/\*

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

If you were only permitted to complete at most one transaction (ie, buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Example 1:

Input: [7, 1, 5, 3, 6, 4]

Output: 5

max. difference = 6-1 = 5 (not 7-1 = 6, as selling price needs to be larger than buying price)

Example 2:

Input: [7, 6, 4, 3, 1]

Output: 0

In this case, no transaction is done, i.e. max profit = 0.

way-1:动态规划,遇到临时最小的，就保存（1），计算后面比它大的差，保留最大（6-1），遇到后面如果有更小的（0），就保存（0），重复上面工作即可。

\*/

class Solution {

public:

int maxProfit(vector<int>& prices)

{

//way-1

if(prices.size()==0)

return 0;

int minm=prices[0];

int maxa=0;

for(int i=1;i<prices.size();i++)

{

minm=min(minm,prices[i]);

maxa=max(maxa,prices[i]-minm);

}

return maxa;

}

};