

## AP Experiment-9

Name: Shivangi Gupta

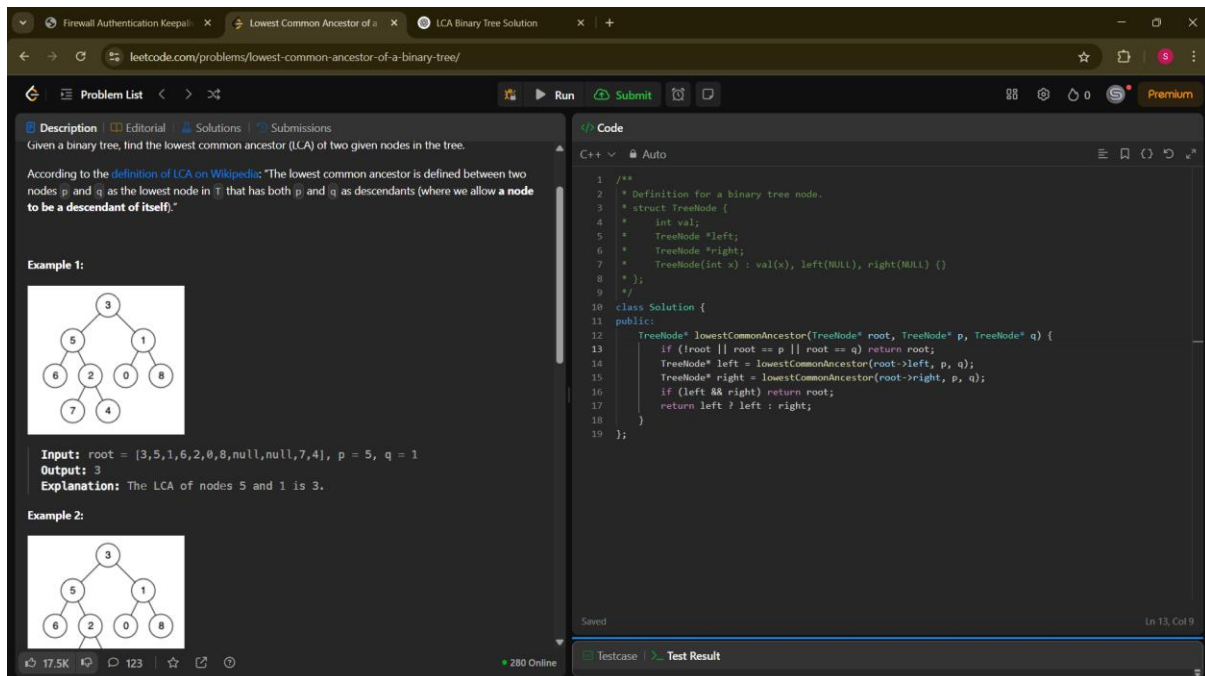
UID: 22BCS15008

Section: 601-A

Q1. Lowest Common Ancestor of a Binary Tree <https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/>

CODE:

```
class Solution {  
public:  
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {  
        if (!root || root == p || root == q) return root;  
        TreeNode* left = lowestCommonAncestor(root->left, p, q);  
        TreeNode* right = lowestCommonAncestor(root->right, p, q);  
        if (left && right) return root;  
        return left ? left : right;  
    }  
};
```



Q2. <https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/description/?envType=problem-list-v2&envId=greedy>

CODE:

```
class Solution {
```

```

public:

int minOperations(vector<int>& nums) {

    int operations=0;

    for(int i=0;i<nums.size()-1;i++){

        if(nums[i]>=nums[i+1]){

            operations += (nums[i] - nums[i+1])+1;

            nums[i+1]=nums[i] +1;

        }

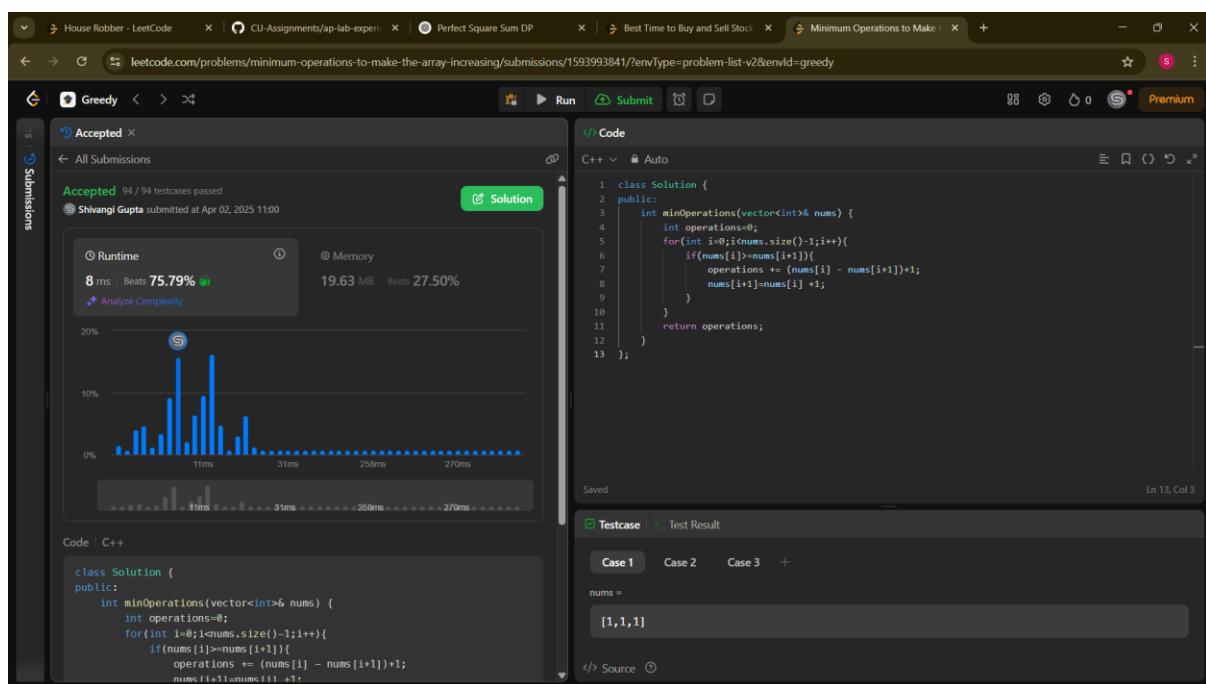
    }

    return operations;

}

};

```



Q3. <https://leetcode.com/problems/remove-stones-to-minimize-the-total/?envType=problem-list-v2&envId=greedy>

CODE:

```

class Solution {

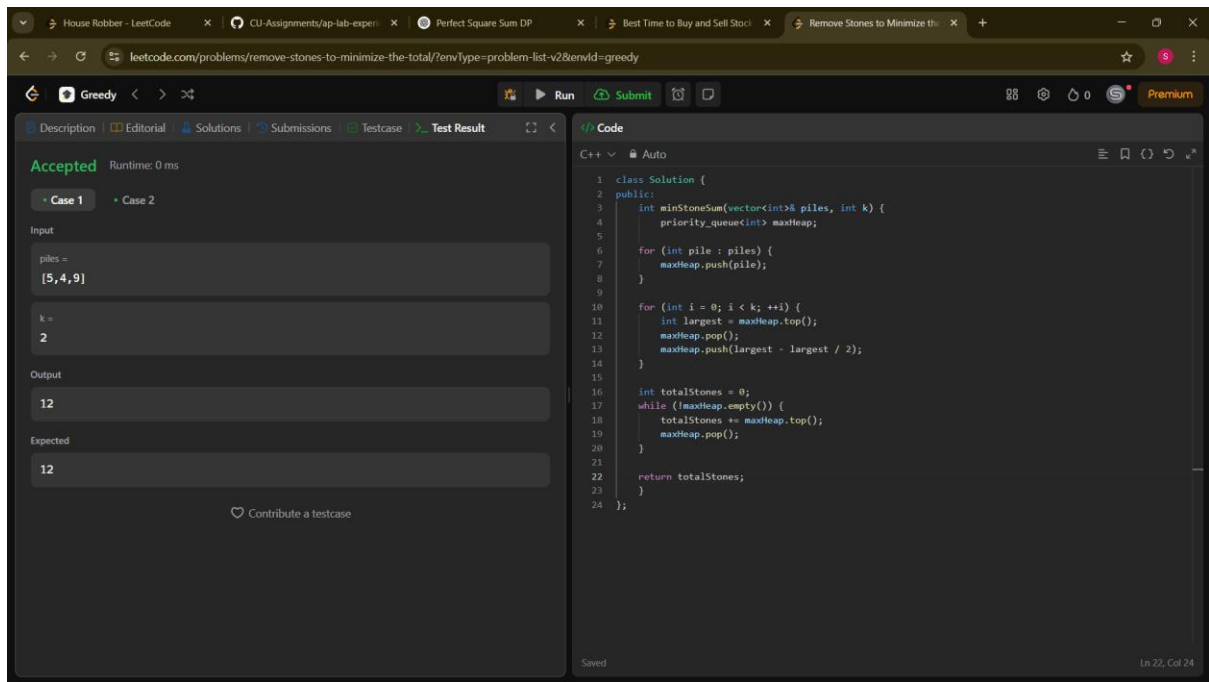
public:

    int minStoneSum(vector<int> & piles, int k) {

        priority_queue<int> maxHeap;
    }
}

```

```
for (int pile : piles) {  
    maxHeap.push(pile);  
}  
  
for (int i = 0; i < k; ++i) {  
    int largest = maxHeap.top();  
    maxHeap.pop();  
    maxHeap.push(largest - largest / 2);  
}  
  
int totalStones = 0;  
while (!maxHeap.empty()) {  
    totalStones += maxHeap.top();  
    maxHeap.pop();  
}  
  
return totalStones;  
}  
};
```



Q4. <https://leetcode.com/problems/maximum-score-from-removing-substrings/?envType=problem-list-v2&envId=greedy>

CODE:

```
class Solution {
```

```
public:
```

```
    int maximumGain(string s, int x, int y) {
```

```
        int points = 0;
```

```
        while (true) {
```

```
            size_t pos_ab = s.find("ab");
```

```
            size_t pos_ba = s.find("ba");
```

```
            if (pos_ab != string::npos && (pos_ba == string::npos || x >= y)) {
```

```
                s.erase(pos_ab, 2);
```

```
                points += x;
```

```
            } else if (pos_ba != string::npos) {
```

```
                s.erase(pos_ba, 2);
```

```
                points += y;
```

```
            } else {
```

```

        break;
    }
}

return points;
}
};

```

The screenshot shows a LeetCode submission for the problem "Maximum Score From Removing Substrings". The problem is marked as "Accepted" with a runtime of 0 ms. The input is a string "cdbcbbaaabab", x=4, and y=5. The output is 19, which matches the expected result. The code editor shows a C++ solution using a greedy approach to remove substrings "ab" and "ba" to maximize the score.

**Problem Details:**

- Problem:** Maximum Score From Removing Substrings
- Difficulty:** Greedy
- Runtime:** 0 ms

**Input:**

- s = "cdbcbbaaabab"
- x = 4
- y = 5

**Output:** 19

**Expected:** 19

**Code:**

```

1 class Solution {
2 public:
3     int maximumGain(string s, int x, int y) {
4         int points = 0;
5
6         while (true) {
7             size_t pos_ab = s.find("ab");
8             size_t pos_ba = s.find("ba");
9
10            if (pos_ab != string::npos && (pos_ba == string::npos || x >= y)) {
11                s.erase(pos_ab, 2);
12                points += x;
13            } else if (pos_ba != string::npos) {
14                s.erase(pos_ba, 2);
15                points += y;
16            } else {
17                break;
18            }
19        }
20
21        return points;
22    }
23 };

```