# 714-买卖股票的最佳时机含手续费

## 题述

714. 买卖股票的最佳时机含手续费

难度 中等 666

给定一个整数数组 `prices`，其中 `prices[i]`表示第 `i` 天的股票价格；整数 `fee` 代表了交易股票的手续费用。

你可以无限次地完成交易，但是你每笔交易都需要付手续费。如果你已经购买了一个股票，在卖出它之前你就不能再继续购买股票了。

返回获得利润的最大值。

**注意：**这里的一笔交易指买入持有并卖出股票的整个过程，每笔交易你只需要为支付一次手续费。

**示例 1：**

```
输入：prices = [1, 3, 2, 8, 4, 9], fee = 2
输出：8
解释：能够达到的最大利润:
在此处买入 prices[0] = 1
在此处卖出 prices[3] = 8
在此处买入 prices[4] = 4
在此处卖出 prices[5] = 9
总利润: ((8 - 1) - 2) + ((9 - 4) - 2) = 8
```

**示例 2：**

```
输入：prices = [1,3,7,5,10,3], fee = 3
输出：6
```

## 思路

使用贪心策略，就是最低值买，最高值（如果算上手续费还盈利）就卖。

此时无非就是要找到两个点，买入日期，和卖出日期。

- 买入日期：其实很好想，遇到更低点就记录一下。
- 卖出日期：这个就不好算了，但也没有必要算出准确的卖出日期，只要当前价格大于（最低价格+手续费），就可以收获利润，至于准确的卖出日期，就是连续收获利润区间里的最后一天（并不需要计算是具体哪一天）。

所以我们在做收获利润操作的时候其实有三种情况：

- 情况一：收获利润的这一天并不是收获利润区间里的最后一天（不是真正的卖出，相当于持有股票），所以后面要继续收获利润。

- 情况二：前一天是收获利润区间里的最后一天（相当于真正的卖出了），今天要重新记录最小价格了。
- 情况三：不作操作，保持原有状态（买入，卖出，不买不卖）

# 题解

## C++

```
class Solution {
public:
    int maxProfit(vector<int>& prices, int fee)
    {
        //最低值买，最高值（如果算上手续费还盈利）就卖
        int result = 0;
        int minPrice = prices[0];   //记录最低价格
        for(int i = 1;i < prices.size(); i++)
        {
            //情况二：相当于买入
            if(prices[i] < minPrice) minPrice = prices[i];

            //情况三：保持原有状态（因为此时买则不便宜，卖则亏本）
            if(prices[i] >= minPrice && prices[i] <= minPrice + fee)
            {
                continue;
            }

            //计算利润，可能需多次计算，最后一次计算利润才是真正意义的卖出
            if(prices[i] > minPrice + fee)
            {
                result += prices[i] - minPrice -fee;
                minPrice = prices[i] -fee;  //情况一 关键步骤
            }
        }
        return result;
    }
};
```


## Python

```python
class Solution:
    def maxProfit(self, prices: List[int], fee: int) -> int:
        result = 0
        minPrice = prices[0]
        for i in range(1,len(prices)):
            if prices[i] < minPrice:
                minPrice = prices[i]
            elif prices[i] >= minPrice and prices[i] <= minPrice + fee:
                continue
            else:
                result += prices[i] - minPrice - fee
                minPrice = prices[i] - fee
        return result
```


| 提交结果 | 执行用时 | 内存消耗 | 语言 | 提交时间 | 备 |
|---|---|---|---|---|---|
| 通过 | 144 ms | 20.8 MB | Python3 | 2022/03/29 09:41 | |
| 通过 | 88 ms | 53.6 MB | C++ | 2022/03/29 09:37 | |

# 思考

这道题使用动态规划更好其实，后面进行补充