Experiment-8

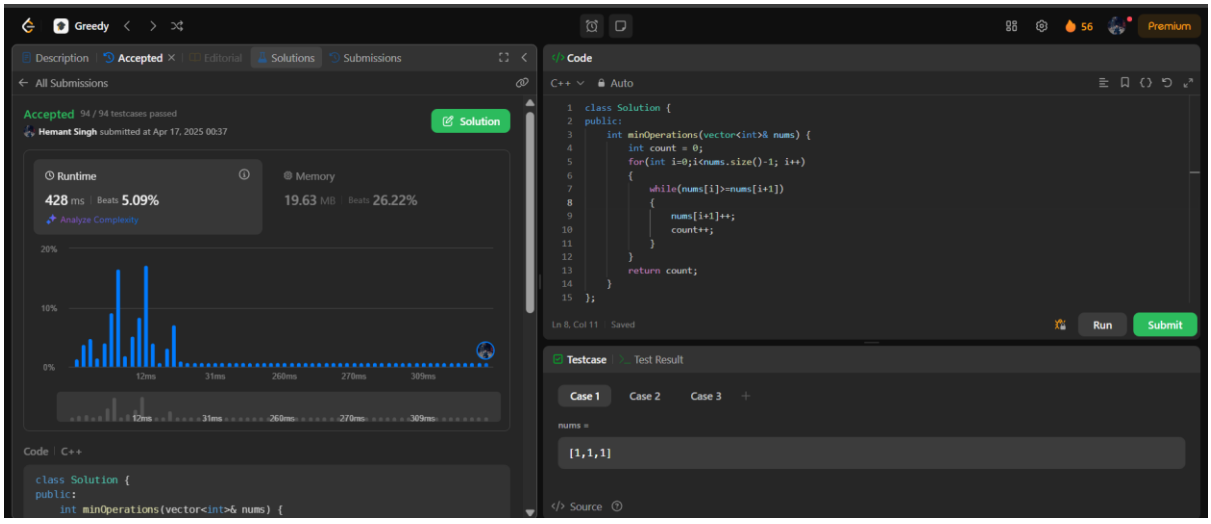
Name:- Hemant Singh **UID:- 22BCS12820**

Q1. 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 count = 0;
        for(int i=0;i<nums.size()-1; i++)
        {
            while(nums[i]>=nums[i+1])
            {
                nums[i+1]++;
                count++;
            }}
        return count;
    }};
```

Output:-



Q2. https://leetcode.com/problems/maximum-score-from-removing-substrings/description/?envType=problem-list-v2&envId=greedy

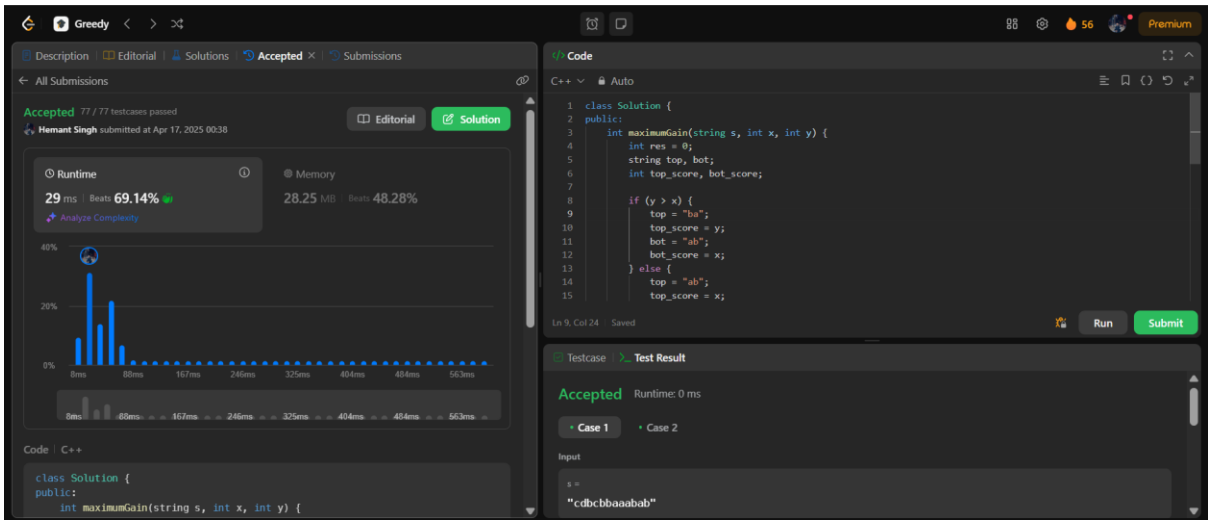
Code:

```
class Solution {
public:
    int maximumGain(string s, int x, int y) {
        int res = 0;
        string top, bot;
        int top_score, bot_score;

        if (y > x) {
            top = "ba";
            top_score = y;
            bot = "ab";
            bot_score = x;
        } else {
            top = "ab";
            top_score = x;
            bot = "ba";
            bot_score = y;
        }
        vector<char> stack;
        for (char ch : s) {
            if (ch == top[1] && !stack.empty() && stack.back() == top[0]) {
                res += top_score;
                stack.pop_back();
            } else {
                stack.push_back(ch);
            }
        }
        vector<char> new_stack;
        for (char ch : stack) {
            if (ch == bot[1] && !new_stack.empty() && new_stack.back() == bot[0]) {
```

```
            res += bot_score;
            new_stack.pop_back();
        } else {
            new_stack.push_back(ch);
        }
    }
    return res;
  }
};
```

Output:



Q3. https://leetcode.com/problems/minimum-operations-to-make-a-subsequence/submissions/1608868078/?envType=problem-list-v2&envId=greedy

Code:

```
class Solution {
public:
    int minOperations(vector<int>& target, vector<int>& arr) {
        unordered_map<int, int> mapping;
        int i = 0;
        for (auto& num : target)
            mapping[num] = ++i;
```

```
        vector<int> A;
        for (int& num : arr)
            if (mapping.find(num) != mapping.end())
                A.push_back(mapping[num]);
        return target.size() - lengthOfLIS(A);
    }
private:
    int lengthOfLIS(vector<int>& nums) {
        if (nums.empty()) return 0;
        vector<int> piles;
        for(int i=0; i<nums.size(); i++) {
            auto it = std::lower_bound(piles.begin(), piles.end(), nums[i]);
            if (it == piles.end())
                piles.push_back(nums[i]);
            else
                *it = nums[i];
        }
        return piles.size();
}
};
```

Output: