ALGORITHMS & DATA STRUCTURES SERIES

What To Expect

- Bi-weekly series for the next 6 months
- Beginner and Intermediate track
- A concept every meetup
- Take home challenges for you to solve before the next meetup
- Communication will be managed via slack
  https://bit.ly/wwcodelagos-slack
  join #algorithms-and-datastructures channel
- Coding Challenges during the course of the series
- Consistent participants will be paired with mentors for mock interviews

Schedule

Date	Concept
March 5	Getting Started
March 19	Strings and Arrays
Apr 16	Sorting and Searching
Apr 30	Hash Tables
May 14	**Coding Challenge 1**
May 28	Linked Lists
June 11	Stacks and Queues
June 25	Recursion
July 9	Trees and Graphs
July 23	Dynamic Programming
Aug 6	**Coding Challenge 2**
Aug 20	Closing

AGENDA

- Check-in
- High level overview of concept
- White board breakout sessions

NB: When joining the call, kindly edit your name and append **B or I** to indicate beginner or intermediate e.g: Hope – B, Ebonyhope - I

Housekeeping Rules

During The Meetup

- Keep your video off
- Stay muted when when not speaking
- Feel free to post your question in the chat section
- Want to ask a question? use the hand raise emoji
- Interact in the break out room

On The Slack Channel

- Make it a safe space for everyone
- Post your questions, progress and achievements
- Be respective of your mentors time

Why Are You Doing This

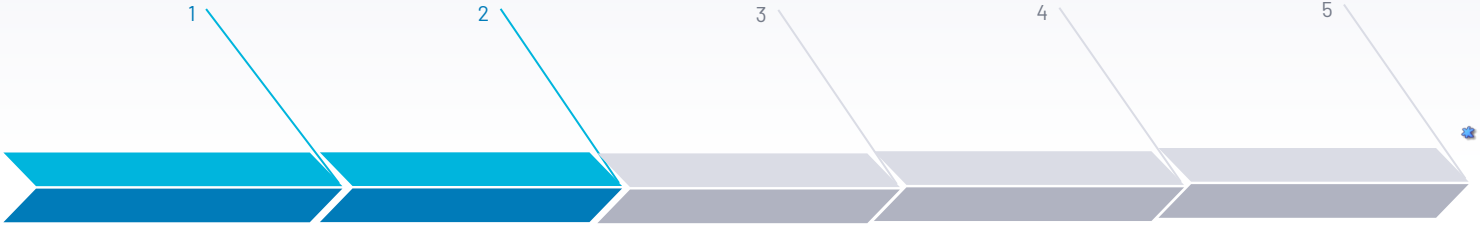
- Get comfortable with solving questions
- Accountability
- Practice for interviews
- Find a community
- Make new friends

https://bit.ly/why-are-you-doing-this

WHEN YOU FEEL LIKE QUITTING

REMEMBER WHY YOU STARTED

Typical Interview Process



CV and Application

- LinkedIn
- Github
- Cover letter

Behavioural Interview

- Story telling
- Soft Skills
- STAR

Coding Test

- Hackerrank
- Codility
- Coderpad

Phone Screen

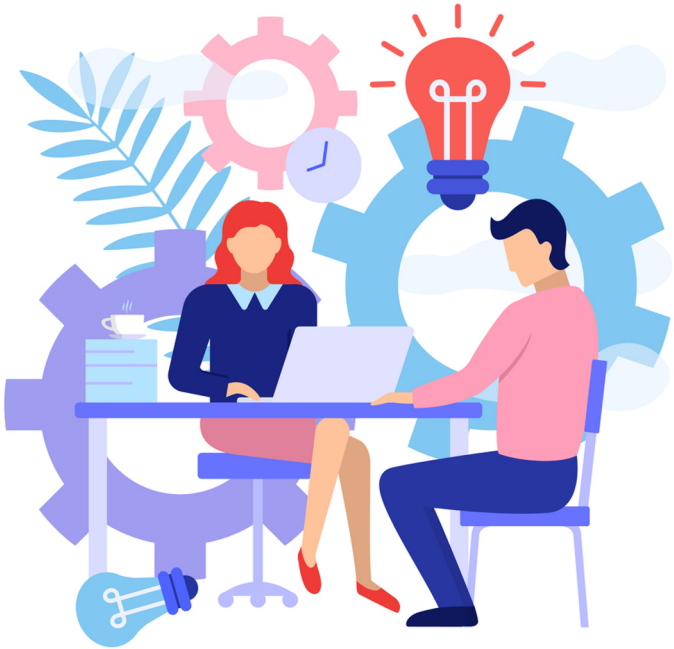
- Google Docs
- Repl.it

Onsite

- 3 to 5 interviews
- White board

What Is Been Tested

- Analytical/Problem solving skills
- Coding skills
- Technical knowledge and computer science fundamentals
- Experience
- Culture fit/ Communication skills



Problem Solving Process

- Repeat the problem back to the interviewer
- Ask clarifying questions
- Draw out your own examples
- Brute Force Solution
- Optimize
- Implement
- Test

state the complexity

Problem https://leetcode.com/problems/first-bad-version/

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

Example 1:

Input: n = 5, first bad = 4

Output: 4

Example 2:

Input: n = 1, bad = 1

Output: 1

Tips and Tricks

- Research about the company and their interview process
- Test your devices and ensure they work properly
- Think out loud
- Take hints
- Use a language that is readable and easy to use
- Ask for feedback after unsuccessful interviews

Time and Space Complexity

Rate of growth

For a given input (n),

Time Complexity: how long it does an algorithm runs for

Space Complexity: how much space or memory does an algorithm uses

Big-O notation:

upper bound, worst case scenario

Big-Ω (Big-Omega) notation:

lower bound, best case scenario

Big-θ (Big-Theta) notation:

average case scenario

Typically we solve for the worst case scenario, Big O

Common Complexities

For a given input (n),

- constant growth - **O(1)** Runtime is constant, regardless of the size of n
- logarithmic growth – **O(log n)** Runtime grows logarithmically in proportion to the size of n
- linear growth – **O(n)** Runtime grows directly in proportion to the size of n
- quadratic growth – **$O(n^2)$** Runtime grows exponentially with respect to n

```
public static int sum(int[ ] array){

        int n = array.Length;
        int sum = 0;

        for(int i=0;i<n;i++){
                sum += arr[ i ];
        }

    return sum;
}
```

```
public static boolean twoSum(int [ ] arr, int sum){

        for (int i = 1; i< arr.Length; i++){
            int firstNum = arr[ i ];
            for (int j = 0; j < i; j++){
                    if (firstNum +  arr[ j ] == sum){
                    return true;
                }
            }
      }
            return false;
}
```

```
public static void print(int[ ] arr){

    for(int i = 0;  i<arr.Length;  i*=2){
        Console.WriteLine("C# is sweet");
    }
}
```

```
public static void minmax(int[ ] arr){
        int min = arr[ 0 ];
        for (int i = 0; i < arr.Length; i++){
           min = Math.Min(min, arr[ i ]);
        }

        int max = arr[ 0 ];
        for (int i = 0; i < arr.Length; i++){
           max = Math.Max(max, arr[ i ]);
        }
}
```

THANKS!

Any questions?

You can find me at:

- Slack: @Oluwalolope Hope
- Twitter: @IAMebonyhope