**1.一种很好的方法，但是暂时理解不了**

The logic to solve this problem is same as "max subarray problem" using Kadane's Algorithm. Since no body has mentioned this so far, I thought it's a good thing for everybody to know.

All the straight forward solution should work, but if the interviewer twists the question slightly by giving the ***difference array of prices***, Ex: for {1, 7, 4, 11}, if he gives {0, 6, -3, 7}, you might end up being confused.

Here, the logic is to calculate the difference (maxCur += prices[i] - prices[i-1]) of the original array, and find a contiguous subarray giving maximum profit. If the difference falls below 0, reset it to zero.

public int maxProfit(int[] prices) {

int maxCur = 0, maxSoFar = 0;

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

maxCur = Math.max(0, maxCur += prices[i] - prices[i-1]);

maxSoFar = Math.max(maxCur, maxSoFar);

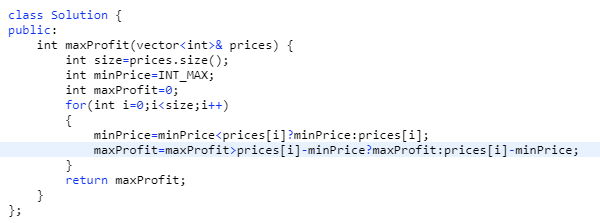
}

return maxSoFar;

}

\*maxCur = current maximum value

\*maxSoFar = maximum value found so far

2. 

一直存储着最小价格，和最大收益。不断的比对