DSA Practice

Date: Nov 20

Problems:

Problems for the day: Leetcode

16

45

49

91

122

200

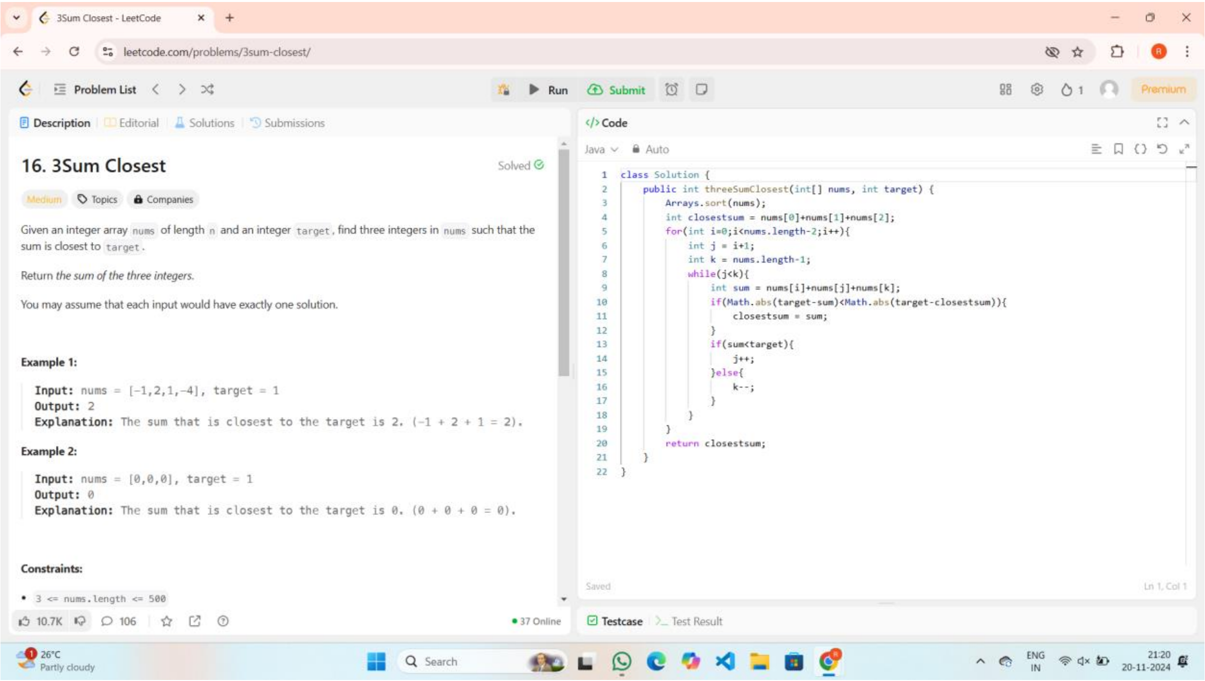
Quick Sort

Merge sort

Ternary Search

Interpolation Search

Problem1:



16. 3Sum Closest

Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to `target`.

Return *the sum of the three integers*.

You may assume that each input would have exactly one solution.

Example 1:

```
Input: nums = [-1,2,1,-4], target = 1
Output: 2
Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

Example 2:

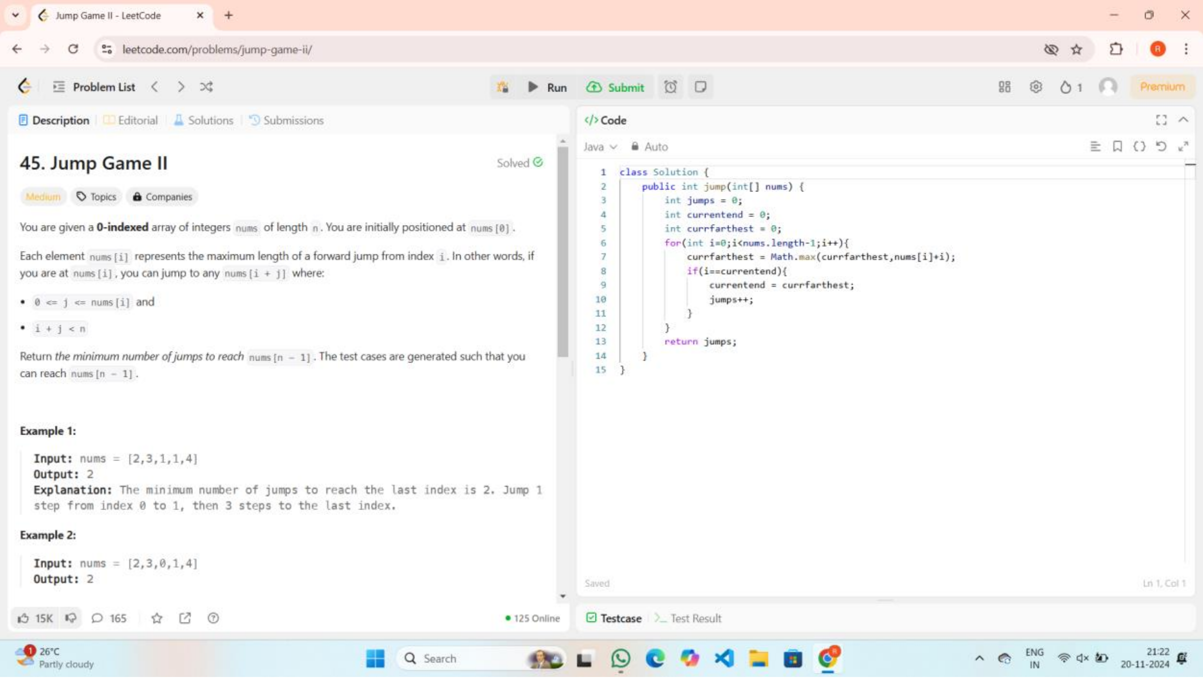
```
Input: nums = [0,0,0], target = 1
Output: 0
Explanation: The sum that is closest to the target is 0. (0 + 0 + 0 = 0).
```

Constraints:

- `3 <= nums.length <= 500`

```java
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int closestsum = nums[0]+nums[1]+nums[2];
        for(int i=0;i<nums.length-2;i++){
            int j = i+1;
            int k = nums.length-1;
            while(j<k){
                int sum = nums[i]+nums[j]+nums[k];
                if(Math.abs(target-sum)<Math.abs(target-closestsum)){
                    closestsum = sum;
                }
                if(sum<target){
                    j++;
                }else{
                    k--;
                }
            }
        }
        return closestsum;
    }
}
```

Problem2:



45. Jump Game II

Solved

Medium | Topics | Companies

You are given a **0-indexed** array of integers `nums` of length `n`. You are initially positioned at `nums[0]`.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at `nums[i]`, you can jump to any `nums[i + j]` where:

- `0 <= j <= nums[i]` and
- `i + j < n`

Return *the minimum number of jumps to reach* `nums[n - 1]`. The test cases are generated such that you can reach `nums[n - 1]`.

Example 1:

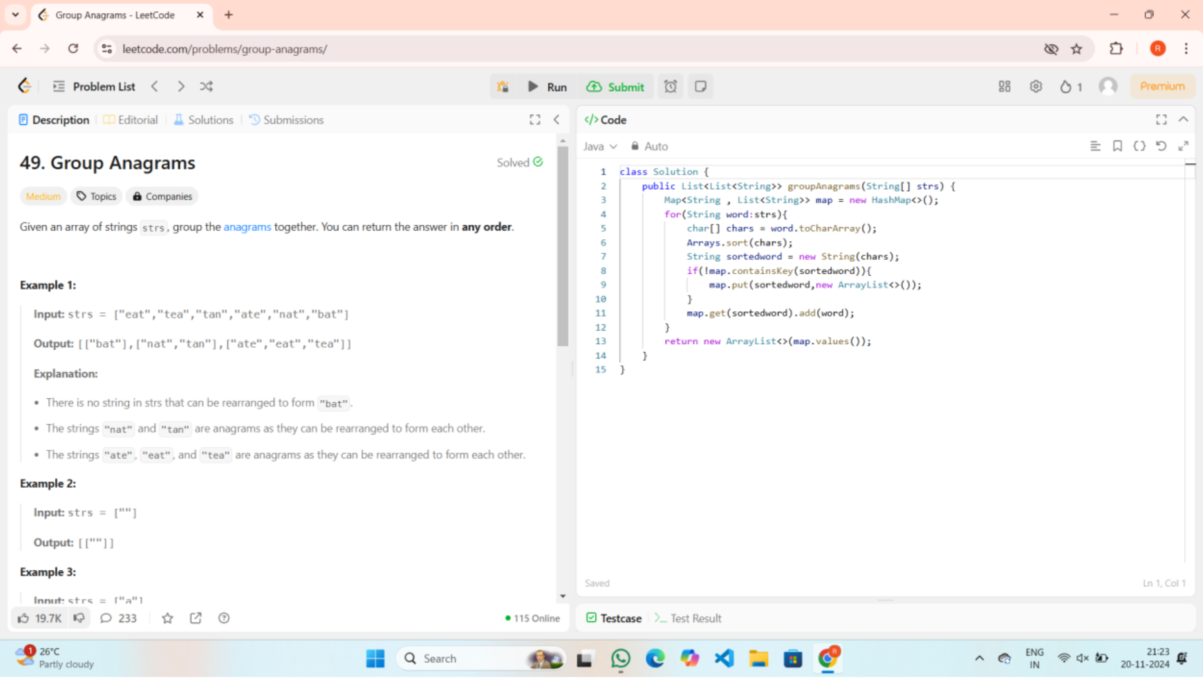
```
Input: nums = [2,3,1,1,4]
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2. Jump 1
step from index 0 to 1, then 3 steps to the last index.
```

Example 2:

```
Input: nums = [2,3,0,1,4]
Output: 2
```

```java
class Solution {
    public int jump(int[] nums) {
        int jumps = 0;
        int currentend = 0;
        int currfarthest = 0;
        for(int i=0;i<nums.length-1;i++){
            currfarthest = Math.max(currfarthest,nums[i]+i);
            if(i==currentend){
                currentend = currfarthest;
                jumps++;
            }
        }
        return jumps;
    }
}
```

Problem3:



49. Group Anagrams

Solved

Medium | Topics | Companies

Given an array of strings `strs`, group the anagrams together. You can return the answer in **any order**.

Example 1:

Input: strs = ["eat","tea","tan","ate","nat","bat"]

Output: [["bat"],["nat","tan"],["ate","eat","tea"]]

Explanation:

- There is no string in strs that can be rearranged to form `"bat"`.
- The strings `"nat"` and `"tan"` are anagrams as they can be rearranged to form each other.
- The strings `"ate"`, `"eat"`, and `"tea"` are anagrams as they can be rearranged to form each other.

Example 2:

Input: strs = [""]

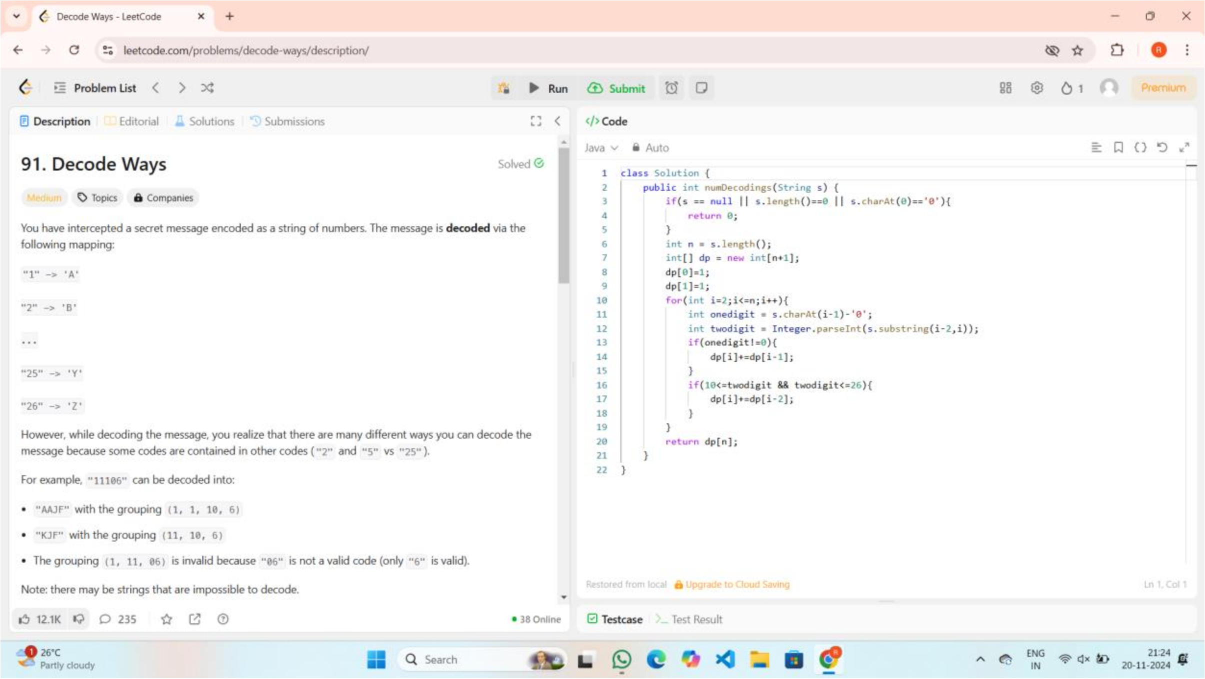
Output: [[""]]

Example 3:

Input: strs = ["a"]

```java
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String , List<String>> map = new HashMap<>();
        for(String word:strs){
            char[] chars = word.toCharArray();
            Arrays.sort(chars);
            String sortedword = new String(chars);
            if(!map.containsKey(sortedword)){
                map.put(sortedword,new ArrayList<>());
            }
            map.get(sortedword).add(word);
        }
        return new ArrayList<>(map.values());
    }
}
```

Problem4:



91. Decode Ways

Solved

You have intercepted a secret message encoded as a string of numbers. The message is **decoded** via the following mapping:

`"1" -> 'A'`

`"2" -> 'B'`

`...`

`"25" -> 'Y'`

`"26" -> 'Z'`

However, while decoding the message, you realize that there are many different ways you can decode the message because some codes are contained in other codes (`"2"` and `"5"` vs `"25"`).

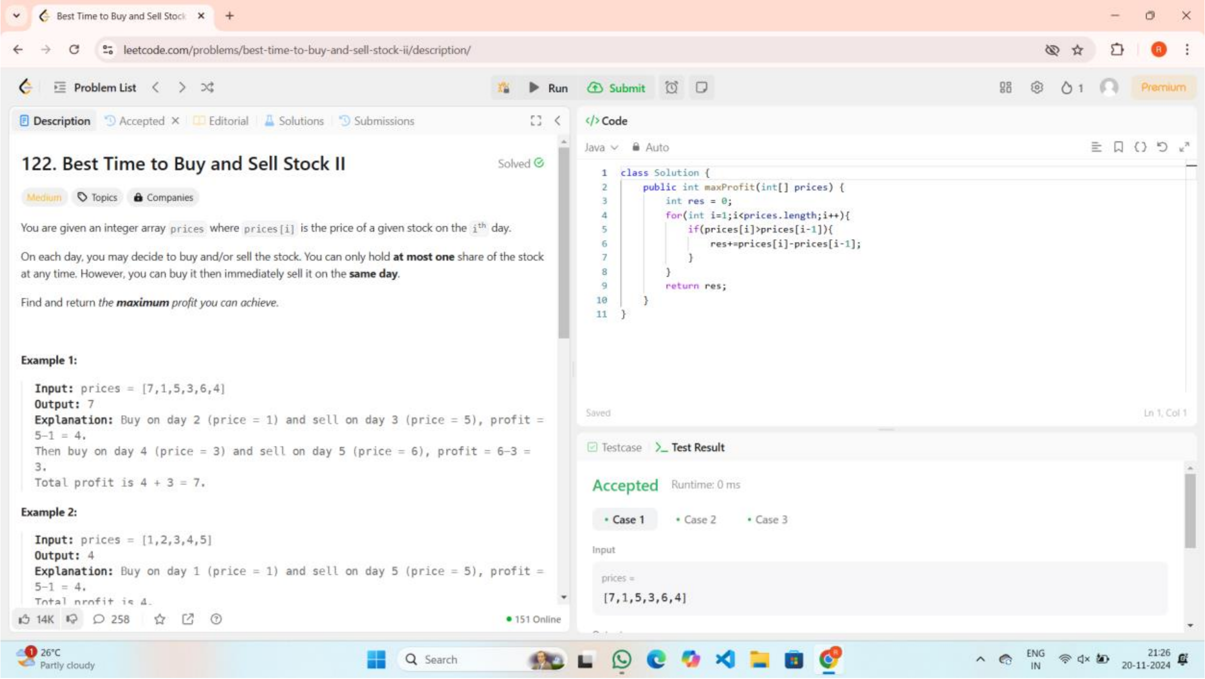
For example, `"11106"` can be decoded into:

- `"AAJF"` with the grouping `(1, 1, 10, 6)`
- `"KJF"` with the grouping `(11, 10, 6)`
- The grouping `(1, 11, 06)` is invalid because `"06"` is not a valid code (only `"6"` is valid).

Note: there may be strings that are impossible to decode.

```java
class Solution {
    public int numDecodings(String s) {
        if(s == null || s.length()==0 || s.charAt(0)=='0'){
            return 0;
        }
        int n = s.length();
        int[] dp = new int[n+1];
        dp[0]=1;
        dp[1]=1;
        for(int i=2;i<=n;i++){
            int onedigit = s.charAt(i-1)-'0';
            int twodigit = Integer.parseInt(s.substring(i-2,i));
            if(onedigit!=0){
                dp[i]+=dp[i-1];
            }
            if(10<=twodigit && twodigit<=26){
                dp[i]+=dp[i-2];
            }
        }
        return dp[n];
    }
}
```

Problem5:



122. Best Time to Buy and Sell Stock II

Solved

You are given an integer array `prices` where `prices[i]` is the price of a given stock on the `i`th day.

On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time. However, you can buy it then immediately sell it on the **same day**.

Find and return *the **maximum** profit you can achieve.*

Example 1:

```
Input: prices = [7,1,5,3,6,4]
Output: 7
Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit =
5-1 = 4.
Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 =
3.
Total profit is 4 + 3 = 7.
```

Example 2:

```
Input: prices = [1,2,3,4,5]
Output: 4
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit =
5-1 = 4.
Total profit is 4.
```

```java
class Solution {
    public int maxProfit(int[] prices) {
        int res = 0;
        for(int i=1;i<prices.length;i++){
            if(prices[i]>prices[i-1]){
                res+=prices[i]-prices[i-1];
            }
        }
        return res;
    }
}
```

Accepted Runtime: 0 ms

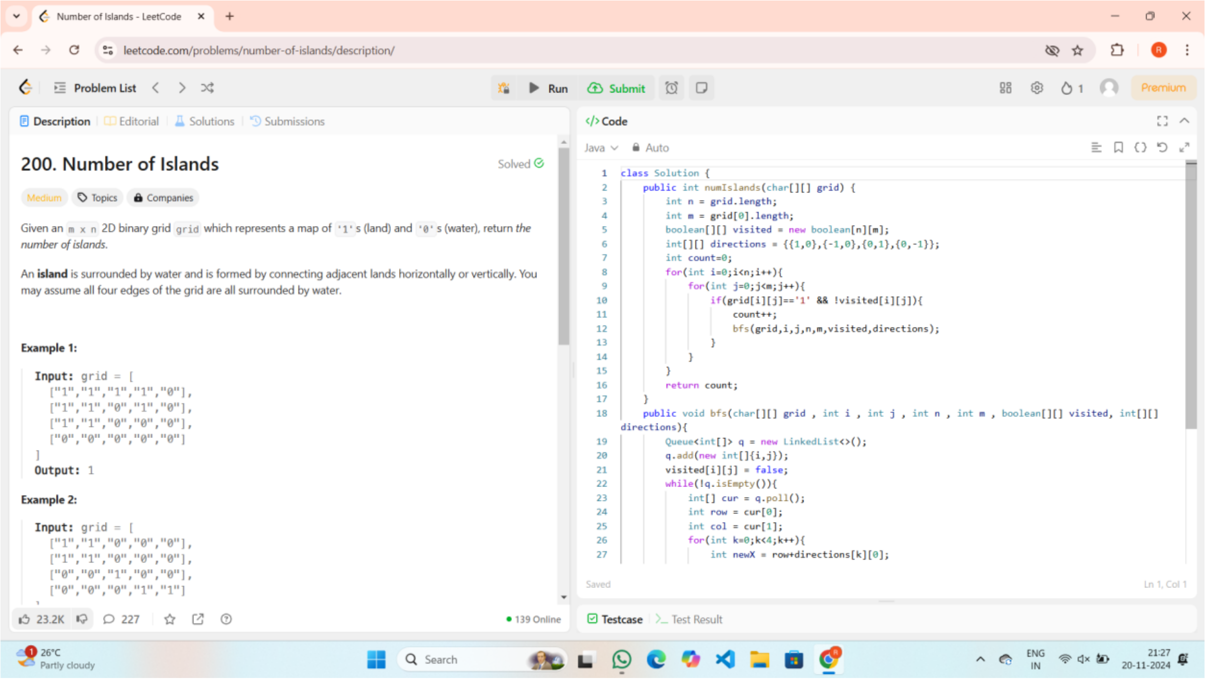
• Case 1 • Case 2 • Case 3

Input

prices =

```
[7,1,5,3,6,4]
```

Problem6:



200. Number of Islands

Solved

Medium · Topics · Companies

Given an `m x n` 2D binary grid `grid` which represents a map of `'1'`s (land) and `'0'`s (water), return *the number of islands*.

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

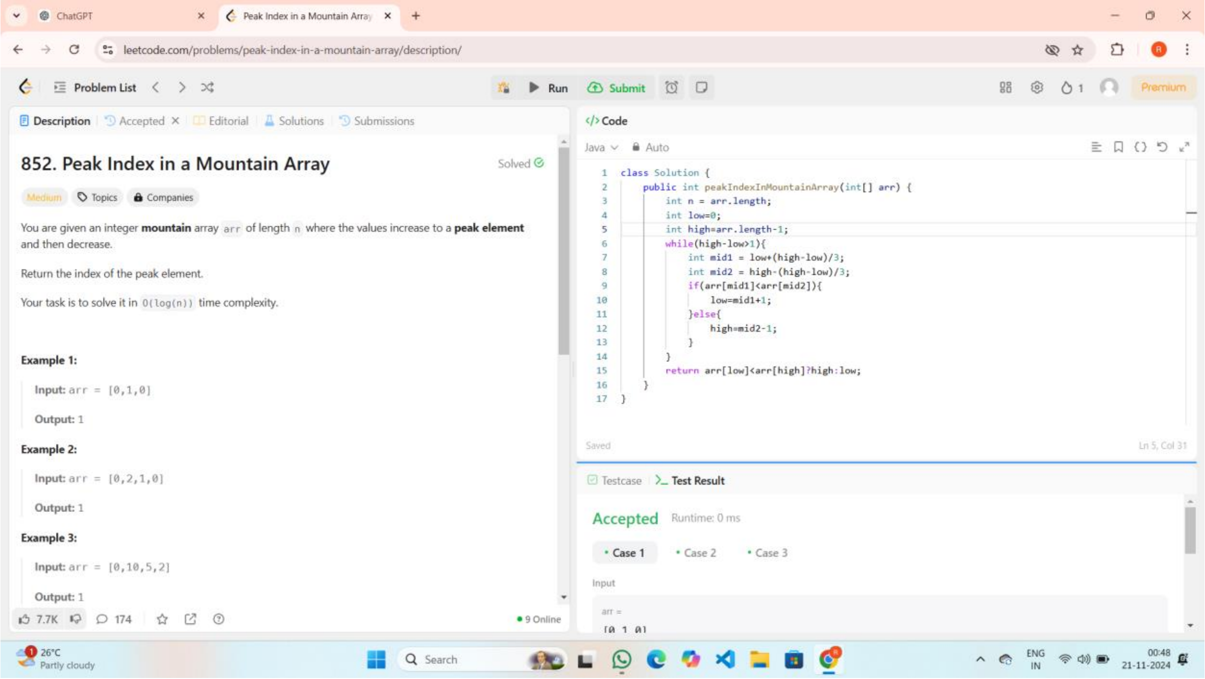
```
Input: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
Output: 1
```

Example 2:

```
Input: grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
```

```java
class Solution {
    public int numIslands(char[][] grid) {
        int n = grid.length;
        int m = grid[0].length;
        boolean[][] visited = new boolean[n][m];
        int[][] directions = {{1,0},{-1,0},{0,1},{0,-1}};
        int count=0;
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(grid[i][j]=='1' && !visited[i][j]){
                    count++;
                    bfs(grid,i,j,n,m,visited,directions);
                }
            }
        }
        return count;
    }
    public void bfs(char[][] grid , int i , int j , int n , int m , boolean[][] visited, int[][]
directions){
        Queue<int[]> q = new LinkedList<>();
        q.add(new int[]{i,j});
        visited[i][j] = false;
        while(!q.isEmpty()){
            int[] cur = q.poll();
            int row = cur[0];
            int col = cur[1];
            for(int k=0;k<4;k++){
                int newX = row+directions[k][0];
```

Problem7:



852. Peak Index in a Mountain Array

Solved

Medium · Topics · Companies

You are given an integer **mountain** array `arr` of length `n` where the values increase to a **peak element** and then decrease.

Return the index of the peak element.

Your task is to solve it in `O(log(n))` time complexity.

Example 1:

Input: arr = `[0,1,0]`

Output: 1

Example 2:

Input: arr = `[0,2,1,0]`

Output: 1

Example 3:

Input: arr = `[0,10,5,2]`

Output: 1

```java
class Solution {
    public int peakIndexInMountainArray(int[] arr) {
        int n = arr.length;
        int low=0;
        int high=arr.length-1;
        while(high-low>1){
            int mid1 = low+(high-low)/3;
            int mid2 = high-(high-low)/3;
            if(arr[mid1]<arr[mid2]){
                low=mid1+1;
            }else{
                high=mid2-1;
            }
        }
        return arr[low]<arr[high]?high:low;
    }
}
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

arr =

[0,1,0]