një program i cili gjen nëse është e mundur që numri ti ketë të gjitha shifrat njësoj, duke ndryshuar vetëm një shifër (nga 1 në 0 ose nga 0 në 1).

Referenca: https://www.codechef.com/problems/LONGSEQ

Shembull

```
$ cat input.txt
4
101
11
0
10000

$ python3 prog.py < input.txt
Yes
No
Yes
Yes
```

Në rastin e parë, duke ndryshuar shifrën e mesit në 1, të gjitha shifrat bëhen njësoj. Në rastin e dytë, pavarësisht se cilën shifër ndryshojmë nuk mund ti bëjmë të gjitha njësoj. Në rastin e tretë mund ta ndryshojmë 0-n në 1 dhe të gjitha shifrat do jenë njësoj. Në rastin e katërt mund ta bëjmë shifrën e parë zero, dhe të gjitha shifrat do jenë njësoj.

Zgjidhja 1

```
T = int(input())
for t in range(T):
    N = input()
    cnt0 = 0
    cnt1 = 0
    for d in N:
        if d == '0':
            cnt0 += 1
        else:
            cnt1 += 1
```

```
    if cnt0 == 1 or cnt1 == 1:
        print('Yes')
    else:
        print('No')
```

Sqarime

Gjejmë numrin e zerove dhe të njëshave. Nëse ndonjëra prej tyre është 1, atere është e mundur, përndryshe jo.

Zgjidhja 2

```
for _ in range(int(input())):
    N = input()
    if N.count('0') == 1 or N.count('1') == 1:
        print('Yes')
    else:
        print('No')
```

Zgjidhja 3

```
for _ in range(int(input())):
    N = input()
    cnt1 = sum(map(int, list(N)))
    cnt0 = len(N) - cnt1
    if cnt1 == 1 or cnt0 == 1:
        print('Yes')
    else:
        print('No')
```

Detyra

Na jepet një numër binar (i cili ka vetëm shifrat **0** dhe **1**). Bëni një program i cili gjen nëse është e mundur që të bëhet i barabartë numri i zerove dhe njëshave, duke ndryshuar të shumtën një shifër (nga 1 në 0 ose nga 0 në 1).

Shembull

```
$ cat input.txt
4
101
11
0
10000

$ python3 prog.py < input.txt
No
Yes
No
No
```

Në rastin e parë nuk është e mundur ti bëjmë të barabarta numrat e zerove dhe njëshave. Në rastin e dytë mund ta bëjmë një njësh zero dhe do kemi numër të barabartë zerosh dhe njëshash. Në rastin e tretë nuk mund të kemi një numër të barabartë zerosh dhe njëshash. Në rastin e katërt gjithashtu nuk mund të bëjmë një numër të barabartë zerosh dhe njëshash.

Problemi 040

Kërkesa

Një listë **L** ka **N** numra të plotë **të ndryshëm nga zero**. Një nën-listë quhet **alternuese** nëse çdo dy elemente fqinje të listës kanë shenja të kundërta (njëra pozitive dhe tjetra negative).

Për çdo element të listës **L** gjeni gjatësinë e nën-listës alternuese më të madhe që fillon me atë element.

Referenca: https://www.codechef.com/problems/ALTARAY

Shembull

```
$ cat input.txt
3
4
1 2 3 4
4
1 -5 1 -5
6
```

```
-5 -1 -1 2 -2 -3

$ python3 prog.py < input.txt
1 1 1 1
4 3 2 1
1 1 3 2 1 1
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    N = int(input())
    L = list(map(int, input().split()))
    P = []
    for i in range(N):
        k = 1
        while i+k < N and L[i+k-1]*L[i+k] < 0:
            k += 1
        P.append(k)
    print(*P)
```

Sqarime

Për çdo element të listës gjejmë se sa elemente alternuese janë mbas tij. Kushtin `i+k < N` e përdorim për tu siguruar që indeksi (treguesi) `i+k` nuk del jashtë listës. Kurse kushtin `x*y < 0` e përdorim për të kontrolluar që dy elementë kanë shenja të kundërta.

Komanda `print(*P)` përdoret për të printuar elementet e një liste, të ndarë me vende bosh.

Zgjidhja 2

```
T = int(input())
for t in range(T):
    N = int(input())
    L = list(map(int, input().split()))
    P = []
    for i in range(N):
        P.append(0)
    L.append(L[-1])
```

```
    i = N
    while i > 0:
        if L[i-1] * L[i] < 0:
            P[i-1] = P[i] + 1
        else:
            P[i-1] = 1
        i -= 1
    print(*P)
```

Sqarime

Kjo zgjidhje bazohet në faktin që gjatësitë e nën-listave më të mëdha alternuese për dy elemente të një-pas-njëshëm `i` dhe `i+1` kanë lidhje me njëra-tjetrën. Nëse elementët `i` dhe `i+1` kanë shenja të kundërta, atëherë gjatësia për elementin `i` është një më tepër sesa për elementin `i+1`. Nëse kanë shenja të njëjta, atere është `1`.

Kështu që zgjidhja bëhet duke filluar nga fundi i listës dhe duke llogaritur gjatësinë për një element duke u bazuar te gjatësia e elementit pasardhës. Elementi i fundit i listës përbën një rast të veçantë, sepse nuk ka element para-ardhës, kështu që ne shtojmë një element fallco në fund të listës, thjesht për të shmangur trajtimin e tij si rast i veçantë.

Kjo zgjidhje mund të duket pak më e vështirë për tu konceptuar në krahasim me zgjidhjen e parë, por në fakt është më e shpejtë. Zgjidhja e parë është zgjidhje katrore (ose e rendit $O(N^2)$) meqënëse kemi dy cikle brënda njëri-tjetrit. Kurse kjo zgjidhje është lineare (ose e rendit $O(N)$) meqënëse kemi vetëm një cikël.

Që zgjidhja e dytë është më e shpejtë mund ta provoni edhe duke i provuar këto zgjidhje këtu: https://www.codechef.com/submit/ALTARAY Zgjidhja e parë nuk e kalon testin sepse është shumë e ngadalshme, kurse e dyta po.

Detyra

Një listë **L** ka **N** numra të plotë **të ndryshëm nga zero**. Një nën-listë quhet **alternuese** nëse çdo dy elemente fqinje të listës kanë shenja të kundërta (njëra pozitive dhe tjetra negative).

Gjeni gjatësinë e nën-listës alternuese më të madhe të listës **L**.

Shembull

```
$ cat input.txt
3
4
1 2 3 4
```

```
4
1 -5 1 -5
6
-5 -1 -1 2 -2 -3

$ python3 prog.py < input.txt
1
4
3
```

6 Problemat 041-050

Problemi 041

Kërkesa

Bëni një program i cili gjen nëse numrat e një liste përmbahen në një listë tjetër.

Referenca: https://www.codechef.com/problems/CHEFSQ

Shembull

```
$ cat input.txt
4
6
1 2 3 4 5 6
3
2 3 4
6
22 5 6 33 1 4
2
4 15
6
22 5 6 33 1 4
2
1 4
6
22 5 6 33 1 4
2
4 1

$ python3 prog.py < input.txt
Yes
No
Yes
Yes
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n1 = int(input())
    L1 = list(map(int, input().split()))
    n2 = int(input())
    L2 = list(map(int, input().split()))
    for n in L2:
        if n not in L1:
            print('No')
            break
    else:
        print('Yes')
```

Sqarime

Për çdo element të listës së dytë kontrollojmë nëse ndodhet në listën e parë.

Zgjidhja 2

```
for _ in range(int(input())):
    input()
    S1 = set(map(int, input().split()))
    input()
    S2 = set(map(int, input().split()))
    print('Yes') if S2.issubset(S1) else print('No')
```

Sqarime

Duke i trajtuar listat e dhëna të numrave si bashkësi, kontrollojmë nëse bashkësia e dytë është nënbashkësi e të parës (me anë të funksionit `issubset()`)

Detyra

Dejvi ka disa shokë të çuditshëm. Me rastin e ditëlindjes së tij secili prej tyre i ka kërkuar që ta qerasë në një datë të caktuar, me kushtin që do ta prishë shoqërinë në rast se nuk e qeras pikërisht në atë datë. Mirëpo Dejvi nuk mund të qerasë më shumë se një prej tyre në një datë të caktuar. Gjeni se me sa prej tyre mund ta ruajë shoqërinë.

Referenca: https://www.codechef.com/problems/CFRTEST

Shembull

```
$ cat input.txt
2
2
3 2
2
1 1

$ python3 prog.py < input.txt
2
1
```

Në rastin e parë, të dy shokët e kërkojnë qerasjen në data të ndryshme, kështu që mund ta ruajë shoqërinë me të dy. Në rastin e dytë, të dy e kërkojnë qerasjen në të njëjtën ditë, kështu që mund ta ruajë shoqërinë vetëm me njërin prej tyre.

Problemi 042

Kërkesa

Igori i vogël i ka qejf filmat. Këtë fundjavë janë **n** filma që mund të shihen, ku secili filëm ka gjatësi L_i nga 1 deri në 100 minuta, dhe vlerësim R_i nga 1 deri në 100 pikë. Por Igori ka mundësi të shohë vetëm 1 filëm, kështu që ai dëshiron të zgjedhë atë filëm që ka vlerë maksimale për $L_i * R_i$. Nëse janë disa filma të tillë, atëherë ai do zgjidhte atë që ka vlerën më të madhe të R_i. Nëse prapë janë disa të tillë, atëherë do të zgjidhte atë që është i pari në listë. Bëni një program për ta ndihmuar Igorin të zgjedhë një filëm.

Referenca: https://www.codechef.com/problems/MOVIEWKN

Shembull

```
$ cat input.txt
2
2
1 2
2 1
4
2 1 4 1
2 4 1 4

$ python3 prog.py < input.txt
```

```
1
2
```

Rreshti i parë jep vlerat **L**, rreshti i dytë jep vlerat **R**.

Në rastin e parë të dy filmat kanë të njëjtën vlerë të $L * R$, por filmi i parë ka vlerësim (R) më të mirë.

Në rastin e dytë, filmi i dytë dhe i katërt janë njësoj të mirë, kështu që kemi zgjedhur të parin që është në listë.

Zgjidhja

```
T = int(input())
for t in range(T):
    n = int(input())
    L = list(map(int, input().split()))
    R = list(map(int, input().split()))
    v_max = 0
    r_max = 0
    i_max = 0
    for i in range(n):
        l = L[i]
        r = R[i]
        v = l * r
        if v > v_max:
            v_max = v
            r_max = r
            i_max = i
        elif v == v_max and r > r_max:
            v_max = v
            r_max = r
            i_max = i
    print(i_max + 1)
```

Detyra

Pas një kohe të gjatë, Cufi vendosi më në fund ta rinovojë shtëpinë. Shtëpia e Cufit ka **N** dhoma. Secila prej dhomave është e lyer me një nga ngjyrat e kuqe, blu dhe jeshile. Konfigurimi i ngjyrave të shtëpisë na jepet me anë të një vargu me shkronjat **R**, **B**, **G**. Cufi do që ta lyejë shtëpinë në mënyrë që të gjitha dhomat të kenë të njëjtën ngjyrë, por ngaqë është pak dembel, do që të lyejë sa më pak dhoma (mundësisht asnjë). Bëni një program që gjen numrin më të vogël të dhomave që duhet të lyejë Cufi.

Referenca: https://www.codechef.com/problems/COLOR

Shembull

```
$ cat input.txt
3
3
RGR
3
RRR
3
RGB

$ python3 prog.py < input.txt
1
0
2
```

Problemi 043

Kërkesa

Një listë **L** ka **N** numra të plotë **të ndryshëm nga zero**. Një nën-listë quhet **alternuese** nëse çdo dy elemente fqinje të listës kanë shenja të kundërta (njëra pozitive dhe tjetra negative).

Gjeni gjatësinë e nën-listës alternuese më të madhe të listës **L**.

Shembull

```
$ cat input.txt
3
4
1 2 3 4
4
1 -5 1 -5
6
-5 -1 -1 2 -2 -3

$ python3 prog.py < input.txt
1
4
3
```

Zgjidhja

```
T = int(input())
for t in range(T):
    N = int(input())
    L = list(map(int, input().split()))
    lmax = 0
    l = 0
    L.insert(0, L[0])
    L.append(L[-1])
    i = N+1
    while i > 0:
        if L[i-1] * L[i] < 0:
            l += 1
        else:
            if l > lmax:
                lmax = l
            l = 1
        i -= 1
    print(lmax)
```

Detyra

Loli, vëllai i vogël i Colit, sapo ka mësuar disa prej shkronjave të alfabetit, dhe mund të lexojë vetëm ato fjalë që përbëhen prej shkronjave që njeh. Coli i jep atij një libër për të lexuar dhe do të dijë se cilat prej fjalëve mund të lexojë Loli.

Referenca: https://www.codechef.com/problems/ALPHABET

Shembull

```
$ cat input.txt
act
2
cat
dog

$ python3 prog.py < input.txt
Yes
No
```

Rreshi i parë përmban shkronjat që ka mësuar Loli. Fjala e parë i ka të gjitha shkronjat të njohura, fjala e dytë jo.

Problemi 044

Kërkesa

Një listë me numra quhet jo-zbritëse nqs çdo numër i listës është <= se numri pasardhës. Një nën-listë e një liste me numra përbëhet nga një ose disa numra të njëpasnjëshëm të listës.

Bëni një program që gjen se sa nën-lista jo-zbritëse ka në një listë të dhënë.

Referenca: https://www.codechef.com/problems/SUBINC

Shembull

```
$ cat input.txt
2
4
1 4 2 3
1
5

$ python3 prog.py < input.txt
6
1
```

Në rastin e parë kemi 6 nën-lista jo-zbritëse: `[1]`, `[1, 4]`, `[4]`, `[2]`, `[2, 3]`, `[3]`. Në rastin e dytë kemi vetëm 1: `[5]`.

Zgjidhja

```
T = int(input())
for t in range(T):
    n = int(input())
    L = list(map(int, input().split()))
    L.append(L[-1] - 1)
    ans = 0
    nr = 1
    for i in range(n):
        if L[i] <= L[i+1]:
            nr +=1
        else:
            ans += nr*(nr+1)//2
```

```
            nr = 1
    print(ans)
```

Sqarime

Mund të vihet re lehtë se të gjitha nën-listat e një liste jo-zbritëse janë edhe ato jo-zbritëse. Po ashtu mund të vërtetohet lehtë se listë me `n` numra ka `n*(n+1)//2` nën-lista.

Zgjidhja gjen segmentet jo-zbritëse të listës së dhënë dhe gjatësinë `nr` të secilit prej tyre. Çdo segment i tillë i shton përgjigjes `nr*(nr+1)//2` nën-lista jo-zbritëse.

Detyra

Dejvi shkoi në pazar dhe pa fshatarët që po shisnin kosha me rrush. Mirëpo numrat e kileve që kishin koshat nuk i pëlqyen se iu dukën pa lidhje. Dejvi do të donte që pjesëtuesi më i madh i përbashkët i të gjithë numrave të ishte i plotpjesëtueshëm me K. Bëni një program që gjen numrin më të vogël të kileve që duhen shtuar ose hequr nëpër kosha, që të plotësohet ky kusht dhe asnjë kosh të mos ngelet bosh.

Referenca: https://www.codechef.com/problems/DEVUGRAP

Shembull

```
$ cat input.txt
2
2 2
3 5
3 7
10 16 18

$ python3 prog.py < input.txt
2
8
```

Në rastin e parë kemi 2 kosha dhe numrat e kileve duhet të plotpjesëtohen me 2. Mjafton të shtojmë ose të heqim 1 kile nga secili prej koshave, që të plotësohet ky kusht.

Në rastin e dytë kemi 3 kosha dhe numrat e kileve duhet të plotpjesëtohen me 7. Mund të heqim 3 kile në koshin e parë, të heqim 2 kile koshin e dytë, dhe të shtojmë 3 kile në koshin e tretë. Gjithsej, $3 + 2 + 3 = 8$.

Problemi 045

Kërkesa

Andi ka mbledhur **A** kokrra manaferrash kurse Beni **B**. Ata po bëjnë një lojë të tillë: Andi ha 1 kokërr, pastaj Beni 2, pastaj Andi 3, e kështu me rradhë. I pari që nuk i kanë mbetur për të ngrënë aq kokrra sa duhet e humbet lojën. Bëni një program që gjen se kush prej tyre e fiton lojën.

Referenca: https://www.codechef.com/problems/CANDY123

Shembull

```
$ cat input.txt
10
3 2
4 2
1 1
1 2
1 3
9 3
9 11
9 12
9 1000
8 11

$ python3 prog.py < input.txt
Beni
Andi
Andi
Beni
Beni
Andi
Andi
Beni
Beni
Beni
```

Në rastin e parë Andi ha 1, Beni 2, dhe Andi nuk mund të hajë 3, kështu që lojën e fiton Beni.

Në rastin e shtatë, Andi ha 1, Beni 2, Andi 3, Beni 4, Andi 5, dhe Beni nuk ka 6 kokrra, kështu që fiton Andi.

Zgjidhja

```
T = int(input())
for t in range(T):
    a, b = map(int, input().split())
    i = 0
    while True:
        i += 1
        if a < i:
            print('Beni')
            break
        else:
            a -= i
        i += 1
        if b < i:
            print('Andi')
            break
        else:
            b -= i
```

Sqarime

Kemi një cikël të pafund i cili ndërpritet kur njëri nga lojtarët e humb lojën, ndërkohë që ne printojmë si fitues emrin e lojtarit tjetër.

Detyra

Na jepen dy vargje shkronjash **A** dhe **B**. A është e mundur të gjejmë një nënvarg **s1** në **A** dhe një nënvarg **s2** në **B**, që mos të jenë bosh, dhe që **s1+s2** të jetë palindromë?

Shenja + tregon bashkimin ose ngjitjen e dy vargjeve, kurse palindromë do të thotë që po ta lexosh vargun nga fundi në fillim është njësoj me vargun fillestar (nga fillimi në fund).

Referenca: https://www.codechef.com/problems/STRPALIN

Shembull

```
$ cat input.txt
3
abc
abc
```

```
a
b
abba
baab

$ python3 prog.py < input.txt
Yes
No
Yes
```

Në rastin e parë, një nga zgjedhjet e mundshme është `s1 = "ab"` dhe `s2 = "a"`. Atere kemi `s1+s2 = "aba"` e cila është palindromë, kështu që përgjigja është "Yes".

Në rastin e dytë kemi `A = "a"` dhe `B = "b"` dhe nuk ka asnjë mundësi që të krijojmë ndonjë palindromë, kështu që përgjigja është "No".

Problemi 046

Kërkesa

Një listë quhet "listë ylber" nëse plotëson këto kushte:

- a_1 elementët e parë janë **1**
- a_2 elementët e tjerë janë **2**
- a_3 elementët e tjerë janë **3**
- a_4 elementët e tjerë janë **4**
- a_5 elementët e tjerë janë **5**
- a_6 elementët e tjerë janë **6**
- a_7 elementët e tjerë janë **7**
- a_6 elementët e tjerë janë **6**
- a_5 elementët e tjerë janë **5**
- a_4 elementët e tjerë janë **4**
- a_3 elementët e tjerë janë **3**
- a_2 elementët e tjerë janë **2**
- a_1 elementët e fundit janë **1**
- ku a_i është një numër natyror (jo-zero) i çfarëdoshëm
- dhe nuk ka elementë të tjerë në listë përveç këtyre

Bëni një program që gjen nëse një listë numrash është listë ylber.

Referenca: https://www.codechef.com/problems/RAINBOWA

Shembull

```
$ cat input.txt
```

```
3
19
1 2 3 4 4 5 6 6 6 7 6 6 6 5 4 4 3 2 1
14
1 2 3 4 5 6 7 6 5 4 3 2 1 1
13
1 2 3 4 5 6 8 6 5 4 3 2 1

$ python3 prog.py < input.txt
yes
no
no
```

Rasti i parë i plotëson të gjitha kushtet.

Rasti i dytë ka vetëm një njësh në fillim dhe dy njësha në fund.

Rasti i tretë nuk ka asnjë element **7** dhe ka elemente tepër (**8**).

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    L = list(map(int, input().split()))
    i = 0
    j = n-1
    for a in range(1,8):
        if L[i] != a:
            print('no')
            break
        while L[i] == a and L[j] == a and i < j:
            i += 1
            j -= 1
    else:
        if i < j:
            print('no')
        else:
            print('yes')
```

Sqarime

Meqënëse elementët e listës duhet të jenë simetrikë në lidhje me qendrën, përdorim dy treguesa `i` dhe `j`, të cilët nisen prej fillimit dhe fundit dhe vazhdojnë të ecin deri sa të

takohen. Gjithashtu kontrollojmë nëse elementët e listës marrin me rradhë vlerat nga 1 deri në 7. Mënyra se si e kemi ndërtuar zgjidhjen siguron që asnjë nga këto vlera të mos mungojë, dhe gjithashtu të mos ketë vlera të tjera përveç këtyre.

Detyra

Alisa dhe Beni po bëjnë një lojë me numra. Fillimisht Alisa ka një numër **A** kurse Beni një numër **B**. Loja ka gjithsej **N** hapa dhe Alisa me Benin luajnë me rradhë. Kur është rradha e tij, secili lojtar e shumëzon numrin e tij me dy. E para luan Alisa.

Bëni një program i cili merr numrat **A**, **B** dhe **N** dhe gjen sa është herësi i pjesëtimit të numrave në fund të lojës (duke pjesëtuar më të madhin me më të voglin).

Referenca: https://www.codechef.com/problems/TWONMS

Shembull

```
$ cat input.txt
3
1 2 1
3 2 3
3 7 2

$ python3 prog.py < input.txt
1
3
2
```

Në rastin e parë, numrat fillestarë janë (A=1, B=2) dhe loja ka vetëm 1 hap. Në këtë hap Alisa e shumëzon numrin e saj me 2, kështu që në fund të lojës kemi (A=2, B=2) dhe herësi është 1.

Në rastin e dytë kemi (A=3, B=2) dhe N=3. Pasi luan Alisa, Beni, Alisa, kemi (A=12, B=4), kështu që herësi është 3.

Në rastin e tretë kemi (A=3, B=7) dhe N=2. Pasi luan Alisa dhe Beni, kemi (A=6, B=14), kështu që herësi është 2.

Problemi 047

Kërkesa

Një rradhë e vlefshme kllapash është një varg jo-bosh, ku çdo element i vargut është ose '(' ose ')', dhe i cili plotëson kushtin që duke fshirë në mënyrë të përsëritur çifte kllapash të njëpasnjëshme '()' është e mundur të bëhet varg bosh.

P.sh. `(())` dhe `((((()))))` janë rradhë të vlefshme kllapash, kurse `)()(` dhe `(()` nuk janë.

Mikeli ka një rradhë të vlefshme kllapash. Ai është i kënaqur me të, përveç faktit që është shumë e gjatë. Kështu që Mikeli ka vendosur ta zëvendësojë me një varg më të shkurtër. Por jo çdo rradhë e vlefshme kllapash është e kënaqshme për të. Për të kuptuar kërkesat e tij, po përkufizojmë një funksion **F(S)** të tillë:

```
FUNCTION F( S - a valid parentheses sequence )
    BEGIN
        count = 0
        max_count = 0
        FOR index FROM 1 TO LENGTH(S)
        BEGIN
            if S[index] == '(' then count = count + 1
            if S[index] == ')' then count = count - 1
            max_count = max( max_count, count )
        END
        RETURN max_count
    END
```

Me fjalë të tjera, **F(S)** është thellësia maksimale e kllapave në një rradhë të vlefshme **S**.

Le të shënojmë me **A** rradhën e kllapave që ka Mikeli dhe **B** një rradhë kandidate për ta zëvendësuar. Mikeli është i gatshëm ta zëvendësojë rradhën **A** me **B** nëse **F(B)** është e barabartë me **F(A)**. Gjithashtu do të donte të zgjidhte një **B** të tillë që ka gjatësi sa më të shkurtër. Nëse ka disa rradhë **B** që i plotësojnë këto kushte do të preferonte atë që është leksikografisht më e vogël, duke konsideruar '(' më të vogël se ')'.

Bëni një program që zgjidh këtë problem për Mikelin.

Referenca: https://www.codechef.com/problems/BRACKETS

Shembull

```
$ cat input.txt
1
()((()()))

$ python3 prog.py < input.txt
(((())))
```

Zgjidhja

```
def F(S):
    cnt = 0
    max_cnt = 0
    for c in S:
        if c == '(':
            cnt += 1
        if c == ')':
            cnt -= 1
        if cnt > max_cnt:
            max_cnt = cnt
    return max_cnt

for _ in range(int(input())):
    n = F(input())
    print(n*'(' + n*')')
```

Sqarime

Mjafton të gjejmë thellësinë maksimale të kllapave duke përdorur funksionin e dhënë, dhe mund të ndërtojmë një rradhë kllapash me të njëjtën thellësi dhe që ka gjatësinë më të vogël të mundshme.

Detyra

Jepen disa shkopinj me gjatësi të përcaktuara. Gjeni sipërfaqen e drejtkëndëshit më të madh që mund të formohet me këta shkopinj, ose printoni -1 nëse asnjë drejtkëndësh nuk mund të formohet.

Referenca: https://www.codechef.com/problems/STICKS

Shembull

```
$ cat input.txt
2
5
1 2 3 1 2
4
1 2 2 3
```

```
$ python3 prog.py < input.txt
2
-1
```

Në rastin e parë mund të formohet një drejtkëndësh me brinjë 1 dhe 2. Në rastin e dytë asnjë drejtkëndësh nuk mund të formohet.

Problemi 048

Kërkesa

Pjesëtuesi më i madh i përbashkët (PMP) i disa numrave është ai numër që i plotpjesëton të gjithë.

Na jepet një listë me **N** numra natyrorë dhe na lejohet të fshimë disa prej tyre (nga **0** deri në **N-1**, mjafton që lista të mos ngelet bosh).

Bëni një program që gjen numrin më të vogël të elementëve që duhen fshirë, në mënyrë që PMP-ja e numrave që mbeten të jetë **1**. Nëse është e pamundur që PMP-ja të bëhet **1**, programi duhet të printojë `-1`.

Referenca: https://www.codechef.com/problems/RD19

Shembull

```
$ cat input.txt
2
2
2 3
2
2 4

$ python3 prog.py < input.txt
0
-1
```

Në rastin e parë nuk është nevoja të fshimë asnjë numër sepse `pmp(2,3) == 1`.

Në rastin e dytë është e pamundur që PMP të bëhet 1, pavarësisht se cilët numra fshijmë.

Zgjidhja 1

```
def pmp(a, b):
    '''
    Gjej pjesetuesin me te madh te perbashket te dy numrave te dhene.
    '''
    while a > 0:
        a, b = b % a, a
    return b

T = int(input())
for t in range(T):
    input()
    L = list(map(int, input().split()))

    # gjej pjesetuesin me te madh te perbashket
    p = L[0]
    for n in L:
        p = pmp(p, n)

    print(0) if p==1 else print(-1)
```

Sqarime

Nqs një numër i plotpjesëton të gjithë numrat e një liste dhe ne heqim disa prej numrave të listës, ky numër i plotpjesëton gjithsesi numrat e mbetur të listës. Kështu që nuk ka shance që PMP i një liste të zvogëlohet duke fshirë disa prej elementeve të listës (vetëm mund të rritet).

Si pasojë, nqs PMP i numrave të listës është **1**, atere nuk është nevoja të fshimë asnjë prej numrave të listës dhe përgjigja është **0**. Përndryshe është e pamundur që PMP të bëhet 1 duke fshirë numra të listës, dhe përgjigja është **-1**.

Programi ynë thjesht gjen PMP e të gjithë numrave të listës dhe shikon nëse është **1** ose jo.

Zgjidhja 2

```
import math

for _ in range(int(input())):
    input()
    L = list(map(int, input().split()))
```

```
    # gjej pjesetuesin me te madh te perbashket
    pmp = L[0]
    for n in L:
        pmp = math.gcd(pmp, n)

    print(0) if pmp==1 else print(-1)
```

Sqarime

Përdorim funksionin e gatshëm `gcd()` të librarisë `math`.

Provoni këto komanda në terminal:

```
$ python3
>>> import math
>>> help(math)
>>> dir(math)
>>> help(math.gcd)
>>> quit()
```

Shënim: Shtypni **q** për të dalë nga `help()`.

Zgjidhja 3

```
from math import gcd
from functools import reduce

for _ in range(int(input())):
    input()
    L = list(map(int, input().split()))

    # gjej pjesetuesin me te madh te perbashket
    pmp = reduce(gcd, L)

    print(0) if pmp==1 else print(-1)
```

Sqarime

Funksioni `functools.reduce()` merr si parameter të parë emrin e një funksioni, dhe si parameter të dytë një listë. Duke përdorur funksionin e dhënë ai i përpunon elementët e listës 2 e nga 2, deri sa ta reduktojë listën në një vlerë të vetme.

P.sh. `reduce(gcd, [2, 3, 4, 5])` është e njëvlerëshme me `gcd(gcd(gcd(2, 3), 4), 5)`.

Shikoni edhe help-in:

```
$ python3
>>> from functools import reduce
>>> help(reduce)
[Ctrl+D]
```

Detyra

Cufi sapo ka marrë një lidhje interneti të re. Të dhënat e përdorimit të internetit prej tij janë si më poshtë.

Në T_1 minutat e para shpejtësia e shkarkimit që ka përdorur është D_1 MB në minutë, në T_2 minutat e tjera është D_2 MB/min, e kështu me rradhë deri në T_N minutat e fundit kur shpejtësia ka qenë D_N MB/min.

Kompania e internetit i faturon 1 dollar për çdo 1 MB të shkarkuar, me përjashtim të K minutave të para, të cilat janë falas (si pjesë e ofertës).

Gjeni sasinë totale të dollarëve që duhet të paguajë Cufi për internetin e përdorur.

Referenca: https://www.codechef.com/problems/DWNLD

Shembull

```
$ cat input.txt
3
2 2
2 1
2 3
2 2
1 2
2 3
3 0
1 2
2 4
10 10

$ python3 prog.py < input.txt
6
3
110
```

Çdo rast testimi ka si rresht të parë numrat **N** dhe **K**, pastaj vijnë **N** rreshta me të dhënat e shkarkimit **T** dhe **D**.

Në testin e parë, 2 minutat e para janë falas, kështu që kostoja totale është: `2*3 = 6`.

Në testin e dytë, 2 minutat e para janë falas, kështu që kostoja totale është: `1*3 = 3`.

Në rastin e tretë, 0 minuta janë falas, kështu që kostoja totale është: `1*2 + 2*4 + 10*10 = 110`.

Problemi 049

Kërkesa

Në një varg **S** me shkronja, gjejmë një nënvarg që formon fjalën `CHEF` dhe i heqim këto shkronja. Gjeni sa herë mund ta bëjmë këtë veprim.

Referenca: https://www.codechef.com/problems/CHRL2

Shembull

```
$ cat input.txt
2
CHEFCHEFFFF
CHHHEEEFFCC

$ python3 prog.py < input.txt
2
1
```

Zgjidhja 1

```
W = 'CHEF'
for _ in range(int(input())):
    S = list(input())
    total_cnt = 0
    while True:
        cnt = 0
        i = 0
        j = 0
        while i < len(S):
            if S[i] == W[j]:
                S[i] = 'X'
```

```
                j += 1
                if j == 4:
                    cnt += 1
                    j = 0
            i += 1
        total_cnt += cnt
        if cnt == 0:
            break
    print(total_cnt)
```

Sqarime

Me indeksin `i` marrim me rradhë shkronjat në vargun e dhënë. Me indeksin `j` shënojmë shkronjat në `CHEF`. Shkronjat që përputhen i ndryshojmë në `X` që mos ti kontrollojmë prapë herën tjetër. Kur gjejmë shkronja koresponduese për të katra shkronjat e `CHEF` (d.m.th kur `j` bëhet 4) rrisim numrin e fjalëve të gjetura (`cnt`) me 1 dhe e kthejmë indeksin `j` në fillim. Këtë punë e përsërisim deri sa të mos gjenden më fjalë të tilla në vargun e dhënë (`cnt == 0`).

Zgjidhja 2

```
for _ in range(int(input())):
    S = input()
    c = ch = che = chef = 0
    for l in S:
        if l == 'C':
            c += 1
        elif l == 'H':
            if c > 0:
                c -= 1
                ch += 1
        elif l == 'E':
            if ch > 0:
                ch -= 1
                che += 1
        elif l == 'F':
            if che > 0:
                che -= 1
                chef += 1
    print(chef)
```

Sqarime

Te ndryshorja `c` ruajmë numrin e shkronjave `C` që kemi gjetur deri në një moment të caktuar, por që nuk mund të lidhen me një `H` nga mbrapa. Te ndryshorja `ch` mbajmë numrin e nënvargjeve `CH` që kemi gjetur, por që nuk mund të formojnë nënvargun `CHE`. E njëjta gjë edhe për ndryshoret `che` dhe `chef`.

Nqs në këtë moment shikojmë p.sh. një `E` në varg, atere kjo mund të lidhet me një nga nënvargjet `CH` që kemi numëruar deri tani dhe të krijojë një nënvarg `CHE`, kështu që pakësojmë me 1 ndryshoren `ch` dhe shtojmë me 1 ndryshoren `che`. Por kjo mund të bëhet nqs kemi parë të paktën një nënvarg `CH` deri tani.

Në përfundim, te ndryshorja `chef` do kemi numrin e gjithë nënvargjeve `CHEF` të mundshëm.

Zgjidhja 3

```
for _ in range(int(input())):
    S = input()
    c = ch = che = chef = 0
    for l in S:
        if l == 'C':
            c += 1
        elif l == 'H':
            if c > ch:
                ch += 1
        elif l == 'E':
            if ch > che:
                che += 1
        elif l == 'F':
            if che > chef:
                chef += 1
    print(chef)
```

Sqarime

Llogjika është e ngjashme me zgjidhjen e dytë, vetëm se te ndryshorja `ch` (për shembull) mbajmë të gjitha nënvargjet e mundshme `CH` që kemi parë deri tani, që mund të jenë pjesë ose jo e nënvargjeve `CHE` ose `CHEF`. Nqs tani shohim një `E` dhe numri i të gjitha nënvargjeve `CH` që kemi parë deri tani është më i madh se numri i të gjitha nënvargjeve `CHE`, atere këtë shkronjë `E` mund ta lidhim me ndonjë nga nënvargjet `CH` që janë tepër dhe të krijojmë një nënvarg të ri `CHE`. Kështu që e rrisim me 1 ndryshoren `che`.

Shënim: Duhet vënë në dukje se zgjidhja e dytë dhe e tretë janë më të shpejta se zgjidhja e parë, sepse këto e gjejnë përgjigjen me një kontroll të vetëm në vargun e dhënë, kurse zgjidhja e parë duhet ta kontrollojë vargun disa herë, deri sa të mos gjejë më ndonjë nga nënvargjet e kërkuar. Këtë gjë mund ta verifikoni edhe duke i provuar këto zgjidhje këtu: https://www.codechef.com/submit/CHRL2 Zgjidhja e parë të jep vetëm 25 pikë, kurse 2 të tjerat të japin 100 pikë të plota.

Detyra

Gimi është i famshëm për dembelizmin e tij në shkollë. Ai i lë gjithmonë gjërat për në minutën e fundit. Tani ka **N** problema për të zgjidhur në një detyrë që duhet ta dorëzoj nesër, dhe siç mund ta merrni me mënd nuk ka zënë gjë me dorë.

Por ai ka një plan, si gjithmonë. Puna e parë, bleu një pako me RedBull. Pastaj do punojë pa pushim deri sa të zgjidhë gjysmat e problemave (nëse **N** është çift gjysma është **N/2**, përndryshe është **(N+1)/2**). Pastaj do pushojë për **B** minuta. Pastaj do vazhdojë me gjysmat e problemave që mbeten dhe do bëjë përsëri një pushim prej **B** minutash, e kështu me rradhë deri sa ti mbarojë të gjitha problemat. Në fillim atij i duhen **M** minuta për të zgjidhur një problem, por pas çdo pushimi koha e zgjidhjes së një problemi dyfishohet.

Sa kohë i duhet Gimit deri sa të mbarojë edhe problemin e fundit?

Referenca: https://www.codechef.com/problems/TALAZY

Shembull

```
$ cat input.txt
2
9 1 2
123456 123456 123456

$ python3 prog.py < input.txt
45
131351258112
```

Në rastin e parë, janë 9 problema për tu zgjidhur, koha e pushimit është 1 minutë, dhe fillimisht i duhen 2 minuta për të zgjidhur një problemë. Fillimisht Gimi do zgjidhë 5 problemat e para, dhe për këto i duhen `5*2 = 10` minuta. Pastaj do bëjë 1 minutë pushim. Pastaj do zgjidhë 2 problemat e tjera, dhe për këto i duhen `2*4 = 8` minuta. Pastaj do bëjë prapë 1 minutë pushim. Pastaj do zgjidhë 1 problem për 8 minuta. Pastaj prapë 1 min pushim. Pastaj do zgjidhë problemin e fundit për 16 min. Gjithsej do ti duhen `10 + 1 + 8 + 1 + 8 + 1 + 16 = 45` minuta për ti përfunduar të gjitha problemat.

Problemi 050

Kërkesa

Një listë me numra **A** quhet *e bukur* nëse për çdo dy numra të listës $a_i, a_j (i <> j)$, gjendet në listë një numër a_k i tillë që $a_k = a_i * a_j$. Vini re që **k** mund të jetë edhe **i** ose **j**.

Gjeni nëse një listë e dhënë është *e bukur* ose jo.

Referenca: https://www.codechef.com/problems/ICPC16B

Shembull

```
$ cat input.txt
3
2
0 1
2
1 2
2
5 6

$ python3 prog.py < input.txt
yes
yes
no
```

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    A = list(map(int, input().split()))
    for i in range(n):
        for j in range(n):
            if i == j:
                continue
            if A[i]*A[j] not in A:
                print('no')
                break
        else:
            continue
```

```
        break
    else:
        print('yes')
```

Sqarime

Programi bën një cikël të dyfishtë për ti shqyrtuar të gjitha dyshet e mundshme të elementeve, por përjashton ato që kanë indeks të njëjtë (është i njëjti element). Nqs gjen dy numra prodhimi i të cilëve nuk ndodhet në listë, nxirret përgjigja "no" dhe ciklet ndërpriten. Përndryshe, nëse cikli i dyfishtë vazhdon deri në fund pa u ndërprerë, nxirret përgjigja "yes".

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    A = list(map(int, input().split()))
    b = True  # let's assume that it is beautiful
    i = 0
    while b and i < n:
        j = 0
        while b and j < n:
            if i != j and  A[i]*A[j] not in A:
                b = False
            j += 1
        i += 1
    print('yes') if b else print('no')
```

Sqarime

Llogjika është e ngjashme me zgjidhjen e parë, në kuptimin që shqyrtohen të gjitha dyshet e mundshme dhe prodhimi i tyre. Në fillim supozojmë se lista është e bukur dhe vazhdojmë kontrollin deri sa të gjejmë një dyshe numrash, prodhimi i të cilave nuk ndodhet në listë. Nëse gjejmë një dyshe të tillë, ndryshorja **b** bëhet **False**, çfarë bën që edhe të dyja ciklet të ndërpriten.

Kompleksiteti (ose ndërlikushmëria) e këtyre dy zgjidhjeve është $O(n^3)$ (ose kubike), sepse janë dy cikle nga **1** në **n** brënda njëri-tjetrit, por gjithashtu edhe kur kërkojmë të dimë nëse një element ndodhet në listë, zakonisht na duhen $O(n)$ veprime (mesatarisht $n/2$).

Nqs i testojmë këto dy zgjidhje te https://www.codechef.com/submit/ICPC16B ato nuk e kalojnë testin sepse ekzekutimi i programit zgjat mbi 10 sekonda, ndërkohë që limiti i kohës së ekzekutimit që ka problemi është vetëm 2 sekonda.

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    A = list(map(int, input().split()))

    n0 = n1 = n_1 = n_other = 0
    for a in A:
        if a == 0:
            n0 += 1
        elif a == 1:
            n1 += 1
        elif a == -1:
            n_1 += 1
        else:
            n_other += 1

    if n == 1:
        print('yes')
    elif n_other > 1:
        print('no')
    elif n_1 == 0:
        print('yes')
    elif n_other == 1:
        print('no')
    elif n_1 == 1:
        print('yes')
    elif n1 > 0:
        print('yes')
    else:
        print('no')
```

Sqarime

Kjo zgjidhje duket pak më ndërlikuar se dy zgjidhjet e mëparshme, por në fakt është shumë më e shpejtë, me kompleksitet të rendit $O(n)$. Si rrjedhojë kjo zgjidhje e kalon testin në https://www.codechef.com/submit/ICPC16B

Kjo zgjidhje bazohet në disa vëzhgime rreth cilësive të listave të bukura:

- Nëse një listë ka vetëm 1 element, është e bukur.
- Nëse në një listë janë 2 (ose më shumë) numra më të mëdhenj se 1, lista nuk mund të jetë e bukur.
- Nëse lista ka vetëm një numër më të madh se 1 dhe të tjerët janë **0** ose **1**, është e bukur.
- Nëse lista ka më shumë se një **-1**, duhet të ketë edhe një **1** për të qenë e bukur.
- Nëse një listë ka një **-1** dhe një numër më të madh se 1, nuk mund të jetë e bukur.

Ndryshorja `n_1` mban sa **-1** ka në listë, `n1` sa **1**, `n0` sa **0**, dhe `n_other` sa numra të tjerë (më të mëdhenj se 1 ose më të vegjël se -1).

Detyra

Sergei ka bërë **N** matje dhe do të gjejë vlerën mesatare të tyre. Por për të gjetur një mesatare sa më të mirë, ai mendon se duhet ti heqë nga lista e matjeve **K** vlerat më të vogla dhe **K** vlerat më të mëdha, para se të llogarisë mesataren. Sa del vlera mesatare në këtë mënyrë?

Referenca: https://www.codechef.com/problems/SIMPSTAT

Shembull

```
$ cat input.txt
3
5 1
2 9 -10 25 1
5 0
2 9 -10 25 1
3 1
1 1 1

$ python3 prog.py < input.txt
4.000000
5.400000
1.000000
```

7 Problemat 051-060

Problemi 051

Kërkesa

Dejvi shkoi në pazar dhe pa fshatarët që po shisnin kosha me rrush. Mirëpo numrat e kileve që kishin koshat nuk i pëlqyen se iu dukën pa lidhje. Dejvi do të donte që pjesëtuesi më i madh i përbashkët i të gjithë numrave të ishte i plotpjesëtueshëm me K. Bëni një program që gjen numrin më të vogël të kileve që duhen shtuar ose hequr nëpër kosha, që të plotësohet ky kusht dhe asnjë kosh të mos ngelet bosh.

Referenca: https://www.codechef.com/problems/DEVUGRAP

Shembull

```
$ cat input.txt
2
2 2
3 5
3 7
10 16 18

$ python3 prog.py < input.txt
2
8
```

Në rastin e parë kemi 2 kosha dhe numrat e kileve duhet të plotpjesëtohen me 2. Mjafton të shtojmë ose të heqim 1 kile nga secili prej koshave, që të plotësohet ky kusht.

Në rastin e dytë kemi 3 kosha dhe numrat e kileve duhet të plotpjesëtohen me 7. Mund të heqim 3 kile në koshin e parë, të heqim 2 kile koshin e dytë, dhe të shtojmë 3 kile në koshin e tretë. Gjithsej, $3 + 2 + 3 = 8$.

Zgjidhja

```
T = int(input())
for t in range(T):
    n, k = map(int, input().split())
    L = list(map(int, input().split()))
    a = 0
    for i in L:
        if i < k:
            a += k - i
        else:
            m = i % k
            a += min(m, k - m)
    print(a)
```

Sqarime

Në mënyrë që pjesëtuesi më i madh i përbashkët i të gjithë numrave të plotëpjesëtohet me K, mjafton që secili prej numrave të plotëpjesëtohet me K. Për një numër të caktuar shikojmë mbetjen M të pjesëtimit me K. Nëse kjo është më e vogël se (K-M), atere numrit i heqim M, përndryshe i shtojmë (K-M). Në këtë mënyrë plotësojmë kushtin që numri i kileve të hequra apo të shtuara të jetë sa më i vogël, dhe të gjithë numrat të plotpjesëtohen me K. Vetëm se në rast se vetë numri është më i vogël se K, atere duhet që gjithmonë ti shtojmë diçka që të bëhet K, në mënyrë që të plotësojmë edhe kushtin që asnjë kosh të mos mbetet bosh (ose asnjë nga numrat të mos bëhet 0).

Detyra

Çdo ditë Majko shkon në punë me autobus, ku pret një biletë. Çdo biletë ka një kod me shkronja latine. Majko beson se dita do ti shkojë mbarë nëse kodi i biletës ka vetëm 2 shkronja alternuese, si p.sh. **xyxyxy...** ku **x!=y**.

Bëni një program që merr kodin e biletës dhe nxjerr "YES" nëse është alternues, dhe "NO" në të kundërt.

Referenca: https://www.codechef.com/problems/TICKETS5

Shembull

```
$ cat input.txt
2
ABABAB
ABC
```

```
$ python3 prog.py < input.txt
YES
NO
```

Problemi 052

Kërkesa

Jepen disa shkopinj me gjatësi të përcaktuara. Gjeni sipërfaqen e drejtkëndëshit më të madh që mund të formohet me këta shkopinj, ose printoni -1 nëse asnjë drejtkëndësh nuk mund të formohet.

Referenca: https://www.codechef.com/problems/STICKS

Shembull

```
$ cat input.txt
2
5
1 2 3 1 2
4
1 2 2 3

$ python3 prog.py < input.txt
2
-1
```

Në rastin e parë mund të formohet një drejtkëndësh me brinjë 1 dhe 2. Në rastin e dytë asnjë drejtkëndësh nuk mund të formohet.

Zgjidhja

```
T = int(input())
for t in range(T):
    n = int(input())
    L = list(map(int, input().split()))
    L.sort(reverse=True)
    a = 0
    i = 0
    while i < n-1:
        if L[i] == L[i+1]:
```

```
            if a > 0:
                b = L[i]
                print(a*b)
                break
            else:
                a = L[i]
                i += 2
        else:
            i += 1
    else:
        print(-1)
```

Sqarime

Duam të gjejmë 4 brinjë 2 e nga dy të barabarta, me gjatësi maksimale. Për ta lehtësuar këtë i rendisim shkopinjtë në rendin zbritës dhe fillojmë kontrollin. Kur gjejmë çiftin e parë të barabartë e ruajmë gjatësinë e tyre te ndryshorja `a`. Kur gjejmë çiftin tjetër printojmë direkt sipërfaqen dhe e mbyllim ciklin e kërkimit (me `break`). Normalisht kjo duhet të ndodhë para se të arrijmë në fund të kërkimit, por nqs arrijmë deri në fund të kërkimit dhe nuk kemi gjetur gjë akoma, atere printojmë -1.

Detyra

Shefi ka gjetur një libër me receta gatimesh, ku çdo gatim ka 4 përbërës. Ai do zgjedh 2 receta për ti gatuar për drekë, por dëshiron që këto dy gatime të mos jenë të ngjashme. Dy gatime quhen të ngjashme nëse kanë 2 ose më shumë përbërës të njëjtë.

Bëni një program që merr përbërësit e dy gatimeve dhe gjen nëse janë të ngjashme ose jo.

Referenca: https://www.codechef.com/problems/SIMDISH

Shembull

```
$ cat input.txt
5
eggs sugar flour salt
sugar eggs milk flour
aa ab ac ad
ac ad ae af
cookies sugar grass lemon
lemon meat chili wood
one two three four
```

```
one two three four
gibberish jibberish lalalalala popopopopo
jibberisz gibberisz popopopopu lalalalalu

$ python3 prog.py < input.txt
similar
similar
dissimilar
similar
dissimilar
```

Problemi 053

Kërkesa

Cufi sapo ka marrë një lidhje interneti të re. Të dhënat e përdorimit të internetit prej tij janë si më poshtë.

Në T_1 minutat e para shpejtësia e shkarkimit që ka përdorur është D_1 MB në minutë, në T_2 minutat e tjera është D_2 MB/min, e kështu me rradhë deri në T_N minutat e fundit kur shpejtësia ka qenë D_N MB/min.

Kompania e internetit i faturon 1 dollar për çdo 1 MB të shkarkuar, me përjashtim të K minutave të para, të cilat janë falas (si pjesë e ofertës).

Gjeni sasinë totale të dollarëve që duhet të paguajë Cufi për internetin e përdorur.

Referenca: https://www.codechef.com/problems/DWNLD

Shembull

```
$ cat input.txt
3
2 2
2 1
2 3
2 2
1 2
2 3
3 0
1 2
2 4
10 10
```

```
$ python3 prog.py < input.txt
6
3
110
```

Çdo rast testimi ka si rresht të parë numrat **N** dhe **K**, pastaj vijnë **N** rreshta me të dhënat e shkarkimit **T** dhe **D**.

Në testin e parë, 2 minutat e para janë falas, kështu që kostoja totale është: `2*3 = 6`.

Në testin e dytë, 2 minutat e para janë falas, kështu që kostoja totale është: `1*3 = 3`.

Në rastin e tretë, 0 minuta janë falas, kështu që kostoja totale është: `1*2 + 2*4 + 10*10 = 110`.

Zgjidhja

```python
for _ in range(int(input())):
    n, k = map(int, input().split())
    total = 0
    for i in range(n):
        t, d = map(int, input().split())
        if k > 0:
            m = min(t, k)
            t -= m
            k -= m
        total += (t * d)
    print(total)
```

Detyra

Cufit i pëlqen të luajë pingpong. Ai gjeti disa shënime me pikët e lojrave që ka bërë para ca kohësh. Me **1** shënohet nëse ka fituar një pikë dhe me **0** nëse ka humbur një pikë (e ka fituar kundërshtari). Loja e pingpongut fitohet nga lojtari që bën i pari 11 pikë, me përjashtim të rastit kur të dy lojtarët kanë nga 10 pikë, në të cilin loja vazhdon deri sa njëri prej lojtarëve të dalë 2 pikë para kundërshtarit. Gjeni se cilat prej lojrave i ka fituar Cufi dhe cilat i ka humbur.

Referenca: https://www.codechef.com/problems/TTENIS

Shembull

```
$ cat input.txt
```

```
2
0101111111111
11100000000000

$ python3 prog.py < input.txt
WIN
LOSE
```

Problemi 054

Kërkesa

Gimi është i famshëm për dembelizmin e tij në shkollë. Ai i lë gjithmonë gjërat për në minutën e fundit. Tani ka **N** problema për të zgjidhur në një detyrë që duhet ta dorëzoj nesër, dhe siç mund ta merrni me mënd nuk ka zënë gjë me dorë.

Por ai ka një plan, si gjithmonë. Puna e parë, bleu një pako me RedBull. Pastaj do punojë pa pushim deri sa të zgjidhë gjysmat e problemave (nëse **N** është çift gjysma është **N/2**, përndryshe është **(N+1)/2**). Pastaj do pushojë për **B** minuta. Pastaj do vazhdojë me gjysmat e problemave që mbeten dhe do bëjë përsëri një pushim prej **B** minutash, e kështu me rradhë deri sa ti mbarojë të gjitha problemat. Në fillim atij i duhen **M** minuta për të zgjidhur një problem, por pas çdo pushimi koha e zgjidhjes së një problemi dyfishohet.

Sa kohë i duhet Gimit deri sa të mbarojë edhe problemin e fundit?

Referenca: https://www.codechef.com/problems/TALAZY

Shembull

```
$ cat input.txt
2
9 1 2
123456 123456 123456

$ python3 prog.py < input.txt
45
131351258112
```

Në rastin e parë, janë 9 problema për tu zgjidhur, koha e pushimit është 1 minutë, dhe fillimisht i duhen 2 minuta për të zgjidhur një problemë. Fillimisht Gimi do zgjidhë 5 problemat e para, dhe për këto i duhen `5*2 = 10` minuta. Pastaj do bëjë 1 minutë pushim. Pastaj do zgjidhë 2 problemat e tjera, dhe për këto i duhen `2*4 = 8` minuta.

Pastaj do bëjë prapë 1 minutë pushim. Pastaj do zgjidhë 1 problem për 8 minuta. Pastaj prapë 1 min pushim. Pastaj do zgjidhë problemin e fundit për 16 min. Gjithsej do ti duhen `10 + 1 + 8 + 1 + 8 + 1 + 16 = 45` minuta për ti përfunduar të gjitha problemat.

Zgjidhja

```
for _ in range(int(input())):
    n, b, m = map(int, input().split())
    koha = 0
    while n > 0:
        if n % 2 == 0:
            p = n // 2
        else:
            p = (n + 1) // 2
        koha += p * m
        koha += b
        n -= p
        m *= 2
    koha -= b
    print(koha)
```

Detyra

Është një ditë feste dhe Cufi duhet të raportojë rreth parakalimeve që po bëhen në bulevard. Grupe të ndryshme njerëzish parakalojnë në formë autokolone, njëra pas tjetrës, ku dy autokolona mund të kenë edhe njëfarë distance nga njëra-tjetra. Sa herë që shikon fillimin e një autokolone Cufi shënon në një fletore një **H**, gjatë kohës që autokolona ecën ai vazhdon të shënojë pika (**.**), dhe kur shikon fundin e autokolonës shënon një **T**. Gjithashtu, për të shënuar distancat midis autokolonave ai përdor pika.

Meqënëse autokolonat kalojnë me rradhë njëra pas tjetrës, një raport i saktë do ishte diçka e tillë: "**..H..T...HTH....T.**", ose "**...**", ose "**HT**". Kurse "**T...H..H.T**", "**H..T..H**" dhe "**H..H..T..T**" do ishin të pasakta.

Formalisht, një autokolonë paraqitet nga një **H**, që pasohet nga disa pika (ndoshta asnjë), dhe në fund ka një **T**. Një raport i saktë është ai që fillon me disa pika (ndoshta asnjë), vazhdon me disa (ndoshta zero) autokolona midis të cilave mund të ketë edhe pika, dhe mbaron me disa pika (ndoshta asnjë).

Cufi ka ndenjur vonë mbrëmë duke festuar dhe është pak përgjumësh, kështu që raporti i tij mund të jetë i pasaktë. Bëni një program që verifikon nëse raporti është i saktë ose jo.

Referenca: https://www.codechef.com/problems/SNAKPROC

Shembull

```
$ cat input.txt
6
18
..H..T...HTH....T.
3
...
10
H..H..T..T
2
HT
11
.T...H..H.T
7
H..T..H

$ python3 prog.py < input.txt
Valid
Valid
Invalid
Valid
Invalid
Invalid
```

"**H..H..T..T**" nuk është e saktë sepse kolona e dytë fillon para se të mbarojë e para.

"**.T...H..H.T**" nuk është i saktë sepse ka një fund kolone **T** pa pasur ndonjë fillim **H**.

"**H..T..H**" nuk është i saktë sepse ka një fillim kolone **H** që nuk ka fund **T**.

Problemi 055

Kërkesa

Është një ditë feste dhe Cufi duhet të raportojë rreth parakalimeve që po bëhen në bulevard. Grupe të ndryshme njerëzish parakalojnë në formë autokolone, njëra pas tjetrës, ku dy autokolona mund të kenë edhe njëfarë distance nga njëra-tjetra. Sa herë që shikon fillimin e një autokolone Cufi shënon në një fletore një **H**, gjatë kohës që autokolona ecën ai vazhdon të shënojë pika (**.**), dhe kur shikon fundin e autokolonës shënon një **T**. Gjithashtu, për të shënuar distancat midis autokolonave ai përdor pika.

Meqënëse autokolonat kalojnë me rradhë njëra pas tjetrës, një raport i saktë do ishte diçka e tillë: "**..H..T...HTH....T.**", ose "**...**", ose "**HT**". Kurse "**T...H..H.T**", "**H..T..H**" dhe "**H..H..T..T**" do ishin të pasakta.

Formalisht, një autokolonë paraqitet nga një **H**, që pasohet nga disa pika (ndoshta asnjë), dhe në fund ka një **T**. Një raport i saktë është ai që fillon me disa pika (ndoshta asnjë), vazhdon me disa (ndoshta zero) autokolona midis të cilave mund të ketë edhe pika, dhe mbaron me disa pika (ndoshta asnjë).

Cufi ka ndenjur vonë mbrëmë duke festuar dhe është pak përgjumësh, kështu që raporti i tij mund të jetë i pasaktë. Bëni një program që verifikon nëse raporti është i saktë ose jo.

Referenca: https://www.codechef.com/problems/SNAKPROC

Shembull

```
$ cat input.txt
6
18
..H..T...HTH....T.
3
...
10
H..H..T..T
2
HT
11
.T...H..H.T
7
H..T..H

$ python3 prog.py < input.txt
Valid
Valid
Invalid
Valid
Invalid
Invalid
```

"**H..H..T..T**" nuk është e saktë sepse kolona e dytë fillon para se të mbarojë e para.

"**.T...H..H.T**" nuk është i saktë sepse ka një fund kolone **T** pa pasur ndonjë fillim **H**.

"**H..T..H**" nuk është i saktë sepse ka një fillim kolone **H** që nuk ka fund **T**.

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    s = input()
    i = 0
    while True:
        while i < n and s[i] == '.':
            i += 1
        if i == n:
            print('Valid')
            break
        if s[i] == 'H':
            i += 1
        else:
            print('Invalid')
            break
        while i < n and s[i] == '.':
            i += 1
        if i == n:
            print('Invalid')
            break
        if s[i] == 'T':
            i += 1
        else:
            print('Invalid')
            break
```

Sqarime

Kontrollojmë plotësimin e kushteve për të qënë një raport i rregullt:

- të ketë një numër të çfardoshëm pikash në fillim, në mes ose në fund
- shkronja **H** të jetë e para dhe pas saj të vijë një shkronjë **T**

Zgjidhja 2

```
def check(n, s):
    i = 0
    while True:
        while i < n and s[i] == '.':
```

```
            i += 1
        if i == n:
            return True
        if s[i] == 'H':
            i += 1
        else:
            return False
        while i < n and s[i] == '.':
            i += 1
        if i == n:
            return False
        elif s[i] == 'T':
            i += 1
        else:
            return False

for _ in range(int(input())):
    n = int(input())
    s = input()
    print('Valid') if check(n, s) else print('Invalid')
```

Sqarime

E njëjtë me zgjidhjen e parë, por e strukturuar me një funksion që kthen True ose False.

Zgjidhja 3

```
import re
for _ in range(int(input())):
    input()
    print('Valid') if re.search("^\.*(H\.*T\.*)*$", input()) else print('Invalid')
```

Sqarime

Zgjidhje që përdor *shprehjet e rregullta* (https://docs.python.org/3/howto/regex.html)

Detyra

Mësuesi ka ngritur disa nxënës në dërrasë dhe u ka dhënë nga një ushtrim për të zgjidhur (klasa ka një dërrasë të gjatë kështu që mësuesi mund të ngrejë në dërrasë shumë nxënës

njëkohësisht). Sapo mësuesi doli një moment nga klasa, disa prej nxënësve u kthyen dhe filluan të komunikojnë me shokun që kishin në krah, kurse të tjerët vazhduan punën. Këtë situatë mund ta paraqesim me një varg të tillë: `*><><><*`, ku një `*` tregon një nxënës që vazhdon punën, një `>` tregon një nxënës që po flet me shokun në të djathtë, dhe një `<` tregon një nxënës që po flet me shokun në të majtë. Sapo dëgjojnë mësuesin të hyjë përsëri, nxënësit që po flisnin kthehen nga frika në krahun e kundërt. Kur mësuesi shikon dy nxënës përball njëri-tjetrit ai mendon se ata po flisnin, kështu që i ndëshkon të dy. Gjeni sa dyshe nxënësish ndëshkohen prej mësuesit.

Referenca: https://www.codechef.com/problems/CHEFSTUD

Shembull

```
$ cat input.txt
4
><
*><*
><><
*><><><*

$ python3 prog.py < input.txt
0
0
1
2
```

Problemi 056

Kërkesa

Një lojë luhet në një shirit letre të ndarë në kuti, ku në çdo kuti është shënuar nga një numër natyror. Fillimisht lojtari i parë ngjyros një kuti (ku të dojë), pastaj lojtari i dytë mund të presë një kuti në skajet e shiritit (nqs është e pangjyrosur), pastaj lojtari i parë ngjyros një kuti tjetër që është ngjitur me një kuti të ngjyrosur më parë, pastaj lojtari i dytë pret një kuti tjetër të pangjyrosur, e kështu me rradhë. Pra lojtari i parë në fillim ngjyros ku të dojë, por pastaj duhet të ngjyros një kuti ngjitur me kutitë e ngjyrosura më parë. Kurse lojtari i dytë mund të presë një kuti në skajet e letrës, por jo nqs është e ngjyrosur.

Qëllimi i lojtarit të parë është që shuma e numrave në kutitë e ngjyrosura të jetë sa më e madhe. Bëni një program që gjen se sa janë pikët maksimale që mund të fitojë lojtari i parë në këtë lojë.

Referenca: https://codingcompetitions.withgoogle.com/kickstart/round/0000000000050ee2/000000000005118a

Shembull

```
$ cat input.txt
4
4
1332
4
9583
3
616
10
1029384756

$ python3 prog.py < input.txt
Case #1: 6
Case #2: 14
Case #3: 7
Case #4: 31
```

Në rastin e parë kemi katër numrat **1332**. Lojtari i parë mund të ngjyrosë dy të mesit dhe të fitojë 6 pikë.

Në rastin e dytë kemi numrat **9583**. Numri më i madh i pikëve fitohet nëse ngjyrosen dy kutitë e para.

Në rastin e tretë mund të ngjyroset fillimisht njëshi dhe pastaj një nga gjashtat.

Në rastin e katërt është e mundur të ngjyrosen **93847*, që japin dhe shumën maksimale 31.

Zgjidhja 1

```
T = int(input())
for t in range(1, T+1):
    N = int(input())
    W = input()
    s = (N + 1) // 2
    B = 0
    for i in range(N-s):
        b = 0
```

```
        for j in range(s):
            b += int(W[i+j])
        if b > B:
            B = b
    print("Case #{}: {}".format(t, B))
```

Sqarime

Po ta shikojmë me kujdes problemin dhe të bëjmë disa prova, mund të bindemi që lojtari i parë gjithmonë mund të ngjyrosë gjysmat e kutive. Bile, nqs numri N i kutive është tek, ai mund të ngjyrosë (N+1)//2, meqënëse luan i pari.

Po ashtu, nqs arsyetojmë pak, mund të bindemi që lojtari i parë mund të ngjyrosë ato kuti të njëpasnjëshme që dëshiron. P.sh. në shembullin e katërt loja mund të zhvillohet kështu. Në fillim ngjyroset kutia me numrin **8**. Pastaj, nqs pritet një kuti nga e djathta atere vazhdohet me ngjyrosjen e numrit **4**, përndryshe vazhdohet me ngjyrosjen e numrit **3**. Në këtë mënyrë mund të ngjyrosim ato kuti që duam pavarësisht se nga cila anë priten kutitë.

Meqënëse mund të ngjyrosim gjysmat e kutive, dhe mund të zgjedhim ato kuti të njëpasnjëshme që duam, atere problemi i gjetjes së shumës maksimale bëhet më i thjeshtë. Programi gjen në fillim shumën e gjysmës së parë të kutive, pastaj i zhvendos kutitë e ngjyrosura me një kuti në të djathtë dhe shikon nëse shuma e re është më e madhe se maksimumi i tanishëm, e kështu me rradhë.

Zgjidhja 2

```
T = int(input())
for t in range(1, T+1):
    N = int(input())
    W = input()
    s = (N + 1) // 2
    B = 0
    for i in range(s):
        B += int(W[i])
    b = B
    for i in range(N-s):
        b -= int(W[i])
        b += int(W[s+i])
        if b > B:
            B = b
    print("Case #{}: {}".format(t, B))
```

Sqarime

Zgjidhja e parë ka dy cikle brenda njëri-tjetrit, ku secili cikël përsëritet N/2 herë. Kështu që kompleksiteti i saj është i rendit $O(N^2)$

Kurse zgjidhja e dytë ka dy cikle që përsëriten N/2 herë, por që nuk janë brenda njëri-tjetrit. Kështu që kompleksiteti i saj është i rendit $O(N)$.

Zgjidhja e dytë është shumë më e shpejtë se zgjidhja e parë.

Detyra

Një lojë luhet në një tabelë drejtkëndore me **N** rreshta dhe **M** shtylla. Fillimisht kutitë e tabelës janë të pangjyrosura. Lojtari fillon ti ngjyrosë një nga një, dhe për çdo kuti që ngjyros merr aq pikë sa fqinjë të ngjyrosur ka kjo kuti (fqinjët e një kutie janë ato kuti që kanë një brinjë të përbashkët me të). Shuma e këtyre pikëve janë pikët që fitohen nga kjo lojë. Bëni një program që gjen se sa janë pikët maksimale që mund të fitohen në këtë lojë.

Referenca: https://www.codechef.com/problems/OMWG

Shembull

```
$ cat input.txt
1
2 2

$ python3 prog.py < input.txt
4
```

Në një tabelë me 2 rreshta dhe 2 shtylla, numri maksimal i pikëve që mund të fitohen është 4.

Problemi 057

Kërkesa

Në spektaklin "Kujt ia mban të jetë milioner?" ("Who dares to be a millionaire?") bëhen N pyetje, ku çdo pyetje ka 26 alternativa që shënohen me shkronjat A-Z, nga të cilat vetëm njëra është e saktë. Pjesëmarrësit i dinë pyetjet që do bëhen dhe e kanë ndarë mëndjen për përgjigjet që do japin, por pa e ditur nëse këto përgjigje janë të sakta.

Loja bëhet kështu. Në fillim pyetjet përzihen në mënyrë që rradha e tyre të jetë e rastësishme. Pastaj pyetjet i drejtohen një nga një pjesëmarrësit. Nëse përgjigja është e saktë, vazhdohet me pyetjen tjetër, përndryshe loja mbaron.

Lojtari shpërblehet në këtë mënyrë. Nëse s'ka dhënë asnjë përgjigje të saktë merr shpërblimin W_0, nëse ka dhënë 1 përgjigje të saktë shpërblimi është W_1, për 2 përgjigje të sakta shpërblimi është W_2, e kështu me rradhë. Vetëm se nuk është e thënë që $W_{i+1} > W_i$ (mesa duket kjo lojë do jetë shpikur nga ndonjë i lojtur).

Nqs dimë përgjigjet e sakta për çdo pyetje, përgjigjet e një lojtari, dhe vlerat e shpërblimeve, sa është shpërblimi më i madh që mund të fitojë ky lojtar?

Referenca: https://www.codechef.com/problems/WDTBAM

Shembull

```
$ cat input.txt
3
5
ABCDE
EBCDA
0 10 20 30 40 50
4
CHEF
QUIZ
4 3 2 1 0
8
ABBABAAB
ABABABAB
100 100 100 100 100 100 100 100 100

$ python3 prog.py < input.txt
30
4
100
```

Në rastin e parë, nëse pyetjet renditen sipas rradhës (2, 3, 4, 5, 1), atere lojtari do jepë 3 përgjigje të sakta dhe do fitojë 30 pikë.

Në rastin e dytë, sido që të renditen pyetjet lojtari nuk ka asnjë përgjigje të saktë, kështu që do fitojë W_0.

Në rastin e tretë, sado përgjigje të sakta të japë, nuk ka për të fituar veçse 100 pikë.

Zgjidhja 1

```
T = int(input())
for _ in range(T):
    n = int(input())
    Q = input()
    A = input()
    W = list(map(int, input().split()))
    c = 0
    for i in range(n):
        if Q[i] == A[i]:
            c += 1
    if c == n:
        print(W[n])
    else:
        wmax = W[0]
        for i in range(c+1):
            if W[i] > wmax:
                wmax = W[i]
        print(wmax)
```

Sqarime

Fillimisht gjejmë se sa është numri i përgjigjeve të sakta (**c** – correct). Nëse të gjitha përgjigjet janë të sakta, atere shpërblimi i vetëm i mundshëm është $W[n]$, përndryshe të gjitha shpërblimet janë të mundshme, nga W_0 te W_c. Dhe meqënëse W nuk është e thënë të jetë e renditur në rendin rritës, duhet të gjejmë se cila është vlera më e madhe nga këto.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    Q = input()
    A = input()
    W = [int(w) for w in input().split()]
    c = 0
    for i in range(n):
        if Q[i] == A[i]:
            c += 1
    print(W[n]) if c==n else print(max(W[0:c+1]))
```

Sqarime

Është e njëjta zgjidhje por pak më ndryshe. `W[0:c+1]` është një nënlistë e listës `W`, me elementet nga **0** deri në **c**. Kurse funksioni `max()` gjen vlerën më të madhe të një liste.

Detyra

Bëni një program që kontrollon nëse një varg numrash plotëson këto kushte:

- Ka një qendër (dmth numri i elementeve në të majtë të qendrës është i barabartë me numrin e elementeve në të djathtë).
- Numri i parë dhe i fundit është **1**.
- Duke u nisur nga qëndra, numrat në të majtë dhe në të djathtë vijnë duke zbritur me nga 1.

Referenca: https://www.codechef.com/problems/TEMPLELA

Shembull

```
$ cat input.txt
7
5
1 2 3 2 1
7
2 3 4 5 4 3 2
5
1 2 3 4 3
5
1 3 5 3 1
7
1 2 3 4 3 2 1
4
1 2 3 2
4
1 2 2 1

$ python3 prog.py < input.txt
yes
no
no
no
yes
no
no
```

Problemi 058

Kërkesa

Në një listë me numra natyrorë përkufizojmë funksionin **prefixSum(i)** si shumën e numrave të listës nga fillimi e deri në pozicionin **i**, dhe funksionin **suffixSum(i)** si shumën e numrave të listës nga pozicioni **i** e deri në fund.

Gjeni indeksin *më të vogël* **i** për të cilën shuma **prefixSum(i) + suffixSum(i)** ka vlerën më të vogël.

Referenca: https://www.codechef.com/problems/CHEFSUM

Shembull

```
$ cat input.txt
2
3
1 2 3
4
2 1 3 1

$ python3 prog.py < input.txt
1
2
```

1. Po të llogarisim shumat, duket që i-ja që jep vlerën më të vogël është **1**: `prefixSum(1) + suffixSum(1) = 1 + 6 = 7` `prefixSum(2) + suffixSum(2) = 3 + 5 = 8` `prefixSum(3) + suffixSum(3) = 6 + 3 = 9`
2. Vlera më e vogël e shumës është **8**, dhe vlera më e vogël e **i**-së që jep shumën më të vogël është **2**: `prefixSum(1) + suffixSum(1) = 2 + 7 = 9` `prefixSum(2) + suffixSum(2) = 3 + 5 = 8` `prefixSum(3) + suffixSum(3) = 6 + 4 = 10` `prefixSum(4) + suffixSum(4) = 7 + 1 = 8`

Zgjidhja 1

```python
def prefixSum(L, i):
    s = 0
    j = 0
    while j <= i:
        s += L[j]
        j += 1
    return s
```

```
def suffixSum(L, i):
    s = 0
    j = i
    while j < len(L):
        s += L[j]
        j += 1
    return s

import math

T = int(input())
for t in range(T):
    n = int(input())
    L = [int(i) for i in input().split()]

    # gjej vlerën më të vogël të shumës
    s_min = math.inf
    for i in range(n):
        s = prefixSum(L, i) + suffixSum(L, i)
        if s < s_min:
            s_min = s

    # gjej i-në më të vogël që jep vlerën më të vogël të shumës
    for i in range(n):
        s = prefixSum(L, i) + suffixSum(L, i)
        if s == s_min:
            print(i+1)
            break
```

Sqarime

Zgjidhja është me dy hapa, në fillim gjejmë vlerën më të vogël të shumës, pastaj gjejmë i-në më të vogël që jep shumën më të vogël.

Kjo zgjidhje e ka kohën $O(N^2)$ (pse?). Për vlera të vogla të N-së (p.sh. deri në 100) kjo është zgjidhje e pranueshme, por për vlera të mëdha është shumë e ngadaltë. Këtë mund ta verifikojmë edhe këtu: https://www.codechef.com/submit/CHEFSUM, ku kjo zgjidhje merr vetëm 20 pikë (nga 100 të mundshme). Kjo për shkak sepse kalon vetëm nëntestin me vlera të vogla të N-së, por jo atë me vlera të mëdha.

Zgjidhja 2

“‘python def prefixSum(L, i): s = 0 j = 0 while j <= i: s += L[j] j += 1 return s

def suffixSum(L, i): s = 0 j = i while j < len(L): s += L[j] j += 1 return s

import math

T = int(input()) for t in range(T): n = int(input()) L = [int(i) for i in input().split()] s_min = math.inf i_min = n i = n-1 while i >= 0: s = prefixSum(L, i) + suffixSum(L, i) if s <= s_min: s_min = s i_min = i i -= 1 print(i_min + 1) “‘

Sqarime

Kjo zgjidhje ka një përmirësim në lidhje me zgjidhjen e parë, sepse vlera më e vogël e shumës dhe vlera më e vogël e i-së gjenden njëkohësisht, me të njëjtin cikël. Pra në vend të dy cikleve tani kemi vetëm një. Kjo e shkurton përgjysëm kohën e punës së programit, por gjithsesi është krejtësisht e pafuqishme për të mposhtur rritjen kuadratike të $O(N^2)$. Këtë e tregon edhe testimi, ku përsëri kjo zgjidhje arrin të kalojë vetëm testin e vogël.

Zgjidhja 3

```
import math

T = int(input())
for t in range(T):
    n = int(input())
    L = [int(i) for i in input().split()]
    l_min = math.inf
    i_min = n
    i = n-1
    while i >= 0:
        if L[i] <= l_min:
            l_min = L[i]
            i_min = i
        i -= 1
    print(i_min + 1)
```

Sqarime

Kjo zgjidhje është e ndryshme nga dy të parat. Po ta mendojmë me kujdes, vëmë re se `prefixSum(L, i) + suffixSum(L, i) == sum(L) + L[i]`. Kjo do të thotë se vlera më të vogël e shumës merret për të njëjtën **i** që ka numri më i vogël i listës. Kështu

që nuk është nevoja ta llogarisim fare shumën por gjejmë direkt numrin më të vogël të listës.

Për të gjetur vlerën më të vogël të i-së, përsëri e bëjmë kontrollin duke filluar nga fundi.

Koha e kësaj zgjidhje është $O(N)$, sepse kemi vetëm një cikël. Ky është një përmirësim i ndjeshëm në krahasim me zgjidhjet e mësipërme, dhe kjo zgjidhje i kalon edhe testet e mëdha.

Zgjidhja 4

```
for _ in range(int(input())):
    input()
    L = [int(i) for i in input().split()]
    print(L.index(min(L)) + 1)
```

Sqarime

E njëjta zgjidhje si më sipër, por më e përmbledhur, duke përdorur funksionet e gatshme `min()` dhe `index()`.

Detyra

Ceni kishte një kuti me **N** numra në të: $A_1, A_2, ..., A_n$. Gjithashtu kishte vendosur edhe numrin **N** në krye, për të ditur sa numra janë. Pra kutia kishte gjithsej **N+1** numra. Por nga gëzimi i olimpiadës IOI ai filloi të kërcente me kutinë në xhep dhe numrat iu ngatërruan, dhe tani nuk e di më se cili nga **N+1** numrat është numri **N** dhe cilët janë numrat e tjerë. Atij i duhet të gjejë më të madhin e numrave të dhënë. A mund ta ndihmoni?

Referenca: https://www.codechef.com/problems/LOSTMAX

Shembull

```
$ cat input.txt
3
1 2 1
3 1 2 8
1 5 1 4 3 2

$ python3 prog.py < input.txt
1
```

```
8
4
```

Në rastin e parë, N=2 dhe numrat janë {1, 1}, kështu që më i madhi është 1.

Në rastin e dytë N=3 dhe numrat janë {1, 2, 8}. Më i madhi është 8.

Në rastin e tretë N=5 dhe numrat janë {1, 1, 4, 3, 2}. Më i madhi është 4.

Problemi 059

Kërkesa

Në një festë morën pjesë **N** djem dhe **M** vajza. Na jepet një matricë A me **N** rreshta dhe **M** shtylla, ku A_{ij} është **1** nëse djalit **i** i pëlqen vajza **j**, përndryshe është **0**. Vini re që nuk është e thënë që nëse djalit **x** i pëlqen vajza **y**, edhe vajzës **y** i pëlqen djali **x**.

Nqs dy djemve **x** dhe **y** u pëlqen vajza **z**, kjo quhet një përplasje. Gjeni numrin e përplasjeve të ndryshme në këtë festë. Vini re që rradha e djemve në një përplasje nuk ka rëndësi.

Referenca: https://www.codechef.com/problems/LCOLLIS

Shembull

```
$ cat input.txt
2
4 3
111
100
110
000
2 2
10
01

$ python3 prog.py < input.txt
4
0
```

Jepet numri i djemve the i vajzave **N** dhe **M**, dhe pastaj matrica e pëlqimeve me 0/1.

Në rastin e parë, tre djem e kanë qejf vajzën e parë, kështu që kemi 3 përplasje (i pari me të dytin, i dyti me të tretin, dhe i pari me të tretin). Djali i parë dhe i tretë pëlqejnë vajzën e dytë, kështu që kemi edhe një përplasje tjetër. Gjithsej 4 përplasje.

Në rastin e dytë kemi 2 djem dhe 2 vajza, dhe asnjë përplasje.

Zgjidhja

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    L = []
    for i in range(n):
        L.append(input())

    c = 0    # number of collisions
    for i in range(m):
        k = 0
        for j in range(n):
            if L[j][i] == '1':
                p += 1
        c += p * (p - 1) // 2
    print(c)
```

Sqarime

Matricën e mbajmë në një listë me rreshta. Për çdo vajzë, gjejmë se sa djem e pëlqejnë atë, dhe këtë e mbajmë në ndryshoren **p**. Numri i përplasjeve për çdo vajzë është sa numri i dysheve që mund të formohen nga **p**, i cili është **p*(p-1)/2**.

Detyra

Një makinë llogaritëse është me difekt. Kur mbledh dy numra ajo nuk e transferon tepricën, p.sh 12 + 9 sipas saj del 11. Bëni një program që merr 2 numra dhe nxjerr rrezultatin që do nxirrte kjo makinë llogaritëse për mbledhjen e tyre.

Referenca: https://www.codechef.com/problems/BUGCAL

Shembull

```
$ cat input.txt
2
12 9
25 25

$ python3 prog.py < input.txt
```

```
11
40
```

Problemi 060

Kërkesa

Alisa është zënë me Blertën këto kohët e fundit dhe s'do të ketë më punë me të. Ato kanë secila një grup me numra natyrorë. Numrat e Alisës janë $A_1, A_2, ..., A_N$, kurse numrat e Blertës janë $B_1, B_2, ..., B_M$. Tani Alisa do që të fshijë të gjithë numrat në grupin e saj që janë të përbashkët me numrat e Blertës. Sa është numri më i vogël i numrave që duhet të fshijë Alisa?

Referenca: https://www.codechef.com/problems/NOTINCOM

Shembull

```
$ cat input.txt
2
3 4
1 2 3
3 4 5 6
3 3
1 2 3
4 5 6

$ python3 prog.py < input.txt
1
0
```

Zgjidhja 1

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    A = list(map(int, input().split()))
    B = list(map(int, input().split()))
    nr = 0
    for a in A:
        if a in B:
            nr += 1
    print(nr)
```

Sqarime

Kjo zgjidhje është llogjikisht e thjeshtë: për çdo element të listës A kontrollojmë nëse ndodhet edhe në B. Kompleksiseti është i rendit $O(N * M)$, meqënëse kemi një cikël që përsëritet **N** herë, dhe brenda tij kërkojmë në një listë që ka **M** elementë.

Po ta testojmë: https://www.codechef.com/submit/NOTINCOM kjo zgjidhje kalon vetëm testin e vogël dhe merr vetëm gjysmat e pikëve.

Zgjidhja 2

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    A = list(map(int, input().split()))
    B = list(map(int, input().split()))
    A.sort()
    B.sort()
    nr = 0
    i = 0
    j = 0
    while i < n and j < m:
        if A[i] == B[j]:
            nr += 1
            i += 1
        elif A[i] < B[j]:
            i += 1
        else:
            j += 1
    print(nr)
```

Sqarime

Kjo zgjidhje duket pak më e ndërlikuar se e para, por është më e shpejtë. Kjo zgjidhje i kalon të dyja grupet e testimit: https://www.codechef.com/submit/NOTINCOM

Fillimisht renditen të dyja listat dhe pastaj kontrollohet për të parë se sa elemente janë të barabarta midis tyre. Renditjen e listave ne e bëjmë me funksione të gatshme, por një renditje e shpejtë zakonisht merr një kohë të rendit $O(N * ln(N))$. Kështu që në total, koha që i duhet programit është përafërsisht: $O(N * ln(N)) + O(M * ln(M) + N + M)$, ose përafërsisht $O(N * ln(N))$ (nqs supozojmë që $N \approx M$). Kjo kohë është më e mirë se koha e zgjidhjes së mëparshme $O(N^2)$ (nqs supozojmë që $N \approx M$).

Detyra

Nga rezultatet e një olimpiade mund të hamendësojmë nivelin e aftësive për secilin pjesëmarrës. Pjesëmarrësit mund të klasifikohen bazuar në numrin e problemave të zgjidhura në këtë mënyrë:

- 0 problema të zgjidhura: Beginner
- 1 problem të zgjidhur: Junnior Developper
- 2 problema të zgjidhura: Middle Developer
- 3 problema të zgjidhura: Senior Developer
- 4 problema të zgjidhura: Hacker
- 5 problema të zgjidhura: Jeff Dean

Bëni një program që merr rezultatet e pjesëmarrësve dhe nxjerr klasifikimin e tyre.

Referenca: https://www.codechef.com/problems/CCOOK

Shembull

```
$ cat input.txt
7
0 0 0 0 0
0 1 0 1 0
0 0 1 0 0
1 1 1 1 1
0 1 1 1 0
0 1 1 1 1
1 1 1 1 0

$ python3 prog.py < input.txt
Beginner
Middle Developer
Junior Developer
Jeff Dean
Senior Developer
Hacker
Hacker
```

8 Problemat 061-070

Problemi 061

Kërkesa

Çdo ditë Majko shkon në punë me autobus, ku pret një biletë. Çdo biletë ka një kod me shkronja latine. Majko beson se dita do ti shkojë mbarë nëse kodi i biletës ka vetëm 2 shkronja alternuese, si p.sh. **xyxyxy...** ku **x!=y**.

Bëni një program që merr kodin e biletës dhe nxjerr "YES" nëse është alternues, dhe "NO" në të kundërt.

Referenca: https://www.codechef.com/problems/TICKETS5

Shembull

```
$ cat input.txt
2
ABABAB
ABC

$ python3 prog.py < input.txt
YES
NO
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    S = input()
    if S[0] == S[1]:
        print('NO')
    else:
        for i in range(2, len(S)):
            if S[i] != S[i % 2]:
```

```
                print('NO')
                break
        else:
            print('YES')
```

Zgjidhja 2

```
T = int(input())
for t in range(T):
    S = input()
    if S[0] == S[1]:
        print('NO')
    else:
        i = 2
        while i < len(S):
            if S[i] != S[i - 2]:
                print('NO')
                break
            i += 1
        else:
            print('YES')
```

Detyra

Një ari i vogël polar ka qejf të hajë biskota dhe të pijë qumësht. Për këtë arsye shkon shpesh në kuzhinë. Ai qëndron në kuzhinë **N** minuta, dhe çdo minutë ose ha një biskotë, ose pi qumësht. Meqënëse biskotat janë shumë të ëmbla, prindërit e kanë këshilluar që sa herë të hajë një biskotë, në minutën tjetër duhet të pijë qumësht.

Na jepet sa minuta ka ndenjur në kuzhinë arushi dhe çfarë ka bërë çdo minutë: ka ngrënë biskotë apo ka pirë qumësht. Gjeni nëse arushi i ka zbatuar ose jo këshillat e prindërve, dmth që pas ngrënies së një biskote ka pirë qumësht.

Referenca: https://www.codechef.com/problems/COOMILK

Shembull

```
$ cat input.txt
4
7
cookie milk milk cookie milk cookie milk
```

```
5
cookie cookie milk milk milk
4
milk milk milk milk
1
cookie

$ python3 prog.py < input.txt
YES
NO
YES
NO
```

Problemi 062

Kërkesa

Kemi një listë me **N** numra natyrorë. Gjejmë shumën **S** të numrave të listës dhe pastaj shtojmë në listë një numër më të madh se **S**. Këtë veprim e përsërisim **K** herë, kështu që në fund lista do ketë **N+K** numra. Gjeni vlerën më të vogël të mundshme të numrit që do shtohet i fundit në listë, dhe printoni nëse është çift apo tek.

Referenca: https://www.codechef.com/problems/UTMOPR

Shembull

```
$ cat input.txt
2
2 3
5 7
3 1
5 2 3

$ python3 prog.py < input.txt
even
odd
```

Dy numrat e parë janë **N** dhe **K**, pastaj vijnë numrat e listës.

Zgjidhja 1

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    L = list(map(int, input().split()))
    while k > 0:
        s = sum(L)
        L.append(s + 1)
        k -= 1
    print('even') if L[-1] % 2 == 0 else print('odd')
```

Sqarime

Vlera më e vogël e mundshme është kur elementi i ri që shtohet në listë është gjithmonë 1 më i madh se shuma.

Kjo zgjidhje duket e thjeshtë por nuk është shumë e shpejtë. Në fakt ajo nuk e kalon testin te https://www.codechef.com/submit/UTMOPR sepse nuk përfundon brenda kohës së lejuar. Kompleksieti i saj është i rendit $O(K*N)$, sepse kemi një cikël që përsëritet **K** herë, dhe brenda tij bëjmë shumën e **N** elementeve të listës.

Zgjidhja 2

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    L = list(map(int, input().split()))
    s = sum(L)
    while k > 0:
        a = s + 1
        s += a
        k -= 1
    print('even') if a % 2 == 0 else print('odd')
```

Sqarime

Kjo zgjidhje është pak më e shpejtë se e para, sepse shumën nuk e llogarisim nga e para, por e llogarisim në mënyrë progresive (duke shtuar në shumë vetëm elementin e ri). Kompleksiteti i saj është i rendit $O(N + K)$, meqënëse gjejmë në fillim shumën e **N** elementëve të listës dhe pastaj kemi një cikël që përsëritet **K** herë. Megjithatë as kjo zgjidhje nuk e kalon testin sepse nuk është e shpejtë sa duhet.

Zgjidhja 3

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    L = list(map(int, input().split()))
    if k > 1:
        print('even')
    else:
        print('odd') if sum(L) % 2 == 0 else print('even')
```

Sqarime

Po ta shikojmë problemin me kujdes dhe të bëjmë disa prova, mund të vërejmë dhe të vërtetojmë që elementi i ri që shtohet në listë është gjithmonë çift, me përjashtim të rastit kur shuma e elementeve fillestare të listës është çift dhe **K** është 1.

Kjo zgjidhje është shumë e shpejtë dhe e kalon testin. Kompleksiteti i saj në rastin më të keq është $O(N)$ sepse gjejmë vetëm një herë shumën e elementëve të listës.

Detyra

Një bashkësi me numra natyrorë quhet *e mirë* nëse nuk ekzistojnë në të tre elemente të ndryshëm **a**, **b**, **c** të tillë që $\mathbf{a} + \mathbf{b} = \mathbf{c}$.

Bëni një program që merr një numër **n** (nga 1 deri në 100) dhe nxjerr një bashkësi të mirë me **n** elementë, ku çdo element i bashkësisë mund të jetë një numër nga 1 deri në 500.

Referenca: https://www.codechef.com/problems/GOODSET

Shembull

```
$ cat input.txt
5
1
2
3
4
5

$ python3 prog.py < input.txt
7
1 2
```

```
1 2 4
1 2 4 16
3 2 15 6 10
```

Problemi 063

Kërkesa

Bëni një program që kontrollon nëse një varg numrash plotëson këto kushte:

- Ka një qendër (dmth numri i elementeve në të majtë të qendrës është i barabartë me numrin e elementeve në të djathtë).
- Numri i parë dhe i fundit është **1**.
- Duke u nisur nga qëndra, numrat në të majtë dhe në të djathtë vijnë duke zbritur me nga 1.

Referenca: https://www.codechef.com/problems/TEMPLELA

Shembull

```
$ cat input.txt
7
5
1 2 3 2 1
7
2 3 4 5 4 3 2
5
1 2 3 4 3
5
1 3 5 3 1
7
1 2 3 4 3 2 1
4
1 2 3 2
4
1 2 2 1

$ python3 prog.py < input.txt
yes
no
no
no
yes
```

```
no
no
```

Zgjidhja 1

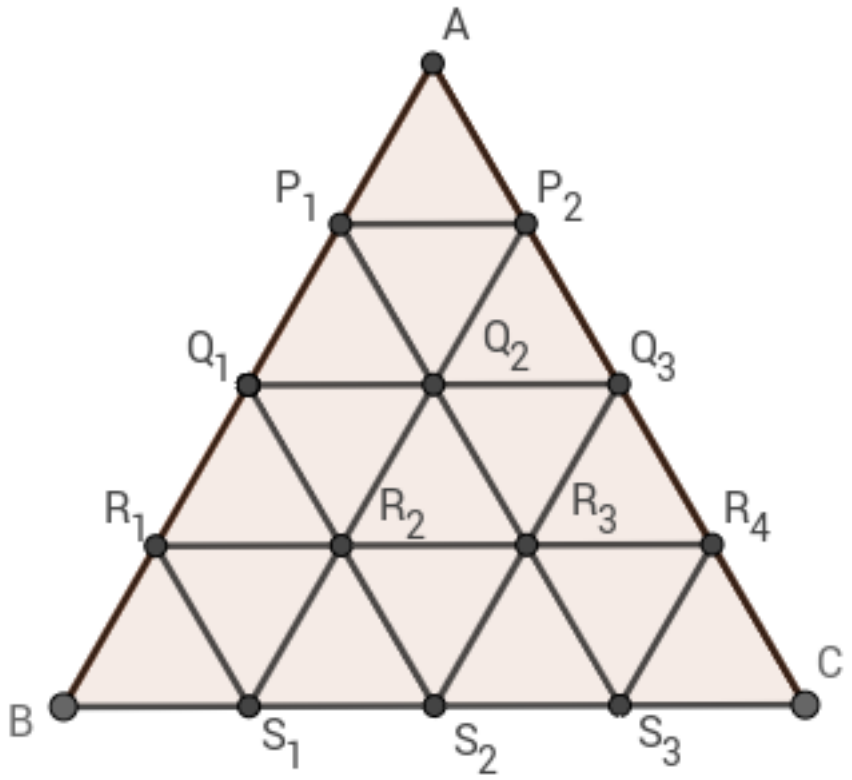
```
for _ in range(int(input())):
    n = int(input())
    L = list(map(int, input().split()))
    if (n % 2 == 0) or (L[0] != 1) or (L[n-1] != 1):
        print('no')
    else:
        i = 1
        while i <= n // 2:
            if (L[i] != L[i-1] + 1) or (L[n-i] + 1 != L[n-i-1]):
                print('no')
                break
            i += 1
        else:
            print('yes')
```

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    L = list(map(int, input().split()))
    if (n % 2 == 0) or (L[0] != 1) or (L[n-1] != 1):
        print('no')
    else:
        i = 0
        while i < n // 2:
            if (L[i] + 1 != L[i+1]) or (L[n-1 - i] + 1 != L[n-2 - i]):
                print('no')
                break
            i += 1
        else:
            print('yes')
```

Detyra

Numrat natyrorë vendosen në nyjet e një grafi që ka formën e shinave të trenit, si në figurë:



Bëni një program që merr dy numra dhe gjen nëse ka brinjë midis këtyre dy nyjeve.

Referenca: https://www.codechef.com/problems/BRLADDER

Shembull

```
$ cat input.txt
7
1 4
4 3
5 4
10 12
1 3
999999999 1000000000
```

```
17 2384823

$ python3 prog.py < input.txt
NO
YES
NO
YES
YES
YES
NO
```

Problemi 064

Kërkesa

Një makinë llogaritëse është me difekt. Kur mbledh dy numra ajo nuk e transferon tepricën, p.sh 12 + 9 sipas saj del 11. Bëni një program që merr 2 numra dhe nxjerr rrezultatin që do nxirrte kjo makinë llogaritëse për mbledhjen e tyre.

Referenca: https://www.codechef.com/problems/BUGCAL

Shembull

```
$ cat input.txt
2
12 9
25 25

$ python3 prog.py < input.txt
11
40
```

Zgjidhja 1

```
for _ in range(int(input())):
    a, b = map(int, input().split())
    c = 0
    p = 1
    while a > 0 or b > 0:
        d1 = a % 10
        d2 = b % 10
```

```
        d = (d1 + d2) % 10
        c += p * d
        a //= 10
        b //= 10
        p *= 10
    print(c)
```

Sqarime

Marrim shifrën e fundit të dy numrave, duke gjetur mbetjen e pjesëtimit me 10. I mbledhim këto dhe gjejmë mbetjen e pjesëtimit me 10, e cila është njëra nga shifrat e numrit të ri. Të ndryshorja **p** mbajmë fuqitë e dhjetës (1, 10, 100, 1000, etj.) të cilat tregojnë edhe se çfarë është kjo shifra e numrit të ri që sapo gjetëm (njëshe, dhjetëshe, qindëshe, etj.) Kështu që e shumëzojmë shifrën me **p** dhe ia shtojmë numrit. Ndërkohë marrim edhe herësin e pjesëtimit me dhjetë për dy numrat **a** dhe **b**, që cikli të vazhdojë me shifrën tjetër.

Zgjidhja 2

```
for _ in range(int(input())):
    a, b = map(int, input().split())
    l = []
    while a > 0 or b > 0:
        d1 = a % 10
        d2 = b % 10
        l.append((d1 + d2) % 10)
        a //= 10
        b //= 10
    l.reverse()
    print(''.join(map(str, l)))
```

Sqarime

Pak a shumë e njëjta zgjidhje, vetëm se shifrat e numrit të ri mbajmë në një listë dhe pastaj i bashkojmë si një varg (string).

Detyra

Në një listë me **N** numra natyrorë mund të shtojmë edhe **K** numra të tjerë sipas dëshirës, dhe pastaj gjejmë të mesmen e të gjithë numrave. E mesmja e një liste është elementi

që ndodhet në mesin e listës, pasi ti kemi renditur. P.sh. e mesmja e [**2, 1, 5, 2, 4**] është **2**, kurse e mesmja e [**3, 3, 1, 3, 3**] është 3.

Gjeni se sa është vlera më e madhe e të mesmes që mund të merret në këtë mënyrë. Na jepet gjithashtu që $\mathbf{K} < \mathbf{N}$ dhe që $\mathbf{N} + \mathbf{K}$ është numër tek.

Referenca: https://www.codechef.com/problems/CK87MEDI

Shembull

```
$ cat input.txt
3
2 1
4 7
4 3
9 2 8 6
5 2
6 1 1 1 1

$ python3 prog.py < input.txt
7
9
1
```

Na jepen numrat **N** dhe **K**, dhe pastaj N numrat e listës.

Në rastin e parë, një nga zgjidhjet e mundshme është të shtojmë 9, duke formuar listën [4, 7, 9], e mesmja e të cilës është 7.

Në rastin e tretë, sido që të jenë 2 numrat që do shtohen, e mesmja do jetë gjithmonë 1.

Problemi 065

Kërkesa

Një bashkësi me numra natyrorë quhet *e mirë* nëse nuk ekzistojnë në të tre elemente të ndryshëm **a**, **b**, **c** të tillë që $\mathbf{a} + \mathbf{b} = \mathbf{c}$.

Bëni një program që merr një numër **n** (nga 1 deri në 100) dhe nxjerr një bashkësi të mirë me **n** elementë, ku çdo element i bashkësisë mund të jetë një numër nga 1 deri në 500.

Referenca: https://www.codechef.com/problems/GOODSET

Shembull

```
$ cat input.txt
5
1
2
3
4
5

$ python3 prog.py < input.txt
7
1 2
1 2 4
1 2 4 16
3 2 15 6 10
```

Zgjidhja 1

```python
for _ in range(int(input())):
    n = int(input())
    s = []
    for i in range(n):
        s.append(500 - i)
    print(*s)
```

Sqarime

Mqs bshkësia nuk mund të ketë më shumë se 100 numra, dhe këta numra mund të jenë deri në 500, mund ti zgjedhim numrat nga 400 deri në 500, dhe jemi të sigurtë që shuma e tyre do jetë më e madhe se 500, kështu që nuk do jetë në bashkësi.

Zgjidhja 2

```python
for _ in range(int(input())):
    n = int(input())
    s = []
    for i in range(n):
        s.append(250 + i)
    print(*s)
```

Sqarime

Pak a shumë njësoj si zgjidhja e parë, vetëm se numrat zgjidhen duke filluar nga 250 e lartë.

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    s = []
    for i in range(n):
        s.append(2*i + 1)
    print(*s)
```

Sqarime

Nqs zgjedhim në bashkësi vetëm numra tek, shuma e tyre do jetë numër çift, kështu që është e garantuar që nuk do jetë në bashkësi.

Detyra

Një bankomat (ATM) ka brënda **K** para. Në rradhë janë **N** klientë, ku secili do që të tërheqë A_i para. Nëse bankomati ka gjëndje aq sa kërkon klienti ose më shumë, ai ia jep lekët klientit, përndryshe nxjerr një mesazh gabimi dhe klienti largohet pa marrë lekë. Gjeni për secilin klient nëse i ka marrë lekët ose jo.

Referenca: https://www.codechef.com/problems/ATM2

Shembull

```
$ cat input.txt
2
5 10
3 5 3 2 1
4 6
10 8 6 4

$ python3 prog.py < input.txt
11010
0010
```

Kur klienti i merr lekët shënohet me **1**, përndryshe shënohet me **0**.

Në rastin e parë janë 5 klientë dhe bankomati ka 10 lekë. Pasi merr lekët klienti i parë dhe i dytë, në bankomat mbeten 2 lekë, të cilat janë të pamjaftueshme për ti dhënë lekët klientit të tretë, por mjaftojnë për ti dhënë lekët klientit të katërt.

Në rastin e dytë janë 4 klientë dhe bankomati ka 6 lekë. Dy klientët e parë nuk mund ti tërheqin lekët, por i treti po. Pastaj nuk ngelet më asnjë lekë, kështu që as klienti i katërt nuk mund ti tërheqë lekët.

Problemi 066

Kërkesa

Është një numër magjik me 16 letra ku letrat vendosen në një tabelë 4x4 dhe spektatorit i kërkohet të zgjdhë një letër dhe të thotë rreshtin në të cilin ndodhet. Pastaj "magjistari" i mbledh letrat, i pret, dhe i shpërndan përsëri në një tabelë 4x4. Spektatori thotë përsëri rreshtin në të cilin ndodhet letra që ka menduar, dhe magjistari gjen se cilën letër ka menduar.

Letrat i shënojmë me numra nga 1 deri në 16, dhe na jepet vendosja e letrave në fillim dhe rreshti që ka zgjedhur spektatori, si dhe vendosja e letrave herën e dytë dhe rreshti që ka zgjedhur spektatori. Bëni një program që gjen se cila ka qenë letra e menduar prej spektatorit. Nqs ju del se ka më shumë se një të tillë, i bie që magjistari ka bërë ndonjë gabim, kështu që programi duhet të nxjerrë "Bad magician!". Nqs nuk del asnjë letër, i bie që spektatori duhet të jetë ngatërruar ose ka gënjyer, kështu që programi duhet të nxjerrë "Volunteer cheated!",

Referenca: https://code.google.com/codejam/contest/2974486/dashboard#s=p0

Shembull

```
$ cat input.txt
3
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
3
1 2 5 4
3 11 6 15
9 10 7 12
13 14 8 16
```

```
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
3
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16

$ python3 prog.py < input.txt
Case #1: 7
Case #2: Bad magician!
Case #3: Volunteer cheated!
```

Në rastin e parë letra e menduar ndodhet në rreshtin e dytë, dhe pastaj në rreshtin e tretë, kështu që i bie që të jetë letra 7.

Zgjidhja

```python
for _ in range(int(input())):
    # lexo rreshtin e pare dhe tabelen e pare
    r1 = int(input())
    M1 = []
    for _ in range(4):
        M1.append([int(i) for i in input().split()])
    # lexo rreshtin e dyte dhe tabelen e dyte
    r2 = int(input())
    M2 = []
    for _ in range(4):
        M2.append([int(i) for i in input().split()])
```

```
    # merr rreshtat e perzgjedhur dhe gjej prerjen e tyre
    R1 = M1[r1 - 1]
    R2 = M2[r2 - 1]
    I = []
    for i in R1:
        if i in R2:
            I.append(i)
    # jep pergjigjen ne varesi te prerjes
    if len(I) > 2:
        print('Bad magician!')
    elif len(I) == 0:
        print('Volunteer cheated!')
    else:
        print(I[0])
```

Sqarime

Kemi shumë topa pinpongu të bardhë dhe të zinj dhe kemi krijuar dy rreshta me gjatësi **N** përkrah njëri-tjetrit. Këta rreshta i shënojmë si vargjet **X** dhe **Y**, të përbërë prej shkronjave **W** and **B**, ku **W** shënon një top të bardhë (white) kurse **B** një top të zi (black).

Tani duam të krijojmë një rresht të tretë **Z** në mënyrë që **ham(X, Z) + ham(Y, Z)** të jetë sa më e madhe. Funksioni **ham(A, B)** është Hamming Distance midis vargjeve **A** dhe **B** me të njëjtën gjatësi. Kjo distancë midis dy vargjeve është numri i pozicioneve ku këto vargje kanë shkronja të ndryshme. P.sh. **ham('WBB', 'BBW')** është **2**, meqënëse shkronjat e para dhe të treta janë të ndrushme.

Meqënëse mund të ketë zgjidhje të ndryshme, programi duhet të nxjerrë atë zgjidhje që është alfabetikisht më e vogël. P.sh. shkronja **B** është më e vogël se shkronja **W** sepse ndodhet përpara saj në alfabet.

Referenca: https://www.codechef.com/problems/ACBALL

Shembull

```
$ cat input.txt
1
WBWB
WBBB

$ python3 prog.py < input.txt
BWBW
```

Problemi 067

Kërkesa

Cookie Clicker është një lojë e ndërtuar me JavaScript ku lojtarët klikojnë te figura e një biskote gjigande. Duke klikuar aty fitojnë biskota. Ata mund ti shpenzojnë këto biskota për të ndërtuar punishte. Këto punishte i ndihmojnë që të bëjnë edhe më shumë biskota.

Në këtë problemë ju filloni me 0 biskota. Ju fitoni 2 biskota çdo sekondë duke klikuar te biskota gjigande. Nqs keni të paktën **C** biskota ju mund të bleni një punishte, e cila ju kushton **C** biskota, por ju ndihmon që të fitoni **F** biskota më tepër në çdo sekondë.

Kur të keni fituar **X** biskota (pa ato që keni shpenzuar për të blerë punishte) ju fitoni. Gjeni sa kohë do tju duhet për të fituar, duke përdorur strategjinë më të mirë të mundshme (dmth gjeni kohën më të shkurtër të mundshme për të fituar).

Referenca: https://code.google.com/codejam/contest/dashboard?c=2974486#s=p1

Shembull

```
$ cat input.txt
4
30.0 1.0 2.0
30.0 2.0 100.0
30.50000 3.14159 1999.19990
500.0 4.0 2000.0

$ python3 prog.py < input.txt
Case #1: 1.0000000
Case #2: 39.1666667
Case #3: 63.9680013
Case #4: 526.1904762
```

Jepen numrat **C**, **F** dhe **X**.

Në rastin e fundit, **C**=500, **F**=4 dhe **X**=2000. Strategjia më e mirë e mundshme do ishte kështu: 1. Ju filloni me 0 biskota, duke prodhuar 2 biskota në sekondë. 1. Pas **250** sekondash do keni **C**=500 biskota dhe mund të blini një punishte që prodhon **F**=4 biskota në sekondë. 1. Pasi të blini punishten ju do keni 0 biskota dhe mund të prodhoni 6 biskota në sek. 1. Punishtja tjetër do tju kushtojë 500 biskota dhe mund ta bleni pas **83.3333333** sekondash. 1. Pasi ta keni blerë punishten e dytë do keni 0 biskota, por prodhimi i biskotave në sekondë shkon në 10. 1. Një punishte tjetër do tju kushtojë 500 biskota, të cilën mund ta blini pas **50** sekondash. 1. Pasi të blini punishten e tretë do keni 0 biskota, por mund të prodhoni 14 biskota në sekondë. 1. Për të blerë një punishte

tjetër do tju duhen 500 biskota, por në fakt është më mirë që mos ta blini atë dhe të prisni deri sa të bëni **X**=2000 biskota, për të cilën ju duhen afërsisht **142.8571429** sekonda. 1. Koha totale për të fituar është: **250 + 83.3333333 + 50 + 142.8571429 = 526.1904762** sekonda.

Vini re që prodhimi i biskotave bëhet në mënyrë të vazhdueshme (dmth pas 0.1 sekondave të para të lojës ju do keni prodhuar 0.2 biskota).

Zgjidhja

```
for i in range(int(input())):
    c, f, x = map(float, input().split())
    time = 0.0                    # total time spent
    r = 2.0                       # rate of getting cookies
    while True:
        t1 = x/r                  # time to win without rate increase
        t2 = c/r + x/(r + f)      # time to win with a rate increase
        if t1 <= t2:              # no need to increase the rate again
            print("Case #{}: {}".format(i+1, time + t1))
            break
        else:                     # increate the rate once more
            time += c/r
            r += f
```

Sqarime

Sa kohë që nuk janë bërë akoma **C** biskota (që duhen për të blerë një punishte) ne vetëm mund të vazhdojmë prodhimin, se nuk kemi çtë bëjmë tjetër. Kur të kemi **C** biskota duhet të vendosim nëse është më mirë të blejmë një punishte tjetër apo të vazhdojmë prodhimin me këto punishte që kemi. Këtë vendim e marrim duke llogaritur kohën që duhet për të fituar pa punishte tjetër dhe kohën që duhet për të fitur duke blerë një punishte tjetër, dhe duke parë se cila prej tyre është më e shkurtër.

Detyra

Në një rrugë të drejtë dhe të ngushtë po lëvizin disa makina, të cilat kanë shpejtësi maksimale të ndryshme. Nqs rrugën përpara do ta kishin bosh, makinat do preferonin që të ecnin me shpejtësinë e tyre maksimale. Por nqs kanë përpara një makinë më të ngadaltë, makinat janë të detyruara ta ulin shpejtësinë, meqënëse nuk mund të parakalojnë, dhe të ecin njëra-pas-tjetrës vargan.

Bëni një program që merr shpejtësitë e makinave dhe gjen se sa prej tyre po ecin me shpejtësi maksimale. Mund të supozojmë se rruga është shumë e gjatë dhe makinat kanë qenë duke ecur në të për një kohë të gjatë.

Referenca: https://www.codechef.com/problems/CARVANS

Shembull

```
$ cat input.txt
3
1
10
3
8 3 6
5
4 5 1 2 3

$ python3 prog.py < input.txt
1
2
2
```

Makinat po lëvizin nga e djathta në të majtë.

Në rastin e dytë, makina me shpejtësi 6 ndodhet pas asaj me shpejtësi 3, kështu që nuk mund të ecë me shpejtësi maksimale.

Në rastin e tretë, makina me shpejtësi 5 ndodhet pas asaj me shpejtësi 4, kështu që nuk mund të ecë me shpeltësi maksimale. Po ashtu, makinat 2 dhe 3 ndodhen pas asaj me shpejtësi 1, kështu janë futur në vargan. Me shpejtësinë e tyre maksimale ecin vetëm makinat 4 dhe 1.

Problemi 068

Kërkesa

Le ta quajmë **tidy** (të rregullt) një numër që i ka shifrat të renditura në rendin jo-zbritës. P.sh. numrat 8, 123, 555, 224488 janë **tidy**, kurse numrat 20, 321, 495, 999990 nuk janë. Gjeni numrin më të vogël **tidy** që është më i madh ose i barabartë me një numër të dhënë **N**.

Referenca:

Shembull

```
$ cat input.txt
4
132
1000
7
9000000000000000000

$ python3 prog.py < input.txt
Case #1: 133
Case #2: 1111
Case #3: 7
Case #4: 999999999999999999
```

Zgjidhja 1

```python
def tidy(n):
    d = n % 10
    while n > 0:
        if n % 10 > d:
            return False
        d = n % 10
        n //= 10
    return True

for t in range(int(input())):
    N = int(input())
    while not tidy(N):
        N += 1
    print('Case #{}: {}'.format(t+1, N))
```

Sqarime

Kemi një funksion që kontrollon nëse një numër i dhënë është tidy, pastaj vazhdojmë ta rrisim numrin deri sa të gjejmë një që është tidy.

Kjo zgjidhje është llogjikisht e thjeshtë, por për numra të mëdhenj (p.sh. për rastin e fundit) vonohet shumë. Duhet gjetur një zgjidhje më e shpejtë.

Zgjidhja 2

```python
for t in range(int(input())):
    D = [int(d) for d in list(input())]
    i = 1
    while i < len(D) and D[i] >= D[i-1]:
        i += 1
    while i < len(D):
        D[i] = D[i-1]
        i += 1
    print('Case #', t+1, ': ', *D, sep='')
```

Sqarime

Kontrollojmë shifrat e numrit të dhënë deri sa të gjejmë një vend ku rregulli prishet. Që aty e deri në fund të numrit i bëjmë shifrat të barabarta me shifrën e fundit që ishte në rregull. Ky numër do jetë tidy dhe më i vogli i mundshëm nga të gjithë numrat më të mëdhenj (ose të barabartë) se numri i dhënë.

Kjo zgjidhje është shumë më e shpejtë se zgjidhja e parë. Edhe për numra shumë të mëdhenj (p.sh. rasti i fundit) jep përgjigje të menjëhershme.

Detyra

Le ta quajmë **tidy** (të rregullt) një numër që i ka shifrat të renditura në rendin jo-zbritës. P.sh. numrat 8, 123, 555, 224488 janë **tidy**, kurse numrat 20, 321, 495, 999990 nuk janë.

Nqs numërojmë numrat nga 1 në **N**, cili ka qenë numri i fundit **tidy** që kemi numëruar? Ose me fjalë të tjera, gjeni numrin më të madh **tidy** që është më i vogël (ose i barabartë) se një numër i dhënë **N**.

Referenca: https://code.google.com/codejam/contest/3264486/dashboard#s=p1&a=1

Shembull

```
$ cat input.txt
4
132
1000
7
111111111111111110

$ python3 prog.py < input.txt
```

```
Case #1: 129
Case #2: 999
Case #3: 7
Case #4: 99999999999999999
```

Problemi 069

Kërkesa

Regjimi ditor i Çufos është shumë i thjeshtë, në fillim të ditës ai gatuan, pastaj ha, dhe në fund shkon të flejë. Çufoja ka instaluar disa pajisje smart-home, midis të cilave edhe disa kamera, të cilat në mënyrë periodike shënojnë veprimin që është duke bërë Çufoja në atë moment. Kur Çufoja është duke gatuar sistemi shënon një '**C**' (cooking), kur është duke ngrënë shënon një '**E**' (eating), dhe kur është duke fjetur shënon një '**S**' (sleeping). Duke parë shënimet e sistemit për një ditë, gjeni nëse ato mund të jenë të sakta ose jo.

Referenca: https://www.codechef.com/problems/CHEFROUT

Shembull

```
$ cat input.txt
6
CES
CCCEESS
CS
CCC
SC
ECCC

$ python3 prog.py < input.txt
yes
yes
yes
yes
no
no
```

1. Është e qartë që rasti i parë është i saktë.
2. Në rastin e dytë, largësia midis rregjistrimeve mund të ketë qenë më e vogël, kështu që të njëjtin veprim e ka rregjistruar disa herë, por prapë shënimet janë të sakta.

3. Në rastin e tretë, largësia midis rregjistrimeve mund të ketë qenë shumë e madhe, kështu që ngrënien nuk e ka kapur fare, por prapë shënimet mund të jenë të rregullta.
4. Në rastin e katërt intervalet e rregjistrimit kanë qenë të tilla që ngrënien e ka kapur disa herë, kurse veprimet e tjera nuk i ka kapur fare, megjithatë shënimet mund të jenë të rregullta.
5. Në rastin e pestë shënimet nuk mund të jenë të sakta, sepse nuk ka mundësi që të ketë fjetur njëherë dhe pastaj të ketë gatuar.
6. Në rastin e fundit shënimet nuk mund të jenë të sakta, sepse nuk ka mundësi që të ketë ngrënë njëherë dhe pastaj të ketë gatuar.

Zgjidhja 1

```
for _ in range(int(input())):
    s = input()
    l = [s[0]]
    for i in range(1, len(s)):
        if s[i] != s[i-1]:
            l.append(s[i])
    if len(l)==3 and l[0]=='C' and l[1]=='E' and l[2]=='S':
        print('yes')
    elif len(l)==2 and ((l[0]=='C' and l[1]=='E') or (l[0]=='C' and l[1]=='S') or (l[0]=='E' a
        print('yes')
    elif len(l) == 1:
        print('yes')
    else:
        print('no')
```

Sqarime

Në fillim e bëjmë kompakt vargun, duke mbajtur vetëm një kopje të shkronjave që përsëriten, pastaj e kontrollojmë.

Zgjidhja 2

```
for _ in range(int(input())):
    s = input()
    l = [s[0]]
    for i in range(1, len(s)):
        if s[i] != s[i-1]:
```

```
            l.append(s[i])
    s1 = ''.join(l)
    if len(l)==1 or s1=='CE' or s1=='CS' or s1=='ES' or s1=='CES':
        print('yes')
    else:
        print('no')
```

Sqarime

Kontrolli mund të bëhet më kollaj nëse shkronjat i bashkojmë në një varg.

Zgjidhja 3

```
for _ in range(int(input())):
    L = list(input())
    for i in range(len(L)):
        if L[i] == 'C':
            L[i] = 1
        elif L[i] == 'E':
            L[i] = 2
        else:
            L[i] = 3

    for i in range(1, len(L)):
        if L[i] < L[i-1]:
            print('no')
            break
    else:
        print('yes')
```

Sqarime

Kjo është një zgjidhje e ndryshme. Në fillim i zëvendësojmë shkoronjat C, E, F me 1, 2, 3 dhe pastaj kontrollojmë që numrat vijnë gjithmonë në rritje.

Zgjidhja 4

```
from re import match
for _ in range(int(input())):
    print('yes') if match('^C*E*S*$',input()) else print('no')
```

Sqarime

Në këtë zgjidhje përdorim shprehjet e rregullta. Shprehja 'C' *do të thotë* disa C njëra pas tjetrës (ndoshta asnjë). *Po kështu edhe 'E'* dhe 'S*'. Kurse shënja ^ do të thotë që përputhja e shprehjes me vargun duhet të fillojë që në fillim të vargut. Po ashtu, shënja **$** do të thotë që përputhja e shprehjes me vargun duhet të vazhdojë dero në fund të vargut.

Detyra

Në dhomën e senatit ka rënë një zjarr i vogël dhe duhet të evakuohet! Aty ndodhen disa senatorë të cilët i përkasin njërës prej **N** partive politike, të cilat emërtohen me **N** shkronjat e para (të mëdha) të alfabetit anglisht.

Dera e emergjencës është e gjerë sa për dy persona, kështu që në çdo hap ju mund të zgjidhni për të nxjerë ose një senator ose dy.

Rregullat e senatit lejojnë mundësinë që senatorët që janë në dhomë mund të votojnë çdo ligj me shumicë votash, edhe në rast emergjence apo evakuimi! Kështu që senatorët duhen nxjerrë në mënyrë të tillë që në asnjë moment ndonjëra nga partitë politike të ketë shumicën absolute në senat. Dmth gjatë boshatisjes, në asnjë moment nuk duhet lejuar që më shumë se gjysma e senatorëve brenda në senat të jenë të një partie.

A mund të ndërtoni një plan evakuimi?

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000000130/00000000000004c0

Shembull

```
$ cat input.txt
4
2
2 2
3
3 2 2
3
1 1 2
3
```

```
2 3 1

$ python3 prog.py < input.txt
Case #1: AB BA
Case #2: AA BC C BA
Case #3: C C AB
Case #4: BA BB CA
```

Shembujt tregojnë njërën nga mënyrat e mundshme, por mund të ketë edhe mënyra të tjera.

1. Kemi 2 senatorë nga partia A dhe 2 nga partia B. Nxjerrim AB dhe na ngelen 1A 1B. Nxjerrim BA.
2. Kemi 3A 2B dhe 2C. Nqs AA na ngelen 1A 2B 2C. Kur nxjerrim BC na ngelen 1A 1B 1C. Pasi nxjerim C na ngelen 1A 1B. Nxjerrim BA.
3. Kemi 1A 1B 2C. Nxjerrim C dhe na ngelen 1B 1B 1C. Nxjerrim C dhe ngelen 1A 1B. Nxjerrim AB.
4. Kemi 2A 3B 1C. Nxjerrim BA dhe ngelen 1A 2B 1C. Nxjerrim BB dhe mbeten 1A 1C. Nxjerrim AC.

Problemi 070

Kërkesa

PANGRAM quhet një tekst që i përdor të paktën një herë të gjitha shkronjat e alfabetit anglisht.

Në një dokument të vjetër është shkruar diçka. Nëse teksti nuk është PANGRAM ne mund të blejmë disa shkronja për ta plotësuar. Nqs dimë edhe koston e çdo shkronje, sa është sasia më e vogël e parave që duhet të shpenzojmë për ta kthyer tekstin në PANGRAM?

Referenca: https://www.codechef.com/problems/MATPAN

Shembull

```
$ cat input.txt
2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
abcdefghijklmopqrstuvwz
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
thequickbrownfoxjumpsoverthelazydog
```

```
$ python3 prog.py < input.txt
63
0
```

Në rastin e parë mungojnë shkronjat **n** (çmimi 14), **x** (24) dhe **y** (25). Kështu që shuma që duhet të shpenzojmë është 14+24+25=63.

Në rastin e dytë nuk mungon asnjë nga shkronjat e alfabetit, kështu që përgjigja është 0.

Zgjidhja

```
for _ in range(int(input())):
    prices = [int(i) for i in input().split()]
    text = input()
    letters = [0 for i in range(26)]
    for l in text:
        letters[ord(l) - ord('a')] += 1
    cost = 0
    for i in range(26):
        if letters[i] == 0:
            cost += prices[i]
    print(cost)
```

Sqarime

Funksioni `ord()` jep kodin e një shkronje në tabelën ASCII. Kurse `ord(x) - ord('a')` i kthen shkronjat në një indeks që fillon nga 0. Programi i kalon me rradhë shkronjat e tekstit dhe numëron sa herë përsëritet çdo shkronjë. Në fund kontrollon se cilat shkronja përsëriten 0 herë dhe i shton çmimet e tyre në kosto.

Detyra

Në një dyqan çokollatash shkon një turist. Shitësi dhe turisti nuk e dinë gjuhën e njëritjetrit, por turisti tregon me dorë një lloj çokollatash dhe i jep shitësit disa kartmonedha. Duke ditur çmimin e një çokollate shitësi mund ta marrë me mend se sa copë do që të blejë turisti. Por nqs duke hequr një prej kartmonedhave mund të blihet përsëri i njëjti numër çokollatash, atere ky është një rast i paqartë dhe shitësi nuk di çfarë të bëjë, kështu që ia kthen lekët mbrapsht turistit.

Bëni një program që merr çmimin e një çokollate dhe kartmonedhat që jep turisti dhe gjen se sa copë çokollata do që të blejë turisti. Nëse rasti është i paqartë programi duhet të nxjerrë -1.

Referenca: https://www.codechef.com/problems/BUYING2

Shembull

```
$ cat input.txt
3
4 7
10 4 8 5
1 10
12
2 10
20 50

$ python3 prog.py < input.txt
-1
1
7
```

Në rastin e parë, sasia gjithsej e parave është 27, dhe meqë çmimi është 7, turisti mund të marrë vetëm 3 copë. Por nqs nuk do kishte dhënë kartmonedhën 5 përsëri do merrte 3 copë. Kështu që është rast i paqartë.

Në rastin e dytë sasia e parave është 12, çmimi 10, kështu që mund të marrë 1 çokollatë.

Në rastin e tretë sasia e parave është 20+50, çmimi 10, mund të marrë 7 çokollata.

9 Problemat 071-080

Problemi 071

Kërkesa

Në një rrugë të drejtë dhe të ngushtë po lëvizin disa makina, të cilat kanë shpejtësi maksimale të ndryshme. Nqs rrugën përpara do ta kishin bosh, makinat do preferonin që të ecnin me shpejtësinë e tyre maksimale. Por nqs kanë përpara një makinë më të ngadaltë, makinat janë të detyruara ta ulin shpejtësinë, meqënëse nuk mund të parakalojnë, dhe të ecin njëra-pas-tjetrës vargan.

Bëni një program që merr shpejtësitë e makinave dhe gjen se sa prej tyre po ecin me shpejtësi maksimale. Mund të supozojmë se rruga është shumë e gjatë dhe makinat kanë qenë duke ecur në të për një kohë të gjatë.

Referenca: https://www.codechef.com/problems/CARVANS

Shembull

```
$ cat input.txt
3
1
10
3
8 3 6
5
4 5 1 2 3

$ python3 prog.py < input.txt
1
2
2
```

Makinat po lëvizin nga e djathta në të majtë.

Në rastin e dytë, makina me shpejtësi 6 ndodhet pas asaj me shpejtësi 3, kështu që nuk mund të ecë me shpejtësi maksimale.

Në rastin e tretë, makina me shpejtësi 5 ndodhet pas asaj me shpejtësi 4, kështu që nuk mund të ecë me shpeltësi maksimale. Po ashtu, makinat 2 dhe 3 ndodhen pas asaj me shpejtësi 1, kështu janë futur në vargan. Me shpejtësinë e tyre maksimale ecin vetëm makinat 4 dhe 1.

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    M = [int(m) for m in input().split()]
    nr = 1
    for i in range(1, n):
        if M[i] > M[i-1]:
            M[i] = M[i-1]
        else:
            nr += 1
    print(nr)
```

Sqarime

Makina e parë gjithmonë ecën me shpejtësinë e vetë maksimale, sepse nuk ka makina të tjera përpara që ta pengojnë.

Nqs një makinë e ka shpejtësinë shpejtësinë maksimale më të madhe se shpejtësia me të cilën po ecën makina përpara saj, atere është e detyruar ta ulë shpejtësinë dhe të ecë me të njëjtën shpejtësi si makina para saj.

Përndryshe (kur e ka shpejtësinë maksimale më të vogël ose të barabartë me shpejtësinë e makinës përpara saj), mund të ecë me shpejtësinë e vetë maksimale.

Detyra

Kemi një përkëmbim të numrave **1, 2, ..., N**. Quajmë **të kundërt** një çift numrash **A[i]** dhe **A[j]** në këtë përkëmbim nëse **1 <= i < j <= N** dhe **A[i] > A[j]**. Quajmë **të kundërt fqinj** një çift numrash **A[i]** dhe **A[i+1]** të tillë që **A[i] > A[i+1]**, ku **1 <= i < N**. Një përkëmbim quhet **i mirë** nëse numri i të kundërtave në të është i njëjtë me numrin e të kundërtave fqinje.

Gjeni nëse një përkëmbim i dhënë është **i mirë** ose jo.

Referenca: https://www.codechef.com/problems/LEPERMUT

Shembull

```
$ cat input.txt
4
1
1
2
2 1
3
3 2 1
4
1 3 2 4

$ python3 prog.py < input.txt
YES
YES
NO
YES
```

1. Kemi vetëm 1 element, nuk ka çifte numrash të kundërta ose të kundërta fqinje, kështu që përkëmbimi është i mirë.
2. Kemi vetëm një çift të kundërt dhe 1 të kundërt fqinj. Përkëmbimi është i mirë.
3. Kemi 3 çifte të kundërta: (3,2), (3,1), (2,1) dhe 2 të kundërta fqinje: (3,2), (2,1). Përkëmbimi nuk është i mirë.
4. Kemi vetëm një çift të kundërt: (3,2), dhe një të kundërt fqinj: (3,2). Përkëmbimi është i mirë.

Problemi 072

Kërkesa

Në dhomën e senatit ka rënë një zjarr i vogël dhe duhet të evakuohet! Aty ndodhen disa senatorë të cilët i përkasin njërës prej **N** partive politike, të cilat emërtohen me **N** shkronjat e para (të mëdha) të alfabetit anglisht.

Dera e emergjencës është e gjerë sa për dy persona, kështu që në çdo hap ju mund të zgjidhni për të nxjerë ose një senator ose dy.

Rregullat e senatit lejojnë mundësinë që senatorët që janë në dhomë mund të votojnë çdo ligj me shumicë votash, edhe në rast emergjence apo evakuimi! Kështu që senatorët duhen nxjerrë në mënyrë të tillë që në asnjë moment ndonjëra nga partitë politike të ketë shumicën absolute në senat. Dmth gjatë boshatisjes, në asnjë moment nuk duhet lejuar që më shumë se gjysma e senatorëve brenda në senat të jenë të një partie.

A mund të ndërtoni një plan evakuimi?

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000000130/00000000000004c0

Shembull

```
$ cat input.txt
4
2
2 2
3
3 2 2
3
1 1 2
3
2 3 1

$ python3 prog.py < input.txt
Case #1: AB BA
Case #2: AA BC C BA
Case #3: C C AB
Case #4: BA BB CA
```

Shembujt tregojnë njërën nga mënyrat e mundshme, por mund të ketë edhe mënyra të tjera.

1. Kemi 2 senatorë nga partia A dhe 2 nga partia B. Nxjerrim AB dhe na ngelen 1A 1B. Nxjerrim BA.
2. Kemi 3A 2B dhe 2C. Nqs AA na ngelen 1A 2B 2C. Kur nxjerrim BC na ngelen 1A 1B 1C. Pasi nxjerim C na ngelen 1A 1B. Nxjerrim BA.
3. Kemi 1A 1B 2C. Nxjerrim C dhe na ngelen 1B 1B 1C. Nxjerrim C dhe ngelen 1A 1B. Nxjerrim AB.
4. Kemi 2A 3B 1C. Nxjerrim BA dhe ngelen 1A 2B 1C. Nxjerrim BB dhe mbeten 1A 1C. Nxjerrim AC.

Zgjidhja

```python
for t in range(int(input())):
    n = int(input())
    S = [int(i) for i in input().split()]
    X = []    # keep the steps of the solution
    if n == 2:
        for _ in range(S[0]):
```

```
            X.append('AB')
    else:
        P = []    # names of each party
        for i in range(n):
            P.append(chr(ord('A') + i))
        # sort in decreasing order
        for i in range(n-1):
            for j in range(i+1, n):
                if S[j] > S[i]:
                    S[i], S[j] = S[j], S[i]
                    P[i], P[j] = P[j], P[i]
        while len(S) > 2:
            X.append(P[0])
            S[0] -= 1
            if S[0] == 0:
                del(S[0])
                del(P[0])
            else:
                # keep it sorted
                for i in range(1, len(S)):
                    if S[i] <= S[i-1]:
                        break
                    else:
                        S[i], S[i-1] = S[i-1], S[i]
                        P[i], P[i-1] = P[i-1], P[i]
        X.append(P[0] + P[1])

    print('Case #{}:'.format(t+1), *X)
```

Sqarime

Strategjia e përdorur është që të mbajmë një listë të partive të renditur në rendin zbritës sipas numrit të senatorëve të secilës parti, dhe të nxjerrim një senator nga partia që ka numrin më të madh. Kjo garanton që asnjë parti të mos ketë shumicën absolute, me përjashtim të rastit kur kemi vetëm dy parti që në fillim, me një numër të barabartë senatorësh. Këtë e trajtojmë si rast të veçantë, duke nxjerrë njëherësh një senator nga njëra parti dhe një nga tjetra.

Detyra

Në një varg që përbëhet nga shifrat **0** dhe **1** kontrolloni nëse të gjitha shifrat **1** formojnë një segment të vetëm dhe jo-bosh.

Referenca: https://www.codechef.com/problems/SEGM01

Shembull

```
$ cat input.txt
6
001111110
00110011
000
1111
101010101
101111111111

$ python3 prog.py < input.txt
YES
NO
NO
YES
NO
NO
```

Problemi 073

Kërkesa

Përpara zgjedhjeve është e rëndësishme që listat e votuesve të jenë sa më të sakta. Kryetari i komisionit të zgjedhjeve ka ngarkuar tre punonjës që të përpilojnë listat e votuesve të qytetit në mënyrë të pavarur nga njëri-tjetri. Në këto lista secili votues shënohet me një numër që është identiteti i tij. Pastaj ai i merr këto lista dhe harton një listë tjetër që përmban vetëm ata persona që ndodhen në të paktën dy prej listave të mësipërme. Bëni një program që mund të ndihmojë kryetarin në hartimin e kësaj liste.

Referenca: https://www.codechef.com/problems/VOTERS

Shembull

```
$ cat input.txt
1
23 30 42 57 90
21 23 35 57 90 92
21 23 30 57 90
```

```
$ python3 prog.py < input.txt
21 23 30 57 90
```

Për çdo rast testimi kemi tre listat e punonjësve. Numri 21 ndodhet në listën e dytë dhe të tretë, kështu që është edhe në listën përfundimtare. Numri 23 ndodhet në të treja listat. Numri 30 ndodhet në të parën dhe të tretën. Numrat 35 dhe 42 ndodhen vetëm në një listë, kështu që nuk janë hedhur në listën përfundimtare. Numrat 57 dhe 90 ndodhen në të treja listat, kurse numri 92 ndodhet vetëm në një prej tyre.

Zgjidhja 1

```
for _ in range(int(input())):
    L1 = [int(i) for i in input().split()]
    L2 = [int(i) for i in input().split()]
    L3 = [int(i) for i in input().split()]
    L4 = []
    for v in L1:
        if v in L2 or v in L3:
            L4.append(v)
    for v in L2:
        if v not in L1 and v in L3:
            L4.append(v)
    print(*L4)
```

Sqarime

Llogjika e k;saj zgjidhje ;sht; e thjeshtë, por koha e ekzekutimit është kuadratike ($O(N^2)$).

Zgjidhja 2

```
def union(l1, l2):
    l = []
    i = j = 0
    while i < len(l1) and j < len(l2):
        if l1[i] < l2[j]:
            l.append(l1[i])
            i += 1
        elif l2[j] < l1[i]:
```

```
            l.append(l2[j])
            j += 1
        else:    # l1[i] == l2[j]
            l.append(l1[i])
            i += 1
            j += 1
    while i < len(l1):
        l.append(l1[i])
        i += 1
    while j < len(l2):
        l.append(l2[j])
        j += 1
    return l

for _ in range(int(input())):
    L1 = [int(i) for i in input().split()]
    L2 = [int(i) for i in input().split()]
    L3 = [int(i) for i in input().split()]
    L1.sort()
    L2.sort()
    L3.sort()
    I1 = intersection(L1, L2)
    I2 = intersection(L1, L3)
    I3 = intersection(L2, L3)
    U1 = union(I1, I2)
    U2 = union(U1, I3)
    print(*U2)
```

Sqarime

Kjo zgjidhje është pak më e gjatë dhe më e vështirë, por koha e saj është më e mirë se koha e zgjidhjes së parë. Funksionet `intersection()` dhe `union()` kanë kohë lineare ($O(N)$), por renditja zakonisht ka kohë të rendit $O(N * ln(N))$, kështu që kjo e fundit mbizotëron. Por $O(N * ln(N))$ është më e mirë se $O(N^2)$.

Zgjidhja 3

```
for _ in range(int(input())):
    L1 = set(int(i) for i in input().split())
    L2 = set(int(i) for i in input().split())
    L3 = set(int(i) for i in input().split())
```

```
    L4 = L1.intersection(L2).union(L2.intersection(L3)).union(L1.intersection(L3))
    print(*L4)
```

Sqarime

Duke përdorur bashkësitë (`set()`) dhe funksionet e gatshme të tyre zgjidhja është shumë më e shkurtër. Por veprimet në bashkësi janë të rendit $O(N * ln(N))$, kështu që koha e kësaj zgjidhje është njëlloj si koha e zgjidhjes së dytë.

Detyra

Ada ka **N** lapsa me ngjyra, disa me majën poshtë dhe disa me majën lart. Ada mendon se do ishte më murë sikur të gjithë lapsat të ishin në të njëjtin drejtim. Ajo mund të kthejë njëherësh një segment të tërë me lapsa të njëpasnjëshëm. Pas një kthimi, ata që ishin me majë poshtë do kthehen me majën lart, dhe anasjelltas. Sa është numri më i vogël i kthimeve për të bërë të gjithë lapsat me të njëjtin drejtim?

Referenca: https://www.codechef.com/problems/ADACRA

Shembull

```
$ cat input.txt
1
UUDDDUUU

$ python3 prog.py < input.txt
1
```

Me **U** shënohet një laps me majën lart dhe me **D** një laps që e ka majën poshtë. Në rastin e dhënë, mjafton vetëm një kthim për ti vendosur me majën lart të gjithë lapsat që janë me majën poshtë.

Problemi 074

Kërkesa

Ani po ecën me kalë në një rrugë të drejtë që shkon nga perëndimi në lindje. Ajo ndodhet në kilometrin **0** kurse vend-mbërritja e saj ndodhet në kilometrin **D**, ku kilometrat përgjatë rrugës shënohen nga perëndimi në lindje.

Janë edhe **N** kuaj të tjerë që po udhëtojnë në të njëjtën rrugë. Të gjithë ata do të vazhdojnë të udhëtojnë përgjithmonë, dhe të gjithë ndodhen midis Anit dhe vend-mbërritjes së saj. Fillimisht kali **i** ndodhet në kilometrin K_i dhe ecën me shpejtësinë e tij maksimale prej S_i kilometra në orë.

Kuajt janë shumë të sjellshëm dhe kali H_1 nuk e parakalon një kalë tjetër H_2 që ndodhet përpara tij. Dy ose më shumë kuaj mund të ndodhen në të njëjtën pikë për një kohë të çfarëdoshme; kuajt mund ti konsiderojmë sikur janë pika me përmasa të papërfillshme. Kuajt, përveç atij të Anit, vazhdojnë të ecin me shpejtësinë e tyre maksimale sa kohë që rruga para tyre është e lirë, por nqs u del përpara ndonjë kalë më i ngadaltë, atere e ulin shpejtësinë e tyre që të jetë njësoj me atë të kalit që kanë përpara.

Kurse kali i Anit nuk ka ndonjë shpejtësi maksimale dhe mund të ecë me një shpejtësi të çfarëdoshme që Ani mund të zgjedhë, sa kohë që përpara nuk i del ndonjë kalë më i ngadaltë. Për të pasur një udhëtim sa më të shtruar për veten dhe për kalin e saj, Ani do që të zgjedhë një shpejtësi të tillë që të mos ta ndryshojë shpejtësinë deri sa të arrijë në fund të udhëtimit, dhe të mos kalojë asnjë kalë tjetër. Cila është shpejtësia më e madhe që mund të zgjedhë?

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000000130/0000000000000524

Shembull

```
$ cat input.txt
3
2525 1
2400 5
300 2
120 60
60 90
100 2
80 100
70 10

$ python3 prog.py < input.txt
Case #1: 101.000000
Case #2: 100.000000
Case #3: 33.333333
```

Dy rreshtat e parë janë **D** dhe **N**, pastaj për N rreshtat që vijojnë kemi K_i dhe S_i.

Në rastin e parë më rrugë ndodhet vetëm një kal shumë i ngadaltë, i cili do të arrijë vend-mbërritjen e Anit pas 25 orësh. Çdo shpejtësi më e madhe se 101 km/orë do bënte që Ani ta kalonte këtë kal para se të mbërinte në destinacion.

Në rastin e dytë janë 2 kuaj në rrugë. Ai më i shpejti e arrin më të ngadaltin te kilometri 240, pas 2 orësh. Pastaj të dy kuajt do të vazhdojnë të ecin me shpejtësinë e atij të ngadaltit edhe për një orë tjetër, deri sa të arrijnë në destinacionin e Anit, te kilometri 300. Shpejtësia më e madhe që mund të zgjedhë Ani pa kaluar ndonjë prej kuajve është 100 km/orë.

Zgjidhja

```
for t in range(int(input())):
    d, n = map(int, input().split())
    time_max = 0
    for i in range(n):
        k, s = map(int, input().split())
        time = (d - k) / s
        if time > time_max:
            time_max = time
    print('Case #{}: {:f}'.format(t+1, d / time_max))
```

Sqarime

Ky problem duket shumë i vështirë po ta krahasojmë me zgjidhjen, që nuk ka veçse 9 rreshta.

Shpejtësinë maksimale të mundshme kali i Anit mund ta ketë vetëm atere kur arrin në destinacion në të njëjtin moment me kalin që ka përpara (të cilin po e quajmë A). Kali A ose do arrijë në destinacion pa qënë i detyruar ta ulë shpejtësinë gjatë rrugës (dhe kështu do jetë kali i vetëm që e kufizon shpejtësinë e Anit), ose do ta ulë shpejtësinë në një pikë të caktuar për shkak të kalit që ka përpara (le ta quajmë B). E njëjta gjë vlen edhe për kalin B: ose do arrijë në destinacion duke ecur me shpejtësinë e tij maksimale, ose do jetë i detyruar ta ulë shpejtësinë gjatë rrugës për shkak të kalit që ka përpara. E kështu me rradhë. Kështu që do jetë një kalë i vetëm kufizues që e përcakton se sa shpejt mund ta arrijë destinacionin kali i Anit. Mund të vërtetohet që ky është kali i vetëm që ka rëndësi, kurse të tjerët mund të mos ti marrim parasysh.

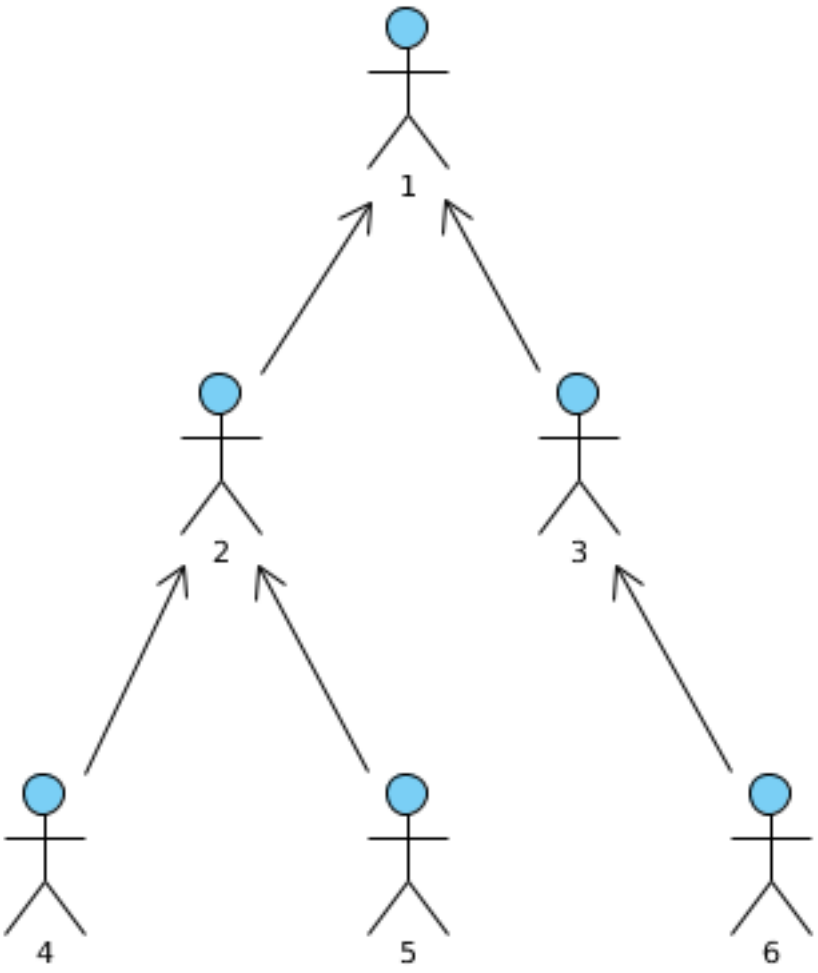
Është e qartë që kuajt në lindje të kalit kufizues mund të mos ti marrim parasysh sepse do kalojnë te destinacioni përpara se kali kufizues të vejë atje. Po kuajt që ndodhen midis Anit dhe kalit kufizues? Nga mënyra se si e përkufizuam kalin kufizues e dimë që çdo kal i ndërmjetëm do ta arrijë kalin kufizues përpara destinacionit. (Nëse ndonjëri nuk e arrin, atere *ky* do ishte kali kufizues). E zëmë se Ani zgjedh një shpejtësi me të cilën arrin në destinacion në të njëjtën kohë me kalin kufizues. Një shpejtësi më e madhe se kjo do ishte e pamundur. Gjithashtu me një shpejtësi të tillë nuk mund të parakalojë ndonjë nga kuajt e ndërmjetëm, sepse kuajt e ndërmjetëm e arrijnë kalin kufizues përpara destinacionit, kurse kali i Anit e arrin atë pikërisht në destinacion.

Kështu që nqs gjejmë kalin kufizues zgjidhja është e thjeshtë: ec me një shpejtësi të tillë që të mbërrish në destinacion në të njëjtën kohë me kalin kufizues. Për të gjetur se cili është kali kufizues llogarisim se sa kohë i duhet secilit prej kuajve që të mbërrijë në destinacion prej aty ku është. Pikërisht kali që ka kohën më të madhe është kali kufizues. Këtë kohë mund ta përdorim për të llogaritur shpejtësinë e kalit të Anit.

Kjo zgjidhje e ka kohën të rendit $O(N)$.

Detyra

Një kompani ka një strukture hierarkike si kjo në figurë:



Në krye është pronari i cili nuk varet nga askush, pastaj janë manaxherët, të cilët kanë

disa vartës dhe varen nga një manaxher tjetër ose nga pronari, pastaj janë të punësuarit e thjeshtë, të cilët nuk kanë vartës. Kjo strukturë mund të paraqitet duke treguar për çdo person përgjegjësin e tij në këtë mënyrë: **0 1 1 2 2 3** Personi 1 nuk varet nga askush, prandaj përgjegjësi i tij është shënuar me **0**. Personat 2 dhe 3 varen nga personi **1**, personat 4 dhe 5 varen nga personi **2**, dhe personi 6 varet nga personi **3**.

Bëni një program që merr paraqitjen e hirarkisë në këtë formë dhe gjen se kush janë punonjësit e thjeshtë.

Referenca: https://www.codechef.com/problems/CHEFDETE

Shembull

```
$ cat input.txt
2
6
0 1 1 2 2 3
13
0 1 2 3 1 3 5 7 1 9 9 10 11

$ python3 prog.py < input.txt
4 5 6
4 6 8 12 13
```

Problemi 075

Kërkesa

Le ta quajmë **tidy** (të rregullt) një numër që i ka shifrat të renditura në rendin jo-zbritës. P.sh. numrat 8, 123, 555, 224488 janë **tidy**, kurse numrat 20, 321, 495, 999990 nuk janë.

Nqs numërojmë numrat nga 1 në **N**, cili ka qenë numri i fundit **tidy** që kemi numëruar? Ose me fjalë të tjera, gjeni numrin më të madh **tidy** që është më i vogël (ose i barabartë) se një numër i dhënë **N**.

Referenca: https://code.google.com/codejam/contest/3264486/dashboard#s=p1&a=1

Shembull

```
$ cat input.txt
4
132
1000
7
```

```
11111111111111110

$ python3 prog.py < input.txt
Case #1: 129
Case #2: 999
Case #3: 7
Case #4: 99999999999999999
```

Zgjidhja 1

```python
def tidy(n):
    d = n % 10
    while n > 0:
        if n % 10 > d:
            return False
        d = n % 10
        n //= 10
    return True

for t in range(int(input())):
    N = int(input())
    while N > 0 and not tidy(N):
        N -= 1
    print('Case #{}: {}'.format(t+1, N))
```

Sqarime

Kemi një funksion që kontrollon nëse një numër i dhënë është tidy, pastaj vazhdojmë ta zvogëlojmë numrin deri sa të gjejmë një që është tidy.

Kjo zgjidhje është llogjikisht e thjeshtë, por për numra të mëdhenj (p.sh. për rastin e fundit) vonohet shumë. Duhet gjetur një zgjidhje më e shpejtë.

Zgjidhja 2

```python
def get_tidy(n):
    '''
    Get the smallest tidy number that is >= n
    '''
    D = [int(d) for d in list(str(n))]
```

```
    i = 1
    while i < len(D) and D[i] >= D[i-1]:
        i += 1
    while i < len(D):
        D[i] = D[i-1]
        i += 1
    return(int(''.join([str(d) for d in D])))

for t in range(int(input())):
    N = input()
    n = len(N)
    if n == 1:
        print('Case #{}: {}'.format(t+1, N))
        continue
    N = int(N)
    last_t = int('9'*(n-1))
    tidy = last_t
    while tidy < N:
        last_t = tidy
        tidy = get_tidy(tidy + 1)
    if tidy > N:
        tidy = last_t
    print('Case #{}: {}'.format(t+1, tidy))
```

Sqarime

Kjo zgjidhje fillon nga një numër që ne e dimë që është tidy dhe më i vogël se numri i dhënë, dhe gjen vazhdimisht numrin pasardhës tidy, deri sa ky numër të bëhet më i madh se N. Atere numri i fundit tidy që kemi parë është zgjidhja e kërkuar.

Kjo zgjidhje në përgjithësi është më e shpejtë se zgjidhja e parë, sepse nuk i kontrollon numrat me rradhë por vetëm ata që janë tidy, dhe numrat tidy nuk janë shumë të shpeshtë. Gjithsesi edhe kjo zgjidhje është e ngadaltë për numra shumë të mëdhenj (rasti i fundit).

Zgjidhja 3

```
for t in range(int(input())):
    D = [int(d) for d in list(input())]
    i = 1
    while i < len(D) and D[i] >= D[i-1]:
```

```
        i += 1
    if i == len(D):
        print('Case #', t+1, ': ', *D, sep='')
        continue
    i -= 1
    while i > 0 and D[i-1] == D[i]:
        i -= 1
    D[i] = D[i] - 1

    i += 1
    while i < len(D):
        D[i] = 9
        i += 1

    if D[0] == 0:
        D[0] = ''
    print('Case #', t+1, ': ', *D, sep='')
```

Sqarime

Kjo zgjidhje është jashtëzakonisht e shpejtë edhe për numra shumë të mëdhenj. Numrin e kërkuar e përftojmë duke ndryshuar shifrat e numrit të dhënë në mënyrën e duhur.

Detyra

Trajneri i futbollit të një shkolle duhet të zgjedhë **P** nxënës për të përfaqësuar shkollën në kampionat. Shkolla ka **N** nxënës dhe aftësitë e secilit në futboll shënohen me një numër natyror S_i.

Trajneri mendon se skuadra duhet të jetë e niveluar, dmth të ketë **P** nxënës me të njëjtin nivel aftësish, sepse kështu do të luajnë të gjithë si skuadër. Në fillim mund të mos jetë e mundur që të gjenden **P** nxënës me aftësi të barabarta, kështu që trajnerit do ti duhet ti stërvisë individualisht disa nxënës, për të rritur aftësitë e tyre. Me një orë stërvitje aftësia e një nxënësi mund të ngrihet me 1 pikë.

Sa është minimumi i orëve të stërvitjes që duhen për të krijuar një skuadër të niveluar.

Referenca: https://codingcompetitions.withgoogle.com/kickstart/round/0000000000050e01/00000000000698d6

Shembull

```
$ cat input.txt
```

```
3
4 3
3 1 9 100
6 2
5 5 1 2 3 4
5 5
7 7 1 7 7

$ python3 prog.py < input.txt
Case #1: 14
Case #2: 0
Case #3: 6
```

Jepen **N** dhe **P**, dhe më pas aftësitë e çdo nxënësi.

Në rastin e parë mund të trajnohet për 6 orë nxënësi i parë dhe për 8 orë nxënësi i dytë, dhe atere tre nxënësit e parë do kenë të njëjtin nivel aftësish.

Në rastin e dytë, dy nxënësit e parë kanë të njëjtin nivel aftësish, kështu që nuk është nevoja për trajnime shtesë.

Në rastin e tretë, nxënësi i tretë duhet trajnuar për 6 orë, dhe kështu të 5 nxënësit do kenë të njëjtin nivel.

Problemi 076

Kërkesa

Kemi një përkëmbim të numrave **1, 2, ..., N**. Quajmë **të kundërt** një çift numrash **A[i]** dhe **A[j]** në këtë përkëmbim nëse **1 <= i < j <= N** dhe **A[i] > A[j]**. Quajmë **të kundërt fqinj** një çift numrash **A[i]** dhe **A[i+1]** të tillë që **A[i] > A[i+1]**, ku **1 <= i < N**. Një përkëmbim quhet **i mirë** nëse numri i të kundërtave në të është i njëjtë me numrin e të kundërtave fqinje.

Gjeni nëse një përkëmbim i dhënë është **i mirë** ose jo.

Referenca: https://www.codechef.com/problems/LEPERMUT

Shembull

```
$ cat input.txt
4
1
1
2
```

```
2 1
3
3 2 1
4
1 3 2 4

$ python3 prog.py < input.txt
YES
YES
NO
YES
```

1. Kemi vetëm 1 element, nuk ka çifte numrash të kundërta ose të kundërta fqinje, kështu që përkëmbimi është i mirë.
2. Kemi vetëm një çift të kundërt dhe 1 të kundërt fqinj. Përkëmbimi është i mirë.
3. Kemi 3 çifte të kundërta: (3,2), (3,1), (2,1) dhe 2 të kundërta fqinje: (3,2), (2,1). Përkëmbimi nuk është i mirë.
4. Kemi vetëm një çift të kundërt: (3,2), dhe një të kundërt fqinj: (3,2). Përkëmbimi është i mirë.

Zgjidhja 1

```python
for _ in range(int(input())):
    n = int(input())
    A = [int(i) for i in input().split()]
    nr1 = nr2 = 0
    for i in range(n-1):
        if A[i] > A[i+1]:
            nr2 += 1
        for j in range(i+1, n):
            if A[i] > A[j]:
                nr1 += 1
    print('YES') if nr1 == nr2 else print('NO')
```

Sqarime

Numërojmë çiftet e kundërta dhe çiftet e kundërta fqinje dhe i krahasojmë.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    A = [int(i) for i in input().split()]
    m = True    # supozojme se perkembimi eshte i mire
    i = 0
    while m and i < n - 2:
        j = i + 2
        while m and j < n:
            if A[i] > A[j]:
                m = False
            j += 1
        i += 1
    print('YES') if m else print('NO')
```

Sqarime

Mund të vihet re kollaj se çiftet e kundërta fqinje janë njëkohësisht çifte të kundërta. Kështu që nqs gjejmë një çift të kundërt që nuk është fqinj, mund të konkludojmë që përkëmbimi nuk është i mirë.

Detyra

Babai ka shkuar të bëjë pazarin bashkë me të birin 5-vjeçar. Deri tani kanë blerë **N** gjëra, të cilat peshojnë W_i secila. Djali ngul këmbë që ta ndihmojë babanë për ti mbajtur sendet e blera. Por që babai mos tia bëjë "me hile" dhe ti japë gjëra shumë të lehta, djali kërkon që sendet të ndahen në dy grupe, ku njëri prej tyre ka **K** elementë, dhe pastaj njërin prej grupeve ta mbajë djali dhe tjetrin babain. Babai sigurisht që do ti japë djalit grupin që peshon më pak. Por si mund ti ndajë sendet në dy grupe në mënyrë të tillë që diferenca në peshë midis tyre të jetë sa më e madhe?

Bëni një program që gjen diferencën në peshë më të madhe, nqs **N** sende i ndajmë në dy grupe, ku njëri nga grupet ka **K** sende.

Referenca: https://www.codechef.com/problems/MAXDIFF

Shembull

```
$ cat input.txt
2
5 2
```

```
8 4 5 2 10
8 3
1 1 1 1 1 1 1 1

$ python3 prog.py < input.txt
17
2
```

Në rastin e parë kemi N=5 dhe K=2. Diferenca më e madhe do jetë nëse djalit i jep sendet me peshë 2 dhe 4. Diferenca në këtë rast do jetë (8+5+10)-(4+2)=17.

Në rastin e dytë sido që të bëhet ndarja, diferenca do jetë gjithmonë 2.

Problemi 077

Kërkesa

Në një varg që përbëhet nga shifrat **0** dhe **1** kontrolloni nëse të gjitha shifrat **1** formojnë një segment të vetëm dhe jo-bosh.

Referenca: https://www.codechef.com/problems/SEGM01

Shembull

```
$ cat input.txt
6
001111110
00110011
000
1111
101010101
101111111111

$ python3 prog.py < input.txt
YES
NO
NO
YES
NO
NO
```

Zgjidhja 1

```
for _ in range(int(input())):
    s = input()
    i = 0
    j = len(s) - 1
    while i < j:
        if s[i] == '1' and s[j] == '1':
            break
        if s[i] == '0':
            i += 1
        if s[j] == '0':
            j -= 1
    if i > j:
        print('NO')
    else:
        while i < j:
            if s[i] == '0':
                break
            i += 1
        if i == j and s[i] == '1':
            print('YES')
        else:
            print('NO')
```

Sqarime

Marrim dy indekse **i** dhe **j**, njëri që niset nga fillimi dhe tjetri nga fundi, dhe i lëvizim deri sa të gjejmë fillimin dhe fundin e njëshave. Pastaj kontrollojmë nëse midis këtyre ka ndonjë zero.

Zgjidhja 2

```
for _ in range(int(input())):
    s = input()
    nr1 = s.count('1')
    if nr1 == 0:
        print('NO')
    else:
        i = 0
```

```
        while s[i] != '1':
            i += 1
        j = len(s) - 1
        while s[j] != '1':
            j -= 1
        if j - i + 1 == nr1:
            print('YES')
        else:
            print('NO')
```

Sqarime

Fillimisht gjejmë numrin e njëshave. Pastaj indeksin e njëshit të parë nga fillimi, dhe të njëshit të parë nga fundi. Pastaj kontrollojmë nëse largësia midis këtyre dy indekseve është e barabartë me numrin e njëshave.

Detyra

Në një kompani janë **N** punonjës, pagat e të cilëve janë W_i (për **i** nga **1** në **N**). Një ditë pronari vendosi që tua bëjë rrogat të barabarta të gjithëve. Por për të arritur këtë qëllim ai përdor një veprim të tillë: Zgjedh një punonjës, dhe rrit me 1 rrogën e gjithë punonjësve të tjerë (por jo të këtij që ka zgjedhur). Sigurisht që punonjësi i zgjedhur mund të jetë i ndryshëm për secilin veprim. Sa është numri më i vogël i veprimeve të tilla që duhen kryer deri sa të gjithë punonjësit të kenë rroga të barabarta?

Referenca: https://www.codechef.com/problems/SALARY

Shembull

```
$ cat input.txt
2
3
1 2 3
2
42 42

$ python3 prog.py < input.txt
3
0
```

Në rastin e parë, fillimisht zgjidhet punonjësi i tretë, dhe pagat bëhen (2, 3, 3). Pastaj mund të zgjidhet punonjësi i dytë, dhe pagat bëhen (3, 3, 4). Pastaj mund të zgjidhet punonjësi i tretë, dhe pagat bëhen (4, 4, 4). Pra duhen 3 veprime për ti barazuar.

Në rastin e dytë janë tashmë të barabarta, kështu që nuk është nevoja për asnjë veprim.

Problemi 078

Kërkesa

Babai ka shkuar të bëjë pazarin bashkë me të birin 5-vjeçar. Deri tani kanë blerë **N** gjëra, të cilat peshojnë W_i secila. Djali ngul këmbë që ta ndihmojë babanë për ti mbajtur sendet e blera. Por që babai mos tia bëjë "me hile" dhe ti japë gjëra shumë të lehta, djali kërkon që sendet të ndahen në dy grupe, ku njëri prej tyre ka **K** elementë, dhe pastaj njërin prej grupeve ta mbajë djali dhe tjetrin babain. Babai sigurisht që do ti japë djalit grupin që peshon më pak. Por si mund ti ndajë sendet në dy grupe në mënyrë të tillë që diferenca në peshë midis tyre të jetë sa më e madhe?

Bëni një program që gjen diferencën në peshë më të madhe, nqs **N** sende i ndajmë në dy grupe, ku njëri nga grupet ka **K** sende.

Referenca: https://www.codechef.com/problems/MAXDIFF

Shembull

```
$ cat input.txt
2
5 2
8 4 5 2 10
8 3
1 1 1 1 1 1 1 1

$ python3 prog.py < input.txt
17
2
```

Në rastin e parë kemi N=5 dhe K=2. Diferenca më e madhe do jetë nëse djalit i jep sendet me peshë 2 dhe 4. Diferenca në këtë rast do jetë (8+5+10)-(4+2)=17.

Në rastin e dytë sido që të bëhet ndarja, diferenca do jetë gjithmonë 2.

Zgjidhja 1

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    W = [int(i) for i in input().split()]
```

```
    W.sort()
    s1 = s2 = 0
    for i in range(k):
        s1 += W[i]
        s2 += W[n-1-i]
    s = 0
    for i in range(n):
        s += W[i]
    print(max(abs(s1 - (s - s1)), abs(s2 - (s - s2))))
```

Sqarime

Nqs duam të gjejmë **K** numrat që kanë shumën më të madhe, mjafton që ti rendisim dhe të marrim **K** numrat e fundit. Gjithashtu **(N-K)** kanë shumën më të vogël të mundshme, kështu që diferenca midis këtyre dy grupeve është më e madhja e mundshme.

Nqs duam të gjejmë **K** numrat që kanë shumën më të vogël të mundshme, mjafton që ti rendisim dhe të marrim **K** numrat e parë. Atere **(N-K)** numrat që mbeten kanë shumën më të vogël të mundshme, kështu që përsëri diferenca është më e madhja e mundshme.

Programi i shqyrton të dyja këto raste dhe gjen diferencën më të madhe prej tyre.

Zgjidhja 2

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    W = [int(i) for i in input().split()]
    W.sort()
    s = sum(W)
    s1 = sum(W[0:k])
    s2 = sum(W[-k:])
    print(max(abs(2*s1 - s), abs(2*s2 - s)))
```

Sqarime

E njëjta zgjidhje por pak më kompakte, ku `W[0:k]` jep nënlistën me k elementet e para, dhe `W[-k:]` jep nënlistën me k elementet e fundit.

Detyra

Në një listë me numra natyrorë bëjmë këtë veprim: Zgjedhim disa prej numrave të listës dhe i fshijmë nga lista. Si kosto të këtij veprimi përkufizojmë rezultatin e kryerjes së një

Bitwise OR midis elementeve të fshira. Këtë veprim e përsërisim deri sa në listë të mos ngelet më asnjë element. Kostoja e përgjithshme është shuma e kostos së çdo veprimi. Gjeni vlerën më të vogël të mundshme të kostos së përgjithshme.

Referenca: https://www.codechef.com/problems/CHNGOR

Shembull

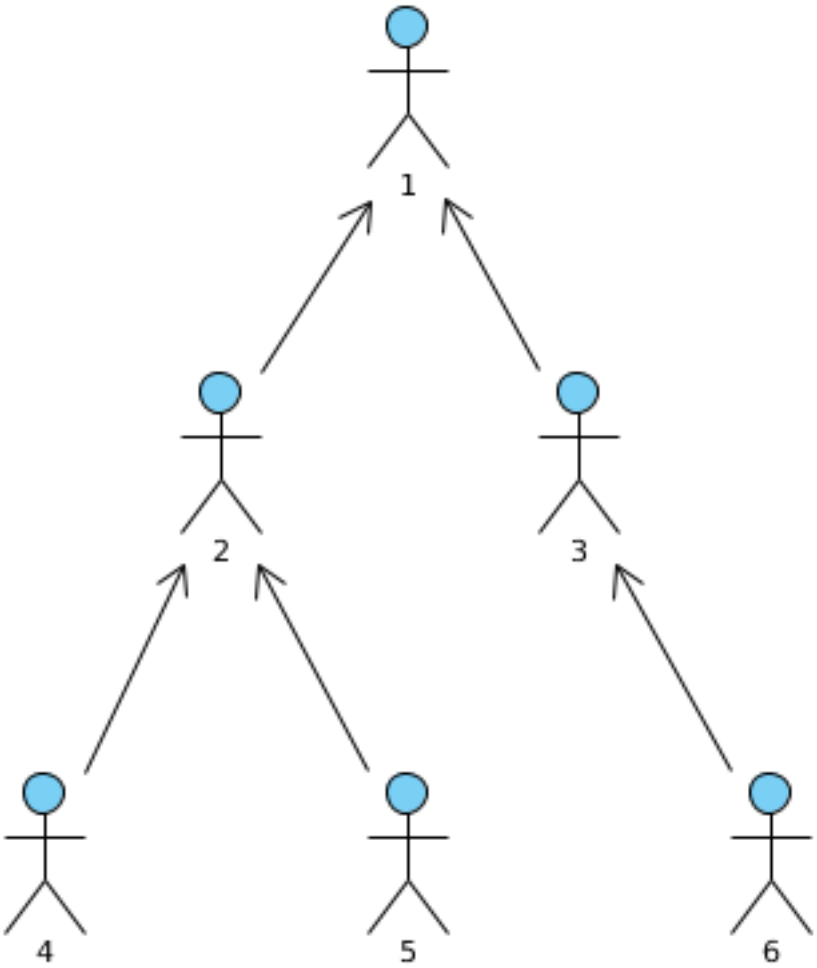
```
$ cat input.txt
2
2
1 2
3
1 3 6

$ python3 prog.py < input.txt
3
7
```

Problemi 079

Kërkesa

Një kompani ka një strukture hierarkike si kjo në figurë:



Në krye është pronari i cli nuk varet nga askush, pastaj janë manaxherët, të cilët kanë disa vartës dhe varen nga një manaxher tjetër ose nga pronari, pastaj janë të punësuarit e thjeshtë, të cilët nuk kanë vartës. Kjo strukturë mund të paraqitet duke treguar për çdo person përgjegjësin e tij në këtë mënyrë: **0 1 1 2 2 3** Personi 1 nuk varet nga askush, prandaj përgjegjësi i tij është shënuar me **0**. Personat 2 dhe 3 varen nga personi **1**, personat 4 dhe 5 varen nga personi **2**, dhe personi 6 varet nga personi **3**.

Bëni një program që merr paraqitjen e hirarkisë në këtë formë dhe gjen se kush janë punonjësit e thjeshtë.

Referenca: https://www.codechef.com/problems/CHEFDETE

Shembull

```
$ cat input.txt
2
6
0 1 1 2 2 3
13
0 1 2 3 1 3 5 7 1 9 9 10 11

$ python3 prog.py < input.txt
4 5 6
4 6 8 12 13
```

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    V = [int(i) for i in input().split()]
    P = []
    for i in range(1, n+1):
        if i not in V:
            P.append(i)
    print(*P)
```

Sqarime

Duhet të gjejmë ata persona që nuk shfaqen në listën e dhënë, sepse ata nuk kanë vartës, pra janë punonjës të thjeshtë. Koha e kësaj zgjidhje është e rendit $O(N^2)$.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    V = [int(i) for i in input().split()]
    P = []
    V.sort()
    i = 1
    j = 0
    while i <= n and j < len(V):
        if V[j] < i:
```

```
            j += 1
        elif V[j] > i:
            P.append(i)
            i += 1
        else:    # V[j] == i
            i += 1
            j += 1
    while i <= n:
        P.append(i)
        i += 1
    print(*P)
```

Sqarime

Për të gjetur personat që nuk janë në listën e dhënë, fillimisht e renditim listën dhe pastaj bëjmë një kërkim linear në listën e dhënë. Koha e kësaj zgjidhje është $O(N * ln(N))$ (që është koha për renditjen e listës), e cila është më e mirë se $O(N^2)$

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    V = [int(i) for i in input().split()]
    P = []
    ka_vartes = [False for i in range(n+1)]
    for i in range(len(V)):
        ka_vartes[V[i]] = True
    for i in range(n+1):
        if not ka_vartes[i]:
            P.append(i)
    print(*P)
```

Sqarime

Për të gjetur ata persona që nuk kanë vartës, krijojmë një listë me vlera vërtetësie (True/False), dhe shënojmë me True në të personat që kanë vartës.

Koha e kësaj zgjidhje është $O(N)$, që është më e mirë se $O(N * ln(N))$.

Detyra

Në koncertin e operas këngëtarja kryesore është një mikja juaj. Ju doni të siguroheni që në fund të shfaqjes e gjithë salla të ngrihet në këmbë dhe të duartrokasë.

Në fillim spektatorët janë të ulur. Çdo spektator ka një nivel turpi. Një spektator me nivel turpi S_i pret deri sa të paktën S_i spektatorë të tjerë të jenë çuar në këmbë duke duartrokitur, pastaj çohet edhe ai. Nëse $S_i = 0$ atere spektatori çohet gjithmonë në këmbë për të duartrokitur, pavarësisht nëse është çuar ndonjë tjetër. Nëse një spektator e ka $S_i = 2$ atere pret deri sa të çohen të paktën 2 të tjerë, para se të çohet.

Ju e njihni nivelin e turpit të spektatorëve që do shkojnë në koncert dhe mund të ftoni edhe njerëz të tjerë me një nivel turpi të çfarëdoshëm, jo domosdoshmërisht të njëjtë. Sa është numri më i vogël i njerëzve që duhet të ftoni për të garantuar që e gjithë salla do çohet në këmbë.

Referenca: https://code.google.com/codejam/contest/6224486/dashboard#s=p0

Shembull

```
$ cat input.txt
4
4 11111
1 09
5 110011
0 1

$ python3 prog.py < input.txt
Case #1: 0
Case #2: 1
Case #3: 2
Case #4: 0
```

Për çdo rast testimi jepet niveli më i lartë i turpit, dhe pastaj jepet një varg numrin e spektatorëve për secilin nivel, duke filluar nga 0. P.sh. “409” do të thotë që 4 persona kanë nivel turpi 0, asnjë me nivel turpi 1, dhe 9 me nivel turpi 2.

Në rastin e parë e gjithë salla do çohet në këmbë pa shtuar asnjë të ftuar tjetër.

Në rastin e dytë, duhet të ftojmë një person me nivel turpi 0, dhe pastaj 9 të tjerët do çohen në këmbë, meqë e kanë nivelin 1.

Në rastin e tretë mund të ftojmë një person me nivel 2 dhe një me nivel 3, ose 2 persona me nivel 2. Të dyja këto raste bëjnë që edhe spektatori me nivel 4 të çohet në këmbë, dhe pas atij edhe spektatori me nivel 5.

Problemi 080

Kërkesa

Trajneri i futbollit të një shkolle duhet të zgjedhë **P** nxënës për të përfaqësuar shkollën në kampionat. Shkolla ka **N** nxënës dhe aftësitë e secilit në futboll shënohen me një numër natyror S_i.

Trajneri mendon se skuadra duhet të jetë e niveluar, dmth të ketë **P** nxënës me të njëjtin nivel aftësish, sepse kështu do të luajnë të gjithë si skuadër. Në fillim mund të mos jetë e mundur që të gjenden **P** nxënës me aftësi të barabarta, kështu që trajnerit do ti duhet ti stërvisë individualisht disa nxënës, për të rritur aftësitë e tyre. Me një orë stërvitje aftësia e një nxënësi mund të ngrihet me 1 pikë.

Sa është minimumi i orëve të stërvitjes që duhen për të krijuar një skuadër të niveluar.

Referenca: https://codingcompetitions.withgoogle.com/kickstart/round/0000000000050e01/00000000000698d6

Shembull

```
$ cat input.txt
3
4 3
3 1 9 100
6 2
5 5 1 2 3 4
5 5
7 7 1 7 7

$ python3 prog.py < input.txt
Case #1: 14
Case #2: 0
Case #3: 6
```

Jepen **N** dhe **P**, dhe më pas aftësitë e çdo nxënësi.

Në rastin e parë mund të trajnohet për 6 orë nxënësi i parë dhe për 8 orë nxënësi i dytë, dhe atere tre nxënësit e parë do kenë të njëjtin nivel aftësish.

Në rastin e dytë, dy nxënësit e parë kanë të njëjtin nivel aftësish, kështu që nuk është nevoja për trajnime shtesë.

Në rastin e tretë, nxënësi i tretë duhet trajnuar për 6 orë, dhe kështu të 5 nxënësit do kenë të njëjtin nivel.

Zgjidhja 1

```
for t in range(int(input())):
    n, p = map(int, input().split())
    L = [int(i) for i in input().split()]
    L.sort()
    c = 0
    for i in range(p):
        c += L[p-1] - L[i]
    c_min = c
    for j in range(p, n):
        c = 0
        for i in range(j-p+1, j):
            c += L[j] - L[i]
        if c < c_min:
            c_min = c
    print('Case #{}: {}'.format(t+1, c_min))
```

Sqarime

Në fillim i rendisim nxënësit sipas aftësive dhe pastaj gjejmë ata **P** nxënës të njëpasnjëshëm që kanë nevojë për më pak trajnim. Këtë e bëjmë duke i kontrolluar të gjitha nënvargjet e njëpasnjëshme me **P** elementë. Meqënëse kemi dy cikle brenda njëri-tjetrit, koha e kësaj zgjidhje është e rendit $O(N * P)$.

Zgjidhja 2

```
for t in range(int(input())):
    n, p = map(int, input().split())
    L = [int(i) for i in input().split()]
    L.sort()
    s = 0
    for i in range(p):
        s += L[i]
    c_min = p*L[p-1] - s
    for i in range(p, n):
        s += L[i]
        s -= L[i-p]
        c = p*L[i] - s
        if c < c_min:
```

```
            c_min = c
    print('Case #{}: {}'.format(t+1, c_min))
```

Sqarime

Koha e trajnimit për një nënvarg nga **i-p+1** në **i** është `(L[i] - L[i-1]) + (L[i] - L[i-2]) + . . . + (L[i] - L[i-p+1])` ose `p*L[i] - (L[i-p+1] + L[i-p+2] + . . . + L[i-1] + L[i])`. Shumën në kllapa mund ta llogarisim në mënyrë inkrementale, kështu që nuk është nevoja të bëjmë një cikël tjetër për të llogaritur koston e një nënvargu. Kjo bën që koha e kësaj zgjidhje të jetë e rendit $O(N)$, që është më e mirë se zgjidhja e parë.

Detyra

Jepet një varg me zero-njësha. Gjeni numrin e nënvargjeve që fillojnë dhe mbarojnë me **1**.

Referenca: https://www.codechef.com/problems/CSUB

Shembull

```
$ cat input.txt
2
4
1111
5
10001

$ python3 prog.py < input.txt
10
3
```

10 Problemat 081-090

Problemi 081

Kërkesa

Në një kompani janë **N** punonjës, pagat e të cilëve janë W_i (për **i** nga **1** në **N**). Një ditë pronari vendosi që tua bëjë rrogat të barabarta të gjithëve. Por për të arritur këtë qëllim ai përdor një veprim të tillë: Zgjedh një punonjës, dhe rrit me 1 rrogën e gjithë punonjësve të tjerë (por jo të këtij që ka zgjedhur). Sigurisht që punonjësi i zgjedhur mund të jetë i ndryshëm për secilin veprim. Sa është numri më i vogël i veprimeve të tilla që duhen kryer deri sa të gjithë punonjësit të kenë rroga të barabarta?

Referenca: https://www.codechef.com/problems/SALARY

Shembull

```
$ cat input.txt
2
3
1 2 3
2
42 42

$ python3 prog.py < input.txt
3
0
```

Në rastin e parë, fillimisht zgjidhet punonjësi i tretë, dhe pagat bëhen (2, 3, 3). Pastaj mund të zgjidhet punonjësi i dytë, dhe pagat bëhen (3, 3, 4). Pastaj mund të zgjidhet punonjësi i tretë, dhe pagat bëhen (4, 4, 4). Pra duhen 3 veprime për ti barazuar.

Në rastin e dytë janë tashmë të barabarta, kështu që nuk është nevoja për asnjë veprim.

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    W = [int(i) for i in input().split()]
    nr = 0
    W.sort(reverse=True)
    while W[0] > W[-1]:
        nr += 1
        for i in range(1, n):
            W[i] += 1
        i = 0
        while W[i] < W[i+1]:
            W[i], W[i+1] = W[i+1], W[i]
            i += 1
    print(nr)
```

Sqarime

Për të bërë sa më pak veprime, duhet të zgjdhim gjithmonë punonjësin që ka pagën më të lartë (ose një nga ata që kanë pagën më të lartë), dhe të rrisim pagat e të tjerëve me 1.

Për ta lehtësuar këtë proces, listën e pagave e mbajmë të renditur në rendin zbritës. Meqënëse lista është e renditur, më i madhi është gjithmonë i pari në listë, kështu që në çdo hap mjafton të rrisim të tjerët përveç të parit. Pasi kemi bërë një hap, renditjen e listës mund ta rivendosim duke zhvendosur elementin e parë drejt fundit, deri sa të ndodhet në vendin e duhur sipas renditjes.

Duke qënë se lista është e renditur, edhe kontrollin nëse të gjithë elementët e listës janë të barabartë mund ta bëjmë kollaj: mjafton që të krahasojmë elementin e parë me të fundit.

Kjo zgjidhje është llogjikisht e thjeshtë, por për numra të mëdhenj është shumë e ngadaltë dhe nuk e kalon testin (https://www.codechef.com/submit/SALARY).

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    W = [int(i) for i in input().split()]
    print(sum(W) - n*min(W))
```

Sqarime

Mund të vëmë re që veprimi që kryhet është i njëvlershëm me këtë veprim: Zgjedhim një punonjës dhe i zbresim 1 pagës së tij, pastaj u shtojmë 1 të gjithë punonjësve. Meqënëse ne interesohemi që ti bëjmë rrogat të barabarta, pjesën e dytë të këtij veprimi (shtojmë 1 për të gjithë punonjësit) mund të mos ta marrim parasysh.

Atere problemi shndërrohet në këtë formë: Zgjedhim një punonjës dhe i zbresim 1 pagës së tij. Sa herë duhet ta përsërisim këtë veprim deri sa të gjitha rrogat të bëhen të barabarta.

Është e qartë se ne mund të zgjedhim një punonjës çfarëdo dhe ti zbresim 1 deri sa paga e tij të bëhet e barabartë me pagën më të vogël, pastaj të zgjedhim një tjetër, e kështu me rradhë, deri sa të gjitha pagat të bëhen të barabarta me pagën më të vogël. Kështu që numri total i veprimeve është $(W_1 - W_{min}) + (W_2 - W_{min}) + ... + (W_n - W_{min})$, që është `sum(W) - n*min(W)`.

Detyra

Në një varg numrash natyrorë $A_1, A_2, ..., A_n$ gjeni vlerën më të madhe të shprehjes $A_i mod A_j$ (mbetja e pjesëtimit të A_i me A_j) për çdo vlerë të **i** dhe **j**.

Referenca: https://www.codechef.com/APRIL19B/problems/MAXREM

Shembull

```
$ cat input.txt
2
5
1 2 3 4 5
6
5 5 5 2 3 8

$ python3 prog.py < input.txt
4
5
```

Problemi 082

Kërkesa

Në një listë me numra natyrorë bëjmë këtë veprim: Zgjedhim disa prej numrave të listës dhe i fshijmë nga lista. Si kosto të këtij veprimi përkufizojmë rezultatin e kryerjes së një

Bitwise OR midis elementeve të fshira. Këtë veprim e përsërisim deri sa në listë të mos ngelet më asnjë element. Kostoja e përgjithshme është shuma e kostos së çdo veprimi. Gjeni vlerën më të vogël të mundshme të kostos së përgjithshme.

Referenca: https://www.codechef.com/problems/CHNGOR

Shembull

```
$ cat input.txt
2
2
1 2
3
1 3 6

$ python3 prog.py < input.txt
3
7
```

Zgjidhja 1

```python
for _ in range(int(input())):
    input()
    c = 0
    for i in input().split():
        c |= int(i)
    print(c)
```

Sqarime

Veprimi **bitwise OR** merr paraqitjen binare të dy numrave dhe bën veprimin llogjik **OR** për çiftet korresponduese të biteve. P.sh. `3 = 0011`, `5 = 0101`, `3 OR 5 = 0011 | 0101 = 0111 = 7`. Dmth nqs njëri nga bitet korresponduese është **1** (ose të dy janë **1**) rezultati del **1**. Nqs të dy janë **0** rezultati del **0**.

Le ta zëmë se kemi 2 numra të listës që kanë **1** në të njëjtin pozicion. Nqs këta numra do ishin fshirë në veprime të ndryshme, vlera përkatëse që paraqet ky njësh (që është një nga fuqitë e dyshit) do ti shtohej kostos së përgjithshme dy herë. Kurse po të fshihen nga lista në të njëjtin veprim, kjo vlerë i shtohet kostos totale vetëm një herë, kështu që kostoja totale do jetë më e vogël se në rastin e parë.

Kjo do të thotë që vlera minimale e kostos së përgjithshme arrihet kur kryhet vetëm një veprim, i cili i fshin të gjithë numrat e listës.

Zgjidhja 2

```
from functools import reduce
for _ in range(int(input())):
    input()
    L = [int(i) for i in input().split()]
    print(reduce(lambda x,y: x|y, L))
```

Zgjidhja 3

```
from functools import reduce
from operator import __or__
for _ in range(int(input())):
    input()
    L = [int(i) for i in input().split()]
    print(reduce(__or__, L))
```

Detyra

Na jepet një varg **S** me gjatësi **N** dhe një shkronjë **X**. Gjeni numrin e nënvargjeve të **S** që e përmbajnë shkronjën **X** të paktën një herë.

Referenca: https://www.codechef.com/APRIL19B/problems/STRCH

Shembull

```
$ cat input.txt
2
3
abb b
6
abcabc c

$ python3 prog.py < input.txt
5
15
```

Në rastin e parë, vargu `abb` ka gjashtë nënvargje: `a`, `b`, `b`, `ab`, `bb`, `abb`. Nënvargjet që përmbajnë `b` janë: `b`, `b`, `ab`, `bb`, `abb`.

Problemi 083

Kërkesa

Në koncertin e operas këngëtarja kryesore është një mikja juaj. Ju doni të siguroheni që në fund të shfaqjes e gjithë salla të ngrihet në këmbë dhe të duartrokasë.

Në fillim spektatorët janë të ulur. Çdo spektator ka një nivel turpi. Një spektator me nivel turpi S_i pret deri sa të paktën S_i spektatorë të tjerë të jenë çuar në këmbë duke duartrokitur, pastaj çohet edhe ai. Nëse $S_i = 0$ atere spektatori çohet gjithmonë në këmbë për të duartrokitur, pavarësisht nëse është çuar ndonjë tjetër. Nëse një spektator e ka $S_i = 2$ atere pret deri sa të çohen të paktën 2 të tjerë, para se të çohet.

Ju e njihni nivelin e turpit të spektatorëve që do shkojnë në koncert dhe mund të ftoni edhe njerëz të tjerë me një nivel turpi të çfarëdoshëm, jo domosdoshmërisht të njëjtë. Sa është numri më i vogël i njerëzve që duhet të ftoni për të garantuar që e gjithë salla do çohet në këmbë.

Referenca: https://code.google.com/codejam/contest/6224486/dashboard#s=p0

Shembull

```
$ cat input.txt
4
4 11111
1 09
5 110011
0 1

$ python3 prog.py < input.txt
Case #1: 0
Case #2: 1
Case #3: 2
Case #4: 0
```

Për çdo rast testimi jepet niveli më i lartë i turpit, dhe pastaj jepet një varg numrin e spektatorëve për secilin nivel, duke filluar nga 0. P.sh. “409” do të thotë që 4 persona kanë nivel turpi 0, asnjë me nivel turpi 1, dhe 9 me nivel turpi 2.

Në rastin e parë e gjithë salla do çohet në këmbë pa shtuar asnjë të ftuar tjetër.

Në rastin e dytë, duhet të ftojmë një person me nivel turpi 0, dhe pastaj 9 të tjerët do çohen në këmbë, meqë e kanë nivelin 1.

Në rastin e tretë mund të ftojmë një person me nivel 2 dhe një me nivel 3, ose 2 persona me nivel 2. Të dyja këto raste bëjnë që edhe spektatori me nivel 4 të çohet në këmbë, dhe pas atij edhe spektatori me nivel 5.

Zgjidhja 1

```
for t in range(int(input())):
    _, audience = input().split()
    extra = 0
    invite = 0
    for i in range(len(audience)):
        if audience[i] == '0':
            if extra > 0:
                extra -= 1
            else:
                invite += 1
        else:
            extra += int(audience[i]) - 1
    print('Case #{}: {}'.format(t+1, invite))
```

Sqarime

Në ndryshoren `invite` mbajmë numrin e personave që duhen ftuar. Në ndryshoren `extra` mbajmë numrin e personave që janë më tepër se sa duhen për të ngritur në këmbë personat e nivelit **i**.

Zgjidhja bazohet në faktin që po të kemi të paktën një person për çdo nivel, atere të gjithë do ngrihen në këmbë. Nqs kemi më shumë se 1 person për një nivel të caktuar, atere personat "e tepërt" i shtojmë te `extra`. Në rast se kemi 0 persona për një nivel të caktuar, atere këtë mund ta kompensojmë me një person nga ata që janë te `extra`, por nqs edhe vetë `extra` është bosh (0), atere për të zgjidhur situatën duhet të ftojmë një person tjetër.

Zgjidhja 2

```
for t in range(int(input())):
    _, audience = input().split()
    up = 0
    invite = 0
    for i in range(len(audience)):
        invite = max(invite, i - up)
        up += int(audience[i])
    print('Case #{}: {}'.format(t+1, invite))
```

Sqarime

Në ndryshoren `up` mbajmë shënim personat që janë çuar deri në një nivel të caktuar (dhe e shtojmë për çdo nivel). Nqs ky numër është më i vogël se niveli **i**, atere na duhet të ftojmë `(i - up)` persona të tjerë, në mënyrë që këta të nivelit **i** të çohen në këmbë. Personat e ftuar mund të jenë të gjithë të nivelit 0, dmth që çohen menjëherë në këmbë. Meqënëse për çdo nivel **i** llogarisim numrin e personave që duhet të ftojmë për këtë nivel, atere mjafton që të ftojmë maksimumin e këtyre numrave, dhe kjo do bëjë që personat e çdo niveli të ngihen në këmbë. Pra ne gjejmë maksimumin e vlerës për shprehjen `(i - up)`.

Detyra

Një arë është e ndarë në parcela, të cilat formojnë **N** rreshta (nga 1 në N) dhe **M** shtylla (nga 1 në M). Nga këto, vetëm **K** parcela janë të mbjella, kurse të tjerat janë me barishte. Dy parcela quhen fqinje nëse kanë një anë të përbashkët. Duam të ndërtojmë një gardh që rrethon parcelat e mbjella, në mënyrë të tillë që të mos ketë gardh midis dy parcelave të mbjella por vetëm midis një parcele të mbjellë dhe një të pambjellë, ose midis një parcele të mbjellë dhe pjesës jashtë arës. Sa është numri më i vogël i anëve ku duhet ndërtuar gardh?

Referenca: https://www.codechef.com/APRIL19B/problems/FENCE

Shembull

```
$ cat input.txt
2
4 4 9
1 4
2 1
2 2
2 3
3 1
3 3
4 1
4 2
4 3
4 4 1
1 1

$ python3 prog.py < input.txt
20
4
```

Për çdo rast testimi jepen **N**, **M** dhe **K**. Më pas vijnë **K** rreshta me nga dy numra **r** dhe **s**, që janë rreshti dhe shtylla e një parcele të mbjellë.

Në rastin e parë fusha duket e tillë (ku me **x** është shënuar një parcelë e mbjellë dhe me **-** një parcelë e pambjellë:

```
---x
xxx-
x-x-
xxx-
```

Një zgjidhje optimale është që të ndërtohet gardh rreth parcelës lart në cep (4 anë), pastaj rreth grupit të parcelave të mbjella poshtë (12 anë), si dhe rreth parcelës së pambjellë në qendër (4 anë). Gjithsej 4 + 12 + 4 = 20 anë.

Problemi 084

Kërkesa

Në një varg numrash natyrorë $A_1, A_2, ..., A_n$ gjeni vlerën më të madhe të shprehjes $A_i mod A_j$ (mbetja e pjesëtimit të A_i me A_j) për çdo vlerë të **i** dhe **j**.

Referenca: https://www.codechef.com/APRIL19B/problems/MAXREM

Shembull

```
$ cat input.txt
2
5
1 2 3 4 5
6
5 5 5 2 3 8

$ python3 prog.py < input.txt
4
5
```

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    L = [int(i) for i in input().split()]
    m = 0
    for i in range(n):
        for j in range(n):
            if L[i] % L[j] > m:
                m = L[i] % L[j]
    print(m)
```

Sqarime

Zgjidhja naive, duke i gjetur të gjitha mbetjet e mundshme dhe duke marrë maksimumin e tyre, është me kohë të rendit $O(N^2)$, dhe shumë e ngadalshme për vlera të mëdhaja të N.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    L = [int(i) for i in input().split()]
    L.sort()
    m = 0
    for i in range(n-1, 0, -1):
        m = L[i-1] % L[i]
        if m > 0:
            break
    print(m)
```

Sqarime

Vëmë re që nëse $a < b$, atere `a % b = a`. Kështu që për të gjetur mbetjen më të madhe mjafton që ti rendisim numrat dhe të marrin numrin e parafundit. Vetëm se kjo nuk funksionon nëse dy numrat e fundit qëllon të jenë të barabartë. Në këtë rast duhet marrë numri i parë që nuk është i barabartë me numrin e fundit. Gjithashtu duhet të kemi parasysh edhe rastin që të gjithë numrat mund të jenë të barabartë.

Koha mbizotëruese në këtë zgjidhje është koha që duhet për të renditur numrat, e cila zakonisht është e rendit $O(N * log(N))$, pra më e mirë se zgjidhja e parë.

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    L = [int(i) for i in input().split()]
    L.sort()
    i = n - 2
    while i > 0 and L[i] == L[i+1]:
        i -= 1
    print(L[i] % L[i+1])
```

Sqarime

E njëjtë si zgjidhja 2, vetëm se e shkruar pak më ndryshe.

Detyra

Rregjistrimi në shkollën e lartë bëhet me anë të konkursit. Gjithsej zhvillohen **E** provime, ku pikët maksimale në çdo provim janë **M**, dhe merret shuma e pikëve të tyre. Nga **N** studentë që hyjnë në provime, vetëm **K** prej tyre që kanë marrë në total më shumë pikë se të tjerët mund të rregjistrohen.

Ju i dini pikët që kanë marrë konkurentët e tjerë në **E-1** provimet e mëparshme. Gjithashtu, me një program të inteligjencës artificiale keni parashikuar edhe pikët që do marrin konkurentët e tjerë në provimin e fundit. Gjeni se sa janë pikët minimale që duhet të merrni në provimin e fundit, në mënyrë që të hyni në shkollë të lartë. Nëse kjo është e pamundur programi duhet të nxjerrë 'Impossible'.

Referenca: https://www.codechef.com/problems/ENTEXAM

Shembull

```
$ cat input.txt
1
4 2 3 10
7 7 7
4 6 10
7 10 9
9 9

$ python3 prog.py < input.txt
4
```

Kemi 1 rast testimi. Rreshti i parë na jep numrat N=4, K=2, E=3, M=10. Katër rreshtat e tjerë na japin pikët që kanë marrë studentët e tjerë në E=3 provimet që janë zhvilluar, me përjashtim të rreshtit të fundit, i cili jep pikët tuaja në E-1 provimet e kaluara. Pikët totale të konkurentëve të tjerë janë 7+7+7=21, 4+6+10=20, dhe 7+10+9=26. Meqënëse vetëm K=2 veta do jenë fitues, juve ju duhen të paktën 22 pikë për të fituar. Deri tani keni marrë 9+9=18 pikë, kështu që në provimin e fundit duhet të merrni të paktën **4** pikë.

Problemi 085

Kërkesa

Na jepet një varg **S** me gjatësi **N** dhe një shkronjë **X**. Gjeni numrin e nënvargjeve të **S** që e përmbajnë shkronjën **X** të paktën një herë.

Referenca: https://www.codechef.com/APRIL19B/problems/STRCH

Shembull

```
$ cat input.txt
2
3
abb b
6
abcabc c

$ python3 prog.py < input.txt
5
15
```

Në rastin e parë, vargu `abb` ka gjashtë nënvargje: `a`, `b`, `b`, `ab`, `bb`, `abb`. Nënvargjet që përmbajnë `b` janë: `b`, `b`, `ab`, `bb`, `abb`.

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    S, x = input().split()
    nr = n*(n + 1)//2
    m = 0
    for c in S+x:
```

```
        if c == x:
            nr -= m*(m + 1)//2
            m = 0
        else:
            m += 1
    print(nr)
```

Sqarime

Nqs një nënvarg fillon te shkronja e parë, mund ta ketë fundin te secila nga **n** shkronjat. Nqs fillon te shkronja e dytë mund ta ketë fundin te secila nga **(n-1)** shkronjat nga e dyta deri në fund. Kështu që gjithsej, një varg me gjatësi **n** ka $n+(n-1)+(n-2)+...+2+1$ nënvargje. Vlera e kësaj shume mund të llogaritet me formulën: `n*(n + 1)//2`.

Nqs nga të gjitha nënvargjet e mundshme heqim ato që nuk përmbajnë **X**, na ngelen ato që përmbajnë të paktën një **X**. Për të gjetur numrin e nënvargjeve që nuk përmbajnë **X**, gjejmë segmentet që nuk përmbajnë **X**. Nqs një segment i tillë e ka gjatësinë **m**, atere ka `m*(m + 1)//2` nënvargje brenda tij.

Detyra

Cufi duhet të paguajë për qeranë e shtëpisë 1000 lekë çdo muaj. Por nqs pagesën e bën me vonesë (nuk ka rëndësi se sa vonë) duhet të paguajë edhe 100 lekë gjobë. Ai ka jetuar për **N** muaj në këtë shtëpi dhe tani duhet të shkojë diku tjetër, kështu që i duhet të shlyejë të gjitha pagesat e prapambetura (bashkë me gjobat).

Nga veprimet që ka bërë në llogarinë bankare ai mund të shikojë se disa muaj ka bërë pagesa 1000 lekëshe për qeranë, por asnjëherë nuk ka shlyer ndonjë gjobë. Ky informacion (se në cilin muaj ka paguar dhe në cilin jo) na jepet në formën e një vargu me **N** zero dhe njësha (zero kur nuk ka paguar dhe njësh kur ka paguar). Por nqs nuk ka paguar muajin e parë dhe ka paguar muajin e dytë, kjo pagesë i shkon si pagesë e muajit të parë, dhe është pagesë e vonuar, kështu që prapë i ngelet për të paguar gjobën e muajit të parë. Po kështu edhe muaji i dytë konsiderohet si pagesë e vonuar, kështu që edhe për muajin e dytë duhet të paguajë gjobë.

Gjeni se sa duhet paguar për ti shlyer të gjitha detyrimet.

Referenca: https://www.codechef.com/problems/CHEFAPAR

Shembull

```
$ cat input.txt
4
2
```

```
1 1
2
0 0
3
0 1 0
2
0 1

$ python3 prog.py < input.txt
0
2200
2300
1200
```

1. S'ka lënë asnjë pagesë pa paguar.
2. S'ka paguar asnjë nga 2 muajt. Bashkë me gjobat duhet të paguajë 2200.
3. Ka bërë një pagesë muajin e dytë. Ka për të paguar edhe muajin e parë dhe të tretë, si dhe 3 gjoba. Gjithsej 2300.
4. Ka bërë një pagesë muajin e dytë. Ka për të paguar muajin e parë dhe 2 gjoba, gjithsej 1200.

Problemi 086

Kërkesa

Një arë është e ndarë në parcela, të cilat formojnë **N** rreshta (nga 1 në N) dhe **M** shtylla (nga 1 në M). Nga këto, vetëm **K** parcela janë të mbjella, kurse të tjerat janë me barishte. Dy parcela quhen fqinje nëse kanë një anë të përbashkët. Duam të ndërtojmë një gardh që rrethon parcelat e mbjella, në mënyrë të tillë që të mos ketë gardh midis dy parcelave të mbjella por vetëm midis një parcele të mbjellë dhe një të pambjellë, ose midis një parcele të mbjellë dhe pjesës jashtë arës. Sa është numri më i vogël i anëve ku duhet ndërtuar gardh?

Referenca: https://www.codechef.com/APRIL19B/problems/FENCE

Shembull

```
$ cat input.txt
2
4 4 9
1 4
```

```
2 1
2 2
2 3
3 1
3 3
4 1
4 2
4 3
4 4 1
1 1

$ python3 prog.py < input.txt
20
4
```

Për çdo rast testimi jepen **N**, **M** dhe **K**. Më pas vijnë **K** rreshta me nga dy numra **r** dhe **s**, që janë rreshti dhe shtylla e një parcele të mbjellë.

Në rastin e parë fusha duket e tillë (ku me **x** është shënuar një parcelë e mbjellë dhe me **-** një parcelë e pambjellë:

```
---x
xxx-
x-x-
xxx-
```

Një zgjidhje optimale është që të ndërtohet gardh rreth parcelës lart në cep (4 anë), pastaj rreth grupit të parcelave të mbjella poshtë (12 anë), si dhe rreth parcelës së pambjellë në qendër (4 anë). Gjithsej $4 + 12 + 4 = 20$ anë.

Zgjidhja 1

```
for _ in range(int(input())):
    n, m, k = map(int, input().split())
    F = [[False for j in range(m+2)] for i in range(n+2)]
    while k > 0:
        r, c = map(int, input().split())
        F[r][c] = True
        k -= 1
    nr = 0
    for i in range(n+1):
        for j in range(m+1):
```

```
            if F[i][j]:
                if not F[i-1][j]:
                    nr += 1
                if not F[i+1][j]:
                    nr += 1
                if not F[i][j-1]:
                    nr += 1
                if not F[i][j+1]:
                    nr += 1
    print(nr)
```

Sqarime

Në këtë zgjidhje ne ndërtojmë një *matricë* si një listë me **N+2** lista, ku secila listë ka **M+2** vlera. Elementët e kësaj matrice fillimisht janë `False` për të treguar një parcelë të pambjellë, dhe pastaj i bëjmë `True` për parcelat e mbjella.

Përmasat e matricës i kemi zgjedhur pak më të mëdhaja se madhësia e fushës për dy arsye. E para na ndihmon që ti kapim parcelat direkt, pa qenë nevoja të konvertojmë nga indekset me bazë 1 (që na jep problemi) në indekset me bazë 0 (siç i ka Python-i). Arsyeja e dytë është se na lehtëson edhe kontrollin për parcelat që janë në kufi të arës.

Për të gjetur numrin e gardheve, i marrim me rradhë parcelat, dhe për çdo parcelë të mbjellë shikojmë parcelat fqinje me të (majtas, djathtas, lartë, poshtë). Për secilën nga këto parcela që është e pambjellë duhet të ndërtojmë një gardh.

Koha e kësaj zgjidhje është e rendit $O(N * M)$, dhe po kështu edhe hapësira është e rendit $O(N * M)$. Për të dhëna me përmasa të mëdha kjo zgjidhje nuk është dhe aq efiçente.

Zgjidhja 2

```
for _ in range(int(input())):
    n, m, k = map(int, input().split())
    F = []
    while k > 0:
        r, c = map(int, input().split())
        F.append((r, c))
        k -= 1
    nr = 0
    for x in F:
        r, c = x
```

```
        if (r+1, c) not in F:
            nr += 1
        if (r-1, c) not in F:
            nr += 1
        if (r, c+1) not in F:
            nr += 1
        if (r, c-1) not in F:
            nr += 1
    print(nr)
```

Sqarime

Në këtë zgjidhje, në vend që të ndërtojmë një matricë, e mbajmë informacionin për parcelat e mbjella në një listë. Për parcelat fqinje shikojmë nëse ndodhen në këtë listë ose jo. Nëse nuk ndodhen atere janë të pambjella.

Koha e kësaj zgjidhje është e rendit $O(K * K)$ dhe hapësira e rendit $O(K)$. Nëse matrica është e rrallë (nuk ka shumë parcela të mbjella, **K** është shumë e vogël në krahasim me **N*M**), kjo zgjidhje mund të jetë më efiçente se zgjidhja e parë. Por në përgjithësi edhe kjo zgjidhje nuk është mjaft e shpejtë.

Zgjidhja 3

```
for _ in range(int(input())):
    n, m, k = map(int, input().split())
    F = {}
    while k > 0:
        r, c = map(int, input().split())
        F[(r, c)] = True
        k -= 1
    nr = 0
    for x in F:
        r, c = x
        if not F.get((r+1, c), False):
            nr += 1
        if not F.get((r-1, c), False):
            nr += 1
        if not F.get((r, c+1), False):
            nr += 1
        if not F.get((r, c-1), False):
            nr += 1
    print(nr)
```

Sqarime

Në këtë zgjidhje listën e kemi zëvendësuar me një fjalor (dictionary). Kjo strukturë është shumë më e shpejtë se lista kur kërkon nëse një element ndodhet në të ose jo (listës i duhet një kohë e rendit $O(K)$ për këtë veprim, kurse fjalori e bën në kohë konstante). Si rezultat, koha e kësaj zgjidhje është e rendit $O(K)$ dhe hapësira e rendit $O(K)$. Pra në përgjithësi është më e mirë se zgjidhja e dytë (kur **K** është shumë më e vogël se **N*M**).

Detyra

Fituesit të një olimpiade programimi i takon të marrë si shpërblim një çek me vlerën **N**. Mirëpo të paktën një nga shifrat e numrit **N** është **4**, ndërkohë që tastiera e makinës që shkruan çekun e ka të prishur tastin me numrin **4**. Megjithatë organizatorët e gjetën zgjidhjen duke shkruar dy çeqe, shuma e të cilëve është **N**, dhe asnjëri prej të cilëve nuk e përmban shifrën **4**. Gjeni dy numra të tillë.

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000051705/0000000000088231

Shembull

```
$ cat input.txt
3
4
940
4444

$ python3 prog.py < input.txt
Case #1: 2 2
Case #2: 852 88
Case #3: 667 3777
```

Problemi 087

Kërkesa

Rregjistrimi në shkollën e lartë bëhet me anë të konkursit. Gjithsej zhvillohen **E** provime, ku pikët maksimale në çdo provim janë **M**, dhe merret shuma e pikëve të tyre. Nga **N** studentë që hyjnë në provime, vetëm **K** prej tyre që kanë marrë në total më shumë pikë se të tjerët mund të rregjistrohen.

Ju i dini pikët që kanë marrë konkurentët e tjerë në **E-1** provimet e mëparshme. Gjithashtu, me një program të inteligjencës artificiale keni parashikuar edhe pikët që do marrin konkurentët e tjerë në provimin e fundit. Gjeni se sa janë pikët minimale që duhet të merrni në provimin e fundit, në mënyrë që të hyni në shkollë të lartë. Nëse kjo është e pamundur programi duhet të nxjerrë 'Impossible'.

Referenca: https://www.codechef.com/problems/ENTEXAM

Shembull

```
$ cat input.txt
1
4 2 3 10
7 7 7
4 6 10
7 10 9
9 9

$ python3 prog.py < input.txt
4
```

Kemi 1 rast testimi. Rreshti i parë na jep numrat N=4, K=2, E=3, M=10. Katër rreshtat e tjerë na japin pikët që kanë marrë studentët e tjerë në E=3 provimet që janë zhvilluar, me përjashtim të rreshtit të fundit, i cili jep pikët tuaja në E-1 provimet e kaluara. Pikët totale të konkurentëve të tjerë janë 7+7+7=21, 4+6+10=20, dhe 7+10+9=26. Meqënëse vetëm K=2 veta do jenë fitues, juve ju duhen të paktën 22 pikë për të fituar. Deri tani keni marrë 9+9=18 pikë, kështu që në provimin e fundit duhet të merrni të paktën **4** pikë.

Zgjidhja

```
for _ in range(int(input())):
    N, K, E, M = map(int, input().split())
    P = []
    for i in range(N):
        P.append(sum([int(p) for p in input().split()]))
    p = P[-1]
    del(P[-1])
    P.sort(reverse=True)
    p1 = max(0, P[K-1] - p + 1)
    print(p1) if p1<=M else print('Impossible')
```

Sqarime

Shumën e pikëve të studentëve të tjerë e vendosim në një listë dhe e rendisim në rendin zbritës. Ai që ndodhet i K-ti në këtë listë do jetë i fundit që do rregjistrohet. Për tia kaluar atij ju duhet të merrni 1 pikë më tepër.

Detyra

Labirinti më i thjeshtë në botë përbëhet nga një tabelë me përmasa **NxN**. Duke filluar në cepin e majtë sipër, duhet vajtur në cepin e djathtë poshtë, duke bërë vetëm lëvizje majtas (**E**ast) ose poshtë (**S**outh).

Na jepen përmasat e tabelës dhe zgjidhja që ka bërë Lida. A mund të gjeni një zgjidhje tjetër që është komplet e ndryshme nga ajo që ka bërë Lida (dmth nqs zgjidhja e Lidës përmban një kalim nga qeliza A në qelizën B, zgjidhja juaj nuk duhet të përmbajë një kalim të tillë).

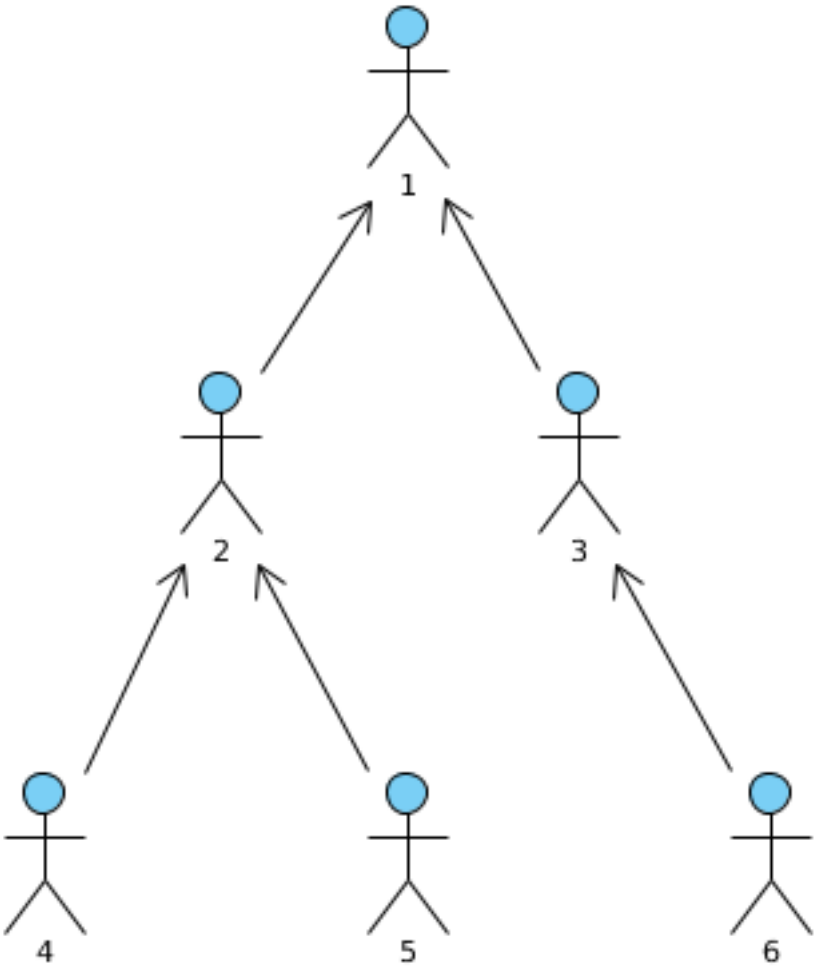
Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000051705/00000000000881da

Shembull

```
$ cat input.txt
2
2
SE
5
EESSSESE

$ python3 prog.py < input.txt
Case #1: ES
Case #2: SEEESSES
```

Rasti i dytë është si në figurë:



Problemi 088

Kërkesa

Cufi duhet të paguajë për qeranë e shtëpisë 1000 lekë çdo muaj. Por nqs pagesën e bën me vonesë (nuk ka rëndësi se sa vonë) duhet të paguajë edhe 100 lekë gjobë. Ai ka jetuar për **N** muaj në këtë shtëpi dhe tani duhet të shkojë diku tjetër, kështu që i duhet të shlyejë të gjitha pagesat e prapambetura (bashkë me gjobat).

Nga veprimet që ka bërë në llogarinë bankare ai mund të shikojë se disa muaj ka bërë pagesa 1000 lekëshe për qeranë, por asnjëherë nuk ka shlyer ndonjë gjobë. Ky infor-

macion (se në cilin muaj ka paguar dhe në cilin jo) na jepet në formën e një vargu me **N** zero dhe njësha (zero kur nuk ka paguar dhe njësh kur ka paguar). Por nqs nuk ka paguar muajin e parë dhe ka paguar muajin e dytë, kjo pagesë i shkon si pagesë e muajit të parë, dhe është pagesë e vonuar, kështu që prapë i ngelet për të paguar gjobën e muajit të parë. Po kështu edhe muaji i dytë konsiderohet si pagesë e vonuar, kështu që edhe për muajin e dytë duhet të paguajë gjobë.

Gjeni se sa duhet paguar për ti shlyer të gjitha detyrimet.

Referenca: https://www.codechef.com/problems/CHEFAPAR

Shembull

```
$ cat input.txt
4
2
1 1
2
0 0
3
0 1 0
2
0 1

$ python3 prog.py < input.txt
0
2200
2300
1200
```

1. S'ka lënë asnjë pagesë pa paguar.
2. S'ka paguar asnjë nga 2 muajt. Bashkë me gjobat duhet të paguajë 2200.
3. Ka bërë një pagesë muajin e dytë. Ka për të paguar edhe muajin e parë dhe të tretë, si dhe 3 gjoba. Gjithsej 2300.
4. Ka bërë një pagesë muajin e dytë. Ka për të paguar muajin e parë dhe 2 gjoba, gjithsej 1200.

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
```

```
    M = input().split()
    s = 0
    j = 0
    for i in range(n):
        if M[i] == '1':
            if i > j:
                s += 100
            j += 1
    while j < n:
        s += 1000 + 100
        j += 1
    print(s)
```

Sqarime

Në fillim gjejmë (te ndryshorja **j**) se sa muaj ka paguar. Nqs këta muaj të paguar kanë qenë me vonesë, llogarisim edhe gjobën. Pastaj, muajt që kanë mbetur pa paguar janë të gjithë të vonuar, kështu që llogarisim për ta edhe pagesën edhe gjobën.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    M = input().split()
    s = 0
    for i in range(n):
        if M[i] == '0':
            s += 1000
    j = 0
    while j < n and M[j] == '1':
        j += 1
    while j < n:
        s += 100
        j += 1
    print(s)
```

Sqarime

Në fillim llogarisim pagesën për çdo muaj të pa paguar (`'0'`). Pastaj gjemë muajin e parë që nuk është paguar, dhe që aty e deri në fund llogarisim gjobat.

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    M = input().split()
    s = M.count('0') * 1000
    try:
        i = M.index('0')
        s += (n - i) * 100
    except:
        pass
    print(s)
```

Sqarime

Numrin e muajve të pa paguar mund ta gjejmë edhe me funksionin `count('0')`. Kurse muajin e parë të pa paguar mund ta gjejmë edhe me funksionin `index('0')`. Vetëm se ky funksion jep një kundërshtim në rast se nuk ka asnjë `'0'` në listë, prandaj kemi përdorur `try...except`.

Zgjidhja 4

```
for _ in range(int(input())):
    n = int(input())
    M = input().split()
    s = M.count('0') * 1000
    i = (M + ['0']).index('0')
    s += (n - i) * 100
    print(s)
```

Sqarime

Në këtë variant e kemi shmangur nevojën për `try...except` duke shtuar artificialisht një `'0'` në fund të listës. Kjo zero nuk na prish punë dhe siguron që funksioni `index()` nuk do jap kundërshtim.

Detyra

Kemi një pemë me **N** nyje (të shënuara me numrat **1** deri në **N**), ku rrënja është nyja **1**. Çdo nyje **i** ka një vlerë A_i (që mund të jetë edhe numër negativ).

Mund të zgjedhim një nyje çfarëdo të pemës dhe të fshijmë gjithë nënpemën (degën) poshtë saj, përfshirë edhe vetë nyjen. Këtë veprim mund ta përsërisim disa ose asnjë herë.

Le të përkufizojmë si **përfitim** shumën e të gjitha vlerave të nyjeve të mbetura në pemë, minus $X * k$, ku **X** është një numër i dhënë paraprakisht, dhe **k** është numri i herëve që është kryer veprimi i përshkruar më sipër. Gjeni vlerën më të madhe të mundshme të përfitimit.

Referenca: https://www.codechef.com/APRIL19B/problems/SUBREM

Shembull

```
$ cat input.txt
1
3 5
1 -5 -10
1 2
2 3

$ python3 prog.py < input.txt
-4
```

Kemi një rast testimi. Në rreshtin e parë jepen numrat **N** dhe **X**. Në rreshtin e dytë jepen **N** vlerat A_i. Në **N-1** rreshtat pasardhës jepen dy numra **u** dhe **v**, që tregojnë se ka një brinjë midis dy nyjeve të shënuara me **u** dhe **v**.

Në rastin e dhënë pema është e tillë: (1)–>(2)–>(3). Duke shënuar edhe vlerat përkatëse të çdo nyje mund ta paraqesim kështu: (1,1)–>(2,-5)–>(3,-10).

Nqs nuk fshijmë asnjë nyje/degë përfitimi do jetë: 1 + -5 + -10 = **-14**. Nqs fshijmë nyjen 3, do jetë: 1 + -5 - 1*5* = ***-9***. *Nqs fshijmë nyjen 3 dhe pastaj fshijmë nyjen 2, do jetë: 1 - 2*5 = **-9**. Nqs fshijmë vetëm nyjen 2 do jetë: 1 - 1*5* = ***-4***. *Nqs fshijmë nyjen 3, pastaj 2, pastaj 1, do jetë: 0 - 3*5 = **-15**. Nqs fshijmë nyjen 2 pastaj nyjen 1, do jetë: 0 - 2*5* = ***-10***. *Nqs fshijmë vetëm nyjen 1, përfitimi do jetë: 0 - 1*5 = **-5**.

Nga të gjitha këto vlera të përfitimeve, vlera më e madhe është **-4**, e cila është edhe përgjigja.

Problemi 089

Kërkesa

Fituesit të një olimpiade programimi i takon të marrë si shpërblim një çek me vlerën **N**. Mirëpo të paktën një nga shifrat e numrit **N** është **4**, ndërkohë që tastiera e makinës që

shkruan çekun e ka të prishur tastin me numrin **4**. Megjithatë organizatorët e gjetën zgjidhjen duke shkruar dy çeqe, shuma e të cilëve është **N**, dhe asnjëri prej të cilëve nuk e përmban shifrën **4**. Gjeni dy numra të tillë.

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000051705/0000000000088231

Shembull

```
$ cat input.txt
3
4
940
4444

$ python3 prog.py < input.txt
Case #1: 2 2
Case #2: 852 88
Case #3: 667 3777
```

Zgjidhja

```python
for t in range(int(input())):
    n = input()
    n1 = []
    n2 = []
    for d in n:
        if d == '4':
            n1.append('3')
            n2.append('1')
        else:
            n1.append(d)
            n2.append('0')
    a = ''.join(n1)
    b = ''.join(n2)
    print('Case #{}: {} {}'.format(t+1, a, b))
```

Sqarime

Kur krijojmë dy numrat, për çdo shifër të ndryshme nga **4** e vendosim këtë shifër te numri i parë, ndërkohë që te i dyti vendosim **0**, kurse për çdo shifër të barabartë me **4**

vendosim shifrën **3** te numri i parë dhe shifrën **1** tek i dyti. Në këtë mënyrë garantojmë që shuma e tyre të jetë sa numri i dhënë, dhe asnjë prej tyre të mos përmbajë shifrën **4**.

Detyra

Sot keni një axhendë të ngjeshur, plot me gjëra të rëndësishme. Jeni lodhur goxha për tu pregatitur dhe për tu siguruar që veprimtaritë nuk kanë mbivendosje me njëra-tjetrën. Tani është mëngjes dhe po filloni të merakoseni se pavarësisht nga entuziazmi dhe dëshira e mirë, ndoshta nuk do të keni energji të mjaftueshme për ti bërë të gjitha me përkushtim të plotë.

Duhet ti përdorni energjitë tuaja me kujdes. Ditën e filloni plot me energji - **E** xhaul (joules) për të qënë më të saktë. Niveli juaj i energjisë asnjëherë nuk mund të bjerë më poshtë se 0 (përndryshe do të binit i pafuqishëm), dhe po ashtu nuk mund të rritet më shumë se **E**. Për secilën nga veprimtaritë mund të shpenzoni një sasi jo-negative energjie (ndoshta zero), dhe pas çdo veprimtarie do rifitoni **R** xhaul energji, por gjithmonë pa e kaluar nivelin maksimal të energjisë prej **E** xhaul (energjia e tepërt thjesht shkon dëm).

Disa prej veprimtarive janë më të rëndësishme se të tjerat. Për secilën veprimtari **i** keni një vlerë v_i që shpreh rëndësinë e kësaj veprimtarie për ju. Përfitimi që keni nga secila veprimtari është vlera e kësaj veprimtarie, shumëzuar me sasinë e energjisë që shpenzoni për të. Sa është përfitimi më i madh total që mund të keni?

Rradha e veprimtarive në kalendar nuk mund të ndryshohet, duhet ta përdorni energjinë sa më mirë të jetë e mundur me atë kalendar që keni.

Referenca: https://code.google.com/codejam/contest/2418487/dashboard#s=p1&a=1

Shembull

```
$ cat input.txt
3
5 2 2
2 1
5 2 2
1 2
3 3 4
4 1 3 5

$ python3 prog.py < input.txt
Case #1: 12
Case #2: 12
Case #3: 39
```

Në rreshtin e parë kemi numrat **E**, **R** dhe **N**. Në rreshtin pasardhës kemi N vlerat v_i për secilën prej N veprimtarive, sipas rradhës.

Në rastin e parë, mund ti shpenzojmë të 5-ta xhaulët për veprimtarinë e parë, pas asaj do rifitojmë 2 xhaul, të cilat mund ti shpenzojmë për veprintarinë e dytë. Kështu që përfitimi do jetë $5\mathit{2} + \mathit{2}1 = \mathbf{12}$.

Në rastin e dytë mund të shpenzojmë 2 xhaul për veprimtarinë e parë, i rifitojmë ato, dhe shpenzojmë 5 xhaul për veprimtarinë e dytë. Përfitimi: $2\mathit{1} + \mathit{5}2 = \mathbf{12}$.

Në rastin e tretë, rifitimi i energjisë është i barabartë me energjinë maksimale, që do të thotë se pas çdo veprimtarie i rifitojmë plotësisht të gjitha energjitë e harxhuara. Kështu që mund të shpenzojmë plot 3 xhaul për secilën veprimtari. Përfitimi total: $3\mathit{4} + \mathit{3}1 + 3\mathit{3} + \mathit{3}5 = \mathbf{39}$.

Problemi 090

Kërkesa

Kemi një pemë me **N** nyje (të shënuara me numrat **1** deri në **N**), ku rrënja është nyja **1**. Çdo nyje **i** ka një vlerë A_i (që mund të jetë edhe numër negativ).

Mund të zgjedhim një nyje çfarëdo të pemës dhe të fshijmë gjithë nënpemën (degën) poshtë saj, përfshirë edhe vetë nyjen. Këtë veprim mund ta përsërisim disa ose asnjë herë.

Le të përkufizojmë si **përfitim** shumën e të gjitha vlerave të nyjeve të mbetura në pemë, minus $X * k$, ku **X** është një numër i dhënë paraprakisht, dhe **k** është numri i herëve që është kryer veprimi i përshkruar më sipër. Gjeni vlerën më të madhe të mundshme të përfitimit.

Referenca: https://www.codechef.com/APRIL19B/problems/SUBREM

Shembull

```
$ cat input.txt
1
3 5
1 -5 -10
1 2
2 3

$ python3 prog.py < input.txt
-4
```

Kemi një rast testimi. Në rreshtin e parë jepen numrat **N** dhe **X**. Në rreshtin e dytë jepen **N** vlerat A_i. Në **N-1** rreshtat pasardhës jepen dy numra **u** dhe **v**, që tregojnë se ka një brinjë midis dy nyjeve të shënuara me **u** dhe **v**.

Në rastin e dhënë pema është e tillë: (1)–>(2)–>(3). Duke shënuar edhe vlerat përkatëse të çdo nyje mund ta paraqesim kështu: (1,1)–>(2,-5)–>(3,-10).

Nqs nuk fshijmë asnjë nyje/degë përfitimi do jetë: 1 + -5 + -10 = **-14**. Nqs fshijmë nyjen 3, do jetë: 1 + -5 - 1x5 = **-9**. Nqs fshijmë nyjen 3 dhe pastaj fshijmë nyjen 2, do jetë: 1 - 2x5 = **-9**. Nqs fshijmë vetëm nyjen 2 do jetë: 1 - 1x5 = **-4**. Nqs fshijmë nyjen 3, pastaj 2, pastaj 1, do jetë: 0 - 3x5 = **-15**. Nqs fshijmë nyjen 2 pastaj nyjen 1, do jetë: 0 - 2x5 = **-10**. Nqs fshijmë vetëm nyjen 1, përfitimi do jetë: 0 - 1x5 = **-5**.

Nga të gjitha këto vlera të përfitimeve, vlera më e madhe është **-4**, e cila është edhe përgjigja.

Zgjidhja

```
for _ in range(int(input())):
    # get the input
    n, x = map(int, input().split())
    values = [int(i) for i in input().split()]
    values = [0] + values
    edges = [[] for i in range(n+1)]
    for i in range(n-1):
        u, v = map(int, input().split())
        edges[v].append(u)
        edges[u].append(v)

    # make a pre-order tranversal of the tree and
    # find the parent and the children of each node
    parent = [0 for i in range(n+1)]
    children = [[] for i in range(n+1)]
    stack = [1]
    while stack:
        v = stack.pop()
        children[v] = [u for u in edges[v] if u != parent[v]]
        for u in children[v]:
            parent[u] = v
        stack.extend(children[v])

    # make a post-order tranversal of the tree and
    # find the maximum possible profit for each node
    visited = [False for i in range(n+1)]
```

```
    stack = [1]
    while stack:
        v = stack.pop()
        unvisited_children = [u for u in children[v] if not visited[u]]
        if unvisited_children:
            stack.append(v)
            stack.extend(unvisited_children)
        else:
            # process (visit) node v
            s = values[v] + sum([values[u] for u in children[v]])
            values[v] = max(s, -1*x)
            visited[v] = True

    # print the maximum possible profit for the root node
    print(values[1])
```

Sqarime

Pasi lexojmë vlerat N dhe X, te lista `values` ruajmë vlera A_i, dhe më pas te lista `edges` ndërtojmë listën e brinjëve për çdo nyje. Lista e brinjëve për një nyje të caktuar në fakt është lista e nyjeve ku mund të shkojmë me anë të një brinje, duke u nisur nga nyja e dhënë. Kjo listë përmban edhe prindin (paraardhësin në hierarkinë e pemës), edhe fëmijët (pasardhësit në hierarkinë e pemës) e nyjes së dhënë, sepse mënyra se si na jepen brinjët nuk e përcakton se cila prej nyjeve të brinjës është paraardhësi dhe cila është pasardhësi në hierarkinë e pemës.

Vini re që listat `values` dhe `edges` (dhe të gjitha listat e tjera që përdoren më poshtë) janë me gjatësi **N+1**, dhe kjo bëhet për shkak se nyjet e pemës në problemë janë të numëruara duke filluar nga 1-shi, dhe jo duke filluar nga 0-ja. Me këtë marifet shmangim nevojën për të konvertuar indekset.

Më poshtë, duke bërë një bredhje para-rendore në pemë, gjejmë se cili është prindi për çdo nyje, dhe cilët janë fëmijët. Bredhje para-rendore do të thotë që fillimish vizitohet prindi, dhe më pas fëmijët e tij. Këtë mund ta realizojmë duke përdorur një `stack` (stivë). Në një stivë, ajo që futet e fundit, del e para. Fillimisht fusim në stivë rrënjën e pemës (nyjen **1**). Pastaj, përderisa stiva nuk ka ngelur bosh, nxjerrim nyjen që është në krye të stivës, e përpunojmë (vizitojmë), pastaj fusim në stivë gjithë fëmijët e saj. Përpunimi që i bëhet nyjes në këtë rast është përcaktimi i fëmijëve të saj, si dhe caktimi si prind për të gjithë fëmijët e saj. Është e rëndësishme që bredhja të jetë para-rendore, sepse caktimi i prindit për fëmijët e një nyje bën të mundur gjetjen e fëmijëve të fëmijëve.

Tani jemi gati të gjemë përgjigjen e problemit.

Fillimisht vërejmë se nuk ka kuptim të presim një degë dhe pastaj të presim një prind të saj, sepse kjo vetëm rrit numrin e veprimeve të kryera dhe zvogëlon përfitimin.

Gjithashtu mund të vërejmë se gjatë llogaritjes së përfitimit, vlerën e një nyje mund ta zëvendësojmë me përfitimin më të madh të degës që varet nga ajo nyje, meqënëse kjo vetëm mund ta rrisë rezultatin përfundimtar (dhe ne atë duam, të gjejmë vlerën më të madhe të përfitimit). Një vëzhgim tjetër është që nqs përfitimi më i madh për një nyje faktikisht është më i vogël se vlera **(-1*X)**, atere na del më mirë që ta presim këtë degë (me një goditje/veprim), sepse në këtë rast vlera që do kontribonte kjo nyje në përfitimin total do ishte (-1*X), më e madhe se kontributi në rast se nuk do ta prisnim.

Përgjigja gjendet nga një bredhje pas-rendore e pemës, ku fillimisht vizitohen fëmijët e një nyje dhe pastaj vetë nyja. Për çdo nyje llogarisim vlerën më të madhe të përfitimit për atë nyje, duke ditur vlerat më të mëdha të përfitimit për secilin nga fëmijët (prandaj është e rëndësishme që bredhja të jetë pas-rendore). Në fund, përgjigja do jetë vlera më e madhe e përfitimit e llogaritur për nyjen rrënjë (nyjen 1).

Bredhjen pas-rendore mund ta realizojmë me ndihmën e një stive, si në rastin më sipër, vetlm se në këtë rast është e nevojshme të mbajmë shënim edhe nyjet që kemi vizituar tashmë. Kjo është e nevojshme sepse në stivë futet e para vetë nyja dhe pastaj fëmijët e saj, në mënyrë që nyja të përpunohet pasi të jenë përpunuar fëmijët e saj. Nëse nuk do të mbanim shënim nyjet që kemi përpunuar (vizituar) tashmë, nuk do ta dinim nëse fëmijët janë përpunuar ose jo, kur ta nxjerrim nga stiva nyjen për herë të dytë.

Detyra

Një nxënës ka studjuar me kujdes algoritmin e renditjes Bubble Sort dhe ka shpikur një variant të ri të tij.

Veprimi kryesor i algoritmit Bubble Sort është shqyrtimi i dy numrave të njëpasnjëshëm dhe ndërrimi i vendit të tyre, nëse i pari është më i madh se i dyti. Por algoritmi i shpikur shqyrton **3** numra të njëpasnjëshëm, dhe nëse i majti është më i madh se i djathti, ndryshon rradhën e tyre në të kundërt. Meqënëse ky algoritëm është si një "triplet bubble sort", shkurtimisht po e quajmë "Trouble Sort".

```
TroubleSort(L): // L is a 0-indexed list of integers
  let done := false
  while not done:
    done = true
    for i := 0; i < len(L)-2; i++:
      if L[i] > L[i+2]:
        done = false
        reverse the sublist from L[i] to L[i+2], inclusive
```

Për shembull, për listën L = 5 6 6 4 3, Trouble Sort do funksionojë si më poshtë:

- Hapi i parë:
- shqyrton 5 6 6 dhe nuk bën asgjë: 5 6 6 4 3

- shqyrton 6 6 4, shikon që 6 > 4, i ndryshon rradhën: 5 4 6 6 3
- shqyrton 6 6 3, shikon që 6 > 3, i ndryshon rradhën: 5 4 3 6 6
- Hapi i dytë:
- shqyrton 5 4 3, shikon që 5 > 3, i ndryshon rradhën: 3 4 5 6 6
- shqyrton 4 5 6, nuk bën gjë: 3 4 5 6 6
- shqyrton 5 6 6, nuk bën gjë: 3 4 5 6 6
- Hapi i tretë i shqyrton të treja treshet dhe nuk bën asgjë, kështu që algoritmi përfundon.

Megjithatë është e mundur që ky algoritëm mos ta bëjë gjithmonë sakt renditjen e listës. P.sh. shqyrtoni listën: 8 9 7.

Për një listë numrash të dhënë, gjeni nëse Trouble Sort e rendit saktë këtë listë ose jo. Nëse jo, gjeni indeksin e gabimit të parë në renditjen e bërë nga Trouble Sort, dmth gjeni pozicionin e numrit të parë që është më i madh se numri pasardhës.

Referenca: https://codejam.withgoogle.com/2018/challenges/00000000000000cb/dashboard/00000000000079cb

Shembull

```
$ cat input.txt
2
5
5 6 8 4 3
3
8 9 7

$ python3 prog.py < input.txt
Case #1: OK
Case #2: 1
```

Në rastin e parë Trouble Sort e rendit saktë listën e dhënë.

Në rastin e dytë, numri i dytë i listës (me treguesin 1) është më i madh se ai pasardhës, pas renditjes me Trouble Sort.

11 Problemat 091-100

Problemi 091

Kërkesa

Sot keni një axhendë të ngjeshur, plot me gjëra të rëndësishme. Jeni lodhur goxha për tu pregatitur dhe për tu siguruar që veprimtaritë nuk kanë mbivendosje me njëra-tjetrën. Tani është mëngjes dhe po filloni të merakoseni se pavarësisht nga entuziazmi dhe dëshira e mirë, ndoshta nuk do të keni energji të mjaftueshme për ti bërë të gjitha me përkushtim të plotë.

Duhet ti përdorni energjitë tuaja me kujdes. Ditën e filloni plot me energji - **E** xhaul (joules) për të qënë më të saktë. Niveli juaj i energjisë asnjëherë nuk mund të bjerë më poshtë se 0 (përndryshe do të binit i pafuqishëm), dhe po ashtu nuk mund të rritet më shumë se **E**. Për secilën nga veprimtaritë mund të shpenzoni një sasi jo-negative energjie (ndoshta zero), dhe pas çdo veprimtarie do rifitoni **R** xhaul energji, por gjithmonë pa e kaluar nivelin maksimal të energjisë prej **E** xhaul (energjia e tepërt thjesht shkon dëm).

Disa prej veprimtarive janë më të rëndësishme se të tjerat. Për secilën veprimtari **i** keni një vlerë v_i që shpreh rëndësinë e kësaj veprimtarie për ju. Përfitimi që keni nga secila veprimtari është vlera e kësaj veprimtarie, shumëzuar me sasinë e energjisë që shpenzoni për të. Sa është përfitimi më i madh total që mund të keni?

Rradha e veprimtarive në kalendar nuk mund të ndryshohet, duhet ta përdorni energjinë sa më mirë të jetë e mundur me atë kalendar që keni.

Referenca: https://code.google.com/codejam/contest/2418487/dashboard#s=p1&a=1

Shembull

```
$ cat input.txt
3
5 2 2
2 1
5 2 2
1 2
3 3 4
4 1 3 5
```

```
$ python3 prog.py < input.txt
Case #1: 12
Case #2: 12
Case #3: 39
```

Në rreshtin e parë kemi numrat **E**, **R** dhe **N**. Në rreshtin pasardhës kemi N vlerat v_i për secilën prej N veprimtarive, sipas rradhës.

Në rastin e parë, mund ti shpenzojmë të 5-ta xhaulët për veprimtarinë e parë, pas asaj do rifitojmë 2 xhaul, të cilat mund ti shpenzojmë për veprintarinë e dytë. Kështu që përfitimi do jetë $5\mathit{2} + \mathit{2}1 = \mathbf{12}$.

Në rastin e dytë mund të shpenzojmë 2 xhaul për veprimtarinë e parë, i rifitojmë ato, dhe shpenzojmë 5 xhaul për veprimtarinë e dytë. Përfitimi: $2\mathit{1} + \mathit{5}2 = \mathbf{12}$.

Në rastin e tretë, rifitimi i energjisë është i barabartë me energjinë maksimale, që do të thotë se pas çdo veprimtarie i rifitojmë plotësisht të gjitha energjitë e harxhuara. Kështu që mund të shpenzojmë plot 3 xhaul për secilën veprimtari. Përfitimi total: $3\mathit{4} + \mathit{3}1 + 3\mathit{3} + \mathit{3}5 = \mathbf{39}$.

Zgjidhja 1

```python
def max_gain(E, R, V, e_initial, e_remaining):
    """
    We start processing the array of values by having an initial energy
    of 'e_initial'. After processing the array of values we must still
    have 'e_remaining' energy left.

    """
    # if V is empty, gain is zero
    if not V: return 0

    # find max value and its index
    v_max = max(V)
    i = V.index(v_max)

    # split the array of values at the max value
    V_prev = V[:i]
    V_next = V[i+1:]

    # calculate the max energy that we can spend on v_max
    e1 = min(E, e_initial + R*len(V_prev))
```

```
    e2 = max(0, e_remaining - R*len(V_next))

    # calculate the max gain recursively
    gain = v_max * (e1 - e2)
    gain += max_gain(E, R, V_prev, e_initial, max(0, e1 - R))
    gain += max_gain(E, R, V_next, min(E, e2 + R), e_remaining)

    return gain

for t in range(int(input())):
    E, R, N = map(int, input().split())
    R = min(E, R)
    V = [int(v) for v in input().split()]
    print('Case #{}: {}'.format(t+1, max_gain(E, R, V, E, 0)))
```

Sqarime

Kjo zgjidhje përdor një algoritëm lakmitar (ose grykës, greedy). Fillimisht gjejmë veprimtarinë me vlerën më të madhe, dhe shpenzojmë për të sasinë më të madhe të mundshme të energjisë, pastaj vazhdojmë me vlerën tjetër më të madhe dhe shpenzojmë për të vlerën më të madhe të mundshme të energjisë, e kështu me rradhë. Për një diskutim se pse dhe si ky algoritëm lakmitar jep një zgjidhje optimale për këtë problem shikoni: https://code.google.com/codejam/contest/2418487/dashboard#s=a&a=1

Implementimi është bërë me një funksion rekursiv (që thërret vetveten me parametra më të vegjël). Ky funksion gjen përfitimin më të madh të mundshëm për një pjesë të veprimtarive (nënvarg i V), duke ditur energjinë që kemi para fillimit të tyre (`e_initial`), dhe duke u munduar që të na ketë mbetur akoma një sasi e caktuar energjie pas përfundimit të tyre (`e_remaining`).

Rasti bazë, që mbyll rekursivitetin, është kur lista e veprimtarive është bosh. Në këtë rast është e qartë që përfitimi është zero.

Kur kemi disa veprimtari, fillimisht gjejmë atë që ka vlerën më të madhe dhe mundohemi të harxhojmë sa më shumë energji që të jetë e mundur për këtë. Pastaj e thërrasim funksionin në mënyrë rekursive për listën e veprimtarive në të majtë dhe në të djathtë të tij.

Kjo zgjidhje e ka kohën të rendit $O(N^2)$, meqë gjetja e elementit më të madh të një liste normalisht është e rendit $O(N)$, dhe ky veprim duhet përsëritur pak a shumë për çdo element. Megjithatë, duke përdorur disa struktura të veçanta (si pirgjet/heaps), mund ta zvogëlojmë kohën e gjetjes së elementit më të madh, dhe mund ta ulim kohën totale të zgjidhjes në $O(N * ln(N))$.

Por ekziston edhe një zgjidhje e këtij problemi me kohë $O(N)$, të cilën do ta shikojmë

më poshtë. E bukura është se krijuesve të këtij problemi nuk u kishte vajtur në mëndje kjo zgjidhje, por e kishin gjetur disa prej konkuruesve.

Zgjidhja 2

```
for t in range(int(input())):
    E, R, N = map(int, input().split())
    R = min(E, R)
    V = [int(v) for v in input().split()]

    # for each V[i] find the index of the next value that is higher
    next_v = [None for i in range(N)]
    stack = [ {'v': max(V), 'i': None} ]
    for i in range(N):
        v = V[i]
        while v > stack[-1]['v']:
            item = stack.pop()
            next_v[item['i']] = i
        stack.append({'v': v, 'i':i})

    # find max_gain
    max_gain = 0
    e = E
    for i in range(N):
        if next_v[i] == None:
            max_gain += e*V[i]
            e = 0
        else:
            # find energy that needs to be left
            e1 = max(0, E - R*(next_v[i] - i))
            # find energy that can be spent for this activity
            spe = max(0, e - e1)
            max_gain += spe*V[i]
            e -= spe
        e = min(E, e + R)

    print('Case #{}: {}'.format(t+1, max_gain))
```

Sqarime

Kjo zgjidhje fillimisht gjen (në ndryshoren `next_v`) aktivitetin pasardhës që ka vlerë më të madhe se një aktivitet i caktuar. Me ndihmën e një stive (stack) kjo gjë mund të

bëhet në kohë lineare ($O(N)$), përndryshe do të duhej një kohë që në rastin më të keq është kuadratike.

Më pas i shqyrtojmë aktivitetet me rradhë (në kohë lineare). Nëse aktiviteti që po shqyrtohet nuk ka aktivitet pasardhës me vlerë më të madhe, atere të gjithë energjinë që kemi duam ta shpenzojmë për këtë aktivitet, sepse kështu do kemi përfitimin maksimal. Përndryshe, nëse ka një aktivitet pasardhës me vlerë më të madhe, atere llogarisim se sa energji mund të shpenzojmë tani, me qëllim që kur të vijë rradha e atij aktiviteti të kemi sa më shumë energji që të jetë e mundur (mundësisht **E**).

Detyra

Një robot alien po kërcënon gjithësinë, me një rreze që mund të shkatërrojë gjithë njohurinë mbi algoritmet. Duhet ta ndalojmë!

Për fat, e dimë se si punon roboti. Duke filluar me një rreze me fuqi **1**, ai zbaton njëri pas tjetrit udhëzimet në një program, nga e majta në të djathtë. Çdo udhëzim është një nga këto: - **C** (charge): Dyfisho fuqinë e rrezes. - **S** (shoot): Godit me rreze, duke shkaktuar një dëm të barabartë me fuqinë e rrezes.

P.sh. nëse programi është **SCCSSC**, roboti do bëjë këto: - Godit, duke shkaktuar 1 dëm. - Dyfisho, fuqia bëhet 2. - Dyfisho, fuqia bëhet 4. - Godit, dëmi 4. - Godit, dëmi 4. - Dyfisho, fuqia bëhet 8.

Në këtë rast dëmi total është: $1 + 4 + 4 = \mathbf{9}$.

Algoritmistat më të mirë të gjithësisë kanë krijuar një mbështjellë mbrojtëse e cila mund të përballojë një dëm total deri në **D**. Mirëpo programi i robotit mund të shkaktojë dëm edhe më shumë se kaq.

Presidenti i Gjithësisë ka dalë vullnetar që të shkojë në hapësirë dhe të hakojë programin e robotit para se ai ta zbatojë atë. Mënyra e vetme se si mund ta hakojë (pa e kuptuar roboti) është duke u ndërruar vendin dy udhëzimeve fqinje. P.sh. programin e mësipërm mund ta ndryshojë duke u shkëmbyer vendet udhëzimeve 3 dhe 4, duke e bërë programin **SCSCSC**. Kjo do ta pakësonte dëmin total në **7**. Pastaj mund ta ndryshonte prapë duke e bërë **SCSSCC** dhe duke e pakësuar dëmin total në **5**, e kështu me rradhë.

Që roboti mos të dyshojë, presidenti duhet të bëjë sa më pak hakime. Cili është numri më i vogël i hakimeve që janë të nevojshme për të siguruar që dëmi total që shkakton programi nuk e kalon vlerën **D**, nëse kjo është e mundur?

Referenca: https://codejam.withgoogle.com/2018/challenges/00000000000000cb/dashboard

Shembull

```
$ cat input.txt
```

```
6
1 CS
2 CS
1 SS
6 SCCSSC
2 CC
3 CSCSS

$ python3 prog.py < input.txt
Case #1: 1
Case #2: 0
Case #3: IMPOSSIBLE
Case #4: 2
Case #5: 0
Case #6: 5
```

Jepet numri **D** dhe programi **P**.

Problemi 092

Kërkesa

Një nxënës ka studjuar me kujdes algoritmin e renditjes Bubble Sort dhe ka shpikur një variant të ri të tij.

Veprimi kryesor i algoritmit Bubble Sort është shqyrtimi i dy numrave të njëpasnjëshëm dhe ndërrimi i vendit të tyre, nëse i pari është më i madh se i dyti. Por algoritmi i shpikur shqyrton **3** numra të njëpasnjëshëm, dhe nëse i majti është më i madh se i djathti, ndryshon rradhën e tyre në të kundërt. Meqënëse ky algoritëm është si një "triplet bubble sort", shkurtimisht po e quajmë "Trouble Sort".

```
TroubleSort(L): // L is a 0-indexed list of integers
  let done := false
  while not done:
    done = true
    for i := 0; i < len(L)-2; i++:
      if L[i] > L[i+2]:
        done = false
        reverse the sublist from L[i] to L[i+2], inclusive
```

Për shembull, për listën L = 5 6 6 4 3, Trouble Sort do funksionojë si më poshtë:

- Hapi i parë:

- shqyrton 5 6 6 dhe nuk bën asgjë: 5 6 6 4 3
- shqyrton 6 6 4, shikon që $6 > 4$, i ndryshon rradhën: 5 4 6 6 3
- shqyrton 6 6 3, shikon që $6 > 3$, i ndryshon rradhën: 5 4 3 6 6
- Hapi i dytë:
- shqyrton 5 4 3, shikon që $5 > 3$, i ndryshon rradhën: 3 4 5 6 6
- shqyrton 4 5 6, nuk bën gjë: 3 4 5 6 6
- shqyrton 5 6 6, nuk bën gjë: 3 4 5 6 6
- Hapi i tretë i shqyrton të treja treshet dhe nuk bën asgjë, kështu që algoritmi përfundon.

Megjithatë është e mundur që ky algoritëm mos ta bëjë gjithmonë sakt renditjen e listës. P.sh. shqyrtoni listën: 8 9 7.

Për një listë numrash të dhënë, gjeni nëse Trouble Sort e rendit saktë këtë listë ose jo. Nëse jo, gjeni indeksin e gabimit të parë në renditjen e bërë nga Trouble Sort, dmth gjeni pozicionin e numrit të parë që është më i madh se numri pasardhës.

Referenca: https://codejam.withgoogle.com/2018/challenges/00000000000000cb/dashboard/00000000000079cb

Shembull

```
$ cat input.txt
2
5
5 6 8 4 3
3
8 9 7

$ python3 prog.py < input.txt
Case #1: OK
Case #2: 1
```

Në rastin e parë Trouble Sort e rendit saktë listën e dhënë.

Në rastin e dytë, numri i dytë i listës (me treguesin 1) është më i madh se ai pasardhës, pas renditjes me Trouble Sort.

Zgjidhja 1

```
def troublesort(L):
    done = False
    while not done:
```

```
        done = True
        for i in range(len(L) - 2):
            if L[i] > L[i+2]:
                done = False
                L[i], L[i+2] = L[i+2], L[i]

def check(L):
    for i in range(len(L)-1):
        if L[i] > L[i+1]:
            return i
    return 'OK'

T = int(input())
for t in range(T):
    N = int(input())
    L = list(map(int, input().split()))
    troublesort(L)
    print("Case #{}: {}".format(t+1, check(L)))
```

Sqarime

Trouble Sort ka të njëjtën kohë renditje si Bubble Sort, që është $O(N^2)$. Kështu që edhe kjo zgjidhje ka po këtë kohë. Zgjidhja e dytë është më e shpejtë.

Zgjidhja 2

```
def troublesort(L):
    n = len(L)
    L1 = [L[i] for i in range(n) if i % 2 == 0]
    L2 = [L[i] for i in range(n) if i % 2 == 1]
    L1.sort()
    L2.sort()
    for i in range(n):
        L[i] = L1[i//2] if i % 2 == 0 else L2[i//2]

def check(L):
    for i in range(len(L)-1):
        if L[i] > L[i+1]:
            return i
    return 'OK'
```

```
T = int(input())
for t in range(T):
    N = int(input())
    L = list(map(int, input().split()))
    troublesort(L)
    print("Case #{}: {}".format(t+1, check(L)))
```

Sqarime

Mund të vihet re se Trouble Sort krahason vetëm numrat që janë në vënde tek me njëri-tjetrin, dhe numrat që janë në vënde çift me njëri-tjetrin, por asnjëherë një numër që ndodhet në një vend çift me një numër që ndodhet në një vend tek. Kështu që kryen renditje të pavarur të dy nënlistave. (Në fakt ky është edhe problemi se pse rezultati i algoritmit Trouble Sort nuk është gjithmonë i saktë.)

Të njëjtin rezultat që jep Trouble Sort ne mund ta marrim edhe nëse i veçojmë dy nënlistat, i rendisim në mënyrë të pavarur nga njëra-tjetra, por me një algoritëm me kohë më të shpejtë se $O(N^2)$, dhe i bashkojmë sërish. Në këtë mënyrë kjo zgjidhje do ketë një kohë më të shpejtë se $O(N^2)$ (zakonisht $O(N * log(N))$.

Detyra

Në grupin e programimit ne kemi qejf ti dërgojmë njëri tjetrit **panagrama**. Një fjali ose shprehje quhet **pangram** kur i përmban të gjitha shkronjat e alfabetit anglisht të paktën një herë. P.sh. shprehja "the quick brown fox jumps over the lazy dog" është një panagram. Ndonjëherë panagramat tona përmbajnë informata sekrete, si p.sh. "CJ QUIZ: KNOW BEVY OF DP FLUX ALGORITHMS", dhe duhen mbajtur të fshehta.

Pasi lexuam për disa minuta një libër rreth shifrimit(enkriptimit), dhe mësuam se është shumë e vështirë të gjenden përbërësit e prodhimit të dy numrave të thjeshtë shumë të mëdhenj, hartuam një skemë shifrimi të bazuar në këtë fakt. Fillimisht bëmë këto përgatitje:

- Zgjodhëm 26 numra të thjeshtë të ndryshëm nga njëri tjetri dhe më të vegjël se **N**.
- I renditëm nga më i vogli te më i madhi. Pastaj më të voglit i caktuam shkronjën A, të dytit shkronjën B, e kështu me rradhë.
- Të gjithë pjesëtarët e grupit e mbajtën shënim këtë listë.

Tani, kur duam të dërgojmë ndonjë panagram si mesazh, fillimisht heqim të gjitha vëndet bosh midis fjalëve, pastaj shkruajmë prodhimin e numrave të thjeshtë që i korrespondojnë shkronjës së parë dhe të dytë të mesazhit, pastaj shkruajmë prodhimin e numrave të thjeshtë të shkronjës së dytë dhe të tretë, e kështu me rradhë, duke përfunduar me

prodhimin e numrave të shkronjës së fundit dhe të parafundit. Kjo listë me numra që formohet është versioni i shifruar i mesazhit.

P.sh. nqs **N**=103 dhe kemi zgjedhur të përdorim 26 numrat e parë të thjeshtë pas numrit 2 (sepse kemi frikë se faktorizimi i numrave çift është shumë i thjeshtë), do kemi A=3, B=5, C=7, D=11, e kështu me rradhë, deri te Z=103. Nqs duam të kriptojmë panagramën "CJ QUIZ..." më sipër, teksti do jetë "CJQUIZKNOWBEVYOFDPFLUXALGORITHMS". Atere numri i parë i mesazhit të shifruar do jetë 7*31=217 (sepse C=7 dhe J=31), numri i dytë do jetë 1891, e kështu me rradhë, numri i fundit do jetë 3053.

Nëse ju jepet një mesazh i shifruar në këtë mënyrë dhe numri **N** i përdorur, por pa ju thënë se cilët numra të thjeshtë janë përdorur, a mund ta deshifroni atë dhe të gjeni mesazhin origjinal.

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000051705/000000000008830b

Shembull

```
$ cat input.txt
2
103 31
217 1891 4819 2291 2987 3811 1739 2491 4717 445 65 1079 8383 5353 901 187 649 1003
10000 25
3292937 175597 18779 50429 375469 1651121 2102 3722 2376497 611683 489059 2328901 3

$ python3 prog.py < input.txt
Case #1: CJQUIZKNOWBEVYOFDPFLUXALGORITHMS
Case #2: SUBDERMATOGLYPHICFJKNQVWXZ
```

Për çdo rast testimi jepen numrat **N** dhe **L**, ku **N** është kufiri i numrave të thjeshtë të përdorur, dhe **L** është gjatësia e listës me prodhimet e numrave të thjeshtë, e cila jepet në rreshtin pasardhës.

Problemi 093

Kërkesa

Një robot alien po kërcënon gjithësinë, me një rreze që mund të shkatërrojë gjithë njohurinë mbi algoritmet. Duhet ta ndalojmë!

Për fat, e dimë se si punon roboti. Duke filluar me një rreze me fuqi **1**, ai zbaton njëri pas tjetrit udhëzimet në një program, nga e majta në të djathtë. Çdo udhëzim është

një nga këto: - **C** (charge): Dyfisho fuqinë e rrezes. - **S** (shoot): Godit me rreze, duke shkaktuar një dëm të barabartë me fuqinë e rrezes.

P.sh. nëse programi është **SCCSSC**, roboti do bëjë këto: - Godit, duke shkaktuar 1 dëm. - Dyfisho, fuqia bëhet 2. - Dyfisho, fuqia bëhet 4. - Godit, dëmi 4. - Godit, dëmi 4. - Dyfisho, fuqia bëhet 8.

Në këtë rast dëmi total është: $1 + 4 + 4 = \mathbf{9}$.

Algoritmistat më të mirë të gjithësisë kanë krijuar një mbështjellë mbrojtëse e cila mund të përballojë një dëm total deri në **D**. Mirëpo programi i robotit mund të shkaktojë dëm edhe më shumë se kaq.

Presidenti i Gjithësisë ka dalë vullnetar që të shkojë në hapësirë dhe të hakojë programin e robotit para se ai ta zbatojë atë. Mënyra e vetme se si mund ta hakojë (pa e kuptuar roboti) është duke u ndërruar vendin dy udhëzimeve fqinje. P.sh. programin e mësipërm mund ta ndryshojë duke u shkëmbyer vendet udhëzimeve 3 dhe 4, duke e bërë programin **SCSCSC**. Kjo do ta pakësonte dëmin total në **7**. Pastaj mund ta ndryshonte prapë duke e bërë **SCSSCC** dhe duke e pakësuar dëmin total në **5**, e kështu me rradhë.

Që roboti mos të dyshojë, presidenti duhet të bëjë sa më pak hakime. Cili është numri më i vogël i hakimeve që janë të nevojshme për të siguruar që dëmi total që shkakton programi nuk e kalon vlerën **D**, nëse kjo është e mundur?

Referenca: https://codejam.withgoogle.com/2018/challenges/00000000000000cb/dashboard

Shembull

```
$ cat input.txt
6
1 CS
2 CS
1 SS
6 SCCSSC
2 CC
3 CSCSS

$ python3 prog.py < input.txt
Case #1: 1
Case #2: 0
Case #3: IMPOSSIBLE
Case #4: 2
Case #5: 0
Case #6: 5
```

Jepet numri **D** dhe programi **P**.

Zgjidhja

```python
def total_damage(P):
    damage = 0
    strength = 1
    for c in P:
        if c == 'S': damage += strength
        if c == 'C': strength += strength
    return damage

for t in range(int(input())):
    D, P = input().split()
    D = int(D)
    P = list(P)

    hacks = 0
    damage = total_damage(P)
    while True:
        if damage <= D:
            ans = hacks
            break

        # make a hack
        for i in range(len(P)-1, 0, -1):
            if P[i]=='S' and P[i-1]=='C':
                P[i] = 'C'
                P[i-1] = 'S'
                hacks += 1
                break

        new_damage = total_damage(P)
        if new_damage == damage:
            ans = 'IMPOSSIBLE'
            break
        else:  # new_damage < damage
            damage = new_damage

    print('Case #{}: {}'.format(t+1, ans))
```

Sqarime

Meqënëse duam të bëjmë sa më pak hakime, atere duhet të fillojmë ti bëjmë këto nga fundi i programit, sepse aty kanë efektin më të madh. Sa herë që bëjmë një hakim llogarisim përsëri dëmin total. Nëse ky është bërë `<= D`, atere kemi mbaruar punë, nuk është nevoja të vazhdojmë me hakime. Nëse është i barabartë me dëmin e mëparshëm, kjo do të thotë që nuk kemi arritur të bëjmë asnjë hakim dhe nuk është e mundur ta ulim akoma më tepër dëmin, kështu që në këtë rast e ndërpresim ciklin dhe japim përgjigjen 'IMPOSSIBLE'.

Detyra

Osmos është një lojë me disa shirita me gjatësi të ndryshme që lëvizin në një fushë si gjarpërinj. Një shirit mund të "gëlltisë" një shirit tjetër nqs gjatësia e tij është më e madhe, dhe në këtë rast gjatësia e shiritit shtohet me gjatësinë e shiritit të gëlltitur.

P.sh. nqs shiriti që po komandojmë ne e ka gjatësinë 10, dhe në fushë janë edhe tre shirita të tjerë me gjatësi 9, 13 dhe 19, fillimisht mund të gëlltisim vetëm shiritin me gjatësi 9. Pas kësaj gjatësia e shiritit tonë bëhet 19, kështu që mund të gëlltisim edhe shiritin me gjatësi 13 (por jo atë me gjatësi 19). Gjatësia e shiritit tonë bëhet 31, kështu që mund të gëlltisim edhe shiritin e fundit me gjatësi 19 dhe loja mbaron.

Programi i lojës në fillim gjeneron shiritin e lojtarit dhe disa shirita të tjerë me gjatësi të rastësishme. Mirëpo është e mundur që me ato gjatësi të gjeneruara rastësisht të mos ketë një mënyrë për ti gëlltitur të gjithë shiritat dhe për të fituar lojën. Kjo situatë mund të rregullohet me këto dy veprime: duke shtuar shtuar një shirit tjetër me gjatësi të caktuar, ose duke hequr ndonjë nga shiritat ekzistues. Cili është numri më i vogël i këtyre veprimeve për ta bërë lojën të zgjidhshme?

P.sh. le ta zëmë se në fillim gjatësia e shiritit të lojtarit është 10 dhe shiritat e tjerë kanë gjatësi [9, 20, 25, 100]. Kjo lojë nuk është e zgjidhshme. Por po të shtojmë një shirit me gjatësi 3 dhe të heqim shiritin me gjatësi 100, mund ta bëjmë të zgjidhshme me vetëm 2 veprime. Përgjigja në këtë rast është 2.

Referenca: https://code.google.com/codejam/contest/dashboard?c=2434486

Shembull

```
$ cat input.txt
4
2 2
2 1
2 4
2 1 1 6
10 4
```

```
25 20 9 100
1 4
1 1 1 1

$ python3 prog.py < input.txt
Case #1: 0
Case #2: 1
Case #3: 2
Case #4: 4
```

Rreshti i parë i çdo rasti testimi jep gjatësinë e shiritit të lojtarit dhe numrin N të shiritave të tjerë. Rreshti i dytë jep gjatësitë e N shiritave të tjerë.

Problemi 094

Kërkesa

Në grupin e programimit ne kemi qejf ti dërgojmë njëri tjetrit **panagrama**. Një fjali ose shprehje quhet **pangram** kur i përmban të gjitha shkronjat e alfabetit anglisht të paktën një herë. P.sh. shprehja "the quick brown fox jumps over the lazy dog" është një panagram. Ndonjëherë panagramat tona përmbajnë informata sekrete, si p.sh. "CJ QUIZ: KNOW BEVY OF DP FLUX ALGORITHMS", dhe duhen mbajtur të fshehta.

Pasi lexuam për disa minuta një libër rreth shifrimit(enkriptimit), dhe mësuam se është shumë e vështirë të gjenden përbërësit e prodhimit të dy numrave të thjeshtë shumë të mëdhenj, hartuam një skemë shifrimi të bazuar në këtë fakt. Fillimisht bëmë këto përgatitje:

- Zgjodhëm 26 numra të thjeshtë të ndryshëm nga njëri tjetri dhe më të vegjël se **N**.
- I renditëm nga më i vogli te më i madhi. Pastaj më të voglit i caktuam shkronjën A, të dytit shkronjën B, e kështu me rradhë.
- Të gjithë pjesëtarët e grupit e mbajtën shënim këtë listë.

Tani, kur duam të dërgojmë ndonjë panagram si mesazh, fillimisht heqim të gjitha vëndet bosh midis fjalëve, pastaj shkruajmë prodhimin e numrave të thjeshtë që i korrespondojnë shkronjës së parë dhe të dytë të mesazhit, pastaj shkruajmë prodhimin e numrave të thjeshtë të shkronjës së dytë dhe të tretë, e kështu me rradhë, duke përfunduar me prodhimin e numrave të shkronjës së fundit dhe të parafundit. Kjo listë me numra që formohet është versioni i shifruar i mesazhit.

P.sh. nqs **N**=103 dhe kemi zgjedhur të përdorim 26 numrat e parë të thjeshtë pas numrit 2 (sepse kemi frikë se faktorizimi i numrave çift është shumë i thjeshtë), do kemi A=3, B=5, C=7, D=11, e kështu me rradhë, deri te Z=103. Nqs duam të kriptojmë

panagramën “CJ QUIZ...” më sipër, teksti do jetë “CJQUIZKNOWBEVYOFDPFLUX-ALGORITHMS”. Atere numri i parë i mesazhit të shifruar do jetë 7*31=217 (sepse C=7 dhe J=31), numri i dytë do jetë 1891, e kështu me rradhë, numri i fundit do jetë 3053.

Nëse ju jepet një mesazh i shifruar në këtë mënyrë dhe numri **N** i përdorur, por pa ju thënë se cilët numra të thjeshtë janë përdorur, a mund ta deshifroni atë dhe të gjeni mesazhin origjinal.

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000051705/000000000008830b

Shembull

```
$ cat input.txt
2
103 31
217 1891 4819 2291 2987 3811 1739 2491 4717 445 65 1079 8383 5353 901 187 649 1003 697 3239 76
10000 25
3292937 175597 18779 50429 375469 1651121 2102 3722 2376497 611683 489059 2328901 3150061 8299

$ python3 prog.py < input.txt
Case #1: CJQUIZKNOWBEVYOFDPFLUXALGORITHMS
Case #2: SUBDERMATOGLYPHICFJKNQVWXZ
```

Për çdo rast testimi jepen numrat **N** dhe **L**, ku **N** është kufiri i numrave të thjeshtë të përdorur, dhe **L** është gjatësia e listës me prodhimet e numrave të thjeshtë, e cila jepet në rreshtin pasardhës.

Zgjidhja

```
from math import gcd
for t in range(int(input())):
    N, L = map(int, input().split())
    C = [int(i) for i in input().split()]    # ciphertext

    # find prime numbers comprising the original message
    S = [0]*(L + 1)
    i = 0
    while C[i] == C[i+1]:
        i += 1
    S[i+1] = gcd(C[i], C[i+1])
    j = i
```

```
    while j >= 0:
        S[j] = C[j] // S[j+1]
        j -= 1
    i += 1
    while i < L:
        S[i+1] = C[i] // S[i]
        i += 1

    # find the mapping between the prime numbers and the letters of the aplhabet
    A = list(set(S))
    A.sort()
    M = {}    # mapping
    for i in range(len(A)):
        M[A[i]] = chr(ord('A') + i)

    # find the original deciphered message
    msg = ''.join([M[i] for i in S])

    print('Case #{}: {}'.format(t+1, msg))
```

Sqarime

Le të shënojmë me Prime(X) numrin e thjeshtë që i korrespondon shkronjës X, dhe le të jenë XYZ tre shkronja të njëpasnjëshme në mesazhin e pa shifruar. Atere në listën e numrave të dhënë do kemi dy numra të njëpasnjëshëm L1=Prime(X)*Prime(Y) dhe L2=Prime(Y)*Prime(Z). Mund të vëmë re se Prime(Y) është pjesëtuesi më i madh i përbashkët i numrave të njëpasnjëshëm L1 dhe L2, dhe PMP i dy numrave mund të gjendet shumë lehtë (me algoritmin e Euklidit). Pasi kemi gjetur Prime(Y)=PMP(L1, L2), mund të gjejmë shumë kollaj edhe Prime(X)=L1/Prime(Y) dhe Prime(Z)=L2/Prime(Y).

Duke përdorur këtë metodë mund të gjejmë të gjithë numrat e thjeshtë që janë përdorur në mesazh, dhe meqënëse mesazhi i dhënë është panagram dhe aty ndodhen të 26 shkronjat e alfabetit (anglisht), do na dalin 26 numra të thjeshtë. Mund ti rendisim këta numra dhe të parin ta vëmë në korrespondencë me shkronjën A, të dytin me shkronjën B, e kështu me rradhë. Duke ditur këtë korrespondence dhe numrat e thjeshtë që përbëjnë mesazhin, mund të gjejmë shkronjat e mesazhit fillestar.

Shënim: Nqs mesazhi fillesar fillon me shkronja që përsëriten, si p.sh. ABABAB... atere numrat e parë të mesazhit të shifruar do jenë të barabartë: Prime(A)*Prime(B) dhe Prime(B)*Prime(A), dhe kështu që PMP e tyre do jetë 0. Kështu që zbërthimin me PMP duhet ta fillojmë aty ku kemi dy numra të njëpasnjëshëm që janë të ndryshëm.

Detyra

Jepet një numër **n**. Gjeni mbetjen e pjesëtimit të shumës $1! + 2! + ... + n!$ me $10^9 + 7$ (i cili është numër i thjeshtë).

Referenca: https://www.codechef.com/DARG2019/problems/SUMMOD

Shembull

```
$ cat input.txt
3
1000
10000
100000

$ python3 prog.py < input.txt
980630010
396626132
789934569
```

Problemi 095

Kërkesa

Osmos është një lojë me disa shirita me gjatësi të ndryshme që lëvizin në një fushë si gjarpërinj. Një shirit mund të "gëlltisë" një shirit tjetër nqs gjatësia e tij është më e madhe, dhe në këtë rast gjatësia e shiritit shtohet me gjatësinë e shiritit të gëlltitur.

P.sh. nqs shiriti që po komandojmë ne e ka gjatësinë 10, dhe në fushë janë edhe tre shirita të tjerë me gjatësi 9, 13 dhe 19, fillimisht mund të gëlltisim vetëm shiritin me gjatësi 9. Pas kësaj gjatësia e shiritit tonë bëhet 19, kështu që mund të gëlltisim edhe shiritin me gjatësi 13 (por jo atë me gjatësi 19). Gjatësia e shiritit tonë bëhet 31, kështu që mund të gëlltisim edhe shiritin e fundit me gjatësi 19 dhe loja mbaron.

Programi i lojës në fillim gjeneron shiritin e lojtarit dhe disa shirita të tjerë me gjatësi të rastësishme. Mirëpo është e mundur që me ato gjatësi të gjeneruara rastësisht të mos ketë një mënyrë për ti gëlltitur të gjithë shiritat dhe për të fituar lojën. Kjo situatë mund të rregullohet me këto dy veprime: duke shtuar shtuar një shirit tjetër me gjatësi të caktuar, ose duke hequr ndonjë nga shiritat ekzistues. Cili është numri më i vogël i këtyre veprimeve për ta bërë lojën të zgjidhshme?

P.sh. le ta zëmë se në fillim gjatësia e shiritit të lojtarit është 10 dhe shiritat e tjerë kanë gjatësi [9, 20, 25, 100]. Kjo lojë nuk është e zgjidhshme. Por po të shtojmë një shirit me gjatësi 3 dhe të heqim shiritin me gjatësi 100, mund ta bëjmë të zgjidhshme me vetëm 2 veprime. Përgjigja në këtë rast është 2.

Referenca: https://code.google.com/codejam/contest/dashboard?c=2434486

Shembull

```
$ cat input.txt
4
2 2
2 1
2 4
2 1 1 6
10 4
25 20 9 100
1 4
1 1 1 1

$ python3 prog.py < input.txt
Case #1: 0
Case #2: 1
Case #3: 2
Case #4: 4
```

Rreshti i parë i çdo rasti testimi jep gjatësinë e shiritit të lojtarit dhe numrin N të shiritave të tjerë. Rreshti i dytë jep gjatësitë e N shiritave të tjerë.

Zgjidhja

```
def make_solvable(A, L):
    """A is the size of the mot of the player, L is a sorted list of the
    mots according to their size. Find and return the minimal number
    of moves that can make this case solvable.

    """
    # If the list of mots is empty, we are done.
    if not L: return 0

    # If A is 1, the only way to make it solvable is to delete all the
    # mots, because it cannot absorb any mot and we cannot inrease its
    # length.
    if A == 1: return len(L)

    # If A > L[0], we can absorb it without any additions or deletions.
```

```
    if A > L[0]:
        return make_solvable(A + L[0], L[1:])

    # At this point we have A <= L[0]. We can either add mots (in
    # front of L[0]) until we can absorb it, or delete all the
    # remaining mots. Deleting just the first mot will not improve
    # the situation, since the next one is not smaller than it. The
    # same goes for deletting any other mots.

    # Let's find the number of mots we have to insert in order to
    # absorb L[0].
    nr = 0   # nr of mots we have to insert
    while A <= L[0]:
        nr += 1
        A += A - 1

    # If number of motes we have to insert is greater than the
    # remaining mots, we better just delete the remaining mots, and
    # then we are done.
    if nr >= len(L):
        return len(L)

    # Find the number of changes if we absorb the first mot and continue.
    nr1 = make_solvable(A + L[0], L[1:])

    # Return the smallest number of deleting all mots
    # or absorbing the first one and continuing.
    return min(nr + nr1, len(L))

for t in range(int(input())):
    A, N = [int(i) for i in input().split()]
    L = [int(i) for i in input().split()]
    L.sort()
    print('Case #{}: {}'.format(t+1, make_solvable(A, L)))
```

Detyra

Në një rrugë janë $\mathbf{N} + 2$ vënde parkimi në një rresht të vetëm. Vëndi i parë dhe i fundit janë gjithmonë të zënë nga policia bashkiake, kurse $\mathbf{N}$ të tjerat janë për përdoruesit.

Kur vjen një person për të parkuar ai përpiqet të zërë një vend që të jetë sa më larg nga makinat e tjera. Për të shmangur ngatërresat ata ndjekin disa rregulla të paracaktuara. Për çdo vënd parkimi bosh V llogarisin vlerat M_V dhe D_V, që janë numrat e vendeve bosh

midis këtij vendi dhe vendeve të zëna në të majtë dhe në të djathtë. Pastaj shqyrtojnë ato vënde që kanë fqinjin më të largët, dmth ato vënde V për të cilat $min(M_V, D_V)$ është maksimale. Nëse është vetëm një vend i tillë, atere atë zgjedhin. Përndryshe zgjedhin prej tyre atë vend që e ka vlerën $max(M_V, D_V)$ maksimale. Nëse prapë ka disa vënde të tilla, atere zgjedhin atë prej tyre që është sa më majtas.

Në parkim vijnë **K** persona. Secili prej tyre parkon para se të vijë personi pasardhës (duke ndjekur rregullat më sipër), dhe asnjë prej tyre nuk largohet.

Kur parkon personi i fundit, sa do jenë vlerat $max(M_V, D_V)$ dhe $min(M_V, D_V)$?

Referenca: https://code.google.com/codejam/contest/3264486/dashboard#s=p2&a=2

Shembull

```
$ cat input.txt
7
4 2
5 2
6 2
1000 1000
1000 1
1000000 500000
1000000000000000000 500000000000000000

$ python3 prog.py < input.txt
Case #1: 1 0
Case #2: 1 0
Case #3: 1 1
Case #4: 0 0
Case #5: 500 499
Case #6: 1 0
Case #7: 1 0
```

Për çdo rast testimi jepen numrat **N** dhe **K**.

Në rastin e parë situata është e tillë: **0.12.0** (me **0** shënojmë makinat e policisë, me pikë (**.**) shënojmë vëndet bosh, dhe me numrat 1 dhe 2 shënojmë vëndet që kanë zënë i pari dhe i dyti). Kështu që përgjigja është "**1 0**".

Në rastin e dytë situata është e tillë: **02.1..0**, kështu që përgjigja është "**1 0**".

Në rastin e tretë situata është e tillë: **0..1.2.0**, kështu që përgjigja është "**1 1**".

Në rastin e katërt çdo vënd do jetë i zënë në fund, pavarësisht nga rradha se si zihen, kështu që përgjigja është "**0 0**".

Problemi 096

Kërkesa

Jepet një numër **n**. Gjeni mbetjen e pjesëtimit të shumës $1! + 2! + ... + n!$ me $10^9 + 7$ (i cili është numër i thjeshtë).

Referenca: https://www.codechef.com/DARG2019/problems/SUMMOD

Shembull

```
$ cat input.txt
3
1000
10000
100000

$ python3 prog.py < input.txt
980630010
396626132
789934569
```

Zgjidhja 1

```
import sys
sys.setrecursionlimit(100000)

def factorial(n):
    return 1 if n==1 else n*factorial(n-1)

def factorial_sum(n):
    return 1 if n==1 else factorial(n) + factorial_sum(n - 1)

for _ in range(int(input())):
    n = int(input())
    print(factorial_sum(n) % 1000000007)
```

Sqarime

Kjo zgjidhje është e thjeshtë po tepër jo-efiçente. Ajo kalon vetëm testin e parë (n=1000).

Zgjidhja 2

```
import sys
sys.setrecursionlimit(100000)

hash_factorials = {1:1, 2:2, 3:6}
hash_factorial_sums = {1:1}

def factorial(n):
    f = hash_factorials.get(n, False)
    if f: return f

    f = n * factorial(n - 1)
    f %= 1000000007
    hash_factorials[n] = f
    return f

def factorial_sum(n):
    s = hash_factorial_sums.get(n, False)
    if s: return s

    s = factorial(n) + factorial_sum(n - 1)
    s %= 1000000007
    hash_factorial_sums[n] = s
    return s

for _ in range(int(input())):
    n = int(input())
    print(factorial_sum(n))
```

Sqarime

Kjo zgjidhje është më e shpejtë se e para sepse nqs është gjetur njëherë faktoriali i një numri, ai mbahet shënim dhe nuk rillogaritet.

Gjithashtu mbahen shënim vetëm mbetjet përkatëse, meqënëse `(a * b) % n == ((a % n) * (b % n)) % n` dhe `(a + b) % n == ((a % n) + (b % n)) % n`.

Edhe pse është më e shpejtë se e para, kjo zgjidhje nuk e kalon testin e tretë.

Zgjidhja 3

```
n = 10**5
m = 10**9 + 7
sums = [0 for i in range(n+1)]

f = 1
s = 0
for i in range(1, n+1):
    f *= i
    f %= m
    s += f
    s %= m
    sums[i] = s

for _ in range(int(input())):
    n = int(input())
    print(sums[n])
```

Sqarime

Mund ti llogarisim paraprakisht të gjitha shumat, në mënyrë iterative dhe rezultatet ti mbajmë shënim në një listë. Pastaj për çdo **n** që të na jepet ne e kemi përgjigjen gati (te lista `sums`).

Kjo zgjidhje është mjaft e shpejtë dhe e kalon edhe testin e tretë.

Detyra

Një klasë ka **N** nxënës dhe pikët e secilit prej tyre na jepen në një listë **L**. Gjeni të gjitha prerjet e mundshme $P = (x, y)$ që mund ti bëhen kësaj liste, të tilla që shuma e pikëve brënda prerjes të jetë e barabartë me shumën e pikëve jashtë prerjes. Dmth $L_x + L_{x+1} + ... + L_{y-1} + L_y = (L_0 + L_1 + ... + L_{x-1}) + (L_{y+1} + L_{y+2} + ... + L_{N-1})$.

Nëse nuk ka asnjë prerje të tillë printoni **-1**, përndryshe printoni numrin e prerjeve dhe shumën e indekseve **x** dhe **y** për të gjitha prerjet. Këto indekse janë me bazë 0, dhe $0 < x <= y < N - 1$. Gjithashtu $3 <= N <= 10^5$ dhe $1 <= L_i <= 10^5$.

Referenca: https://www.codechef.com/problems/EQSBAR

Shembull

```
$ cat input.txt
```

```
2
10
7 8 12 8 13 9 5 7 12 3
5
32 36 34 20 28

$ python3 prog.py < input.txt
2 17
-1
```

Problemi 097

Kërkesa

Ana ka një rresht me **N** kuba, ku në secilin kub ndodhet një nga shkronjat e alfabetit (anglisht) nga A deri në Z. Kubat mund ti shënojmë me numrat nga 1 deri në N.

Boni i jep Anës **Q** pyetje, ku çdo pyetje **i** është e tillë: A mund të formohet një palindromë me shkronjat që ndodhen nga kubi L_i në kubin R_i, duke u ndërruar vëndet nëse është e nevojshme? Pas çdo pyetje kubet rivendosen në rradhën që ishin në fillim. Ana në fund duhet të thotë se në sa prej pyetjeve përgjigja ka qenë "po".

Shënim: Palindromë quhet një varg me shkronja që është njëlloj po ta lexosh nga fillimi në fund dhe nga fundi në fillim. P.sh. këto janë palindroma: ANNA, RACECAR, AAA, X, kurse këto nuk janë: AB, FROG, YOYO.

Referenca: https://codingcompetitions.withgoogle.com/kickstart/round/0000000000050eda/0000000000119866

Shembull

```
$ cat input.txt
2
7 5
ABAACCA
3 6
4 4
2 5
6 7
3 7
3 5
XYZ
1 3
1 3
```

```
1 3
1 3
1 3

$ python3 prog.py < input.txt
Case #1: 3
Case #2: 0
```

Në rastin e parë kemi $\mathbf{N} = 7$ dhe $\mathbf{Q} = 5$.

- Në pyetjen e parë Ana duhet të përdorë shkronjat AACC. Me këto mund të formohet palindroma ACCA (ose CAAC).
- Në pyetjen e dytë duhet të përdorë shkronjën A, e cila është palindromë.
- Në pyetjen e tretë duhet të përdorë shkronjat BAAC, por me këto shkronja nuk mund të formohet asnjë palindromë, sido që ti rendisim.
- Në pyetjen e katërt duhet të përdorë shkronjat CA, me të cilat nuk mund të formohet palindromë.
- Në pyetjen e pestë duhet të përdorë shkronjat AAC, me të cilat mund të formohet palindroma ACA.

Gjithsej, mund tu përgjigjet "po" **3** pyetjeve.

Në rastin e dytë kemi $\mathbf{N} = 3$ dhe $\mathbf{Q} = 5$. Në secilën nga pyetjet duhen përdorur shkronjat XYZ, por nuk është e mundur që të formohet ndonjë palindromë me to. Kështu që rezultati është **0**.

Zgjidhja 1

```python
for t in range(int(input())):
    N, Q = [int(i) for i in input().split()]
    S = input()

    ans = 0
    for q in range(Q):
        l, r = [int(i) for i in input().split()]
        nr_odd = 0
        f = [0 for i in range(26)]
        i = l - 1
        while i < r:
            j = ord(S[i])-ord('A')
            f[j] += 1
            i += 1
        for j in range(26):
```

```
            if f[j] % 2 == 1:
                nr_odd += 1
        if nr_odd <= 1:
            ans += 1
    print('Case #{}: {}'.format(t+1, ans))
```

Sqarime

Mund të vëmë re që në një palindromë të gjitha shkronjat përsëriten një numër çift herësh, me përjashtim të shkronjës së mesit (kur kemi një numër tek shkronjash). Kështu që për të gjetur nëse me disa shkronja mund të formohet palindromë, mjafton të shikojmë se sa herë përsëritet secila shkronjë, dhe nëse secila prej tyre përsëritet një numër çift herësh, me përjashtim të njërës, atere mund të formohet palindromë, përndryshe jo.

Koha e kësaj zgjidhje është e rendit $O(Q * N)$.

Zgjidhja 2

```
for t in range(int(input())):
    N, Q = [int(i) for i in input().split()]
    S = input()

    F = []
    F.append([0 for i in range(26)])
    for i in range(0, N):
        F.append(F[-1][:])
        j = ord(S[i])-ord('A')
        F[-1][j] += 1

    ans = 0
    for q in range(Q):
        l, r = [int(i) for i in input().split()]
        nr_odd = 0
        for j in range(26):
            if (F[r][j] - F[l-1][j]) % 2 == 1:
                nr_odd += 1
        if nr_odd <= 1:
            ans += 1
    print('Case #{}: {}'.format(t+1, ans))
```

Sqarime

Në këtë zgjidhje fillimisht bëjmë disa llogaritje paraprake, të cilat na e thjeshtojnë punën për tju përgjigjur më pas pyetjeve. Për çdo pozicion të vargut të dhënë S ne gjejmë se sa herë është përsëritur secila shkronjë që nga fillimi e deri aty. Kjo gjendet shumë kollaj sepse mjafton të shtojmë 1 te përsëritjet e pozicionit të mëparshëm.

Duke ditur përsëritjet e shkronjave për secilin pozicion, ne mund të gjejmë shumë lehtë përsëritjet e secilës shkronjë midis dy pozicioneve të dhëna, duke zbritur nga pozicioni djathtas përsëritjet e pozicionit majtas.

Për punën paraprake na duhet një kohë $O(N)$ (mund ta bëjmë me një bredhje të vargut). Për secilën pyetje na duhet koha $O(26)$, dhe për të gjitha pyetjet koha $O(Q * 26)$ ose $O(Q)$ (meqë konstantet nuk ngrënë peshë). Gjithsej koha e zgjidhjes është $O(N + Q)$, që është shumë më e mirë se koha e zgjidhjes së mëparshme.

Zgjidhja 3

```
for t in range(int(input())):
    N, Q = [int(i) for i in input().split()]
    S = input()
    char_frequencies = {}
    nr_odd = [[0 for i in range(N)] for j in range(N)]
    nr_even = [[0 for i in range(N)] for j in range(N)]
    for i in range(0, N):
        char_frequencies = {S[i]: 1}
        nr_odd[i][i] = 1
        nr_even[i][i] = 0
        for j in range(i+1, N):
            c = S[j]
            f = char_frequencies.get(c, 0)
            char_frequencies[c] = f + 1
            if f == 0:
                nr_odd[i][j] = nr_odd[i][j-1] + 1
                nr_even[i][j] = nr_even[i][j-1]
            else:
                if f % 2 == 0:
                    nr_odd[i][j] = nr_odd[i][j-1] + 1
                    nr_even[i][j] = nr_even[i][j-1] - 1
                else:
                    nr_odd[i][j] = nr_odd[i][j-1] - 1
                    nr_even[i][j] = nr_even[i][j-1] + 1
    ans = 0
```

```
    for q in range(Q):
        l, r = [int(i) for i in input().split()]
        if nr_odd[l-1][r-1] <= 1:
            ans += 1
    print('Case #{}: {}'.format(t+1, ans))
```

Sqarime

Kjo zgjidhje llogarit paraprakisht se sa përsëritje teke ka për çdo segment. Edhe koha e kësaj zgjidhje është $O(N + Q)$. Por hapësira në memorje që duhet për këtë zgjidhje është e rendit $O(N^2)$, që është e papranueshme për vlera të mëdha të N (ndërkohë që hapësira në kujtesë që duhet për zgjidhjen e dytë është e rendit $O(N)$). Përveç kësaj, kjo zgjidhje është edhe pak më e komplikuar dhe më e gjatë se zgjidhja e dytë. Kështu që zgjidhja e dytë është më e mirë se kjo.

Detyra

Kemi një listë me pikë (numra natyrorë) të shkruara në një shirit të tejdukshëm. Kemi edhe një shirit me veprime modifikuese, me gjatësi më të madhe se lista e pikëve, mbi të cilin mund të vendoset shiriti i tejdukshëm dhe të llogariten veprimet totale.

Shiriti modifikues ka këto veprime:

- Pikë (**.**): nuk bën asnjë modifikim.
- Dyfisho pikët (**d**): dyfishon numrin e pikëve që përputhen me të.
- Trefisho pikët (**t**): trefishon numrin e pikëve që përputhen me të.
- Dyfisho totalin (**D**): dyfishon shumën e të gjitha pikëve, dhe zbatohet në fund.
- Trefisho totalin (**T**): trefishon shumën e të gjitha pikëve, dhe zbatohet në fund.

Shiriti me pikë vendoset mbi shiritin e modifikuesve në një pozicion të caktuar dhe bëhet shuma e pikëve, duke e dyfishuar ose trefishuar numrin e pikëve më parë, nëse përputhen me veprimet **d** dhe **t**. Kurse veprimet **D** dhe **T** bëjnë dyfishimin ose trefishimin e shumës totale dhe zbatohen gjithmonë në fund.

Gjeni pozicionin që jep vlerën më të madhe totale të pikëve dhe thoni sa është ajo vlerë.

Referenca: https://www.codechef.com/problems/MAXSCRAB

Shembull

```
$ cat input.txt
2
10
..d.t.D..d
```

```
10 11 12 9 8 10 11 15
22
dtDtTD..ddT.TtTdDT..TD
12297 5077 28888 17998 12125 27400 31219 21536

$ python3 prog.py < input.txt
270
35629632
```

Problemi 098

Kërkesa

Një lloj petëzash kanë një futurë të qeshur të bërë me çokollatë nga njëra anë, dhe asgjë nga ana tjetër. Petëzat janë duke u pjekur të vëna në rresht mbi një sipërfaqe të nxehtë. Në kuzhinë ndodhet një kthyes petëzash që mund të kthejë **K** petëza njëherësh. Kjo do të thotë që petëzat që janë në anën e fytyrës kthehen nga ana bosh, dhe anasjelltas, por pa ndryshuar rradhën e tyre.

Nuk është është e mundur që me këtë lopatëz të kthehen më pak se **K** petëza, as në anët e rreshtit, sepse sipërfaqja e sobës ka disa të ngritura në anë që e pengojnë këtë. Dmth me këtë lopatëz është e mundur që të kthejmë **K** petëzat e para të rreshtit, por jo vetëm **K-1** petëzat e para.

Nxënësi që ndihmon mjeshtrin, i cili akoma është duke e mësuar zanatin, ka kthyer disa petëza me lopatëzën e vogël, e cila mund të kthejë vetëm një petëz, dhe pastaj shkoi te banaku bashkë me të, pikërisht në kohën që disa klientë janë duke ardhur për të vizituar kuzhinën. Tani mjeshtrit i duhet që ti kthejë sa më shpejt të gjitha petëzat që të jenë me fytyrën e qeshur sipër, duke përdorur lopatëzën e gjerë.

Gjeni numrin më të vogël të kthimeve që duhen për ti bërë të gjitha petëzat me fytyrën sipër, ose thoni që kjo është e pamundur.

Referenca: https://code.google.com/codejam/contest/3264486/dashboard#s=p0&a=0

Shembull

```
$ cat input.txt
3
---+-++- 3
+++++ 4
-+-+- 4

$ python3 prog.py < input.txt
```

```
Case #1: 3
Case #2: 0
Case #3: IMPOSSIBLE
```

Shënja "-" tregon një petëz me faqen bosh sipër, kurse shenja "+" tregon një petëz me faqen me fytyrë sipër. Numri në fund tregon gjerësinë e lopatëzës (dmth sa petëza mund të kthejë njëherësh).

Në rastin e parë mund të kthejmë në fillim tre petëzat e para, pastaj tre të fundit, pastaj tre petëzat që ngelen me faqen bosh. Dmth me tre kthime mund ti rregullojmë të gjitha.

Në rastin e dytë janë të gjitha në rregull kështu që nuk është nevoja të kthejmë ndonjë gjë.

Në rastin e tretë është e pamundur që ti bëjmë të gjitha petëzat me faqen e duhur lart.

Zgjidhja

```
for t in range(int(input())):
    S, K = input().split()
    S = list(S)
    K = int(K)
    ans = 0
    i = 0
    while i < len(S) - K + 1:
        if S[i] == '-':
            ans += 1
            for j in range(i, i + K):
                S[j] = ('+' if S[j]=='-' else '-')
        i += 1
    while i < len(S):
        if S[i] == '-':
            ans = 'IMPOSSIBLE'
            break
        i += 1
    print('Case #{}: {}'.format(t+1, ans))
```

Sqarime

Fillimisht vërejmë se rradha e kthimit të petëzave nuk ka rëndësi. Dmth nëse bëjmë një kthim k1 dhe pastaj një kthim k2, rezultati do jetë i njëjtë sikur të bëjmë në fillim kthimin k2 dhe pastaj kthimin k1.

Gjithashtu, nuk ka kuptim që në një vënd të caktuar të bëjmë më shumë se një kthim, sepse kthimi i dytë do anullonte të parin, kështu që më mirë mos ti bënim fare këto dy kthime.

Kjo zgjidhje zbaton një strategji lakmitare (greedy). Nqs petëza që ndodhet e para nga e majta është "-" atere duhet patjetër një kthim i K petëzave të para për ta vendosur në rregull. Meqënëse dy kthime në të njëjtin vend s'kanë kuptim, atere vazhdojmë me petëzën e dytë dhe përsërisim të njëjtën gjë. Nqs pas kthimit të fundit ka mbetur ndonjë petëz e pa rregulluar nga K petëzat e fundit, atere është e pamundur që të rregullohen të gjitha petëzat.

Meqënëse kemi dy cikle brënda njëri-tjetrit, koha e kësaj zgjidhje është e madhësisë $O(N^2)$. Rasti më i keq është kur K=N/2. Zgjidhja më poshtë e ka kohën të madhësisë $O(N)$.

Zgjidhja 2

```
for t in range(int(input())):
    S, K = input().split()
    S = list(S)
    K = int(K)
    M = [0 for i in range(len(S) + K)]
    flips = 0
    i = 0
    while i < len(S) - K + 1:
        flips -= M[i]
        if (flips % 2 == 0 and S[i] == '-') or (flips % 2 == 1 and S[i] == '+'):
            flips += 1
            M[i+K] = 1
        i += 1
    ans = sum(M)
    while i < len(S):
        flips -= M[i]
        if (flips % 2 == 0 and S[i] == '-') or (flips % 2 == 1 and S[i] == '+'):
            ans = 'IMPOSSIBLE'
            break
        i += 1
    print('Case #{}: {}'.format(t+1, ans))
```

Sqarime

Kjo zgjidhje arrin një kohë të rendit $O(N)$ duke mos i kthyer të gjitha petëzat por duke kthyer vetëm të parën dhe duke mbajtur shënim numrin e herëve që duhen kthyer

petëzat pasardhëse. P.sh. nqs K=5 dhe petëza e parë është "`-`", atere ne nuk kthejmë 5 petëzat e para, por kthejmë vetëm të parën, shtojmë 1 te `flips` për të ditur që petëzat pasardhëse do kthehen një herë, dhe mbajmë shënim 1 te `M` në pozicionin e gjashtë, për të ditur që duhet ta pakësojmë `flips` me 1 kur të vijmë te petëza e gjashtë.

Vazhdojmë në këtë mënyrë për çdo petëz dhe shikojmë nëse `flips` është çift dhe prapë petëza është "`-`" atere duhet kthyer. Po ashtu, nëse `flips` është tek dhe petëza është "+" atere duhet kthyer edhe një herë.

Njëlloj si në zgjidhjen e mëparshme, nëse ka mbetur ndonjë petëz e pa rregulluar nga K petëzat e fundit, atere është e pamundur që të rregullohen të gjitha petëzat.

Detyra

Vanesa ka **N** xhingla në raftin e saj, të numëruara 1, 2, …, N nga e majta në të djathtë. Xhinglat janë të llojeve të ndryshme, ku çdo lloj shënohet me një numër. Xhingla që ndodhet në pozicionin **i** në raftin e saj është e llojit A_i.

Vanesa studjon jashtë dhe sot do kthehet që të vizitojë familjen. Ajo do që të marrë sa më shumë xhingla me vete, por ngaqë është me nxitim i duhet të rrëmbejë një varg të vazhdueshëm xhinglash. Po ta themi me gjuhë shkencore, Vanesa zgjedh me mënd dy pozicione **l** dhe **r** dhe vërvit në çantë të gjitha xhinglat që ndodhen në vëndet $l, l + 1, ..., r - 1, r$. Gjithashtu, për shkak të rregullave të aeroportit, kontrolluesit do hedhin poshtë **të gjitha** xhinglat e një lloji të caktuar, nëse ajo ka më shumë se **S** të tilla.

P.sh. nqs **S** = 2 dhe Vanesa ka me vete 6 xhingla, një të llojit 0, dy të llojit 1, dhe tre të llojit 2, asaj do ti lejohen xhinglat e llojit 0 dhe 1, por do ti hidhen poshtë të treja xhinglat e llojit 2.

Vanesa duhet të zgjedhë **l** dhe **r** të tilla që të marrë sa më shumë xhingla me vete. Sa është numri më i madh i xhinglave që mund të marrë me vete Vanesa?

Referenca: https://codingcompetitions.withgoogle.com/kickstart/round/0000000000050eda/00000000001198c1

Shembull

```
$ cat input.txt
4
6 2
1 1 4 1 4 4
8 1
1 2 500 3 4 500 6 7
10 1
100 200 8 8 8 8 8 300 400 100
12 2
```

```
40 50 1 1 1 60 70 2 2 2 80 90

$ python3 prog.py < input.txt
Case #1: 4
Case #2: 6
Case #3: 4
Case #4: 6
```

Për çdo rast testimi jepet numri i xhinglave **N** dhe maksimumi i lejuar në aeroport **S**. Pastaj jepet vargu i xhinglave sipas llojit të secilës prej tyre.

Në rastin e parë Vanesa duhet të zgjedhë l=2 dhe r=5. Kjo i lejon asaj të marrë xhinglat [1, 4, 1, 4], asnjë prej të cilave nuk hidhet poshtë në aeroport.

Në rastin e dytë Vanesa duhet të zgjedhë l=1 dhe r=8. Xhinglat e llojit 500 do hidhen poshtë meqë janë më shumë se S=1 të tilla, kështu që ajo do marrë me vete 6 xhingla.

Në rastin e tretë duhet të zgjedhë l=1 dhe r=9 dhe të sjallë në aeroport xhinglat [100, 200, 8, 8, 8, 8, 8, 300, 400]. Xhinglat e llojit 8 do hidhen poshtë dhe do ti ngelen 4 xhingla me vete.

Në rastin e katërt duhet të zgjedhë l=1 dhe r=12. Xhinglat e llojit 1 dhe 2 do hidhen poshtë meqë ka më shumë se S=2 të tilla, kështu që do ti mbeten 6 xhingla.

Problemi 099

Kërkesa

Një qytet (le të themi Manhattan) i ka rrugët paralele me njëra tjetrën dhe të barazlarguara, në drejtimin veri-jug dhe lindje-perëndim, të tilla që formojnë një rrjetë të rregullt. Le të themi që kjo rrjetë ka formë katrore dhe le ti numërojmë rrugët në drejtimin veri-jug me numrat 0, 1, 2, …, **Q**, dhe po ashtu edhe rrugët në drejtimin lindje-perëndim me numrat 0, 1, 2, …, **Q**. Njerëzit në një moment të caktuar ndodhen në një kryqëzim rrugësh, p.sh. (0, 0) ose (1, 2), dhe shkojnë në destinacion në rrugën më të shkurtër, duke lëvizur sipas drejtimit të rrugëve.

Ti do që të shkosh te një piceri e famshme, por nuk e di se ku ndodhet. Ndërkohë, **P** nga shokët e tu janë gjithashtu në lëvizje dhe ti mendon se shumica e tyre po shkojnë pikërisht aty. Me anë të një programi që keni instaluar në celular ti mund të shikosh vendodhjen e secilit prej tyre, p.sh. (x0, y0) dhe drejtimin në të cilin po lëvizin: **N** (north/veri) është drejtimi rritës i boshtit **y**, **S** (south/jug) drejtimi zbritës i boshtit **y**, **E** (east/lindje) drejtimi rritës i boshtit **x**, **W** (west/perëndim) drejtimi zbritës i boshtit **x**.

Gjeni kordinatat ku mund të ndodhet piceria. Nëse disa pozicione janë të mundshme, zgjidhni atë që ka kordinatat më të vogla.

Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000051706/000000000012295c

Shembull

```
$ cat input.txt
3
1 10
5 5 N
4 10
2 4 N
2 6 S
1 5 E
3 5 W
8 10
0 2 S
0 3 N
0 3 N
0 4 N
0 5 S
0 5 S
0 8 S
1 5 W

$ python3 prog.py < input.txt
Case #1: 0 6
Case #2: 2 5
Case #3: 0 4
```

Për çdo rast testimi jepen fillimisht numrat **P** dhe **Q**, pastaj janë P rreshta që përmbajnë kordinatat (x, y) dhe drejtimin e personave.

Në rastin e parë kemi vetëm një person në pozicionin (5, 5), i cili po shkon drejt veriut. Atere të gjitha pikat me kordinata $\mathbf{y > 5}$ janë të mundshme, por si përgjigje zgjidhet ajo që i ka kordinatat më të vogla: **(0, 6)**.

Në rastin e dytë janë 4 persona, të gjithë duke lëvizur në drejtim të pozicionit **(2, 5)**. Nuk ka pozicion tjetër që të ketë 4 persona që mund të jenë duke shkuar drejt tij.

Në rastin e tretë, 6 nga 8 personat po shkojnë drejt pozicionit **(0, 4)**. Nuk ka ndonjë pozicion tjetër që të ketë kaq shumë persona që mund të jenë duke shkuar drejt tij.

Zgjidhja 1

```
for t in range(int(input())):
    P, Q = [int(i) for i in input().split()]
    M = [[0 for i in range(Q+1)] for j in range(Q+1)]
    for p in range(P):
        x, y, d = input().split()
        x = int(x)
        y = int(y)
        x1, x2, y1, y2 = 0, Q+1, 0, Q+1
        if d == 'N': y1 = y + 1
        if d == 'S': y2 = y
        if d == 'E': x1 = x + 1
        if d == 'W': x2 = x
        for i in range(x1, x2):
            for j in range(y1, y2):
                M[i][j] += 1

    x, y, nr = 0, 0, 0
    for i in range(Q+1):
        for j in range(Q+1):
            if M[i][j] > nr:
                x, y, nr = i, j, M[i][j]
    print('Case #{}: {} {}'.format(t+1, x, y))
```

Sqarime

Kjo zgjidhje mban një matricë me përmasa (Q+1)x(Q+1), dhe për çdo person të dhënë shton +1 te destinacionet e mundshme ku ai mund të jetë duke shkuar. Në fund gjejmë atë pozicion që ka numrin e personave më të madh (dhe kordinatat x dhe y sa më të vogla).

Koha e kësaj zgjidhje është e rendit $O(P * Q^2)$, kurse hapësira që zë në kujtesë e rendit $O(Q^2)$. Për vlera të mëdha të Q kjo zgjidhje nuk është e përshtatshme.

Zgjidhja 2

```
for t in range(int(input())):
    P, Q = [int(i) for i in input().split()]
    Px = []
    Py = []
```

```
    for i in range(P):
        x, y, d = input().split()
        if d == 'E' or d == 'W':
            Px.append((int(x), d))
        else:
            Py.append((int(y), d))

    X = [0 for i in range(Q+1)]
    for (x, d) in Px:
        r1, r2 = (0, Q+1)
        if d == 'E':
            r1 = x + 1
        else:
            r2 = x
        for i in range(r1, r2):
            X[i] += 1
    Y = [0 for i in range(Q+1)]
    for (y, d) in Py:
        r1, r2 = (0, Q+1)
        if d == 'N':
            r1 = y + 1
        else:
            r2 = y
        for i in range(r1, r2):
            Y[i] += 1

    print('Case #{}: {} {}'.format(t+1, X.index(max(X)), Y.index(max(Y))))
```

Sqarime

Mund të vëmë re se në këtë problem drejtimet X dhe Y janë të pavarur nga njëri-tjetri. Dmth kur një person po lëviz në drejtimin **E** ose **W**, vetëm kordinata **x** e tij ka rëndësi, dhe ndikon në përcaktimin e kordinatës x të destinacionit, kurse kordinata **y** e tij nuk ka ndikim. E njëjta gjë nëse lëviz në drejtimin **N** ose **S**, vetëm kordinata **y** e tij ka rëndësi kurse kordinata **x** nuk ka rëndësi. Kështu që kordinatat x dhe y të destinacionit mund ti gjejmë në mënyrë të pavarur nga njëra-tjetra.

Kjo bën që për të gjetur zgjidhjen të mos mbajmë një matricë me përmasa (Q+1)x(Q+1), por dy lista me përmasa (Q+1). Si rezultat, hapësira në kujtesë që duhet për këtë zgjidhje është e rendit $O(Q)$. Por gjithashtu edhe koha e zgjidhjes është e rendit $O(P*Q)$.

Zgjidhja 3

```
def min_key_max_value(D):
    ''' Return the minimum key that has the max value '''
    min_key, max_value = 0, 0
    for k in D.keys():
        if (D[k] > max_value) or (D[k] == max_value and k < min_key):
            min_key, max_value = k, D[k]
    return min_key

for t in range(int(input())):
    P, Q = [int(i) for i in input().split()]
    Px = []
    Py = []
    for i in range(P):
        x, y, d = input().split()
        if d == 'E' or d == 'W':
            Px.append((int(x), d))
        else:
            Py.append((int(y), d))

    X = {0:0}
    for (x, d) in Px: X[x+1] = 0
    for (x, d) in Px:
        for k in X.keys():
            if (d == 'E' and x < k) or (d == 'W' and k < x):
                X[k] += 1
    Y = {0:0}
    for (y, d) in Py: Y[y+1] = 0
    for (y, d) in Py:
        for k in Y.keys():
            if (d == 'N' and y < k) or (d == 'S' and k < y):
                Y[k] += 1

    print('Case #{}: {} {}'.format(t+1, min_key_max_value(X), min_key_max_value(Y)))
```

Sqarime

Kjo zgjidhje bazohet në faktin që jo të gjitha kordinatat x nga 0 në Q janë të mundshme si përgjigje. Meqënëse problemi na kërkon kordinatën më të vogël, atere si përgjigje e mundshme mund të jetë pika 0 ose pikat (x+1), ku x është kordinata e një prej personave që po lëvizin në drejtimin W-E. E njëjta gjë mund të thuhet edhe për kordinatat y.

Si rezultat kjo zgjidhje ka një kohë të rendit $O(P^2)$ dhe hapësirë në kujtesë të rendit $O(P)$, të cilat janë më të mira se zgjidhja e mëparshme nëse P < Q.

Zgjidhja 4

```
def min_key_max_value(D):
    ''' Return the minimum key that has the max value '''
    min_key, max_value = 0, 0
    for k in D.keys():
        if (D[k] > max_value) or (D[k] == max_value and k < min_key):
            min_key, max_value = k, D[k]
    return min_key

for t in range(int(input())):
    P, Q = [int(i) for i in input().split()]
    Px = {}
    Py = {}
    for i in range(P):
        x, y, d = input().split()
        if d == 'E' or d == 'W':
            p = Px.get(int(x), {'W': 0, 'E': 0})
            p[d] += 1
            Px[int(x)] = p
        else:
            p = Py.get(int(y), {'S': 0, 'N': 0})
            p[d] += 1
            Py[int(y)] = p

    Kx = sorted(Px.keys())
    W = sum([Px[k]['W'] for k in Kx])
    X = {0: W}
    for k in Kx:
        W -= Px[k]['W']
        W += Px[k]['E']
        X[k+1] = W - Px.get(k+1, {'W': 0, 'E': 0})['W']

    Ky = sorted(Py.keys())
    S = sum([Py[k]['S'] for k in Ky])
    Y = {0: S}
    for k in Ky:
        S -= Py[k]['S']
        S += Py[k]['N']
```

```
        Y[k+1] = S - Py.get(k+1, {'S': 0, 'N': 0})['S']

    print('Case #{}: {} {}'.format(t+1, min_key_max_value(X), min_key_max_value(Y)))
```

Sqarime

Nqs personat që po lëvizin në drejtimin X i rendisim sipas kordinatës x të tyre, atere është e mundur që përgjigjen për drejtimin X ta gjejmë në kohë lineare. Por vetë renditja faktikisht ka një kohë pak më shumë se lineare, kështu koha e kësaj zgjidhje është e rendit $O(P * log(P))$ që është më e mirë se zgjidhja e mëparshme. Kurse hapësira në kujtesë është prapë e rendit $O(P)$.

Detyra

Një shkop kërcimi *pogo* ka një sustë në fund dhe me të mund të hidhesh nga një vënd në një vënd tjetër, duke qëndruar mbi të. Por shkopi që ju kanë bërë dhuratë është i veçantë sepse herën e parë kërcen me një largësi 1 njësi, herën e dytë 2 njësi, herën e tretë 3 njësi, e kështu me rradhë. Ju mund të kërceni në një nga këto drejtime: veri, jug, lindje, perëndim.

Nqs ndodheni në pikën (0, 0), si mund të shkoni në pikën (**X**, **Y**)? Si mund të shkoni me sa më pak kërcime?

Boshti X është sipas drejtimit S-N (jug-veri) dhe boshti Y është sipas drejtimit W-E (perëndim-lindje). Pika (X, Y) është e ndryshme nga (0, 0).

Referenca: https://code.google.com/codejam/contest/2437488/dashboard#s=p1&a=1

Shembull

```
$ cat input.txt
2
3 4
-3 4

$ python3 prog.py < input.txt
Case #1: ENWSEN
Case #2: ENSWN
```

Vini re që në rastin e parë ka edhe një rrugë tjetër që është më e shkurtër: WNSEN

Problemi 100

Kërkesa

Në një rrugë janë $\mathbf{N} + 2$ vënde parkimi në një rresht të vetëm. Vëndi i parë dhe i fundit janë gjithmonë të zënë nga policia bashkiake, kurse $\mathbf{N}$ të tjerat janë për përdoruesit.

Kur vjen një person për të parkuar ai përpiqet të zërë një vend që të jetë sa më larg nga makinat e tjera. Për të shmangur ngatërresat ata ndjekin disa rregulla të paracaktuara. Për çdo vënd parkimi bosh V llogarisin vlerat M_V dhe D_V, që janë numrat e vendeve bosh midis këtij vendi dhe vendeve të zëna në të majtë dhe në të djathtë. Pastaj shqyrtojnë ato vënde që kanë fqinjin më të largët, dmth ato vënde V për të cilat $min(M_V, D_V)$ është maksimale. Nëse është vetëm një vend i tillë, atere atë zgjedhin. Përndryshe zgjedhin prej tyre atë vend që e ka vlerën $max(M_V, D_V)$ maksimale. Nëse prapë ka disa vënde të tilla, atere zgjedhin atë prej tyre që është sa më majtas.

Në parkim vijnë $\mathbf{K}$ persona. Secili prej tyre parkon para se të vijë personi pasardhës (duke ndjekur rregullat më sipër), dhe asnjë prej tyre nuk largohet.

Kur parkon personi i fundit, sa do jenë vlerat $max(M_V, D_V)$ dhe $min(M_V, D_V)$?

Referenca: https://code.google.com/codejam/contest/3264486/dashboard#s=p2&a=2

Shembull

```
$ cat input.txt
7
4 2
5 2
6 2
1000 1000
1000 1
1000000 500000
1000000000000000000 500000000000000000

$ python3 prog.py < input.txt
Case #1: 1 0
Case #2: 1 0
Case #3: 1 1
Case #4: 0 0
Case #5: 500 499
Case #6: 1 0
Case #7: 1 0
```

Për çdo rast testimi jepen numrat $\mathbf{N}$ dhe $\mathbf{K}$.

Në rastin e parë situata është e tillë: **0.12.0** (me **0** shënojmë makinat e policisë, me pikë (**.**) shënojmë vëndet bosh, dhe me numrat 1 dhe 2 shënojmë vëndet që kanë zënë i pari dhe i dyti). Kështu që përgjigja është "**1 0**".

Në rastin e dytë situata është e tillë: **02.1..0**, kështu që përgjigja është "**1 0**".

Në rastin e tretë situata është e tillë: **0..1.2.0**, kështu që përgjigja është "**1 1**".

Në rastin e katërt çdo vënd do jetë i zënë në fund, pavarësisht nga rradha se si zihen, kështu që përgjigja është "**0 0**".

Zgjidhja 1

```
for t in range(int(input())):
    n, k = [int(i) for i in input().split()]

    segments = [n]      # sorted list of segment sizes
    for i in range(k):  # repeat k times
        # get one of the largest segments
        s = segments[0]
        del(segments[0])

        # split it into 2 other segments
        s1 = (s - 1) // 2
        s2 = s // 2

        # insert these 2 segments on the list
        if s1 > 0: segments.append(s1)
        if s2 > 0: segments.append(s2)

        # sort the list of segments
        segments.sort(reverse=True)

    # print the last two segments
    print('Case #{}: {} {}'.format(t+1, s2, s1))
```

Sqarime

Në një simulim naiv mund të mbanim një listë me vëndet e parkimit, për çdo vend bosh të gjenim sa vënde bosh ka në të majtë dhe në të djathtë, të gjenim maksimumin e minimumeve etj. (sipas rregullave të përshkruara në problem). Kjo do ishte edhe e vështirë për tu implementuar edhe jashtëzakonisht jo-efiçente.

Mund të vëmë re se pozicionet e segmenteve me vënde bosh nuk kanë rëndësi, rëndësi ka vetëm madhësia e tyre. Po ashtu, kur ndahet një segment, ai në të majtë është ose i barabartë ose një më pak se ai në të djathtë.

Në këtë zgjidhje ne i mbajmë segmentet (madhësinë e tyre) në një listë të renditur nga më i madhi te më i vogli. Për çdo person që vjen heqim nga lista segmentin më të madh (të parin), e ndajmë në dy segmente të tjera, i shtojmë këto në fund të listës, dhe e rendisim listën sërish.

Kjo zgjidhje është shumë e ngadaltë për testin 6. Koha e saj është e rendit $O(K * K * log(K))$ (sepse zakonisht $O(K * log(K))$ është koha që duhet për të renditur listën).

Zgjidhja 2

```
import heapq

for t in range(int(input())):
    n, k = [int(i) for i in input().split()]

    segments = [-1*n]       # sorted list of segment sizes
    heapq.heapify(segments)
    for i in range(k):  # repeat k times
        # get one of the largest segments
        s = -1 * heapq.heappop(segments)

        # split it into 2 other segments
        s1 = (s - 1) // 2
        s2 = s // 2

        # insert these 2 segments on the list
        if s1 > 0: heapq.heappush(segments, -1*s1)
        if s2 > 0: heapq.heappush(segments, -1*s2)

    # print the last two segments
    print('Case #{}: {} {}'.format(t+1, s2, s1))
```

Sqarime

Kur na intereson vetëm elementi më i madh, **heap** (pingu) është një strukturë më efiçente sesa një listë e renditur. Veprimet e saj janë të kohës $O(log(K))$, kështu që kjo e përmirëson kohën e zgjidhjes së parë dhe e bën $O(K * log(K))$. Kjo zgjidhje e kalon edhe testin 6, por ngec te testet e mëposhtme.

Vini re që moduli **heapq** i Python-it implementon vetëm një **min-heap**, dmth një pirg që në majë ka elementin më të vogël. Por ne na intereson elementi më i madh, kështu që të gjitha vlerat që fusim në pirg i shumëzojmë me **-1**, dhe vlerat që nxjerrim nga pirgu i shumëzojmë përsëri me **-1**. Në këtë mënyrë, praktikisht e bëjmë që të funksionojë si një **max-heap**.

Zgjidhja 3

```
import heapq

for t in range(int(input())):
    n, k = [int(i) for i in input().split()]

    segments = [-1*n]   # sorted list of segment sizes
    counts = { n: 1 }   # keep the count of each segment size
    heapq.heapify(segments)
    for i in range(k):  # repeat k times
        # remove one of the largest segments
        s = -1 * segments[0]
        counts[s] -= 1
        if counts[s] == 0:
            del(counts[s])
            heapq.heappop(segments)

        # split it into 2 other segments
        s1 = (s - 1) // 2
        s2 = s // 2

        # insert these 2 segments
        if s1 > 0:
            counts[s1] = counts.get(s1, 0) + 1
            if counts[s1] == 1:
                heapq.heappush(segments, -1*s1)
        if s2 > 0:
            counts[s2] = counts.get(s2, 0) + 1
            if counts[s2] == 1:
                heapq.heappush(segments, -1*s2)

    # print the last two segments
    print('Case #{}: {} {}'.format(t+1, s2, s1))
```

Sqarime

Mund të vihet re që ka shumë segmente me madhësi të njëjtë. P.sh. nqs N është përafërsisht 2^10 atere do jenë përafërsisht 2^9 segmente me gjatësi 2. Nqs do të mbanim shënim numrin e segmenteve me të njëjtën gjatësi, atere gjatësia e listës/pirgut `segments` do ishte shumë e vogël dhe veprimet në të do ishin më të shpejta. Kjo zgjidhje bën pikërisht këtë, dmth për një segment me një gjatësi të caktuar mban shënim se sa herë ndodhet ky segment në listë.

Mund të vërtetohet që gjatësia e pirgut `segments` është $O(log(K))$, kështu që koha e kësaj zgjidhje është e rendit $O(K * log(log(K)))$, që praktikisht është pothuajse $O(K)$.

Gjithsesi, te testi 7 kjo zgjidhje ngec.

Zgjidhja 4

```
import heapq

for t in range(int(input())):
    n, k = [int(i) for i in input().split()]

    segments = [-1*n]   # sorted list of segment sizes
    counts = { n: 1 }   # keep the count of each segment size
    heapq.heapify(segments)
    while k > 0:   # repeat until all the people are done
        # process all the largest segments of the same size
        s = -1 * heapq.heappop(segments)
        c = counts[s]
        del(counts[s])

        # process *c* people at once
        k -= c

        # split the segments
        s1 = (s - 1) // 2
        s2 = s // 2

        # insert these *c* _s1_ segments and *c* _s2_ segments
        if s1 > 0:
            counts[s1] = counts.get(s1, 0) + c
            if counts[s1] == c:
                heapq.heappush(segments, -1*s1)
        if s2 > 0:
```

```
            counts[s2] = counts.get(s2, 0) + c
            if counts[s2] == c:
                segments.append(s2)
                heapq.heappush(segments, -1*s2)

    # print the last two segments
    print('Case #{}: {} {}'.format(t+1, s2, s1))
```

Sqarime

Mund të vërejmë gjithashtu se të gjithë segmentet me gjatësi të njëjtë përpunohen njëri pas tjetrit (sepse gjithmonë marrim një nga segmentet që kanë gjatësinë më të madhe). Kështu që në vend që ti përpunojmë veç e veç, mund ti përpunojmë të gjithë së bashku. Kjo i shkurton shumë përsëritjet e ciklit, sepse në një hap të vetëm ne mund të përpunojmë p.sh. 2^10 segmente, në vend që të bëjmë 2^10 hapa për çdo segment.

Koha e kësaj zgjidhje është e rendit $O(log(K) * log(log(K)))$ ose praktikisht $O(log(K))$. Kjo është jashtëzakonisht e shpejtë edhe për vlera shumë të mëdha të K, dhe përfundon shumë shpejt të gjitha rastet e testimit.

Zgjidhja 5

```
for t in range(int(input())):
    n, k = [int(i) for i in input().split()]

    people = k
    segs = 1
    size = n
    free = n
    while people > segs:
        people -= segs
        free -= segs
        size = size // 2
        segs *= 2

    if people > free - segs*(size - 1):
        size -= 1
    print('Case #{}: {} {}'.format(t+1, size // 2, (size - 1) // 2))
```

Sqarime

```
for t in range(int(input())):
    n, k = [int(i) for i in input().split()]

    people = k  # number of people to come
    segs = 1    # total number of segments
    size = n    # size of the largest segment
    free = n    # total number of free places
    while people > segs:
        people -= segs    # one person splits each segment
        free -= segs      # each of these persons occupies a place
        size = size // 2  # halve the size of the segments
        segs *= 2         # double the number of the segments

    # Now we have people <= segs. The last person will split one of
    # these segments. Some of these segments are of size "size" and
    # the others of size "size-1". Which one of these will split the
    # last person? The number of segments of size "size" is
    # "free - segs*(size - 1)". If there are more people left than
    # this number, then the last person will split a segment of size
    # "size - 1"
    if people > free - segs*(size - 1):
        size -= 1
    print('Case #{}: {} {}'.format(t+1, size // 2, (size - 1) // 2))
```

Po ta shikojmë me kujdes zgjidhjen e mëparshme (zgjidhjen 4) mund të vërejmë që në çdo hap numri i segmenteve që përpunohen është sa dyfishi i segmenteve që u përpunuan në hapin e mëparshëm, kurse gjatësia e segmenteve është pothuajse sa gjysma e gjatësisë së segmenteve që përpunohen në hapin e mëparshëm (mund të ndryshojnë ndoshta me 1). Bazuar në këtë vëzhgim, mund ta heqim nga përdorimi listën (pirgun) me gjatësitë e segmenteve dhe listën (hartën) me numrin e segmenteve të çdo gjatësie.

Zgjidhja që rezulton është më e shkurtër, më e thjeshtë dhe më elegante se zgjidhja e mëparshme. Megjithatë koha e kësaj zgjidhje është e rendit $O(log(K))$, njëlloj sa ajo e zgjidhjes së mëparshme. Kështu që edhe kjo zgjidhje jep rezultat shumë shpejt, sado e madhe të jetë K-ja dhe N-ja.

Detyra

Jepet një varg me numra natyrorë $A_1, A_2, ..., A_N$. Ju mund të zgjidhni një numër natyror **X** dhe të bëni me të veprimin **bitwise XOR** me secilin nga numrat e vargut, pastaj të merrni shumën e këtyre numrave. Sa është vlera më e vogël e mundshme e kësaj shume.

Referenca: https://www.codechef.com/LTIME71B/problems/MINARRS

Shembull

```
$ cat input.txt
3
5
2 3 4 5 6
4
7 7 7 7
3
1 1 3

$ python3 prog.py < input.txt
14
0
2
```

Në rastin e parë mund të zgjedhin X = 6, dhe vargu bëhet (4, 5, 2, 3, 0).

Në rastin e dytë mund të zgjedhim X = 7 dhe të gjithë numrat e vargut bëhen 0.

Në rastin e tretë mund të zgjedhim X = 1, dhe vargu bëhet (0, 0, 2), shuma e të cilit është 2.

12 Shtojca 1

Problemi 00

Kërkesa

Bëni një program i cili nxjerr diferencën e dy numrave të dhënë, nëse i pari është më i madh se i dyti, ose shumën e tyre në të kundërt.

Referenca: https://www.codechef.com/problems/DIFFSUM

Shembull

```
$ cat input.txt
82
28

$ python3 prog.py < input.txt
54
```

Zgjidhja

```
n1 = int(input())
n2 = int(input())
if n1 > n2:
    print(n1 - n2)
else:
    print(n1 + n2)
```

Problemi 01

Kërkesa

Bëni një program i cili merr një numër dhe e rrit me 1 nëse ai është i plotpjesëtueshëm me 4, përndryshe e zvogëlon me 1.

Referenca: https://www.codechef.com/problems/DECINC

Shembull

```
$ cat input.txt
2
4
5

$ python3 prog.py < input.txt
5
4
```

Zgjidhja

```
T = int(input())
for t in range(T):
    n = int(input())
    if n % 4 == 0:
        print(n+1)
    else:
        print(n-1)
```

Problemi 02

Kërkesa

Bëni një program që gjen nëse 4 numra natyrorë **a**, **b**, **c**, **d**, mund të jenë brinjë të një drejtkëndëshi.

Referenca: https://www.codechef.com/problems/RECTANGL

Shembull

```
$ cat3 input.txt
3
1 1 2 2
3 2 2 3
1 2 2 2
```

```
$ python3 prog.py < input.txt
YES
YES
NO
```

Zgjidhja

```
T = int(input())
for t in range(T):
    a, b, c, d = map(int, input().split())
    if (a == b and c == d) or (a == c and b == d) or (a == d and b == c):
        print('YES')
    else:
        print('NO')
```

Problemi 03

Kërkesa

Një numër të dhënë **X** e shumëzojmë me 2 deri sa të bëhet i pjesëtueshëm me 10. Gjeni numrin më të vogël të shumëzimeve deri sa të plotësohet ky kusht, ose përcaktoni që kjo është e pamundur.

Referenca: https://www.codechef.com/problems/TWOVSTEN

Shembull

```
$ cat input.txt
3
10
25
1

$ python3 prog.py < input.txt
0
1
-1
```

Kur është e pamundur e shënojmë përgjigjen -1.

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    if n % 10 == 0:
        print(0)
    elif n % 5 == 0:
        print(1)
    else:
        print(-1)
```

Problemi 04

Kërkesa

Bëni një program që merr një shkronjë të alfabetit anglisht, dhe nxjerr "Vowel" nëse është zanore, ose "Consonant" nëse është bashkëtingëllore.

Zanoret në alfabetin anglisht janë: 'A', 'E', 'I', 'O', 'U'.

Referenca: https://www.codechef.com/problems/VOWELTB

Shembull

```
$ cat input.txt
2
Z
A

$ python3 prog.py < input.txt
Consonant
Vowel
```

Zgjidhja 1

```
for _ in range(int(input())):
    c = input()
    if c == 'A' or c == 'E' or c == 'I' or c == 'O' or c == 'U':
        print('Vowel')
    else:
        print('Consonant')
```

Zgjidhja 2

```
for _ in range(int(input())):
    c = input()
    if c in ['A', 'E', 'I', 'O', 'U']:
        print('Vowel')
    else:
        print('Consonant')
```

Zgjidhja 3

```
for _ in range(int(input())):
    c = input()
    if c in 'AEIOU':
        print('Vowel')
    else:
        print('Consonant')
```

Zgjidhja 4

```
for _ in range(int(input())):
    print('Vowel') if input() in 'AEIOU' else print('Consonant')
```

Problemi 05

Kërkesa

Bëni një program që merr gjatësinë dhe lartësinë e një drejtkëndëshi, dhe gjen se kush është më i madh: sipërfaqja apo perimetri. Programi duhet të nxjerrë 'Area' nëse është më e madhe sipërfaqja, 'Peri' nëse është më i madh perimetri, ose 'Eq' nëse të dyja janë të barabarta. Gjithashtu duhet të nxjerrë edhe vlerën më të madhe të llogaritur (sipërfaqen ose perimetrin.

Referenca: https://www.codechef.com/problems/AREAPERI

Shembull

```
$ cat input.txt
3
1 2
5 6
4 4

$ python3 prog.py < input.txt
```

Peri 6 Area 30 Eq 16

Zgjidhja

```
t = int(input())
while t > 0:
    a, b = map(int, input().split())
    s = a * b
    p = a + b + a + b
    if s > p:
        print('Area', s)
    elif p > s:
        print('Peri', p)
    else:
        print('Eq', s)
    t -= 1
```

Problemi 06

Kërkesa

Bëni një program që merr një numër natyror dhe nxjerr **1** nëse numri ka një shifër, **2** nëse numri ka dy shifra, **3** nëse numri ka tre shifra. dhe **More than 3 digits** nëse numri ka më shumë se tre shifra.

Referenca: https://www.codechef.com/problems/HOWMANY#

Shembull

```
$ cat input.txt
5
```

```
9
17
235
62627
31

$ python3 prog.py < input.txt
1
2
3
More than 3 digits
2
```

Zgjidhja

```python
for _ in range(int(input())):
    n = int(input())
    if n < 10:
        print(1)
    elif n < 100:
        print(2)
    elif n < 1000:
        print(3)
    else:
        print('More than 3 digits')
```

Problemi 07

Kërkesa

Një universitet përdor sistemin e vlerësimit me 5 pikë. Në një provim studentët mund të vlerësohen me pikët nga 2 (që është ngelës) deri në 5 (që është shkëlqyer). Universiteti u jep bursa studentëve nqs plotësojnë këto kushte:

- Nuk janë ngelës në asnjë lëndë.
- Kanë të paktën një lëndë ku janë të shkëlqyer.
- Mesataren e të gjitha lëndëve e kanë jo më pak se **4.0**

Nqs jepen vlerësimet e studentëve në të gjitha provimet, bëni një program që gjen se cilët prej tyre përfitojnë bursa.

Referenca: https://www.codechef.com/problems/EGRANDR

Shembull

```
$ cat input.txt
2
5
3 5 4 4 3
5
3 4 4 4 5

$ python3 prog.py < input.txt
No
Yes
```

Studenti i parë e ka mesataren 3.8, kështu që nuk përfiton bursë.

Studenti i dytë i plotëson të gjitha kushtet, kështu që përfiton bursë.

Zgjidhja 1

```
t = int(input())
while t > 0:
    t -= 1
    n = int(input())
    P = [int(p) for p in input().split()]
    has_2 = False
    has_5 = False
    s = 0
    for p in P:
        if p == 2:
            has_2 = True
        if p == 5:
            has_5 = True
        s += p
    if s/n >= 4.0 and has_5 and not has_2:
        print('Yes')
    else:
        print('No')
```

Zgjidhja 2

```
t = int(input())
while t > 0:
    t -= 1
    n = int(input())
    P = [int(p) for p in input().split()]
    if (sum(P) / n >= 4.0) and (5 in P) and (not 2 in P):
        print('Yes')
    else:
        print('No')
```

Problemi 08

Kërkesa

Mësuesi u ka dhënë nxënësve për të lexuar një libër historik, dhe u ka kërkuar si detyrë që të shkruajnë emrat e disa prej personaliteteve të shquar që përmenden në të. Mirëpo kur nxënësit dorëzuan detyrat mësuesi mbeti i pakënaqur sepse nuk i kishin formatuar emrat siç duhet.

Një emër mund të ketë tre pjesë, emrin e personit, emrin e babait dhe mbiemrin. Nga këto, mbiemri nuk mungon asnjëherë, kurse dy të tjerët mund edhe të mos shkruhen. Të treja pjesët duhet të fillojnë me shkronjë të madhe, kurse shkronjat e tjera duhet të jenë të vogla. Dy pjesët e para duhet të shkruhen me shkurtim, vetëm shkronja e parë dhe një pikë pas saj.

Bëni një program që lexon disa emra dhe i shkruan siç duhet.

Referenca: https://www.codechef.com/problems/NITIKA

Shembull

```
$ cat input.txt
3
gandhi
mahatma gandhI
Mohndas KaramChand gandhi

$ python3 prog.py < input.txt
Gandhi
M. Gandhi
M. K. Gandhi
```

Zgjidhja

```
t = int(input())
while t > 0:
    t -= 1
    L = input().split()
    if len(L) == 1:
        print(L[0].capitalize())
    elif len(L) == 2:
        print(L[0][0].upper() + '. ' + L[1].capitalize())
    else:
        print(L[0][0].upper() + '. ' + L[1][0].upper() + '. ' + L[2].capitalize())
```

Problemi 09

Kërkesa

Në një olimpiadë janë **P** pjesëmarrës. Nqs një problem zgjidhet nga më shumë se gjysma e pjesëmarrësve, ky problem quhet i thjeshtë. Nqs një problem zgjidhet nga e shumta **P/10** pjesëmarrës, quhet i vështirë. Teza e olimpiadës quet e balancuar nëse ka vetëm një problem të thjeshtë dhe vetëm 2 të vështirë.

Bëni një program që merr numrin e pjesëmarrësve që kanë zgjidhur secilin problem të olimpiadës, dhe nxjerr nëse teza e olimpiadës ka qenë e balancuar ose jo.

Referenca: https://www.codechef.com/problems/PERFCONT

Shembull

```
$ cat input.txt
6
3 100
10 1 100
3 100
11 1 100
3 100
10 1 10
3 100
10 1 50
4 100
50 50 50 50
4 100
```

```
1 1 1 1

$ python3 prog.py < input.txt
yes
no
no
yes
no
no
```

1. Janë dy problema të vështira dhe një i lehtë, kështu që është e balancuar
2. Problemi i dytë është i vështirë dhe i treti është i lehtë. I pari nuk është as i lehtë as i vështirë. Teza nuk është e balancuar.
3. Tre problema të vështira. Jo e balancuar.
4. Dy të parat janë të vështira, i treti është i lehtë. E balancuar.
5. Katër problema të lehta. Jo e balancuar.
6. Katër problema të vështira. Jo e balancuar.

Zgjidhja

```python
for _ in range(int(input())):
    n, p = map(int, input().split())
    L = list(map(int, input().split()))
    n1 = 0    # problemat e lehta
    n2 = 0    # problemat e veshtira
    for i in L:
        if i >= p // 2:
            n1 += 1
        if i <= p // 10:
            n2 += 1
    if n1 == 1 and n2 == 2:
        print('yes')
    else:
        print('no')
```

Problemi 10

Kërkesa

Në një restorant shkojnë shumë programues. Ata i lënë shënimet e tyre në librin e përshtypjeve në formatin binar (me zero dhe njësha). Por shefi i restorantit nuk ka

haber nga formati binar, kështu që i ka dhënë karar që një përshtypje ta quajë pozitive nëse përmban vargun **010** ose **101**, dhe ta quajë negative në të kundërt.

Bëni një program që merr përshtypjet në formatin me zero njësha dhe gjen nëse janë të mira ose të këqija.

Referenca: https://www.codechef.com/problems/ERROR

Shembull

```
$ cat input.txt
2
11111110
10101010101010

$ python3 prog.py < input.txt
Bad
Good
```

Rasti i parë nuk përmban asnjë nënvarg 010 ose 101. Rasti i dytë i përmban të dyja.

Zgjidhja 1

```
for _ in range(int(input())):
    p = input()
    for i in range(len(p) - 2):
        if p[i]=='1' and p[i+1]=='0' and p[i+2]=='1':
            print('Good')
            break
        if p[i]=='0' and p[i+1]=='1' and p[i+2]=='0':
            print('Good')
            break
    else:
        print('Bad')
```

Zgjidhja 2

```
for _ in range(int(input())):
    p = input()
    for i in range(len(p) - 2):
```

```
        if p[i:i+3]=='101' or p[i:i+3]=='010':
            print('Good')
            break
    else:
        print('Bad')
```

Zgjidhja 3

```
for _ in range(int(input())):
    p = input()
    if '101' in p or '010' in p:
        print('Good')
    else:
        print('Bad')
```

Problemi 11

Kërkesa

Xheni dhe Ylli ishin të gëzuar për festat që po afronin dhe do punonin sëbashku për përgatitjet e nevojshme. Janë **N** punë për tu kryer. Për një punë **i** Xhenit i duhen X_i sekonda kurse Yllit i duhen Y_i sekonda. Ata vendosën që ti bënin punët në mënyrë të alternuar. Nqs punën e parë e bën Xheni, punën e dytë e bën Ylli (ndërkohë që Xheni shplodhet), punën e tretë e bën prapë Xheni, e kështu me rradhë. Nqs punën e parë e bën Ylli, të dytën e bën Xheni, e kështu me rradhë.

Na jepen kohët që i duhen Xhenit dhe Yllit për çdo punë, gjeni kohën më të vogël që u duhet për ti mbaruar të gjitha punët.

Referenca: https://www.codechef.com/problems/XENTASK

Shembull

```
$ cat input.txt
1
3
2 1 2
3 2 1

$ python3 prog.py < input.txt
5
```

Nqs punën e parë do ta bëjë Xheni do duhen 2+2+2=6 sekonda. Kurse po ta bëjë Ylli punën e parë do duhen 3+1+1=5 sekonda. Koha më e vogël është 5 sekonda.

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    X = [int(i) for i in input().split()]
    Y = [int(i) for i in input().split()]
    s1 = s2 = 0
    for i in range(n):
        if i % 2 == 0:
            s1 += X[i]
            s2 += Y[i]
        else:
            s1 += Y[i]
            s2 += X[i]
    print(min(s1, s2))
```

Sqarime

Gjejmë shumën sipas të dyja mënyrave, pastaj printojmë shumën më të vogël.

Problemi 12

Kërkesa

Majkua është pjesëtar i shoqatës së koleksionistëve të pullave postare. Shoqata ka **N** pjesëtarë dhe secili prej tyre ka C_i pulla në koleksionin e tij. Majkua pretendon se të gjitha këto pulla poste mund të rishpërndahen në mënyrë të tillë që pjesëtari **i** i shoqatës të ketë **i** pulla poste. Kontrolloni nëse Majkua ka të drejtë.

Referenca: https://www.codechef.com/problems/DIVIDING

Shembull

```
$ cat input.txt
2
5
7 4 1 1 2
```

```
5
1 1 1 1 1

$ python3 prog.py < input.txt
YES
NO
```

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    C = [int(c) for c in input().split()]
    s1 = s2 = 0
    for i in range(n):
        s1 += C[i]
        s2 += (i + 1)
    print('YES') if s1==s2 else print('NO')
```

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    C = [int(c) for c in input().split()]
    print('YES') if sum(C) == n*(n + 1) // 2 else print('NO')
```

Problemi 13

Kërkesa

Bëni një program që merr dy numra të plotë dhe një veprim arithmetic dhe nxjerr rezultatin.

Referenca: https://www.codechef.com/problems/URCALC

Shembull

```
$ cat input.txt
4
8 / 2
```

```
5 + 3
7 * 5
1 - 9

$ python3 prog.py < input.txt
4.0
8
35
-8
```

Zgjidhja 1

```python
for _ in range(int(input())):
    a, op, b = input().split()
    a = int(a)
    b = int(b)
    if op == '+':
        r = a + b
    elif op == '-':
        r = a - b
    elif op == '*':
        r = a * b
    elif op == '/':
        r = a / b
    print(r)
```

Zgjidhja 2

```python
for _ in range(int(input())):
    print(eval(input()))
```

Problemi 14

Kërkesa

Në një konkurs Alisa dhe Bobi bëjnë **N** gara vrapimi, ku për çdo **i** nga 1 në N, Alisa e mbaron garën **i** në kohën A_i minuta, kurse Bobi e mbaron në kohën B_i minuta. Konkursin e fiton ai që ka shumën e kohëve më të vogël. Sipas rregullave, Alisa mund të heqë nga

llogaritja njërën prej garave, kë të dojë. Po ashtu dhe Bobi. Gjeni se cili prej tyre e fiton konkursin. Nqs shuma e kohëve del e barabartë, atere janë barazim.

Referenca: https://www.codechef.com/problems/CO92JUDG

Shembull

```
$ cat input.txt
3
5
3 1 3 3 4
1 6 2 5 3
5
1 6 2 5 3
3 1 3 3 4
3
4 1 3
2 2 7

$ python3 prog.py < input.txt
Alice
Bob
Draw
```

Në rastin e parë kemi 5 gara. Koha e Alisës është $3 + 1 + 3 + 3 + 0 = 10$. Koha e Bobit është $1 + 0 + 2 + 5 + 3 = 11$. Fiton Alisa.

Në rastin e tretë kemi $0 + 1 + 3 = 4$ kohën e Alisës dhe $2 + 2 + 0 = 4$ kohën e Bobit, kështu që dalin barazim.

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    A = [int(i) for i in input().split()]
    B = [int(i) for i in input().split()]
    s1 = 0
    m1 = 0
    for a in A:
        s1 += a
        if a > m1:
            m1 = a
```

```
    s2 = 0
    m2 = 0
    for b in B:
        s2 += b
        if b > m2:
            m2 = b
    if s1 - m1 < s2 - m2:
        print('Alice')
    elif s1 - m1 > s2 - m2:
        print('Bob')
    else:
        print('Draw')
```

Zgjidhja 2

```
for _ in range(int(input())):
    input()
    A = [int(i) for i in input().split()]
    B = [int(i) for i in input().split()]
    d = (sum(A) - max(A)) - (sum(B) - max(B))
    if d < 0:
        print('Alice')
    elif d > 0:
        print('Bob')
    else:
        print('Draw')
```

Problemi 15

Kërkesa

Jepet një varg me zero-njësha. Numërojmë kalimet 0-1 ose 1-0 në këtë varg, duke e menduar vargun si ciklik (shifrën e fundit ta mendojmë si të ngjitur me shifrën e parë). Këtë varg e quajmë *uniform* nëse ka të shumtën 2 kalime të tilla, përndryshe e quajmë *jo-uniform*. Gjeni nëse vargu i dhënë është uniform ose jo.

Referenca: https://www.codechef.com/problems/STRLBP

Shembull

```
$ cat input.txt
```

```
4
00000000
10101010
10000001
10010011

$ python3 prog.py < input.txt
uniform
non-uniform
uniform
non-uniform
```

Zgjidhja

```
for _ in range(int(input())):
    V = input()
    nr = 0
    for i in range(len(V) - 1):
        if V[i] != V[i+1]:
            nr += 1
    if V[0] != V[-1]:
        nr += 1
    print('non-uniform') if nr > 2 else print('uniform')
```

Problemi 16

Kërkesa

Jepen kordinatat e tre pikave në një plan: $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Gjeni nëse këto pika janë kulme të një trekëndëshi kënddrejtë.

Referenca: https://www.codechef.com/problems/RIGHTRI

Shembull

```
$ cat input.txt
5
0 5 19 5 0 0
17 19 12 16 19 0
5 14 6 13 8 7
0 4 0 14 3 14
```

```
0 2 0 14 9 2

$ python3 prog.py < input.txt
yes
no
no
yes
yes
```

Janë dhënë numrat x1, y1, x2, y2, x3, y3.

Zgjidhja 1

```
for _ in range(int(input())):
    x1, y1, x2, y2, x3, y3 = map(int, input().split())
    d1 = (x1 - x2)**2 + (y1 - y2)**2
    d2 = (x1 - x3)**2 + (y1 - y3)**2
    d3 = (x2 - x3)**2 + (y2 - y3)**2
    if max(d1, d2, d3) * 2 == d1 + d2 + d3:
        print('yes')
    else:
        print('no')
```

Sqarime

Përdorim teoremën e Pitagorës ($a^2 + b^2 = c^2$).

Zgjidhja 2

```
for _ in range(int(input())):
    x1, y1, x2, y2, x3, y3 = map(int, input().split())
    p1 = (x1 - x2)*(x1 - x3) + (y1 - y2)*(y1 - y3)
    p2 = (x1 - x2)*(x2 - x3) + (y1 - y2)*(y2 - y3)
    p3 = (x1 - x3)*(x2 - x3) + (y1 - y3)*(y2 - y3)
    if p1*p2*p3 == 0:
        print('yes')
    else:
        print('no')
```

Sqarime

Përdorim faktin që prodhimi i dy vektorëve pingulë është zero.

Problemi 17

Kërkesa

Ti ke shoqëri me Anin dhe Benin dhe shkëmben mesazhe me ta. Ani shkruan gjithmonë me shkronja të vogla, kurse Beni shkruan gjithmonë me shkronja të mëdha.

Sapo të ka ardhur një mesazh që ka edhe shkronja të vogla edhe të mëdha. Në fakt Ani dhe Beni mund të bëjnë ndonjëherë edhe gabime gjatë shkrimit, dhe në vend të një shkronje të vogël të bëjnë një të madhe, ose anasjelltas. Por jo më shumë se **K** të tilla në një mesazh.

Gjeni nëse këtë mesazh mund ta ketë shkruar Ani, ose Beni, ose të dy, ose asnjëri.

Referenca: https://www.codechef.com/problems/GOODBAD

Shembull

```
$ cat input.txt
4
5 1
frauD
5 1
FRAUD
4 4
Life
10 4
sTRAWBerry

$ python3 prog.py < input.txt
Ani
Beni
Secili
Asnjeri
```

1. Mund të ketë vetëm 1 gabim, kështu që këtë mesazh duhet ta ketë shkruar Ani (por shkronja 'd' gabimisht i ka dalë e madhe). Nuk mund ta ketë shkruar Beni, sepse në këtë rast Beni do kishte bërë 4 gabime (gjë që nuk ka mundësi).
2. Mund të ketë vetëm 1 gabim, kështu që duhet ta ketë shkruar Beni. Po ta kishte shkruar Ani do kishte bërë 5 gabime (gjë që s'ka mundësi).

3. Meqënëse janë 4 gabime të mundshme, atere mund ta ketë shkruar edhe Ani edhe Beni dhe secili të ketë bërë gabime (Ani 1 ose Beni 3).
4. Janë 4 gabime të mundshme, por nqs do ta kishte shkruar Ani do kishte bërë 5 gabime, dhe po ta kishte shkruar Beni do kishte bërë 5 gabime. Kështu që asnjëri prej tyre nuk mund ta ketë dërguar këtë mesazh.

Zgjidhja 1

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    S = input()
    l = u = 0
    for c in S:
        if c.islower():
            l += 1
        else:
            u += 1
    if l <= k and u <= k:
        print('Secili')
    elif l > k and u > k:
        print('Asnjeri')
    elif l <= k and u > k:
        print('Ani')
    else: # l > k and u <= k
        print('Beni')
```

Zgjidhja 2

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    S = input()
    l = sum([1 for c in S if c.islower()])
    u = n - l
    if l <= k and u <= k:
        print('Secili')
    elif l > k and u > k:
        print('Asnjeri')
    elif l <= k and u > k:
        print('Ani')
```

```
    else: # l > k and u <= k
        print('Beni')
```

Problemi 18

Kërkesa

Kemi një listë me numra. Duke përdorur shifrat e këtyre numrave mund të formojmë një numër tjetër. Cili është numri më i madh që mund të formojmë?

Referenca: https://www.codechef.com/DARG2019/problems/LOALL

Shembull

```
$ cat input.txt
2
2 23 1
3 7 11 20

$ python3 prog.py < input.txt
3221
732110
```

Zgjidhja

```
for _ in range(int(input())):
    D = []    # list of all digits
    for n in input().split():
        D.extend(list(n))
    D.sort(reverse=True)
    print(*D, sep='')
```

Problemi 19

Kërkesa

Jepen 2 numra **a** dhe **n**. Bëni një program që gjen $k = a^n$, pastaj merr shumën e shifrave të **k** dhe gjen nëse kjo shumë është numër i thjeshtë ose jo.

Referenca: https://www.codechef.com/problems/GMPOW

Shembull

```
$ cat input.txt
2
2 4
5 3

$ python3 prog.py < input.txt
1
0
```

Në rastin e parë, $2^4 = 16$ dhe $1 + 6 = 7$ është numër i thjeshtë.

Në rastin e dytë, $5^3 = 125$ dhe $1 + 2 + 5 = 8$ nuk është numër i thjeshtë.

Zgjidhja

```python
import math

def is_prime(n):
    if n == 1: return False
    if n % 2 == 0 and n > 2:
        return False
    for i in range(3, int(math.sqrt(n)) + 1, 2):
        if n % i == 0:
            return False
    return True

for _ in range(int(input())):
    a = int(input())
    n = int(input())
    k = a**n
    s = sum([int(d) for d in list(str(k))])
    print(1) if is_prime(s) else print(0)
```

Problemi 20

Kërkesa

Para se të fillojë mësimi mësuesi bën apelin, duke i thirrur emrat e nxënësve një nga një. Nqs dy nxënës kanë të njëjtin emër (por mbiemra të ndryshëm) atere mësuesi e thërret emrin të plotë, me emër dhe mbiemër. Përndryshe thërret vetëm emrin (pa mbiemër).

Bëni një program që merr emrat e nxënësve dhe nxjerr se si i thërret mësuesi ata.

Referenca: https://www.codechef.com/LTIME71B/problems/ATTND

Shembull

```
$ cat input.txt
1
4
hasan jaddouh
farhod khakimiyon
kerim kochekov
hasan khateeb

$ python3 prog.py < input.txt
hasan jaddouh
farhod
kerim
hasan khateeb
```

Kemi 1 rast testimi, i cili ka 4 emra.

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    L = []
    F = []
    for i in range(n):
        first, last = input().split()
        L.append([first, last])
        F.append(first)
    for name in L:
        first, last = name
        if F.count(first) > 1:
            print(first, last)
        else:
            print(first)
```

Koha e rendit $O(N^2)$

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    L = []
    F = {}
    for i in range(n):
        first, last = input().split()
        L.append([first, last])
        F[first] = F.get(first, 0) + 1
    for name in L:
        first, last = name
        if F[first] > 1:
            print(first, last)
        else:
            print(first)
```

Koha e rendit $O(N * log(N))$

Problemi 21

Kërkesa

Një byrektore do zhvendoset nga qyteti A në qytetin B në një nga **N** ditët e ardhshme (të numëruara nga 1 deri në N). Ajo mund të zhvendoset brënda natës, ose para ditës 1, ose pas ditës N, ose në ndonjë nga netët midis tyre. I zoti do që ta zgjedhë natën e zhvendosjes në mënyrë të tillë që fitimi të jetë sa më i madh. Duke ditur fitimin që mund të nxirret në secilën nga ditët, në secilin qytet, sa është fitimi më i madh që mund të nxjerrë byrektorja në këto N ditë?

Referenca: https://www.codechef.com/LTIME71B/problems/FASTFOOD

Shembull

```
$ cat input.txt
3
3
2 3 2
10 3 4
4
7 5 3 4
```

```
2 3 1 3
2
10 1
1 10

$ python3 prog.py < input.txt
17
19
20
```

Jepet numri i ditëve N, dhe në dy rreshtat pasardhës jepen fitimet për secilën nga këto ditë në qytetet A dhe B.

Në rastin e parë fitimi më i madh merret nqs lëvizja bëhet para ditës së parë.

Në rastin e dytë fitimi më i madh merret nqs lëvizja bëhet pas ditës së fundit.

Në rastin e tretë fitimi më i madh merret nqs lëvizja bëhet pas ditës së parë.

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    A = [int(i) for i in input().split()]
    B = [int(i) for i in input().split()]
    max_p = sum(B)
    for i in range(n+1):
        p = sum(A[:i]) + sum(B[i:])
        if p > max_p:
            max_p = p
    print(max_p)
```

Koha e rendit $O(N^2)$

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    A = [int(i) for i in input().split()]
    B = [int(i) for i in input().split()]
    p = sum(B)
```

```
    max_p = p
    for i in range(n):
        p += A[i]
        p -= B[i]
        if p > max_p:
            max_p = p
    print(max_p)
```

Koha e rendit $O(N)$

Problemi 22

Kërkesa

Që ta zërë gjumi shpejt, delja Bleatrix merr një numër **N**, pastaj dyfishin 2N, pastaj trefishin 3N, e kështu me rradhë, deri sa ti ketë parë të paktën një herë të gjitha shifrat: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

P.sh. nqs merr **N** = 1692 ka parë shifrat (1, 6, 9, 2). Pastaj merr numrin 2N = 3384 dhe ka parë edhe shifrat (3, 8, 4). Pastaj merr numrin 3N = 5076 dhe tani i ka parë të gjitha shifrat dhe e zë gjumi.

Cili ka qenë numri i fundit që ka parë Bleatrix para se ta zinte gjumi? Shkruani INSOMNIA nqs asnjëherë nuk i shikon të gjitha shifrat.

Referenca: https://code.google.com/codejam/contest/6254486/dashboard#s=p0&a=0

Shembull

```
$ cat input.txt
5
0
1
2
11
1692

$ python3 prog.py < input.txt
Case #1: INSOMNIA
Case #2: 10
Case #3: 90
Case #4: 110
Case #5: 5076
```

Në rastin e parë, shumëfishat e zeros janë gjithmonë 0, kështu që asnjëherë nuk arrin ti shikojë të gjithë numrat.

Në rastin e dytë, i shikon me rradhë shifrat 1, 2, 3, 4, 5, 6, 7, 8, 9, dhe kur arrin the 10 shikon edhe 0.

Në rastin e tretë, kur arrin te 90 shikon edhe 9-ën, që është shifra e fundit.

Në rastin e katërt shikon numrat 11, 22, ..., 99, 110.

Rasti i pestë është diskutuar më sipër.

Zgjidhja

```
for t in range(int(input())):
    N = int(input())
    if N == 0:
        print('Case #{}: INSOMNIA'.format(t+1))
        continue
    n = 0
    digits = set([])
    while len(digits) < 10:
        n += N
        digits |= set(str(n))
    print('Case #{}: {}'.format(t+1, n))
```

13 Shtojca 2

Problemi 101

Kërkesa

Jepet një numër natyror. Gjeni shumën e shifrës së parë dhe të fundit të tij.

Referenca: https://www.codechef.com/problems/FLOW004

Shembull

```
$ cat input.txt
3
1234
124894
242323

$ python3 prog.py < input.txt
5
5
5
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n = input()
    d1 = int(n[0])
    d2 = int(n) % 10
    print(d1 + d2)
```

Zgjidhja 2

```
T = int(input())
for t in range(T):
    n = input()
    s1 = n[0]
    s2 = n[len(n) - 1]
    print(int(s1) + int(s2))
```

Zgjidhja 3

```
T = int(input())
for t in range(T):
    n = input()
    print(int(n[0]) + int(n[-1]))
```

Sqarime

Numrin e lexojmë si string në `n`. Shifra e parë është `n[0]` kurse shifra e fundit është `n[-1]`. Por këto janë si *varg* (*string*). Që të mund ti mbledhim duhet ti kthejmë më parë në numra të plotë (*integer*).

Problemi 102

Kërkesa

Një pikë shitje bën zbritje 10% nëse sasia e mallit të blerë është mbi 1000 copë. Bëni një program që llogarit vlerën e blerjes, duke ditur sasinë dhe çmimin.

Referenca: https://www.codechef.com/problems/FLOW009

Shembull

```
$ cat input.txt
3
100 120
10 20
1200 20

$ python3 prog.py < input.txt
12000.000000
```

```
200.000000
21600.000000
```

Zgjidhja

```
T = int(input())
for t in range(T):
    sasia, cmimi = map(int, input().split())
    vlera = sasia * cmimi
    if sasia > 1000:
        vlera -= vlera / 10
    print(vlera)
```

Problemi 103

Kërkesa

Një olimpiadë programimi është zhvilluar në një qytet në vitet 2010, 2015, 2016, 2017, 2019. Bëni një program që merr disa vite dhe për secilin prej tyre nxjerr `HOSTED` nëse olimpiada është zhvilluar atë vit në qytet, ose `NOT HOSTED` nëse nuk është zhvilluar.

Referenca: https://www.codechef.com/problems/SNCKYEAR

Shembull

```
$ cat input.txt
2
2019
2018

$ python3 prog.py < input.txt
HOSTED
NOT HOSTED
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    y = int(input())
```

```
    if y == 2010 or y == 2015 or y == 2016 or y == 2017 or y == 2019:
        print ('HOSTED')
    else:
        print ('NOT HOSTED')
```

Zgjidhja 2

```
T = int(input())
for t in range(T):
    y = int(input())
    if y in [2010, 2015, 2016, 2017, 2019]:
        print ('HOSTED')
    else:
        print ('NOT HOSTED')
```

Problemi 104

Kërkesa

Në një restorant ka filluar punë një kamarier i ri. Meqënëse nuk është dhe aq i mirë në matematikë, ndonjëherë e kthen reston gabim. Shefi i jep atij një problem të thjeshtë: Sa bëjnë $a - b$? Përgjigja e tij është gabim. Dhe për çudi ka vetëm një shifër gabim. E merrni dot me mënd?

A mund të shkruani një program që bën të njëjtin gabim? Programi merr dy numra `a` dhe `b` dhe duhet të japë një diference të gabuar `a-b`. Përgjigja duhet të jetë një numër pozitiv që ka të njëjtin numër shifrash me diferencën e saktë, ku të gjitha shifrat janë njësoj me diferencën e saktë, me përjashtim të njërës që duhet të jetë e ndryshme. Zeroja nuk duhet të jetë si shifër e parë (nga e majta). Nëse ka përgjigje të ndryshme që i plotësojnë këto kushte, mjafton të jepni vetëm njërën.

Referenca: https://www.codechef.com/problems/CIELAB

Shembull

```
$ cat input.txt
3
5858 1234
239 230
4375 4375
```

```
$ python3 prog.py < input.txt
1624
8
1
```

Në rastin e parë, diferenca e saktë është `5858 - 1234 = 4624`, kështu që këto përgjigje do ishin të sakta: `2624`, `4324`, `4623`, `4604`, `4629`, kurse këto do ishin të gabuara: `0624`, `624`, `5858`, `4624`, `04624`.

Zgjidhja

```
T = int(input())
for t in range(T):
    a, b = map(int, input().split())
    diff = a - b
    print(diff-1 if diff % 10 == 9 else diff+1)
```

Sqarime

Nëse shifra e fundit e diferencës është `9` mjafton që ti zbresim diferencës `1`, përndryshe mjaton që ti shtojmë `1`, dhe në çdo rast do jemi të garantuar që vetëm shifra e fundit do jetë e ndryshme nga ajo e diferencës.

Problemi 105

Kërkesa

Kur pohojmë zakonisht e tundim kokën nga lart poshtë. Le ta shënojmë me `Y`, për "yes". Kur mohojmë e tundim kokën nga e majta në të djathtë, rreth boshtit vertikal. Le ta shënojmë me `N` për "no".

Por indianët zakonisht përdorin një shenjë tjetër për të pohuar, duke e tundur kokën nga e majta në të djathtë rreth boshtit para-mbrapa. Le ta shënojmë këtë me `I`.

Një djalë vëzhgoi disa persona në një stacion treni dhe mbajti shënim shënjat që bënin me kokë (duke përdorur shkronjat `Y`, `N` dhe `I`). Bëni një program që shfaq nëse personi në fjalë është indian, jo indian, ose nuk mund ta themi me siguri.

Referenca: https://www.codechef.com/problems/HEADBOB

Shembull

```
$ cat input.txt
3
5
NNNYY
6
NNINNI
4
NNNN

$ python3 prog.py < input.txt
NOT INDIAN
INDIAN
NOT SURE
```

Zgjidhja

```python
T = int(input())
for t in range(T):
    n = int(input())
    str = input()
    if 'I' in str:
        print('INDIAN')
    elif 'Y' in str:
        print('NOT INDIAN')
    else:
        print('NOT SURE')
```

Sqarime

Operatori `in` në këtë rast teston nëse një shkronjë ndodhet brenda një vargu. P.sh:

```
$ python3
>>> 'b' in 'abc'
True
>>> 'x' in 'abc'
False
```

Problemi 106

Kërkesa

Bëni një program që gjen rrënjën katrore natyrore (të përafërt) të një numri natyror të dhënë.

Referenca: https://www.codechef.com/problems/FSQRT

Shembull

```
$ cat input.txt
3
10
5
10000

$ python3 prog.py < input.txt
3
2
100
```

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    i = 1
    while i*i <= n:
        i += 1
    print(i)
```

Problemi 107

Kërkesa

Një qenush gjeti një arkë me N monedha. Ai nuk e hap dot vetë, por mund të lehë që tu tërheq vëmendjen njerëzve që ndodhen aty pranë. Ai e di që njerëzit do marrin secili një sasi të barabartë monedhash, dhe të tjerat do ti lënë në shesh, kështu që ai mund ti marrë.

Qenushi mund ta ndryshojë fortësinë e të lehurit, në mënyrë që ti tërheqë vëmendjen 1 personi, ose 2 personave, e kështu me rradhë, deri në K persona. Sa persona duhet të thërrasë, në mënyrë që në fund ti ngelen sa më shumë monedha?

Referenca: https://www.codechef.com/problems/GDOG

Shembull

```
$ cat input.txt
2
5 2
11 3

$ python3 prog.py < input.txt
1
2
```

Zgjidhja

```
T = int(input())
for t in range(T):
    N, K = map(int, input().split())
    m = 0
    for k in range(1, K+1):
        if N % k > m:
            m = N % k
    print(m)
```

Problemi 108

Kërkesa

Çufoja ka bërë N copa keku të vogla dhe tani po mendon si ti paketojë. Çdo paketë duhet të ketë numër të njëjtë kekësh. Çufoja duhet të zgjedhë një numër të plotë A, nga 1 në N, dhe duhet të vendosë fiks A copa keku në çdo paketë. Pasi ka plotësuar aq paketa sa të jetë e mundur, copat e kekut që kanë mbetur pa paketuar Çufoja mund ti hajë vetë. Çufos i pëlqejnë shumë këta dreq kekë. Sa duhet ta zgjedhë numrin A të paketimit që ti mbeten sa më shumë copa keku për të ngrënë?

Bëni një program që merr numrin N dhe gjen numrin A, ku $2 \leq N \leq 100000000(10^8)$ Ky program duhet ta kthejë përgjigjen në më pak se 1 sek.

Referenca: https://www.codechef.com/problems/MUFFINS3

Shembull

```
$ cat input.txt
2
2
5

$ python3 prog.py < input.txt
2
3
```

Zgjidhja 1

```python
T = int(input())
for t in range(T):
    n = int(input())
    m = 0   # mbetja
    p = 0   # madhesia e paketes
    for i in range(1, n+1):
        if n % i >= m:
            m = n % i
            p = i
    print(p)
```

Zgjidhja 2

```python
for _ in range(int(input())):
    n = int(input())
    print(n // 2 + 1)
```

Sqarime

Zgjidhja e parë është e ngadaltë kur N-ja është shumë e madhe, p.sh. kur `N=100000000` (provojeni!). Kurse zgjidhja e dytë është jashtëzakonisht e shpejtë, sepse nuk i provon të gjitha rastet por e gjen përgjigjen me formulë.

Problemi 109

Kërkesa

Mamaja i ka blerë Cufit dy shporta me fruta, njëra ka N mollë dhe tjera ka M portokalle. Cufit i pëlqejnë edhe mollët edhe portokallet dhe do që ti ketë në sasi të barabartë. Çdo kokërr mollë ose portokalle kushton 1 monedhë, dhe Cufi ka K monedha në xhep. Me këto monedha ai mund të blejë mollë ose portokalle dhe të përpiqet ta bëjë ndryshimin midis tyre sa më të vogël.

Bëni një program që të gjejë ndryshimin më të vogël të mundshëm që mund të arrijë Cufi midis mollëve dhe portokalleve.

Referenca: https://www.codechef.com/problems/FRUITS

Shembull

```
$ cat input.txt
3
3 4 1
5 2 1
3 4 3

$ python3 prog.py < input.txt
0
2
0
```

Jepen numrat N, M dhe K. Në rastin e parë Cufi mund të blejë një mollë dhe ti bëjë të barabarta. Në rastin e dytë mund të blejë vetëm një portokalle dhe diferenca bëhet 2. Në rasting e tretë mund të blejë 2 mollë dhe një portokalle, duke e bërë diferencën 0.

Zgjidhja

```python
T = int(input())
for t in range(T):
    n, m, k = map(int, input().split())
    d = abs(n - m)
    print(0) if k > d else print(d - k)
```

Problemi 110

Kërkesa

Një fermer do të shesë tokën e tij, e cila është në formë drejtkëndëshi me përmasa M dhe N. Meqënëse copat e tokës në formë katrore kanë çmim më të lartë, ai do që ta ndajë tokën e tij në copa katrore të barabarta. Sa është numri më i vogël i katrorëve të barabartë që mund të krijohen në tokën e tij?

Referenca: https://www.codechef.com/problems/RECTSQ

Shembull

```
$ cat input.txt
2
10 15
4 6

$ python3 prog.py < input.txt
6
6
```

Zgjidhja

```
import math
for _ in range(int(input())):
    m, n = map(int, input().split())
    d = math.gcd(m, n)
    print((m // d) * (n // d))
```

Sqarime

Duhet të gjemë pjesëtuesin më të madh të përbashkët të dy brinjëve të drejtkëndëshit, dhe pastaj ta ndajmë secilën brinjë sipas tij.

Problemi 111

Kërkesa

Një drejtkëndësh me përmasa A dhe B duam ta ndajmë në copa katrore (jo detyrimisht të barabarta). Sa është numri më i vogël i katrorëve në të cilët mund të ndahet ky

drejtkëndësh?

Shembull

```
$ cat input.txt
2
10 15
4 7

$ python3 prog.py < input.txt
3
5
```

Zgjidhja

```
for _ in range(int(input())):
    a, b = map(int, input().split())
    nr = 0
    while b > 0:
        nr += a // b
        a, b = b, a % b
    print(nr)
```

Problemi 112

Kërkesa

Sa katrorë me madhësi **2x2** mund të futen në një trekëndësh këndrejt dybrinjëshëm, ku kateti është me gjatësi **B**, dhe brinjët e katrorëve janë paralel me katetet?

Referenca: https://www.codechef.com/problems/TRISQ

Shembull

```
$ cat input.txt
11
1
2
3
4
```

```
5
6
7
8
9
10
11

$ python3 prog.py < input.txt
0
0
0
1
1
3
3
6
6
10
10
```

Zgjidhja

```
for _ in range(int(input())):
    b = int(input())
    k = b // 2
    nr = k * (k - 1) // 2
    print(nr)
```

Problemi 113

Kërkesa

Vogëlushja Meli i ka qejf balonat me ngjyra dhe mami i ka blerë për ditëlindje disa balona me ngjyra të kuqe dhe blu. Mirëpo Meli do të dëshironte ti kishte të gjitha balonat me të njëjtën ngjyrë, kështu që mori kolorat dhe ju fut punës për ti ngjyrosur.

Bëni një program që gjen sa është numri më i vogël i balonave që duhen ngjyrosur, për ti bërë të gjitha me të njëjtën ngjyrë.

Referenca: https://www.codechef.com/problems/CHN09

Shembull

```
$ cat input.txt
3
ab
bb
baaba

$ python3 prog.py < input.txt
1
0
2
```

Balonat e kuqe i kemi shënuar me **a**, blutë me **b**. Në rastin e parë mjafton të ngjyrosim një balonë të kuqe me blu, ose një blu me të kuqe. Në rastin e dytë të gjitha balonat kanë të njëjtën ngjyrë. Kurse në rastin e tretë mjafton të ngjyrosim dy balonat blu me të kuqe.

Zgjidhja 1

```python
for _ in range(int(input())):
    str = input()
    nr_a = 0
    nr_b = 0
    for c in str:
        if c == 'a':
            nr_a += 1
        else:
            nr_b += 1
    print(min(nr_a, nr_b))
```

Sqarime

Gjemë numrin e balonave të kuqe dhe blu, dhe pastaj printojmë minimumin e tyre.

Zgjidhja 2

```python
for _ in range(int(input())):
    str = input()
```

```
    nr_a = 0
    for c in str:
        if c == 'a':
            nr_a += 1
    nr_b = len(s) - nr_a
    print(min(nr_a, nr_b))
```

Zgjidhja 3

```
for _ in range(int(input())):
    str = input()
    print(min(str.count('a'), str.count('b')))
```

Problemi 114

Kërkesa

Vogëlushja Meli i ka qejf balonat me ngjyra dhe mami i ka blerë për ditëlindje disa balona me ngjyra të ndryshme. Mirëpo Meli do të dëshironte ti kishte të gjitha balonat me të njëjtën ngjyrë, kështu që mori kolorat dhe ju fut punës për ti ngjyrosur.

Bëni një program që gjen sa është numri më i vogël i balonave që duhen ngjyrosur, për ti bërë të gjitha me të njëjtën ngjyrë.

Shembull

```
$ cat input.txt
3
abc
bb
baabacd

$ python3 prog.py < input.txt
2
0
4
```

Çdo ngjyrë shënohet me një shkronjë të veçantë. Në rastin e parë mjafton të ngjyrosim balonat **b** dhe **c** me ngjyrën **a**. Në rastin e dytë të gjitha balonat kanë të njëjtën ngjyrë. Kurse në rastin e tretë balonat me ngjyrat **b**, **c** dhe **d** duhet ti ngjyrosim me ngjyrën **a**.

Zgjidhja 1

```
for _ in range(int(input())):
    str = input()
    nr = {}    # fjalor per shkronjat dhe numrin e seciles shkronje
    for c in str:
        nr[c] = nr.get(c, 0) + 1
    print(len(str) - max(nr.values()))
```

Sqarime

Gjemë numrin e balonave të secilës ngjyrë, dhe pastaj nga numri i gjithë balonave heqim numrin e balonave që janë më shumë.

Zgjidhja 2

```
for _ in range(int(input())):
    str = input()
    print(len(str) - max([str.count(c) for c in set(str)]))
```

Problemi 115

Kërkesa

Colit i pëlqen të luaj me numrat dhe sot po bën një lojë të tillë:

Në një listë me numra A zgjedh dy numra të njëpasnjëshëm. Më të madhin prej tyre e heq nga lista, dhe kështu gjatësia e listës zvogëlohet me 1. Kostoja e këtij veprimi është sa më i vogli prej tyre. Këtë veprim e përsërit deri sa në listë të ketë mbetur vetëm 1 numër, dhe gjen shumën e kostove për të gjitha veprimet.

Sa është vlera më e vogël e kësaj shume?

Referenca: https://www.codechef.com/problems/MNMX

Shembull

```
$ cat input.txt
2
2
```

```
3 4
3
4 2 5

$ python3 prog.py < input.txt
3
4
```

Në rastin e parë mund të kryejmë vetëm një veprim. Ky veprim fshin numrin 4 nga lista dhe e ka koston 3 (që është edhe kostoja e përgjithshme).

Në rastin e dytë mund të zgjdhim në fillim dyshen `4 2`, ku fshihet numri `4` dhe kostoja e veprimit është **2**, pastaj mund të zgjedhim dyshen `2 5`, ku fshihet numri `5` dhe kostoja e veprimit është **2**. Kostoja e përgjithshme është **4**.

Zgjidhja 1

```python
for _ in range(int(input())):
    n = int(input())
    l = map(int, input().split())

    # gjejmë numrin më të vogël të listës
    m = l[0]
    for a in l:
        if a < m:
            m = a

    print(m*(n - 1))
```

Sqarime

Meqënëse mund të zgjedhim një çift numrash çfarëdo, ne na intereson që gjithmonë të zgjedhim numrin më të vogël të listës dhe një numër tjetër pranë tij. Në këtë rast numri tjetër do hiqet nga lista dhe kostoja e veprimit do jetë sa numri më i vogël. Duke zgjedhur gjithmonë numrin më të vogël ne minimizojmë koston totale.

Meqënëse gkithsej kryhen `(n - 1)` veprime të tilla, kostoja totale është sa prodhimi i numrit më të vogël me `(n - 1)`.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    l = map(int, input().split())
    print(min(l)*(n - 1))
```

Problemi 116

Kërkesa

Në një listë me këngë, secila prej këngëve e ka gjatësinë një numër të plotë minutash, dhe secila këngë e ka gjatësinë të ndryshme nga të tjerat. Kënga juaj e preferuar ndodhet në pozicionin **K**. Më pas ju i rendisni këngët sipas gjatësisë së tyre. Në cilin pozicion ndodhet tani kënga juaj e preferuar?

Referenca: https://www.codechef.com/problems/JOHNY

Shembull

```
$ cat input.txt
3
4
1 3 4 2
2
5
1 2 3 9 4
5
5
1 2 3 9 4
1

$ python3 prog.py < input.txt
3
4
1
```

1. Në rastin e parë kemi: `N = 4`, `L = [1, 3, 4, 2]`, `K = 2`. Kënga e preferuar është **3**. Pas renditjes kemi `L = [1, 2, 3, 4]`, kështu që përgjigja është **3**.
2. Në rastin e dytë: `N = 5`, `L = [1, 2, 3, 9, 4]`, `K = 5`. Kënga e preferuar është **4**. Pas renditjes do jetë në pozicionin **4**.

3. Në rastin e tretë: `N = 5`, `L = [1, 2, 3, 9, 4]`, `K = 1`. Kënga e preferuar është **1**. Pas renditjes do jetë prapë në pozicionin **1**.

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    L = list(map(int, input().split()))
    k = int(input())
    p = L[k - 1]    # kenga e preferuar
    L.sort()
    for i in range(n):
        if L[i] == p:
            break
    print(i + 1)
```

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    L = [0] + list(map(int, input().split()))
    k = int(input())
    p = L[k]    # kenga e preferuar
    L.sort()
    for i in range(n):
        if L[i] == p:
            break
    print(i)
```

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    L = [0] + list(map(int, input().split()))
    k = int(input())
    p = L[k]    # kenga e preferuar
    L.sort()
    print(L.index(p))
```

Problemi 117

Kërkesa

Në një lojë kemi **N** monedha të numëruara nga **1** në **N**, dhe fillimisht të gjitha monedhat janë nga e njëjta anë (**H** ose **T**). Loja luhet me **N** raunde, dhe në çdo round **k**, lojtari kthen **k** monedhat e para (nga **1** deri në **k**) në anën tjetër. Duam të gjejmë se sa monedha do jenë në një anë të caktuar (**H** ose **T**) në fund të lojës.

Referenca: https://www.codechef.com/problems/CONFLIP

Shembull

```
$ cat input.txt
2
1 5 1
1 5 2

$ python3 prog.py < input.txt
2
3
```

Kemi 2 raste testimi. Në rastin e parë kemi `I=1`, `N=5`, `Q=1`, që do të thotë se fillimisht monedhat janë të gjitha në anën **H**, kemi **5** monedha dhe **5** raunde, dhe në fund të raundit të 5-të duam të dimë se sa monedha janë në anën **H**.

Në rastin e dytë është e njëjta gjë, vetëm se duam të dimë se sa monedha janë në anën **T** (`Q=2`).

Gjendja e monedhave pas çdo raundi do jetë e tillë: 1. **T** H H H H 2. **H T** H H H 3. **T H T** H H 4. **H T H T** H 5. **T H T H T**

Kështu që përgjigja për rastin e parë është **2**, kurse për rastin e dytë është **3**.

Zgjidhja 1

```
for _ in range(int(input())):
    I, N, Q = map(int, input().split())
    L = ['H']*N if I==1 else ['T']*N
    for r in range(N):    # for each round
        for i in range(r+1):
            # flip coin
            if L[i] == 'H':
```

```
                L[i] = 'T'
            else:
                L[i] = 'H'
        #print(L)    # debug
    nr = 0
    for i in range(N):
        if (Q == 1 and L[i] == 'H') or (Q == 2 and L[i] == 'T'):
            nr += 1
    print(nr)
```

Zgjidhja 2

```
for _ in range(int(input())):
    I, N, Q = map(int, input().split())
    nr = N // 2
    if N % 2 != 0 and I != Q:
        nr += 1
    print(nr)
```

Sqarime

Zgjidhja e parë është e rendit $O(N^2)$, që do të thotë se duhen kryher përafërsisht N^2 veprime. Kur N bëhet e madhe, N^2 bëhet shumë shpejt jashtëzakonisht e madhe.

Kurse zgjidhja e e dytë është e rendit $O(C)$, që do të thotë se nuk ka rëndësi se sa e madhe është N-ja, ne mund ta gjejmë gjithmonë përgjigjen me një numër konstant veprimesh.

Problemi 118

Kërkesa

Gjuhët e harruara janë gjuhë që janë përdorur dikur gjerësisht por sot nuk përdoren më. Megjithatë disa prej fjalëve të tyre mund të jenë ende në përdorim në gjuhët e sotme.

You keni gjetur në internet N fjalë të një gjuhe të harruar. Gjithashtu keni edhe K fjali që përdoren në gjuhët e sotme. Detyra juaj është që të përcaktoni për secilën prej N fjalëve nëse gjendet në ndonjërën nga këto K fjalitë ose jo.

Referenca: https://www.codechef.com/problems/FRGTNLNG

Shembull

```
$ cat input.txt
2
3 2
piygu ezyfo rzotm
1 piygu
6 tefwz tefwz piygu ezyfo tefwz piygu
4 1
kssdy tjzhy ljzym kegqz
4 kegqz kegqz kegqz vxvyj

$ python3 prog.py < input.txt
YES YES NO
NO NO NO YES
```

Kemi 2 raste testimi. Në rastin e parë, kemi `N=3` fjalë të gjuhës së harruar dhe `K=2` fjali të gjuhëve të sotme. Pastaj vijnë 2 fjalitë, ku e para ka 1 fjalë dhe e dyta ka 6 fjalë. Dy fjalët e para ndodhen në këtë fjali, kurse e treta jo.

Zgjidhja

```python
for _ in range(int(input())):
    n, k = map(int, input().split())
    fjalet_e_vjetra = input().split()
    fjalet_e_reja = []
    for i in range(k):
        fjalet_e_reja += input().split()[1:]
    for fjale in fjalet_e_vjetra:
        if fjale in fjalet_e_reja:
            print('YES', end=' ')
        else:
            print('NO', end=' ')
    print()
```

Problemi 119

Kërkesa

Në kohët e lashta, komandanti i një ushtrie ishte superstticioz. Ai e quante një ushtar "me fat" nëse numri i armëve që kishte ushtari ishte çift dhe "pa fat" në të kundërt.

Ushtrinë e quante “gati për betejë” nëse numri i ushtarëve me fat ishte më i madh se ai i ushtarëve pa fat. Në të kundërt e quante “jo gati”.

Nëse ju jepet numri i armëve që ka çdo ushtar, bëni një program që gjen nëse ushtria është gati për betejë ose jo.

Referenca: https://www.codechef.com/problems/AMR15A

Shembull

```
$ cat input.txt
5
1
1
1
2
4
11 12 13 14
3
2 3 4
5
1 2 3 4 5

$ python3 prog.py < input.txt
NOT READY
READY FOR BATTLE
NOT READY
READY FOR BATTLE
NOT READY
```

Zgjidhja

```
T = int(input())
for t in range(T):
    n = int(input())
    L = list(map(int, input().split()))
    odd = 0
    even = 0
    for a in L:
    if a % 2 == 0:
       even += 1
    else:
        odd +=1
```

```
    if even > odd:
        print('READY FOR BATTLE')
    else:
        print('NOT READY')
```

Problemi 120

Kërkesa

Roboti Bani ka humbur rrugën. Ai ndodhet në kordinatat (x_1, y_1) në një plan 2 dimensional, kurse shtëpia e tij ndodhet në kordinatat (x_2, y_2). Bëni një program që ta ndihmojë duke i treguar drejtimin në të cilin duhet të lëvizë për të arritur në shtëpi. Nëse i tregojmë një drejtim, roboti do të vazhdojë të lëvizë në atë drejtim deri sa të arrijë në shtëpi. Janë katër drejtime që mund tia japim si komandë: **majtas**, **djathtas**, **lartë**, **poshtë**. Nëse asnjë nga këto drejtime nuk e çon robotin në shtëpi, programi duhet të nxjerrë: **më vjen keq**.

Referenca: https://www.codechef.com/problems/ICPC16A

Shembull

```
$ cat input.txt
5
0 0 1 0
0 0 0 1
0 0 1 1
2 0 0 0
0 2 0 0

$ python3 prog.py < input.txt
right
up
sad
left
down
```

Zgjidhja

```
for _ in range(int(input())):
    x1, y1, x2, y2 = map(int, input().split())
```

```
    if x1 == x2:
        if y1 < y2:
            print('up')
        else:
            print('down')
    elif y1 == y2:
        if x1 < x2:
            print('right')
        else:
            print('left')
    else:
        print('sad')
```

Problemi 121

Kërkesa

Na jepet një numër binar (i cili ka vetëm shifrat **0** dhe **1**). Bëni një program i cili gjen nëse është e mundur që të bëhet i barabartë numri i zerove dhe njëshave, duke ndryshuar të shumtën një shifër (nga 1 në 0 ose nga 0 në 1).

Shembull

```
$ cat input.txt
4
101
11
0
10000

$ python3 prog.py < input.txt
No
Yes
No
No
```

Në rastin e parë nuk është e mundur ti bëjmë të barabarta numrat e zerove dhe njëshave. Në rastin e dytë mund ta bëjmë një njësh zero dhe do kemi numër të barabartë zerosh dhe njëshash. Në rastin e tretë nuk mund të kemi një numër të barabartë zerosh dhe njëshash. Në rastin e katërt gjithashtu nuk mund të bëjmë një numër të barabartë zerosh dhe njëshash.

Zgjidhja 1

```
T = int(input())
for t in range(T):
    N = input()
    cnt0 = 0
    cnt1 = 0
    for d in N:
        if d == '0':
            cnt0 += 1
        else:
            cnt1 += 1
    if cnt0 == cnt1 or abs(cnt0 - cnt1) == 2:
        print('Yes')
    else:
        print('No')
```

Sqarime

Gjejmë numrin e zerove dhe të njëshave. Nëse janë të barabartë nuk është nevoja të ndryshojmë gjë. Nëse diferenca midis tyre është 2, me një ndryshim mund ti bëjmë të barabarta.

Zgjidhja 2

```
for _ in range(int(input())):
    N = input()
    cnt0 = N.count('0')
    cnt1 = N.count('1')
    if cnt0 == cnt1 or abs(cnt0 - cnt1) == 2:
        print('Yes')
    else:
        print('No')
```

Zgjidhja 3

```
for _ in range(int(input())):
    N = input()
```

```
    cnt1 = sum(map(int, list(N)))
    cnt0 = len(N) - cnt1
    if cnt0 == cnt1 or abs(cnt0 - cnt1) == 2:
        print('Yes')
    else:
        print('No')
```

Problemi 122

Kërkesa

Pas një kohe të gjatë, Cufi vendosi më në fund ta rinovojë shtëpinë. Shtëpia e Cufit ka **N** dhoma. Secila prej dhomave është e lyer me një nga ngjyrat e kuqe, blu dhe jeshile. Konfigurimi i ngjyrave të shtëpisë na jepet me anë të një vargu me shkronjat **R**, **B**, **G**. Cufi do që ta lyejë shtëpinë në mënyrë që të gjitha dhomat të kenë të njëjtën ngjyrë, por ngaqë është pak dembel, do që të lyejë sa më pak dhoma (mundësisht asnjë). Bëni një program që gjen numrin më të vogël të dhomave që duhet të lyejë Cufi.

Referenca: https://www.codechef.com/problems/COLOR

Shembull

```
$ cat input.txt
3
3
RGR
3
RRR
3
RGB

$ python3 prog.py < input.txt
1
0
2
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
```

```
    n = int(input())
    S = input()
    r = 0
    g = 0
    b = 0
    for c in S:
        if c == 'R':
            r += 1
        elif c == 'G':
            g += 1
        else:
            b += 1
    print(n - max(r, g, b))
```

Sqarime

Gjejmë sa dhoma janë me secilën ngjyrë. Numri më i vogël i lyerjeve bëhet nqs të gjitha dhomat i kthejmë në ngjyrën që kanë shumica e dhomave.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    S = input()
    r = S.count('R')
    g = S.count('G')
    b = S.count('B')
    print(n - max(r, g, b))
```

Problemi 123

Kërkesa

Dejvi ka disa shokë të çuditshëm. Me rastin e ditëlindjes së tij secili prej tyre i ka kërkuar që ta qerasë në një datë të caktuar, me kushtin që do ta prishë shoqërinë në rast se nuk e qeras pikërisht në atë datë. Mirëpo Dejvi nuk mund të qerasë më shumë se një prej tyre në një datë të caktuar. Gjeni se me sa prej tyre mund ta ruajë shoqërinë.

Referenca: https://www.codechef.com/problems/CFRTEST

Shembull

```
$ cat input.txt
2
2
3 2
2
1 1

$ python3 prog.py < input.txt
2
1
```

Në rastin e parë, të dy shokët e kërkojnë qerasjen në data të ndryshme, kështu që mund ta ruajë shoqërinë me të dy. Në rastin e dytë, të dy e kërkojnë qerasjen në të njëjtën ditë, kështu që mund ta ruajë shoqërinë vetëm me njërin prej tyre.

Zgjidhja

```
for _ in range(int(input())):
    input()
    print(len(set(map(int, input().split()))))
```

Sqarime

Nqs krijojmë një bashkësi me të dhënat (`set()`), atere secila prej tyre do ruhet vetëm një herë, kështu që mjafton të gjejmë numrin e tyre (me `len()`).

Problemi 124

Kërkesa

Loli, vëllai i vogël i Colit, sapo ka mësuar disa prej shkronjave të alfabetit, dhe mund të lexojë vetëm ato fjalë që përbëhen prej shkronjave që njeh. Coli i jep atij një libër për të lexuar dhe do të dijë se cilat prej fjalëve mund të lexojë Loli.

Referenca: https://www.codechef.com/problems/ALPHABET

Shembull

```
$ cat input.txt
act
2
cat
dog

$ python3 prog.py < input.txt
Yes
No
```

Rreshi i parë përmban shkronjat që ka mësuar Loli. Fjala e parë i ka të gjitha shkronjat të njohura, fjala e dytë jo.

Zgjidhja

```
alphabet = input()
N = int(input())
for i in range(N):
    word = input()
    for c in word:
        if c not in alphabet:
            print('No')
            break
    else:
        print('Yes')
```

Problemi 125

Kërkesa

Na jepen dy vargje shkronjash **A** dhe **B**. A është e mundur të gjejmë një nënvarg **s1** në **A** dhe një nënvarg **s2** në **B**, që mos të jenë bosh, dhe që **s1**+**s2** të jetë palindromë?

Shenja + tregon bashkimin ose ngjitjen e dy vargjeve, kurse palindromë do të thotë që po ta lexosh vargun nga fundi në fillim është njësoj me vargun fillestar (nga fillimi në fund).

Referenca: https://www.codechef.com/problems/STRPALIN

Shembull

```
$ cat input.txt
3
abc
abc
a
b
abba
baab

$ python3 prog.py < input.txt
Yes
No
Yes
```

Në rastin e parë, një nga zgjedhjet e mundshme është `s1 = "ab"` dhe `s2 = "a"`. Atere kemi `s1+s2 = "aba"` e cila është palindromë, kështu që përgjigja është "Yes".

Në rastin e dytë kemi `A = "a"` dhe `B = "b"` dhe nuk ka asnjë mundësi që të krijojmë ndonjë palindromë, kështu që përgjigja është "No".

Zgjidhja

```
for _ in range(int(input())):
    if len(set(input()).intersection(set(input()))) == 0:
        print('No')
    else:
        print('Yes')
```

Sqarime

Meqënëse nënvargjet **s1** dhe **s2** nuk mund të jenë bosh dhe shkronja e parë e **s1+s2** duhet të jetë e njëjtë me shkronjën e fundit (nga përkufizimi i palindromës), i bie që vargjet **A** dhe **B** duhet të kenë të paktën një shkronjë të përbashkët, përndryshe nuk mund të formojmë një palindromë. Nga ana tjetër, nqs **A** dhe **B** kanë një shkronjë të përbashkët, p.sh. **x**, atere ne mund të zgjedhim `s1 = "x"` dhe `s2 = "x"` dhe `s1+s2 = "xx"` do jetë një palindromë.

Kështu që kusht i nevojshëm dhe i mjaftueshëm që problemi të ketë zgjidhje është që **A** dhe **B** të kenë të paktën një shkronjë të përbashkët. Edhe programi i zgjidhjes pikërisht për këtë kontrollon: i kthen vargjet **A** dhe **B** në bashkësi shkronjash, gjen prerjen e tyre, dhe shikon nëse prerja është bosh ose jo.

Problemi 126

Kërkesa

Alisa dhe Beni po bëjnë një lojë me numra. Fillimisht Alisa ka një numër **A** kurse Beni një numër **B**. Loja ka gjithsej **N** hapa dhe Alisa me Benin luajnë me rradhë. Kur është rradha e tij, secili lojtar e shumëzon numrin e tij me dy. E para luan Alisa.

Bëni një program i cili merr numrat **A**, **B** dhe **N** dhe gjen sa është herësi i pjesëtimit të numrave në fund të lojës (duke pjesëtuar më të madhin me më të voglin).

Referenca: https://www.codechef.com/problems/TWONMS

Shembull

```
$ cat input.txt
3
1 2 1
3 2 3
3 7 2

$ python3 prog.py < input.txt
1
3
2
```

Në rastin e parë, numrat fillestarë janë (A=1, B=2) dhe loja ka vetëm 1 hap. Në këtë hap Alisa e shumëzon numrin e saj me 2, kështu që në fund të lojës kemi (A=2, B=2) dhe herësi është 1.

Në rastin e dytë kemi (A=3, B=2) dhe N=3. Pasi luan Alisa, Beni, Alisa, kemi (A=12, B=4), kështu që herësi është 3.

Në rastin e tretë kemi (A=3, B=7) dhe N=2. Pasi luan Alisa dhe Beni, kemi (A=6, B=14), kështu që herësi është 2.

Zgjidhja

```
for _ in range(int(input())):
    a, b, n = map(int, input().split())
    if n % 2 == 1:
        a *= 2
    print(max(a,b)//min(a,b))
```

Sqarime

Meqënëse dyshat thjeshtohen me njëri-tjetrin gjatë pjesëtimit, mjafton që të shumëzojmë a-në me 2 nëse n-ja është tek, dhe pastaj mund të gjejmë herësin.

Problemi 127

Kërkesa

Sergei ka bërë **N** matje dhe do të gjejë vlerën mesatare të tyre. Por për të gjetur një mesatare sa më të mirë, ai mendon se duhet ti heqë nga lista e matjeve **K** vlerat më të vogla dhe **K** vlerat më të mëdha, para se të llogarisë mesataren. Sa del vlera mesatare në këtë mënyrë?

Referenca: https://www.codechef.com/problems/SIMPSTAT

Shembull

```
$ cat input.txt
3
5 1
2 9 -10 25 1
5 0
2 9 -10 25 1
3 1
1 1 1

$ python3 prog.py < input.txt
4.000000
5.400000
1.000000
```

Zgjidhja 1

```
T = int(input())
for t in range(T):
    n, k = map(int, input().split())
    A = list(map(int, input().split()))
    A.sort()
    s = 0
    i = k
```

```
    while i < n - k:
        s += A[i]
        i += 1
    print(s / (n - 2*k))
```

Zgjidhja 2

```
T = int(input())
for t in range(T):
    n, k = map(int, input().split())
    A = list(map(int, input().split()))
    A.sort()
    s = 0
    for i in range(k, n - k):
        s += A[i]
    print(s / (n - 2*k))
```

Zgjidhja 3

```
for _ in range(int(input())):
    n, k = map(int, input().split())
    A = list(map(int, input().split()))
    A.sort()
    print(sum(A[k:n-k]) / (n - 2*k))
```

Problemi 128

Kërkesa

Shefi ka gjetur një libër me receta gatimesh, ku çdo gatim ka 4 përbërës. Ai do zgjedh 2 receta për ti gatuar për drekë, por dëshiron që këto dy gatime të mos jenë të ngjashme. Dy gatime quhen të ngjashme nëse kanë 2 ose më shumë përbërës të njëjtë.

Bëni një program që merr përbërësit e dy gatimeve dhe gjen nëse janë të ngjashme ose jo.

Referenca: https://www.codechef.com/problems/SIMDISH

Shembull

```
$ cat input.txt
5
eggs sugar flour salt
sugar eggs milk flour
aa ab ac ad
ac ad ae af
cookies sugar grass lemon
lemon meat chili wood
one two three four
one two three four
gibberish jibberish lalalalala popopopopo
jibberisz gibberisz popopopopu lalalalalu

$ python3 prog.py < input.txt
similar
similar
dissimilar
similar
dissimilar
```

Zgjidhja 1

```python
T = int(input())
for _ in range(T):
    R1 = input().split()
    R2 = input().split()
    n = 0
    for r in R1:
        if r in R2:
            n += 1
    if n >= 2:
        print('similar')
    else:
        print('dissimilar')
```

Zgjidhja 2

```
for _ in range(int(input())):
    R1 = set(input().split())
    R2 = set(input().split())
    n = len(R1.intersection(R2))
    print('similar') if n >= 2 else print('dissimilar')
```

Problemi 129

Kërkesa

Cufit i pëlqen të luajë pingpong. Ai gjeti disa shënime me pikët e lojrave që ka bërë para ca kohësh. Me **1** shënohet nëse ka fituar një pikë dhe me **0** nëse ka humbur një pikë (e ka fituar kundërshtari). Loja e pingpongut fitohet nga lojtari që bën i pari 11 pikë, me përjashtim të rastit kur të dy lojtarët kanë nga 10 pikë, në të cilin loja vazhdon deri sa njëri prej lojtarëve të dalë 2 pikë para kundërshtarit. Gjeni se cilat prej lojrave i ka fituar Cufi dhe cilat i ka humbur.

Referenca: https://www.codechef.com/problems/TTENIS

Shembull

```
$ cat input.txt
2
0101111111111
11100000000000

$ python3 prog.py < input.txt
WIN
LOSE
```

Zgjidhja 1

```
for _ in range(int(input())):
    piket = input()
    p0 = p1 = 0
    for p in piket:
        if p == '0':
            p0 += 1
        else:
            p1 += 1
```

```
    if p1 > p0:
        print('WIN')
    else:
        print('LOSE')
```

Zgjidhja 2

```
for _ in range(int(input())):
    piket = input()
    p0 = piket.count('0')
    p1 = piket.count('1')
    print('WIN') if p1 > p0 else print('LOSE')
```

Problemi 130

Kërkesa

Mësuesi ka ngritur disa nxënës në dërrasë dhe u ka dhënë nga një ushtrim për të zgjidhur (klasa ka një dërrasë të gjatë kështu që mësuesi mund të ngrejë në dërrasë shumë nxënës njëkohësisht). Sapo mësuesi doli një moment nga klasa, disa prej nxënësve u kthyen dhe filluan të komunikojnë me shokun që kishin në krah, kurse të tjerët vazhduan punën. Këtë situatë mund ta paraqesim me një varg të tillë: `*><><><*`, ku një `*` tregon një nxënës që vazhdon punën, një `>` tregon një nxënës që po flet me shokun në të djathtë, dhe një `<` tregon një nxënës që po flet me shokun në të majtë. Sapo dëgjojnë mësuesin të hyjë përsëri, nxënësit që po flisnin kthehen nga frika në krahun e kundërt. Kur mësuesi shikon dy nxënës përball njëri-tjetrit ai mendon se ata po flisnin, kështu që i ndëshkon të dy. Gjeni sa dyshe nxënësish ndëshkohen prej mësuesit.

Referenca: https://www.codechef.com/problems/CHEFSTUD

Shembull

```
$ cat input.txt
4
><
*><*
><><
*><><><*

$ python3 prog.py < input.txt
```

```
0
0
1
2
```

Zgjidhja 1

```
for _ in range(int(input())):
    S = list(input())
    for i in range(len(S)):
        if S[i] == '>':
            S[i] = '<'
        elif S[i] == '<':
            S[i] = '>'
    nr = 0
    for i in range(len(S) - 1):
        if S[i] == '>' and S[i+1] == '<':
            nr += 1
    print(nr)
```

Zgjidhja 2

```
for _ in range(int(input())):
    S = input()
    nr = 0
    for i in range(len(S) - 1):
        if S[i] == '<' and S[i+1] == '>':
            nr += 1
    print(nr)
```

Zgjidhja 3

```
for _ in range(int(input())):
    print(input().count('<>'))
```

Problemi 131

Kërkesa

Një lojë luhet në një tabelë drejtkëndore me **N** rreshta dhe **M** shtylla. Fillimisht kutitë e tabelës janë të pangjyrosura. Lojtari fillon ti ngjyrosë një nga një, dhe për çdo kuti që ngjyros merr aq pikë sa fqinjë të ngjyrosur ka kjo kuti (fqinjët e një kutie janë ato kuti që kanë një brinjë të përbashkët me të). Shuma e këtyre pikëve janë pikët që fitohen nga kjo lojë. Bëni një program që gjen se sa janë pikët maksimale që mund të fitohen në këtë lojë.

Referenca: https://www.codechef.com/problems/OMWG

Shembull

```
$ cat input.txt
1
2 2

$ python3 prog.py < input.txt
4
```

Në një tabelë me 2 rreshta dhe 2 shtylla, numri maksimal i pikëve që mund të fitohen është 4.

Zgjidhja

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    print(2*n*m - n - m)
```

Sqarime

Sido që të ngjyrosen katrorët, në fund do kemi fituar një pikë për çdo brinjë që ndodhet midis dy katrorëve. Kështu që numri total i pikëve është gjithmonë i barabartë me numrin e këtyre brinjëve që ndodhen midis dy katrorëve.

Në një tabelë me përmasa **NxM** kemi **N*M katrorë të vegjël, të cilët gjithsej kanë 4*N*M brinjë. Po të heqim brinjët anësore, që nuk ndodhen midis dy katrorëve, na ngelen 4*N*M - 2*(N + M)** brinjë. Këto janë brinjët e brendshme, por të numëruara 2 herë, njëherë për llogari të njërit katror dhe njëherë për llogari të katrorit ngjitur me të. Kështu që numri i brinjëve të brendshme (që është edhe numri i pikëve) është: **2*N*M - N - M**

Problemi 132

Kërkesa

Ceni kishte një kuti me **N** numra në të: $A_1, A_2, ..., A_n$. Gjithashtu kishte vendosur edhe numrin **N** në krye, për të ditur sa numra janë. Pra kutia kishte gjithsej **N+1** numra. Por nga gëzimi i olimpiadës IOI ai filloi të kërcente me kutinë në xhep dhe numrat iu ngatërruan, dhe tani nuk e di më se cili nga **N+1** numrat është numri **N** dhe cilët janë numrat e tjerë. Atij i duhet të gjejë më të madhin e numrave të dhënë. A mund ta ndihmoni?

Referenca: https://www.codechef.com/problems/LOSTMAX

Shembull

```
$ cat input.txt
3
1 2 1
3 1 2 8
1 5 1 4 3 2

$ python3 prog.py < input.txt
1
8
4
```

Në rastin e parë, N=2 dhe numrat janë {1, 1}, kështu që më i madhi është 1.

Në rastin e dytë N=3 dhe numrat janë {1, 2, 8}. Më i madhi është 8.

Në rastin e tretë N=5 dhe numrat janë {1, 1, 4, 3, 2}. Më i madhi është 4.

Zgjidhja

```
for _ in range(int(input())):
    A = list(map(int, input().split()))
    A.sort(reverse=True)
    n = len(A) - 1
    if A[0] > n:
        print(A[0])
    else:
        print(A[1])
```

Problemi 133

Kërkesa

Nga rezultatet e një olimpiade mund të hamendësojmë nivelin e aftësive për secilin pjesëmarës. Pjesëmarrësit mund të klasifikohen bazuar në numrin e problemave të zgjidhura në këtë mënyrë:

- 0 problema të zgjidhura: Beginner
- 1 problem të zgjidhur: Junnior Developper
- 2 problema të zgjidhura: Middle Developer
- 3 problema të zgjidhura: Senior Developer
- 4 problema të zgjidhura: Hacker
- 5 problema të zgjidhura: Jeff Dean

Bëni një program që merr rezultatet e pjesëmarrësve dhe nxjerr klasifikimin e tyre.

Referenca: https://www.codechef.com/problems/CCOOK

Shembull

```
$ cat input.txt
7
0 0 0 0 0
0 1 0 1 0
0 0 1 0 0
1 1 1 1 1
0 1 1 1 0
0 1 1 1 1
1 1 1 1 0

$ python3 prog.py < input.txt
Beginner
Middle Developer
Junior Developer
Jeff Dean
Senior Developer
Hacker
Hacker
```

Zgjidhja 1

```
for _ in range(int(input())):
    P = list(map(int, input().split()))
    p = sum(P)    # or p = P.count(1)
    if p == 0:
        print('Beginner')
    elif p == 1:
        print('Junior Developer')
    elif p == 2:
        print('Middle Developer')
    elif p == 3:
        print('Senior Developer')
    elif p == 4:
        print('Hacker')
    else:
        print('Jeff Dean')
```

Zgjidhja 2

```
RANK = ['Beginner', 'Junior Developer', 'Middle Developer', 'Senior Developer', 'Ha
for _ in range(int(input())):
    P = list(map(int, input().split()))
    print(RANK[sum(P)])
```

Problemi 134

Kërkesa

Një ari i vogël polar ka qejf të hajë biskota dhe të pijë qumësht. Për këtë arsye shkon shpesh në kuzhinë. Ai qëndron në kuzhinë **N** minuta, dhe çdo minutë ose ha një biskotë, ose pi qumësht. Meqënëse biskotat janë shumë të ëmbla, prindërit e kanë këshilluar që sa herë të hajë një biskotë, në minutën tjetër duhet të pijë qumësht.

Na jepet sa minuta ka ndenjur në kuzhinë arushi dhe çfarë ka bërë çdo minutë: ka ngrënë biskotë apo ka pirë qumësht. Gjeni nëse arushi i ka zbatuar ose jo këshillat e prindërve, dmth që pas ngrënies së një biskote ka pirë qumësht.

Referenca: https://www.codechef.com/problems/COOMILK

Shembull

```
$ cat input.txt
```

```
4
7
cookie milk milk cookie milk cookie milk
5
cookie cookie milk milk milk
4
milk milk milk milk
1
cookie

$ python3 prog.py < input.txt
YES
NO
YES
NO
```

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    L = input().split()
    if L[-1] == 'cookie':
        print('NO')
    else:
        for i in range(n-1):
            if L[i] == 'cookie' and L[i+1] == 'cookie':
                print('NO')
                break
        else:
            print('YES')
```

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    L = input().split()
    L.append('cookie')
    for i in range(n):
        if L[i] == 'cookie' and L[i+1] == 'cookie':
            print('NO')
```

```
            break
    else:
        print('YES')
```

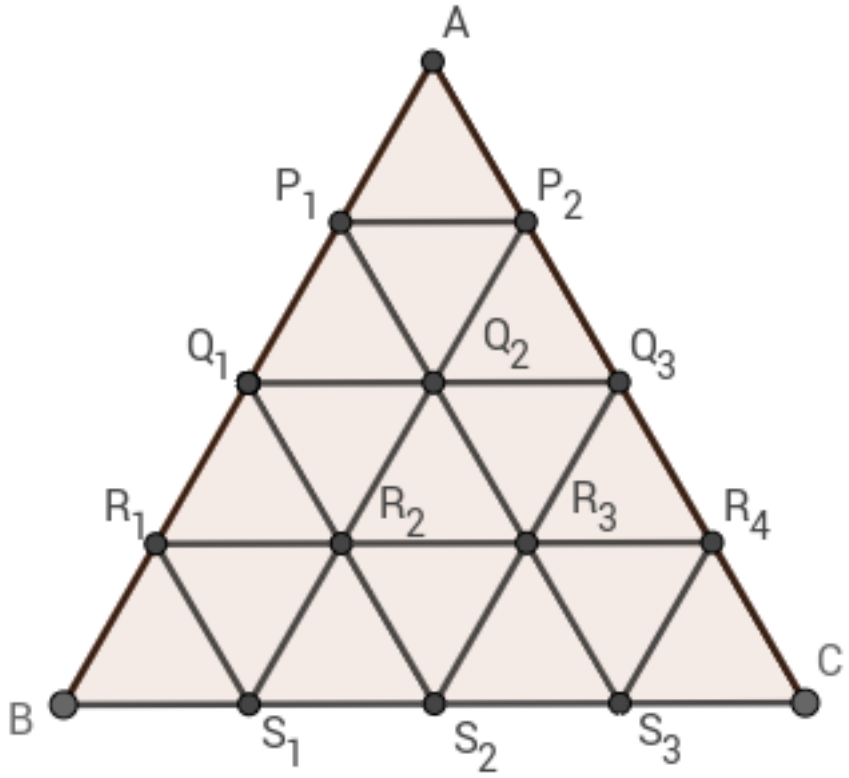
Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    L = input().split()
    L.append(None)
    for i in range(n):
        if L[i] == 'cookie' and L[i+1] != 'milk':
            print('NO')
            break
    else:
        print('YES')
```

Problemi 135

Kërkesa

Numrat natyrorë vendosen në nyjet e një grafi që ka formën e shinave të trenit, si në figurë:



Bëni një program që merr dy numra dhe gjen nëse ka brinjë midis këtyre dy nyjeve.

Referenca: https://www.codechef.com/problems/BRLADDER

Shembull

```
$ cat input.txt
7
1 4
4 3
5 4
10 12
1 3
999999999 1000000000
17 2384823

$ python3 prog.py < input.txt
NO
```

```
YES
NO
YES
YES
YES
NO
```

Zgjidhja

```
for _ in range(int(input())):
    a, b = map(int, input().split())
    if abs(a - b) > 2:
        print('NO')
    elif abs(a - b) == 2:
        print('YES')
    elif abs(a - b) == 1:
        if a // 2 == b // 2:
            print('NO')
        else:
            print('YES')
```

Problemi 136

Kërkesa

Në një listë me **N** numra natyrorë mund të shtojmë edhe **K** numra të tjerë sipas dëshirës, dhe pastaj gjejmë të mesmen e të gjithë numrave. E mesmja e një liste është elementi që ndodhet në mesin e listës, pasi ti kemi renditur. P.sh. e mesmja e **[2, 1, 5, 2, 4]** është **2**, kurse e mesmja e **[3, 3, 1, 3, 3]** është 3.

Gjeni se sa është vlera më e madhe e të mesmes që mund të merret në këtë mënyrë. Na jepet gjithashtu që $\mathbf{K} < \mathbf{N}$ dhe që $\mathbf{N} + \mathbf{K}$ është numër tek.

Referenca: https://www.codechef.com/problems/CK87MEDI

Shembull

```
$ cat input.txt
3
2 1
4 7
```

```
4 3
9 2 8 6
5 2
6 1 1 1 1

$ python3 prog.py < input.txt
7
9
1
```

Na jepen numrat **N** dhe **K**, dhe pastaj N numrat e listës.

Në rastin e parë, një nga zgjidhjet e mundshme është të shtojmë 9, duke formuar listën [4, 7, 9], e mesmja e të cilës është 7.

Në rastin e tretë, sido që të jenë 2 numrat që do shtohen, e mesmja do jetë gjithmonë 1.

Zgjidhja

```python
for _ in range(int(input())):
    n, k = map(int, input().split())
    L = [int(i) for i in input().split()]
    L.sort()
    print(L[(n + k) // 2])
```

Problemi 137

Kërkesa

Një bankomat (ATM) ka brënda **K** para. Në rradhë janë **N** klientë, ku secili do që të tërheqë A_i para. Nëse bankomati ka gjëndje aq sa kërkon klienti ose më shumë, ai ia jep lekët klientit, përndryshe nxjerr një mesazh gabimi dhe klienti largohet pa marrë lekë. Gjeni për secilin klient nëse i ka marrë lekët ose jo.

Referenca: https://www.codechef.com/problems/ATM2

Shembull

```
$ cat input.txt
2
5 10
3 5 3 2 1
```

```
4 6
10 8 6 4

$ python3 prog.py < input.txt
11010
0010
```

Kur klienti i merr lekët shënohet me **1**, përndryshe shënohet me **0**.

Në rastin e parë janë 5 klientë dhe bankomati ka 10 lekë. Pasi merr lekët klienti i parë dhe i dytë, në bankomat mbeten 2 lekë, të cilat janë të pamjaftueshme për ti dhënë lekët klientit të tretë, por mjaftojnë për ti dhënë lekët klientit të katërt.

Në rastin e dytë janë 4 klientë dhe bankomati ka 6 lekë. Dy klientët e parë nuk mund ti tërheqin lekët, por i treti po. Pastaj nuk ngelet më asnjë lekë, kështu që as klienti i katërt nuk mund ti tërheqë lekët.

Zgjidhja

```python
for _ in range(int(input())):
    n, k = map(int, input().split())
    L = [int(i) for i in input().split()]
    for l in L:
        if l <= k:
            print(1, end='')
            k -= l
        else:
            print(0, end='')
    print()
```

Problemi 138

Kërkesa

Kemi shumë topa pinpongu të bardhë dhe të zinj dhe kemi krijuar dy rreshta me gjatësi **N** përkrah njëri-tjetrit. Këta rreshta i shënojmë si vargjet **X** dhe **Y**, të përbërë prej shkronjave **W** and **B**, ku **W** shënon një top të bardhë (white) kurse **B** një top të zi (black).

Tani duam të krijojmë një rresht të tretë **Z** në mënyrë që **ham(X, Z) + ham(Y, Z)** të jetë sa më e madhe. Funksioni **ham(A, B)** është Hamming Distance midis vargjeve **A** dhe **B** me të njëjtën gjatësi. Kjo distancë midis dy vargjeve është numri i pozicioneve ku

këto vargje kanë shkronja të ndryshme. P.sh. **ham('WBB', 'BBW')** është **2**, meqënëse shkronjat e para dhe të treta janë të ndrushme.

Meqënëse mund të ketë zgjidhje të ndryshme, programi duhet të nxjerrë atë zgjidhje që është alfabetikisht më e vogël. P.sh. shkronja **B** është më e vogël se shkronja **W** sepse ndodhet përpara saj në alfabet.

Referenca: https://www.codechef.com/problems/ACBALL

Shembull

```
$ cat input.txt
1
WBWB
WBBB

$ python3 prog.py < input.txt
BWBW
```

Zgjidhja

```
for _ in range(int(input())):
    X = input()
    Y = input()
    Z = []
    for i in range(len(X)):
        if X[i] != Y[i]:
            Z.append('B')
        elif X[i] == 'B':
            Z.append('W')
        else:
            Z.append('B')
    print(''.join(Z))
```

Sqarime

Nqs topat korrespondues në një pozicion të caktuar në **X** dhe **Y** janë me ngjyra të ndryshme, atere sido që ta zgjedhim topin për rreshtin **Z** shuma e kërkuar do rritet vetëm me 1. Por meqënëse programi duhet të nxjerrë një zgjidhje që të jetë alfabetikisht sa më e vogël, atere ne zgjedhim një top **B** (black).

Në rastin kur topat në **X** dhe **Y** janë me të njëjtën ngjyrë, atere në **Z** zgjedhim një top me ngjyrë të kundërt, sepse kjo e shton vlerën e shumës me 2.

Problemi 139

Kërkesa

Në një dyqan çokollatash shkon një turist. Shitësi dhe turisti nuk e dinë gjuhën e njëritjetrit, por turisti tregon me dorë një lloj çokollatash dhe i jep shitësit disa kartmonedha. Duke ditur çmimin e një çokollate shitësi mund ta marrë me mend se sa copë do që të blejë turisti. Por nqs duke hequr një prej kartmonedhave mund të blihet përsëri i njëjti numër çokollatash, atere ky është një rast i paqartë dhe shitësi nuk di çfarë të bëjë, kështu që ia kthen lekët mbrapsht turistit.

Bëni një program që merr çmimin e një çokollate dhe kartmonedhat që jep turisti dhe gjen se sa copë çokollata do që të blejë turisti. Nëse rasti është i paqartë programi duhet të nxjerrë -1.

Referenca: https://www.codechef.com/problems/BUYING2

Shembull

```
$ cat input.txt
3
4 7
10 4 8 5
1 10
12
2 10
20 50

$ python3 prog.py < input.txt
-1
1
7
```

Në rastin e parë, sasia gjithsej e parave është 27, dhe meqë çmimi është 7, turisti mund të marrë vetëm 3 copë. Por nqs nuk do kishte dhënë kartmonedhën 5 përsëri do merrte 3 copë. Kështu që është rast i paqartë.

Në rastin e dytë sasia e parave është 12, çmimi 10, kështu që mund të marrë 1 çokollatë.

Në rastin e tretë sasia e parave është 20+50, çmimi 10, mund të marrë 7 çokollata.

Zgjidhja

```
for _ in range(int(input())):
    n, x = map(int, input().split())
    A = list(map(int, input().split()))
    print(sum(A) // x) if (sum(A) % x < min(A)) else print(-1)
```

Sqarime

Nqs mbetja e parave (restoja) është më e madhe se ndonjëra nga kartmonedhat, kjo do të thotë që po ta heqim atë kartmonedhë përsëri mund të blejmë të njëjtën sasi çokollatash, kështu që rasti është i paqartë.

Problemi 140

Kërkesa

Ada ka **N** lapsa me ngjyra, disa me majën poshtë dhe disa me majën lart. Ada mendon se do ishte më murë sikur të gjithë lapsat të ishin në të njëjtin drejtim. Ajo mund të kthejë njëherësh një segment të tërë me lapsa të njëpasnjëshëm. Pas një kthimi, ata që ishin me majë poshtë do kthehen me majën lart, dhe anasjelltas. Sa është numri më i vogël i kthimeve për të bërë të gjithë lapsat me të njëjtin drejtim?

Referenca: https://www.codechef.com/problems/ADACRA

Shembull

```
$ cat input.txt
1
UUDDDUUU

$ python3 prog.py < input.txt
1
```

Me **U** shënohet një laps me majën lart dhe me **D** një laps që e ka majën poshtë. Në rastin e dhënë, mjafton vetëm një kthim për ti vendosur me majën lart të gjithë lapsat që janë me majën poshtë.

Zgjidhja

```
for _ in range(int(input())):
    s = input()
    u = 0    # nr i segmenteve me majen lart
    d = 0    # nr i segmenteve me majen poshte
    if s[0] == 'U':
        u += 1
    else:
        d += 1
    for i in range(1, len(s)):
        if s[i] != s[i-1]:
            if s[i] == 'U':
                u += 1
            else:
                d += 1
    print(min(u, d))
```

Problemi 141

Kërkesa

Jepet një varg me zero-njësha. Gjeni numrin e nënvargjeve që fillojnë dhe mbarojnë me **1**.

Referenca: https://www.codechef.com/problems/CSUB

Shembull

```
$ cat input.txt
2
4
1111
5
10001

$ python3 prog.py < input.txt
10
3
```

Zgjidhja 1

```
for _ in range(int(input())):
    input()
    V = input()
    nr = 0
    for d in V:
        if d == '1':
            nr += 1
    print( nr*(nr + 1)//2 )
```

Sqarime

Nqs numri i njëshave është `n`, atere numri i segmenteve që fillojnë dhe mbarojnë me **1** është `n*(n + 1)/2`. Vërtet, nqs marrim njëshin e parë si fillim të segmentit, si fund të segmentit mund të marrim secilin nga **n** njëshat. Nqs si fillim marrim njëshin e dytë, si fund mund të marrim **n-1** njësha, duke filluar nga i dyti. E kështu me rradhë, numri i segmenteve që mund të formohen është `n + (n-1) + (n-2) + ... + 2 + 1`.

Zgjidhja 2

```
for _ in range(int(input())):
    input()
    V = input()
    nr = V.count('1')
    print( nr*(nr + 1)//2 )
```

Problemi 142

Kërkesa

Labirinti më i thjeshtë në botë përbëhet nga një tabelë me përmasa **NxN**. Duke filluar në cepin e majtë sipër, duhet vajtur në cepin e djathtë poshtë, duke bërë vetëm lëvizje majtas (**E**ast) ose poshtë (**S**outh).

Na jepen përmasat e tabelës dhe zgjidhja që ka bërë Lida. A mund të gjeni një zgjidhje tjetër që është komplet e ndryshme nga ajo që ka bërë Lida (dmth nqs zgjidhja e Lidës përmban një kalim nga qeliza A në qelizën B, zgjidhja juaj nuk duhet të përmbajë një kalim të tillë).

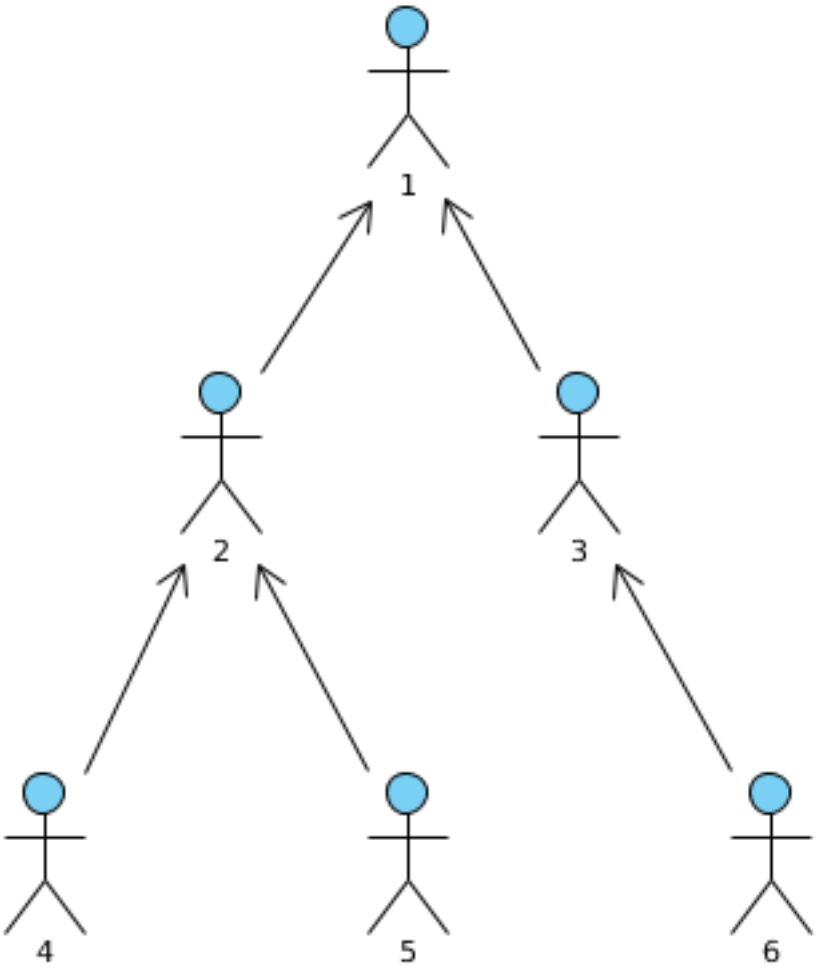
Referenca: https://codingcompetitions.withgoogle.com/codejam/round/0000000000051705/00000000000881da

Shembull

```
$ cat input.txt
2
2
SE
5
EESSSESE

$ python3 prog.py < input.txt
Case #1: ES
Case #2: SEEESSES
```

Rasti i dytë është si në figurë:



Zgjidhja

```
for t in range(int(input())):
    input()
    p1 = input()
    p2 = []
    for c in p1:
        p2.append('E' if c=='S' else 'S')
    print('Case #{}: {}'.format(t+1, ''.join(p2)))
```

Sqarime

Mjafton të marrim simetriken e zgjidhjes së dhënë. Kjo prapë është zgjidhje dhe është e garantuar që është komplet e ndryshme nga zgjidhja e dhënë.

Problemi 143

Kërkesa

Jepet një listë me numra. A mund të formohet një palindromë me këta numra?

Referenca: https://www.codechef.com/DARG2019/problems/TUPALIN

Shembull

```
$ cat input.txt
1
6
10 729 10 10 729 10

$ python3 prog.py < input.txt
YES
```

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    L = [int(i) for i in input().split()]
    F = {}
    for i in L:
        F[i] = F.get(i, 0) + 1
    odd = 0
    for i in F.keys():
        if F[i] % 2 == 1:
            odd += 1
    print('YES') if odd < 2 else print('NO')
```

Sqarime

Gjejmë se sa herë përsëritet çdo numër në këtë listë. Një palindromë mund të formohet vetëm nëse numri i përsëritjeve për çdo numër është çift, me përjashtim të njërit që

mund të jetë tek.

Problemi 144

Kërkesa

Jepet një varg me numra natyrorë $A_1, A_2, ..., A_N$. Ju mund të zgjidhni një numër natyror **X** dhe të bëni me të veprimin **bitwise XOR** me secilin nga numrat e vargut, pastaj të merrni shumën e këtyre numrave. Sa është vlera më e vogël e mundshme e kësaj shume.

Referenca: https://www.codechef.com/LTIME71B/problems/MINARRS

Shembull

```
$ cat input.txt
3
5
2 3 4 5 6
4
7 7 7 7
3
1 1 3

$ python3 prog.py < input.txt
14
0
2
```

Në rastin e parë mund të zgjedhin X = 6, dhe vargu bëhet (4, 5, 2, 3, 0).

Në rastin e dytë mund të zgjedhim X = 7 dhe të gjithë numrat e vargut bëhen 0.

Në rastin e tretë mund të zgjedhim X = 1, dhe vargu bëhet (0, 0, 2), shuma e të cilit është 2.

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    L = [format(int(i), '030b') for i in input().split()]
    X = ['0' for i in range(30)]
    for i in range(30):
```

```
        c = 0
        for j in range(n):
            if L[j][i] == '1':
                c += 1
        if c > n - c:
            X[i] = '1'
    x = int(''.join(X), 2)
    s = 0
    for a in L:
        s += int(a,2) ^ x
    print(s)
```

Problemi 145

Kërkesa

Një klasë ka **N** nxënës dhe pikët e secilit prej tyre na jepen në një listë **L**. Gjeni të gjitha prerjet e mundshme $P = (x, y)$ që mund ti bëhen kësaj liste, të tilla që shuma e pikëve brënda prerjes të jetë e barabartë me shumën e pikëve jashtë prerjes. Dmth $L_x + L_{x+1} + ... + L_{y-1} + L_y = (L_0 + L_1 + ... + L_{x-1}) + (L_{y+1} + L_{y+2} + ... + L_{N-1})$.

Nëse nuk ka asnjë prerje të tillë printoni **-1**, përndryshe printoni numrin e prerjeve dhe shumën e indekseve **x** dhe **y** për të gjitha prerjet. Këto indekse janë me bazë 0, dhe $0 < x <= y < N - 1$. Gjithashtu $3 <= N <= 10^5$ dhe $1 <= L_i <= 10^5$.

Referenca: https://www.codechef.com/problems/EQSBAR

Shembull

```
$ cat input.txt
2
10
7 8 12 8 13 9 5 7 12 3
5
32 36 34 20 28

$ python3 prog.py < input.txt
2 17
-1
```

Zgjidhja 1

```
for _ in range(int(input())):
    n = int(input())
    L = [int(i) for i in input().split()]

    nr = 0
    idx_sum = 0
    for i in range(1, n - 1):
        for j in range(i, n - 1):
            if sum(L[:i]) + sum(L[j+1:]) == sum(L[i:j+1]):
                nr += 1
                idx_sum += i + j

    if nr == 0:
        print(-1)
    else:
        print(nr, idx_sum)
```

Sqarime

Koha e zgjidhjes është e rendit $O(N^3)$.

Zgjidhja 2

```
for _ in range(int(input())):
    n = int(input())
    L = [int(i) for i in input().split()]

    S = [0 for i in range(n)]
    S[0] = L[0]
    for i in range(1, n):
        S[i] = S[i-1] + L[i]

    nr = 0
    idx_sum = 0
    scores = sum(L) // 2
    for i in range(1, n - 1):
        for j in range(i, n - 1):
            if S[j] - S[i-1] == scores:
```

```
                nr += 1
                idx_sum += i + j

    if nr == 0:
        print(-1)
    else:
        print(nr, idx_sum)
```

Sqarime

Në listën S mbajmë shumat e pjesëshme nga fillimi deri në atë pozicion ku jemi. Këto mund të gjenden shumë kollaj me një bredhje nga fillimi deri në fund, dhe ndihmojnë që të gjendet në mënyrë efiçente shuma e numrave të një prerje.

Koha e kësaj zgjidhje është e rendit $O(N^2)$.

Zgjidhja 3

```
for _ in range(int(input())):
    n = int(input())
    L = [int(i) for i in input().split()]
    if sum(L) % 2 == 1:
        print(-1)
    else:
        idx_sum = 0
        scores = sum(L) // 2
        nr = 0
        i = j = 1
        s = L[1]
        while True:
            while s < scores and j < n - 1:
                j += 1
                s += L[j]
            if s < scores:
                break
            if s == scores:
                nr += 1
                idx_sum += i + j
                if j < n - 1:
                    j += 1
                    s += L[j]
```

```
            while s > scores:
                s -= L[i]
                i += 1
            if s == scores:
                nr += 1
                idx_sum += i + j
                s -= L[i]
                i += 1
        if nr == 0:
            print(-1)
        else:
            print(nr, idx_sum)
```

Sqarime

Së pari vërejmë që shuma e pikëve brënda segmentit duhet të jetë sa gjysma e pikëve totale. Nqs pikët totale janë numër tek, atere nuk mund të ketë asnjë prerje që plotëson kushtet.

Mbajmë dy indekse **i** dhe **j** që shënojnë numrin e parë dhe të fundit të segmentit. Vazhdojmë të rrisim indeksin **j**, duke llogaritur edhe shumën brenda segmentit, deri sa ta kalojë gjysmën e pikëve, pastaj vazhdojmë të lëvizim indeksin **i** deri sa shuma të bëhet më pak se gjysma e pikëve. Gjatë kësaj kohe kontrollojmë edhe nëse shuma bëhet e barabartë me gjysmën e pikëve, e cila do ishte një prerje që plotëson kushtet.

Koha e kësaj zgjidhje është e rendit $O(N)$, që është shumë më e mirë se zgjidhja e parë.

Detyra

Problemi 146

Kërkesa

Kemi një listë me pikë (numra natyrorë) të shkruara në një shirit të tejdukshëm. Kemi edhe një shirit me veprime modifikuese, me gjatësi më të madhe se lista e pikëve, mbi të cilin mund të vendoset shiriti i tejdukshëm dhe të llogariten veprimet totale.

Shiriti modifikues ka këto veprime:

- Pikë (**.**): nuk bën asnjë modifikim.
- Dyfisho pikët (**d**): dyfishon numrin e pikëve që përputhen me të.
- Trefisho pikët (**t**): trefishon numrin e pikëve që përputhen me të.
- Dyfisho totalin (**D**): dyfishon shumën e të gjitha pikëve, dhe zbatohet në fund.

- Trefisho totalin (**T**): trefishon shumën e të gjitha pikëve, dhe zbatohet në fund.

Shiriti me pikë vendoset mbi shiritin e modifikuesve në një pozicion të caktuar dhe bëhet shuma e pikëve, duke e dyfishuar ose trefishuar numrin e pikëve më parë, nëse përputhen me veprimet **d** dhe **t**. Kurse veprimet **D** dhe **T** bëjnë dyfishimin ose trefishimin e shumës totale dhe zbatohen gjithmonë në fund.

Gjeni pozicionin që jep vlerën më të madhe totale të pikëve dhe thoni sa është ajo vlerë.

Referenca: https://www.codechef.com/problems/MAXSCRAB

Shembull

```
$ cat input.txt
2
10
..d.t.D..d
10 11 12 9 8 10 11 15
22
dtDtTD..ddT.TtTdDT..TD
12297 5077 28888 17998 12125 27400 31219 21536

$ python3 prog.py < input.txt
270
35629632
```

Zgjidhja

```
for _ in range(int(input())):
    n = int(input())
    B = input()
    L = [int(i) for i in input().split()]
    max_score = 0
    for i in range(n - len(L) + 1):
        L1 = []
        B1 = []
        for j in range(len(L)):
            c = B[i+j]
            if c == '.':
                L1.append(L[j])
            elif c == 'd':
                L1.append(2*L[j])
            elif c == 't':
```

```
                L1.append(3*L[j])
            else:
                L1.append(L[j])
                B1.append(c)
        score = sum(L1)
        for c in B1:
            if c == 'D':
                score *= 2
            else:
                score *= 3
        if score > max_score:
            max_score = score
    print(max_score)
```

Problemi 147

Kërkesa

Vanesa ka **N** xhingla në raftin e saj, të numëruara 1, 2, ..., N nga e majta në të djathtë. Xhinglat janë të llojeve të ndryshme, ku çdo lloj shënohet me një numër. Xhingla që ndodhet në pozicionin **i** në raftin e saj është e llojit A_i.

Vanesa studjon jashtë dhe sot do kthehet që të vizitojë familjen. Ajo do që të marrë sa më shumë xhingla me vete, por ngaqë është me nxitim i duhet të rrëmbejë një varg të vazhdueshëm xhinglash. Po ta themi me gjuhë shkencore, Vanesa zgjedh me mënd dy pozicione **l** dhe **r** dhe vërvit në çantë të gjitha xhinglat që ndodhen në vëndet $l, l + 1, ..., r - 1, r$. Gjithashtu, për shkak të rregullave të aeroportit, kontrolluesit do hedhin poshtë **të gjitha** xhinglat e një lloji të caktuar, nëse ajo ka më shumë se **S** të tilla.

P.sh. nqs **S** = 2 dhe Vanesa ka me vete 6 xhingla, një të llojit 0, dy të llojit 1, dhe tre të llojit 2, asaj do ti lejohen xhinglat e llojit 0 dhe 1, por do ti hidhen poshtë të treja xhinglat e llojit 2.

Vanesa duhet të zgjedhë **l** dhe **r** të tilla që të marrë sa më shumë xhingla me vete. Sa është numri më i madh i xhinglave që mund të marrë me vete Vanesa?

Referenca: https://codingcompetitions.withgoogle.com/kickstart/round/0000000000050eda/00000000001198c1

Shembull

```
$ cat input.txt
4
6 2
```

```
1 1 4 1 4 4
8 1
1 2 500 3 4 500 6 7
10 1
100 200 8 8 8 8 8 300 400 100
12 2
40 50 1 1 1 60 70 2 2 2 80 90

$ python3 prog.py < input.txt
Case #1: 4
Case #2: 6
Case #3: 4
Case #4: 6
```

Për çdo rast testimi jepet numri i xhinglave **N** dhe maksimumi i lejuar në aeroport **S**. Pastaj jepet vargu i xhinglave sipas llojit të secilës prej tyre.

Në rastin e parë Vanesa duhet të zgjedhë l=2 dhe r=5. Kjo i lejon asaj të marrë xhinglat [1, 4, 1, 4], asnjë prej të cilave nuk hidhet poshtë në aeroport.

Në rastin e dytë Vanesa duhet të zgjedhë l=1 dhe r=8. Xhinglat e llojit 500 do hidhen poshtë meqë janë më shumë se S=1 të tilla, kështu që ajo do marrë me vete 6 xhingla.

Në rastin e tretë duhet të zgjedhë l=1 dhe r=9 dhe të sjallë në aeroport xhinglat [100, 200, 8, 8, 8, 8, 8, 300, 400]. Xhinglat e llojit 8 do hidhen poshtë dhe do ti ngelen 4 xhingla me vete.

Në rastin e katërt duhet të zgjedhë l=1 dhe r=12. Xhinglat e llojit 1 dhe 2 do hidhen poshtë meqë ka më shumë se S=2 të tilla, kështu që do ti mbeten 6 xhingla.

Zgjidhja

```python
for t in range(int(input())):
    N, S = [int(i) for i in input().split()]
    A = [int(i) for i in input().split()]

    max_items = 0
    for i in range(N):
        type_freq = {}
        nr_items = 0
        for j in range(i,N):
            a = A[j]
            type_freq[a] = type_freq.get(a, 0) + 1
            if type_freq[a] <= S:
```

```
                nr_items += 1
            elif type_freq[a] == S + 1:
                nr_items -= S
            if nr_items > max_items:
                max_items = nr_items

    print('Case #{}: {}'.format(t+1, max_items))
```

Sqarime

Koha e kësaj zgjidhje është e rendit $O(N^2)$.

Detyra

Problemi 148

Kërkesa

Një shkop kërcimi *pogo* ka një sustë në fund dhe me të mund të hidhesh nga një vënd në një vënd tjetër, duke qëndruar mbi të. Por shkopi që ju kanë bërë dhuratë është i veçantë sepse herën e parë kërcen me një largësi 1 njësi, herën e dytë 2 njësi, herën e tretë 3 njësi, e kështu me rradhë. Ju mund të kërceni në një nga këto drejtime: veri, jug, lindje, perëndim.

Nqs ndodheni në pikën (0, 0), si mund të shkoni në pikën (**X**, **Y**)? Si mund të shkoni me sa më pak kërcime?

Boshti X është sipas drejtimit S-N (jug-veri) dhe boshti Y është sipas drejtimit W-E (perëndim-lindje). Pika (X, Y) është e ndryshme nga (0, 0).

Referenca: https://code.google.com/codejam/contest/2437488/dashboard#s=p1&a=1

Shembull

```
$ cat input.txt
2
3 4
-3 4

$ python3 prog.py < input.txt
Case #1: ENWSEN
Case #2: ENSWN
```

Vini re që në rastin e parë ka edhe një rrugë tjetër që është më e shkurtër: WNSEN

Zgjidhja 1

Kjo është një nga zgjidhjet më të thjeshta.

```
for t in range(int(input())):
    x, y = [int(i) for i in input().split()]
    ans = ''
    while x > 0:
        ans += 'WE'
        x -= 1
    while x < 0:
        ans += 'EW'
        x += 1
    while y > 0:
        ans += 'SN'
        y -= 1
    while y < 0:
        ans += 'NS'
        y += 1
    print('Case #{}: {}'.format(t+1, ans))
```

Zgjidhja 2

Kjo zgjidhje gjen rrugën më të shkurtër.

```
for t in range(int(input())):
    x, y = [int(i) for i in input().split()]
    n = 0
    s = 0
    while s < abs(x) + abs(y) or (s + x + y) % 2 == 1:
        n += 1
        s += n
    ans = ''
    while n > 0:
        if abs(x) > abs(y):
            if x > 0:
                ans += 'E'
                x -= n
            else:
                ans += 'W'
                x += n
        else:
            if y > 0:
```

```
                ans += 'N'
                y -= n
            else:
                ans += 'S'
                y += n
        n -= 1
    print('Case #{}: {}'.format(t+1, ''.join(reversed(ans))))
```

Problemi 149

Kërkesa

Në një varg **S** me shkronja të vogla të alfabetit (anglisht), gjeni se sa nënvargje përmbajnë **N** bashkëtingëllore të njëpasnjëshme.

Referenca: https://code.google.com/codejam/contest/2437488/dashboard#s=p0&a=0

Shembull

```
$ cat input.txt
4
quartz 3
straight 3
gcj 2
tsetse 2

$ python3 prog.py < input.txt
4
quartz 3
straight 3
gcj 2
tsetse 2
```

Për çdo rast testimi jepet vargu **S** dhe numri **N**.

Zgjidhja 1

```
for t in range(int(input())):
    S, n = input().split()
    n = int(n)
```

```
    L = len(S)
    cnt = 0
    for i in range(L):
        for j in range(i+1, L+1):
            c = 0
            for k in range(i, j):
                if S[k] not in "aeiou":
                    c += 1
                else:
                    c = 0
                if c >= n:
                    cnt += 1
                    break

    print('Case #{}: {}'.format(t+1, cnt))
```

Sqarime

Shqyrtojmë të gjitha nënvargjet e mundshme, dhe për çdo nënvarg kontrollojmë nëse përmban **n** bashkëtingëllore të njëpasnjëshme.

Koha e kësaj zgjidhje është e rendit $O(L^3)$.

Zgjidhja 2

```
for t in range(int(input())):
    S, n = input().split()
    n = int(n)

    L = len(S)
    cnt = 0
    for i in range(L):
        c = 0
        for j in range(i, L):
            if S[j] not in "aeiou":
                c += 1
            else:
                c = 0
            if c >= n:
                cnt += L - j
                break
```

```
    print('Case #{}: {}'.format(t+1, cnt))
```

Sqarime

Kjo zgjidhje shfrytëzon faktin që nqs një nënvarg që fillon te **i** dhe mbaron te **j** përmban **n** bashkëtingëllore të njëpasnjëshme, atere edhe të gjitha nënvargjet e tjera që fillojnë në **i** dhe mbarojnë në **j+1**, **j+2**, e deri në **L** do përmbajnë të paktën **n** bashkëtingëllore të njëpasnjëshme.

Kjo zgjidhje është e rendit $O(L^2)$.

Zgjidhja 3

```
for t in range(int(input())):
    S, n = input().split()
    n = int(n)

    L = len(S)
    cnt, r, c = 0, 0, 0
    for i in range(L):
        if S[i] not in "aeiou":
            c += 1
        else:
            c = 0
        if c >= n:
            cnt += (i - n - r + 2) * (L - i)
            r = i - n + 2

    print('Case #{}: {}'.format(t+1, cnt))
```

Sqarime

Kjo zgjidhje është e rendit $O(L)$. Për një analizë të hollësishme dhe për më shumë sqarime shikoni: https://code.google.com/codejam/contest/2437488/dashboard#s=a&a=0

Problemi 150

Kërkesa

Në një konkurs jepet një varg **S** me shkronja të mëdha (anglisht). Shkronjat e kësaj fjale i jepen sipas rradhës konkurentit, i cili po formon një varg të ri. Konkurenti mund ti vendosë shkronjat që i jepen në fillim ose në fund të vargut që po formon. Ai fiton nqs vargu që formon është i fundit (sipas rendit alfabetik) nga të gjitha vargjet që mund të formohen.

P.sh. nqs vargu i dhënë është **S** = CAB, mund të formojë këto vargje:

- vendos A para C dhe formohet AC, vendos B para AC dhe formohet BAC
- vendos A para C dhe formohet AC, vendos B pas AC dhe formohet ACB
- vendos A pas C dhe formohet CA, vendos B para CA dhe formohet BCA
- vendos A pas C dhe formohet CA, vendos B pas CA dhe formohet CAB

Vargu fitues (i fundit alfabetikisht) nga këta është CAB. Nqs **S** = JAM, atere vargu fitues do ishte MJA.

Referenca: https://code.google.com/codejam/contest/4304486/dashboard#s=p0&a=0

Shembull

```
$ cat input.txt
8
CAB
JAM
CODE
ABAAB
CABCBBABC
ABCABCABC
ZXCASDQWE
DGFKGKTYRYEIRMDBACDFSEORLFNDF

$ python3 prog.py < input.txt
Case #1: CAB
Case #2: MJA
Case #3: OCDE
Case #4: BBAAA
Case #5: CCCABBBAB
Case #6: CCCBAABAB
Case #7: ZXCASDQWE
Case #8: YYTKKGDFGREIRMDBACDFSEORLFNDF
```

Zgjidhja 1

```
for t in range (int(input())):
    S = input()
    A = set([''])    # set of all possible strings
    for c in S:
        A = set([c + s for s in A] + [s + c for s in A])
    print('Case #{}: {}'.format(t+1, max(A)))
```

Sqarime

Kjo zgjidhje gjen të gjitha vargjet e mundshme që mund të krijohen dhe pastaj kthen më të madhin prej tyre. Mund të quhet edhe zgjidhje kombinatoriale, meqë merr në shqyrtim të gjitha kombinimet e mundshme.

Ndërlikueshmëria (kompleksiteti) i kësaj zgjidhjeje është eksponencial, edhe në kohë, edhe në hapësirën që duhet në kujtesë: $O(2^N)$ ku $\mathbf{N}$ është gjatësia e vargut të dhënë.

P.sh. te rasti i fundit kjo zgjidhje ngec. Zgjidhjet eksponenciale janë shpërthyese. (Kujdes se mos ju hedh kompjuterin në erë! Edhe bomba atomike shpërthen ngaqë grimcat e uranit fillojnë të ndahen në mënyrë eksponenciale.)

Zgjidhja 2

```
for t in range (int(input())):
    S = input()
    S1 = ''    # the new string
    for c in S:
        S1 = c + S1 if c + S1 > S1 + c else S1 + c
    print('Case #{}: {}'.format(t+1, S1))
```

Sqarime

Kjo zgjidhje është lineare. Ndërlikushmëria: $O(N)$.

Për një diskutim të detajuar se pse kjo zgjidhje jep rezultatin e duhur shikoni: https://code.google.com/codejam/contest/4304486/dashboard#s=a&a=0

Zgjidhja 3

```
for t in range (int(input())):
    S = input()
    S1 = ''     # the new string
    for c in S:
        S1 = min(c + S1, S1 + c)
    print('Case #{}: {}'.format(t+1, S1))
```

Problemi 151

Kërkesa

Viola nuk është shqiptare por ka edhe gjak shqiptar. Të paktën një nga stërgjyshërit e saj ka qënë puro shqiptar, por ajo nuk e di sa gjenerata më parë.

Ligjet e trashëgimisë gjenetike funksionojnë kështu: Nëse njëri nga prindërit është A/B shqiptar dhe tjetri është C/D shqiptar, atere fëmija do jetë (A/B + C/D) / 2 shqiptar. P.sh. nqs njëri nga prindërit nuk është fare shqiptar (0/1) kurse tjetri është gjysëm shqiptar (1/2), atere fëmija do jetë çerek shqiptar (1/4).

Viola pretendon se është **P**/**Q** shqiptare. Para sa gjeneratash ka pasur një stërgjysh puro shqiptar? Në nuk ka mundësi që të jetë **P**/**Q** shqiptare, atere duhet të thoni që kjo është e pamundur.

Referenca: https://code.google.com/codejam/contest/3004486/dashboard#s=p0&a=0

Shembull

```
$ cat input.txt
5
1/2
3/4
1/4
2/23
123/31488

$ python3 prog.py < input.txt
Case #1: 1
Case #2: 1
Case #3: 2
Case #4: impossible
Case #5: 8
```

Thyesa e dhënë $\mathbf{P/Q}$ mund të mos jetë e thjeshtuar (siç është p.sh. rasti i fundit).

Në rastin e parë Viola mund ti ketë pasur prindërit 1/1 (puro shqiptar) dhe 0/1 (jo shqiptar).

Në rastin e dytë mund ti ketë pasur prindërit 1/1 (puro shqiptare) dhe 1/2.

Në rastin e tretë mund ti ketë pasur prindërit 0/1 dhe 1/2, dhe prindi 1/2 i ka pasur prindërit 0/1 dhe 1/1 (puro shqiptar).

Në rastin e katërt, sido që të ketë qënë kombinimi i prindërve, gjyshërve dhe stërgjyshërve, nuk ka mundësi që rezultati të dalë 2/23.

Zgjidhja 1

```
from itertools import combinations_with_replacement
from math import gcd

for t in range(int(input())):
    P, Q = [int(i) for i in input().split('/')]
    g = gcd(P, Q)
    P = P // g
    Q = Q // g

    # kontrollo nese Q eshte fuqi e dyshit
    if Q & (Q - 1) != 0:
        print('Case #{}: {}'.format(t+1, 'impossible'))
        continue

    generation = [(0, 1, None, None), (1, 1, None, None)]
    found = False
    while not found:
        new_generation = []
        for parent1, parent2 in combinations_with_replacement(generation, 2):
            p = parent1[0] + parent2[0]
            q = parent1[1] + parent2[1]
            person = (p, q, parent1, parent2)
            new_generation.append(person)
            g = gcd(p, q)
            p //= g
            q //= g
            if p == P and q == Q:
                found = True
                break
```

```
        generation = new_generation

    gen = 0
    ancestors = [person]
    found = False
    while not found:
        previous_ancestors = []
        for p, q, parent1, parent2 in ancestors:
            if p == q:
                found = True
                break
            previous_ancestors.append(parent1)
            previous_ancestors.append(parent2)
        ancestors = previous_ancestors
        if not found:
            gen += 1

    print('Case #{}: {}'.format(t+1, gen))
```

Sqarime

Po të shikojmë rregullin e trashëgimisë, mund të vëmë kollaj se Q është gjithmonë një fuqi e dyshit. Nëse Q-ja e dhënë nuk është e tillë, atere raporti i dhënë është i pamundur që të gjenerohet me anë të rregullave të trashëgimisë.

Duke u nisur nga raportet 0/1 dhe 1/1 ne gjenerojmë të gjitha kombinimet e mundshme për çdo gjeneratë, dhe ndalojmë kur na del një raport i njëjtë me atë që na është dhënë.

Ndërkohë, për çdo person kemi ruajtur edhe prindërit e tij, kështu që pasi kemi gjetur personin me raportin e duhur, ndjekim pemën familjare të tij sipas gjeneratave, deri sa të na dalë një paraardhës që e ka raportin 1/1.

Edhe koha edhe kujtesa e kësaj zgjidhje janë $O(2^Q)$. Exponenciale! Kjo zgjidhje ngec te rasti i fundit.

Zgjidhja 2

```
from math import gcd

for t in range(int(input())):
    P, Q = [int(i) for i in input().split('/')]
    g = gcd(P, Q)
    P = P // g
```

```
    Q = Q // g

    # kontrollo nese Q eshte fuqi e dyshit
    if Q & (Q - 1) != 0:
        print('Case #{}: {}'.format(t+1, 'impossible'))
        continue

    gen = 0
    while P < Q:
        P = P * 2
        gen += 1

    print('Case #{}: {}'.format(t+1, gen))
```

Sqarime

Koha e kësaj zgjidhje është $O(ln(Q))$. Logaritmike! Kurse kujtesa e nevojshme është pothuajse zero (konstante)!

Uou! Ka zgjidhje e zgjidhje. Në krahasim me zgjidhjen e parë kjo është 1000 herë më e thjeshtë për tu shkruar dhe pafundësisht më efiçente. Por si arrihet në këtë zgjidhje dhe pse rezultati i saj është i saktë? Për më shumë sqarime shikoni këtë analizë: https://code.google.com/codejam/contest/3004486/dashboard#s=a&a=0

Siç ka thënë edhe Richard Feynman, një problem i vështirë mund të zgjidhet edhe me mjete të thjeshta, por kjo kërkon një sasi inteligjence të pafundme: https://youtu.be/xdIjYBtnvZU?t=200

Problemi 152

Kërkesa

Një tabelë (matricë) me përmasa **N**x**N** e ka çdo rresht të renditur nga e majta në të djathtë në rendin rritës, dhe çdo shtyllë të renditur nga lart poshtë në rendin rritës.

Çdo rresht dhe çdo shtyllë e matricës është shënuar në disa shirita letre (janë 2x**N** shirita të tillë, ku çdo shirit ka **N** numra të renditur). Vetëm se një nga këta shirita na ka humbur.

A mund të gjeni se cilët kanë qenë numrat në shiritin e humbur, duke ditur numrat që janë në shiritat e tjerë?

Referenca: https://code.google.com/codejam/contest/4304486/dashboard#s=p1&a=1

Shembull

```
$ cat input.txt
1
3
1 2 3
2 3 5
3 5 6
2 3 4
1 2 3

$ python3 prog.py < input.txt
Case #1: 3 4 6
```

Kemi vetëm 1 rast testimi, ku $\mathbf{N} = 3$. Më pas vijnë 2*3-1 rreshta me numra të renditur.

Zgjidhja

```
for t in range(int(input())):
    n = int(input())
    L = []
    for i in range(2*n - 1):
        L.extend([int(i) for i in input().split()])
    F = {}
    for i in L:
        F[i] = F.get(i, 0) + 1
    A = []
    for i in F.keys():
        if F[i] % 2 == 1:
            A.append(i)
    A.sort()
    print('Case #{}:'.format(t+1), *A)
```

Sqarime

Çdo numër që ndodhej te tabela është shënuar dy herë në shiritat e letrës, një herë sipas rreshtit dhe një herë sipas shtyllës. Kështu që çdo numër normalisht do shfaqej një numër çift herësh në shiritat e letrës. Më përjashtim të numrave që ndodhen në shiritin e humbur, të cilët shfaqen një numër tek herësh. Pasi të gjejmë ata numra që shfaqen një numër tek herësh, mjafton që ti rendisim, dhe këta do jenë numrat e shiritit të humbur.

Problemi 153

Kërkesa

Në një stivë me libra disa prej librave janë përmbys. Ne mund të kapim disa prej librave që janë në krye të stivës (ose të gjithë stivën) dhe ta kthejmë në krahun tjetër.

Le ta shënojmë një libër që është përbys me "-" dhe një libër që është në rregull me "+". Nqs stiva e librave, duke filluar nga kreu, është e tillë: −+-, atere një mënyrë për ti kthyer të gjithë librat në rregull mund të jetë kjo:

- Kapim 3 librat e sipërm dhe i kthejmë përmbys: -++-
- Kapim librin e parë dhe e kthejmë përmbys: +++-
- Kapim 3 librat e sipërm dhe i kthejmë përmbys: —-
- Kapim gjithë stivën dhe e kthejmë përmbys: ++++

Gjeni numrin më të vogël të lëvizjeve që duhen për të rregulluar një stivë të dhënë.

Referenca: https://code.google.com/codejam/contest/6254486/dashboard#s=a&a=0

Shembull

```
$ cat input.txt
5
-
-+
+-
+++
--+-

$ python3 prog.py < input.txt
Case #1: 1
Case #2: 1
Case #3: 2
Case #4: 0
Case #5: 3
```

Zgjidhja

```
for t in range(int(input())):
    books = input()
    ans = books.count('-+') + books.count('+-')
    if books.endswith('-'):
```

```
        ans += 1
    print('Case #{}: {}'.format(t+1, ans))
```

Sqarime

Strategjia më e thjeshtë dhe më e shkurtër është që gjithmonë të kapim dhe të kthejmë grupin me libra në krye të stivës që kanë të njëjtin drejtim. Kështu që mjafton të numërojmë se sa grupe të tilla kemi. Në fund, nqs të gjithë librat e stivës janë me krye poshtë, duhet të kthejmë të gjithë stivën.

Për një diskutim më të detajuar shiko: https://code.google.com/codejam/contest/6254486/dashboard#s=a&a=1

Problemi 154

Kërkesa

Jepet një listë L që ka numrat nga 1 deri në N. Në këtë listë bëhet ky veprim: Merren dy elementë X dhe Y të listës, fshihen nga lista, dhe pastaj shtohet në listë numri X + Y + X*Y. Nëse ky veprim bëhet N-1 herë në listë do ngelet vetëm 1 element. Sa është vlera më e madhe që mund të ketë ky element? Meqë rezultati mund të jetë një numër shumë i madh, jepni si përgjigje vetëm mbetjen e pjesëtimit të tij me $10^9 + 7$.

Referenca: https://www.codechef.com/problems/REDONE

Shembull

```
$ cat input.txt
2
1
2

$ python3 prog.py < input.txt
1
5
```

Zgjidhja

```
P = 10**9 + 7
N = 10**6
```

```
R = [0 for i in range(N+1)]
R[1] = 1
for i in range(2, N+1):
    R[i] = (R[i-1] + i + i*R[i-1]) % P

for _ in range(int(input())):
    print(R[int(input())])
```

Sqarime

Po ta bëjmë atë veprim në një listë me tre numra [x1, x2, x3] do shikojmë që rezultati është gjithmonë njëlloj dhe i pavarur nga rradha e veprimit. Kjo mund të përgjithësohet edhe për një listë me më shumë numra. Kjo na lejon që ti llogarisim paraprakisht në mënyrë efiçente të gjitha rezultatet e mundshme.

Gjithashtu mund të vihet re edhe se: $(X + Y + XY)\ \%\ P = X\ \%\ P + Y\ \%\ P + (XY)\ \%\ P$. Kjo na lejon që të punojmë vetëm dhe mbetjet e veprimeve dhe jo me numrat e plotë, që janë jashtëzakonisht të mëdhenj, dhe kjo e bën programin edhe më efiçent.

Problemi 155

Kërkesa

Ari dhe Riku po luajnë një lojë me fije shkrepsesh. Në fillim janë dy grumbuj me fije shkrepësesh me **N** dhe **M** fije, dhe Ari luan gjithmonë i pari. Në çdo lëvizje, lojtari që ka rradhën heq **X** fije nga njëri prej grumbujve, të tilla që X është shumëfish i numrit të fijeve në grumbullin tjetër. Fiton ai që boshatis njërin prej grumbujve (heq fijet e fundit).

Duke ditur numrin e fijeve në secilin pirg, si dhe që Ari luan i pari, dhe të dy luajnë në mënyrën më të mirë të mundshme (pa gabime), gjeni se kush e fiton lojën.

Referenca: https://www.codechef.com/problems/MATCHS

Shembull

```
$ cat input.txt
5
1 1
2 2
1 3
155 47
```

```
6 4

$ python3 prog.py < input.txt
Ari
Ari
Ari
Ari
Rich
```

Zgjidhja

```python
for _ in range(int(input())):
    a, b = [int(i) for i in input().split()]
    a, b = max(a,b), min(a,b)
    turn = 'Ari'
    while True:
        if a % b == 0 or a // b > 1:
            print(turn)
            break
        a, b = b, a % b
        turn = 'Rich' if turn == 'Ari' else 'Ari'
```

14 Shtojca 3

Problemi 201

Kërkesa

https://codingcompetitions.withgoogle.com/kickstart/round/0000000000051060/00000000000588f4

Zgjidhja

```
import sys

t = int(input())
wrong_answer = False
while t > 0 and not wrong_answer:
    t -= 1
    a, b = map(int, input().split())
    n = int(input())
    while not wrong_answer:
        g = (a + b) // 2 + 1
        print(g)
        sys.stdout.flush()
        r = input()
        if r == 'WRONG_ANSWER':
            wrong_answer = True
        elif r == 'TOO_SMALL':
            a = g
        elif r == 'TOO_BIG':
            b = g - 1
        else:
            break
```

Problemi 202

Kërkesa

https://codingcompetitions.withgoogle.com/codejam/round/00000000000000cb/0000000000007a30

Zgjidhja

```
import sys

M = 4
N = 5
board = {}

def go(x, y):
    if x == 0:    x += 1
    if x == M-1:  x -=1
    if y == 0:    y += 1
    if y == N-1:  y -= 1

    print(x+2, y+2)
    sys.stdout.flush()
    i, j = map(int, input().split())

    if j == -1 and j == -1:
        sys.exit()

    if i == 0 and j == 0:
        return True
    else:
        board[i-2, j-2] = True
        return False

def solve():
    # initialize the board
    for m in range(M):
        for n in range(N):
            board[m, n] = False

    # fill each cell of the board
    for m in range(M):
```

```
        for n in range(N):
            while not board[m, n]:
                if go(m, n): return

T = int(input())
for t in range(T):
    A = int(input())
    if A == 10:
        M = 3
        N = 4
    if A == 20:
        M = 4
        N = 5
    if A == 200:
        M = 14
        N = 15
    board = {}
    solve()
```

Problemi 203

Kërkesa

https://codingcompetitions.withgoogle.com/codejam/round/0000000000516b9/0000000000134c90

Zgjidhja

```
for t in range(int(input())):
    n = int(input())
    oponents = []
    for i in range(n):
        oponents.append(input())
    program = []
    pc = [0 for i in range(n)]
    while True:
        oponent_moves = set([oponents[i][pc[i]] for i in range(n) if oponents[i][pc[i]] != 'X'
        if len(oponent_moves) == 3:
            print('Case #{}: IMPOSSIBLE'.format(t+1))
            break
```

```
        elif len(oponent_moves) == 1:
            if 'P' in oponent_moves:
                program.append('S')
            elif 'R' in oponent_moves:
                program.append('P')
            elif 'S' in oponent_moves:
                program.append('R')
            print('Case #{}: {}'.format(t+1, ''.join(program)))
            break
        elif len(oponent_moves) == 2:
            if 'R' in oponent_moves and 'P' in oponent_moves:
                program.append('P')
                mx = 'R'
            elif 'R' in oponent_moves and 'S' in oponent_moves:
                program.append('R')
                mx = 'S'
            elif 'P' in oponent_moves and 'S' in oponent_moves:
                program.append('S')
                mx = 'P'
            for i in range(n):
                if oponents[i][pc[i]] == mx:
                    oponents[i] = ['X']
        # advance the program counter to the next statement
        for i in range(n):
            pc[i] += 1
            pc[i] %= len(oponents[i])
```

Problemi 204

Kërkesa

https://codingcompetitions.withgoogle.com/codejam/round/00000000000516b9/0000000000134e91

Zgjidhja

```
### To test this program with testing_tool.py first create a pipe (fifo) file:
### $ mkfifo pipe
### Then run it like this:
### $ python3 testing_tool.py 0 < pipe | python3 program.py > pipe
```

```
### $ python3 testing_tool.py 1 < pipe | python3 program.py > pipe

import sys

def ask(i):
    sys.stderr.write(">> {}\n".format(i+1))
    print(i + 1)
    sys.stdout.flush()
    reply = input()
    sys.stderr.write("<<<<< {}\n".format(reply))
    if reply == 'N':
        sys.exit()
    return reply

def answer(ans):
    sys.stderr.write(">> {}\n".format(ans))
    print(''.join(ans))
    sys.stdout.flush()
    reply = input()
    sys.stderr.write("<<<<< {}\n".format(reply))
    if reply == 'N':
        sys.exit()

letters = ['A', 'B', 'C', 'D', 'E']

T, F = [int(i) for i in input().split()]
for t in range(T):
    M = [['' for i in range(5)] for j in range(119)]
    ans = ['', '', '', '', '']

    # ask 119 questions and find the first char of the answer
    for i in range(119):
        M[i][0] = ask(i*5)
    freq = {}
    for c in [M[i][0] for i in range(119)]:
        freq[c] = freq.get(c, 0) + 1
    for c in freq.keys():
        if freq[c] == 23:
            ans[0] = c

    # ask 23 questions and find the second char of the answer
    for i in range(119):
        if M[i][0] != ans[0]:
```

```
            continue
        M[i][1] = ask(i*5 + 1)
    freq = {}
    for c in [M[i][1] for i in range(119)]:
        freq[c] = freq.get(c, 0) + 1
    for c in freq.keys():
        if freq[c] == 5:
            ans[1] = c

    # ask 5 questions and find the third char of the answer
    for i in range(119):
        if M[i][0] != ans[0] or M[i][1] != ans[1]:
            continue
        M[i][2] = ask(i*5 + 2)
    freq = {}
    for c in [M[i][2] for i in range(119)]:
        freq[c] = freq.get(c, 0) + 1
    for c in freq.keys():
        if freq[c] == 1:
            ans[2] = c

    # ask 1 more question and find the forth char
    for i in range(119):
        if M[i][0] != ans[0] or M[i][1] != ans[1] or M[i][2] != ans[2]:
            continue
        c1 = ask(i*5 + 3)
    for c in letters:
        if c != c1 and c not in ans:
            ans[3] = c
            break

    # find the last char
    for c in letters:
        if c not in ans:
            ans[4] = c
            break

    # send the answer
    answer(ans)
```