123. Best Time to Buy and Sell Stock III

题目描述：https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/

> 给定n天的股票价格，求只经过2次交易可取的的最大收益。要求开始一次交易前必须结束前一次。

解题思路：

动态规划问题，将其扩展为k次交易，则f[i][j]代表共i次交易到第j天的最大收益。 f[i][j] = max(f[i][j-1], max 0=<k<j(f[i-1][k]-prices[k]+prices[j])

代码1：

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        if(prices.size() <= 1) return 0;
        int k = 2, len = prices.size(), maxProf = 0;
        vector<vector<int> > f(k+1, vector<int>(len, 0));
        for(int i = 1; i <= k; i++) {
            int tmpMax = f[i-1][0]-prices[0];
            for(int j = 1; j < len; j++) {
                f[i][j] = max(f[i][j-1], tmpMax+prices[j]);
                tmpMax = max(tmpMax, f[i-1][j]-prices[j]);
                maxProf = max(maxProf, f[i][j]);
            }
        }
        return maxProf;
    }
};
```

代码（超时）：

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        if(prices.size() <= 1) return 0;
        int k = 2, len = prices.size(), maxProf = 0;
        vector<vector<int> > f(k+1, vector<int>(len, 0));
        for(int i = 1; i <= k; i++) {
            int tmpMax = INT_MIN;
            for(int j = 1; j < len; j++) {
                for(int l = 0; l < j; l++) {
                    tmpMax = max(tmpMax, f[i-1][l]-prices[l]);
                }
                f[i][j] = max(f[i][j-1], tmpMax+prices[j]);
                maxProf = max(maxProf, f[i][j]);
            }
        }
        return maxProf;
    }
};
```