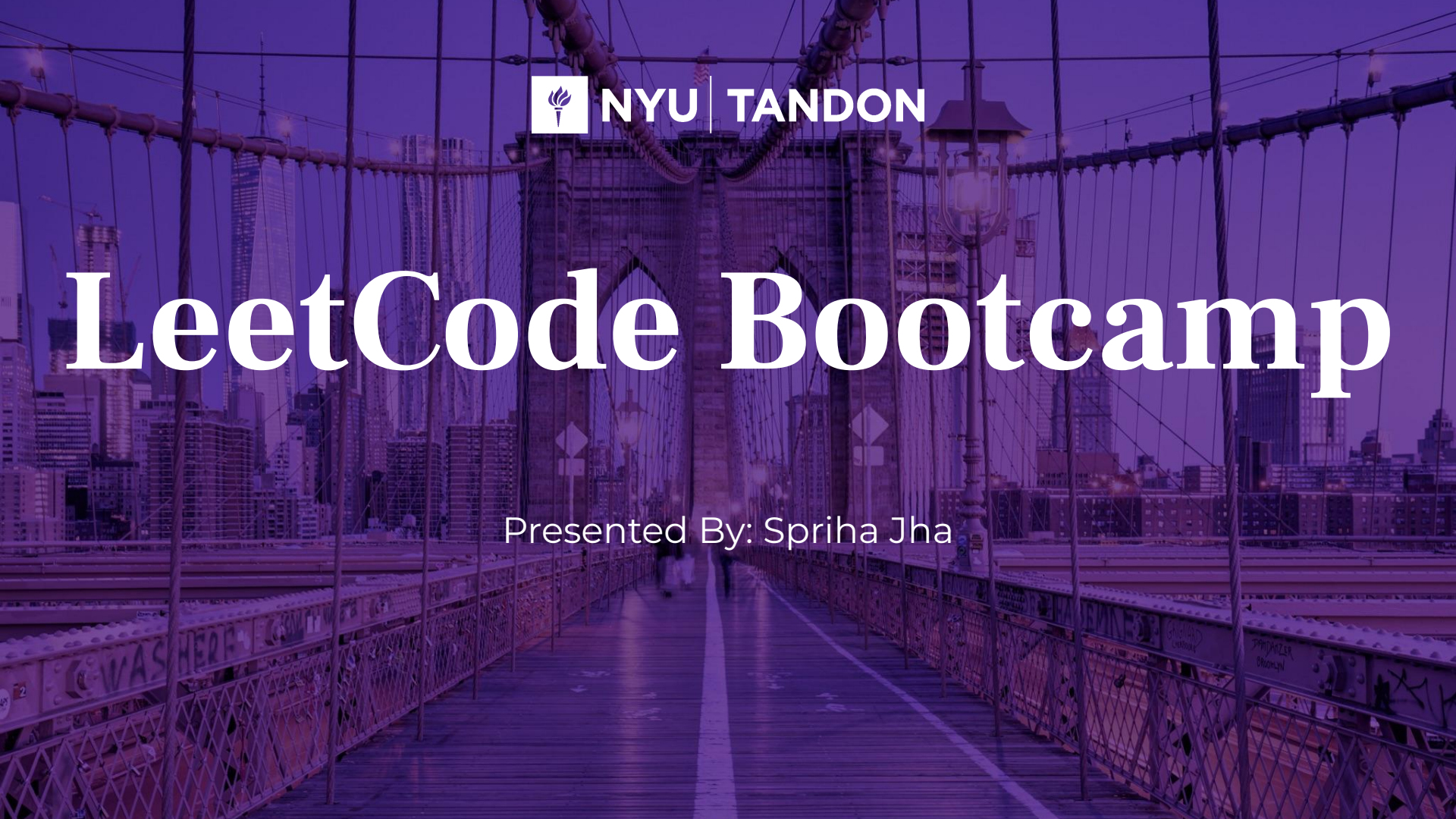

NYU TANDON
LeetCode Bootcamp
Presented By: Spriha Jha

Session Outline

01. LeetCode platform overview

02. Tech Interview Resources

03. Bootcamp Timeline

04. Introduction to List, Arrays & Sorting

05. Problem Sets

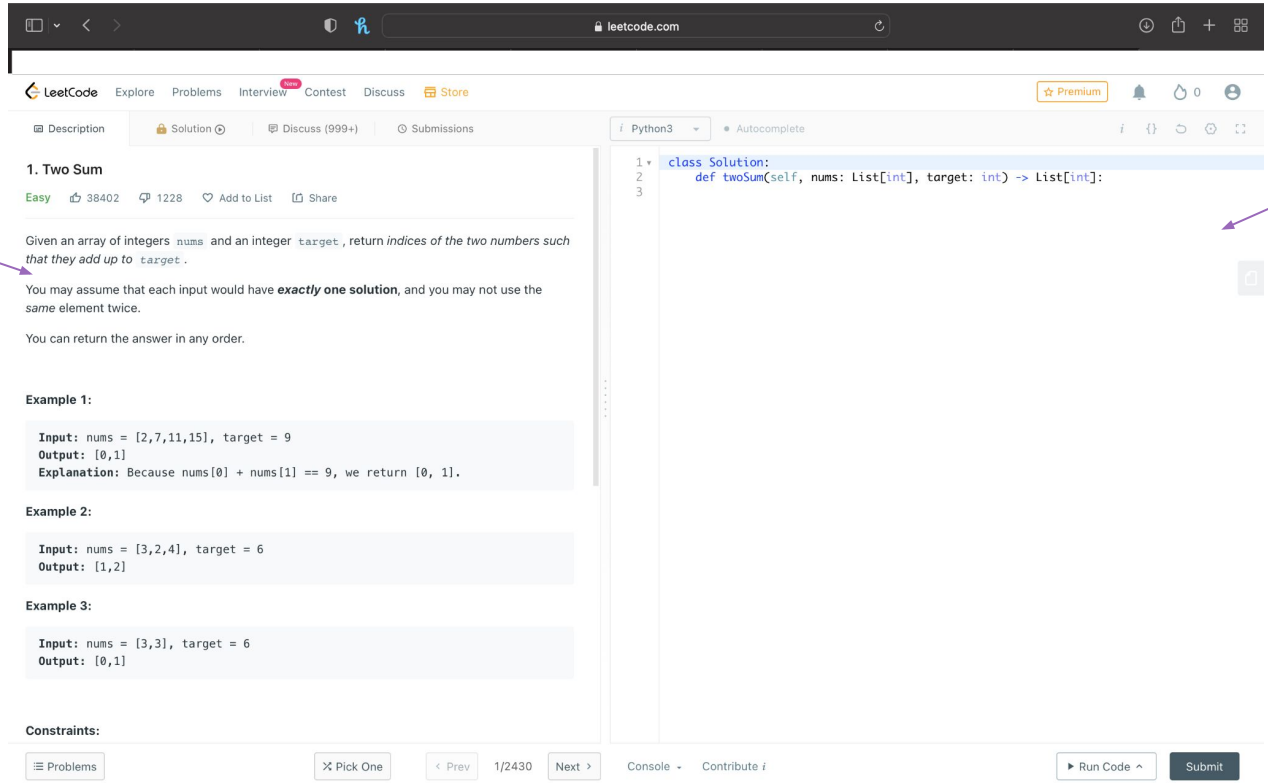
06. Debrief & Q/A

PART 01

LeetCode Platform Overview

Free Tier

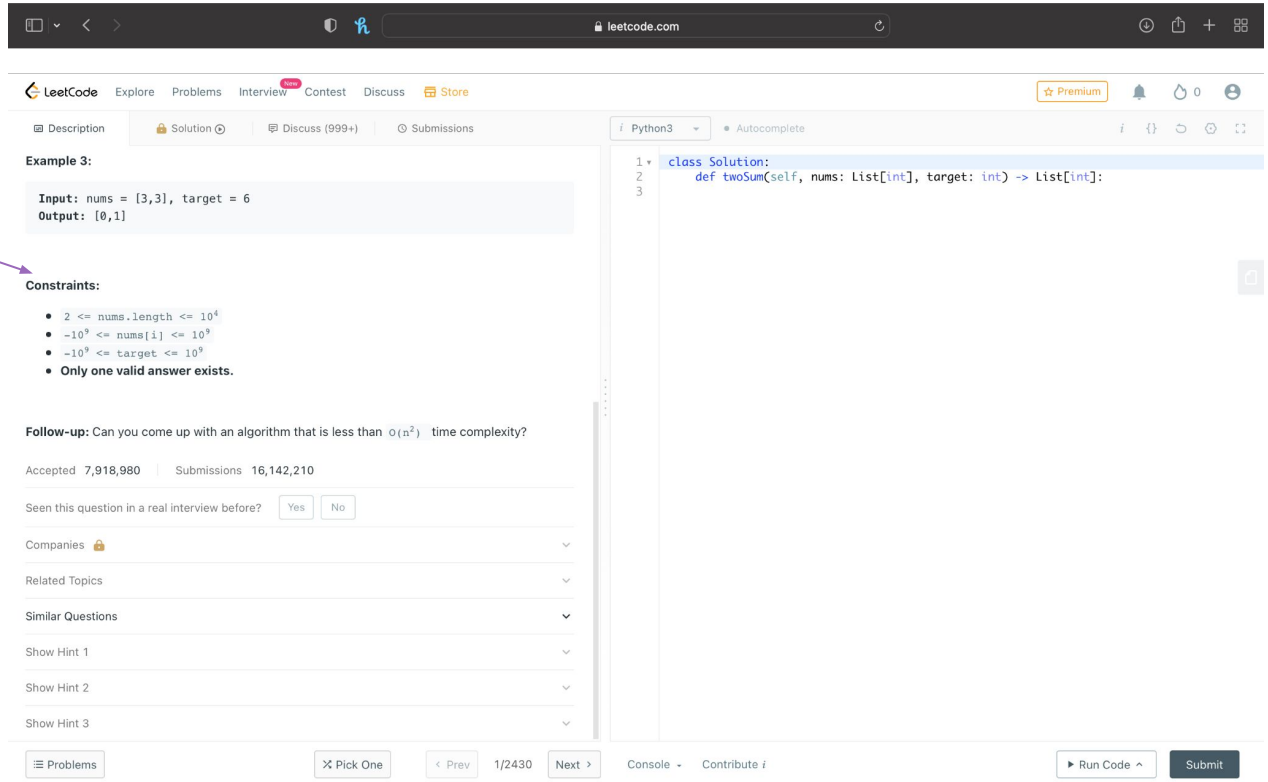
Problem Statement



Add your code here!

NYU TANDON

Identifying edge-cases



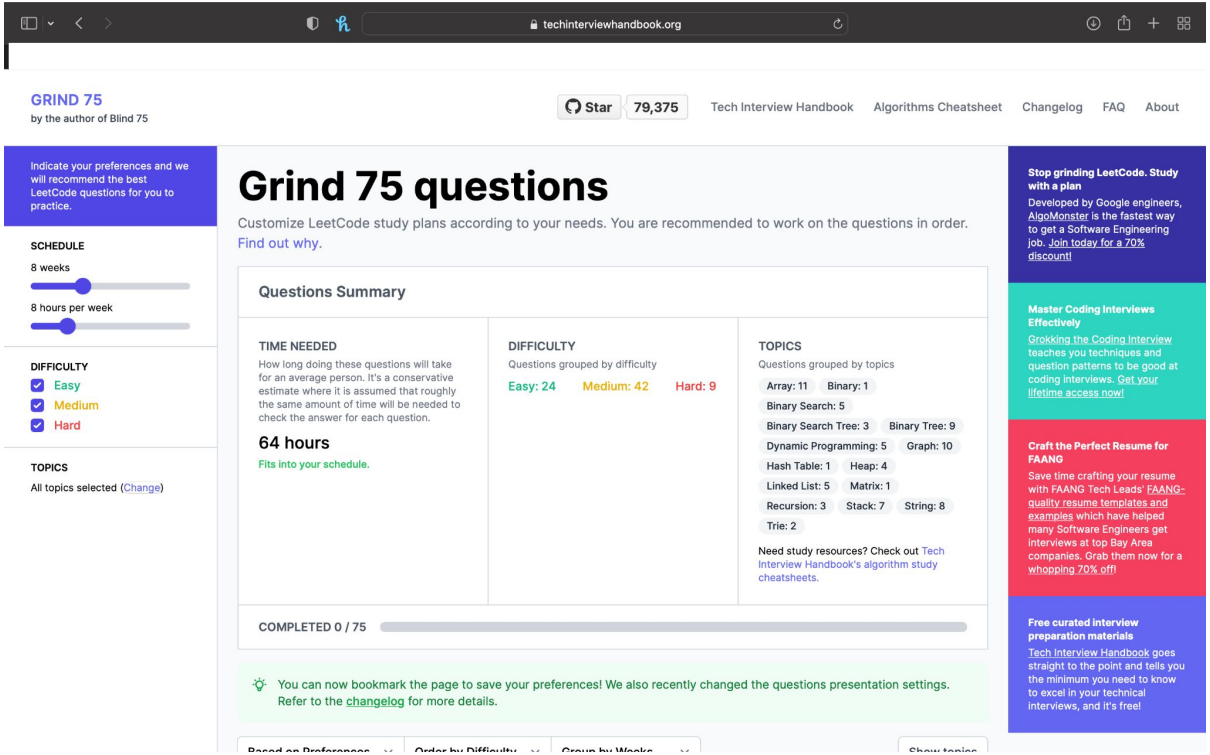
NYU TANDON

PART 02

Tech Interview Resources

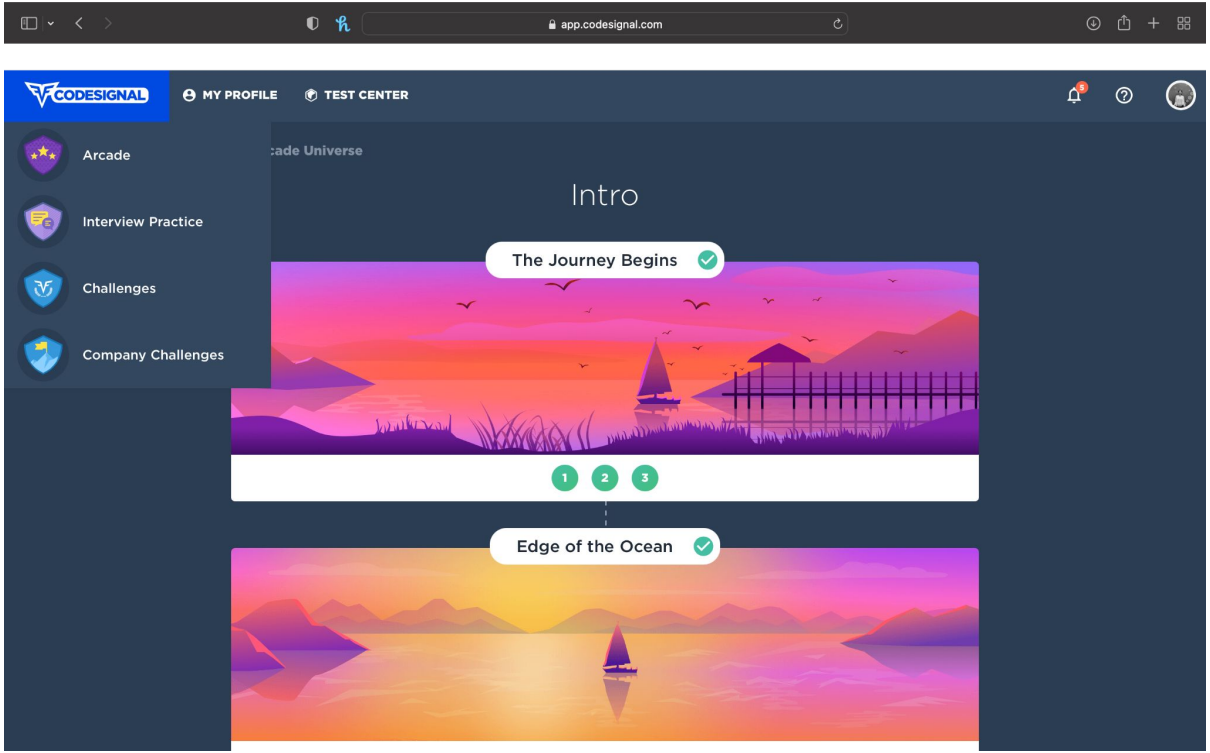
Grind75 | CodeSignal | GitHub

Tech Interview Handbook



PART 02

CodeSignal

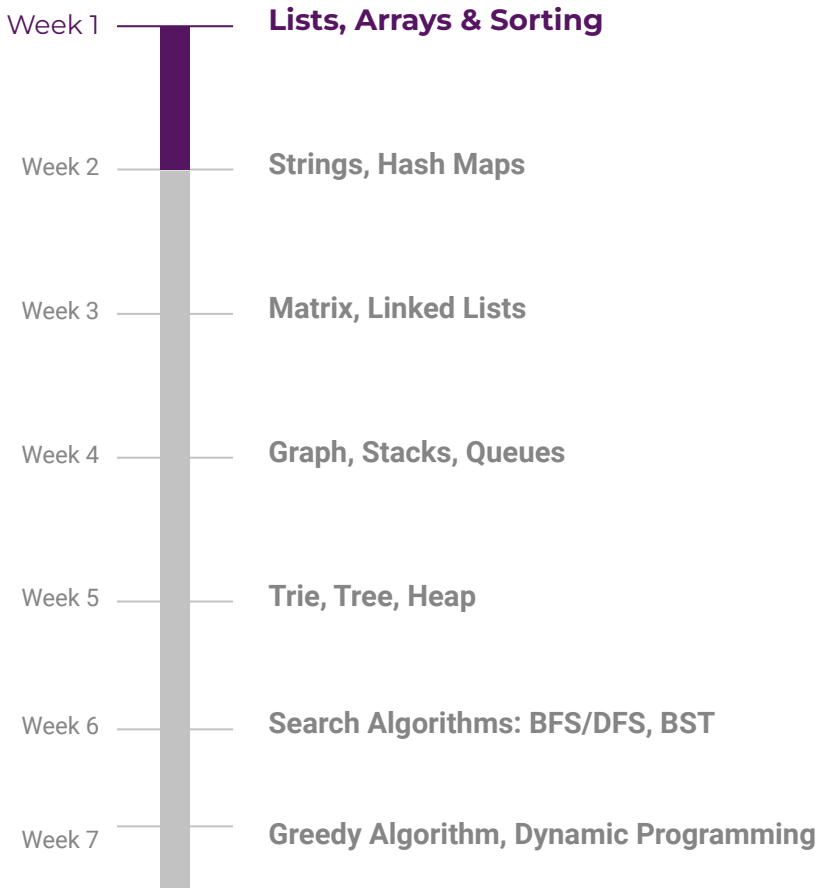


PART 03

Bootcamp Course Structure

PART 03

Bootcamp Timeline



NYU TANDON

PART 04

Introduction to Lists, Arrays & Sorting

Array

Container of elements of similar data type.

Stored at adjacent memory locations.

Disadvantages

Memory wastage.

Insertion and deletion of elements are costly.

Advantages

Mutable and not fixed in size, making them flexible.

Faster access of elements using the index value.

How to use?

Using the *array* module in Python.

Using the NumPy package arrays.

Built-in Methods

- len( )
- append( )
- clear( )
- count( )
- index( )
- insert( )
- pop( )
- remove( )
- reverse( )
- sort( )

PART 05

Problem Sets

Steps to approach the question:

Understand the problem

Take time to carefully read through the problem from start to finish is critical in finding the correct and complete solution to the problem in hand.

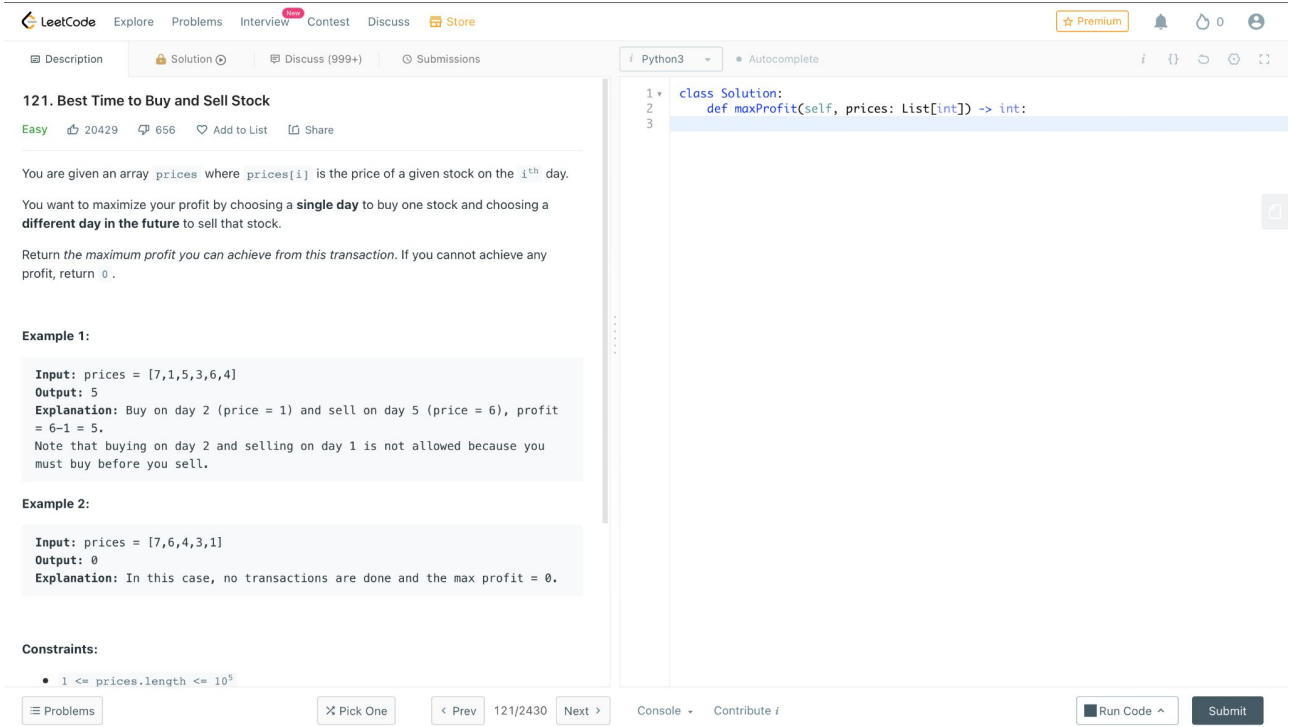
Code your solution

Map out your solution before you write any code. Avoid too much time trying to find the perfect solution. Validate your solution early and often.

Manage your time

Don’t forget, you have multiple questions to complete within a said time. Make sure you allocate enough time to carefully consider all problems.

Problem 1: Best time to Buy and Sell Stock



NYU TANDON

Approach 1: Brute Force

```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        max_profit = 0
        for i in range(len(prices) - 1):
            for j in range(i + 1, len(prices)):
                profit = prices[j] - prices[i]
                if profit > max_profit:
                    max_profit = profit

        return max_profit
```

Time complexity: $O(n^2)$, loop runs n(n-1)/2 times

Space complexity: O(1), only two variables *max_profit* and *profit*

Approach 2: Single Pass

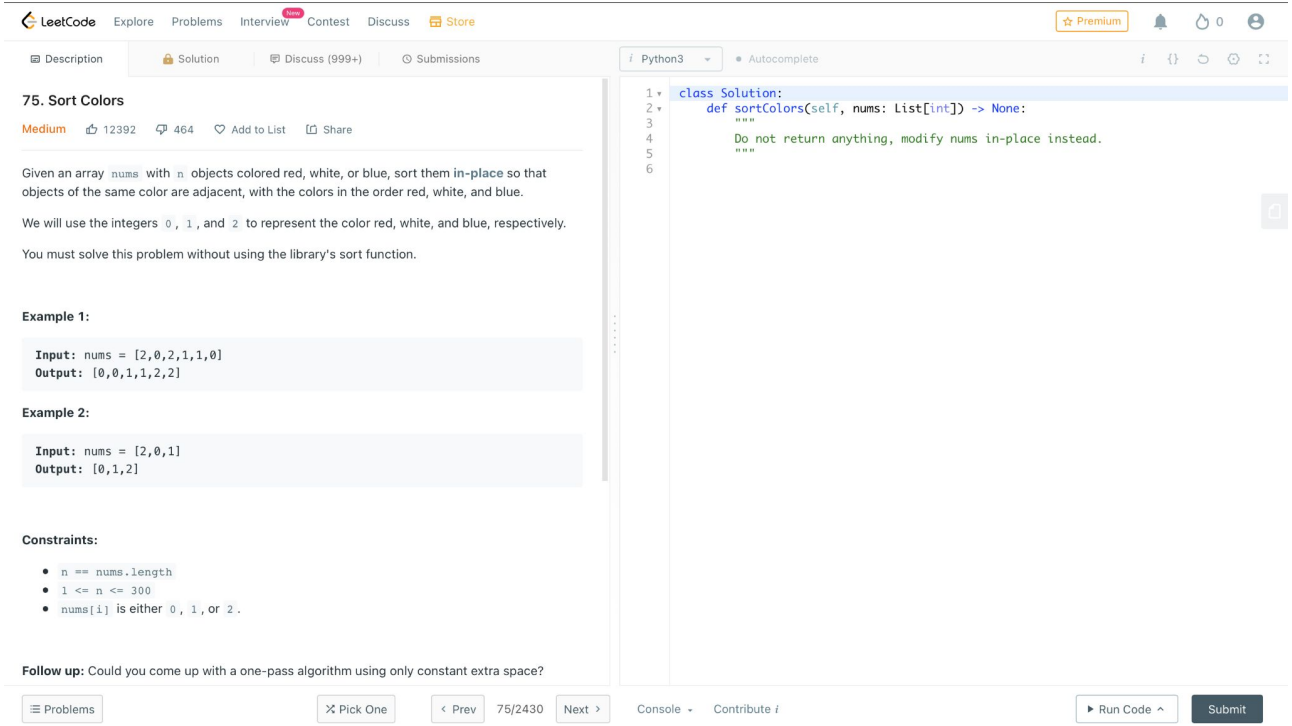
```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        min_price = float('inf')
        max_profit = 0

        for i in range(len(prices)):
            if prices[i] < min_price:
            min_price = prices[i]
            elif prices[i] - min_price > max_profit:
            max_profit = prices[i] - min_price
        return max_profit
```

Time complexity: O(n), only single pass is needed

Space complexity: O(1), only two variables *max_profit* and *min_price*

PART 05

Problem 2: Sort Colors



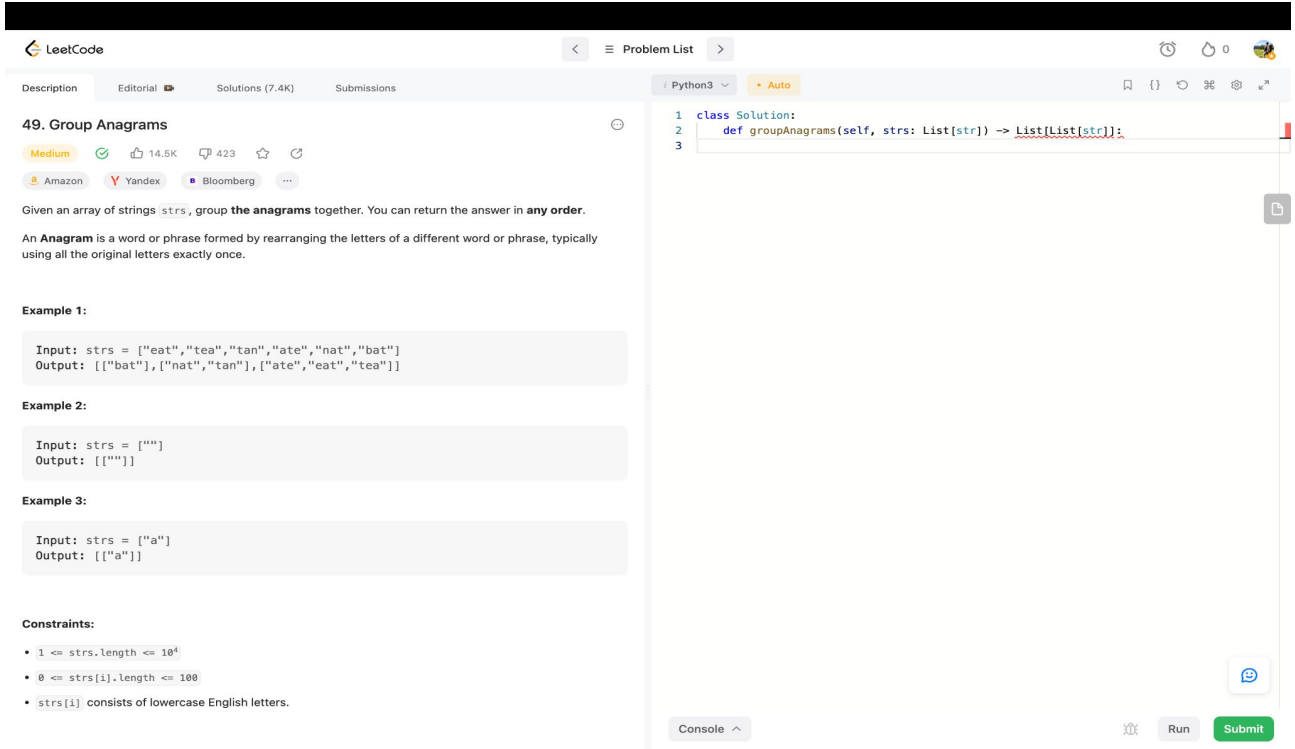
Approach 1: Single Pass

```
class Solution:
    def sortColors(self, nums: List[int]) -> None:
        p0 = curr = 0
        p2 = len(nums) - 1

        while curr <= p2:
            if nums[curr] == 0:
                nums[p0], nums[curr] = nums[curr], nums[p0]
                p0 += 1
                curr += 1
            elif nums[curr] == 2:
                nums[curr], nums[p2] = nums[p2], nums[curr]
                p2 -= 1
            else:
                curr += 1
```

Time complexity: O(n), loop runs one pass along the length

Space complexity: O(1), constant space

Problem 3: Group Anagrams



Approach 1: Categorize by Sorted String

```
class Solution(object):
    def groupAnagrams(self, strs):
        ans = collections.defaultdict(list)
        for s in strs:
            ans[tuple(sorted(s))].append(s)
        return ans.values()
```

Time complexity: $O(NK\log K)$, N is the length of strs, and K is the maximum length of a string in strs

Space complexity: $O(NK)$, constant space NK

Approach 2: Categorize by Count

```
class Solution:
    def groupAnagrams(strs):
        ans = collections.defaultdict(list)
        for s in strs:
            count = [0] * 26
            for c in s:
                count[ord(c) - ord('a')] += 1
            ans[tuple(count)].append(s)
        return ans.values()
```

Time complexity: $O(NK)$, N is the length of strs, and K is the maximum length of a string in strs

Space complexity: $O(NK)$, constant space NK

PART 06

Q/A

Slack Invite

Join Slack Workspace!

Problem Assignments

01. Contains Duplicate (Easy)

02. Container With Most Water (Medium)

03. Sliding Window Maximum (Hard)


NYU TANDON
Thank you!
Upcoming: Strings & Hash Maps