# 题目

数组的每个下标作为一个阶梯，第 i 个阶梯对应着一个非负数的体力花费值 cost[i]（下标从 0 开始）。

每当爬上一个阶梯都要花费对应的体力值，一旦支付了相应的体力值，就可以选择向上爬一个阶梯或者爬两个阶梯。

请找出达到楼层顶部的最低花费。在开始时，你可以选择从下标为 0 或 1 的元素作为初始阶梯。

示例 1：

输入：cost = [10, 15, 20]

输出：15

解释：最低花费是从 cost[1] 开始，然后走两步即可到阶梯顶，一共花费 15 。

 示例 2：

输入：cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]

输出：6

解释：最低花费方式是从 cost[0] 开始，逐个经过那些 1 ，跳过 cost[3] ，一共花费 6 。

提示：

2 <= cost.length <= 1000

0 <= cost[i] <= 999

注意：本题与主站 746 题相同： https://leetcode-cn.com/problems/min-cost-climbing-stairs/

# 分析

**思路：**

**代码：**

class Solution {

public:

int minCostClimbingStairs(vector<int>& cost) {

int n = cost.size();

vector<int> dp(n);

//2 <= cost.length <= 1000

dp[0] = cost[0];

dp[1] = cost[1];

for (int i = 2; i < n; ++i) {

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

}

// 注意从代价算下面的楼梯，到达本楼梯之后不用开销

return min(dp[n - 1], dp[n - 2]);

}

};

朴素的实现，需要O(n)的时间以及空间复杂度。

注意到dp[i] 只和 dp[i - 2], dp[i - 1]有关，也就是我们只用2个变量就行了.  
注意我注释里一一对应方便理解。

int minCostClimbingStairs(vector<int>& cost) {

int n = cost.size();

//vector<int> dp(n);

int dp\_0 = cost[0];

int dp\_1 = cost[1];

for (int i = 2; i < n ; ++i) {

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

int tmp = dp\_1;

dp\_1 = min(dp\_1, dp\_0) + cost[i];

dp\_0 = tmp;

}

return min(dp\_0, dp\_1);

}

另一种写法：

class Solution {

public:

int minCostClimbingStairs(vector<int>& cost) {

const int len = cost.size();

vector<int> dp(len+1);

dp[0] = dp[1] = 0;

for(int i=2 ; i<=len ; ++i ){

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

//判断从哪个阶梯来更省力

}

return dp[len];

}

};