Lesson #3

August 26, 2019

Warm up exercise

- Who here remembers Kazaa, Napster, Limelight?
- Implement a sorted, circular, double-linked list
    - Implement put (insertion) and get (fetch) operations

In the previous episode...

- We implemented linear data structures such as unordered linked lists, ordered linked lists, stacks with linked lists, queues with linked lists, stacks with queues, queues with stacks
  - This is an example of abstraction - behavior/functionality (“what does it do?”) versus implementation (“how does it work?”)
- By now, you should recognize the design of data structures are malleable, and can be reshaped and manipulated

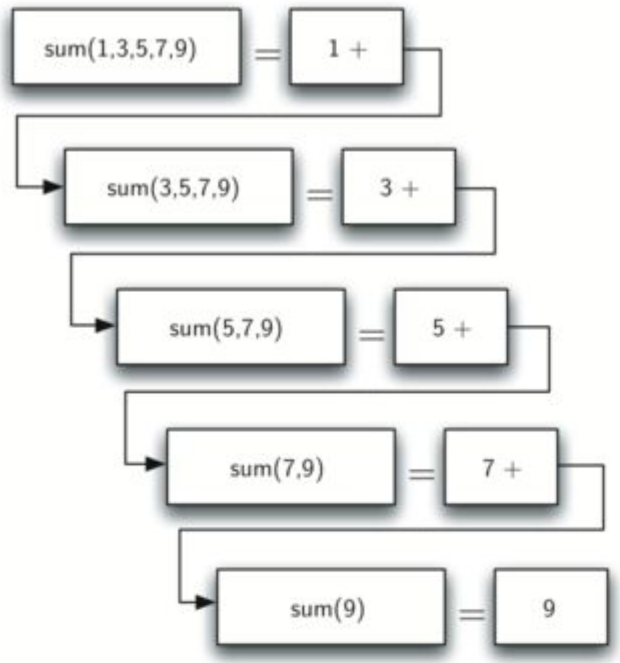
What else can we do?

Build towers of abstractions upon abstractions upon abstractions…

Recursion: A brief tour

- From data primitives (eg nodes), we constructed linked lists
- From linked lists, we constructed stacks
- From stacks, we can do recursion
  - (And then, with recursion, we can think about things like dynamic programming, etc.)

Recursion: A brief tour



Recursion: A brief tour

- See how these data structures build on top of each other?
- That’s why it’s important to understand each concept, as they feed into the succeeding one...

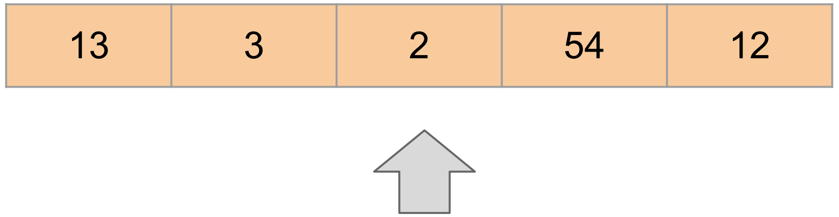
Now What?

Most CS problems boils down to search

- Google, search for knowledge
- Airbnb, search for lodging
- Uber, search for transportation
- Facebook, search for friends
- LinkedIn, search for networking

Most CS Problems boils down to search

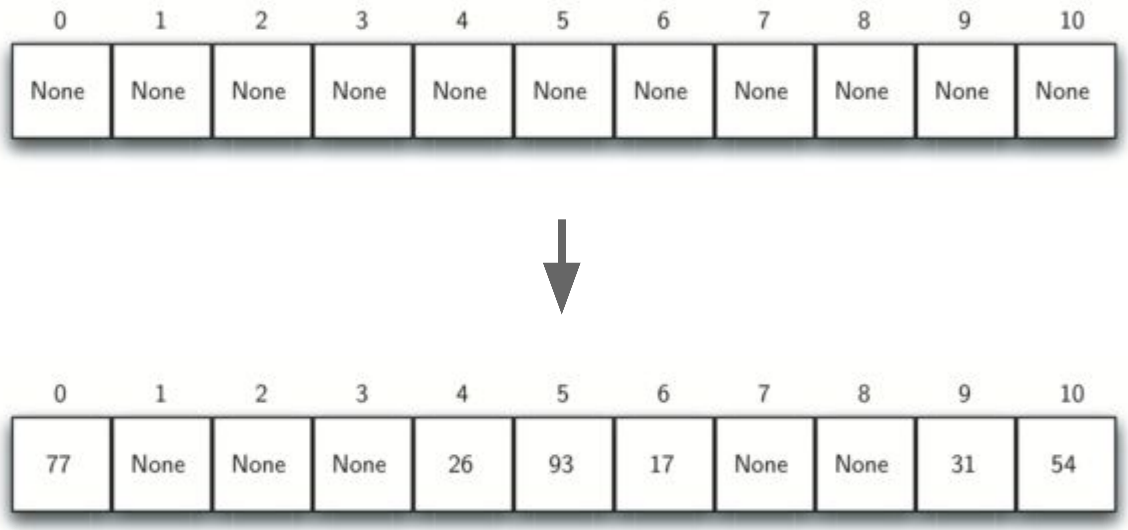
- We hold data in some data structure and then ask questions (eg “does it contain the number 2?”)



Could We Do Better?

Can we ask our data structure (an unsorted list) to do more work? (So we would do less work down the line…)

Sure, let's implement hashmaps!



0 1 2 3 4 5 6 7 8 9 10
None None None None None None None None None None None
0 1 2 3 4 5 6 7 8 9 10
77 None None None 26 93 17 None None 31 54

But hash maps are limited...

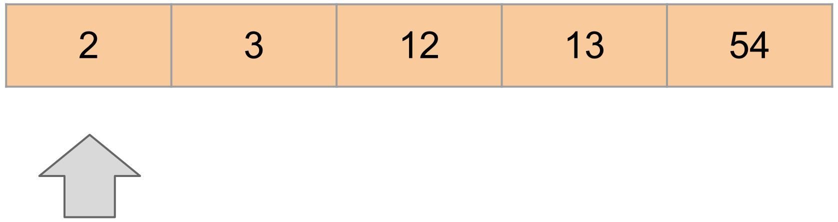
- We could ask questions like “Is the number 2 in this array?” but it’d be difficult to ask questions like “What is the smallest number in this array
    - Why?

Structure and time complexity

Structure	**Amount of “structure”**	**Ease of search**
Unsorted Linked List	Least	O(n)
Priority Queue	Some structure (ie heap property) but not necessarily sorted	Depends. Popping off min/max is O(1). Everything else would be O(log n)
Sorted List	Lots of structure - everything has a place	O(log n)

Could We Do Better?

- Yes! Let's ask our data structures to do more work and be more "structured"
- We could sort the list before searching (we will talk about searching an ordered list soon)
    - Searching an ordered list is O(log n) rather than O(n)



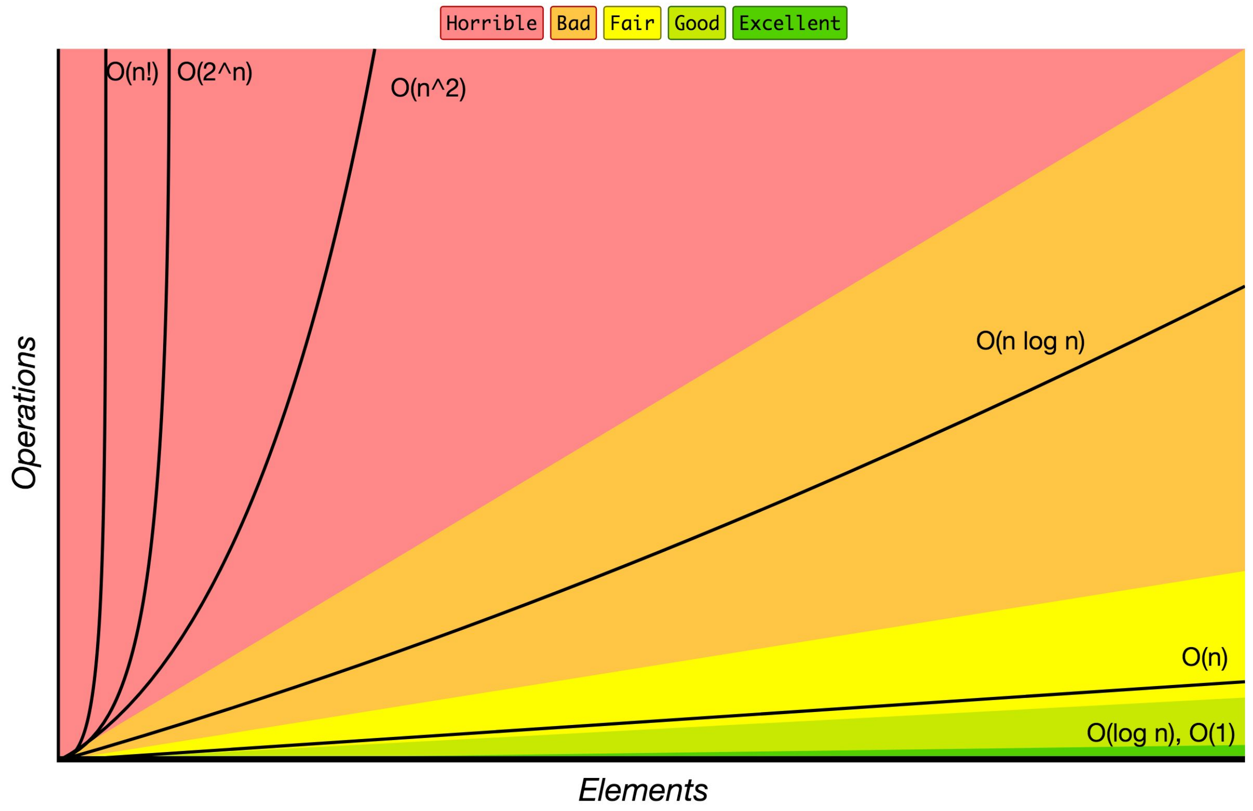
Sorting is an additional step…

- We’re no longer *just* storing stuff in data structures
- We’re now storing stuff in data structures, reshaping that data structure (eg sorting), and then asking questions

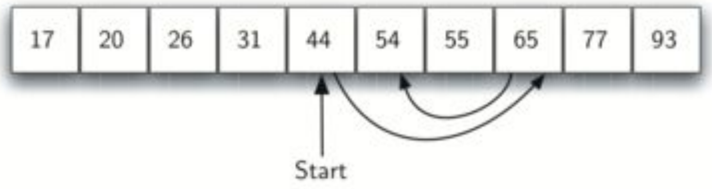
… but it (usually) pays for itself

- Searching an unordered list is O(n)
- Sorting an unordered list, then searching this ordered list is O(nlogn) + O(logn)
    - Benefit accrues with repeated searches
    - Anytime you can search with O(log n), you’re in good shape

How do we shrink O(n) to O(log n)?



Binary Search to the rescue!



Why is binary search O(log n)?

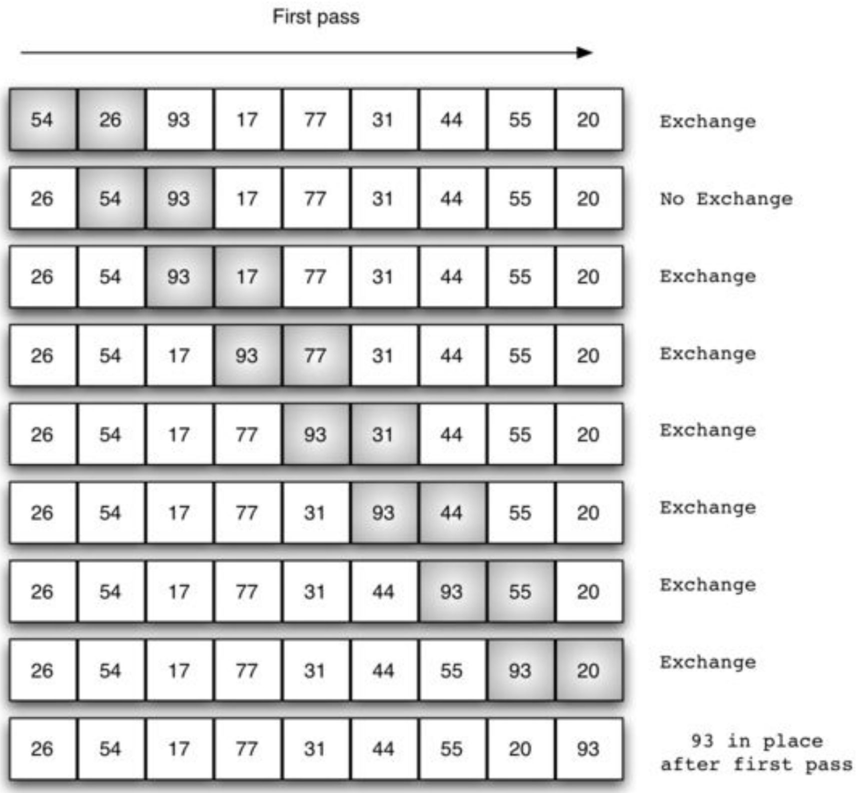
Comparisons	Approximate Number of Items Left
1	$\frac{n}{2}$
2	$\frac{n}{4}$
3	$\frac{n}{8}$
...	
i	$\frac{n}{2^i}$

Now we know why sorting is important, let's learn sorting algorithms!

Sorting Algorithms (the usual suspects)

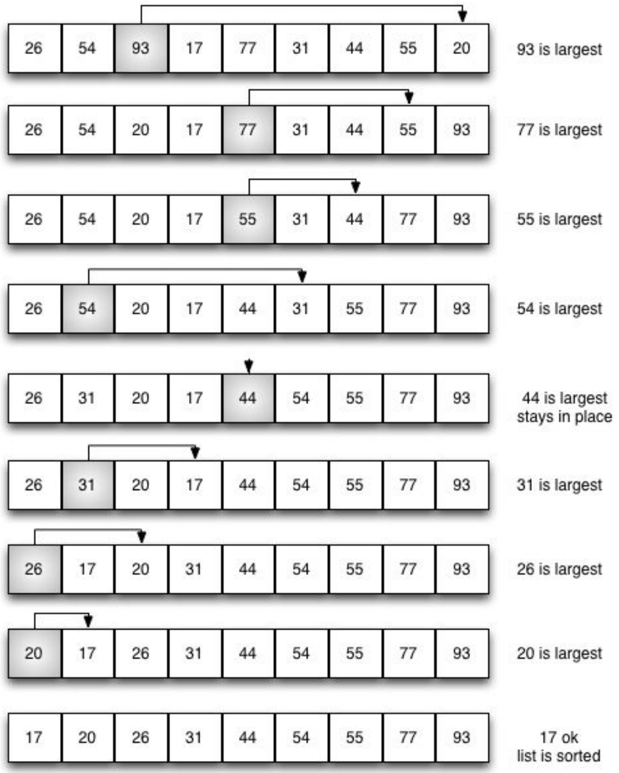
- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort

Bubble Sort



54	26	93	17	77	31	44	55	20	Exchange
26	54	93	17	77	31	44	55	20	No Exchange
26	54	93	17	77	31	44	55	20	Exchange
26	54	17	93	77	31	44	55	20	Exchange
26	54	17	77	93	31	44	55	20	Exchange
26	54	17	77	31	93	44	55	20	Exchange
26	54	17	77	31	44	93	55	20	Exchange
26	54	17	77	31	44	55	93	20	Exchange
26	54	17	77	31	44	55	20	93	93 in place after first pass

Selection Sort



Data Structure & Algorithms

	How Do We Hold Data?	How Can We Ask Data Structures To Do More Work?	What Can We Do With Data Structures?
Easy, Linear Problems	Arrays, Lists	Sorting algorithms like Merge Sort	Find minimum value
Complicated	Trees	Rotations	DFS, BFS
NP Hard Problems	Graphs	Alpha Beta Pruning	MCTS, Beam Search