**Assignment 7**

**Name – Aditya Taragi Uid – 22BCS10422.**

**Problem 1: Maximum Subarray (**<https://leetcode.com/problems/maximum-subarray/> **)**

**Code:**class Solution {

public:

    int maxSubArray(vector<int>& nums) {

        int sum = nums[0];

        int max\_sum = nums[0];

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

            sum = max(nums[i],sum+nums[i]);

            max\_sum = max(sum,max\_sum);

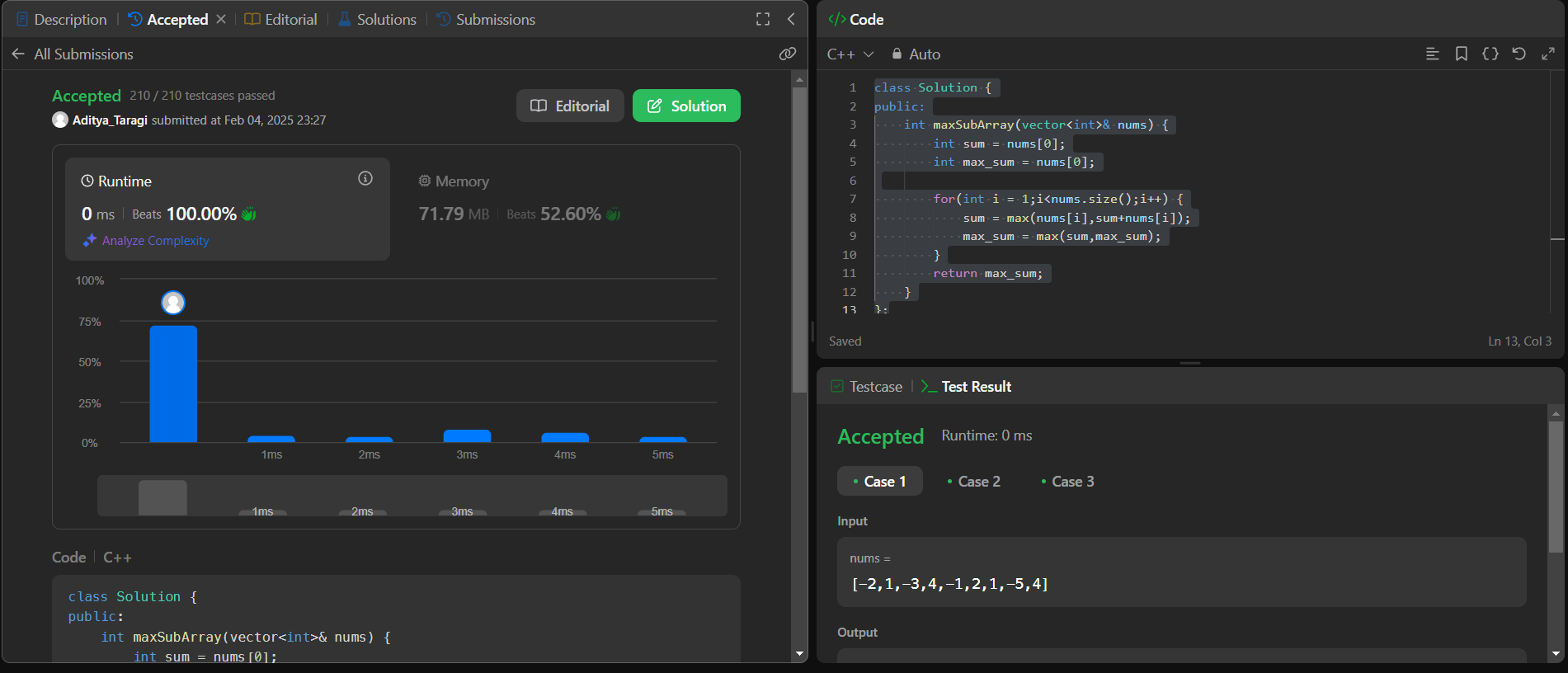
        }

        return max\_sum;

    }

};

**Screenshot:**



**Problem 2: Jump Game (**<https://leetcode.com/problems/jump-game/> **)**

**Code:**class Solution {

public:

    bool canJump(vector<int>& nums) {

        int scope = 0;

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

            if (i > scope) return false;

            scope = max(scope, i + nums[i]);

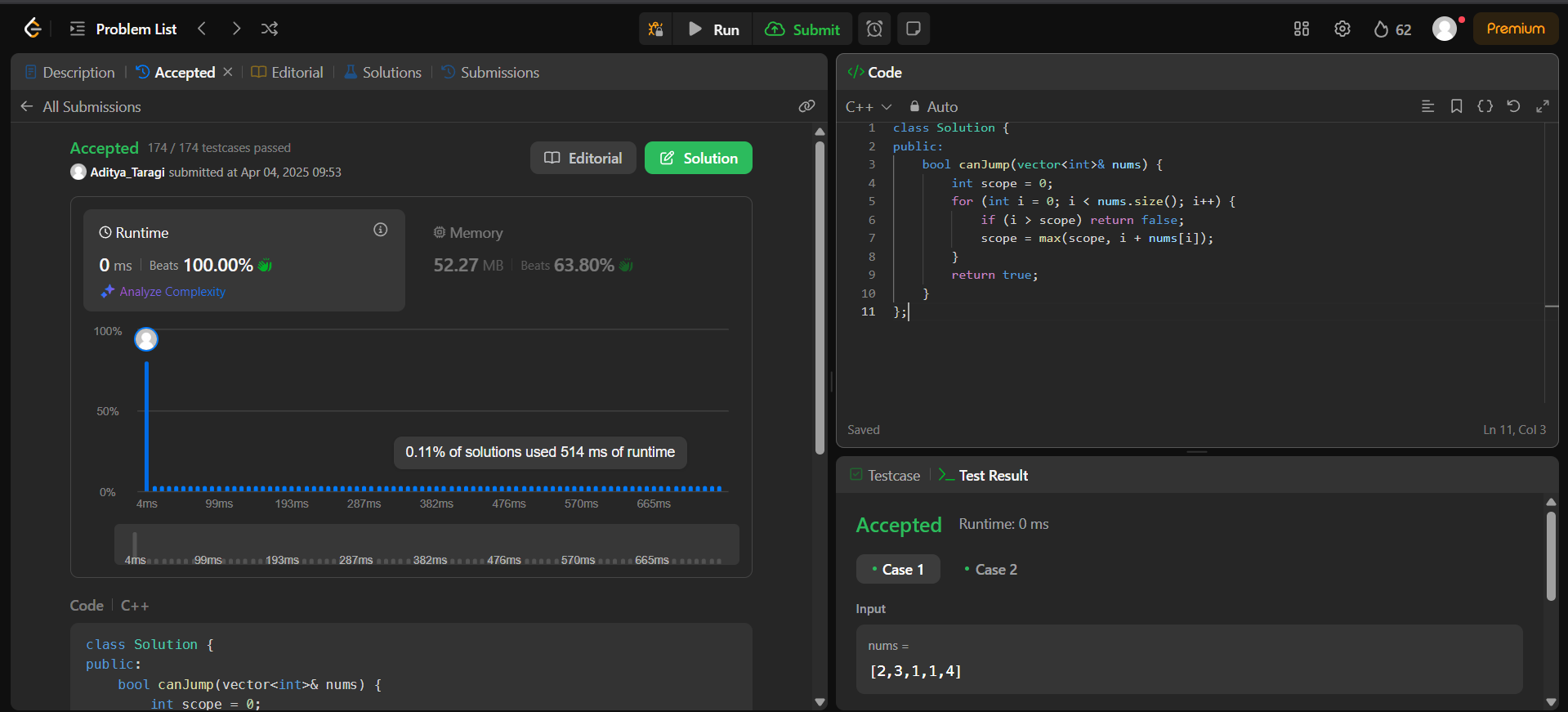
        }

        return true;

    }

};

**Screenshot:**



**Problem 3: House Robber (**<https://leetcode.com/problems/house-robber/> **)**

**Code:**class Solution {

public:

    int rob(vector<int>& nums) {

        int r = 0;

        int no = 0;

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

            int temp = no + nums[i];

            int tempno = max(no, r);

            r = temp;

            no = tempno;

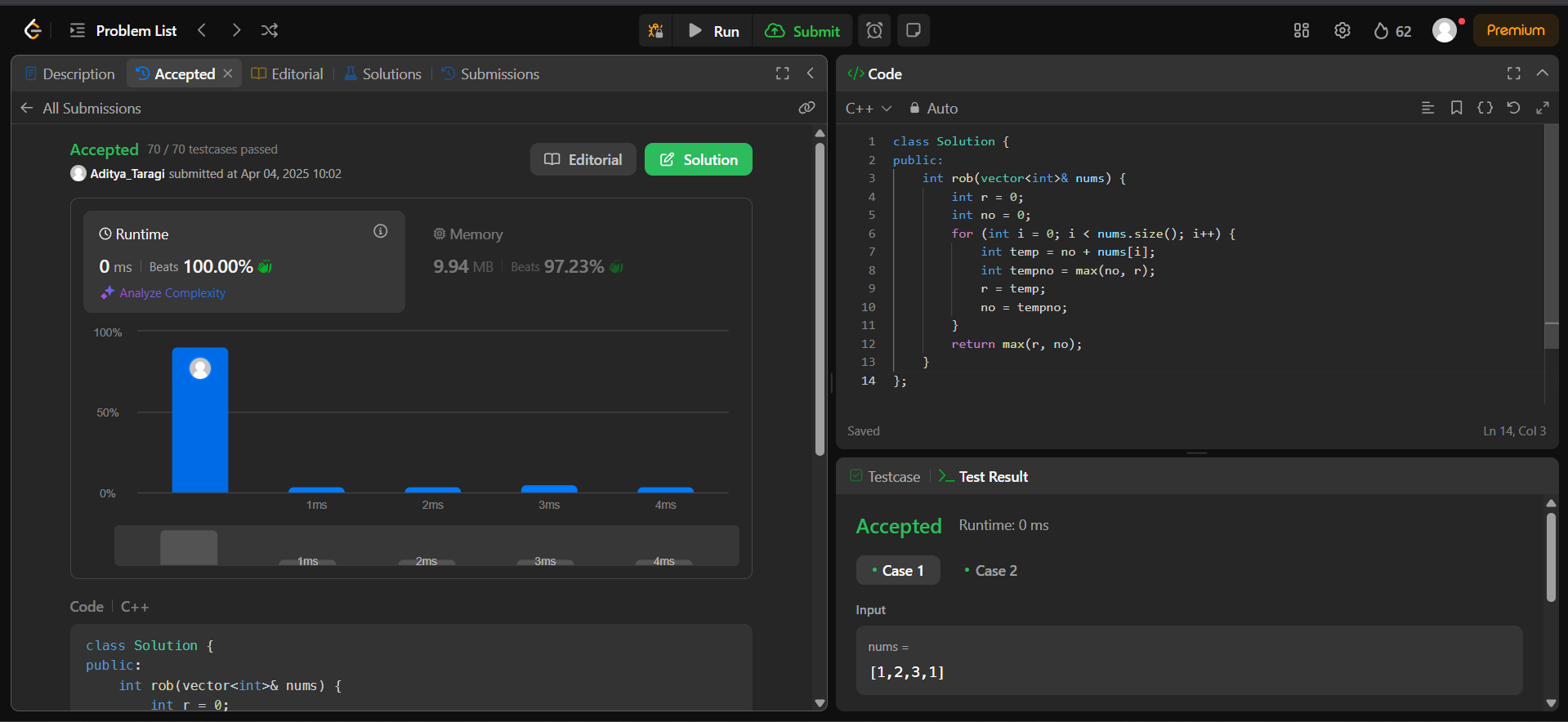
        }

        return max(r, no);

    }

};

**Screenshot:**



**Problem 4: Unique Paths (**<https://leetcode.com/problems/unique-paths/> **)**

**Code:**class Solution {

public:

    int uniquePaths(int m, int n) {

        vector<vector<int>> dp(m, vector<int>(n, 1));

        for (int i = 1; i < m; ++i) {

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

                dp[i][j] = dp[i - 1][j] + dp[i][j - 1];

            }

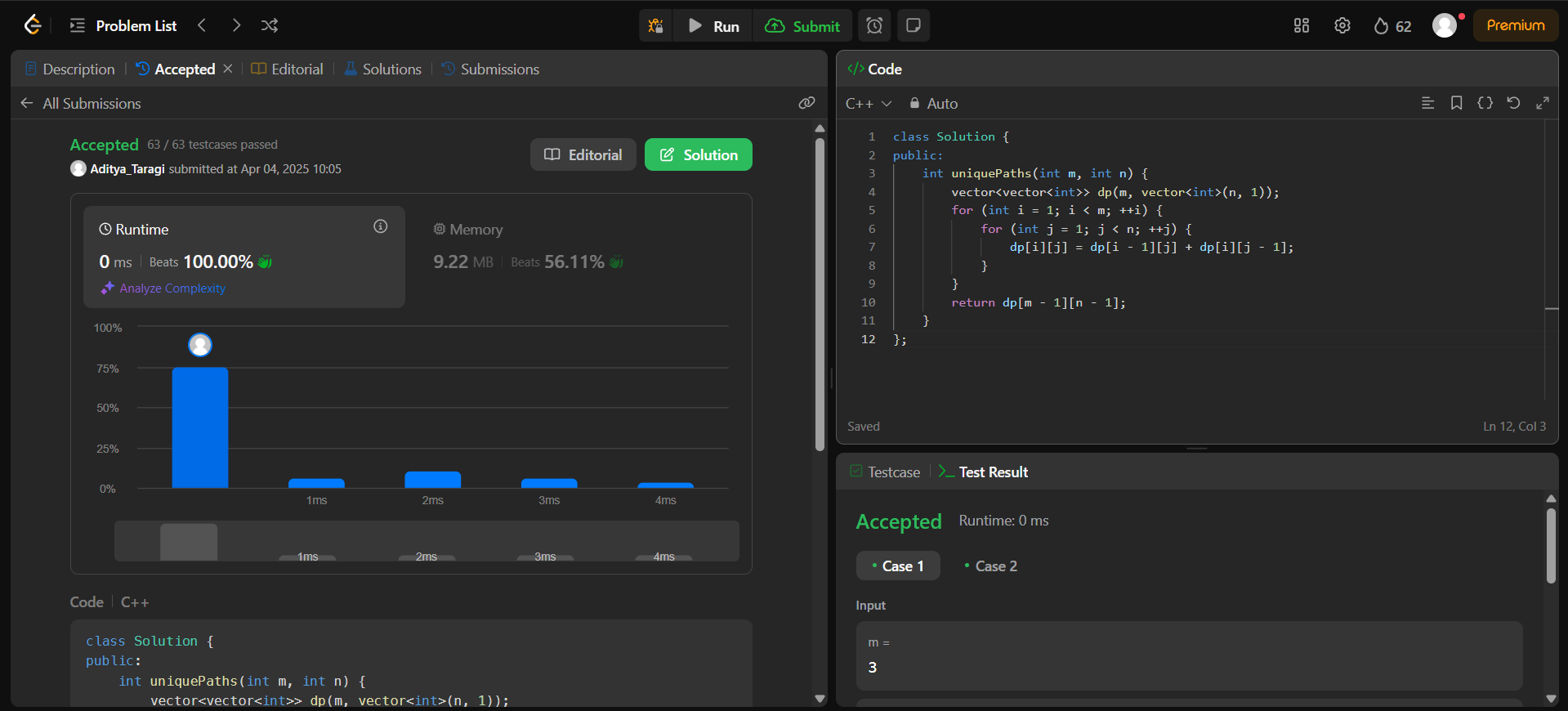
        }

        return dp[m - 1][n - 1];

    }

};

**Screenshot:**



**Problem 5: Coin Change (**<https://leetcode.com/problems/coin-change/> **)**

**Code:**class Solution {

public:

    int coinChange(vector<int>& coins, int amount) {

        vector<int> dp(amount + 1, INT\_MAX);

        dp[0] = 0;

        for (int i = 1; i <= amount; ++i) {

            for (int coin : coins) {

                if (i - coin >= 0 && dp[i - coin] != INT\_MAX) {

                    dp[i] = min(dp[i], dp[i - coin] + 1);

                }

            }

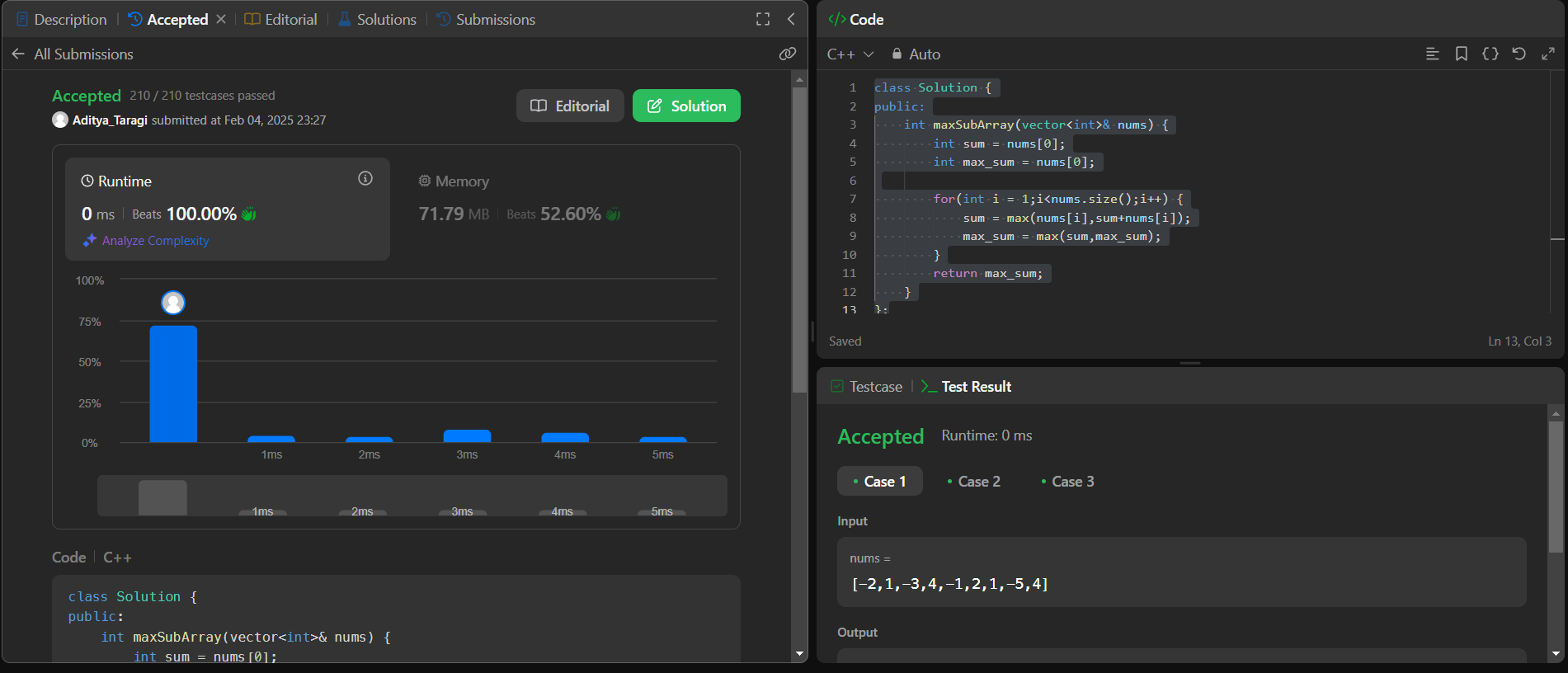
        }

        return dp[amount] == INT\_MAX ? -1 : dp[amount];

    }

};

**Screenshot:**



**Problem 6: Maximum Product Subarray (**<https://leetcode.com/problems/maximum-product-subarray/> **)**

**Code:**class Solution {

public:

    int maxProduct(vector<int>& nums) {

        int n = nums.size();

        int maxProduct = nums[0], currentMax = nums[0], currentMin = nums[0];

        for (int i = 1; i < n; ++i) {

            if (nums[i] < 0) {

                swap(currentMax, currentMin);

            }

            currentMax = max(nums[i], currentMax \* nums[i]);

            currentMin = min(nums[i], currentMin \* nums[i]);

            maxProduct = max(maxProduct, currentMax);

        }

        return maxProduct;

    }

};

**Screenshot:**

