本日学习内容：

1. 开始研究买卖股票问题，完成3，4 并总结为博客
2. 系统总结dp的学过的各种问题写入博客

本日分享内容：

题目一：

给定一个数组，它的第 `i` 个元素是一支给定的股票在第 `i` 天的价格。

设计一个算法来计算你所能获取的最大利润。你最多可以完成 **两笔** 交易。

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：prices = [3,3,5,0,0,3,1,4]
输出：6
解释：在第 4 天（股票价格 = 0）的时候买入，在第 6 天（股票价格 = 3）的时候卖出，这笔交易所能获得利润 = 3-0 = 3 。
     随后，在第 7 天（股票价格 = 1）的时候买入，在第 8 天 （股票价格 = 4）的时候卖出，这笔交易所能获得利润 = 4-1 = 3 。
```

示例 2:

```
输入：prices = [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：prices = [7,6,4,3,1]
输出：0
解释：在这个情况下，没有交易完成，所以最大利润为 0。
```

示例 4:

```
输入：prices = [1]
输出：0
```

提示：

- `1 <= prices.length <= 105`
- `0 <= prices[i] <= 104`

思路：

达到dp[i][1]状态，有两个具体操作：

- 操作一：第i天买入股票了，那么dp[i][1] = dp[i-1][0] - prices[i]
- 操作二：第i天没有操作，而是沿用前一天买入的状态，即：dp[i][1] = dp[i - 1][1]

那么dp[i][1]究竟选 dp[i-1][0] - prices[i]，还是dp[i - 1][1]呢?

一定是选最大的，所以 dp[i][1] = max(dp[i-1][0] - prices[i], dp[i - 1][1]);

同理dp[i][2]也有两个操作：

- 操作一：第i天卖出股票了，那么dp[i][2] = dp[i - 1][1] + prices[i]
- 操作二：第i天没有操作，沿用前一天卖出股票的状态，即：dp[i][2] = dp[i - 1][2]

所以dp[i][2] = max(dp[i - 1][1] + prices[i], dp[i - 1][2])

同理可推出剩下状态部分：

dp[i][3] = max(dp[i - 1][3], dp[i - 1][2] - prices[i]);

dp[i][4] = max(dp[i - 1][4], dp[i - 1][3] + prices[i]);

代码：

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        vector<vector<int>> dp(n, vector<int>(5, 0));

        dp[0][0] = 0;
        dp[0][1] = -prices[0];
        dp[0][2] = 0;
        dp[0][3] = -prices[0];
        dp[0][4] = 0;

        for (int i = 1; i < n; i++) {
            dp[i][0] = dp[i - 1][0];
            dp[i][1] = max(dp[i - 1][1], dp[i - 1][0] - prices[i]);
            dp[i][2] = max(dp[i - 1][2], dp[i - 1][1] + prices[i]);
            dp[i][3] = max(dp[i - 1][3], dp[i - 1][2] - prices[i]);
            dp[i][4] = max(dp[i - 1][4], dp[i - 1][3] + prices[i]);
        }

        return max({dp[n - 1][0], dp[n - 1][2], dp[n - 1][4]});
    }
};
```

题目二：

给你一个整数数组 `prices` 和一个整数 `k` ，其中 `prices[i]` 是某支给定的股票在第 `i` 天的价格。

设计一个算法来计算你所能获取的最大利润。你最多可以完成 `k` 笔交易。也就是说，你最多可以买 `k` 次，卖 `k` 次。

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：k = 2, prices = [2,4,1]
输出：2
解释：在第 1 天 (股票价格 = 2) 的时候买入，在第 2 天 (股票价格 = 4) 的时候卖出，这笔交易所能获得利润 = 4-2 = 2 。
```

示例 2：

```
输入：k = 2, prices = [3,2,6,5,0,3]
输出：7
解释：在第 2 天（股票价格 = 2）的时候买入，在第 3 天（股票价格 = 6）的时候卖出，这笔交易所能获得利润 = 6-2 = 4 。
     随后，在第 5 天（股票价格 = 0）的时候买入，在第 6 天（股票价格 = 3）的时候卖出，这笔交易所能获得利润 = 3-0 = 3 。
```

思路：

我们使用一个三维数组来表示状态转移，

```
dp[i][j][0] 表示第 i 天，完成了 j 次交易后，**手里没有股票** 的最大收益
dp[i][j][1] 表示第 i 天，完成了 j 次交易后，**手里持有股票** 的最大收益
```

i 表示第i天，j表示交易次数，第三维0/1表示是否持股

那么好，接下来，状态转移方程：

```
dp[i][j][0] = max(dp[i-1][j][0], dp[i-1][j][1] + prices[i])      // 卖出 or 不动
dp[i][j][1] = max(dp[i-1][j][1], dp[i-1][j-1][0] - prices[i])    // 买入 or 不动

//注意，买入后交易数j要+1，所以买的时候从j - 1转移过来
```

```
dp[0][0][0] = 0; // 第 0 天不买不卖
dp[0][0][1] = -prices[0]; // 第 0 天买了，交易数 0，持股
其余 dp[0][j][1] = -∞（不合法）
```

代码：

```cpp
int maxProfit(int k, vector<int>& prices) {
    int n = prices.size();
    if (n == 0) {
        return 0;
    }

    // 如果交易次数远大于天数，退化为无限交易（贪心）
    if (k >= n / 2) {
        int profit = 0;
        for (int i = 1; i < n; ++i) {
            if (prices[i] > prices[i - 1]) {
                profit += prices[i] - prices[i - 1];
            }
        }
        return profit;
    }

    vector<vector<vector<int>>> dp(n, vector<vector<int>>(k + 1, vector<int>(2, 0)));

    for (int j = 0; j <= k; ++j) {
        dp[0][j][0] = 0;
        dp[0][j][1] = -prices[0]; // 如果允许 j 次交易，那么一开始就买了
    }

    for (int i = 1; i < n; ++i) {
        for (int j = 1; j <= k; ++j) {
            dp[i][j][0] = max(dp[i - 1][j][0], dp[i - 1][j][1] + prices[i]);
            dp[i][j][1] = max(dp[i - 1][j][1], dp[i - 1][j - 1][0] - prices[i]);
        }
    }

    int maxProfit = 0;
    for (int j = 0; j <= k; ++j) {
        maxProfit = max(maxProfit, dp[n - 1][j][0]);
    }
    return maxProfit;
}
```

本日遇到的问题

1. 对状态机dp的压缩还是不太熟练

明日学习内容

1. 测试管理系统的bug并修复，优化一些功能的逻辑
2. 完成股票问题博客