LeetCode Workshop

PRESENTED BY WOMEN IN COMPUTER SCIENCE AT DAVIS

CLUB finance COUNCIL

asucd

Technical Assessment

1. Listen and read actively
2. Draft a simple example
3. Explain the brute force
4. Optimize
5. Walk though your approach
6. Write beautiful code!
7. Test cases

1. Listen & Read Actively

- Pay very close attention to any info in the problem description
- It is okay to ask the interviewer to repeat the question and ask clarifying questions if you don’t fully understand it

2. Draft a Simple Example

- Form a simple example
  - a simple input to the algorithm that demonstrates the intended behavior of the program
  - not an edge case, not something that can't demonstrate the main behavior of the code
- Walk through how you could get the output with the simple input that you created

3. Brute Force

- Get a brute force solution ASAP!
- Brute force is the most straightforward method that usually does not take efficiency (time and space complexity) into consideration. Relies on pure computing power.
- At this step, don’t worry about having to develop an efficient algorithm just yet
- State a naive algorithm and its runtime

4. Optimize

- Walk through your brute force with "BUD" optimization
  - "BUD": Bottlenecks, Unnecessary and Duplicated work
- Look for any unused info
- Solve it manually with valid example input, then reverse engineer your thought process, how did you solve it?
- Solve it with "incorrect" input and think about why the your code fails. Can you fix those issues?
- Compare the tradeoffs between time and space

5. Walkthrough

- **Now that you have an optimal solution, walk through your approach in detail.**

6. Write Beautiful Code!

- Modularize your code from the beginning and refactor to clean up anything that is bulky
- Choose a coding style and stick with it!
- Properly indent your code
- Make meaningful comments if necessary

7. Test

Test in this order:

- Conceptual test: Walk through your code like you would for a detailed code review
- Small test cases: Use a minimal case to show that your algorithm behaves as expected
- Special cases and edge cases: Test extreme cases such as invalid input, unexpected behaviors
- Hot spot cases: Fix simple things like arithmetic and null nodes.

When you find bugs, fix them carefully and explain what/why you changed it!

What you need to know:

Data structures

- Hash tables, Linked lists, Stacks, Queues, Trees, Graphs, Vectors, Heaps

Algorithms

- Quick sort, Merge sort, Binary search, BFS, DFS

Concepts

- Big-O time, Big-O space, Recursion

What's Leetcode?

What is LeetCode?

- An online platform, well-known to the developer community, that can be used to practice programming skills for technical interviews
- 800+ questions with multiple solutions, discussion boards, ranked by difficulty
- LeetCode-style questions are known as programming problem solving questions

Time to Leetcode!

Please get into groups of 5-6 people.

Question 1:

Given an array of integers nums, calculate the pivot index of this array.
The pivot index is the index where the sum of all the numbers strictly to the left of the index is equal to the sum of all the numbers strictly to the index's right. If the index is on the left edge of the array, then the left sum is 0 because there are no elements to the left. This also applies to the right edge of the array. Return the leftmost pivot index. If no such index exists, return -1.

Input: nums = [1,7,3,6,5,6]
Output: 3
Explanation:
The pivot index is 3.
Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11
Right sum = nums[4] + nums[5] = 5 + 6 = 11

https://leetcode.com/problems/find-pivot-index/

Question 1 Solution:

```
int pivotIndex(int* nums, int numsSize){
    int sum = 0;
    int leftsum = 0;
    for(int j = 0; j < numsSize; j++) sum += nums[j];
    for(int i = 0; i < numsSize; i++){
        if(leftsum == sum - leftsum - nums[i]) return i;
        leftsum += nums[i];
    }
    return -1;
}
```

Question 1 Solution:

```
int pivotIndex(int* nums, int numsSize){
    int sum = 0;
    int leftsum = 0;
    for(int j = 0; j < numsSize; j++) sum += nums[j];
    for(int i = 0; i < numsSize; i++){
        if(leftsum == sum - leftsum - nums[i]) return i;
        leftsum += nums[i];
    }
    return -1;
}
```

Given nums = [1,2,4,7,7]:

leftsum = 1+2+4 = 7
sum = 1+2+4+7+7 = 21
nums[3] = 7
if(leftsum == sum - leftsum - nums[3]): if (7 == 21 - 7 - 7)

Question 2:

Given an array, rotate the array to the right by k steps, where k is non-negative.

Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]

Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]

https://leetcode.com/problems/rotate-array/

Question 2 Solution:

```
class Solution {
    public:
        void rotate(int nums[], int n, int k)
        {
            if ((n == 0) || (k <= 0))
            {
                return;
            }

            // Make a copy of nums
            vector<int> numsCopy(n);
            for (int i = 0; i < n; i++)
            {
                numsCopy[i] = nums[i];
            }

            // Rotate the elements.
            for (int i = 0; i < n; i++)
            {
                nums[(i + k)%n] = numsCopy[i];
            }
        }
    };
```

Question 2 Solution:

```
class Solution {
    public:
        void rotate(int nums[], int n, int k)
        {
            if ((n == 0) || (k <= 0))
            {
                return;
            }

            // Make a copy of nums
            vector<int> numsCopy(n);
            for (int i = 0; i < n; i++)
            {
                numsCopy[i] = nums[i];
            }

            // Rotate the elements.
            for (int i = 0; i < n; i++)
            {
                nums[(i + k)%n] = numsCopy[i];
            }
        }
    };
```

Given nums = [1,2,3,4,5,6,7], n =7, k = 3
Final Output: [5,6,7,1,2,3,4]

For when i is:

0: nums[(0+3)%7] = numsCopy[0]
nums[3] = 1

[1,2,3,1,5,6,7]

1: nums[(1+3)%7] = numsCopy[1]
nums[4] = 2

[1,2,3,1,2,6,7]

....

Question 3:

Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring.

Input: s = "(()"
Output: 2
Explanation: The longest valid parentheses substring is "()".

Input: s = ")()())"
Output: 4
Explanation: The longest valid parentheses substring is "()()".

https://leetcode.com/problems/longest-valid-parentheses/

Question 3 Solution:

```java
public class Solution {

    public int longestValidParentheses(String s) {
        int maxans = 0;
        Stack<Integer> stack = new Stack<>();
        stack.push(-1);
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == '(') {
                stack.push(i);
            } else {
                stack.pop();
                if (stack.empty()) {
                    stack.push(i);
                } else {
                    maxans = Math.max(maxans, i - stack.peek());
                }
            }
        }
        return maxans;
    }
}
```

Input: ())(())

Stack:

-1

0, -1

-1

empty

2

3,2

4, 3, 2

5, 4, 3, 2

4, 3, 2

3, 2

0	1	2	3	4	5	6	7
(	)	)	(	(	(	)	)

push -1, max = 0

push 0, max = 0

pop 0, len = 1-(-1) =2, max =2

pop -1, max = 2

stack empty, max = 2

push 2, max = 2

push 3, max = 2

push 4, max = 2

push 5, max = 2

pop 5, len = 6-4=2, max = 2

pop 4, len = 7-3 = 4, max = 4

https://leetcode.com/problems/longest-valid-parentheses/solutions/127609/longest-valid-parentheses/

Additional Resources:

- neetcode.io
- Cracking the Coding Interview by Gayle Laakmann McDowell
- www.swecareers.com/mock-interviews

Thanks for coming!

Be sure to come out for our next meetings!

November 21, 2022: Networking 101 & Coffee Chat Reveals

November 28, 2022: Destress with Design Interactive & WiCS

CLUB finance COUNCIL

asucd