746 使用最小花费爬楼梯

Label：动态规划

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

输入：cost = [10, 15, 20]
输出：15
输入：cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]
输出：6

- 动态规划

```
class Solution {
    public int minCostClimbingStairs(int[] cost) {
        if (cost.length == 2) return Math.min(cost[0], cost[1]);
        int [] reCost = new int[cost.length + 1];  // 每个点最优结果
        reCost[0] = cost[0];  // 初始化
        reCost[1] = cost[1];
        for (int i = 2; i < reCost.length; i++) {
            if (i == reCost.length -1) {
                reCost[i] = Math.min(reCost[i-2], reCost[i-1]); // 最后一次
            } else{
                reCost[i] = Math.min(reCost[i-2], reCost[i-1]) + cost[i];
            }
        }
        return reCost[reCost.length - 1];
    }
}
```

- 滚动数组 占用内存小

```
class Solution {
    public int minCostClimbingStairs(int[] cost) {
        int n = cost.length;
        int prev = 0, curr = 0;
        for (int i = 2; i <= n; i++) { //  // 有点类似于 滚动数组，只需用三个指针进行循环
            int next = Math.min(curr + cost[i - 1], prev + cost[i - 2]);
            prev = curr;
            curr = next;
        }
        return curr;
    }
}
```