122 买卖股票的最佳时机II

Lable：动态规划、贪心
给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。

设计一个算法来计算你所能获取的最大利润。你可以尽可能地完成更多的交易（多次买卖一支股票）。

注意：你不能同时参与多笔交易（你必须在再次购买前出售掉之前的股票）。

示例1：
    输入：[7,1,5,3,6,4]
    输出：7
    解释：在第 2 天（股票价格 = 1）的时候买入，在第 3 天（股票价格 = 5）的时候卖出，这笔交易所能获得利润 = 5-1 = 4 。
     随后，在第 4 天（股票价格 = 3）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，这笔交易所能获得利润 = 6-3 = 3 。

示例2：
    输入：[1,2,3,4,5]
    输出：4
    解释：在第 1 天（股票价格 = 1）的时候买入，在第 5 天 （股票价格 = 5）的时候卖出，这笔交易所能获得利润 = 5-1 = 4 。
     注意你不能在第 1 天和第 2 天接连购买股票，之后再将它们卖出。
     因为这样属于同时参与了多笔交易，你必须在再次购买前出售掉之前的股票。

示例 3：
    输入：[7,6,4,3,1]
    输出：0
    解释：在这种情况下，没有交易完成，所以最大利润为 0。

- 贪心

```java
// “贪心算法”在每一步总是做出在当前看来最好的选择（局部最优选择）。只加正数。
// 计算的过程并不是真正交易的过程，因为不能单天又买又卖，但是单天又买又卖，可以理解为不做操作，但是却累计收益，所以可以用贪心算法计算题目要求的最大利润
class Solution {

    public int maxProfit(int[] prices) {
        int allProfit = 0;

        for (int i = 0;i < prices.length - 1; i++){
            if (prices[i+1] >= prices[i]) {
                allProfit += (prices[i+1]-prices[i]);
            }
        }
        return allProfit;
    }
}
```

- 动态规划

```java
// 用贪心算法解决的问题，一般情况下都可以用动态规划
class Solution {
    public int maxProfit(int[] prices) {
        int len = prices.length;
        if (len < 2) return 0;

        // 0：持有现金
        // 1：持有股票
        // 状态转移：0 → 1 → 0 → 1 → 0 → 1 → 0
        int[][] dp = new int[len][2];

        dp[0][0] = 0;
        dp[0][1] = -prices[0];

        for (int i = 1; i < len; i++) {
            // 这两行调换顺序也是可以的
            dp[i][0] = Math.max(dp[i - 1][0], dp[i - 1][1] + prices[i]);
            dp[i][1] = Math.max(dp[i - 1][1], dp[i - 1][0] - prices[i]);
        }
        return dp[len - 1][0];
    }
}

// 无状态
public class Solution {
    public int maxProfit(int[] prices) {
        int len = prices.length;
        if (len < 2) return 0;

        // cash：持有现金
        // hold：持有股票
        int cash = 0;
        int hold = -prices[0];

        int preCash = cash;
        int preHold = hold;
        for (int i = 1; i < len; i++) {
            cash = Math.max(preCash, preHold + prices[i]);
            hold = Math.max(preHold, preCash - prices[i]);

            preCash = cash;
            preHold = hold;
        }
        return cash;
    }
}
```