DATA STRUCTURES & ALGORITHMS IN DATA SCIENCE INTERVIEWS

OCTOBER 25, 2018

Columbia Data Science Society

AGENDA

BIG-O

- Time complexity
- Space complexity

DATA STRUCTURES

- Array
- Linked List
- Stack
- Queue
- Binary Tree
- Hash tables

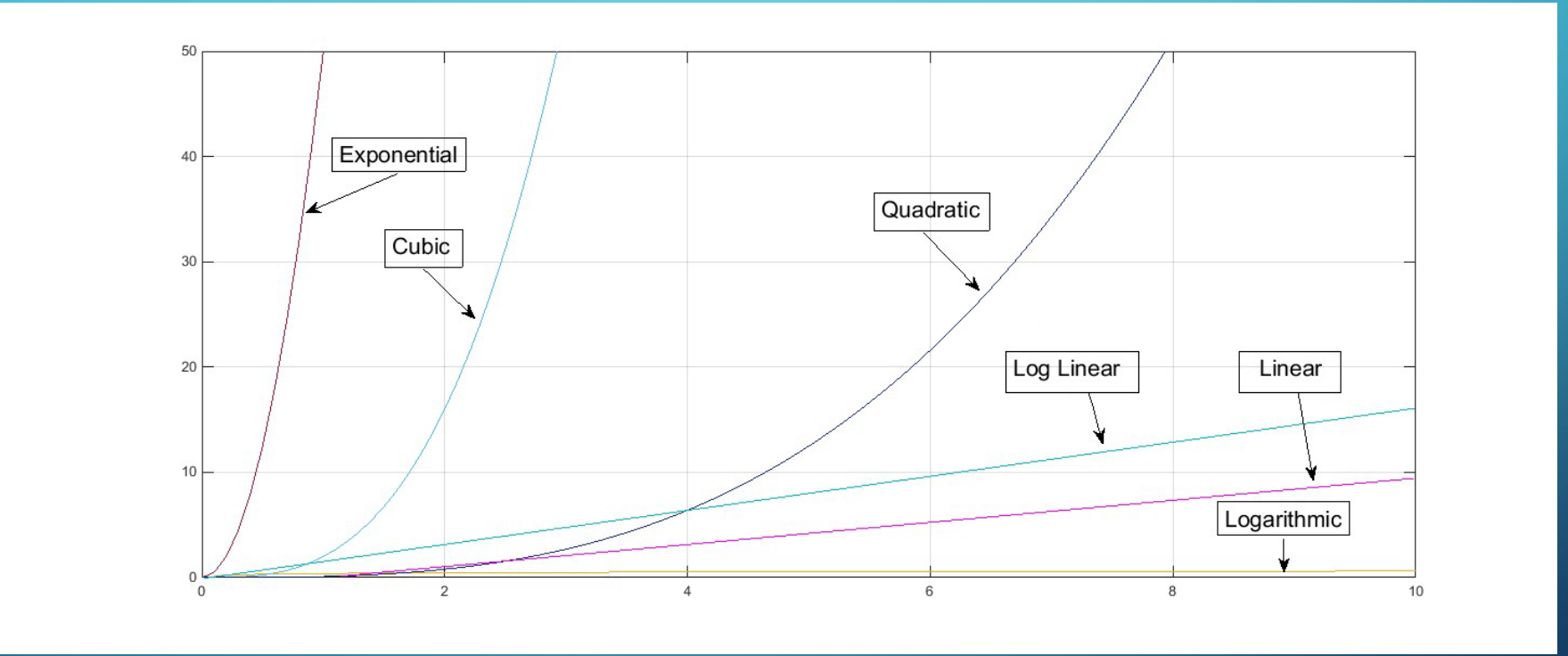
ALGORITHMS

- Searching
- Sorting
- Recursion & Dynamic Programming

WHAT YOU SHOULD EXPECT

- Similar to Software Engineering questions, probably easier versions of it
- Write on the whiteboard, in whatever language you choose (probably Python)
- Asked to improve time/space complexity of your solution

BIG-O (TIME & SPACE)



- Time- does your algorithm scale?
- Space- how much extra memory do you need?

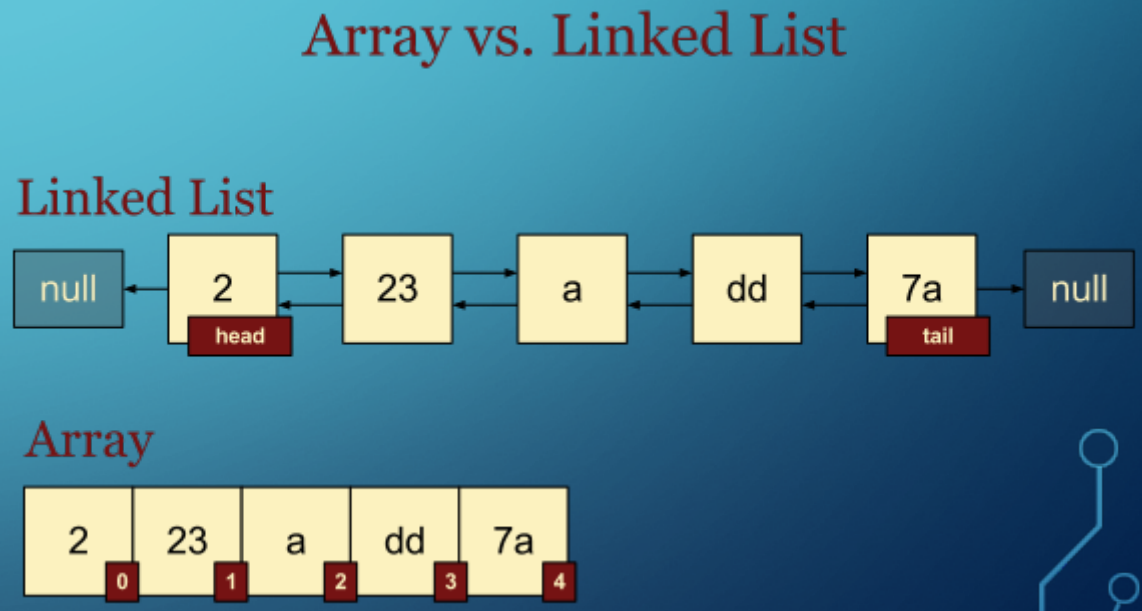
DETERMINE RUNTIME

```
01.  static void MoreExamples4(int[] arr1)
02.          {
03.              for (int i = 0; i < arr1.Length; i++)
04.              {
05.                  for (int j = 0; j < arr1.Length; j++)
06.                  {
07.                      Console.WriteLine(arr1[i] + ", " + arr1[j]);
08.                  }
09.              }
10.
11.              for (int k = 0; k < arr1.Length; k++)
12.              {
13.                  Console.WriteLine(arr1[k]);
14.              }
15.          }
```

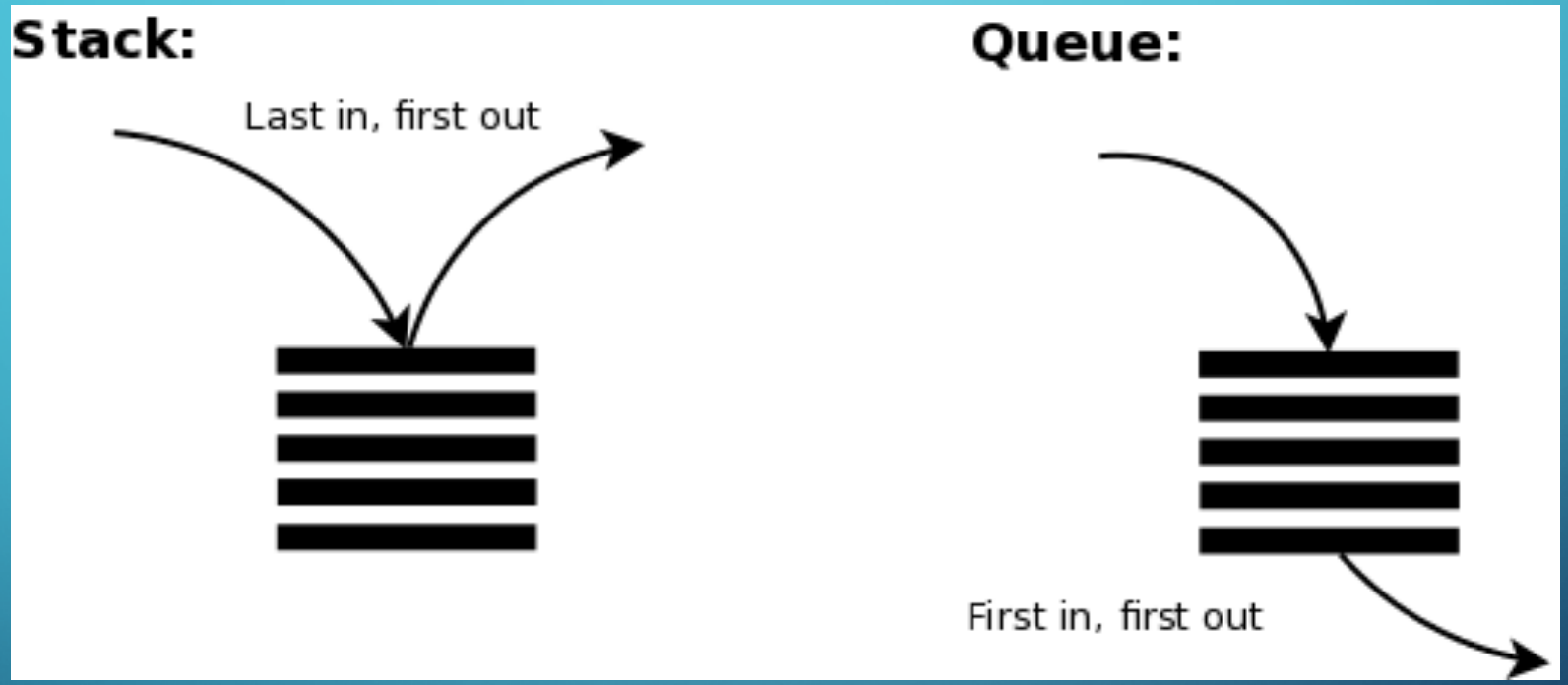
1. Runtime of lines 11-14
2. Runtime of just lines 3-9
3. Runtime of everything

ARRAY & LINKED LIST

- Access time vs. write operations (remove, add) trade-off
  - Array
    - access O(1) – call index
    - write op O(N) – need to shift elements
  - Linked List (singly/doubly linked)
    - access O(N) – iterate thru list to find element
    - Write op O(1) – relink as necessary



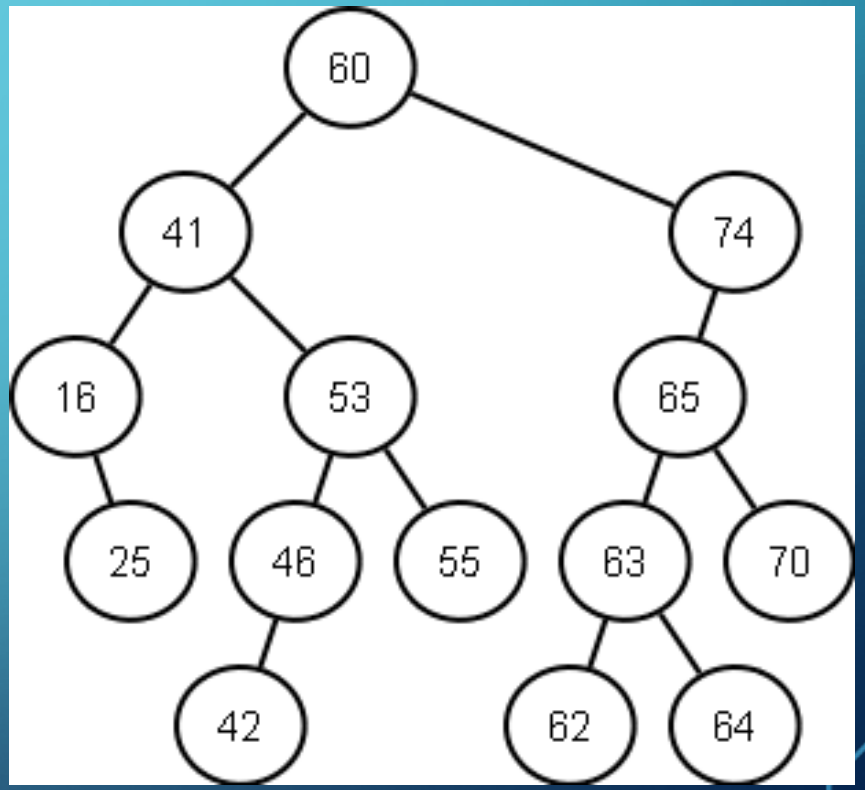
STACK & QUEUE



- Stack often associated with recursion (function calls, depth-first search)
- Queue (breadth-first search)

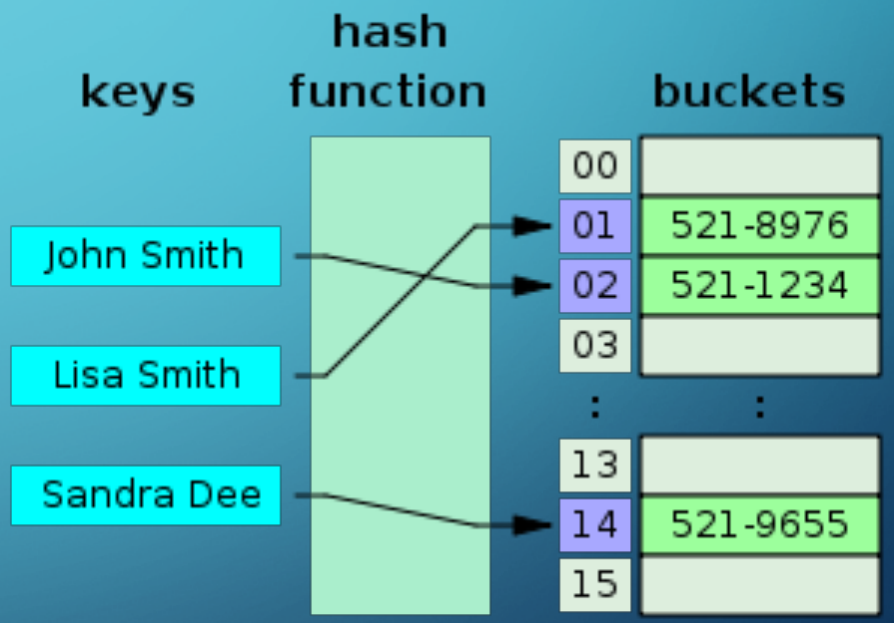
BINARY TREE

- Usually log(N) to find elements
- Useful for keeping track of max/min (heap)



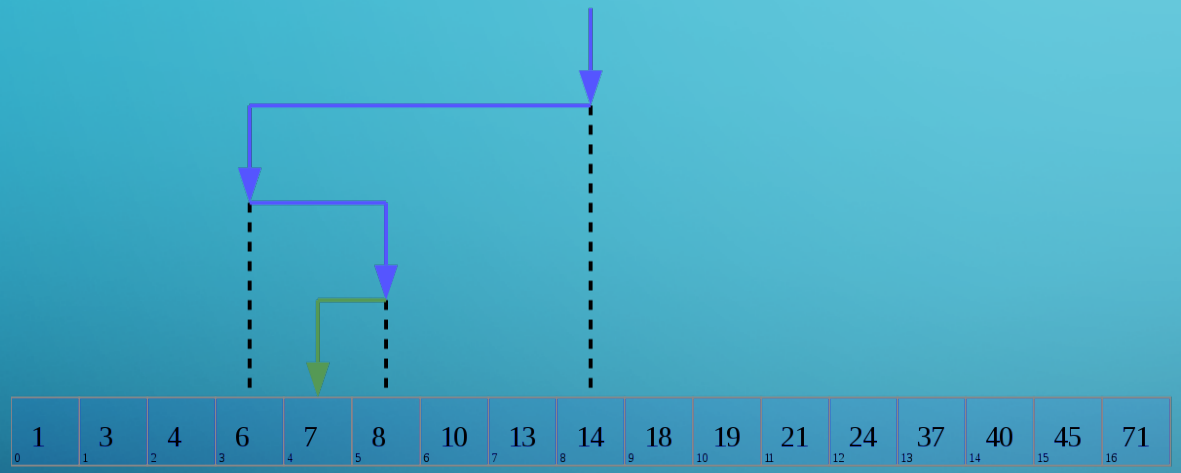
HASH TABLE

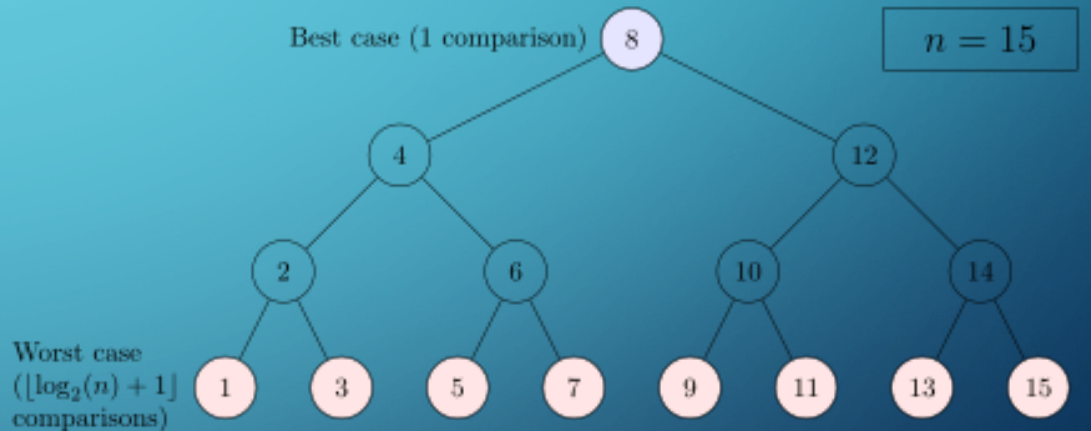
- Key-value pairs
- O(1) for lookup!
- Very useful – most often the answer



SEARCHING

- Binary Search- log(N)





- Can you think of a “better” way to search? Discuss tradeoffs

SORTING

- Many different algorithms
  - O(n^2)- Selection, insertion sort, quicksort worst
  - O(n logn)- Quicksort average, Mergesort
  - O(kn) – radix sot (takes advantage of finite number of bits)
- Just know the runtimes and be able to use it appropriately

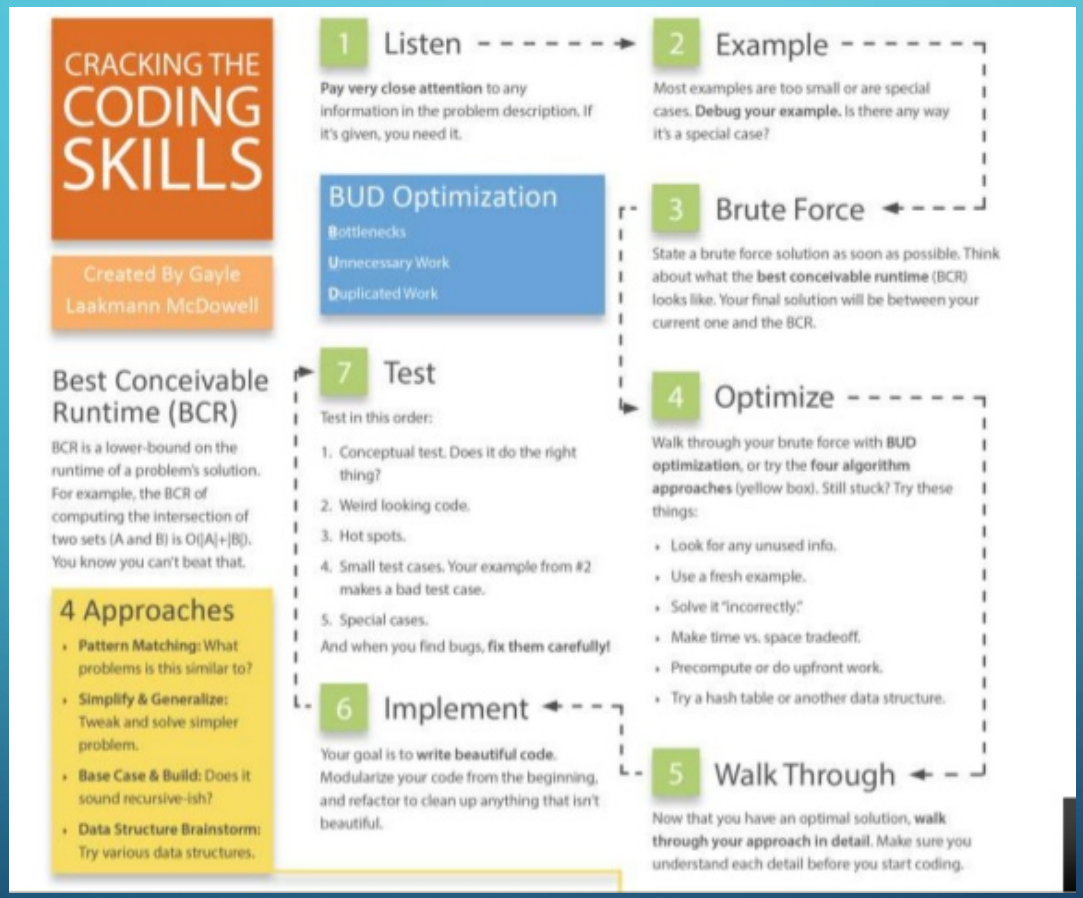
RECURSION & DYNAMIC PROGRAMMING

- Can be tricky to wrap your head around at first, but the only way to get comfortable is to see a lot of examples and practice – draw out the function call stack and optimize
- Classic example: Fibonacci numbers

```
fibo(i):
        if(I == 0) return 0
        if(I == 1) return 1
        return fibo(i-1) + fibo(i-2)
```

What is the runtime of fibo()? How can we improve our algorithm?

HOW TO APPROACH CODING PROBLEMS



CRACKING THE CODING SKILLS

Created By Gayle Laakmann McDowell

1 Listen

Pay very close attention to any information in the problem description. If it's given, you need it.

2 Example

Most examples are too small or are special cases. **Debug your example.** Is there any way it's a special case?

3 Brute Force

State a brute force solution as soon as possible. Think about what the **best conceivable runtime** (BCR) looks like. Your final solution will be between your current one and the BCR.

BUD Optimization

Bottlenecks

Unnecessary Work

Duplicated Work

4 Optimize

Walk through your brute force with **BUD optimization**, or try the **four algorithm approaches** (yellow box). Still stuck? Try these things:

- Look for any unused info.
- Use a fresh example.
- Solve it "incorrectly."
- Make time vs. space tradeoff.
- Precompute or do upfront work.
- Try a hash table or another data structure.

5 Walk Through

Now that you have an optimal solution, **walk through your approach in detail**. Make sure you understand each detail before you start coding.

6 Implement

Your goal is to **write beautiful code**. Modularize your code from the beginning, and refactor to clean up anything that isn't beautiful.

7 Test

Test in this order:

1. Conceptual test. Does it do the right thing?
2. Weird looking code.
3. Hot spots.
4. Small test cases. Your example from #2 makes a bad test case.
5. Special cases.

And when you find bugs, **fix them carefully!**

Best Conceivable Runtime (BCR)

BCR is a lower-bound on the runtime of a problem's solution. For example, the BCR of computing the intersection of two sets (A and B) is O(|A|+|B|). You know you can't beat that.

4 Approaches

- **Pattern Matching:** What problems is this similar to?
- **Simplify & Generalize:** Tweak and solve simpler problem.
- **Base Case & Build:** Does it sound recursive-ish?
- **Data Structure Brainstorm:** Try various data structures.

From Cracking the Coding Interview website (http://www.crackingthecodinginterview.com/uploads/6/5/2/8/6528028/cracking_the_coding_skills_-_v6.pdf)

SAMPLE PROBLEMS

1. Given a string input, figure out if it's a palindrome (ex: racecar, tacocat)
2. Given a string input, figure out if all the parentheses () {} [] are balanced
3. Given an integer array and a number, find pairs in the array whose sum is equal to the input number

- Follow the steps from previous slide
  - Make sure you really understand the question: question 3 asks for pairs, not just one pair. Ask clarifying questions to the interviewer. Are we treating (1,3) and (3,1) pairs differently?
  - Give yourself different examples- question 1, for example: "{()}" "[(])" "{{{}}]" "{"
  - When you test your code, think about edge cases (null input, one character input, negative numbers, etc)

RESOURCES

- Cracking the coding interview
- Practice, practice, practice
  - Online: Hackerrank, Leetcode
  - Offline: do mock interviews with friends
- There are millions of articles on software engineering interview prep- maybe read a few

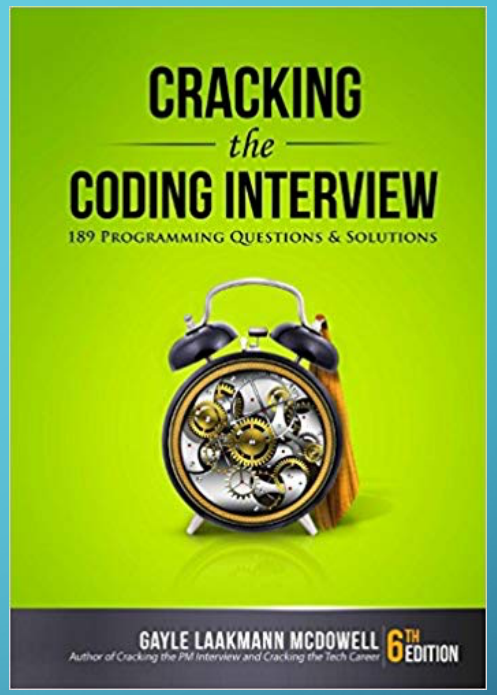


IMAGE CREDITS

- http://cs.lmu.edu/~ray/images/bstexample.png
- https://cdn-images-1.medium.com/max/1600/1*a3_Y3J907tIAPA5gTRFMNg.png
- https://4cawmi2va33i3w6dek1d7y1m-wpengine.netdna-ssl.com/wp-content/uploads/2018/07/Computer-science-fundamentals_6.1.png
- http://www.stoimen.com/blog/wp-content/uploads/2012/06/0.-Arrays-vs.-linked-list.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/8/83/Binary_Search_Depiction.svg/1200px-Binary_Search_Depiction.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/a/aa/Binary_search_complexity.svg/440px-Binary_search_complexity.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/5/58/Hash_table_4_1_1_0_0_1_0_LL.svg/240px-Hash_table_4_1_1_0_0_1_0_LL.svg.png
- https://www.codeproject.com/KB/recipes/1203514/MoreExample4.PNG