Best Time to Buy and Sell Stock

Say you have an array for which the i^{th} 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.

Solution 1

```
int maxProfit(vector<int> &prices) {
    int maxPro = 0;
    int minPrice = INT_MAX;
    for(int i = 0; i < prices.size(); i++){
        minPrice = min(minPrice, prices[i]);
        maxPro = max(maxPro, prices[i] - minPrice);
    }
    return maxPro;
}
```

minPrice is the minimum price from day 0 to day i. And maxPro is the maximum profit we can get from day 0 to day i.

How to get maxPro? Just get the larger one between current maxPro and prices[i] - minPrice.

written by zxyperfect original link here

Solution 2

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`

written by andywhite original link here

Solution 3

1.for prices[0] .... prices[n], prices[n+1]..... if (prices[n] < prices[0]) then, the max profit is in prices[0]...prices[n], or begin from prices[n+1], otherwise, suppose prices[n+1] > prices[0], and max profit is happened between prices[n+1] , and prices1">n+k, then if we buy at day 0, and sell at day n+k, we get a bigger profit.

Base on logic above, we can have a O(1*n) solution:

```
public class Solution {
    public int maxProfit(int[] prices) {

        if (prices.length == 0)
        {
            return 0;
        }

        int max = 0, min = prices[0];
        int profit = 0;

        for (int i = 1; i < prices.length; i++)
        {
            if (prices[i] < min)
            {

                min = prices[i];
            }
            else
            {
                if (prices[i] - min > profit)
                {
                    profit = prices[i] - min;
                }

            }
        }



     return profit;

    }
}
```

written by wycl16514 original link here