剑指63 股票的最大利润

一开始觉得思考方式很像 offer 42 连续子数组最大和（需要选择起点和终点的）
我的思路：求出在第1，2，3...n天卖出时，各自的最佳方案，然后在n种情况中其中选择利润最大的。要求出如第x天卖出的最佳方案，则只要知道前面x天价格最小值就行。

但是状态转移还是很奇怪，没写出来，而题目却做对了。（24分钟耗时）

我后来写的状态转移方程（可能不算状态转移）：
f(n)代表当在第n日卖出时，最佳方案的起点index。
f(n) = f(n-1) 当value[n] >= minvalue
f(n) = n 当value[n] < minvalue
应该把f(n)改为第n日卖出时的最大利润。

k神的思路！！！！：
定义dp[i]为前i日的最大利益（这才是正常的动态规划状态定义）
转移方程：前i日的最大利益= 前i-1日的最大利益 和 第i日卖出的最大利益 中的最大值
转移方程：dp[i] = max(dp[i - 1], prices[i] - min(prices[0 : i]))

```
// 我解法：好像有点偏离了动态规划思路。

class Solution {
public:
    int maxProfit(vector<int>& prices) {
        if (prices.empty()) return 0;
        int maxprofit = 0;
        int minvalue = prices[0];
        for(int i = 1; i < prices.size(); i++) {
            if(prices[i] < minvalue) {
                minvalue = prices[i];
            }
            else {
                int curprofit = prices[i] - minvalue;
                maxprofit = curprofit > maxprofit ? curprofit : maxprofit;
            }
        }
        return maxprofit;
    }
};
```

```
// k神解法翻译为c++
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int profit = 0;
        int cost = INT_MAX; // 负责保存前面的最小买入值,用INT_MAX避免了需要提前赋初始值为prices[0],所以避免了判断prices的是否为空的操作。
```

```
        for (int price:prices) {
            cost = min(cost, price);
            profit = max(profit, price - cost);
        }
        return profit;
    }
};
```

后话(比喻也不太贴近，忽略)：
假如你有且仅有发送dmail改变过去让自己在过去某一天买入/卖出股票的能力。那么你现在应该如何操作才能让自己现在最富有?

股价：7,1,5,3,6,4
比如在第三天，股价历史7，1，5，你发dmai告诉第二天的自己买股票，而你自己今天全部卖出，有利润4。
而到了第四天，股价为3，你会算一下今天卖股票的最大利润，没有之前的高，所以就什么也不做，能获得的利润依然是之前的4.
到了第五天，股价为6，你算了一下，今天卖股票的最大利润是最高的，所以发送dmai告诉第四的天的自己不要卖股票，而你自己今天卖掉，能获得利润5

第四天，股价为3，你知道最大利润依然不变
第五天，股价为