# Technical Interviews

By Rish Jain and Muthu Palaniappan

# What we will cover?

- Intro to Technical Interviews
- Technical Interview Structure
- How can I prepare?
- How should I act?
- Practice!

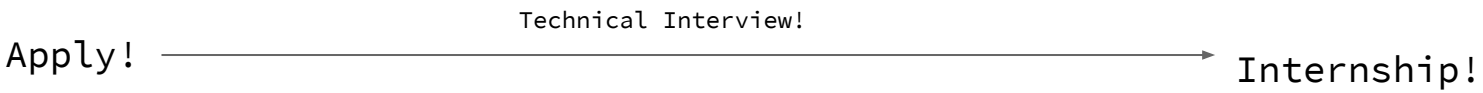
# What are Technical Interviews?

- Technical Interviews are used to test understanding of topics like: Data Structures and Algorithms!
  - CS32 is about 90% of what you need to know, just need practice
  - Time Complexity(Big O), Data Structures, and Algorithms
- It’s usually a one on one interview where they will ask you questions testing your critical thinking, technical skills, and problem-solving skills.
- Internships and jobs will have you (usually) do one before they give you an offer

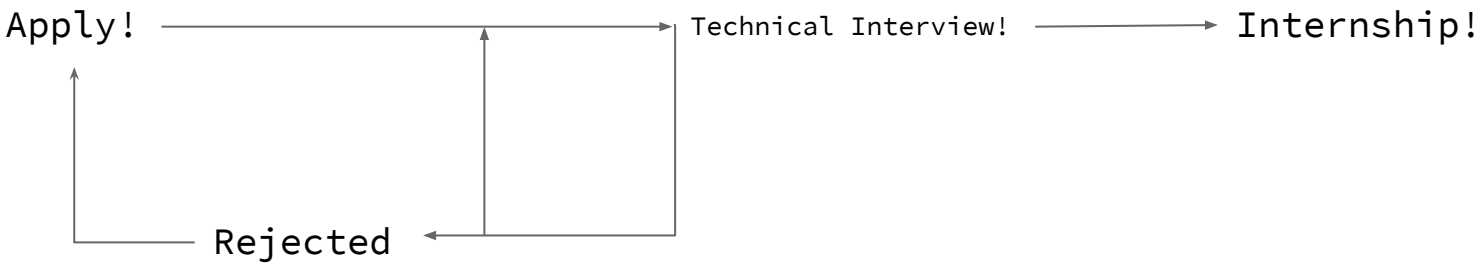
# What is the process to Internships?

1) Create a resume!
2) Apply to places!
3) (Sometimes) Coding Challenge
    a) These are sometimes used to check your basic understanding
4) Interviewer Reaches out
5) Technical Interview
    a) Potentially up to 3 rounds!
6) Get Offer!

# What it seems like getting a Internship Should be



# What it Actually is



# Technical Interview Structure

45-60 minutes total

- Behavioral questions (5-10 min)
  - Resume
  - Skills
- Coding (30-45 min)
  - 1-2 questions
- Questions (5 min)

# How Can I prepare?

1) LeetCode, HackerRank, BinarySearch.com
    a) You might not know how to do the optimal solution for certain problems
2) Carey's Slides
3) *Cracking the Coding Interview*
4) Researching the Company

# Steps to take for a Coding Problem

1) Clarify the Problem
    a) Ask Questions to understand how edge cases would work or understand what they want
2) Propose what you're thinking of doing or using
3) Write out code
    a) Create functions if you're not sure how something should work
    b) For example, if I need to see if a value exists in a linked list deal with that later.
4) Analyze time complexity and space complexity
5) Usually get a theoretical follow up question
    a) i.e) What would be a potential issue with a recursion method?

# General Tips to apply on coding problems

- Always speak what you're thinking or doing (like a sports broadcaster)!
- If you're not sure what to do, just start by doing what you know
  - The interviewers will help you if you get stuck with hints!
- Think back to data structures you've utilized before, algorithms/other examples used with them, etc...

# Link to Lectures done by Old TA

Lectures: Jack Gong

# Practice!

# Example 1: Reverse Integer

Given a number, return the reverse.

```
int reverse(int num)
```

Ex:

    Input: 123, Output: 321

    Input: 8, Output: 8

# Solution

```
#include <iostream>
using namespace std;

int reverse(int n) {
  int output = 0;
```

```
  while(n != 0) {
    output = output*10 + n%10;
    n /= 10;
  }
  return output*sign;
}
```

```
int main() // for testing
{
  cout << reverse(12345) << endl;
}
```

# Example 2: Rotate a vector by k

Write a function rotatebyk(vector<int> vec, int k, int n) that rotates a vector vec of size n to the left by an amount k

Ex:
Input: vec=<1,2,3,4,5>, k=2, n=5
Output: <3,4,5,1,2>

What are some possible questions you would ask the interviewer?

Unfortunately, we aren't too sure about an optimal solution. So, to start, we approached this problem with brute force. Try writing code that uses a brute force approach

# Solution: Rotate by 1, k times

```
/* Rotate by one logic:
Rotate vector [1,2,3,4,5]
<1,2,3,4,5> => <2,3,4,5,1> */
void leftRotatebyOne(vector<int> vec, int n)
{
    int temp = vec[0];
    int i;
    for (i = 0; i < n - 1; i++)
        vec[i] = vec[i + 1];

    vec[n-1] = temp;
}

Void rotatebyk(vector<int> vec, int k, int n)
{
    for (int i = 0; i < k; i++) {
        leftRotatebyOne(vec, n);
    }
}
```

Hmmm, how could we optimize this? This currently just rotates the values k times.

# Optimal Solution

```
void rotatebyk(vector<int> vec, int k, int n)
{
    /* To handle if k >= n */
    k = k % n;
    int g_c_d = gcd(k, n);
    for (int i = 0; i < g_c_d; i++) {
        /* move i-th values of blocks */
        int temp = vec[i];
        int j = i;

        while (1) {
            int b = j + k;
            if (b >= n)
                b = b - n;

            if (b == i)
                break;

            vec[j] = vec[b];
            j = b;
        }
        vec[j] = temp;
    }
}
```

Note: This was a hard solution to come up with, and it's okay to not always get the optimal solution. Recruiters are looking at how you approach this problem rather than how many algorithms you have memorized

# Questions?