给定一个数组，它的第 *i* 个元素是一支给定股票第 *i* 天的价格。

如果你最多只允许完成一笔交易（即买入和卖出一支股票），设计一个算法来计算你所能获取的最大利润。

注意你不能在买入股票前卖出股票。

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。

示例 2:

输入：`[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`。

思路：贪心算法，从左到右，昨天买，今天卖能获取多少利润，用数组统计，再遍历数组，若利润为正或本身利润大于0则相加（到后面可能遇到为正的，正+正>0+正），当利润为负时，变为0，在这里，动态的统计最大利润即可。

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int n=prices.size();
        int max=0;
        int num=0;
        for(int i=0;i+1<n;++i){
            prices[i]=prices[i+1]-prices[i];
            if(num>0){
                num+=prices[i];
                max=num>max?num:max;
                if(num<0){
                    num=0;
                }
            }else if(prices[i]>0){
                num+=prices[i];
                max=num>max?num:max;
            }
        }
        return max;
    }
};
```