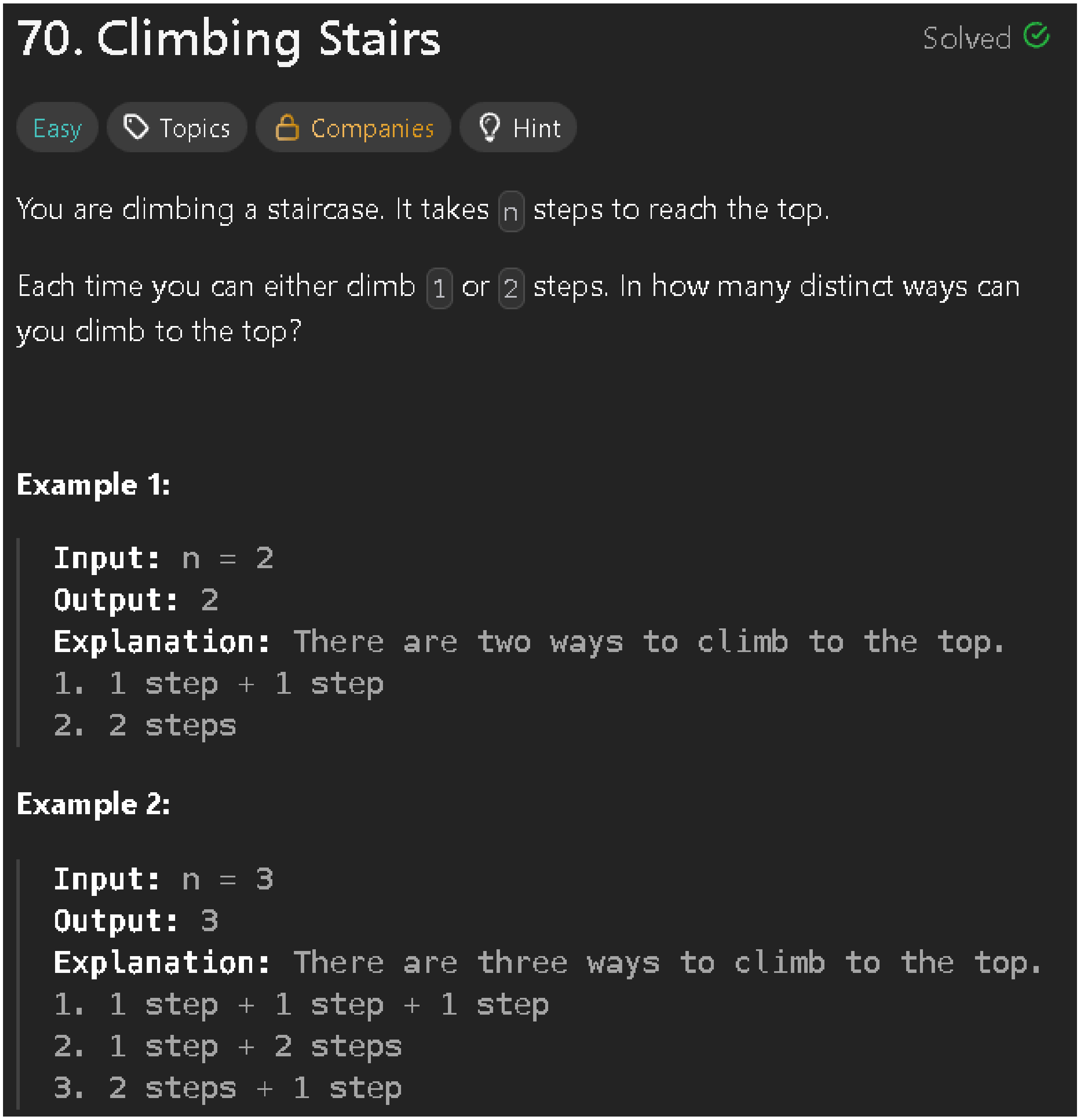


70. Climbing Stairs

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

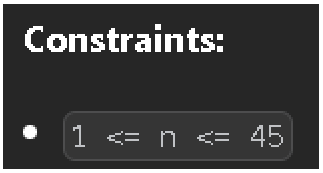
Each time you can either climb `1` or `2` steps. In how many distinct ways can you climb to the top?

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`

Python:

```
class Solution:
    def climbStairs(self, n: int) -> int:
        if n == 1:
            return 1
        if n == 2:
            return 2

        # Dynamic Programming approach
        first, second = 1, 2  # base cases
        for i in range(3, n + 1):
            first, second = second, first + second
        return second
```

JavaScript:

```
/**
 * @param {number} n
 * @return {number}
 */
var climbStairs = function(n) {
    if (n <= 2) return n;  // base cases

    let first = 1; // ways(1)
    let second = 2; // ways(2)
    let result = 0;

    for (let i = 3; i <= n; i++) {
        result = first + second; // ways(i-1) + ways(i-2)
        first = second; // shift window
        second = result;
    }

    return second;
};
```

Java:

```
class Solution {
    public int climbStairs(int n) {
        if (n == 1) return 1;
        if (n == 2) return 2;

        int first = 1; // ways(1)
        int second = 2; // ways(2)
        int result = 0;

        for (int i = 3; i <= n; i++) {
            result = first + second; // recurrence
            first = second;
            second = result;
        }
        return result;
    }
}
```