



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment 07

**Student Name:** Rajni Gandha

**Branch:** BE-IT

**Semester:** 06<sup>th</sup>

**Subject Name:** Advanced Programming-II

**UID:** 22BET10080

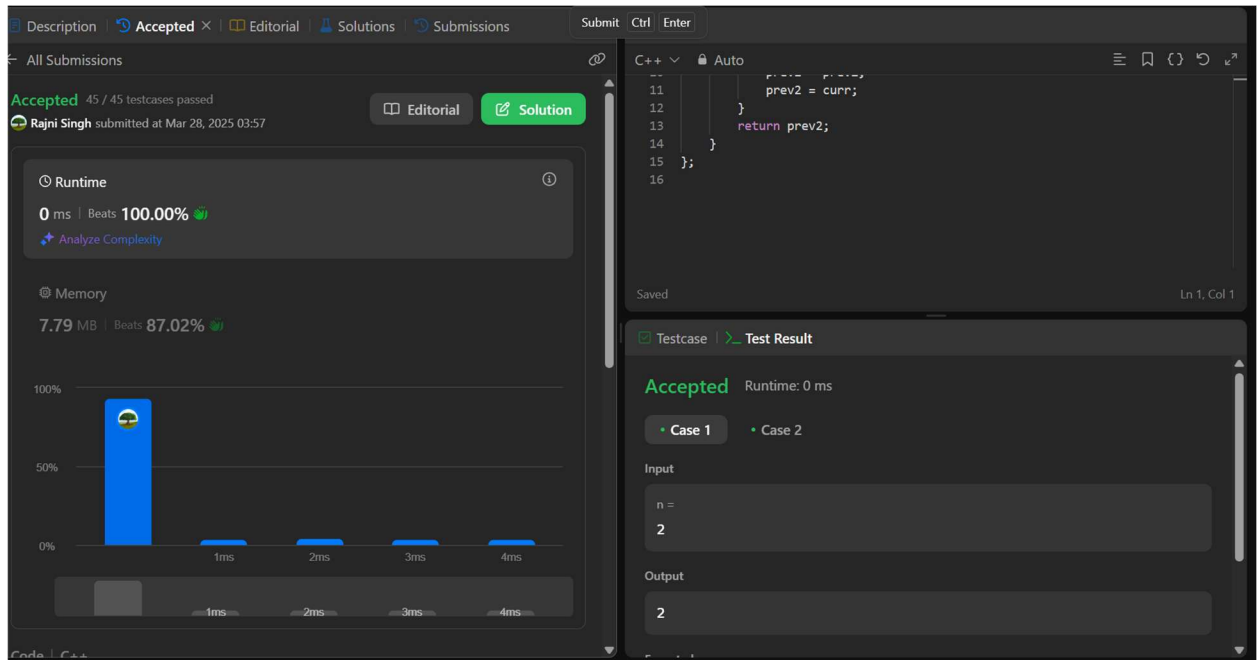
**Section/Group:** BET\_701/A

**Date of Performance:** 21-03-2025

**Subject Code:** 22ITP-351

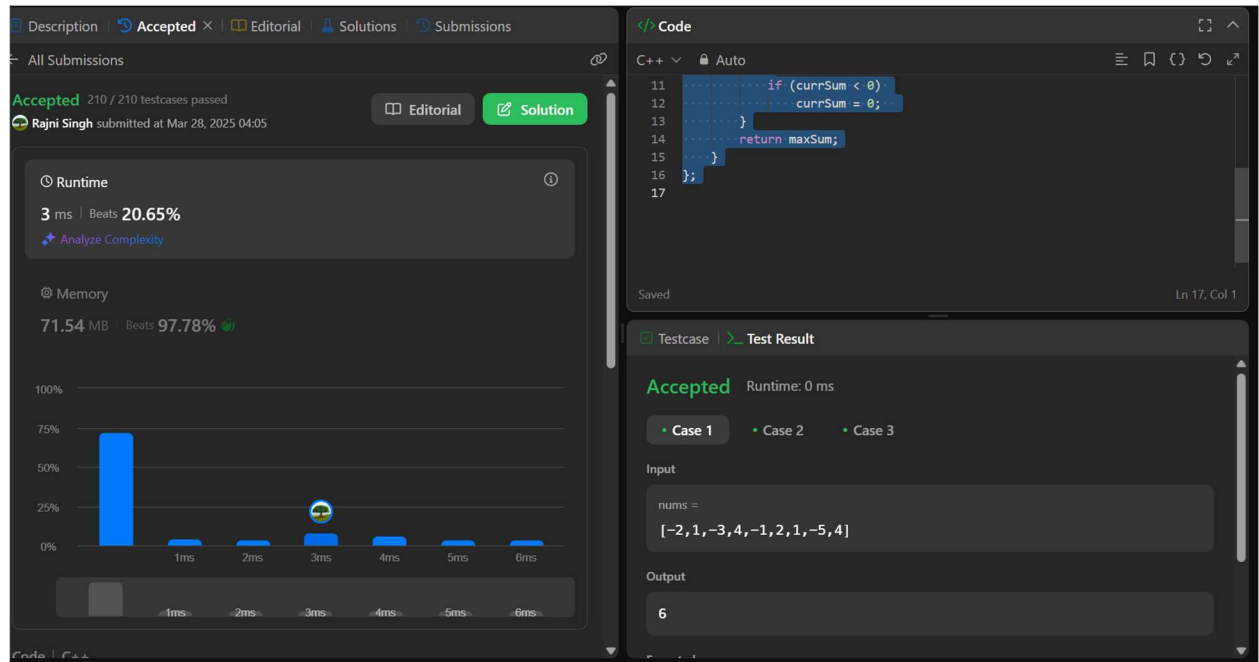
### 1. Problem: Climbing Stairs

```
class Solution {  
public:  
    int climbStairs(int n) {  
        if (n == 1) return 1;  
        if (n == 2) return 2;  
  
        int prev1 = 1, prev2 = 2;  
        for (int i = 3; i <= n; i++) {  
            int curr = prev1 + prev2;  
            prev1 = prev2;  
            prev2 = curr;  
        }  
        return prev2;  
    }  
};
```



## 2. Problem: Maximum Subarray

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0];
        int currSum = 0;
        for (int num : nums) {
            currSum += num;
            maxSum = max(maxSum, currSum);
            if (currSum < 0)
                currSum = 0;
        }
        return maxSum;
    }
};
```



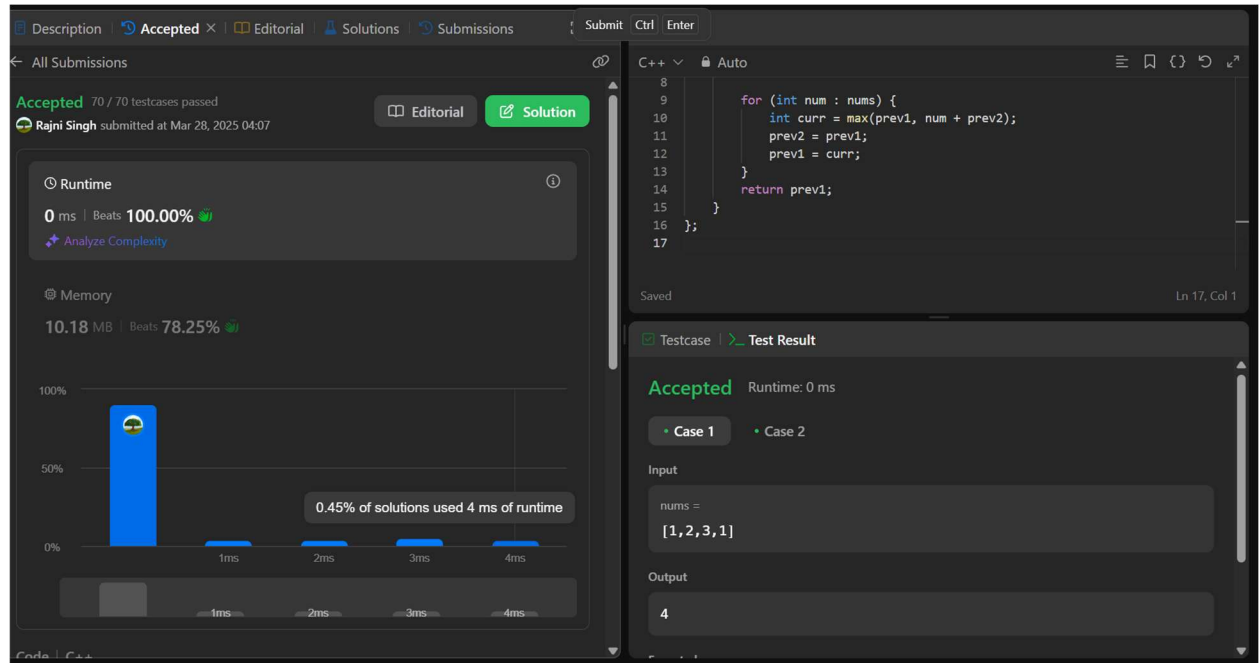
## 3. Problem: House Robber

```
class Solution {
public:
    int rob(vector<int>& nums) {
        if (nums.empty()) return 0;
```

```

    if (nums.size() == 1) return nums[0];
    int prev2 = 0, prev1 = 0; // prev2 -> dp[i-2], prev1 -> dp[i-1]
    for (int num : nums) {
        int curr = max(prev1, num + prev2);
        prev2 = prev1;
        prev1 = curr;
    }
    return prev1;
}
};

```

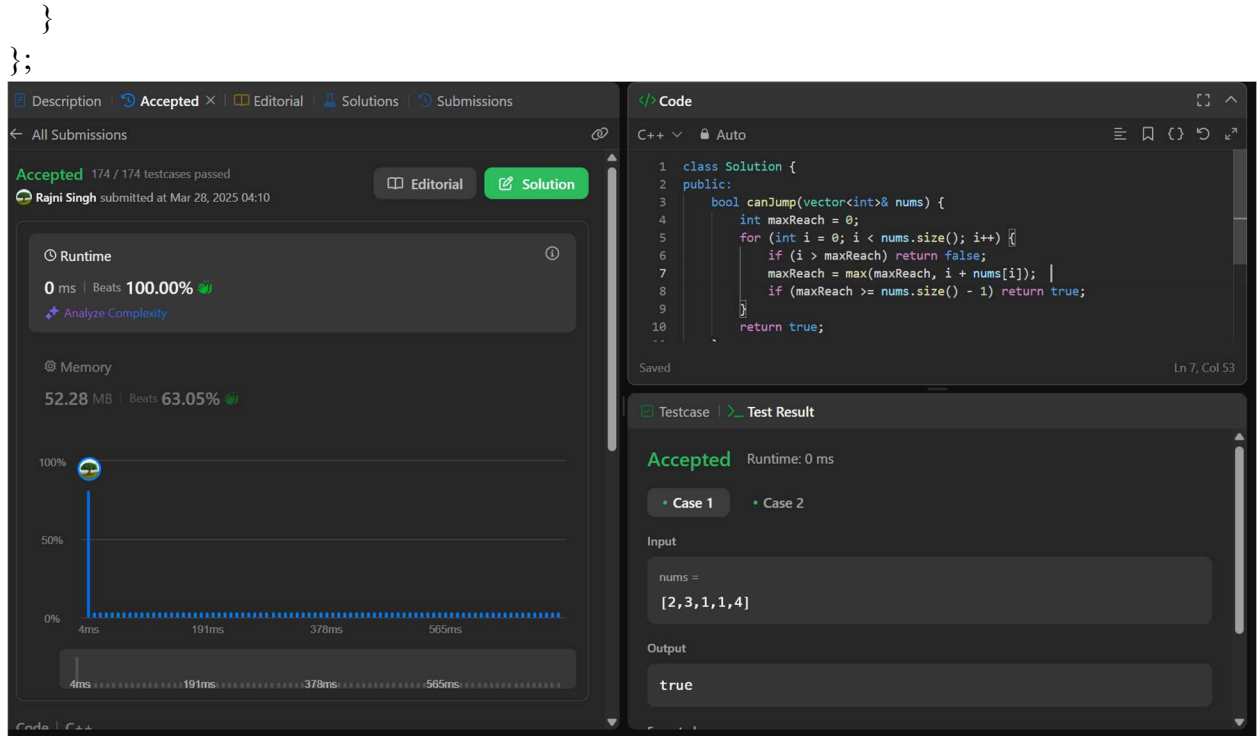


#### 4. Problem: Jump Game

```

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;
    }
};

```



## 5. Problem: Unique Paths

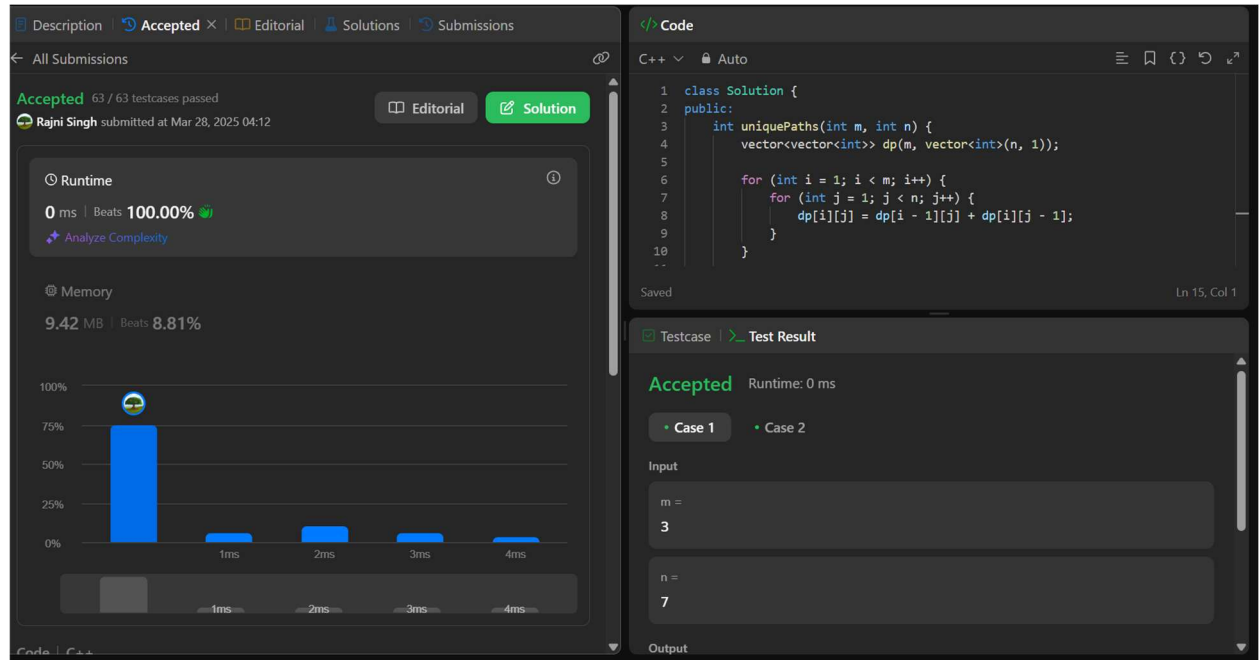
```

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];
    }
};

```

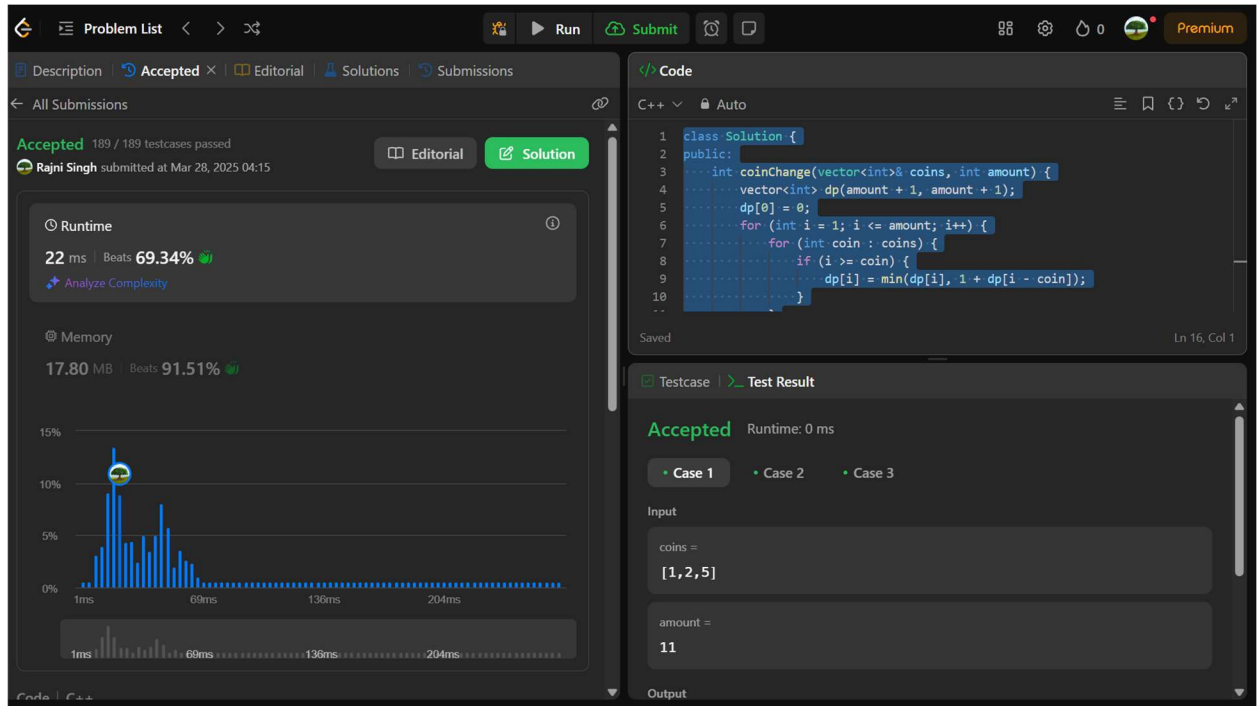


## 6. Problem: Coin Change

```

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], 1 + dp[i - coin]);
                }
            }
        }
        return (dp[amount] == amount + 1) ? -1 : dp[amount];
    }
};

```



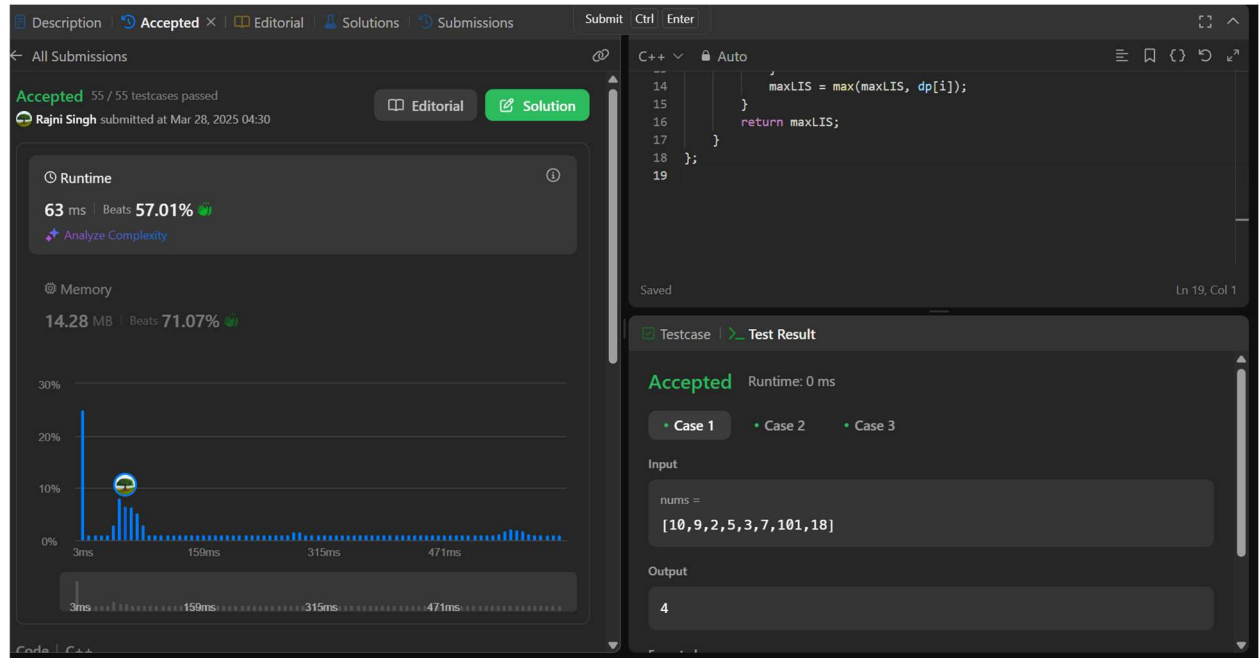
## 7. Problem: Longest Increasing Subsequence

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {
        int n = nums.size();
        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;
    }
};
```



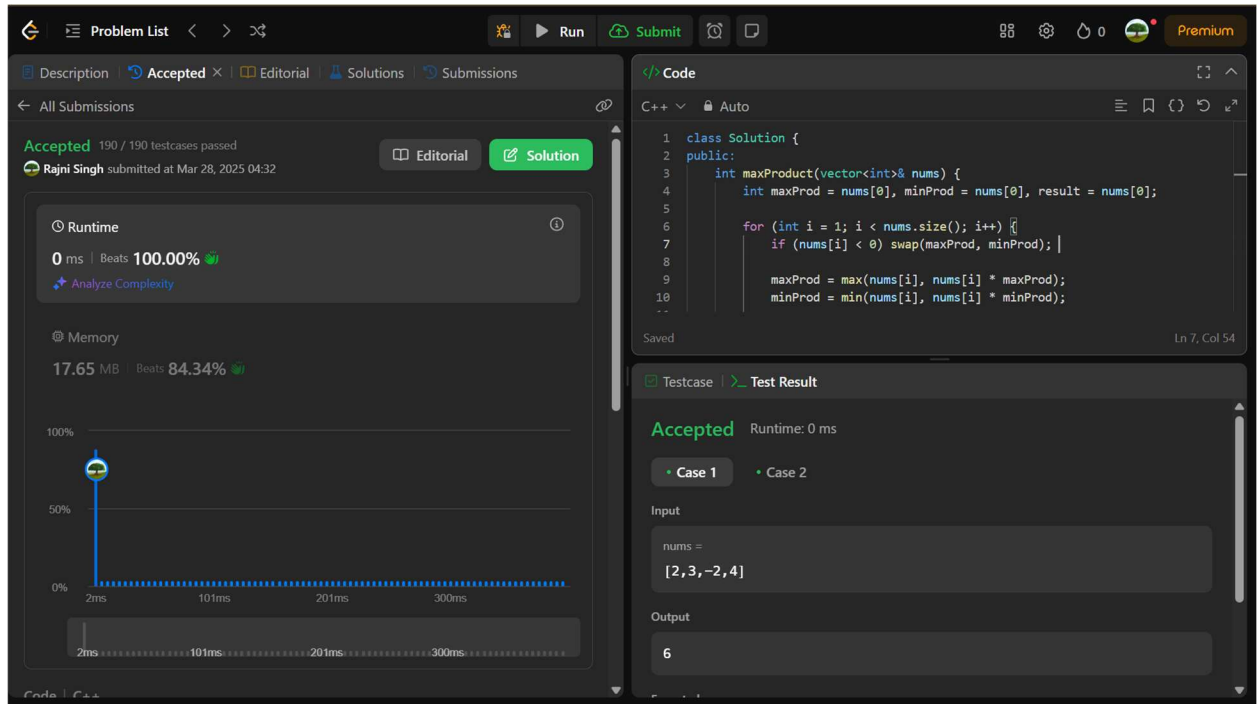
# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.



## 8. Problem: Maximum Product Subarray

```
class Solution {
public:
    int maxProduct(vector<int>& nums) {
        int maxProd = nums[0], minProd = nums[0], result = nums[0];
        for (int i = 1; i < nums.size(); i++) {
            if (nums[i] < 0) swap(maxProd, minProd);
            maxProd = max(nums[i], nums[i] * maxProd);
            minProd = min(nums[i], nums[i] * minProd);
            result = max(result, maxProd);
        }
        return result;
    }
};
```



The screenshot shows a coding competition interface. On the left, there's a 'Problem List' and 'Accepted' status. The main area displays a C++ solution for a problem. The code is as follows:

```

1 class Solution {
2 public:
3     int maxProduct(vector<int>& nums) {
4         int maxProd = nums[0], minProd = nums[0], result = nums[0];
5
6         for (int i = 1; i < nums.size(); i++) {
7             if (nums[i] < 0) swap(maxProd, minProd);
8
9             maxProd = max(nums[i], nums[i] * maxProd);
10            minProd = min(nums[i], nums[i] * minProd);
11        }
12        return maxProd;
13    }
14 }

```

Below the code, there's a 'Testcase' section showing 'Accepted' status with 'Runtime: 0 ms'. The input is 'nums = [2,3,-2,4]' and the output is '6'.

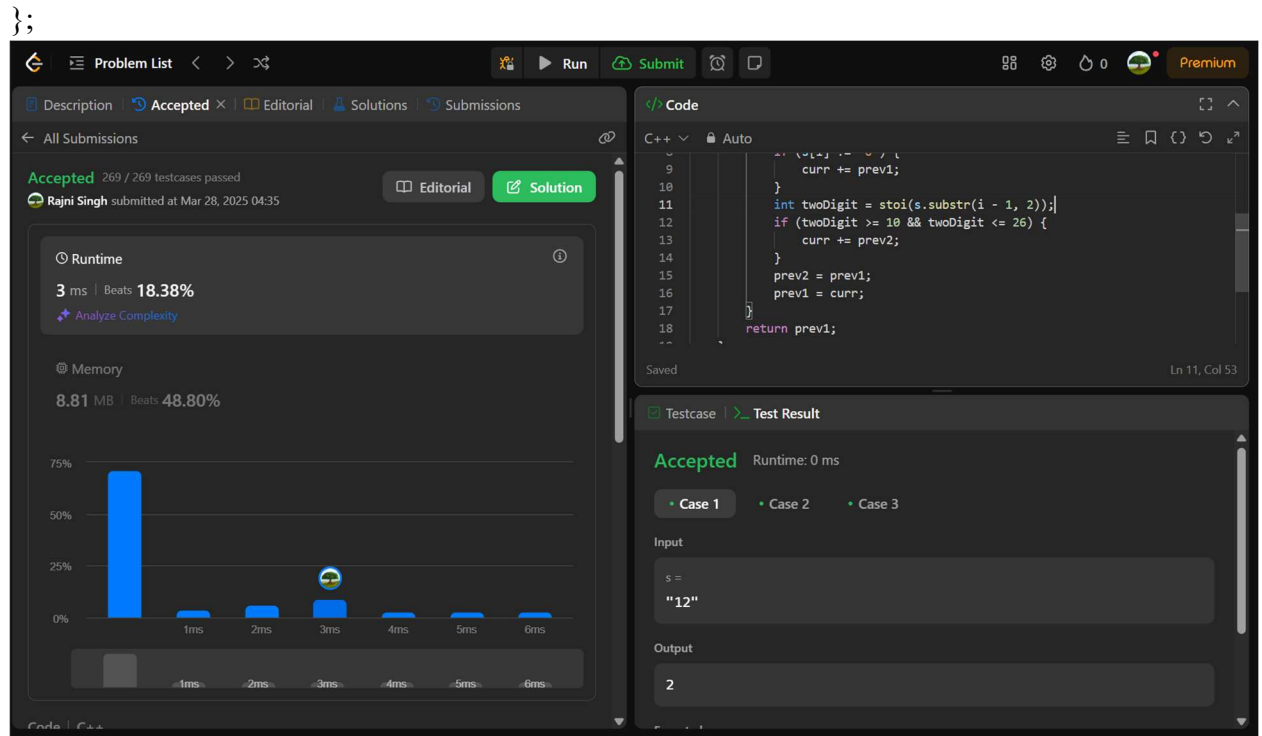
## 9. Problem: Decode Ways

```

class Solution {
public:
    int numDecodings(string s) {
        if (s.empty() || s[0] == '0') return 0;
        int prev1 = 1, prev2 = 1;
        for (int i = 1; i < s.size(); i++) {
            int curr = 0;
            if (s[i] != '0') {
                curr += prev1;
            }
            int twoDigit = stoi(s.substr(i - 1, 2));
            if (twoDigit >= 10 && twoDigit <= 26) {
                curr += prev2;
            }
            prev2 = prev1;
            prev1 = curr;
        }
        return prev1;
    }
}

```

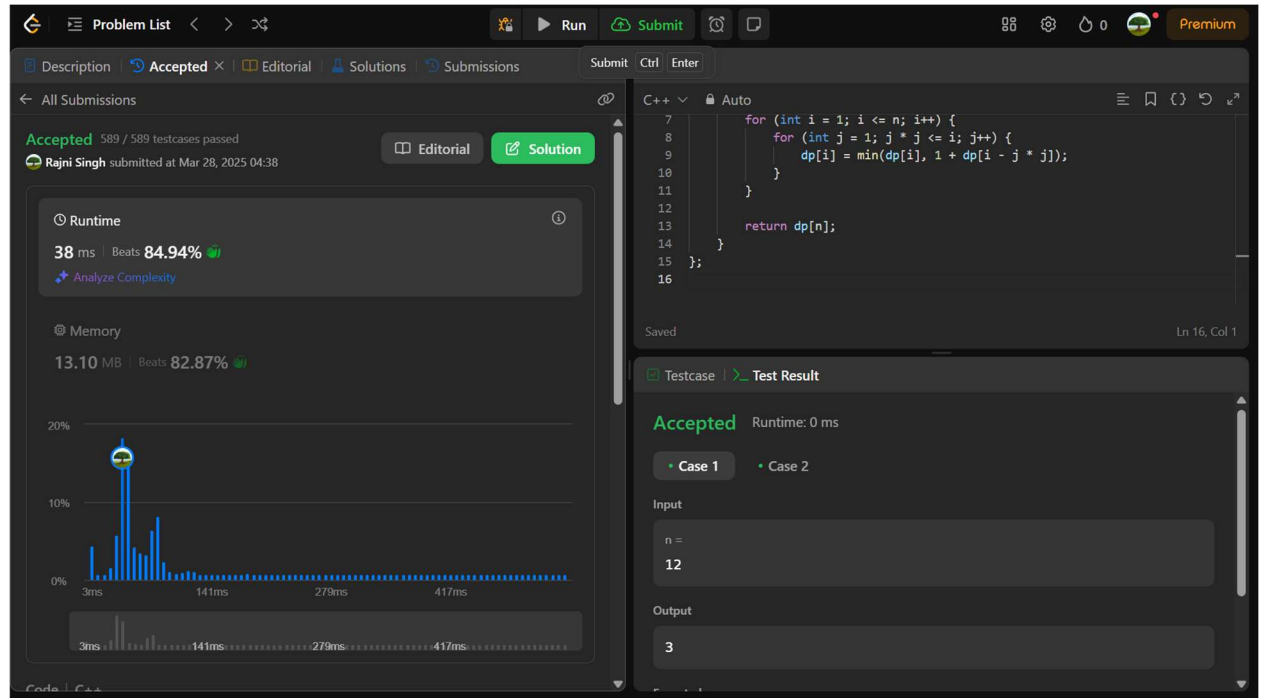




## 10. Problem: Perfect squares

```
class Solution {
public:
    int numSquares(int n) {
        vector<int> dp(n + 1, INT_MAX);
        dp[0] = 0;

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j * j <= i; j++) {
                dp[i] = min(dp[i], 1 + dp[i - j * j]);
            }
        }
        return dp[n];
    }
};
```



## 11. Problem: Word Break

```

class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {
        unordered_set<string> wordSet(wordDict.begin(), wordDict.end());
        int n = s.size();
        vector<bool> dp(n + 1, false);
        dp[0] = true;
        for (int i = 1; i <= n; i++) {
            for (int j = 0; j < i; j++) {
                if (dp[j] && wordSet.find(s.substr(j, i - j)) != wordSet.end()) {
                    dp[i] = true;
                    break;
                }
            }
        }
        return dp[n];
    }
}

```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

};

The screenshot displays a coding competition interface with a dark theme. The top navigation bar includes 'Problem List', 'Run', 'Submit', and 'Premium'. The main content area is divided into three sections: 'All Submissions', 'Runtime', and 'Memory'. The 'All Submissions' section shows a submission by 'Rajni Singh' with a status of 'Accepted' and '47 / 47 testcases passed'. The 'Runtime' section shows a runtime of '15 ms' and 'Beats 32.36%'. The 'Memory' section shows a memory usage of '15.89 MB' and 'Beats 49.69%'. A bar chart below the memory section shows the distribution of memory usage across different test cases. The right side of the interface features a code editor with a C++ solution for a word break problem. The code defines a 'Solution' class with a 'wordBreak' method that uses a dynamic programming approach to determine if a string can be segmented into words from a given dictionary. The 'Testcase' section shows the input string 'leetcode' and the word dictionary ['leet', 'code'], with the output being 'Accepted'.

Accepted 47 / 47 testcases passed  
Rajni Singh submitted at Mar 28, 2025 04:40

Runtime  
15 ms | Beats 32.36%  
Analyze Complexity

Memory  
15.89 MB | Beats 49.69%

30%  
20%  
10%  
0%  
5ms 10ms 15ms 20ms 25ms 30ms 35ms

Code | C++

```
1 class Solution {  
2 public:  
3     bool wordBreak(string s, vector<string>& wordDict) {  
4         unordered_set<string> wordSet(wordDict.begin(), wordDict.end());  
5         int n = s.size();  
6         vector<bool> dp(n + 1, false);  
7         dp[0] = true;  
8         for (int i = 1; i <= n; i++) {  
9             for (int j = 0; j < i; j++) {  
10                 if (dp[j] && wordSet.find(s.substr(j, i - j)) != wordSet.end()) {  
11                     dp[i] = true;  
12                 }  
13             }  
14         }  
15         return dp[n];  
16     }  
17 };
```

Testcase | Test Result  
Accepted Runtime: 0 ms  
Case 1 Case 2 Case 3  
Input  
s =  
"leetcode"  
wordDict =  
["leet", "code"]  
Output