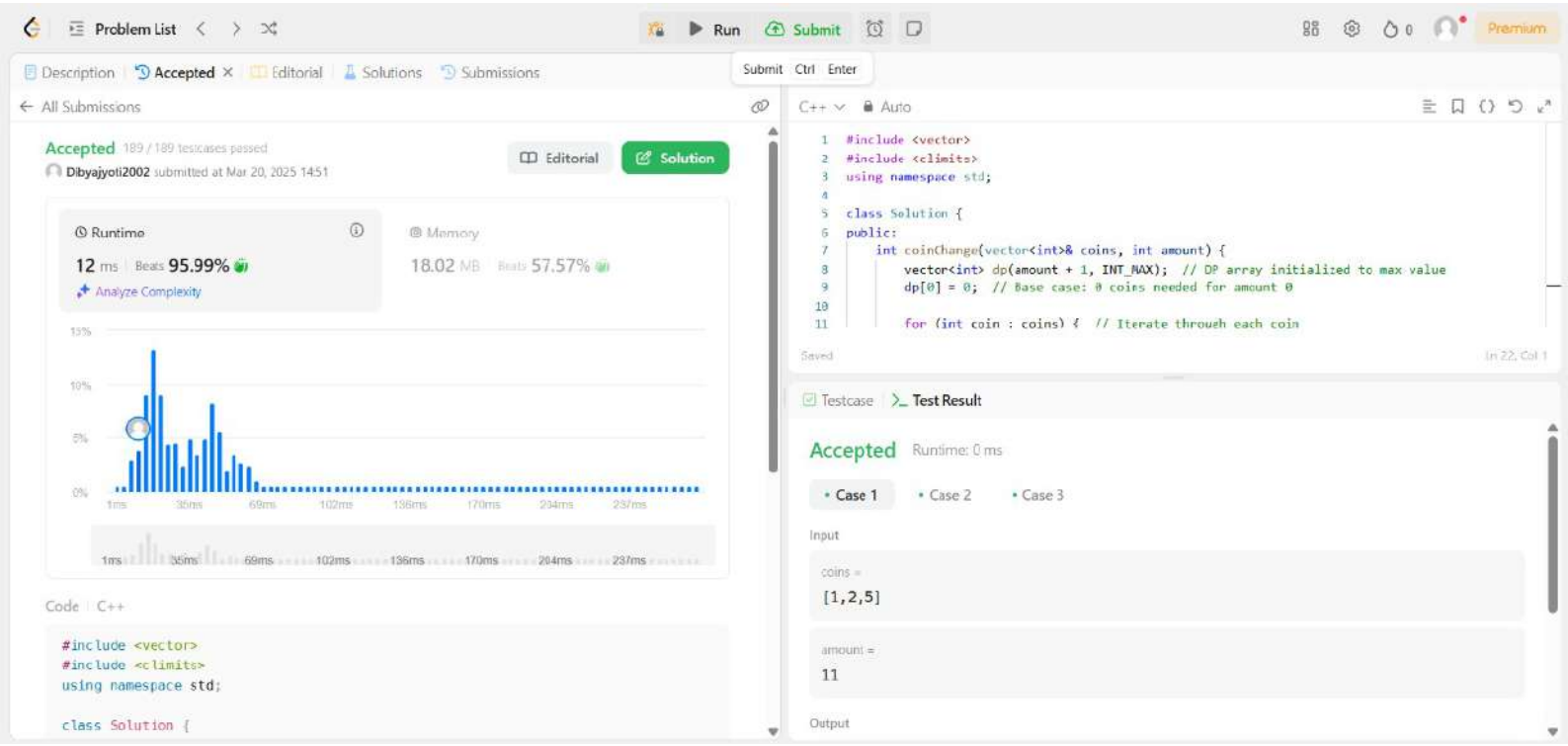
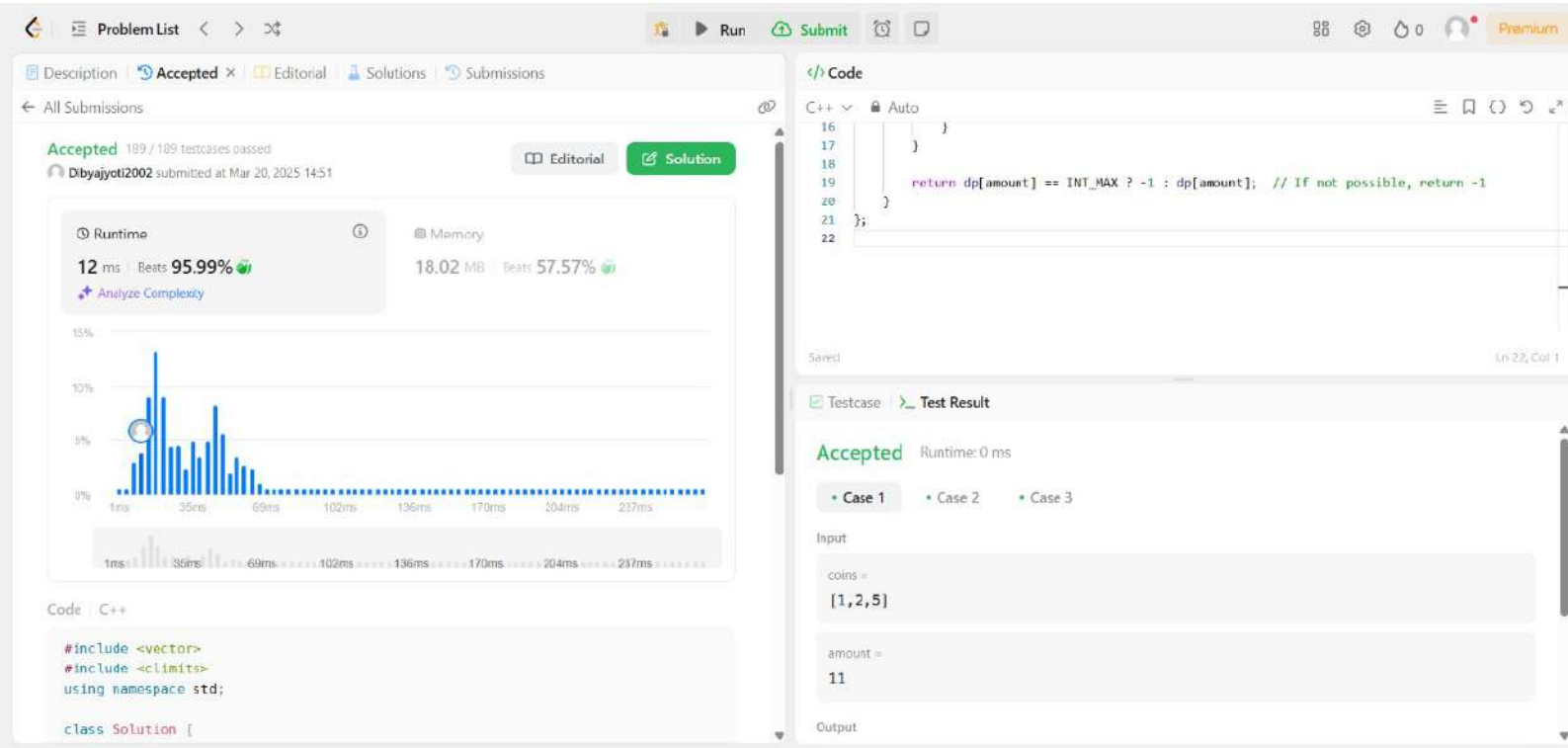
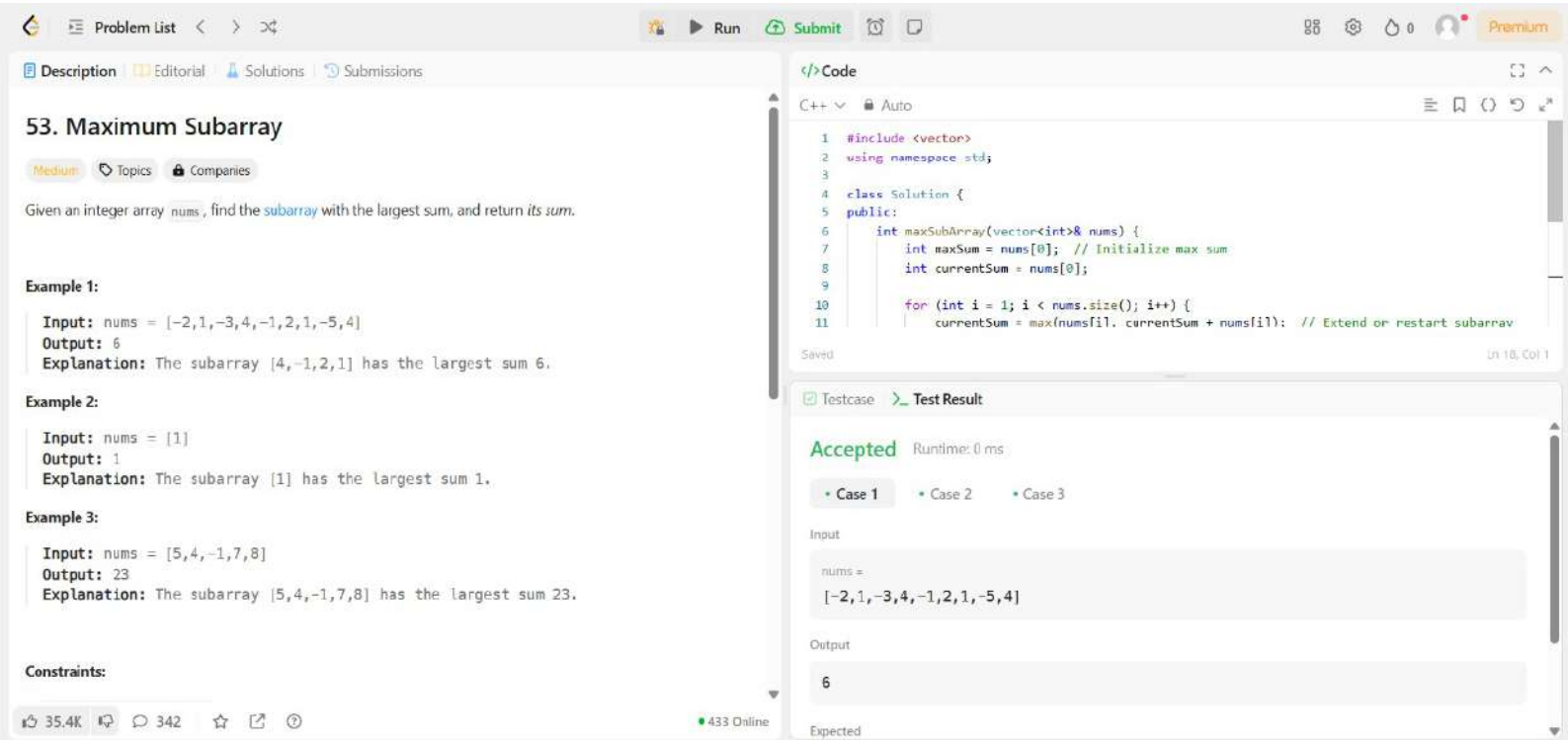

Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
Submit Ctrl Enter
All Submissions
Accepted 189 / 189 testcases passed
Dibyajyoti2002 submitted at Mar 20, 2025 14:51
Editorial
Solution
Runtime
12 ms | Beats 95.99%
Analyze Complexity
Memory
18.02 MB Beats 57.57%
13%
10%
5%
0%
1ms
35ms
69ms
102ms
136ms
170ms
204ms
237ms
Code | C++
#include <vector>
#include <climits>
using namespace std;
class Solution {
C++ Auto
1 #include <vector>
2 #include <climits>
3 using namespace std;
4
5 class Solution {
6 public:
7 int coinChange(vector<int>& coins, int amount) {
8 vector<int> dp(amount + 1, INT_MAX); // DP array initialized to max value
9 dp[0] = 0; // Base case: 0 coins needed for amount 0
10
11 for (int coin : coins) { // Iterate throueh each coin
Saved
Ln 22, Col 1
Testcase | Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
coins =
[1,2,5]
amount =
11
Output


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 189 / 189 testcases passed
Dibyajyoti2002 submitted at Mar 20, 2025 14:51
Editorial
Solution
Runtime
12 ms | Beats 95.99%
Analyze Complexity
Memory
18.02 MB | Beats 57.57%
15%
10%
5%
0%
1ms
35ms
69ms
102ms
136ms
170ms
204ms
237ms
Code | C++
#include <vector>
#include <climits>
using namespace std;
class Solution {
Code
C++
Auto
}
}
return dp[amount] == INT_MAX ? -1 : dp[amount]; // If not possible, return -1
}
};
Saved
Ln 22, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
coins =
[1,2,5]
amount =
11
Output


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
53. Maximum Subarray
Medium
Topics
Companies
Given an integer array nums, find the subarray with the largest sum, and return its sum.
Example 1:
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.
Example 2:
Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.
Example 3:
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.
Constraints:
35.4K
342
433 Online
Code
C++
Auto
#include <vector>
using namespace std;
class Solution {
public:
int maxSubArray(vector<int>& nums) {
int maxSum = nums[0]; // Initialize max sum
int currentSum = nums[0];
for (int i = 1; i < nums.size(); i++) {
currentSum = max(nums[i], currentSum + nums[i]); // Extend or restart subarray
Saved
Ln 18, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
nums =
[-2,1,-3,4,-1,2,1,-5,4]
Output
6
Expected

Problem List

Run Submit

Premium

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

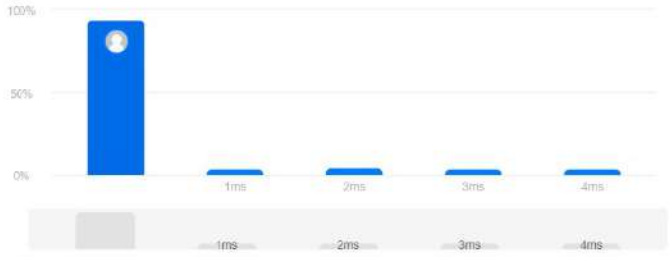
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

7.90 MB Beats 71.49%



Code | C++

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

Code

C++ Auto

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

Saved

Ln 14, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

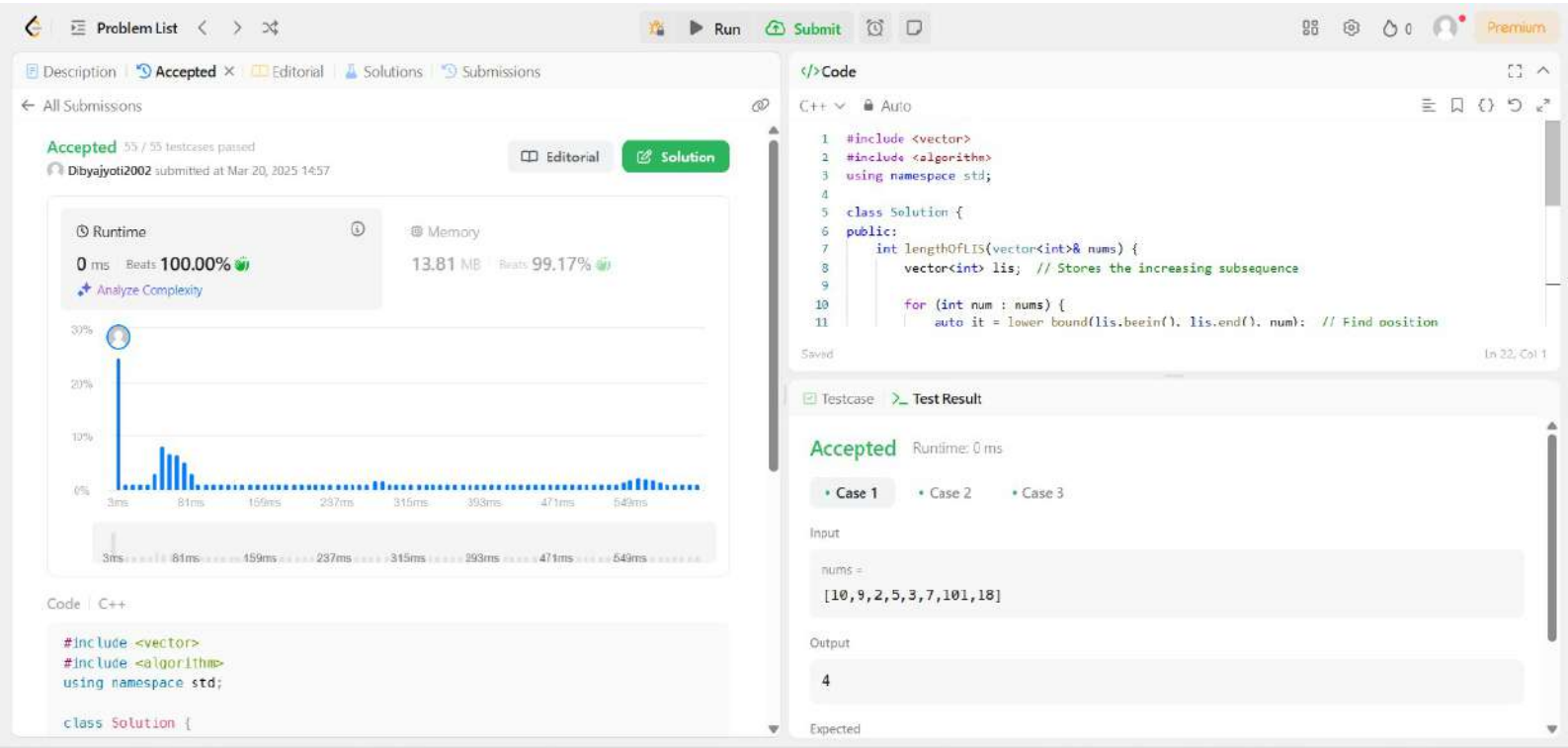
n =

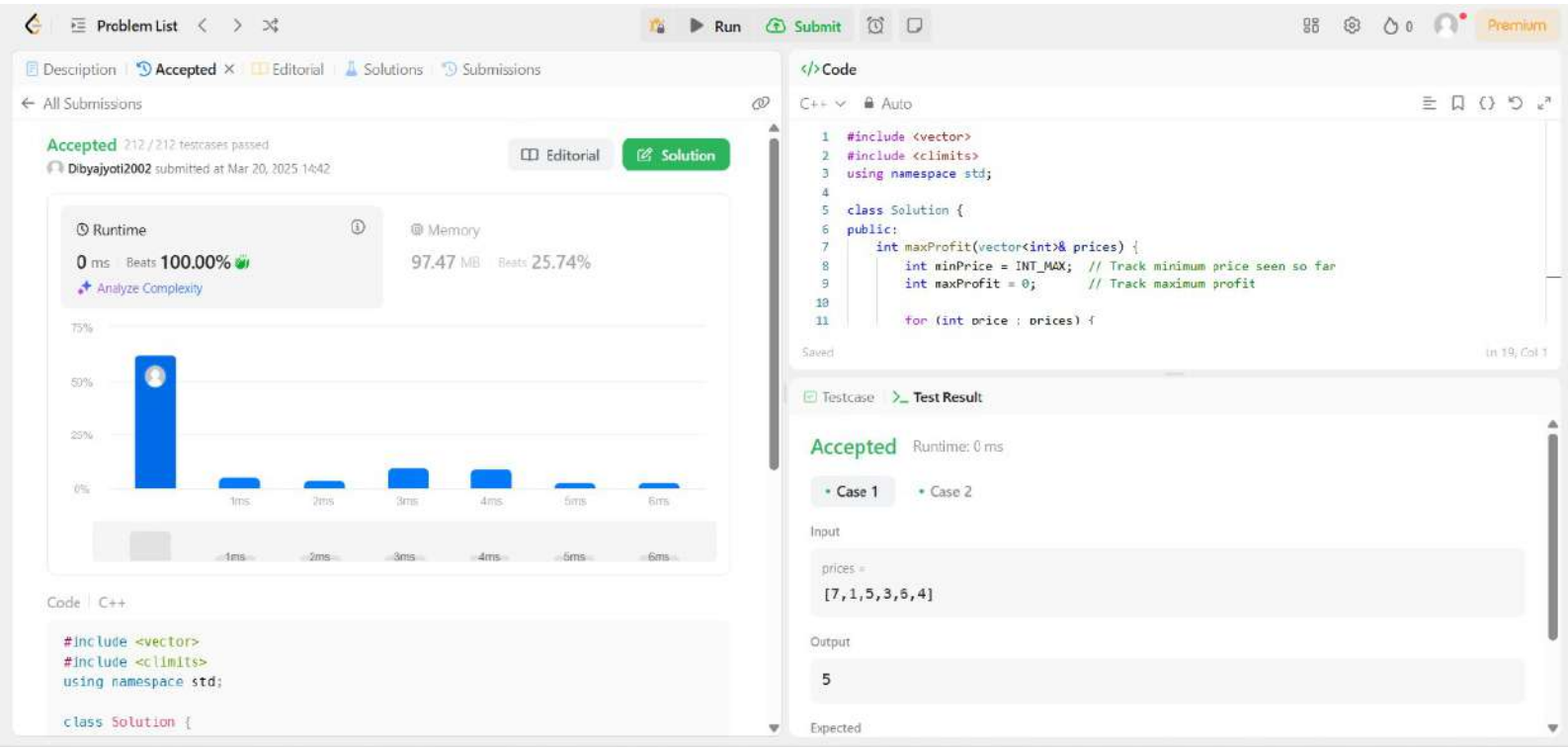
2

Output

2

Expected


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 55 / 55 testcases passed
Dibyajyoti2002 submitted at Mar 20, 2025 14:57
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
13.81 MB Beats 99.17%
30%
20%
10%
0%
3ms
81ms
159ms
237ms
315ms
393ms
471ms
549ms
Code | C++
#include <vector>
#include <algorithm>
using namespace std;
class Solution {
Code
C++
Auto
#include <vector>
#include <algorithm>
using namespace std;
class Solution {
public:
int lengthOfLIS(vector<int>& nums) {
vector<int> lis; // Stores the increasing subsequence
for (int num : nums) {
auto it = lower_bound(lis.begin(), lis.end(), num); // Find position
Saved
Ln 22, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
nums =
[10,9,2,5,3,7,101,18]
Output
4
Expected


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 212 / 212 testcases passed
Dibyajyoti2002 submitted at Mar 20, 2025 14:42
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
97.47 MB Beats 25.74%
75%
50%
25%
0%
1ms
2ms
3ms
4ms
5ms
6ms
Code C++
#include <vector>
#include <climits>
using namespace std;
class Solution {
Code
C++ Auto
#include <vector>
#include <climits>
using namespace std;
class Solution {
public:
int maxProfit(vector<int>& prices) {
int minPrice = INT_MAX; // Track minimum price seen so far
int maxProfit = 0; // Track maximum profit
for (int price : prices) {
Saved
Ln 19, Col 1
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Input
prices =
[7,1,5,3,6,4]
Output
5
Expected