Description | Editorial | Solutions (12.7K) | Submissions

# 70. Climbing Stairs

Hint

Easy 19.6K 639

Companies

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`

Java | Auto

```java
class Solution {
    public int climbStairs(int n) {
        if(n==1) return 1;

        if(n==2) return 2;

        int[] a =  new int[n];
        a[0]=1;
        a[1]=2;

        for(int i=2;i<n;i++){
            a[i]=a[i-1]+a[i-2];
        }
        return a[n-1];
    }
}
```

Ln 16, Col 2

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

n =
2

Output

2

Console

Run Submit

Description | Editorial | Solutions (2.3K) | Submissions

## 68. Text Justification

Hard 2.8K 3.9K

Companies

Given an array of strings `words` and a width `maxWidth`, format the text such that each line has exactly `maxWidth` characters and is fully (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces `' '` when necessary so that each line has exactly `maxWidth` characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line does not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right.

For the last line of text, it should be left-justified, and no extra space is inserted between words.

**Note:**

- A word is defined as a character sequence consisting of non-space characters only.
- Each word's length is guaranteed to be greater than `0` and not exceed `maxWidth`.
- The input array `words` contains at least one word.

**Example 1:**

```
Input: words = ["This", "is", "an", "example", "of", "text",
"justification."], maxWidth = 16
Output:
[
   "This    is    an",
```

Java | Auto

```java
public class Solution {
    public List<String> fullJustify(String[] words, int maxWidth) {
        List<String> res = new ArrayList<>();
        List<String> cur = new ArrayList<>();
        int num_of_letters = 0;

        for (String word : words) {
            if (word.length() + cur.size() + num_of_letters > maxWidth) {
                for (int i = 0; i < maxWidth - num_of_letters; i++) {
                    cur.set(i % (cur.size() - 1 > 0 ? cur.size() - 1 : 1), cur.get(i %
(cur.size() - 1 > 0 ? cur.size() - 1 : 1)) + " ");
                }
                StringBuilder sb = new StringBuilder();
                for (String s : cur) sb.append(s);
                res.add(sb.toString());
                cur.clear();
                num_of_letters = 0;
            }
```

Ln 32, Col 2

Testcase | Result

Accepted Runtime: 1 ms

Case 1 | Case 2 | Case 3

Input

words =

```
["This", "is", "an", "example", "of", "text", "justification."]
```

maxWidth =

```
16
```

Output

Console | Run | Submit