

Name: Akash Pandey
Sec: FL_IOT-601/ A

UID: 22BCS11135
Sub: AP Lab -II

Climbing Stairs

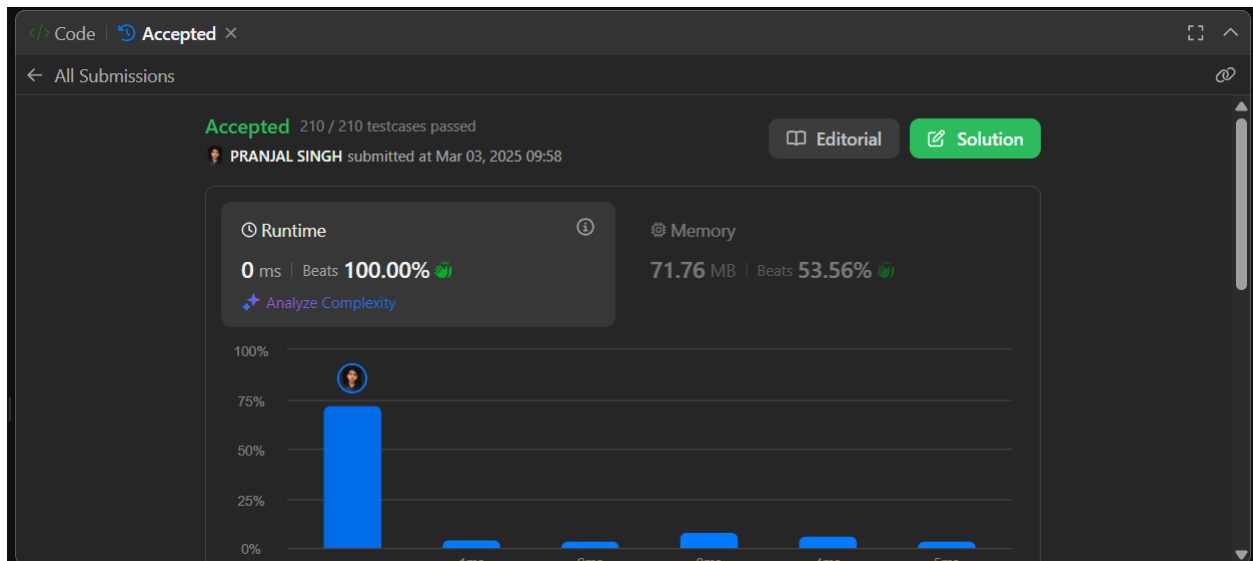
```
class Solution {  
public:  
    int climbStairs(int n) {  
        if (n <= 2) return n;  
  
        vector<int> dp(n + 1, 0);  
        dp[1] = 1;  
        dp[2] = 2;  
  
        for (int i = 3; i <= n; i++) {  
            dp[i] = dp[i - 1] + dp[i - 2];  
        }  
  
        return dp[n];  
    }  
};
```

The screenshot shows a coding platform interface with the following components:

- Problem List:** A navigation bar at the top with icons for problem list, description, editorial, solutions, accepted, and submissions.
- Accepted Submissions:** A section showing that 45 out of 45 testcases passed. The user's submission ID is 22bcs13041, submitted on April 05, 2025, at 12:50. It includes buttons for 'Editorial' and 'Solution'.
- Runtime and Memory:** The runtime is 0 ms, beating 100.00% of other solutions. The memory usage is 8.48 MB, beating 48.50% of other solutions. There is a link to 'Analyze Complexity'.
- Bar Chart:** A chart showing the user's performance relative to others. The user's bar is at 100%.
- Code Editor:** A section showing the C++ code for the solution. The code is saved and includes a test case input 'n = 2'.
- Testcase:** A section showing the test case input 'n = 2' and a button to view the test result.

Maximum Subarray

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        // Kadane's Algo...
        int maximumsum=INT_MIN;
        int currentsum=0;
        for(int i =0;i<nums.size();i++){
            currentsum=currentsum+nums[i];
            maximumsum=max(currentsum,maximumsum);
            if(currentsum<0){
                currentsum=0;
            }
        }
        return maximumsum;
    }
};
```



House Robber

```
#include <vector>
```

```
#include <algorithm>
```

```
using namespace std;
```

```
class Solution {
```

```
private:
```

```
    int rec(int i, vector<int>& nums, vector<int>& dp) {
```

```
        if (i >= nums.size()) return 0;
```

```
        if (dp[i] != -1) return dp[i];
```

```
        int take = nums[i] + rec(i + 2, nums, dp);
```

```
        int dont = rec(i + 1, nums, dp);
```

```
        return dp[i] = max(take, dont);
```

```
    }
```

```
public:
```

```
    int rob(vector<int>& nums) {
```

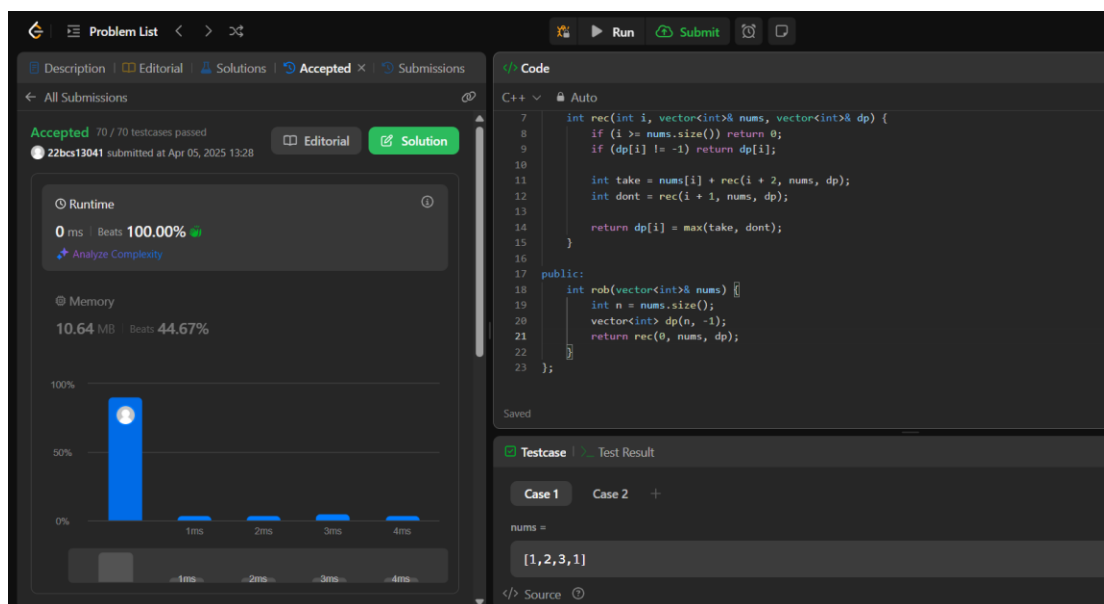
```
        int n = nums.size();
```

```
        vector<int> dp(n, -1);
```

```
        return rec(0, nums, dp);
```

```
    }
```

```
};
```



Jump Game

```
class Solution
{
public:
    bool canJump(vector<int> &nums){
        if(nums.size() == 1) return true;
        int prevGreatestNum = nums[0];
        for (int i = 0; i < nums.size() - 1; i++){
            if (nums[i] != 0){
                if (nums[i] + i + 1 >= nums.size()) {
                    return true;
                }
            }
            else {
                if (prevGreatestNum <= 0) {
                    break;
                }
            }
            prevGreatestNum = max(prevGreatestNum, nums[i]);
            prevGreatestNum--;
        }
        return false;
    }
};
```

