Start Competitive Programming!: Ace the USACO Bronze Competition

Updated and revised

Includes 2023-2024 problems

Zachi Baharav and **Daniel Zingaro**

Start Competitive Programming!: Ace the USACO Bronze Competition

Zachi Baharav and Daniel Zingaro

This book is for sale at http://leanpub.com/start_competitive_programming

This version was published on 2024-06-08

Contents

Letter to the student

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Letter to the parent

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Letter to the trained professional

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Acknowledgments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Part I. Preliminaries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 1. USACO Bronze

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

1.1. USACO Bronze FAQ

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

1.2. Solving and Submitting a USACO Problem

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

1.3. How to Work With This Book

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

1.4. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 2. Solving and Coding: Competition Specifics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.1. Reading and Analyzing a USACO Problem

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.1.1. Reading

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.1.2. Visualizing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.1.3. Algorithm

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.2. Coding Your Algorithm

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.2.1. Form and Style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.2.2. Coding Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.3. Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.3.1. Debugging In Practice (when you have the expected solution)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.3.2. Debugging In The Competition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.4. Using a Solution

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2.5. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 3. Complexity Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

3.1. Big O Notation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

3.2. Time complexity

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 3.1: Exact Group Size

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

3.3. Space complexity

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 3.2: Missing Number

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

3.4. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Part II. Core Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 4. Modeling and Simulation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

4.1. Modeling a Dynamic Process

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

4.1.1. Modeling Time Steps

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 4.1: Walk Around The Lake

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

4.1.2. Modeling Process Steps

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 4.2: Where Is The King?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

4.2. Modeling a Static Process

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 4.3: A Visit To The Mooseum

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

4.3. Modeling a Periodic Process

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 4.4: The Ferris Wheel

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

4.4. Simulation Acceleration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 4.5: Walking To The Opera House

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

4.5. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 5. Searching and Optimization

This chapter covers

- Recognizing search problems in the context of USACO.
- Solving search problems using an exhaustive search algorithm.
- Choosing a domain for performing the search.
- Enumerating the chosen domain.
- Accelerating an exhaustive search algorithm.
- Solving search problems using a greedy algorithm.

In search problems, as the name implies, we are searching for something. Search problems are a wide and intensive field of research and algorithms development in computer science. You are probably familiar with many of the applications of search algorithms: searching for a word in a document you are typing; searching for phrases on the web; searching for the shortest path to get from point A to point B. But searching problems have even broader applications, many of which involve searching in covert ways. For example, your autocorrect identifies the word closest to the one you tried to spell. Behind the scenes, it searches over all the possible words in its dictionary, refers to its knowledge of which words are used more frequently, and suggests a new word.

Often, search problems are called optimization problems. Optimization problems strive to achieve the best result possible for a certain condition. For example, consider the problem of designing a traffic intersection to allow for the maximum flow of cars. We can try giving certain green light time periods to the different directions (not at the same time!), then model how many cars will flow in each direction. Then, we can change those assigned green light times, and model the newly resulting flow of cars. In this problem, we are searching: trying to find the best allocation of green

light times, the one that would yield the maximum flow of cars. In optimization problems like this, we often find the solution by using search algorithms.

Search problems are very common in all levels of USACO. However, worry not: this chapter covers only the search algorithms needed at the Bronze level. You will learn more as you advance through the USACO levels.

The chapter map is described in figure 5.1. The most common searching algorithm at the Bronze level is the exhaustive search, also known as a complete search or brute-force search, described in section 5.1. This type of algorithm entails searching over all possible options. For example, the spellchecker might search over all possible words in the dictionary and decide which is the closest to your misspelled word. Doing an exhaustive search involves two main decisions. First, what "space" are we searching over? For example, are we searching over all the words in a particular British-spelling or American-spelling dictionary? This "space" to be searched is called the domain of the search and is discussed in section 5.2. Second, how do we know we searched all options? Or in other words, how do we order the elements in the domain? In the case of the autocorrect function, we can go over all the words in alphabetical order. In the case of the shortest path between two points on the map, where we need to consider many roads, the answer is not that clear. We need some kind of process for setting an order to search over all the elements. This kind of process is called enumeration and is discussed in section 5.3.

Section 5.4 describes ways to accelerate the search algorithm. This concern is worth exploring for the Bronze level, although it plays a more central role in the advanced levels of USACO. We close in section 5.5 with a discussion of a different search algorithm, the greedy algorithm. Unlike an exhaustive search, a greedy algorithm may reach a solution without examining all options. This may result in a significant reduction in execution time of the algorithm—but might fail to find the best solution. We will examine cases where a greedy algorithm works, as well as describe cases where it fails.

Throughout the chapter, we will encounter many search and optimization problems. One of the main goals of this chapter is to teach you to recognize a problem as a search problem, a skill that makes it much easier to devise an algorithm for a solution. Pay special attention as we highlight the key terms and concepts that indicate we're dealing with a search problem.

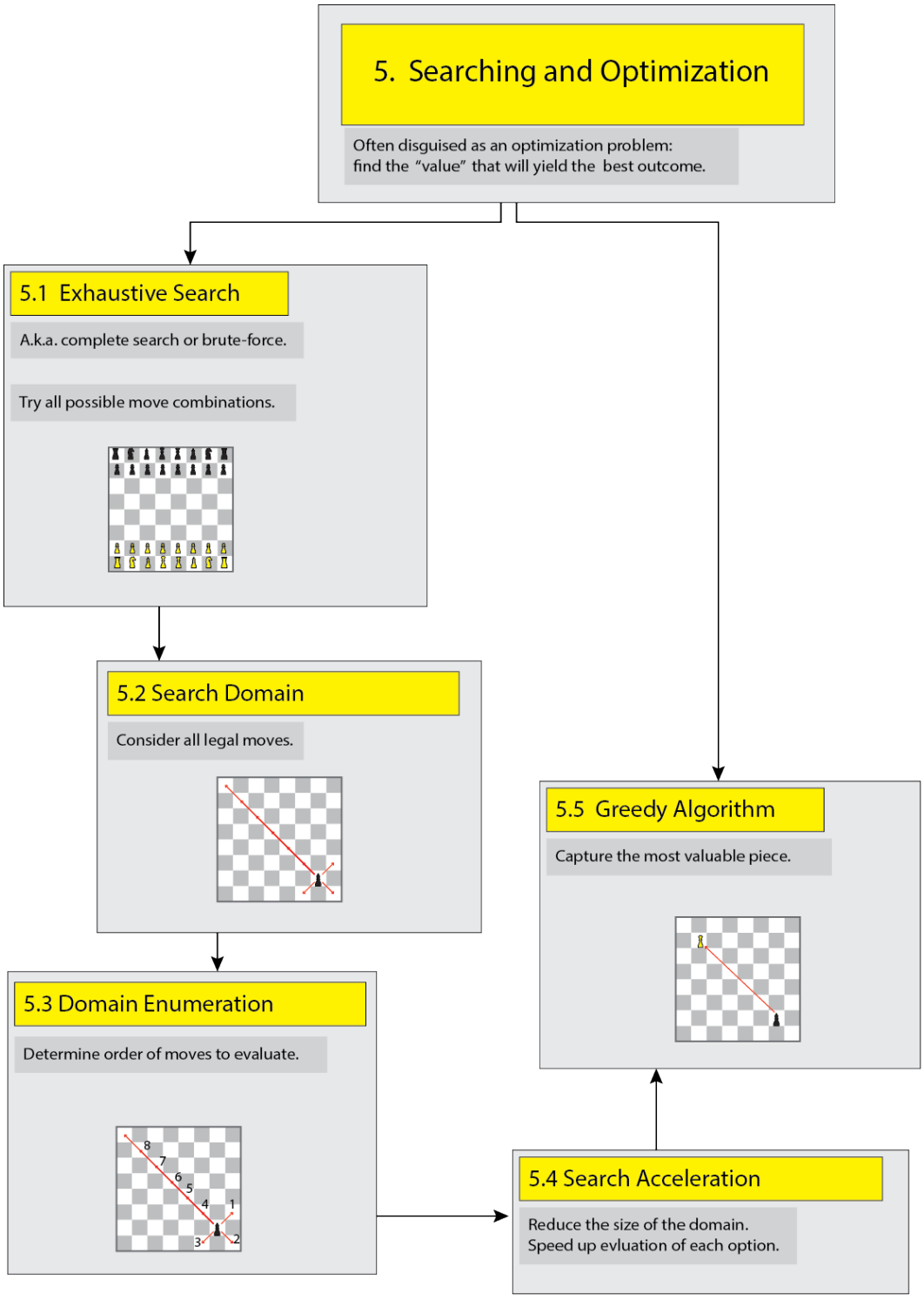

Figure 5.1 Searching and Optimization chapter map. We cover two types of search algorithms: Exhaustive search and Greedy algorithms.

5.1. Exhaustive Search

Coach B: Happy Tuesday, everyone. Today we will learn about exhaustive search algorithms. "Exhaustive search" is a very apt name for this method: it means we search over all possible options; it also alludes to the fact we, or at least the computer, is exhausted after doing this search. This is because it has to search over many, many options. Our first problem finds Bessie and her friends in Hawaii! Go ahead and read the problem, and we'll discuss it.

Problem 5.1: Tiki Torches

Bessie loves Waikiki Beach at night, with tiki torches illuminating the golden sand. But it's expensive to keep these torches lit, and the folks at the Office of Conservation have asked Bessie to help. Her job is to suggest one torch that can be removed that will cause minimum disruption. This torch cannot be the first or last torch in the line, as those are important to orient the guests.

Bessie noted in her notebook that there are N tiki torches, $2 < N < 10^5$, located along the beach in a straight line. A tiki torch location is indicated by a single integer, x_i.

Determine which torch can be removed such that the maximum distance between any two adjacent remaining torches is minimal.

Input Format

Two lines.

The first line contains a single number, N.

The second line contains N integers denoting the locations of the torches, $x_1, x_2, \ldots, x_N$.

It is given that $x_1 < x_2 < x_3 < \ldots < x_N$.

Output Format

One number, the location of the tiki torch that can be removed. If there are multiple locations that would yield the same result, output any of these locations (any will do).

Sample Input

```
6
1 8 10 16 20 23
```

Sample Output

```
20
```

If we remove the torch at location 20, the maximum distance between two adjacent torches is 7, which is the smallest possible.

DISCUSSION

The team reads the problem, then looks around at each other in puzzlement.

Coach B: I see there are a few confused looks around. So let's start from the very beginning. The problem asks us which tiki torch we should remove, right? And there are only so many tiki torches. This tells us this might be a search problem: we will need to search among all tiki torches and find the best one to remove.

Ryan: Thanks, Coach B. I got this part, but I'm still confused about what they actually are asking for. They ask for a maximum distance, but then they want it minimal. Am I reading it wrong?

Coach B: You read it right, Ryan. This is a very common phrasing in optimization problems. In optimization problems, we're searching for the best configuration. In our case, we are looking for the best tiki torch to remove. So let's try and see if we can figure this out by sketching out the problem. Since the problem doesn't totally make sense to us, we start with the parts that do make sense. I know it's hard, but let's try and get comfortable with the uncomfortable! Ryan, or anyone else, can you please draw the sample input for us? That will get us started.

TIP: Don't get stuck on the parts of the problem you don't understand. Start with the things you do understand, and see if you can figure out the rest.

Visualize it: Ryan walks to the board as the rest of the team joins. While Ryan draws the locations, Annie adds the tiki torches, as in figure 5.2.

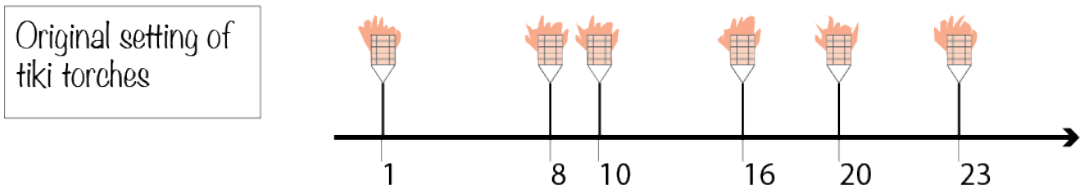

Figure 5.2 The initial placement of the tiki torches.

Coach B: Great. Love the tiki torches. Now, the problem talks about removing one torch. Let's pick one, remove it, and see how it looks.

Rachid: We can't remove the first or the last, so let's remove the one at location 8.

Rachid redraws the setting, as in figure 5.3, without the torch at location 8.

TIP: If possible, try not to erase, or write over, previous drawings. This will allow you to see the progress of your work, and how things evolve. Of course, some problems are too intricate to redraw every time. Find the right path for you, but keep in mind that having clear drawings helps with clear coding.

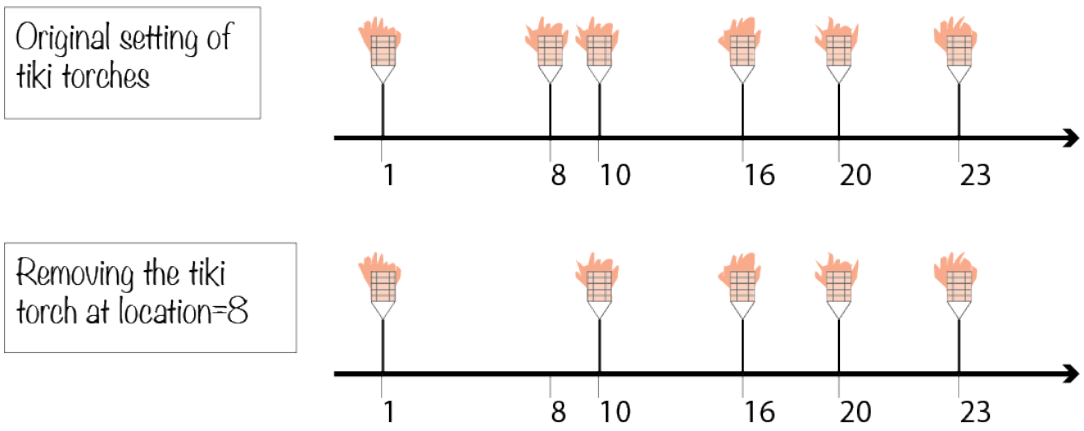

Figure 5.3 Removing tiki torch at location 8.

Coach B: Looks perfect. We are making progress. Now, what is the maximum distance between any two neighboring tiki torches?

Rachid writes the distance between all neighboring torches, as in figure 5.4.

Rachid: The maximum distance is 9, between the torches in location 1 and 10. I'm looking at 1 and 10 together because we removed a torch that was initially between them, the one at location 8.

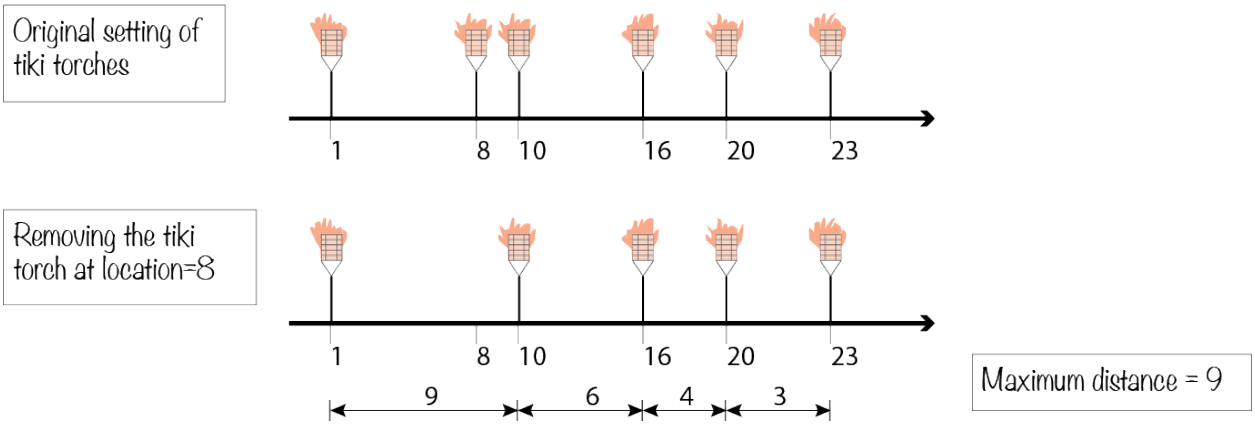

Figure 5.4 After removing one tiki torch, we examine the sketch to find the largest distance between two torches in this setting.

Annie: Oh, I think I get it. Now we need to try and remove other torches, and see what the maximum distance will be then. In the end, we take the minimum among those. Is this correct?

Coach B: Sounds right to me! Go ahead, the board is yours.

Annie and the team start drawing the different cases as in figure 5.5.

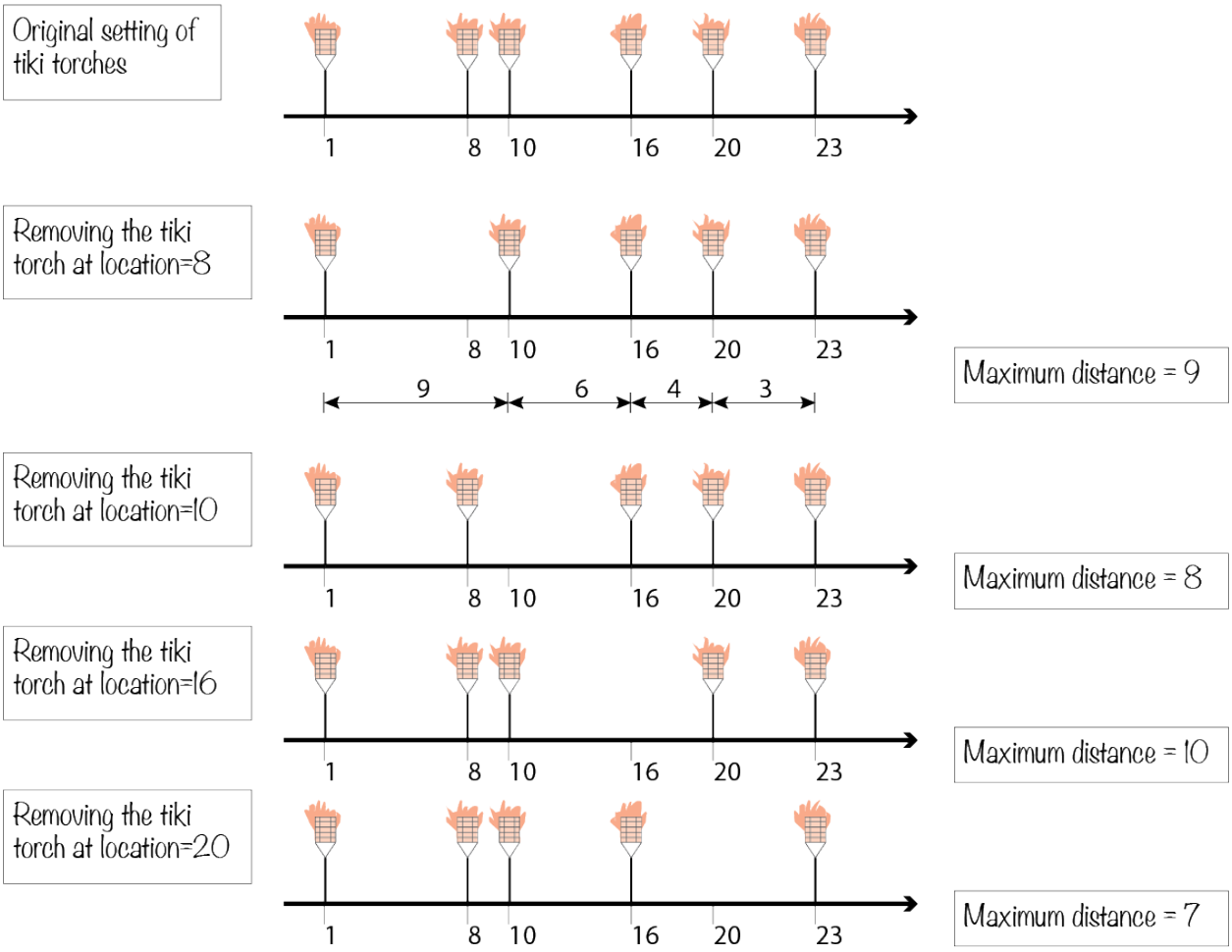

Figure 5.5 Examining all possible tiki torches to remove, and for each case indicating the resulting maximum distance between neighboring torches.

Mei: If we want to take the minimum among these, it's 7. That was when we removed the torch at location 20.

Coach B: And lo and behold, this is the answer they have in the problem for the sample input. Well done! Ryan, does it make sense now? Can you phrase it in your own words?

Ryan: I can try... So here is how I would phrase it: "Bessie wants to help the team and remove one torch. The problem is that any torch she removes makes a stretch of the beach a little less lighted. So her task is to remove one torch such that the resulting length of the beach without a tiki torch is the shortest. Help Bessie determine which tiki torch she should remove."

Mei: Wow, can we nominate Ryan to write USACO questions?

Coach B: I think the prerequisite is passing Bronze. But I agree, this was nicely phrased, Ryan! And I think this also helps us appreciate the use of the Minimum/Maximum phrasing in the original problem: It is much more succinct. The problem said, "Determine which torch can be removed such that the maximum distance between any two adjacent remaining torches is minimal." And we

needed to parse that as, "Look at all the possible resulting largest distances, and pick the smallest one of those." The Minimum/Maximum phrasing here will fit in many optimization problems, as you will see in the practice problems, whereas the tiki torches phrasing will fit only this specific case. But it was fun rephrasing it, so thanks again, Ryan!

TIP: When you see a phrase like this: "such that the maximum distance between any two adjacent remaining torches is minimal," it tells you this is probably an optimization problem. Specifically, this type of problem is called a minimax problem. Yes, all in one word, "minimax." A combination of "minimum" and "maximum".

ALGORITHM

Coach B: Now, any concerns about special cases, or are we ready to move to the algorithm?

Mei: Sharks are born ready. I am ready to give it a try.

Coach B: That's the attitude! Go Mei!

Mei takes the marker and writes listing 5.1.

Mei: First I loop over all the relevant tiki torches. Keep in mind we need to skip the first and last. For each one of these, I have a loop over all the remaining tiki torches, and calculate the distance between neighboring ones. I just calculate distance to the neighbor to the left, and keep the largest value among these.

Listing 5.1 Tiki Torches

```
int min_max_distance = INT_MAX;
int min_max_location;

for (int tiki_removed = 1; tiki_removed < N - 1; ++tiki_removed) {
    int max_dist = 0;
    int dist = 0;

    for (int i = 1; i < N; ++i) {
        if (i == tiki_removed) continue;
        if (i == tiki_removed + 1) {  // Are we to the right of the removed torch?
            // Yes: Our left neighbor is the previous one.
            dist = tiki_location[i] - tiki_location[i - 2];
        }
        else {
            // No: Distance from the left neighbor.
            dist = tiki_location[i] - tiki_location[i - 1];
        }
        max_dist = max(max_dist, dist);
    }
```

```
20
21      if (max_dist < min_max_distance) {
22          min_max_distance = max_dist;
23          min_max_location = tiki_removed;
24      }
25  }
```

Rachid: I can see why you did the first loop from 1 to $N-1$. This is because you wanted to avoid the first and last torches. But why do you skip torch number 0 in the inner loop? You go just from $i = 1$ up to N.

Mei: When I calculate distance, I calculate it from the current torch to its neighbor to the left. The very first torch does not have a neighbor to the left, so I am skipping it. Does this make sense?

Rachid: Oh, I see. Thanks.

Coach B: Very nice. Any comments?

The team seems to be happy with the code.

Coach B: Very well then. Actually, I have one more question before we move on from this problem. Any thoughts about what the time complexity of this algorithm might be?

Silence in the room. Complexity is, well, complex.

Ryan: I can try. If the number of tiki torches is N, then this is our base for talking about the order of the problem. Now, we are doing a nested loop over all the tiki torches, so that means we are going over N^2 cases. So that means our time complexity is $O(N^2)$. Is that... right?

Ryan trails off, uncertain.

Coach B: Very good, Ryan! The only thing missing is some more confidence in your answer! Can you say it with more confidence?

Ryan speaks louder.

Ryan: Our time complexity is $O(N^2)$!

The team shares a laugh.

Coach B: Right you are! Very nice. We will not try it now, but I would like to mention that there is a solution to this problem with a time complexity of only $O(N)$. I invite you to revisit this problem after we talk about accelerating search algorithms.

Mei: Wow, that sounds impossible. Can you please give us a hint at least?

Coach B: I really don't want to confuse you now, so here is what we'll do: I will leave the code, with comments and explanations, on the club's page. But this is a good point to emphasize: In Bronze, you do not necessarily have to find the most efficient algorithm in order to pass a problem. We will see that in some cases you are expected to accelerate your algorithm, but this is not always the case.

If you have a solution, and it passes all the test cases, you should move on to the next problem! So, in our case, you passed all the test cases, we can move on!

The team cheers.

Coach B: Okay. I believe this completes our first search problem! Very nice. In the process, we learned a common phrase used for Minimum/Maximum in optimization problems. We then did an exhaustive search: We tried to remove each and every one of the relevant tiki torches, and found the best one to remove. And to top it all off, Ryan helped us analyze the time complexity of this algorithm, with confidence. Well done!

The team starts packing, ready to bid farewell.

Coach B: I will put a few search problems on the club's page. I will also sprinkle in some hints, as usual. Oh, and I will also put the $O(N)$ solution if you want to see how it is done. See you next week!

TIP: If you are stuck for too long on a problem, you can always take a peek at the solution, and then write it on your own. It is better to get a big hint than to get discouraged. It's a learning process.

EPILOGUE

In exhaustive searches, we examine all possible options. This might be too time-consuming, but at the Bronze level it is often a valid approach. Still, even in exhaustive searches, there are opportunities to save on computation time. We will see ways to save computation time later in this chapter when we discuss acceleration.

VOCABULARY Corner: **OPTIMIZATION** is the process of bringing something to its best, or optimal, position. As a fun note, the words "optimize" and "optimization" grew out of the word "optimist." And Mei here is an optimist: a person with a hopeful and positive attitude, focused on the best of all possible options. Optimists always look on the bright side and expect the best things to happen. Like saving fuel costs while keeping Waikiki Beach well-lit and safe.

PRACTICE PROBLEMS

Hints and full solutions to the problems can be found on the club's page: http://www.usacoclub.com

1. USACO 2014 January Bronze Problem 1: Ski Course Design
   http://usaco.org/index.php?page=viewproblem2&cpid=376
   a. Can you pose the problem as a search question? What are you searching for?
   b. Hint: We are searching for the range of hill-heights that would not need any change.
   c. Hint: If you happen to know the lowest hill-height in the admissible range, can you find the cost of the ski course?

d. Big hint: You will search over the height of the lowest admissible hill. Given that, you can calculate the cost of the ski course. The lowest hill you should consider is the lowest hill height in the provided input, and the largest value to consider is the highest hill (possibly minus 17).

2. USACO 2016 Open Bronze Problem 1: Diamond Collector

   http://usaco.org/index.php?page=viewproblem2&cpid=639

   a. Can you see the similarity to the "Ski Course Design" problem? (2014 January Bronze Problem 1)

   b. Hint: If you happen to know the size of the smallest diamond you can display, can you determine how many diamonds will be presented?

3. USACO 2019 December Bronze Problem 1: Cow Gymnastics

   http://usaco.org/index.php?page=viewproblem2&cpid=963

   a. Arranging the input data in a two-dimensional array would make things easier.

   b. Then, it is exhaustive search over all possible pairs.

4. USACO 2019 December Bronze Problem 2: Where am I?

   http://usaco.org/index.php?page=viewproblem2&cpid=964

   a. Searching over strings.

   b. Exhaustive search over all substrings would work within time.

5.2. Search Domain

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 5.2: Bessie Searches Seashells by the Seashore

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

5.3. Domain Enumeration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 5.3: Crossing Volcanoes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

5.4. Search Acceleration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 5.4: Luaus and Leis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

5.5. Greedy Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 5.5: Kayaking

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Sample Problem: Knapsack Problem (we will use a piece of luggage instead)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

5.6. Summary

- **Search problems** can be hard to identify. They come in many shapes and forms, and often are presented as optimization problems. In optimization problems, we search for a parameter of a process to achieve the best outcome.
- To identify a search problem, try asking yourself the following questions.
  - Could you try different values, and see which one works best? If it seems possible, then maybe you can search over all these values.
  - Would an oracle allow you to solve the problem? That is, if someone appeared, poof, to magically reveal to you the value of the parameter, would you be able to evaluate how good this value is? If yes, then you can build an exhaustive search going over all possible values from the oracle.
  - What's the first decision you'd need to make to solve the problem? For example, taking the heaviest cow. If you kept making this same type of decision again and again, would that lead you to the solution? If yes, maybe a greedy algorithm is possible.

- At the Bronze level, we solve search problems with two main types of algorithms: exhaustive searches and greedy algorithms.
- **Exhaustive searches** evaluate all possible options and choose the best one.
  - Determine the domain of the problem. These are the values you will search over.
  - Enumerate the domain. How are you going to go over the domain one element at a time?
- **Accelerating exhaustive searches**. We do this in two ways:
  - Choose a smaller domain. This way, you get to examine fewer options.
  - Accelerate the evaluation of each option.
- **Greedy algorithms** are based on making simple and quick decisions at each step.
  - They are usually very fast.
  - They don't necessarily guarantee an optimal solution (they work only for some problems!).
  - You may get a better result with a greedy algorithm if you design a new one using a different greedy decision.

Chapter 6. Geometry Concepts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.1. One Dimension: Lines

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.1.1. Location, Length and Distance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 6.1: Walk or Bus?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.1.2. Two Line Segments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 6.2: Golden Gate Bridge Patrol

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.2. Two Dimensions: Rectangles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.2.1. Location and Area

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 6.3: Going Around the Fence

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.2.2. Two Rectangles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 6.4: Two Blankets for the Picnic

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.3. Beyond Ninety Degrees

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.3.1. Circles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 6.5: Seats Around the Arena

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.3.2. General Shapes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 6.6: Path Around the Lake

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

6.4. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 7. Strings

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.1. Strings as Sequences of Characters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.1.1. Representing Characters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.1.2. Problems with Characters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 7.1: Double Doors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.2. Strings as Words

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 7.2: Arrange by Age

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.3. Strings as Objects

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.3.1. String Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 7.3: Bessst Bracelet

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.3.2. Lexicographic order

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

7.4. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 8. Ad Hoc Problems and Advanced Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

8.1. The Forward-Backward Technique

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 8.1: Double Doors Fixing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

8.2. Focusing on Significant Events

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 8.2: Sharks and Moonnows

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

8.3. Trees

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Problem 8.3: The Restaurant at the End of the Farm

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

8.4. Dictionaries and Dynamic Arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

8.5. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Part III. Competition Day and Beyond

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 9. Competition Day

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

9.1. A Week Before

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

9.2. The Competition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

9.3. Post Competition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

9.4. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Chapter 10. Beyond USACO Bronze

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

10.1. Silver and Beyond

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

10.2. Solving your first USACO Silver Problem

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

10.3. Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Part IV. Appendix

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Appendix A. List of All USACO Bronze Problems

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

USACO problems

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2012-2013 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2013-2014 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2014-2015 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2015-2016 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2016-2017 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2017-2018 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2018-2019 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2019-2020 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2020-2021 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2021-2022 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2022-2023 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

2023-2024 Season

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Codeforces problems

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

CSES Problems

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

Appendix B. Practice Beyond USACO

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

B.1. Online Guides and Live Coaching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

B.2. Online Practicing and Competing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.

B.3. BOOKS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/start_competitive_programming.