**The Interview Cake Course**

**You left off on Parenthesis Matching**

[**Go there**](https://www.interviewcake.com/table-of-contents#section_queues-stacks_question_matching-parens)

**0. Algorithmic thinking**

Readings

**[Big O Notation](https://www.interviewcake.com/article/python3/big-o-notation-time-and-space-complexity?course=fc1&section=algorithmic-thinking)**

[Learn how to compare the efficiency of different approaches to a problem.](https://www.interviewcake.com/article/python3/big-o-notation-time-and-space-complexity?course=fc1&section=algorithmic-thinking)

**[Data Structures](https://www.interviewcake.com/article/python3/data-structures-coding-interview?course=fc1&section=algorithmic-thinking)**

[Build the main data structures from the ground up. Learn when to use an array vs. a linked list vs.…](https://www.interviewcake.com/article/python3/data-structures-coding-interview?course=fc1&section=algorithmic-thinking)

**[Logarithms](https://www.interviewcake.com/article/python3/logarithms?course=fc1&section=algorithmic-thinking)**

[Logarithms come up a lot in algorithms. Review how they work and learn the situations where they're…](https://www.interviewcake.com/article/python3/logarithms?course=fc1&section=algorithmic-thinking)

**1. Array and string manipulation**

Readings

**[Array](https://www.interviewcake.com/concept/python3/array?course=fc1&section=array-and-string-manipulation)**

[An array is a low-level data structure where elements are identified by integer indices. Arrays are…](https://www.interviewcake.com/concept/python3/array?course=fc1&section=array-and-string-manipulation)

**[Array Slicing](https://www.interviewcake.com/concept/python3/slice?course=fc1&section=array-and-string-manipulation)**

[This is a common tool, but you need to know what it does under the hood!](https://www.interviewcake.com/concept/python3/slice?course=fc1&section=array-and-string-manipulation)

**[In-Place Algorithms](https://www.interviewcake.com/concept/python3/in-place?course=fc1&section=array-and-string-manipulation)**

[An in-place algorithm operates directly on its input and changes it, instead of creating and return…](https://www.interviewcake.com/concept/python3/in-place?course=fc1&section=array-and-string-manipulation)

**[Dynamic Array](https://www.interviewcake.com/concept/python3/dynamic-array?course=fc1&section=array-and-string-manipulation)**

[A dynamic array automatically doubles its size when you try to make an insertion and there's no mor…](https://www.interviewcake.com/concept/python3/dynamic-array?course=fc1&section=array-and-string-manipulation)

Practice

**[Merging Meeting Times](https://www.interviewcake.com/question/python3/merging-ranges?course=fc1&section=array-and-string-manipulation)**

[Write a function for merging meeting times given everyone's schedules. It's an enterprise end-to-en…](https://www.interviewcake.com/question/python3/merging-ranges?course=fc1&section=array-and-string-manipulation)

**[Reverse String in Place](https://www.interviewcake.com/question/python3/reverse-string-in-place?course=fc1&section=array-and-string-manipulation)**

[Write a function to reverse a string in place.](https://www.interviewcake.com/question/python3/reverse-string-in-place?course=fc1&section=array-and-string-manipulation)

**[Reverse Words](https://www.interviewcake.com/question/python3/reverse-words?course=fc1&section=array-and-string-manipulation)**

[Write a function to reverse the word order of a string, in place. It's to decipher a supersecret me…](https://www.interviewcake.com/question/python3/reverse-words?course=fc1&section=array-and-string-manipulation)

**[Merge Sorted Arrays](https://www.interviewcake.com/question/python3/merge-sorted-arrays?course=fc1&section=array-and-string-manipulation)**

[Write a function for consolidating cookie orders and taking over the world.](https://www.interviewcake.com/question/python3/merge-sorted-arrays?course=fc1&section=array-and-string-manipulation)

**[Cafe Order Checker](https://www.interviewcake.com/question/python3/cafe-order-checker?course=fc1&section=array-and-string-manipulation)**

[Write a function to tell us if cafe customer orders are served in the same order they're paid for.](https://www.interviewcake.com/question/python3/cafe-order-checker?course=fc1&section=array-and-string-manipulation)

**2. Hashing and hash tables**

Readings

**[Hashing and Hash Functions](https://www.interviewcake.com/concept/python3/hashing?course=fc1&section=hashing-and-hash-tables)**

[Suppose you wanted a short but unique identifier for a file or dataset. That's the idea behind hash…](https://www.interviewcake.com/concept/python3/hashing?course=fc1&section=hashing-and-hash-tables)

**[Hash Table](https://www.interviewcake.com/concept/python3/hash-map?course=fc1&section=hashing-and-hash-tables)**

[A hash table (also called a hash, hash map or dictionary) is a data structure that pairs keys to va…](https://www.interviewcake.com/concept/python3/hash-map?course=fc1&section=hashing-and-hash-tables)

Practice

**[Inflight Entertainment](https://www.interviewcake.com/question/python3/inflight-entertainment?course=fc1&section=hashing-and-hash-tables)**

[Writing a simple recommendation algorithm that helps people choose which movies to watch during fli…](https://www.interviewcake.com/question/python3/inflight-entertainment?course=fc1&section=hashing-and-hash-tables)

**[Permutation Palindrome](https://www.interviewcake.com/question/python3/permutation-palindrome?course=fc1&section=hashing-and-hash-tables)**

[Check if any permutation of an input string is a palindrome.](https://www.interviewcake.com/question/python3/permutation-palindrome?course=fc1&section=hashing-and-hash-tables)

**[Word Cloud Data](https://www.interviewcake.com/question/python3/word-cloud?course=fc1&section=hashing-and-hash-tables)**

[You're building a word cloud. Write a function to figure out how many times each word appears so we…](https://www.interviewcake.com/question/python3/word-cloud?course=fc1&section=hashing-and-hash-tables)

**[Top Scores](https://www.interviewcake.com/question/python3/top-scores?course=fc1&section=hashing-and-hash-tables)**

[Efficiently sort numbers in an array, where each number is below a certain maximum.](https://www.interviewcake.com/question/python3/top-scores?course=fc1&section=hashing-and-hash-tables)

**3. Greedy algorithms**

Readings

**[Greedy Algorithms](https://www.interviewcake.com/concept/python3/greedy?course=fc1&section=greedy)**

[A greedy algorithm builds up a solution by choosing the option that looks the best at every step.](https://www.interviewcake.com/concept/python3/greedy?course=fc1&section=greedy)

Practice

**[Apple Stocks](https://www.interviewcake.com/question/python3/stock-price?course=fc1&section=greedy)**

[Figure out the optimal buy and sell time for a given stock, given its prices yesterday.](https://www.interviewcake.com/question/python3/stock-price?course=fc1&section=greedy)

**[Highest Product of 3](https://www.interviewcake.com/question/python3/highest-product-of-3?course=fc1&section=greedy)**

[Find the highest possible product that you can get by multiplying any 3 numbers from an input array.](https://www.interviewcake.com/question/python3/highest-product-of-3?course=fc1&section=greedy)

**[Product of All Other Numbers](https://www.interviewcake.com/question/python3/product-of-other-numbers?course=fc1&section=greedy)**

[For each number in an array, find the product of all the other numbers. You can do it faster than y…](https://www.interviewcake.com/question/python3/product-of-other-numbers?course=fc1&section=greedy)

**[Cafe Order Checker](https://www.interviewcake.com/question/python3/cafe-order-checker?course=fc1&section=greedy)**

[Write a function to tell us if cafe customer orders are served in the same order they're paid for.](https://www.interviewcake.com/question/python3/cafe-order-checker?course=fc1&section=greedy)

**[In-Place Shuffle](https://www.interviewcake.com/question/python3/shuffle?course=fc1&section=greedy)**

[Do an in-place shuffle on an array of numbers. It's trickier than you might think!](https://www.interviewcake.com/question/python3/shuffle?course=fc1&section=greedy)

**4. Sorting, searching, and logarithms**

Readings

**[Binary Search Algorithm](https://www.interviewcake.com/concept/python3/binary-search?course=fc1&section=sorting-searching-logarithms)**

[Binary search is a clever way to find an item in a sorted array in O(lg n) time. It involves iterat…](https://www.interviewcake.com/concept/python3/binary-search?course=fc1&section=sorting-searching-logarithms)

Practice

**[Find Rotation Point](https://www.interviewcake.com/question/python3/find-rotation-point?course=fc1&section=sorting-searching-logarithms)**

[I wanted to learn some big words to make people think I'm smart, but I messed up. Write a function …](https://www.interviewcake.com/question/python3/find-rotation-point?course=fc1&section=sorting-searching-logarithms)

**[Find Repeat, Space Edition](https://www.interviewcake.com/question/python3/find-duplicate-optimize-for-space?course=fc1&section=sorting-searching-logarithms)**

[Figure out which number is repeated. But here's the catch: optimize for space.](https://www.interviewcake.com/question/python3/find-duplicate-optimize-for-space?course=fc1&section=sorting-searching-logarithms)

**[Top Scores](https://www.interviewcake.com/question/python3/top-scores?course=fc1&section=sorting-searching-logarithms)**

[Efficiently sort numbers in an array, where each number is below a certain maximum.](https://www.interviewcake.com/question/python3/top-scores?course=fc1&section=sorting-searching-logarithms)

**[Merging Meeting Times](https://www.interviewcake.com/question/python3/merging-ranges?course=fc1&section=sorting-searching-logarithms)**

[Write a function for merging meeting times given everyone's schedules. It's an enterprise end-to-en…](https://www.interviewcake.com/question/python3/merging-ranges?course=fc1&section=sorting-searching-logarithms)

**5. Trees and graphs**

Readings

**[Binary Tree](https://www.interviewcake.com/concept/python3/binary-tree?course=fc1&section=trees-graphs)**

[A binary tree is a tree where every node has two or fewer children. The children are usually called…](https://www.interviewcake.com/concept/python3/binary-tree?course=fc1&section=trees-graphs)

**[Graph](https://www.interviewcake.com/concept/python3/graph?course=fc1&section=trees-graphs)**

[Graphs are like a trees, but with no set root node. They can be directed or undirected, cyclic or a…](https://www.interviewcake.com/concept/python3/graph?course=fc1&section=trees-graphs)

**[Breadth-First Search (BFS)](https://www.interviewcake.com/concept/python3/bfs?course=fc1&section=trees-graphs)**

[Breadth-first search is a method for walking through a tree or graph where you "fan out" as much as…](https://www.interviewcake.com/concept/python3/bfs?course=fc1&section=trees-graphs)

**[Depth-First Search (DFS)](https://www.interviewcake.com/concept/python3/dfs?course=fc1&section=trees-graphs)**

[Depth-first search is a method for walking through a tree or graph where you go as deep as possible…](https://www.interviewcake.com/concept/python3/dfs?course=fc1&section=trees-graphs)

Practice

**[Balanced Binary Tree](https://www.interviewcake.com/question/python3/balanced-binary-tree?course=fc1&section=trees-graphs)**

[Write a function to see if a binary tree is 'superbalanced'--a new tree property we just made up.](https://www.interviewcake.com/question/python3/balanced-binary-tree?course=fc1&section=trees-graphs)

**[Binary Search Tree Checker](https://www.interviewcake.com/question/python3/bst-checker?course=fc1&section=trees-graphs)**

[Write a function to check that a binary tree is a valid binary search tree.](https://www.interviewcake.com/question/python3/bst-checker?course=fc1&section=trees-graphs)

**[2nd Largest Item in a Binary Search Tree](https://www.interviewcake.com/question/python3/second-largest-item-in-bst?course=fc1&section=trees-graphs)**

[Find the second largest element in a binary search tree.](https://www.interviewcake.com/question/python3/second-largest-item-in-bst?course=fc1&section=trees-graphs)

**[Graph Coloring](https://www.interviewcake.com/question/python3/graph-coloring?course=fc1&section=trees-graphs)**

[Color the nodes in a graph so adjacent nodes always have different colors.](https://www.interviewcake.com/question/python3/graph-coloring?course=fc1&section=trees-graphs)

**[MeshMessage](https://www.interviewcake.com/question/python3/mesh-message?course=fc1&section=trees-graphs)**

[You wrote a trendy new messaging app, MeshMessage, to get around flaky cell phone coverage. But mes…](https://www.interviewcake.com/question/python3/mesh-message?course=fc1&section=trees-graphs)

**[Find Repeat, Space Edition BEAST MODE](https://www.interviewcake.com/question/python3/find-duplicate-optimize-for-space-beast-mode?course=fc1&section=trees-graphs)**

[Figure out which number is repeated. But here's the catch: do it in linear time and constant space!](https://www.interviewcake.com/question/python3/find-duplicate-optimize-for-space-beast-mode?course=fc1&section=trees-graphs)

**6. Dynamic programming and recursion**

Readings

**[Overlapping Subproblems](https://www.interviewcake.com/concept/python3/overlapping-subproblems?course=fc1&section=dynamic-programming-recursion)**

[A problem has overlapping subproblems if finding its solution involves solving the same subproblem …](https://www.interviewcake.com/concept/python3/overlapping-subproblems?course=fc1&section=dynamic-programming-recursion)

**[Memoization](https://www.interviewcake.com/concept/python3/memoization?course=fc1&section=dynamic-programming-recursion)**

[Memoization ensures that a function doesn't run for the same inputs more than once. It's generally …](https://www.interviewcake.com/concept/python3/memoization?course=fc1&section=dynamic-programming-recursion)

**[Bottom-Up Algorithms](https://www.interviewcake.com/concept/python3/bottom-up?course=fc1&section=dynamic-programming-recursion)**

[Going bottom-up is a way to avoid recursion, saving memory cost in the call stack. It's a common st…](https://www.interviewcake.com/concept/python3/bottom-up?course=fc1&section=dynamic-programming-recursion)

Practice

**[Recursive String Permutations](https://www.interviewcake.com/question/python3/recursive-string-permutations?course=fc1&section=dynamic-programming-recursion)**

[Write a recursive function of generating all permutations of an input string.](https://www.interviewcake.com/question/python3/recursive-string-permutations?course=fc1&section=dynamic-programming-recursion)

**[Compute nth Fibonacci Number](https://www.interviewcake.com/question/python3/nth-fibonacci?course=fc1&section=dynamic-programming-recursion)**

[Computer the nth Fibonacci number. Careful--the recursion can quickly spin out of control!](https://www.interviewcake.com/question/python3/nth-fibonacci?course=fc1&section=dynamic-programming-recursion)

**[Making Change](https://www.interviewcake.com/question/python3/coin?course=fc1&section=dynamic-programming-recursion)**

[Write a function that will replace your role as a cashier and make everyone rich or something.](https://www.interviewcake.com/question/python3/coin?course=fc1&section=dynamic-programming-recursion)

**[The Cake Thief](https://www.interviewcake.com/question/python3/cake-thief?course=fc1&section=dynamic-programming-recursion)**

[You've hit the mother lode: the cake vault of the Queen of England. Figure out how much of each cak…](https://www.interviewcake.com/question/python3/cake-thief?course=fc1&section=dynamic-programming-recursion)

**[Balanced Binary Tree](https://www.interviewcake.com/question/python3/balanced-binary-tree?course=fc1&section=dynamic-programming-recursion)**

[Write a function to see if a binary tree is 'superbalanced'--a new tree property we just made up.](https://www.interviewcake.com/question/python3/balanced-binary-tree?course=fc1&section=dynamic-programming-recursion)

**[Binary Search Tree Checker](https://www.interviewcake.com/question/python3/bst-checker?course=fc1&section=dynamic-programming-recursion)**

[Write a function to check that a binary tree is a valid binary search tree.](https://www.interviewcake.com/question/python3/bst-checker?course=fc1&section=dynamic-programming-recursion)

**[2nd Largest Item in a Binary Search Tree](https://www.interviewcake.com/question/python3/second-largest-item-in-bst?course=fc1&section=dynamic-programming-recursion)**

[Find the second largest element in a binary search tree.](https://www.interviewcake.com/question/python3/second-largest-item-in-bst?course=fc1&section=dynamic-programming-recursion)

**7. Queues and stacks**

Readings

**[Queue](https://www.interviewcake.com/concept/python3/queue?course=fc1&section=queues-stacks)**

[A queue is like a line at the movie theater. It's "first in, first out" (FIFO). It's usually best t…](https://www.interviewcake.com/concept/python3/queue?course=fc1&section=queues-stacks)

**[Stack](https://www.interviewcake.com/concept/python3/stack?course=fc1&section=queues-stacks)**

[A stack is like a stack of plates. It's "last in, first out" (LIFO), which means that the item that…](https://www.interviewcake.com/concept/python3/stack?course=fc1&section=queues-stacks)

Practice

**[Largest Stack](https://www.interviewcake.com/question/python3/largest-stack?course=fc1&section=queues-stacks)**

[You've implemented a Stack class, but you want to access the largest element in your stack from tim…](https://www.interviewcake.com/question/python3/largest-stack?course=fc1&section=queues-stacks)

**[Implement A Queue With Two Stacks](https://www.interviewcake.com/question/python3/queue-two-stacks?course=fc1&section=queues-stacks)**

[Implement a queue with two stacks. Assume you already have a stack implementation.](https://www.interviewcake.com/question/python3/queue-two-stacks?course=fc1&section=queues-stacks)

**[Parenthesis Matching](https://www.interviewcake.com/question/python3/matching-parens?course=fc1&section=queues-stacks)**

[Write a function that finds the corresponding closing parenthesis given the position of an opening …](https://www.interviewcake.com/question/python3/matching-parens?course=fc1&section=queues-stacks)

**[Bracket Validator](https://www.interviewcake.com/question/python3/bracket-validator?course=fc1&section=queues-stacks)**

[Write a super-simple JavaScript parser that can find bugs in your intern's code.](https://www.interviewcake.com/question/python3/bracket-validator?course=fc1&section=queues-stacks)

**8. Linked lists**

Readings

**[Linked List](https://www.interviewcake.com/concept/python3/linked-list?course=fc1&section=linked-lists)**

[A linked list is a low-level data structure that stores an ordered list of "nodes." The order is st…](https://www.interviewcake.com/concept/python3/linked-list?course=fc1&section=linked-lists)

Practice

**[Delete Node](https://www.interviewcake.com/question/python3/delete-node?course=fc1&section=linked-lists)**

[Write a function to delete a node from a linked list. Turns out you can do it in constant time!](https://www.interviewcake.com/question/python3/delete-node?course=fc1&section=linked-lists)

**[Does This Linked List Have A Cycle?](https://www.interviewcake.com/question/python3/linked-list-cycles?course=fc1&section=linked-lists)**

[Check to see if a linked list has a cycle. We'll start with a simple solution and move on to some p…](https://www.interviewcake.com/question/python3/linked-list-cycles?course=fc1&section=linked-lists)

**[Reverse A Linked List](https://www.interviewcake.com/question/python3/reverse-linked-list?course=fc1&section=linked-lists)**

[Write a function to reverse a linked list in place.](https://www.interviewcake.com/question/python3/reverse-linked-list?course=fc1&section=linked-lists)

**[Kth to Last Node in a Singly-Linked List](https://www.interviewcake.com/question/python3/kth-to-last-node-in-singly-linked-list?course=fc1&section=linked-lists)**

[Find the kth to last node in a singly-linked list. We'll start with a simple solution and move on t…](https://www.interviewcake.com/question/python3/kth-to-last-node-in-singly-linked-list?course=fc1&section=linked-lists)

**[Find Repeat, Space Edition BEAST MODE](https://www.interviewcake.com/question/python3/find-duplicate-optimize-for-space-beast-mode?course=fc1&section=linked-lists)**

[Figure out which number is repeated. But here's the catch: do it in linear time and constant space!](https://www.interviewcake.com/question/python3/find-duplicate-optimize-for-space-beast-mode?course=fc1&section=linked-lists)

**9. System design**

Practice

**[URL Shortener](https://www.interviewcake.com/question/python3/url-shortener?course=fc1&section=system-design)**

[Design a URL shortener, like bit.ly](https://www.interviewcake.com/question/python3/url-shortener?course=fc1&section=system-design)

**[MillionGazillion](https://www.interviewcake.com/question/python3/compress-url-list?course=fc1&section=system-design)**

[I'm making a new search engine called MillionGazillion(tm), and I need help figuring out what data …](https://www.interviewcake.com/question/python3/compress-url-list?course=fc1&section=system-design)

**[Find Duplicate Files](https://www.interviewcake.com/question/python3/find-duplicate-files?course=fc1&section=system-design)**

[Your friend copied a bunch of your files and put them in random places around your hard drive. Writ…](https://www.interviewcake.com/question/python3/find-duplicate-files?course=fc1&section=system-design)

**10. General programming**

Readings

**[Short Circuit Evaluation](https://www.interviewcake.com/concept/python3/short-circuit-evaluation?course=fc1&section=general-programming)**

[Short circuit evaluation avoids unnecessary work. Here are some examples.](https://www.interviewcake.com/concept/python3/short-circuit-evaluation?course=fc1&section=general-programming)

**[Garbage Collection](https://www.interviewcake.com/concept/python3/garbage-collection?course=fc1&section=general-programming)**

[If you create an object (like an array) inside a function and that function doesn't return a refere…](https://www.interviewcake.com/concept/python3/garbage-collection?course=fc1&section=general-programming)

**[Closures](https://www.interviewcake.com/concept/python3/js-closure?course=fc1&section=general-programming)**

[A closure is a function that accesses a variable "outside" itself. Here's an example where "message…](https://www.interviewcake.com/concept/python3/js-closure?course=fc1&section=general-programming)

**[Mutable vs Immutable Objects](https://www.interviewcake.com/concept/python3/mutable?course=fc1&section=general-programming)**

[Mutable objects can be changed, while immutable objects can't. In Python, strings are immutable, so…](https://www.interviewcake.com/concept/python3/mutable?course=fc1&section=general-programming)

Practice

**[Rectangular Love](https://www.interviewcake.com/question/python3/rectangular-love?course=fc1&section=general-programming)**

[Find the area of overlap between two rectangles. In the name of love.](https://www.interviewcake.com/question/python3/rectangular-love?course=fc1&section=general-programming)

**[Temperature Tracker](https://www.interviewcake.com/question/python3/temperature-tracker?course=fc1&section=general-programming)**

[Write code to continually track the max, min, mean, and mode as new numbers are inserted into a tra…](https://www.interviewcake.com/question/python3/temperature-tracker?course=fc1&section=general-programming)

**11. Bit manipulation**

Readings

**[Binary Numbers](https://www.interviewcake.com/concept/python3/binary-numbers?course=fc1&section=bit-manipulation)**

[An easy-to-understand explanation of how numbers are represented in binary, including negative numb…](https://www.interviewcake.com/concept/python3/binary-numbers?course=fc1&section=bit-manipulation)

**[Bitwise AND](https://www.interviewcake.com/concept/python3/and?course=fc1&section=bit-manipulation)**

[Think of bitwise AND like a hose with two knobs. /Both/ knobs must be set to "on" for water to come…](https://www.interviewcake.com/concept/python3/and?course=fc1&section=bit-manipulation)

**[Bitwise OR](https://www.interviewcake.com/concept/python3/or?course=fc1&section=bit-manipulation)**

[Think of bitwise OR like a bucket with two holes in it. If both holes are closed, no water comes ou…](https://www.interviewcake.com/concept/python3/or?course=fc1&section=bit-manipulation)

**[Bitwise XOR (eXclusive OR)](https://www.interviewcake.com/concept/python3/xor?course=fc1&section=bit-manipulation)**

[Think of bitwise XOR like a narrow bag of chips with that can only fit 1 hand at a time. The only w…](https://www.interviewcake.com/concept/python3/xor?course=fc1&section=bit-manipulation)

**[Bitwise NOT](https://www.interviewcake.com/concept/python3/not?course=fc1&section=bit-manipulation)**

[Bitwise NOT basically "flips" the set of bits you give it, changing all the 1s to 0s and all the 0s…](https://www.interviewcake.com/concept/python3/not?course=fc1&section=bit-manipulation)

**[Bit Shifting](https://www.interviewcake.com/concept/python3/bit-shift?course=fc1&section=bit-manipulation)**

[A bit shift moves each digit in a set of bits left or right. The last bit in the direction of the s…](https://www.interviewcake.com/concept/python3/bit-shift?course=fc1&section=bit-manipulation)

**[Integer Overflow](https://www.interviewcake.com/concept/python3/integer-overflow?course=fc1&section=bit-manipulation)**

[When you create an integer variable, your computer allocates 64 bits for storing it. What if your n…](https://www.interviewcake.com/concept/python3/integer-overflow?course=fc1&section=bit-manipulation)

Practice

**[The Stolen Breakfast Drone](https://www.interviewcake.com/question/python3/find-unique-int-among-duplicates?course=fc1&section=bit-manipulation)**

[In a beautiful Amazon utopia where breakfast is delivered by drones, one drone has gone missing. Wr…](https://www.interviewcake.com/question/python3/find-unique-int-among-duplicates?course=fc1&section=bit-manipulation)

**12. Combinatorics, probability, and other math**

Readings

**[Triangular Series](https://www.interviewcake.com/concept/python3/triangular-series?course=fc1&section=combinatorics-probability-math)**

[Triangular series are simple increasing integers starting from 1, like {1,2,3,4,5}. There's a formu…](https://www.interviewcake.com/concept/python3/triangular-series?course=fc1&section=combinatorics-probability-math)

Practice

**[Which Appears Twice](https://www.interviewcake.com/question/python3/which-appears-twice?course=fc1&section=combinatorics-probability-math)**

[Find the repeat number in an array of numbers. Optimize for runtime.](https://www.interviewcake.com/question/python3/which-appears-twice?course=fc1&section=combinatorics-probability-math)

**[Find in Ordered Set](https://www.interviewcake.com/question/python3/find-in-ordered-set?course=fc1&section=combinatorics-probability-math)**

[Given an array of numbers in sorted order, how quickly could we check if a given number is present …](https://www.interviewcake.com/question/python3/find-in-ordered-set?course=fc1&section=combinatorics-probability-math)

**[In-Place Shuffle](https://www.interviewcake.com/question/python3/shuffle?course=fc1&section=combinatorics-probability-math)**

[Do an in-place shuffle on an array of numbers. It's trickier than you might think!](https://www.interviewcake.com/question/python3/shuffle?course=fc1&section=combinatorics-probability-math)

**[Simulate 5-sided die](https://www.interviewcake.com/question/python3/simulate-5-sided-die?course=fc1&section=combinatorics-probability-math)**

[Given a 7-sided die, make a 5-sided die.](https://www.interviewcake.com/question/python3/simulate-5-sided-die?course=fc1&section=combinatorics-probability-math)

**[Simulate 7-sided die](https://www.interviewcake.com/question/python3/simulate-7-sided-die?course=fc1&section=combinatorics-probability-math)**

[Given a 5-sided die, make a 7-sided die.](https://www.interviewcake.com/question/python3/simulate-7-sided-die?course=fc1&section=combinatorics-probability-math)

**[Two Egg Problem](https://www.interviewcake.com/question/python3/two-egg-problem?course=fc1&section=combinatorics-probability-math)**

[A building has 100 floors. Figure out the highest floor an egg can be dropped from without breaking.](https://www.interviewcake.com/question/python3/two-egg-problem?course=fc1&section=combinatorics-probability-math)

**13. JavaScript**

Readings

**[Closures](https://www.interviewcake.com/concept/python3/js-closure?course=fc1&section=javascript)**

[A closure is a function that accesses a variable "outside" itself. Here's an example where "message…](https://www.interviewcake.com/concept/python3/js-closure?course=fc1&section=javascript)

**[In-Place Algorithms](https://www.interviewcake.com/concept/python3/in-place?course=fc1&section=javascript)**

[An in-place algorithm operates directly on its input and changes it, instead of creating and return…](https://www.interviewcake.com/concept/python3/in-place?course=fc1&section=javascript)

Practice

**[JavaScript Scope](https://www.interviewcake.com/question/python3/js-scope?course=fc1&section=javascript)**

[There's something tricky going on with scope in this JavaScript. Can you guess what will get logged…](https://www.interviewcake.com/question/python3/js-scope?course=fc1&section=javascript)

**[What's Wrong with This JavaScript?](https://www.interviewcake.com/question/python3/js-whats-wrong?course=fc1&section=javascript)**

[There's a tricky bug in this JavaScript. Can you find it?](https://www.interviewcake.com/question/python3/js-whats-wrong?course=fc1&section=javascript)

**14. Coding Interview Tips**

Readings

**[How The Coding Interview Works](https://www.interviewcake.com/interview-process-at-tech-companies?course=fc1&section=interview-tips)**

[First time interviewing for a tech job? Not sure what to expect? This article is for you.](https://www.interviewcake.com/interview-process-at-tech-companies?course=fc1&section=interview-tips)

**[General Coding Interview Advice](https://www.interviewcake.com/coding-interview-tips?course=fc1&section=interview-tips)**

[How to get better at coding interviews RIGHT NOW, without practicing.](https://www.interviewcake.com/coding-interview-tips?course=fc1&section=interview-tips)

**[Impostor Syndrome](https://www.interviewcake.com/impostor-syndrome-in-programming-interviews?course=fc1&section=interview-tips)**

[Feel like you got your interview by luck? Like you're a fraud on the verge of being exposed? That's…](https://www.interviewcake.com/impostor-syndrome-in-programming-interviews?course=fc1&section=interview-tips)

**[Why You Hit Dead Ends](https://www.interviewcake.com/why-youre-hitting-dead-ends-in-whiteboard-interviews?course=fc1&section=interview-tips)**

[The coding interview is like a maze. You can only see what's in front of you, but your interviewer …](https://www.interviewcake.com/why-youre-hitting-dead-ends-in-whiteboard-interviews?course=fc1&section=interview-tips)

**[Tips for Getting Unstuck](https://www.interviewcake.com/tricks-for-getting-unstuck-programming-interview?course=fc1&section=interview-tips)**

[You need a lifeline when you get stuck during a coding interview. Here it is.](https://www.interviewcake.com/tricks-for-getting-unstuck-programming-interview?course=fc1&section=interview-tips)

**[The 24 Hours Before Your Interview](https://www.interviewcake.com/24-hours-before-onsite-whiteboard-coding-interview?course=fc1&section=interview-tips)**

[Feeling anxious? That's normal. Your body is telling you you're about to do something that matters.](https://www.interviewcake.com/24-hours-before-onsite-whiteboard-coding-interview?course=fc1&section=interview-tips)

**[Beating Behavioral Questions](https://www.interviewcake.com/behavioral-questions-programming-interview-story-telling?course=fc1&section=interview-tips)**

[Nothing answers a behavioral coding interview question like a good story. Knowing where to add deta…](https://www.interviewcake.com/behavioral-questions-programming-interview-story-telling?course=fc1&section=interview-tips)

**[Managing Your Interview Timeline](https://www.interviewcake.com/coding-interview-timeline-exploding-offers-burnout-negotiation-leverage?course=fc1&section=interview-tips)**

[Interviewing is time-intensive and can get chaotic. Knowing how to manage your timeline will help y…](https://www.interviewcake.com/coding-interview-timeline-exploding-offers-burnout-negotiation-leverage?course=fc1&section=interview-tips)

# In-Place Algorithm

An **in-place** function modifies data structures or objects outside of its own stack frame ↴ (i.e.: stored on the process heap or in the stack frame of a calling function). Because of this, the changes made by the function remain after the call completes.

In-place algorithms are sometimes called **destructive**, since the original input is "destroyed" (or modified) during the function call.

**Careful: "In-place" does not mean "without creating any additional variables!"** Rather, it means "without creating a new copy of the input." In general, an in-place function will only create additional variables that are O(1)*O*(1) space.

An **out-of-place** function doesn't make any changes that are visible to other functions. Usually, those functions copy any data structures or objects before manipulating and changing them.

In many languages, **primitive** values (integers, floating point numbers, or characters) are copied when passed as arguments, and more complex **data structures** (lists, heaps, or hash tables) are passed by reference. This is what Python does.

Here are two functions that do the same operation on a list, except one is in-place and the other is out-of-place:

def square\_list\_in\_place(int\_list):

for index, element in enumerate(int\_list):

int\_list[index] \*= element

# NOTE: no need to return anything - we modified

# int\_list in place

def square\_list\_out\_of\_place(int\_list):

# We allocate a new list with the length of the input list

squared\_list = [None] \* len(int\_list)

for index, element in enumerate(int\_list):

squared\_list[index] = element \*\* 2

return squared\_list



**Working in-place is a good way to save time and space.** An in-place algorithm avoids the cost of initializing or copying data structures, and it usually has an O(1)*O*(1) space cost.

**But be careful: an in-place algorithm can cause side effects.**Your input is "destroyed" or "altered," which can affect code outside of your function. For example:

original\_list = [2, 3, 4, 5]

square\_list\_in\_place(original\_list)

print("original list: %s" % original\_list)

# Prints: original list: [4, 9, 16, 25], confusingly!



**Generally, out-of-place algorithms are considered safer because they avoid side effects.** You should only use an in-place algorithm if you're space constrained or you're positive you don't need the original input anymore, even for debugging.

# Dynamic Array

[Data Structure](https://www.interviewcake.com/data-structures-reference)

Other names:  
array list, growable array, resizable array, mutable array

## Quick reference

|  | **Average Case** | **Worst Case** |
| --- | --- | --- |
| **space** | O(n)*O*(*n*) | O(n)*O*(*n*) |
| **lookup** | O(1)*O*(1) | O(1)*O*(1) |
| **append** | O(1)*O*(1) | O(n)*O*(*n*) |
| **insert** | O(n)*O*(*n*) | O(n)*O*(*n*) |
| **delete** | O(n)*O*(*n*) | O(n)*O*(*n*) |

A **dynamic array** is an [array](https://www.interviewcake.com/concept/array) with a big improvement: automatic resizing.

One limitation of arrays is that they're fixed size, meaning you need to specify the number of elements your array will hold ahead of time.

A dynamic array expands as you add more elements. So you don't need to determine the size ahead of time.

#### Strengths:

* **Fast lookups**. Just like arrays, retrieving the element at a given index takes O(1)*O*(1) time.
* **Variable size**. You can add as many items as you want, and the dynamic array will expand to hold them.
* **Cache-friendly**. Just like arrays, dynamic arrays place items right next to each other in memory, making efficient use of caches.

#### Weaknesses:

* **Slow worst-case appends**. Usually, adding a new element at the end of the dynamic array takes O(1)*O*(1) time. But if the dynamic array doesn't have any room for the new item, [it'll need to expand](https://www.interviewcake.com/concept/python3/dynamic-array?course=fc1&section=array-and-string-manipulation#doubling_appends), which takes O(n)*O*(*n*) time.
* **Costly inserts and deletes.** Just like arrays, elements are stored adjacent to each other. So adding or removing an item in the middle of the array [requires "scooting over" other elements](https://www.interviewcake.com/concept/array#inserting), which takes O(n)*O*(*n*) time.

## In Python 3.6

In Python, dynamic arrays are called lists.

Here's what they look like:

gas\_prices = []

gas\_prices.append(346)

gas\_prices.append(360)

gas\_prices.append(354)

Python 2.7

## Size vs. Capacity

When you allocate a dynamic array, your dynamic array implementation makes an underlying fixed-size array. The starting size depends on the implementation—let's say our implementation uses 10 indices. Now say we append 4 items to our dynamic array. At this point, our dynamic array has a length of 4. But the underlying array has a length of 10.

We'd say this dynamic array's **size** is 4 and its **capacity** is 10. The dynamic array stores an **end\_index** to keep track of where the dynamic array ends and the extra capacity begins.

## Doubling Appends

What if we try to append an item but our array's capacity is already full?

To make room, dynamic arrays automatically make a new, bigger underlying array. Usually twice as big.

Why not just extend the existing array? Because that memory might already be taken by another program.

Each item has to be individually copied into the new array.

Copying each item over costs O(n)*O*(*n*) time! So whenever appending an item to our dynamic array forces us to make a new double-size underlying array, that append takes O(n)*O*(*n*) time.

That's the worst case. But in the best case (and the average case), appends are just O(1)*O*(1) time.

## Amortized cost of appending

1. The time cost of each special O(n)*O*(*n*) "doubling append" doubles each time.
2. At the same time, the number of *O(1)O(1)* appends you get until the next doubling append also doubles.

These two things sort of "cancel out," and we can say each append has an average cost or **amortized cost** of O(1)*O*(1). ↴

Given this, in industry we usually wave our hands and say dynamic arrays have a time cost of O(1)*O*(1) for appends, even though strictly speaking that's only true for the average case or the amortized cost.

**Your company built an in-house calendar tool called HiCal. You want to add a feature to see the times in a day when everyone is available.**

To do this, you’ll need to know when any team is having a meeting. In HiCal, a meeting is stored as a tuple ↴ of integers (start\_time, end\_time). These integers represent the number of 30-minute blocks past 9:00am.

For example:

(2, 3) # Meeting from 10:00 – 10:30 am

(6, 9) # Meeting from 12:00 – 1:30 pm



Write a function merge\_ranges() that takes a list of multiple meeting time ranges and returns a list of condensed ranges.

For example, given:

[(0, 1), (3, 5), (4, 8), (10, 12), (9, 10)]



your function would return:

[(0, 1), (3, 8), (9, 12)]



**Do not assume the meetings are in order.** The meeting times are coming from multiple teams.

**Write a solution that's efficient even when we can't put a nice upper bound on the numbers representing our time ranges.** Here we've simplified our times down to the number of 30-minute slots past 9:00 am. But we want the function to work even for very large numbers, like Unix timestamps. In any case, the spirit of the challenge is to merge meetings where start\_time and end\_time don't have an upper bound.

### Gotchas

Look at this case:

[(1, 2), (2, 3)]



These meetings should probably be merged, although they don't exactly "overlap"—they just "touch." Does your function do this?

Look at this case:

[(1, 5), (2, 3)]



Notice that although the second meeting starts later, it ends before the first meeting ends. Does your function correctly handle the case where a later meeting is "subsumed by" an earlier meeting?

Look at this case:

[(1, 10), (2, 6), (3, 5), (7, 9)]



Here all of our meetings should be merged together into just (1, 10). We need keep in mind that after we've merged the first two we're not done with the result—the result of that merge may itself need to be merged into other meetings as well.

Make sure that your function won't "leave out" the last meeting.

We can do this in O(n\lg{n})*O*(*n*lg*n*) time.

### Breakdown

What if we only had two ranges? Let's take:

[(1, 3), (2, 4)]



These meetings clearly overlap, so we should merge them to give:

[(1, 4)]



But how did we know that these meetings overlap?

We could tell the meetings overlapped because the end time of the first one was after the start time of the second one! But our ideas of "first" and "second" are important here—this only works after we ensure that we treat the meeting that starts earlier as the "first" one.

How would we formalize this as an algorithm? **Be sure to consider these edge cases:**

1. The end time of the first meeting and the start time of the second meeting are equal. For example: [(1, 2), (2, 3)]
2. The second meeting ends before the first meeting ends. For example: [(1, 5), (2, 3)]

Here's a formal algorithm:

1. We treat the meeting with earlier start time as "first," and the other as "second."
2. If the end time of the first meeting is equal to or greater than the start time of the second meeting, we merge the two meetings into one time range. The resulting time range's start time is the first meeting's start, and its end time is the later of the two meetings' end times.
3. Else, we leave them separate.

So, we could compare every meeting to every other meeting in this way, merging them or leaving them separate.

Comparing all pairs of meetings would take O(n^2)*O*(*n*2) time. We can do better!

If we're going to beat O(n^2)*O*(*n*2) time, maybe we're going to get O(n)*O*(*n*) time? Is there a way to do this in one pass?

It'd be great if, for each meeting, we could just try to merge it with the next meeting. But that's definitely not sufficient, because the ordering of our meetings is random. There might be a non-next meeting that the current meeting could be merged with.

What if we sorted our list of meetings by start time?

Then any meetings that could be merged would always be adjacent!

So we could sort our meetings, then walk through the sorted list and see if each meeting can be merged with the one after it.

Sorting takes O(n\lg{n})*O*(*n*lg*n*) time in the worst case. If we can then do the merging in one pass, that's another O(n)*O*(*n*) time, for O(n\lg{n})*O*(*n*lg*n*) overall. That's not as good as O(n)*O*(*n*), but it's better than O(n^2)*O*(*n*2).

### Solution

First, we sort our input list of meetings by start time so any meetings that might need to be merged are now next to each other.

Then we walk through our sorted meetings from left to right. At each step, either:

1. We can merge the current meeting with the previous one, so we do.
2. We can't merge the current meeting with the previous one, so we know the previous meeting can't be merged with any future meetings and we throw the current meeting into merged\_meetings.

def merge\_ranges(meetings):

# Sort by start time

sorted\_meetings = sorted(meetings)

# Initialize merged\_meetings with the earliest meeting

merged\_meetings = [sorted\_meetings[0]]

for current\_meeting\_start, current\_meeting\_end in sorted\_meetings[1:]:

last\_merged\_meeting\_start, last\_merged\_meeting\_end = merged\_meetings[-1]

# If the current meeting overlaps with the last merged meeting, use the

# later end time of the two

if (current\_meeting\_start <= last\_merged\_meeting\_end):

merged\_meetings[-1] = (last\_merged\_meeting\_start,

max(last\_merged\_meeting\_end,

current\_meeting\_end))

else:

# Add the current meeting since it doesn't overlap

merged\_meetings.append((current\_meeting\_start, current\_meeting\_end))

return merged\_meetings



### Complexity

O(n\lg{n})*O*(*n*lg*n*) time and O(n)*O*(*n*) space.

Even though we only walk through our list of meetings once to merge them, we sort all the meetings first, giving us a runtime of O(n\lg{n})*O*(*n*lg*n*). It's worth noting that if our input were sorted, we could skip the sort and do this in O(n)*O*(*n*) time!

We create a new list of merged meeting times. In the worst case, none of the meetings overlap, giving us a list identical to the input list. Thus we have a worst-case space cost of O(n)*O*(*n*).

### Bonus

1. What if we did have an upper bound on the input values? Could we improve our runtime? Would it cost us memory?
2. Could we do this "in place" on the input list and save some space? What are the pros and cons of doing this in place?

### What We Learned

This one arguably uses a greedy ↴ approach as well, except this time we had to sort the list first.

How did we figure that out?

We started off trying to solve the problem in one pass, and we noticed that it wouldn't work. We then noticed the reason it wouldn't work: to see if a given meeting can be merged, we have to look at all the other meetings! That's because the order of the meetings is random.

That's what got us thinking: what if the list were sorted? We saw that then a greedy approach would work. We had to spend O(n\lg{n})*O*(*n*lg*n*) time on sorting the list, but it was better than our initial brute force approach, which cost us O(n^2)*O*(*n*2) time!