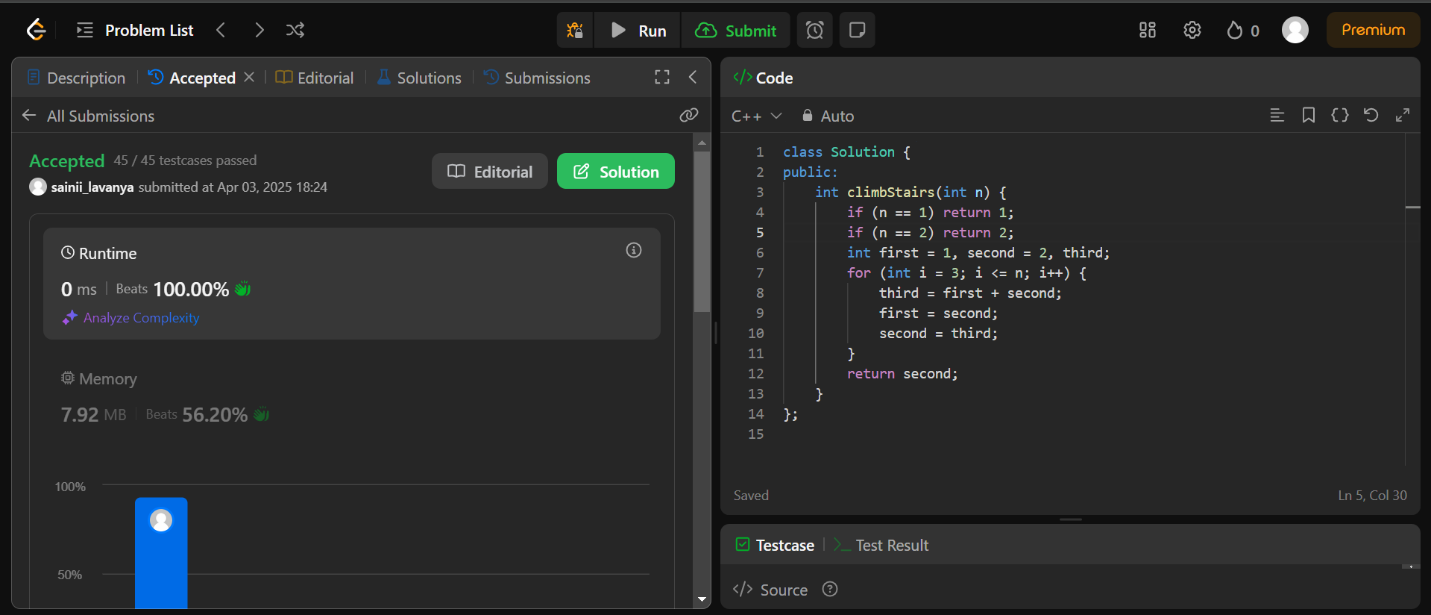
**ASSIGNMENT 7**

**AP LAB**

**Lavanya Saini – 22BCS15497**

1. **Climbing Stairs**
2. class Solution {
3. public:
4. int climbStairs(int n) {
5. if (n == 1) return 1;
6. if (n == 2) return 2;
7. int first = 1, second = 2, third;
8. for (int i = 3; i <= n; i++) {
9. third = first + second;
10. first = second;
11. second = third;
12. }
13. return second;
14. }
15. ****};

**2. Maximum Subarray**

#include <vector>

#include <climits>

class Solution {

public:

    int maxSubArray(std::vector<int>& nums) {

        int maxSum = INT\_MIN;

        int currentSum = 0;

        for (int num : nums) {

            currentSum += num;

            maxSum = max(maxSum, currentSum);

            if (currentSum < 0) {

                currentSum = 0;

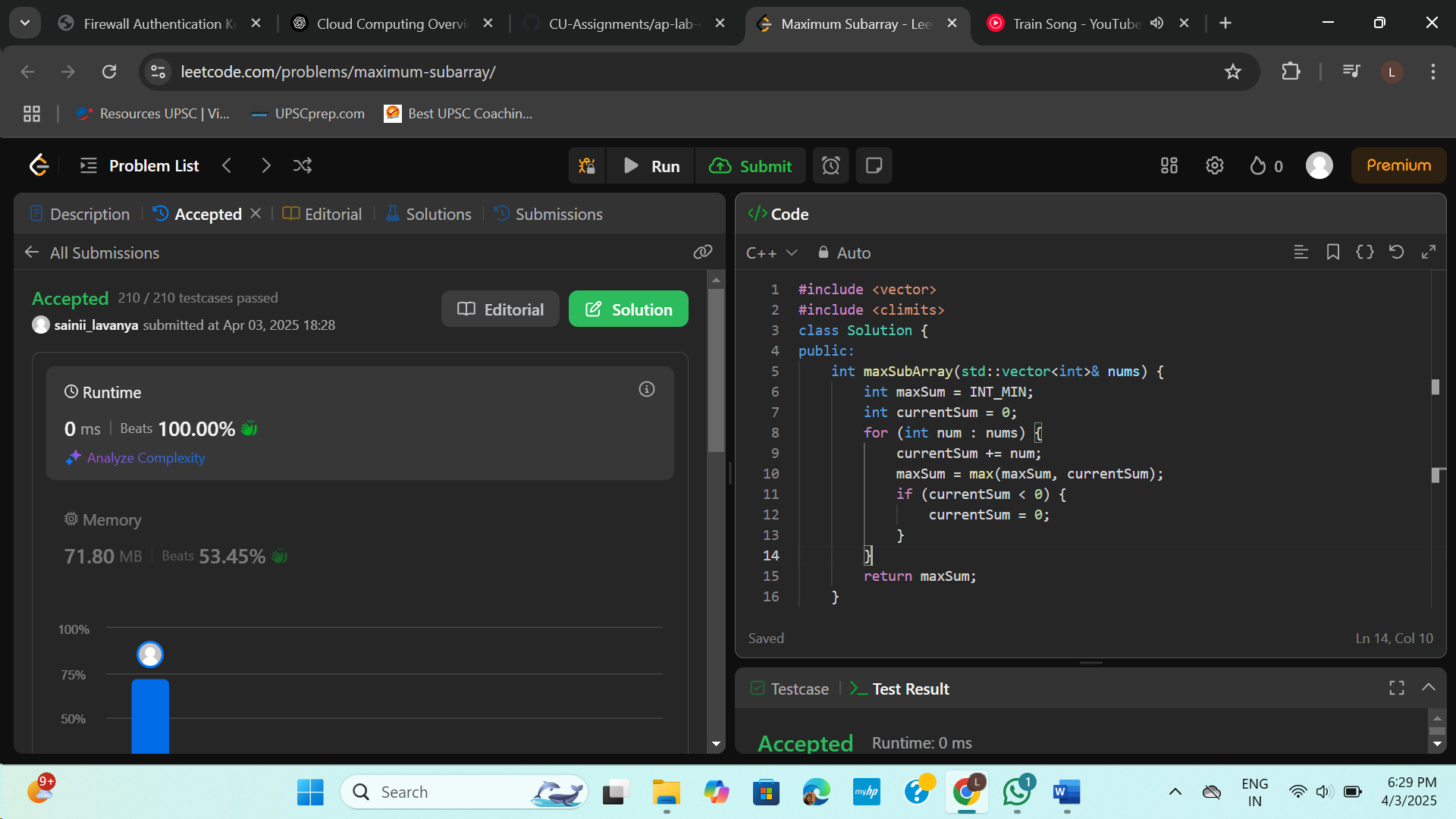
            }

        }

        return maxSum;

    }

};



**3. Unique Paths**

class Solution {

public:

    int uniquePaths(int m, int n) {

        vector<int> dp(n, 1);

        for (int i = 1; i < m; i++) {

            for (int j = 1; j < n; j++) {

                dp[j] += dp[j - 1];

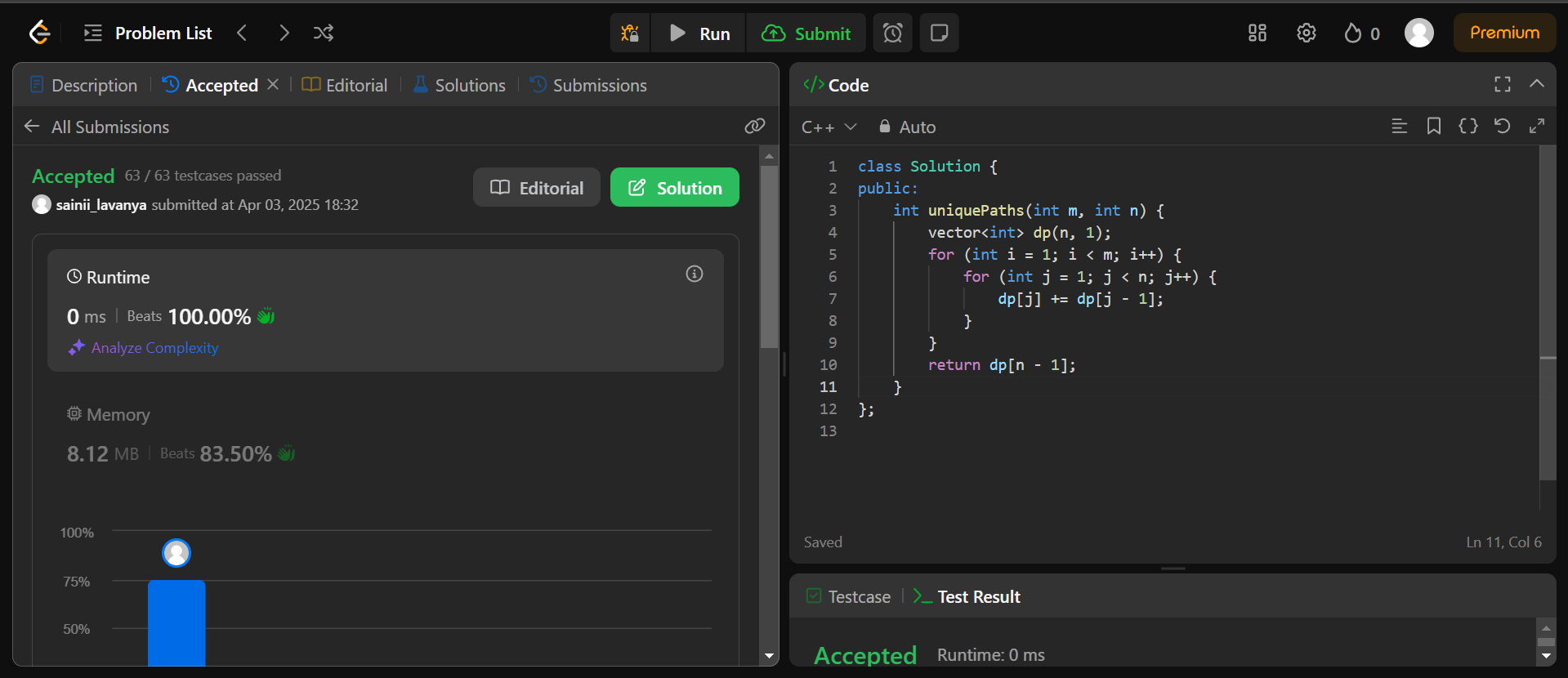
            }

        }

        return dp[n - 1];

    }

};



**4. Longest Increasing Subsequence**

#include <vector>

class Solution {

public:

    int lengthOfLIS(vector<int>& nums) {

        int n = nums.size();

        if (n == 0) return 0;

        vector<int> dp(n, 1);

        int maxLIS = 1;

        for (int i = 1; i < n; i++) {

            for (int j = 0; j < i; j++) {

                if (nums[j] < nums[i]) {

                    dp[i] = max(dp[i], dp[j] + 1);

                }

            }

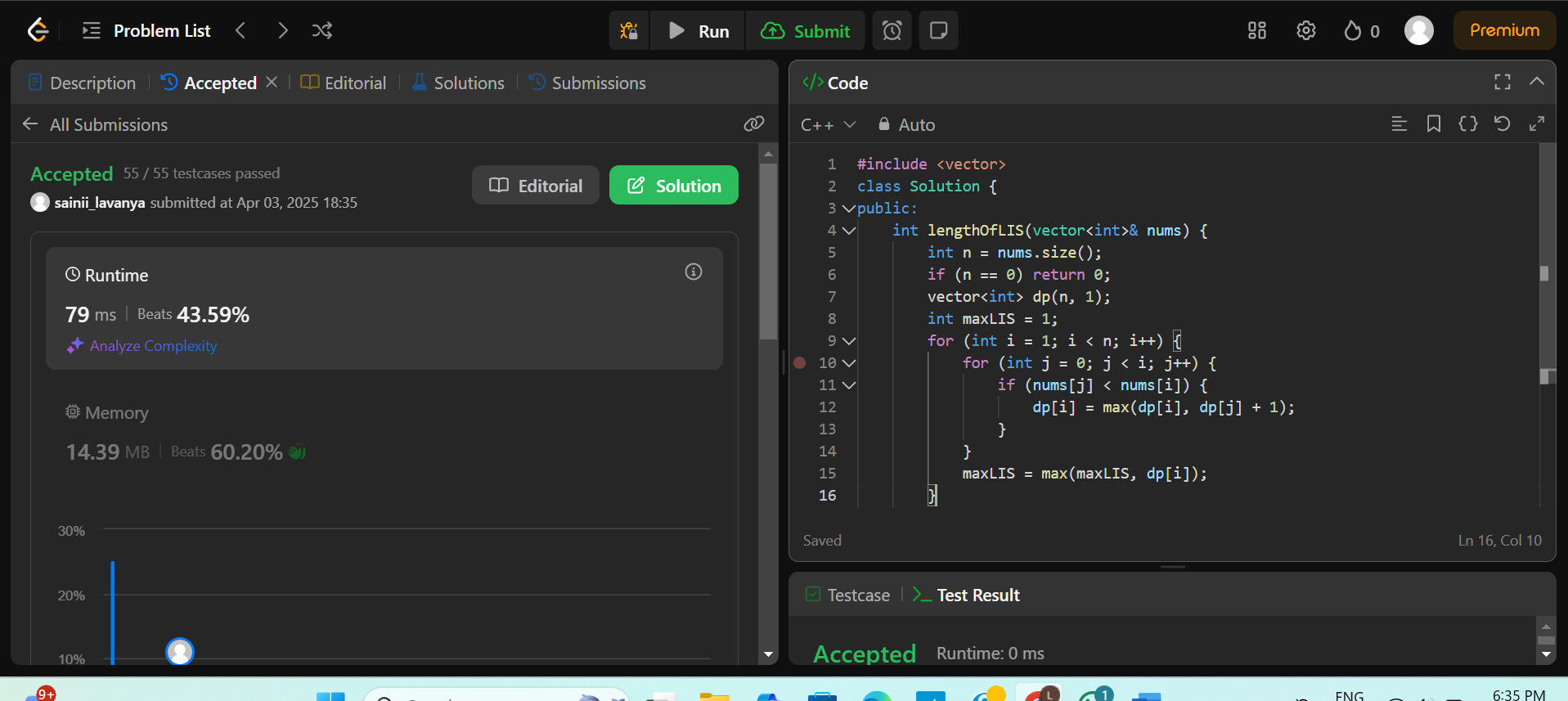
            maxLIS = max(maxLIS, dp[i]);

        }

        return maxLIS;

    }

};



**5. Decode Ways**

class Solution {

public:

    int numDecodings(string s) {

        int n = s.size();

        if (n == 0 || s[0] == '0') return 0;

        vector<int> dp(n + 1, 0);

        dp[0] = 1;

        dp[1] = (s[0] != '0') ? 1 : 0;

        for (int i = 2; i <= n; i++) {

            if (s[i - 1] != '0') {

                dp[i] += dp[i - 1];

            }

            int twoDigit = (s[i - 2] - '0') \* 10 + (s[i - 1] - '0');

            if (twoDigit >= 10 && twoDigit <= 26) {

                dp[i] += dp[i - 2];

            }

        }

        return dp[n];

    }

};

