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。

设计一个算法来计算你所能获取的最大利润。你可以尽可能地完成更多的交易（多次买卖一支股票）。

注意：你不能同时参与多笔交易（你必须在再次购买前出售掉之前的股票）。

示例 1:

输入： [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 。

示例 2:

输入： [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 。
注意你不能在第 1 天和第 2 天接连购买股票，之后再将它们卖出。
因为这样属于同时参与了多笔交易，你必须在再次购买前出售掉之前的股票。

示例 3:

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，只要能赚钱，就买，当a<a1<a2时，买a，再买a1再买a2和买a再买a2获取的利润一样，那么可以有利润就购买，把所以的利润相加即为最大利润

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