

Algorithms and data structures professors watching someone loop through a list instead of creating a custom recursive binary heap object to store 7 integers



Data Structures and Algorithms 😐

Binary Search

Do whenever you can divide the search space by half. Searching for an element in a sorted array.

Example question: Find the peak element in an array i.e. an element that is at least as big as its neighbours (is this guaranteed to exist in every array?)

Cooler example: Find if a point is inside an n sided polygon with vertices given in cyclic order.

Binary Search

Flipping the question:

1. Given an array of positive integers and a sum S , find the smallest subarray where the sum is at least S .

Question: Fixed sum S , find the best size of subarray

Flipped: Fixed size of subarray, find the best sum S

2. Factory Machines. A factory has n machines which can be used to make products. Your goal is to make a total of t products.

For each machine, you know the number of seconds it needs to make a single product. The machines can work simultaneously, and you can freely decide their schedule.

What is the shortest time needed to make t products?

Binary Search

3. Array Division

You are given an array containing n positive integers.

Your task is to divide the array into k subarrays so that the maximum sum in a subarray is as small as possible.

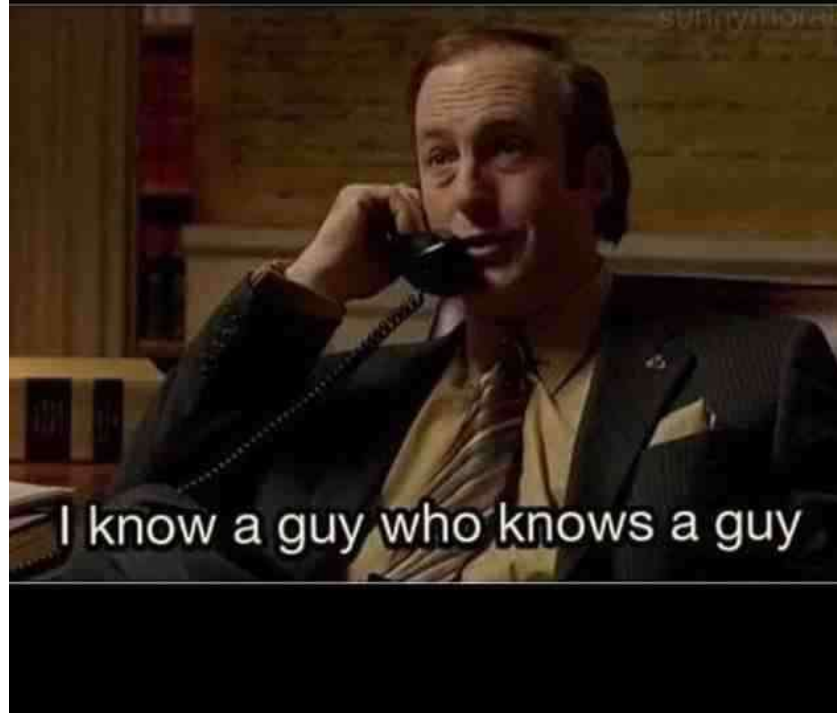
Example: $[2, 4, 7, 3, 5]$, $k = 3$

Original Question?

Flipped Question?

Linked Lists

Linked List data structures be like:



Linked Lists

Given 2 linked lists sorted in ascending order, merge the nodes into a single linked list. You can't modify the values of a node, only change the pointers of each node. $O(m+n)$ time $O(1)$ space.

Can you modify this to do a merge sort algorithm for a linked list? What's the space complexity? $O(n)$? $O(\log n)$? $O(1)$?

Check if the last element of a linked list points to NULL or a previous element in $O(n)$ time $O(1)$ space (fast and slow pointers)

Dictionaries

Idk just read slides lol, nothing much

Remember stuff like open addressing, and revise the random probability calculations, exercises very important.

Trees

Just recurse

In order to understand recursion you must first understand recursion

Don't neglect base cases like the above "joke"

Preorder, postorder, inorder (inorder only for binary trees)

Do they all actually visit nodes in a different order?

Which is the real ordering?

Is level order different from all these?

Some Questions on Trees

For recursion, always remember that sometimes you may need to solve more than the question is asking, in order to recurse.

1. Maybe you'll need to return multiple things from your function.
2. Maybe your function will take even more inputs than just the normal function being asked.

Some important questions: construct a binary tree from any 2 given walks. How to make this memory efficient? (think of point 2)

Given a tree, find the max sum for a path from leaf to leaf. (think of point 1 here)

Check if a binary tree is a BST

Find the sum of all root to leaf paths

BSTs and Balanced BSTs

One of the most useful data structure ever, can implement ordered sets, maps, (heaps?)

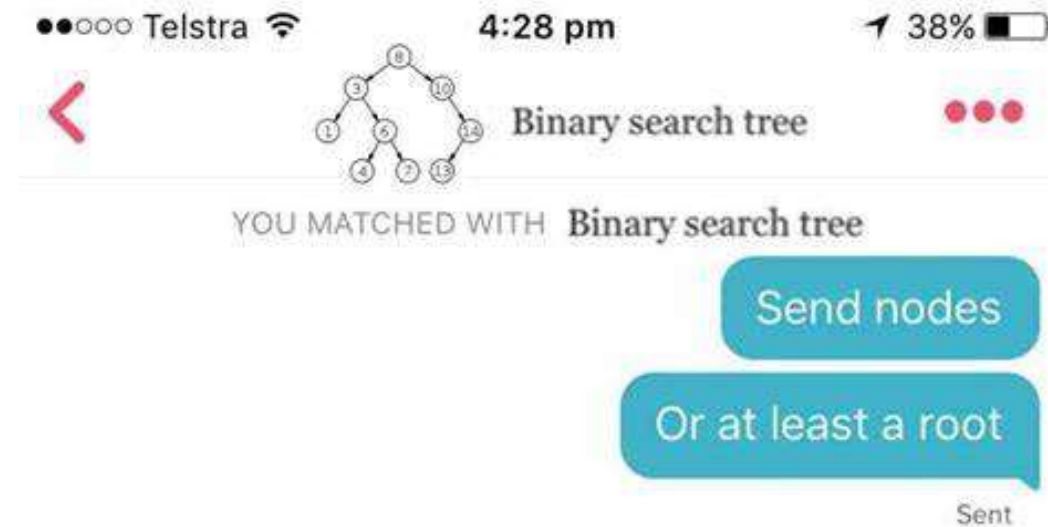
Random question: can you explain the algo to delete a node in a BST?

Can you come up with an algorithm to sort a list using a balanced BST?

Try to understand all cases in the algos mentioned in class, most likely won't be asked but it will help you think of similar logic in random questions, and helps you understand the data structure better

Do you understand the algorithm for successor predecessor?

Random Meme idk lol



Random Question

There's a street of length x . n traffic signals are added one by one on the road, on integer coordinates between 0 and x . (Assume $x \gg n$)

After each traffic signal is added we want to find the longest section of road without no signals.

For example, say $x = 10$, and $n = 5$

The signals are placed at 2, 3, 6, 4, 1.

What is the output?

Think of an $O(n^2)$ solution first. Can we do better?

Priority Queues and Heaps

Coollest data structure in my opinion

Can be just an array but does important stuff in $O(\log n)$ time

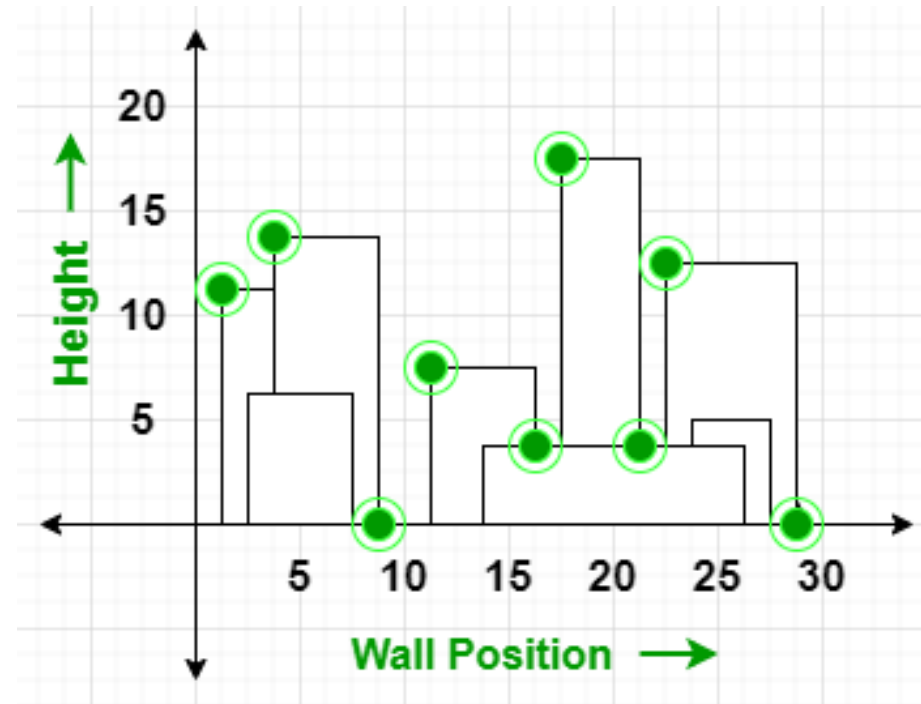
Why is heapify $O(n)$ but heapsort $O(n \log n)$?

Remember to use these if your algorithm is doing a lot of insertions, and wants to retrieve a single element which is “maximum” or minimum.

Merge k sorted linked lists of length n in $n \log k$

Sliding window maximum: for every window of size k , find the maximum element

The Skyline Problem (last year midsem)



Given a set of building $(x_start, x_end, height)$, output the skyline (outline) or for simplicity the area

KMP Algorithm

Do understand and think about it

Computing array h is a bit complicated compared to the normal algorithm

The questions on this aren't too much, just be clear with the tut problems as well as the slides and their exercises

Very big brain algo and has pretty useful applications

Tries

Ugliest data structure in my opinion

Just read slides, understand how to do exercises and tut problems

Understand suffix trees and their uses, how to construct them

There's nothing too complicated

Ideas to Tackle (Hard) Problems

Coming up with the right way/perspective to a problem

Get an important observation by looking at things differently

Given an array of length n , find the maximum subarray sum? $O(n^2)$ to $O(n)$

*reducing to a subproblem idea is very useful (this is more of a DAA concept though)

Harder version, what if instead of an array you have a circular linked list?

Is there a way to deal with arrays that loop back?

Coming up with the right perspective

BITWISE AND

Given a binary tree with n nodes, split the tree into 2 trees by cutting off an edge such that the bitwise and of one tree is the same as the bitwise and of the rest of the tree

What is bitwise and?

Is there any observation you can tell if bitwise and of 2 trees are equal?

2 Pointer methods

Think about for which edge case or boundary case you directly know the answer

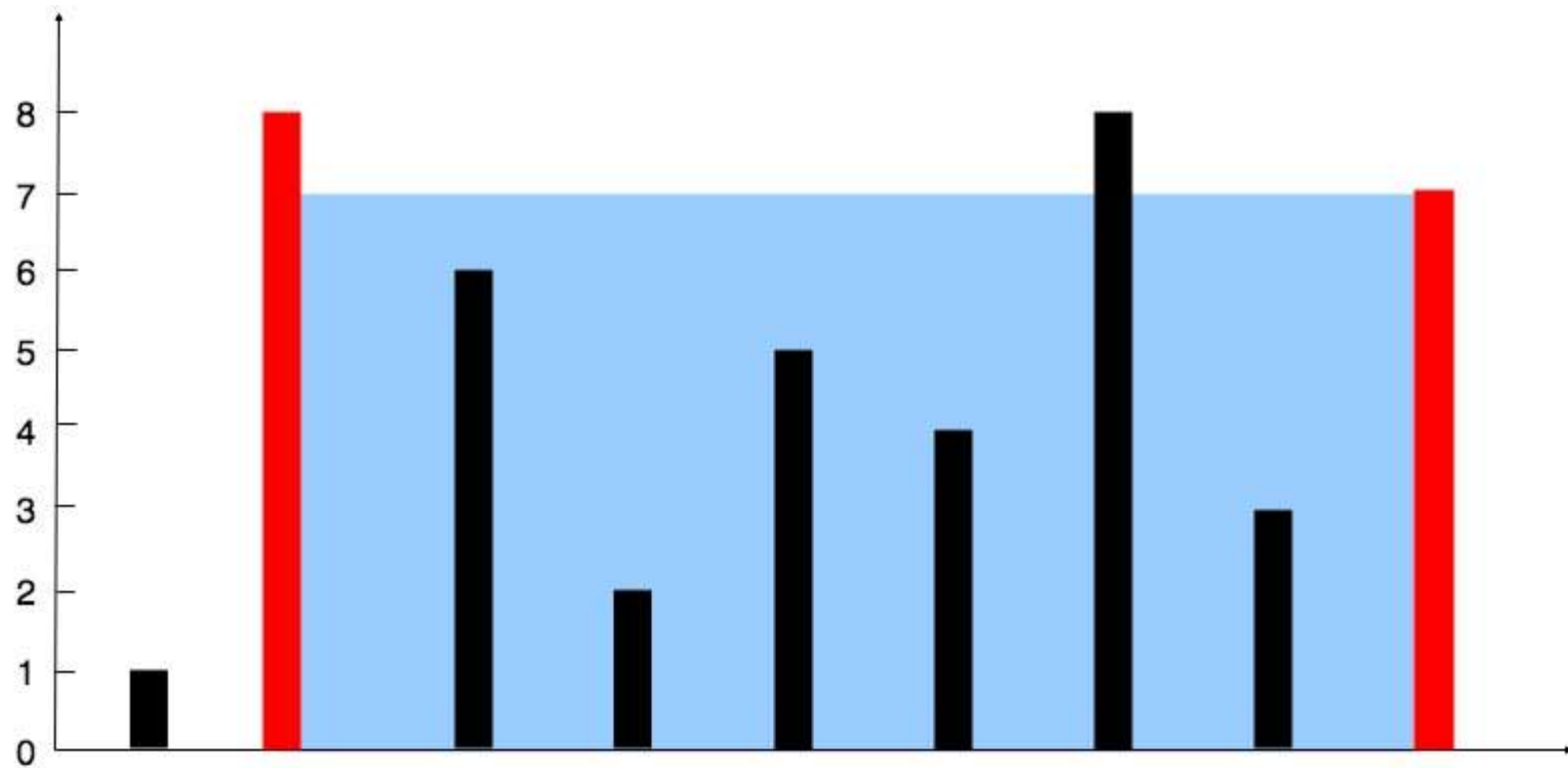
Remove that edge case

Solve your subproblem

Find 2 numbers in an array which sum to a given value x

Given an array of positive integers and a sum S , find the smallest subarray where the sum is at least S (can be done with 2 pointers)

Container with most water



Claiming something to be true

Your lab midsem question

Can you always add parantheses such that you get the maximum score?

Question where idea is simple but proof is hard:

Given 3 stacks, implement a dequeue where you can pop-front, pop-back, push-front, push-back all in amortized $O(1)$

Hopefully you all aren't dead by now

This is a long TSC I know 😞

No segfaults in Psuedo Code 🤪

