1. Best Time to Buy and Sell Stock III

Say you have an array for which the *i*th element is the price of a given stock on day *i*.

Design an algorithm to find the maximum profit. You may complete at most *two* transactions.

**Note:** You may not engage in multiple transactions at the same time (i.e., you must sell the stock before you buy again).

**Example 1:**

Input: [3,3,5,0,0,3,1,4]  
Output: 6  
Explanation: Buy on day 4 (price = 0) and sell on day 6 (price = 3), profit = 3-0 = 3.  
 Then buy on day 7 (price = 1) and sell on day 8 (price = 4), profit = 4-1 = 3.

**Example 2:**

Input: [1,2,3,4,5]  
Output: 4  
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.  
 Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are  
 engaging multiple transactions at the same time. You must sell before buying again.

**Example 3:**

Input: [7,6,4,3,1]  
Output: 0  
Explanation: In this case, no transaction is done, i.e. max profit = 0.

**Solution**

Approach1 以 i 为分界，左边是第一次交易能够获得的最大利润，右边是第二次，最后两边加起来取最大

Note：不能每次都计算一次，用数组存储能够获得的最大利润，否则会超时

class Solution {  
public:  
 int maxProfit(vector<int>& prices) {  
 int ans = 0;  
 int n = prices.size();  
 if(n == 0)return ans;  
 int left[n] = {0}, right[n] = {0};  
 int min\_price = prices[0];  
 for(int i = 1; i < n; ++i){  
 min\_price = min(min\_price, prices[i]);  
 left[i] = max(left[i-1], prices[i] - min\_price);  
 }  
 int max\_price = prices[n-1];  
 for(int i = n-2; i >= 0; --i){  
 max\_price = max(max\_price, prices[i]);  
 right[i] = max(right[i+1], max\_price - prices[i]);  
 }  
 for(int i = 0; i < n; ++i){  
 ans = max(ans, left[i] + right[i]);  
 }  
 return ans;  
 }  
};

Appraoch 2 每次取最值时针对全局的利润，设置4个变量：b1, s1, b2, s2

class Solution {  
public:  
 int maxProfit(vector<int>& prices) {  
 int b1 = INT\_MIN, b2 = INT\_MIN;  
 int s1 = 0, s2 = 0;  
 for(int x : prices){  
 b1=max(b1,-x); //以低价买入  
 s1=max(s1,b1+x); //以高价卖出  
 b2=max(b2,s1-x); //低价买入，即结余要最大  
 s2=max(s2,b2+x); //高价卖出  
 }  
 return s2;  
 }  
};