**ADVANCED PROGRAMMING-II**

**ASSIGNMENT-07**

**Q1. Climbing Stairs:**

**Code:**

class Solution {

public:

    int climbStairs(int n) {

    if (n == 1) return 1;

    if (n == 2) return 2;

    int prev2 = 1, prev1 = 2, curr;

    for (int i = 3; i <= n; i++) {

        curr = prev1 + prev2;

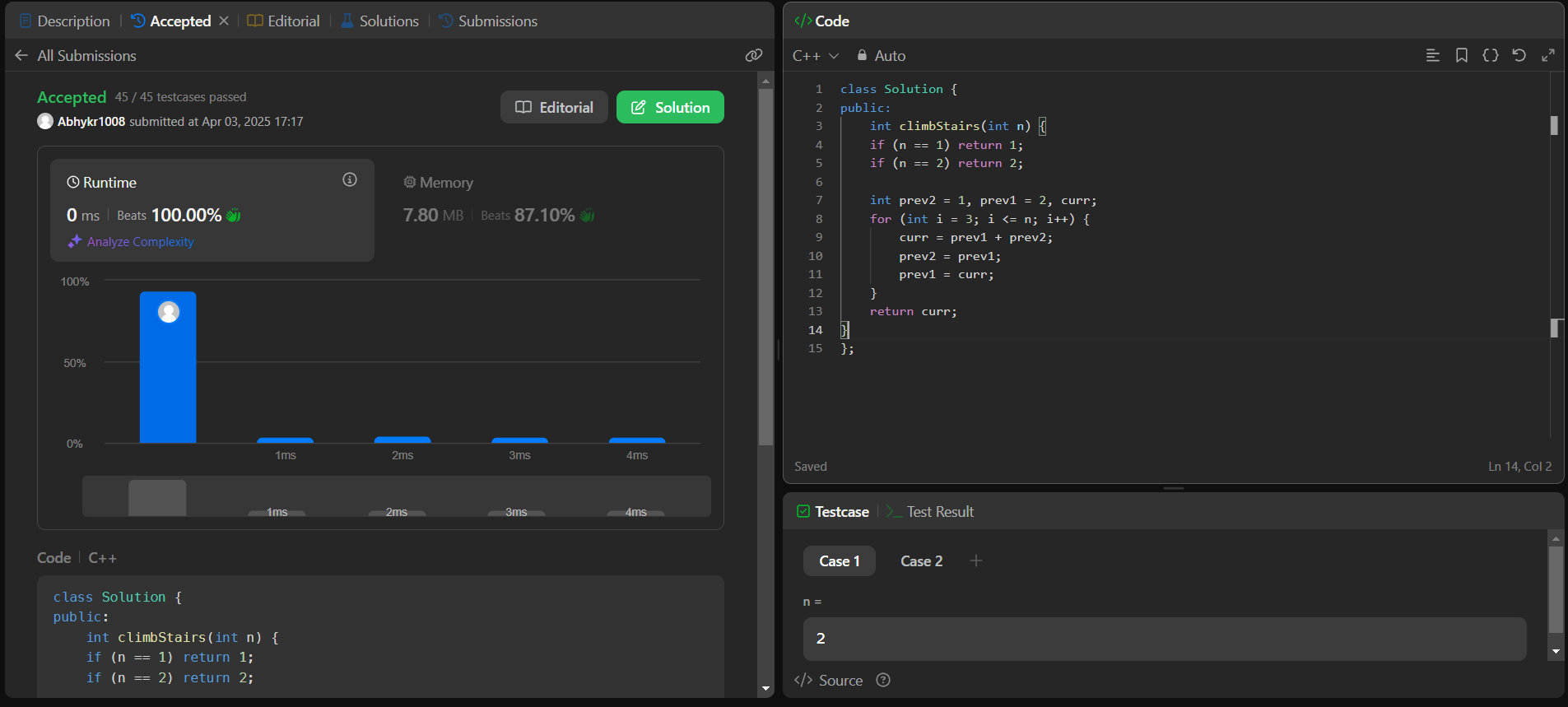
        prev2 = prev1;

        prev1 = curr;

}

    return curr;

}};

**SCREENSHOT: **

**Q2. Best Time to Buy and Sell a Stock:**

**Code:**

class Solution {

public:

    int maxProfit(vector<int>& prices) {

    int minPrice = INT\_MAX;

    int maxProfit = 0;

    for (int price : prices) {

        minPrice = min(minPrice, price);

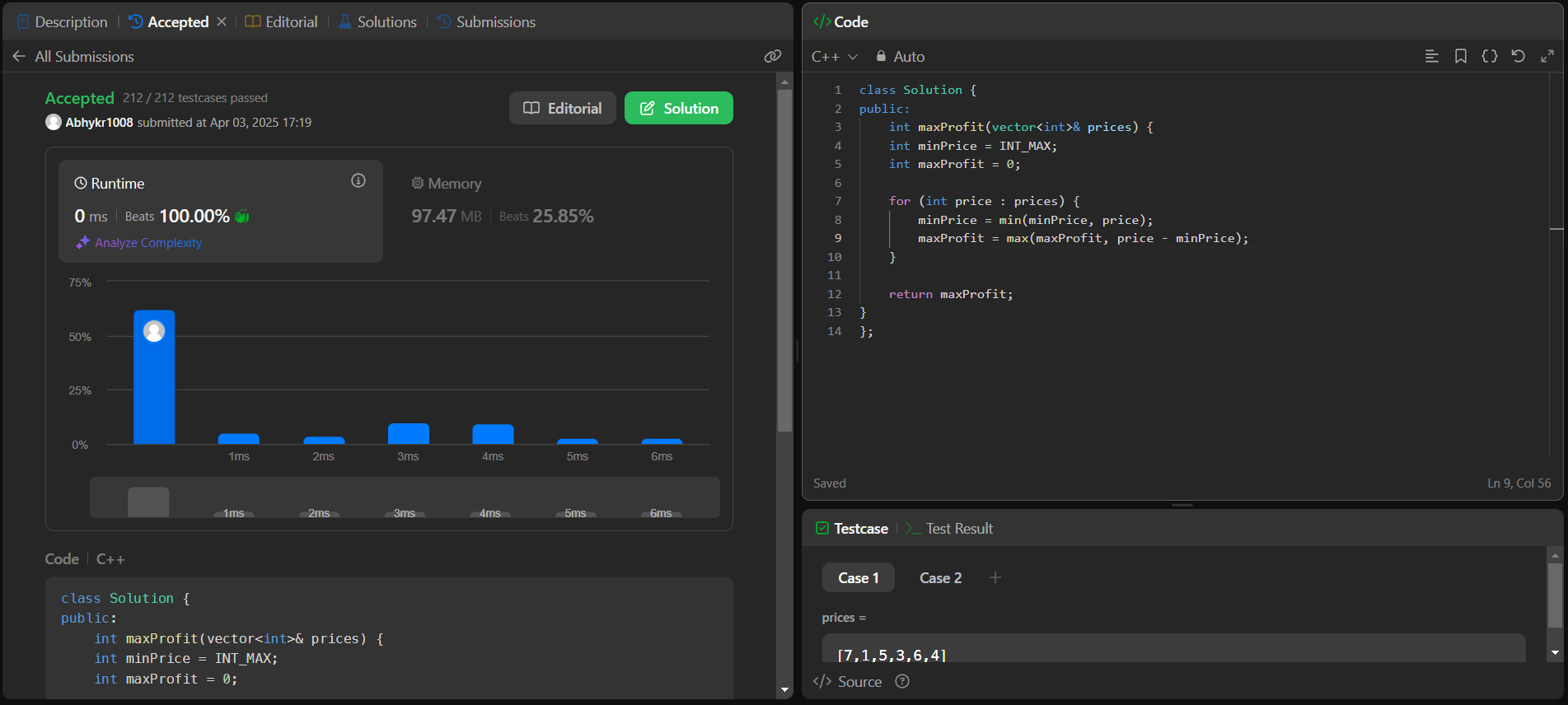
        maxProfit = max(maxProfit, price - minPrice);

    }

    return maxProfit;

}};

**Screenshot:**



**Q3.Jump Game:**

**Code:**

class Solution {

public:

    bool canJump(vector<int>& nums) {

    int maxReach = 0;

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

        if (i > maxReach) return false;

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

        if (maxReach >= nums.size() - 1) return true;

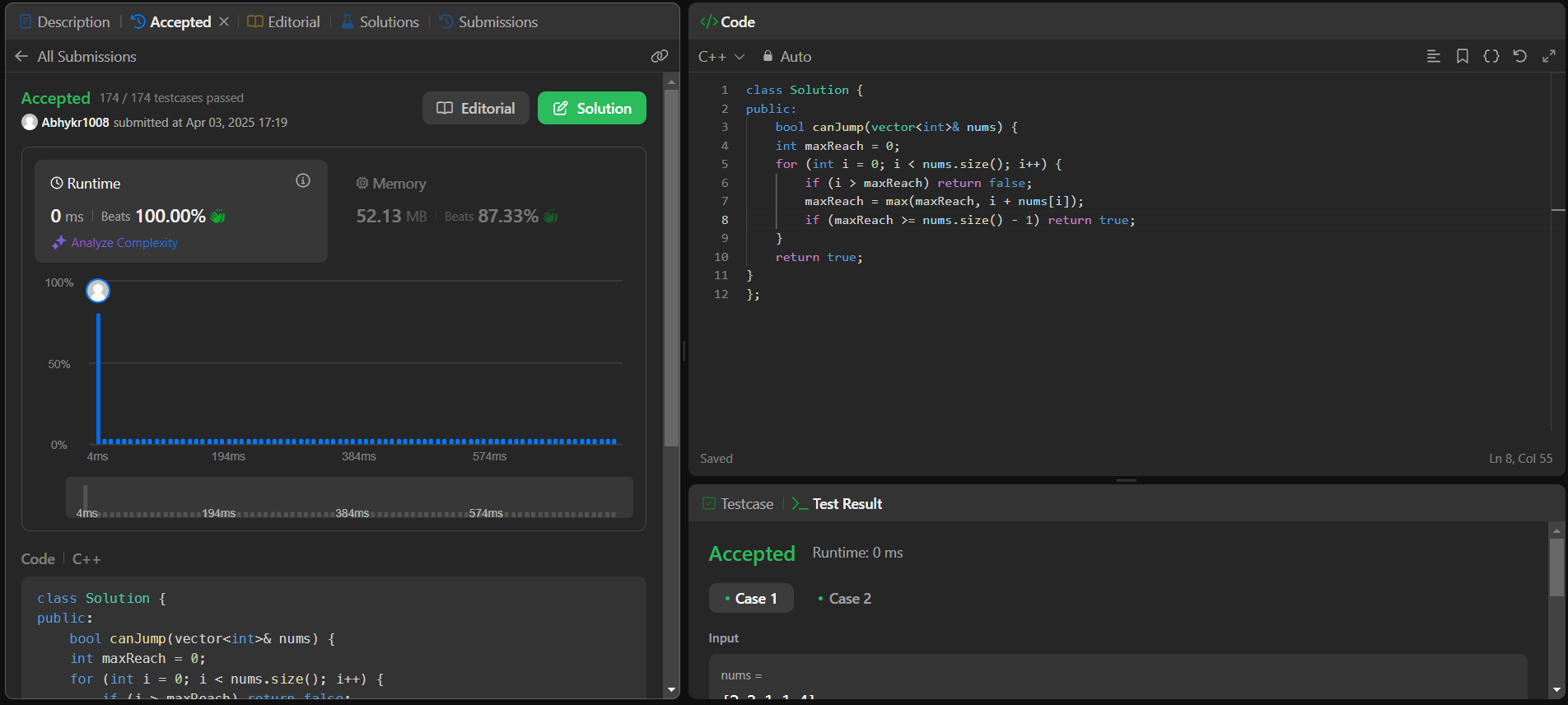
    }

    return true;

}

};

**Screenshot:**



**Q4.** [**Coin Change**](https://leetcode.com/problems/coin-change/)**:**

**Code:**

class Solution {

public:

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

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

    dp[0] = 0;

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

        for (int coin : coins) {

            if (i >= coin) {

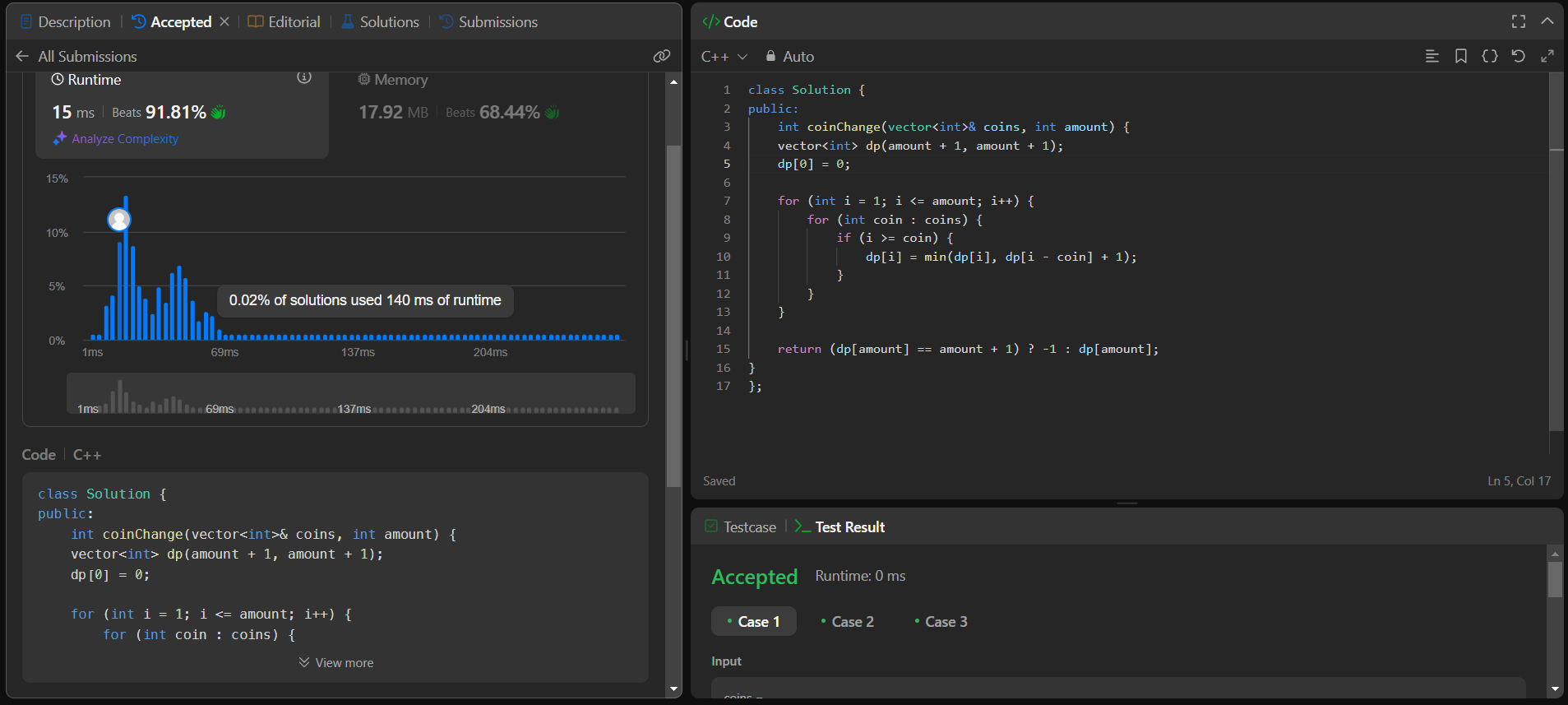
                dp[i] = min(dp[i], dp[i - coin] + 1);

            } }  }

    return (dp[amount] == amount + 1) ? -1 : dp[amount];

}};

**Screenshot:**

****