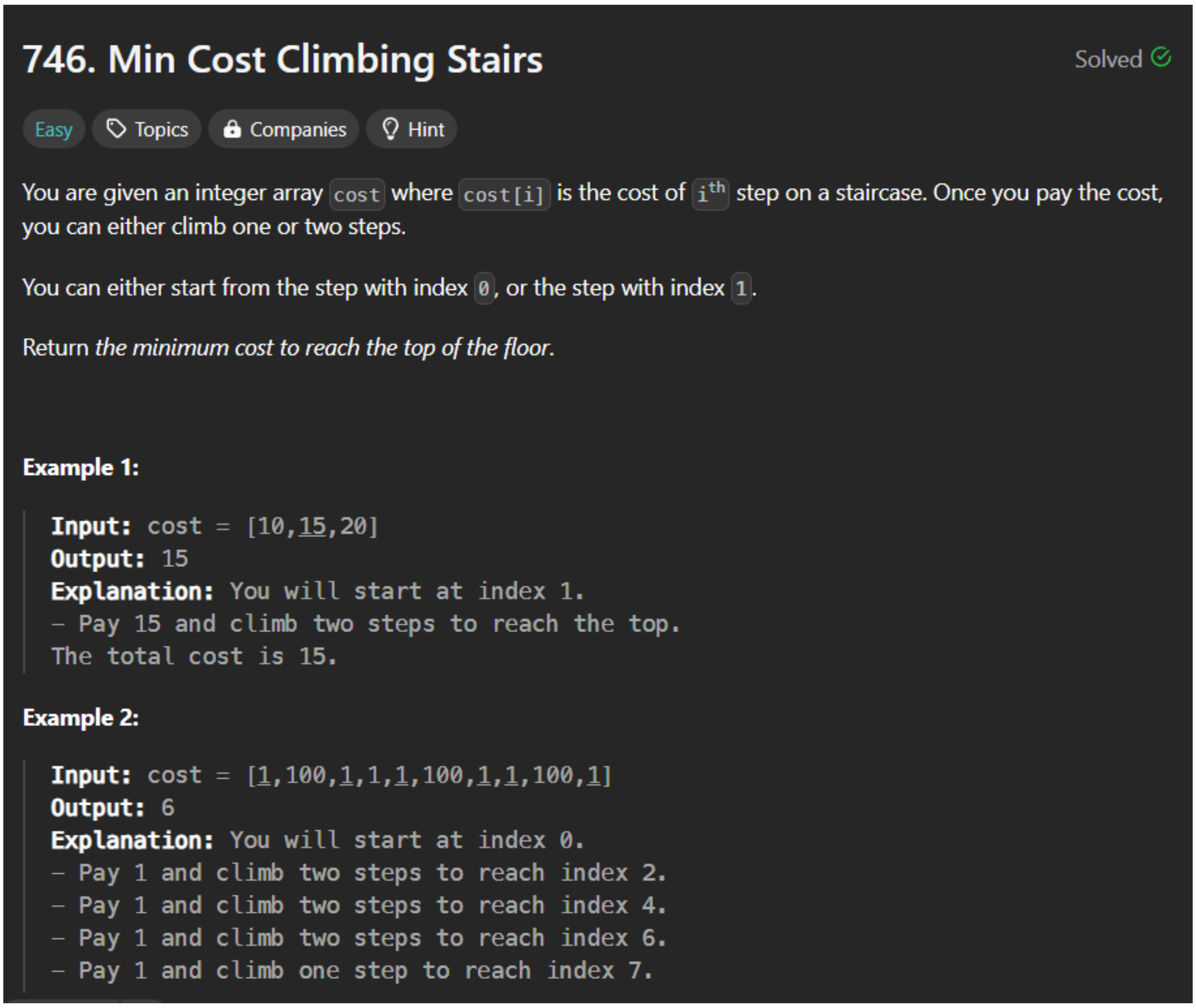


746. Min Cost Climbing Stairs

Solved

Easy · Topics · Companies · Hint

You are given an integer array `cost` where `cost[i]` is the cost of `i`th step on a staircase. Once you pay the cost, you can either climb one or two steps.

You can either start from the step with index `0`, or the step with index `1`.

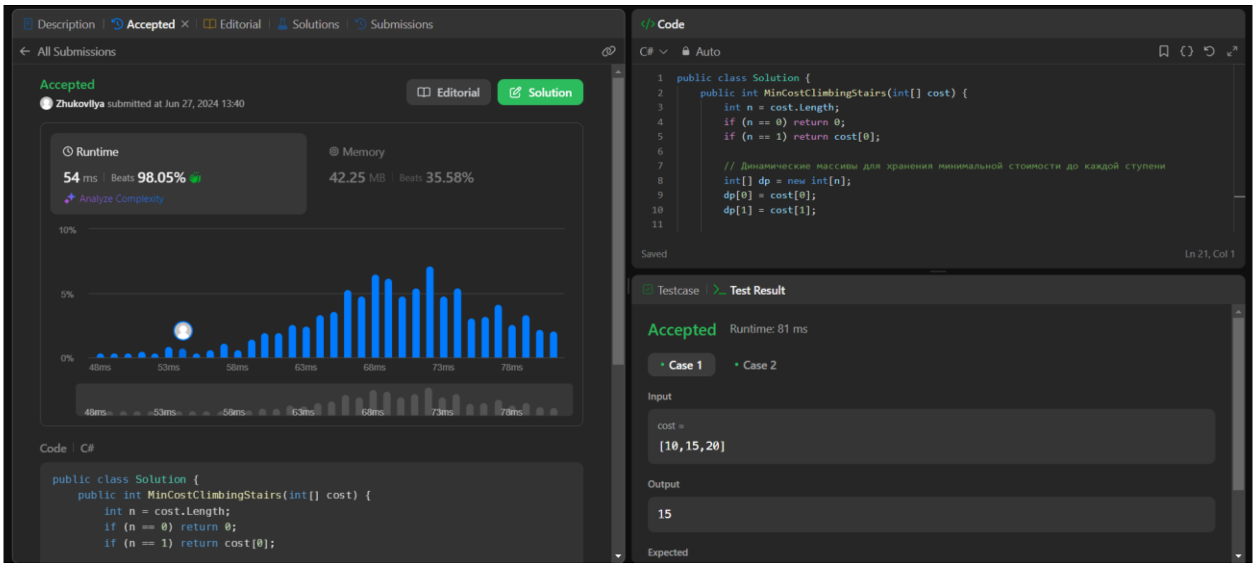
Return *the minimum cost to reach the top of the floor.*

Example 1:

```
Input: cost = [10,15,20]
Output: 15
Explanation: You will start at index 1.
- Pay 15 and climb two steps to reach the top.
The total cost is 15.
```

Example 2:

```
Input: cost = [1,100,1,1,1,100,1,1,100,1]
Output: 6
Explanation: You will start at index 0.
- Pay 1 and climb two steps to reach index 2.
- Pay 1 and climb two steps to reach index 4.
- Pay 1 and climb two steps to reach index 6.
- Pay 1 and climb one step to reach index 7.
```



Accepted

Zhukovllya submitted at Jun 27, 2024 13:40

Runtime: 54 ms | Beats 98.05%

Memory: 42.25 MB | Beats 35.58%

Code | C#

```
public class Solution {
    public int MinCostClimbingStairs(int[] cost) {
        int n = cost.Length;
        if (n == 0) return 0;
        if (n == 1) return cost[0];
```

Code — C# — Auto

```
public class Solution {
    public int MinCostClimbingStairs(int[] cost) {
        int n = cost.Length;
        if (n == 0) return 0;
        if (n == 1) return cost[0];

        // Динамические массивы для хранения минимальной стоимости до каждой ступени
        int[] dp = new int[n];
        dp[0] = cost[0];
        dp[1] = cost[1];

```

Saved — Ln 21, Col 1

Testcase | Test Result

Accepted   Runtime: 81 ms

Case 1 · Case 2

Input

```
cost =
[10,15,20]
```

Output

```
15
```

Expected

Код:

```
using System;

public class Solution
{
    public int MinCostClimbingStairs(int[] cost)
    {
        int n = cost.Length;
```

```
            if (n == 0) return 0;
            if (n == 1) return cost[0];

            // Динамические массивы для хранения минимальной стоимости до каждой ступени
            int[] dp = new int[n];
            dp[0] = cost[0];
            dp[1] = cost[1];

            // Рассчитываем минимальную стоимость для каждой ступени начиная с третьей
            for (int i = 2; i < n; i++)
            {
                dp[i] = cost[i] + Math.Min(dp[i - 1], dp[i - 2]);
            }

            // Минимальная стоимость для достижения вершины
            return Math.Min(dp[n - 1], dp[n - 2]);
        }
    }
```