本日学习内容：

1. 开始研究买卖股票问题，完成1，2
2. 完成学生管理系统的所有基础功能

本日分享内容：

题目一：

给定一个数组 `prices` ，它的第 `i` 个元素 `prices[i]` 表示一支给定股票第 `i` 天的价格。

你只能选择 **某一天** 买入这只股票，并选择在 **未来的某一个不同的日子** 卖出该股票。设计一个算法来计算你所能获取的最大利润。

返回你可以从这笔交易中获取的最大利润。如果你不能获取任何利润，返回 `0` 。

示例 1：

```
输入：[7,1,5,3,6,4]
输出：5
解释：在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。
     注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格；同时，你不能在买入前卖出股票。
```

示例 2：

```
输入：prices = [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。
```

提示：

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

思路：

dp[i] [0]：第 i 天结束后不持有股票的最大利润

dp[i] [1]：第 i 天持有股票的最大利润（买过了）

dp[i] [0] = max(dp[i-1] [0], dp[i-1] [1] + prices[i]);
dp[i] [1] = max(dp[i-1] [1], -prices[i]); // 只能买一次，直接 -prices[i]

代码：

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        vector<vector<int>> dp(n, vector<int>(2));
        dp[0][0] = 0;            // 第一天没买
        dp[0][1] = -prices[0];   // 第一天买入

        for (int i = 1; i < n; ++i) {
            dp[i][0] = max(dp[i - 1][0], dp[i - 1][1] + prices[i]); // 卖出或不动
            dp[i][1] = max(dp[i - 1][1], -prices[i]);              // 买入或不动
        }

        return dp[n - 1][0]; // 最后一天不能持股才能获得利润
    }
};
```

题目二：

给你一个整数数组 `prices` ，其中 `prices[i]` 表示某支股票第 `i` 天的价格。

在每一天，你可以决定是否购买和/或出售股票。你在任何时候 **最多** 只能持有 **一股** 股票。你也可以先购买，然后在 **同一天** 出售。

返回 *你能获得的* ***最大*** *利润* 。

示例 1：

```
输入：prices = [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。
最大总利润为 4 + 3 = 7 。
```

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。
最大总利润为 4 。
```

示例 3：

```
输入：prices = [7,6,4,3,1]
输出：0
解释：在这种情况下，交易无法获得正利润，所以不参与交易可以获得最大利润，最大利润为 0。
```

提示：

- `1 <= prices.length <= 3 * 104`
- `0 <= prices[i] <= 104`

思路：

dp[i] [0]: 第 i 天，**手上不持有股票**时的最大利润；

dp[i] [1]： 第 i 天，**手上持有股票**时的最大利润。

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

代码：

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        vector<vector<int>> dp(n, vector<int>(2));
        dp[0][0] = 0;               // 第一天不持股
        dp[0][1] = -prices[0];      // 第一天买入

        for (int i = 1; i < n; ++i) {
            // 第 i 天不持股：可能是昨天也不持股，或今天卖了
            dp[i][0] = max(dp[i - 1][0], dp[i - 1][1] + prices[i]);
            // 第 i 天持股：可能是昨天也持股，或今天新买了
            dp[i][1] = max(dp[i - 1][1], dp[i - 1][0] - prices[i]);
        }

        return dp[n - 1][0];  // 最后一天不持股才能拿到利润
    }
};
```

本日遇到的问题

1. 管理系统代码量较多，写到后面有些疲劳
2. 刚接触股票问题思路较为单一，没有想到可以多种方法完成

明日学习内容

1. 自行测试管理系统的功能，并添加一些功能如验证码找回密码
2. 继续完成股票问题，并写该专题的博客