Min Cost Climbing Stairs

On a staircase, the `i`-th step has some non-negative cost `cost[i]` assigned (0 indexed).

Once you pay the cost, you can either climb one or two steps. You need to find minimum cost to reach the top of the floor, and you can either start from the step with index 0, or the step with index 1.

Example 1:

```
Input: cost = [10, 15, 20]
Output: 15
Explanation: Cheapest is start on cost[1], pay that cost and go to the top.
```

Example 2:

```
Input: cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]
Output: 6
Explanation: Cheapest is start on cost[0], and only step on 1s, skipping cost[3].
```

Note:

1. `cost` will have a length in the range `[2, 1000]`.
2. Every `cost[i]` will be an integer in the range `[0, 999]`.

Solution 1

Minimum cost to reach step i is the min of costs to reach it from [i-1] and [i-2].

```python
def minCostClimbingStairs(self, cost):
    n = len(cost)
    if n == 0 or n == 1:
        return 0
    min_cost0, min_cost1 = cost[0], cost[1]
    for i in range(2, n):
        min_cost0, min_cost1 = min_cost1, min(min_cost0, min_cost1) + cost[i]

    return min(min_cost0, min_cost1)
```

written by candy.tgz original link here

Solution 2

```
class Solution {
    public int minCostClimbingStairs(int[] cost) {
        int [] mc = new int[cost.length + 1];
        mc[0] = cost[0];
        mc[1] = cost[1];

        for(int i = 2; i <= cost.length; i++){
            int costV = (i==cost.length)?0:cost[i];
            mc[i] = Math.min(mc[i-1] + costV, mc[i-2] + costV);
        }
        return mc[cost.length];
    }
}
```

written by roshan108 original link here

Solution 3

Let `dp[i]` be the minimum cost to reach the i-th stair.

Base cases:

`dp[0]=cost[0]`
`dp[1]=cost[1]`

DP formula:

`dp[i]=cost[i]+min(dp[i-1],dp[i-2])`

Note: the top floor `n` can be reached from either 1 or 2 stairs away, return the minimum.

```
class Solution {
public:
    int minCostClimbingStairs(vector<int>& cost) {
        int n=(int)cost.size();
        vector<int> dp(n);
        dp[0]=cost[0];
        dp[1]=cost[1];
        for (int i=2; i<n; ++i)
            dp[i]=cost[i]+min(dp[i-2],dp[i-1]);
        return min(dp[n-2],dp[n-1]);
    }
};
```

written by claytonjwong original link here