309 最佳买卖股票时机含冷冻期

Label：动态规划

给定一个整数数组，其中第 i 个元素代表了第 i 天的股票价格 。
设计一个算法计算出最大利润。在满足以下约束条件下，你可以尽可能地完成更多的交易（多次买卖一支股票）：

> 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）。
> 卖出股票后，你无法在第二天买入股票 (即冷冻期为 1 天)。
>
> 输入: [1,2,3,0,2]
> 输出: 3
> 解释: 对应的交易状态为: [买入, 卖出, 冷冻期, 买入, 卖出]

- 动态规划

```
class Solution {
    public int maxProfit(int[] prices) {
        if (prices.length == 0) return 0;

        int n = prices.length;
        // dp[i][0]: 第i天结束后不持股
        // dp[i][1]: 第i天结束后持有股票
        // dp[i][2]: 第i天结束后为冷冻期，也就是第i天卖出股票
        int[][] dp = new int[n][3];
        dp[0][1] = -prices[0]; // 表示第0天买了股票

        for (int i = 1; i < prices.length; i++) {  // 表示第i天结束之后要进入的状态
            //第i天结束后不持股可以从两种状态转移而来
            // 1.第i-1天不持股，第i天仍不买股票，保持不持股状态。
            // 2.冷冻期结束了，但是第i天还是不买股票。
            dp[i][0] = Math.max(dp[i-1][0], dp[i-1][2]);

            // 第i天结束后持股可从两种状态转移而来
            // 1.第i-1天不持股(包含昨天是冷冻期，冷冻期结束后转为不持股状态和昨天本身就不持股这两种情况)，第i天买股票。
            // 2.第i-1天持股，第i天不卖出，保持持股状态。
            dp[i][1] = Math.max(dp[i - 1][0] - prices[i], dp[i - 1][1]);

            // 只有第i天卖出了股票，第i天结束后才处于冷冻期。
            dp[i][2] = dp[i-1][1] + prices[i];
        }

        //只有最后一天不持股或者第i-1天已经卖掉了（第i天为冷冻期），那么第i天结束后这两种情况手里是拿着钱的
        return Math.max(dp[prices.length - 1][0], dp[prices.length - 1][2]);
    }
}
```