flex in interviews

A Guide to grinding leetcode

O RLY?

theabbie

A Guide to grinding Leetcode

By Abhishek Chaudhary

Leetcode is famous for being a perfect platform for practicing coding problems and to master coding interviews, unlike others, it is not for competitive programming, this guide will help you to get started with Leetcode without losing hope too early.

A Guide to grinding Leetcode

Follow a list

Biggest reason why people don't find DSA interesting is because they are unable to discover good problems which are easier to solve, simply solving popular questions with no track of questions will bore you very quick. Even worse, you might try solving a really hard problem and lose motivation when it wasn't that hard, you just had to get a grip on it. That's why it's absolutely necessary to follow a list of questions, that way you won't have issues discovering new questions.

Solving problems in right order is very important,

you might see question marked easy which isn't actually easy, the solution will be small, but sometimes, it isn't easy to come up with that solution if you haven't done simpler version of it, thus, it will be demotivating,

Blind 75 Leetcode Questions

This is an awesome list which is asked in interviews and is ordered by actual level of difficulty with prerequisites coming before harder questions, if you follow this, you'll feel interested, once you have done most of this, do problems in “similar questions" section below each problem till you master that category.

Once you feel confident, you can use this,

Leetcode Patterns

and solve problems by category, this will help you master a data structure or some algorithm.

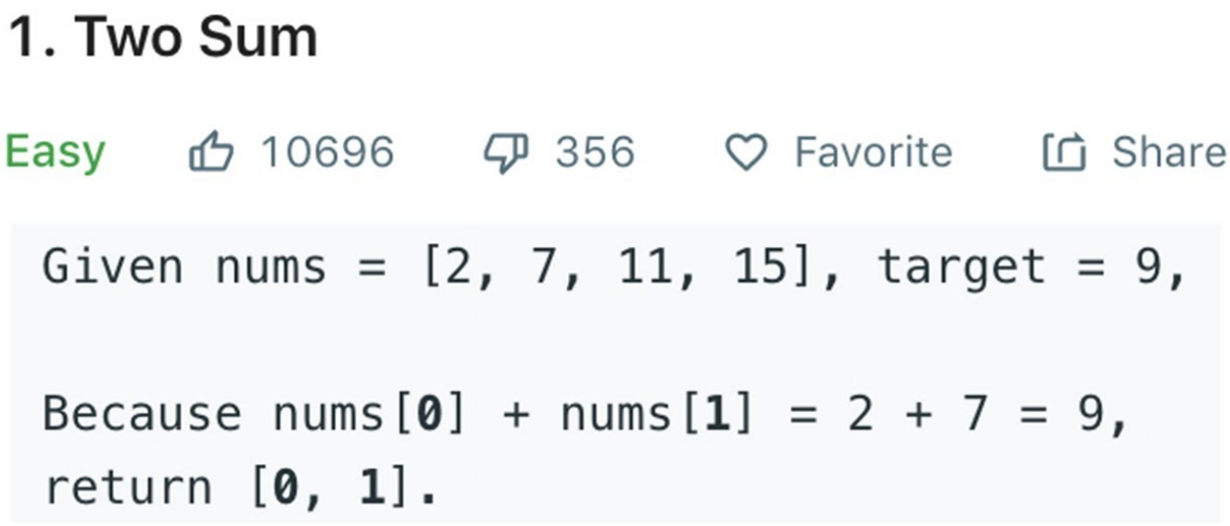
don't get afraid by “hard" questions, there is no hard problem which can't be broken up, try to break it, you might not be able to solve it but you'll convert it to much shorter set of problems which can be solved with some practice.

Thinking abstract and looking at bigger picture is very important, try to convert it to a standard problem. Leetcode is addictive if you improve gradually, try it.

Avoid looking at solutions easily

It's not bad to look at solutions, afterall, you can't know everything and learning is necessary, however, looking at solution just after few minutes of brainstorming is bad, you have to give your absolute best and try every possible "inefficient" solutions you could come up with.

First phase is to figure out what Algorithm and data structure will be used, if you are able to determine what data structure will be used, you can check the **Related Topics** section to verify if your assumption was correct, and if after few minutes you can't figure it out, you should still check the data structure that will be used and then try to figure out how and where it will be used in given problem.



Classic Two Sum Problem

If you are able to come up with a solution which works correctly, just isn't the best one, that's still a success, coming up with a brute force solution is a bare minimum in an interview.

You can try improving the brute force solution by using some optimizations, that might not lead you to the optimal solution, but improving a solution is a great skill. After spending an hour, if you can't solve the problem, you should usnderstand that you just aren't well versed with the given algorithm and should try solving related problems with that data structure and understand how it works.

You should avoid looking at solution, a solution you made yourself will help you much more, you should abandon the question and maybe revisit in future when you have some experience with that data strucure. That way you can also track if you made some progress with that technique and if you could solve a new problem given to you in an interview,

Interview Questions don't come with hints

One thing to remember is that Interview questions won't tell you what data structure will be used for the problem. That's something you can only master with practice, the patterns and requirements of problems determine what's going to be used.

There is no substitute for practice, reading about algorithms will sure improve your range of thinking, but practice is what will help you master it.

Be Consistent

This goes without saying that practice needs consistency, simply overdoing once and abandoning for months will be destructive, it doesn't take much to take out some time everyday for Leetcode, as far as discovering questions is a concern, you can use Daily Challenges to keep the consistency and maybe also earn Leetcode coins which might buy you a Leetcode T-Shirt one day.

Turn Demotivation into learning opportunity

There will be times when you can't solve a problem despite all efforts, that's very common and bound to happen, but some question being too hard is not something that should demotivate you, every question is a learning opportunity, you can always learn it. Demotivation should be avoided and that's only possible if you have confidence in yourself and will to learn as much as you can.

Participate in contests

Eventhough Leetcode isn't a competitive programming platform, there are contests which allow you to try out brand neew problems and even compete with others. They have categories of 1 easy, 2 Medium and 1 Hard, and solving 3 is more than enough. Once you have enough confidence on your problem solving ability, these contests will help you gain interview experience as they don't have any hints and solutions aren't available during contest. This is pretty close to a real interview experience where time is limited.

Keep Hustling

Leetcode is an addiction and soon you'll fall in love with it, all you need to do is start, there is only one good time to start anything great, NOW, just do it and you'll sure be satisfied with your decision and be proud of yourself. That's all, It's never too early and never too late.

10 Ways to contact me

- Mail
- Instagram DM
- Telegram
- Whatsapp
- Linkedin
- Twitter
- Snapchat
- ICQ
- Facebook

- +918928412138

My Leetcode Journey

You can find my Leetcode profile leetcode.com/theabbie and I have also posted all my leetcode solutions on GitHub theabbie/leetcode

All my Leetcode Solutions

- 01-matrix.py
- 1-bit-and-2-bit-characters.py
- 132-pattern.py
- 2-keys-keyboard.py
- 3sum-closest.py
- 3sum-with-multiplicity.py
- 3sum.py
- 4sum-ii.py
- a-number-after-a-double-reversal.py
- add-binary.py
- add-digits.py
- add-strings.py
- add-to-array-form-of-integer.cpp
- add-two-numbers-ii.py
- add-two-numbers.py
- alert-using-same-key-card-three-or-more-times-in-a-one-hour-period.py
- all-ancestors-of-a-node-in-a-directed-acyclic-graph.py
- all-elements-in-two-binary-search-trees.py
- all-paths-from-source-to-target.py
- all-possible-full-binary-trees.py
- angle-between-hands-of-a-clock.py
- append-k-integers-with-minimal-sum.py
- arithmetic-slices.py
- arithmetic-subarrays.py
- arranging-coins.py
- array-nesting.py
- array-partition-i.py
- assign-cookies.py
- asteroid-collision.py

- available-captures-for-rook.py
- average-of-levels-in-binary-tree.py
- average-salary-excluding-the-minimum-and-maximum-salary.py
- backspace-string-compare.py
- balance-a-binary-search-tree.py
- balanced-binary-tree.py
- base-7.py
- baseball-game.py
- basic-calculator-ii.py
- basic-calculator.py
- beautiful-array.py
- best-time-to-buy-and-sell-stock.py
- binary-gap.py
- binary-number-with-alternating-bits.py
- binary-prefix-divisible-by-5.py
- binary-search-tree-iterator.py
- binary-search-tree-to-greater-sum-tree.py
- binary-search.py
- binary-tree-inorder-traversal.py
- binary-tree-level-order-traversal-ii.py
- binary-tree-level-order-traversal.py
- binary-tree-maximum-path-sum.py
- binary-tree-paths.py
- binary-tree-postorder-traversal.py
- binary-tree-preorder-traversal.py
- binary-tree-pruning.py
- binary-tree-tilt.py
- binary-tree-zigzag-level-order-traversal.py
- boats-to-save-people.py
- broken-calculator.py
- build-array-from-permutation.py
- bulb-switcher.py
- bulls-and-cows.py
- camelcase-matching.py
- can-make-arithmetic-progression-from-sequence.py
- can-place-flowers.py
- capitalize-the-title.py

- car-pooling.py
- cells-in-a-range-on-an-excel-sheet.py
- cells-with-odd-values-in-a-matrix.py
- champagne-tower.py
- check-completeness-of-a-binary-tree.py
- check-if-all-1s-are-at-least-length-k-places-away.py
- check-if-all-as-appears-before-all-bs.py
- check-if-all-characters-have-equal-number-of-occurrences.py
- check-if-array-is-sorted-and-rotated.py
- check-if-binary-string-has-at-most-one-segment-of-ones.py
- check-if-every-row-and-column-contains-all-numbers.py
- check-if-it-is-a-good-array.py
- check-if-it-is-a-straight-line.py
- check-if-n-and-its-double-exist.py
- check-if-number-is-a-sum-of-powers-of-three.py
- check-if-numbers-are-ascending-in-a-sentence.py
- check-if-the-sentence-is-pangram.py
- check-if-two-string-arrays-are-equivalent.py
- check-if-word-is-valid-after-substitutions.py
- check-whether-two-strings-are-almost-equivalent.py
- climbing-stairs.py
- clone-graph.py
- closest-divisors.py
- clumsy-factorial.py
- coin-change.py
- combination-sum.py
- combinations.py
- combine-two-tables.sql
- compare-version-numbers.py
- complement-of-base-10-integer.cpp
- complex-number-multiplication.py
- concatenation-of-array.py
- consecutive-characters.py
- consecutive-numbers-sum.py
- construct-binary-search-tree-from-preorder-traversal.py
- construct-binary-tree-from-inorder-and-postorder-traversal.py
- construct-binary-tree-from-preorder-and-inorder-traversal.py

- construct-binary-tree-from-preorder-and-postorder-traversal.py
- construct-string-from-binary-tree.py
- construct-the-rectangle.py
- container-with-most-water.py
- contains-duplicate-ii.py
- contains-duplicate-iii.py
- contains-duplicate.py
- contiguous-array.py
- continuous-subarray-sum.py
- convert-1d-array-into-2d-array.py
- convert-binary-number-in-a-linked-list-to-integer.py
- convert-bst-to-greater-tree.py
- convert-integer-to-the-sum-of-two-no-zero-integers.py
- convert-sorted-array-to-binary-search-tree.py
- convert-sorted-list-to-binary-search-tree.py
- copy-list-with-random-pointer.py
- count-all-valid-pickup-and-delivery-options.py
- count-and-say.py
- count-artifacts-that-can-be-extracted.py
- count-common-words-with-one-occurrence.py
- count-complete-tree-nodes.py
- count-elements-with-strictly-smaller-and-greater-elements.py
- count-good-nodes-in-binary-tree.py
- count-good-triplets.py
- count-hills-and-valleys-in-an-array.py
- count-items-matching-a-rule.py
- count-negative-numbers-in-a-sorted-matrix.py
- count-number-of-nice-subarrays.py
- count-number-of-pairs-with-absolute-difference-k.py
- count-number-of-teams.py
- count-of-matches-in-tournament.py
- count-of-smaller-numbers-after-self.py
- count-sorted-vowel-strings.py
- count-special-quadruplets.py
- count-square-sum-triples.py
- count-the-hidden-sequences.py
- count-the-number-of-consistent-strings.py

- counting-bits.py
- course-schedule-ii.py
- course-schedule.py
- cousins-in-binary-tree.py
- crawler-log-folder.py
- create-binary-tree-from-descriptions.py
- create-target-array-in-the-given-order.py
- custom-sort-string.py
- daily-temperatures.py
- data-stream-as-disjoint-intervals.py
- decode-the-slanted-ciphertext.py
- decode-ways.py
- decode-xored-array.py
- decompress-run-length-encoded-list.py
- deepest-leaves-sum.py
- defanging-an-ip-address.py
- delete-and-earn.py
- delete-characters-to-make-fancy-string.py
- delete-columns-to-make-sorted.py
- delete-leaves-with-a-given-value.py
- delete-node-in-a-bst.py
- delete-node-in-a-linked-list.py
- delete-operation-for-two-strings.py
- delete-the-middle-node-of-a-linked-list.py
- design-a-stack-with-increment-operation.py
- design-add-and-search-words-data-structure.py
- design-hashmap.py
- design-hashset.py
- design-parking-system.py
- detect-capital.py
- determine-if-string-halves-are-alike.py
- diagonal-traverse.py
- diameter-of-binary-tree.py
- distant-barcodes.py
- distribute-candies.py
- divide-a-string-into-groups-of-size-k.py
- divide-array-into-equal-pairs.py

- divide-two-integers.py
- divisor-game.py
- dungeon-game.py
- duplicate-zeros.py
- edit-distance.py
- employee-importance.py
- encode-and-decode-tinyurl.py
- encrypt-and-decrypt-strings.py
- evaluate-division.py
- evaluate-reverse-polish-notation.py
- evaluate-the-bracket-pairs-of-a-string.py
- even-odd-tree.py
- excel-sheet-column-number.py
- factorial-trailing-zeroes.py
- fibonacci-number.py
- final-value-of-variable-after-performing-operations.py
- find-a-corresponding-node-of-a-binary-tree-in-a-clone-of-that-tree.py
- find-all-anagrams-in-a-string.py
- find-all-duplicates-in-an-array.py
- find-all-k-distant-indices-in-an-array.py
- find-all-lonely-numbers-in-the-array.py
- find-all-numbers-disappeared-in-an-array.py
- find-all-possible-recipes-from-given-supplies.py
- find-and-replace-in-string.py
- find-and-replace-pattern.py
- find-bottom-left-tree-value.py
- find-center-of-star-graph.py
- find-common-characters.py
- find-elements-in-a-contaminated-binary-tree.py
- find-first-and-last-position-of-element-in-sorted-array.py
- find-first-palindromic-string-in-the-array.py
- find-greatest-common-divisor-of-array.py
- find-k-closest-elements.py
- find-k-pairs-with-smallest-sums.py
- find-largest-value-in-each-tree-row.py
- find-lucky-integer-in-an-array.py
- find-median-from-data-stream.py

- find-mode-in-binary-search-tree.py
- find-n-unique-integers-sum-up-to-zero.py
- find-nearest-point-that-has-the-same-x-or-y-coordinate.py
- find-numbers-with-even-number-of-digits.py
- find-palindrome-with-fixed-length.py
- find-peak-element.py
- find-pivot-index.py
- find-players-with-zero-or-one-losses.py
- find-positive-integer-solution-for-a-given-equation.py
- find-right-interval.py
- find-smallest-letter-greater-than-target.py
- find-subsequence-of-length-k-with-the-largest-sum.py
- find-target-indices-after-sorting-array.py
- find-the-difference-of-two-arrays.py
- find-the-difference.py
- find-the-distance-value-between-two-arrays.py
- find-the-duplicate-number.py
- find-the-highest-altitude.py
- find-the-kth-largest-integer-in-the-array.py
- find-the-middle-index-in-array.py
- find-the-minimum-and-maximum-number-of-nodes-between-critical-points.py
- find-the-town-judge.py
- find-triangular-sum-of-an-array.py
- find-unique-binary-string.py
- find-words-that-can-be-formed-by-characters.py
- finding-3-digit-even-numbers.py
- first-bad-version.py
- first-missing-positive.py
- first-unique-character-in-a-string.py
- fizz-buzz.py
- flatten-a-multilevel-doubly-linked-list.py
- flatten-binary-tree-to-linked-list.py
- flip-equivalent-binary-trees.py
- flipping-an-image.py
- flood-fill.py
- flower-planting-with-no-adjacent.py
- fraction-addition-and-subtraction.py

- fruit-into-baskets.py
- game-of-life.py
- generate-parentheses.py
- generate-random-point-in-a-circle.py
- get-maximum-in-generated-array.py
- goal-parser-interpretation.py
- goat-latin.py
- gray-code.py
- greatest-sum-divisible-by-three.py
- grid-game.py
- group-anagrams.py
- group-the-people-given-the-group-size-they-belong-to.py
- guess-number-higher-or-lower.py
- h-index-ii.py
- h-index.py
- hamming-distance.py
- happy-number.py
- height-checker.py
- house-robber-iii.py
- house-robber.py
- how-many-numbers-are-smaller-than-the-current-number.py
- image-smoother.py
- implement-rand10-using-rand7.py
- implement-stack-using-queues.py
- implement-strstr.py
- implement-trie-prefix-tree.py
- increasing-order-search-tree.py
- increasing-subsequences.py
- increasing-triplet-subsequence.py
- indexold.html
- insert-delete-getrandom-o1-duplicates-allowed.py
- insert-delete-getrandom-o1.py
- insert-interval.py
- insert-into-a-binary-search-tree.py
- integer-break.py
- integer-replacement.py
- integer-to-roman.py

- intersection-of-two-arrays-ii.py
- intersection-of-two-arrays.py
- intersection-of-two-linked-lists.py
- invert-binary-tree.py
- is-graph-bipartite.py
- is-subsequence.py
- island-perimeter.py
- isomorphic-strings.py
- jewels-and-stones.py
- jump-game-ii.py
- jump-game-iii.py
- jump-game.py
- k-closest-points-to-origin.py
- k-diff-pairs-in-an-array.py
- k-th-smallest-prime-fraction.py
- k-th-symbol-in-grammar.py
- keep-multiplying-found-values-by-two.py
- keyboard-row.py
- keys-and-rooms.py
- kids-with-the-greatest-number-of-candies.py
- koko-eating-bananas.py
- kth-distinct-string-in-an-array.py
- kth-largest-element-in-a-stream.py
- kth-largest-element-in-an-array.py
- kth-missing-positive-number.py
- kth-smallest-element-in-a-bst.py
- kth-smallest-element-in-a-sorted-matrix.py
- largest-number-after-digit-swaps-by-parity.py
- largest-number-at-least-twice-of-others.py
- largest-number.py
- largest-rectangle-in-histogram.py
- last-stone-weight.py
- last-substring-in-lexicographical-order.py
- leaf-similar-trees.py
- length-of-last-word.py
- letter-combinations-of-a-phone-number.py
- letter-tile-possibilities.py

- lexicographical-numbers.py
- license-key-formatting.py
- linked-list-cycle-ii.py
- linked-list-cycle.py
- linked-list-random-node.py
- long-pressed-name.py
- longer-contiguous-segments-of-ones-than-zeros.py
- longest-common-prefix.py
- longest-common-subsequence.py
- longest-consecutive-sequence.py
- longest-happy-prefix.py
- longest-happy-string.py
- longest-harmonious-subsequence.py
- longest-increasing-subsequence.py
- longest-palindrome-by-concatenating-two-letter-words.py
- longest-palindrome.py
- longest-palindromic-subsequence.py
- longest-subarray-of-1s-after-deleting-one-element.py
- longest-substring-with-at-least-k-repeating-characters.py
- longest-substring-without-repeating-characters.cpp
- longest-substring-without-repeating-characters.py
- longest-uncommon-subsequence-i.py
- longest-univalue-path.py
- longest-word-in-dictionary-through-deleting.py
- longest-word-in-dictionary.py
- loud-and-rich.py
- lowest-common-ancestor-of-a-binary-search-tree.py
- lowest-common-ancestor-of-a-binary-tree.py
- lowest-common-ancestor-of-deepest-leaves.py
- lucky-numbers-in-a-matrix.py
- majority-element-ii.py
- majority-element.py
- make-the-string-great.py
- matrix-cells-in-distance-order.py
- matrix-diagonal-sum.py
- max-area-of-island.py
- max-consecutive-ones.py

- max-points-on-a-line.py
- maximal-network-rank.py
- maximal-square.py
- maximize-distance-to-closest-person.py
- maximize-number-of-subsequences-in-a-string.py
- maximum-69-number.py
- maximum-binary-tree-ii.py
- maximum-binary-tree.py
- maximum-candies-allocated-to-k-children.py
- maximum-depth-of-binary-tree.py
- maximum-depth-of-n-ary-tree.py
- maximum-difference-between-increasing-elements.py
- maximum-difference-between-node-and-ancestor.py
- maximum-distance-between-a-pair-of-values.py
- maximum-frequency-stack.py
- maximum-gap.py
- maximum-length-of-pair-chain.py
- maximum-length-of-repeated-subarray.py
- maximum-level-sum-of-a-binary-tree.py
- maximum-nesting-depth-of-the-parentheses.py
- maximum-number-of-balloons.cpp
- maximum-number-of-coins-you-can-get.py
- maximum-number-of-weeks-for-which-you-can-work.py
- maximum-number-of-words-found-in-sentences.py
- maximum-number-of-words-you-can-type.py
- maximum-population-year.py
- maximum-product-after-k-increments.py
- maximum-product-difference-between-two-pairs.py
- maximum-product-of-the-length-of-two-palindromic-subsequences.py
- maximum-product-of-three-numbers.py
- maximum-product-of-word-lengths.py
- maximum-product-subarray.py
- maximum-subarray.py
- maximum-swap.py
- maximum-twin-sum-of-a-linked-list.py
- maximum-width-of-binary-tree.py
- maximum-xor-of-two-numbers-in-an-array.py

- median-of-two-sorted-arrays.py
- merge-intervals.py
- merge-nodes-in-between-zeros.py
- merge-sorted-array.py
- merge-two-binary-trees.py
- merge-two-sorted-lists.py
- middle-of-the-linked-list.py
- min-cost-climbing-stairs.py
- min-cost-to-connect-all-points.py
- min-stack.py
- minimize-deviation-in-array.py
- minimize-maximum-pair-sum-in-array.py
- minimize-result-by-adding-parentheses-to-expression.py
- minimum-absolute-difference-in-bst.py
- minimum-absolute-difference.py
- minimum-add-to-make-parentheses-valid.py
- minimum-bit-flips-to-convert-number.py
- minimum-cost-for-tickets.py
- minimum-cost-of-buying-candies-with-discount.py
- minimum-cost-to-move-chips-to-the-same-position.py
- minimum-deletions-to-make-array-beautiful.py
- minimum-deletions-to-make-string-balanced.py
- minimum-difference-between-highest-and-lowest-of-k-scores.py
- minimum-distance-between-bst-nodes.py
- minimum-distance-to-the-target-element.py
- minimum-domino-rotations-for-equal-row.py
- minimum-index-sum-of-two-lists.py
- minimum-insertion-steps-to-make-a-string-palindrome.py
- minimum-moves-to-reach-target-score.py
- minimum-number-of-operations-to-convert-time.py
- minimum-number-of-operations-to-move-all-balls-to-each-box.py
- minimum-number-of-swaps-to-make-the-string-balanced.py
- minimum-number-of-vertices-to-reach-all-nodes.py
- minimum-operations-to-convert-number.py
- minimum-operations-to-make-array-equal.py
- minimum-operations-to-make-the-array-increasing.py
- minimum-operations-to-reduce-x-to-zero.py

- minimum-path-sum.py
- minimum-remove-to-make-valid-parentheses.py
- minimum-size-subarray-sum.py
- minimum-sum-of-four-digit-number-after-splitting-digits.py
- minimum-time-to-complete-trips.py
- minimum-time-to-type-word-using-special-typewriter.py
- minimum-white-tiles-after-covering-with-carpets.py
- miscellaneous
- missing-number.py
- monotonic-array.py
- most-common-word.py
- most-frequent-number-following-key-in-an-array.py
- most-frequent-subtree-sum.py
- move-zeroes.py
- multiply-strings.py
- n-ary-tree-level-order-traversal.py
- n-ary-tree-postorder-traversal.py
- n-ary-tree-preorder-traversal.py
- n-queens-ii.py
- n-queens.py
- n-repeated-element-in-size-2n-array.py
- n-th-tribonacci-number.py
- network-delay-time.py
- next-greater-element-i.py
- next-greater-element-ii.py
- next-greater-element-iii.py
- next-greater-node-in-linked-list.py
- next-permutation.py
- nim-game.py
- number-complement.py
- number-of-1-bits.py
- number-of-equivalent-domino-pairs.py
- number-of-good-pairs.py
- number-of-islands.py
- number-of-laser-beams-in-a-bank.py
- number-of-pairs-of-strings-with-concatenation-equal-to-target.py
- number-of-recent-calls.py

- number-of-rectangles-that-can-form-the-largest-square.py
- number-of-segments-in-a-string.py
- number-of-smooth-descent-periods-of-a-stock.py
- number-of-steps-to-reduce-a-number-in-binary-representation-to-one.py
- number-of-steps-to-reduce-a-number-to-zero.py
- number-of-strings-that-appear-as-substrings-in-word.py
- number-of-ways-to-divide-a-long-corridor.py
- number-of-ways-to-select-buildings.py
- occurrences-after-bigram.py
- odd-even-linked-list.py
- open-the-lock.py
- palindrome-linked-list.py
- palindrome-number.py
- palindrome-partitioning-ii.py
- palindrome-partitioning.py
- palindromic-substrings.py
- partition-array-according-to-given-pivot.py
- partition-array-into-three-parts-with-equal-sum.py
- partition-equal-subset-sum.py
- partition-labels.py
- partition-list.py
- partitioning-into-minimum-number-of-deci-binary-numbers.py
- pascals-triangle-ii.py
- pascals-triangle.py
- path-crossing.py
- path-sum-ii.py
- path-sum.py
- peak-index-in-a-mountain-array.py
- perfect-number.py
- perfect-squares.py
- permutation-in-string.py
- permutations-ii.py
- permutations.py
- plus-one.py
- populating-next-right-pointers-in-each-node-ii.py
- populating-next-right-pointers-in-each-node.py
- positions-of-large-groups.py

- power-of-four.py
- power-of-three.py
- power-of-two.py
- powx-n.py
- predict-the-winner.py
- prime-arrangements.py
- print-binary-tree.py
- product-of-array-except-self.py
- push-dominoes.py
- queens-that-can-attack-the-king.py
- queries-on-number-of-points-inside-a-circle.py
- random-pick-index.py
- range-addition-ii.py
- range-sum-of-bst.py
- range-sum-query-immutable.py
- rank-transform-of-an-array.py
- ransom-note.py
- reach-a-number.py
- rearrange-array-elements-by-sign.py
- rearrange-words-in-a-sentence.py
- reconstruct-a-2-row-binary-matrix.py
- recover-binary-search-tree.py
- rectangle-area.py
- rectangle-overlap.py
- reduce-array-size-to-the-half.py
- redundant-connection.py
- relative-ranks.py
- relative-sort-array.py
- remove-all-occurrences-of-a-substring.py
- remove-covered-intervals.py
- remove-duplicate-letters.py
- remove-duplicates-from-sorted-array-ii.py
- remove-duplicates-from-sorted-array.py
- remove-duplicates-from-sorted-list-ii.py
- remove-duplicates-from-sorted-list.py
- remove-element.py
- remove-invalid-parentheses.py

- remove-k-digits.py
- remove-linked-list-elements.py
- remove-nth-node-from-end-of-list.py
- remove-one-element-to-make-the-array-strictly-increasing.py
- remove-outermost-parentheses.py
- remove-zero-sum-consecutive-nodes-from-linked-list.py
- removing-minimum-number-of-magic-beans.py
- reorganize-string.py
- repeated-dna-sequences.py
- repeated-substring-pattern.py
- replace-all-digits-with-characters.py
- replace-elements-with-greatest-element-on-right-side.py
- replace-words.py
- reshape-the-matrix.py
- restore-ip-addresses.py
- restore-the-array-from-adjacent-pairs.py
- reverse-bits.py
- reverse-linked-list.py
- reverse-only-letters.py
- reverse-pairs.py
- reverse-prefix-of-word.py
- reverse-string-ii.py
- reverse-string.py
- reverse-vowels-of-a-string.py
- reverse-words-in-a-string-iii.py
- richest-customer-wealth.py
- rings-and-rods.py
- robot-return-to-origin.py
- roman-to-integer.py
- rotate-array.py
- rotate-image.py
- rotate-list.py
- rotate-string.py
- running-sum-of-1d-array.py
- same-tree.py
- satisfiability-of-equality-equations.py
- score-of-parentheses.py

- search-a-2d-matrix-ii.py
- search-a-2d-matrix.py
- search-in-a-binary-search-tree.py
- search-in-rotated-sorted-array-ii.py
- search-insert-position.py
- search-suggestions-system.py
- second-minimum-node-in-a-binary-tree.py
- self-dividing-numbers.py
- sequential-digits.py
- sequentially-ordinal-rank-tracker.py
- serialize-and-deserialize-binary-tree.py
- serialize-and-deserialize-bst.py
- set-matrix-zeroes.py
- set-mismatch.py
- shift-2d-grid.py
- shortest-palindrome.py
- shortest-path-visiting-all-nodes.py
- shortest-subarray-with-sum-at-least-k.py
- shuffle-an-array.py
- shuffle-string.py
- shuffle-the-array.py
- sign-of-the-product-of-an-array.py
- simplified-fractions.py
- simplify-path.py
- single-element-in-a-sorted-array.py
- single-number-ii.py
- single-number-iii.py
- single-number.py
- sliding-window-maximum.py
- sliding-window-median.py
- smallest-index-with-equal-value.py
- smallest-range-i.py
- smallest-range-ii.py
- smallest-string-starting-from-leaf.py
- smallest-string-with-a-given-numeric-value.py
- smallest-subsequence-of-distinct-characters.py
- smallest-subtree-with-all-the-deepest-nodes.py

- smallest-value-of-the-rearranged-number.py
- snakes-and-ladders.py
- solve-the-equation.py
- sort-an-array.py
- sort-array-by-increasing-frequency.py
- sort-array-by-parity-ii.py
- sort-array-by-parity.py
- sort-characters-by-frequency.py
- sort-colors.py
- sort-even-and-odd-indices-independently.py
- sort-integers-by-the-number-of-1-bits.py
- sort-integers-by-the-power-value.py
- sort-list.py
- sort-the-jumbled-numbers.py
- sort-the-matrix-diagonally.py
- sorting-the-sentence.py
- spiral-matrix-ii.py
- spiral-matrix.py
- split-a-string-in-balanced-strings.py
- split-array-largest-sum.py
- split-linked-list-in-parts.py
- sqrtx.py
- squares-of-a-sorted-array.py
- stone-game.py
- string-compression.py
- subarray-sum-equals-k.py
- subdomain-visit-count.py
- subrectangle-queries.py
- subsets-ii.py
- subsets.py
- substrings-of-size-three-with-distinct-characters.py
- subtract-the-product-and-sum-of-digits-of-an-integer.py
- subtree-of-another-tree.py
- sum-of-all-odd-length-subarrays.cpp
- sum-of-digits-in-base-k.py
- sum-of-digits-of-string-after-convert.py
- sum-of-left-leaves.py

- sum-of-nodes-with-even-valued-grandparent.py
- sum-of-root-to-leaf-binary-numbers.py
- sum-of-square-numbers.py
- sum-of-unique-elements.py
- sum-root-to-leaf-numbers.py
- summary-ranges.py
- super-pow.py
- swap-nodes-in-pairs.py
- swapping-nodes-in-a-linked-list.py
- symmetric-tree.py
- target-sum.py
- teemo-attacking.py
- tenth-line.sh
- the-k-weakest-rows-in-a-matrix.py
- third-maximum-number.py
- thousand-separator.py
- three-divisors.py
- to-lower-case.py
- top-k-frequent-elements.py
- transpose-matrix.py
- trapping-rain-water.py
- triangle.py
- truncate-sentence.py
- two-city-scheduling.py
- two-furthest-houses-with-different-colors.py
- two-out-of-three.py
- two-sum-ii-input-array-is-sorted.py
- two-sum-iv-input-is-a-bst.py
- two-sum.cpp
- two-sum.py
- ugly-number-ii.py
- ugly-number.py
- uncommon-words-from-two-sentences.py
- unique-binary-search-trees-ii.py
- unique-binary-search-trees.py
- unique-email-addresses.py
- unique-morse-code-words.py

- unique-paths-ii.py
- unique-paths-iii.py
- unique-paths.py
- univalued-binary-tree.py
- valid-anagram.py
- valid-palindrome-ii.py
- valid-palindrome.py
- valid-parentheses.py
- valid-perfect-square.py
- valid-square.py
- valid-triangle-number.py
- validate-binary-search-tree.py
- validate-stack-sequences.py
- verifying-an-alien-dictionary.py
- vertical-order-traversal-of-a-binary-tree.py
- vowels-of-all-substrings.py
- water-and-jug-problem.py
- water-bottles.py
- widest-vertical-area-between-two-points-containing-no-points.py
- wiggle-subsequence.py
- wildcard-matching.py
- word-break.py
- word-ladder.py
- word-pattern.py
- word-search.py
- xor-operation-in-an-array.py
- zigzag-conversion.py
