Best Time to Buy and Sell Stock II

Say you have an array for which the i^{th} element is the price of a given stock on day$i$.

Design an algorithm to find the maximum profit. You may complete as many transactions as you like (ie, buy one and sell one share of the stock multiple times). However, you may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

Solution 1

```
public class Solution {
public int maxProfit(int[] prices) {
    int total = 0;
    for (int i=0; i< prices.length-1; i++) {
        if (prices[i+1]>prices[i]) total += prices[i+1]-prices[i];
    }

    return total;
}
```

A simple code like this. The designer of this question must thought of something too complicated.

written by jyan original link here

Solution 2

First we post the code here.

```
int maxProfit(vector<int> &prices) {
    int ret = 0;
    for (size_t p = 1; p < prices.size(); ++p)
      ret += max(prices[p] - prices[p - 1], 0);
    return ret;
}
```

Second, suppose the first sequence is "a <= b <= c <= d", the profit is "d - a = (b - a) + (c - b) + (d - c)" without a doubt. And suppose another one is "a <= b >= b' <= c <= d", the profit is not difficult to be figured out as "(b - a) + (d - b')". So you just target at monotone sequences.

written by tian.xia.568 original link here

Solution 3

Hi guys!

The greedy pair-wise approach mentioned in other posts is great for this problem indeed, but if we're not allowed to buy and sell stocks within the same day it can't be applied (logically, of course; the answer will be the same). Actually, the straight-forward way of finding next local minimum and next local maximum is not much more complicated, so, just for the sake of having an alternative I share the code in Java for such case.

```
public int maxProfit(int[] prices) {
    int profit = 0, i = 0;
    while (i < prices.length) {
        // find next local minimum
        while (i < prices.length-1 && prices[i+1] <= prices[i]) i++;
        int min = prices[i++]; // need increment to avoid infinite loop for "[1]"
        // find next local maximum
        while (i < prices.length-1 && prices[i+1] >= prices[i]) i++;
        profit += i < prices.length ? prices[i++] - min : 0;
    }
    return profit;
}
```

Happy coding!

written by shpolsky original link here