121 Best Time to Buy and Sell Stock

2018年4月1日 16:22

Question:

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.

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.

来自 <https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/>

假设你有一个数组，其中第 *i* 个元素是一支给定股票第 *i* 天的价格。

如果您只能完成最多一笔交易（即买入和卖出一股股票），则设计一个算法来找到最大的利润。

示例 1:

输入: [7, 1, 5, 3, 6, 4]

输出: 5

最大利润 = 6-1 = 5（不是 7-1 = 6, 因为卖出价格需要大于买入价格）

示例 2:

输入: [7, 6, 4, 3, 1]

输出: 0

在这种情况下, 没有交易完成, 即最大利润为 0。

Solution for Python3:

```python
# 暴力破解：时间复杂度：O(n^2),  空间复杂度：O(1)
class Solution1:
    def maxProfit(self, prices):
        """
        :type prices: List[int]
        :rtype: int
        """
        maxprofit = 0
        for i in range(len(prices) - 1):
            for j in range(i + 1, len(prices)):
                profit = prices[j] - prices[i]
                maxprofit = max(maxprofit, profit)

```

```
        return maxprofit

# 时间复杂度：O(n)，  空间复杂度：O(1)
class Solution2:
    def maxProfit(self, prices):
        """
        :type prices: List[int]
        :rtype: int
        """
        minprice = sys.maxsize
        maxprofit = 0
        for i in range(len(prices)):
            if prices[i] < minprice:
                minprice = prices[i]
            elif (prices[i] - minprice) > maxprofit:
                maxprofit = prices[i] - minprice
        return maxprofit

class Solution3:
    def maxProfit(self, prices):
        """
        :type prices: List[int]
        :rtype: int
        """
        maxCur, maxSoFar = 0, 0
        for i in range(1, len(prices)):
            maxCur += (prices[i] - prices[i - 1])
            maxCur = max(0, maxCur)
            maxSoFar = max(maxCur, maxSoFar)
        return maxSoFar
     # maxCur:取到当前元素时所能得到的最大价值
     # maxSoFar:遍历到当前元素时所能得到的最大价值
```

Solution for C++:

```
class Solution1 {
public:
    int maxProfit(vector<int>& prices) {
        int minprice = INT_MAX;
        int maxprofit = 0;
        for (int i = 0; i < prices.size(); i++) {
           if (prices[i] < minprice) {
               minprice = prices[i];
           } else if ((prices[i] - minprice) > maxprofit) {
               maxprofit = prices[i] - minprice;
           }
        }
        return maxprofit;
    }
};

class Solution2 {
```

```cpp
public:
    int maxProfit(vector<int>& prices) {
        int maxCur = 0, maxSoFar = 0;
        for (int i = 1; i < prices.size(); i++) {
           maxCur += prices[i] - prices[i - 1];
           maxCur = max(0, maxCur);
           maxSoFar = max(maxCur, maxSoFar);
        }
       return maxSoFar;
    }
};
```