Leetcode Problem 1. (Easy)

Climbing Stairs

You are climbing a staircase. It takes n steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Example 1:

Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps

Example 2:

Input: n = 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step

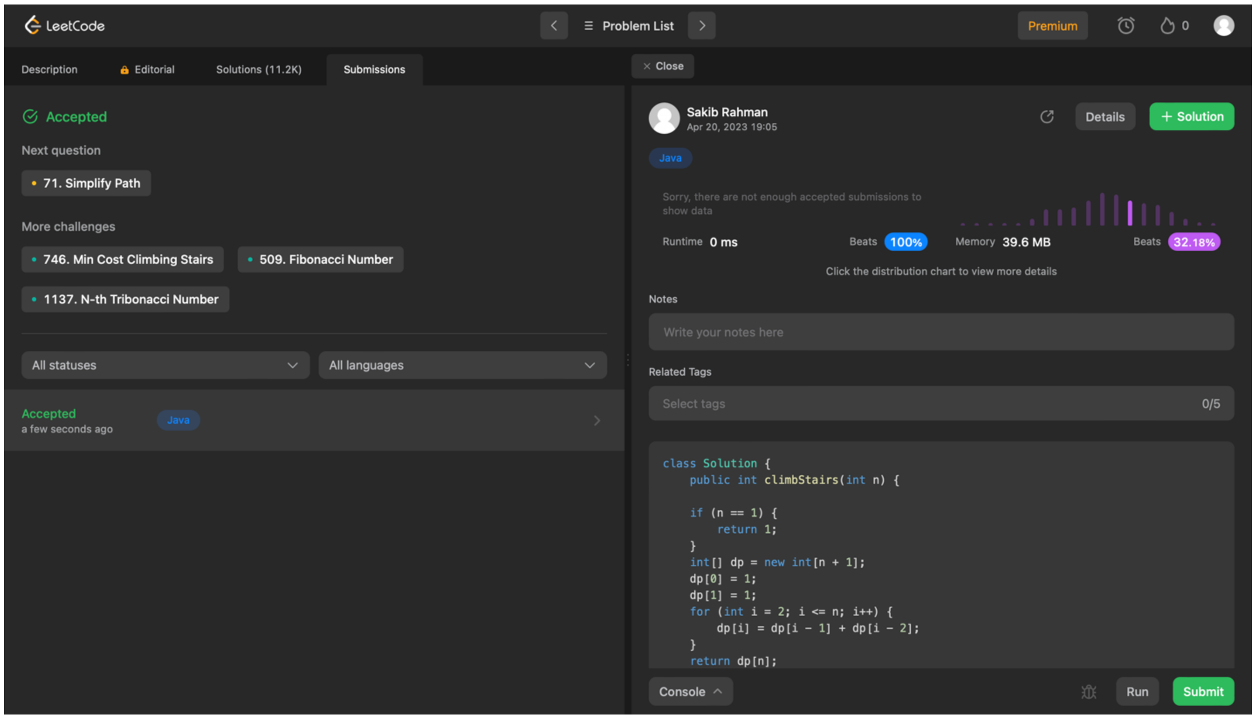
Constraints:

- 1 <= n <= 45

Link: https://leetcode.com/problems/climbing-stairs/

```
class Solution {
    public int climbStairs(int n) {

    if (n == 1) {
        return 1;
    }
    int[] dp = new int[n + 1];
    dp[0] = 1;
    dp[1] = 1;
    for (int i = 2; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 2];
    }
    return dp[n];
}
}
```



Accepted

Next question

71. Simplify Path

More challenges

746. Min Cost Climbing Stairs
509. Fibonacci Number
1137. N-th Tribonacci Number

All statuses
All languages

Accepted
a few seconds ago
Java

Sakib Rahman
Apr 20, 2023 19:05

Java

Sorry, there are not enough accepted submissions to show data

Runtime 0 ms Beats 100%
Memory 39.6 MB Beats 32.18%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```java
class Solution {
    public int climbStairs(int n) {

    if (n == 1) {
        return 1;
    }
    int[] dp = new int[n + 1];
    dp[0] = 1;
    dp[1] = 1;
    for (int i = 2; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 2];
    }
    return dp[n];
```