题目难度：EASY

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

每当你爬上一个阶梯你都要花费对应的体力花费值，然后你可以选择继续爬一个阶梯或者爬两个阶梯。

您需要找到达到楼层顶部的最低花费。

在开始时，你可以选择从索引为 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。

注意：

cost 的长度将会在 [2, 1000]。

每一个 cost[i] 将会是一个Integer类型，范围为 [0, 999]。

思路：DP

DP表达式：cost[i] = min(cost[i] + cost[i - 1], cost[i] + cost[i-2])

先把数组后面加两个零

然后从第三个数组值（前两个位置消耗必为两者最小值）开始取最小可能值

最后返回最后两项的最小值（都能到阶梯顶）即可

时间击败63%（时间一般）

空间击败14%（空间炸了）原地算法你告诉我空间炸了......

上代码：

class Solution {

public:

int minCostClimbingStairs(vector<int>& cost) {

cost.push\_back(0);

cost.push\_back(0);

for(int i = 2; i < cost.size(); i++) cost[i] = min(cost[i] + cost[i - 1], cost[i] + cost[i-2]);

return min(cost[cost.size() - 1], cost[cost.size() - 2]);

}

};