LeetCode – Best Time to Buy and Sell Stock (Java)

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.

Java Solution

Instead of keeping track of largest element in the array, we track the maximum profit so far.

```
public int maxProfit(int[] prices) {
    if(prices==null||prices.length<=1)
        return 0;

    int min=prices[0]; // min so far
    int result=0;

    for(int i=1; i<prices.length; i++){
        result = Math.max(result, prices[i]-min);
        min = Math.min(min, prices[i]);
    }

    return result;
}
```

Related Posts:

1. **LeetCode – Best Time to Buy and Sell Stock II (Java)**
2. **LeetCode – Best Time to Buy and Sell Stock III (Java)**
3. **LeetCode – Best Time to Buy and Sell Stock IV (Java)**
4. **LeetCode – Maximum Product Subarray (Java)**

Category >> Algorithms

If you want someone to read your code, please put the code inside <pre><code> and </code></pre> tags. For example:

```
<pre><code>
String foo = "bar";
</code></pre>
```

♡ Recommend Share Sort by Best

Join the discussion…

Burhan COKCA • 3 years ago

Dynamic programming solution. O(n) time complexity, O(n) space

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

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

int[] profit = new int[prices.length];

for(int i=0; i< profit.length; i++){

profit[i] = 0;

}

int min = prices[0];

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

if(prices[i] < min) min = prices[i];

profit[i] = Math.max(profit[i-1], prices[i]-min);

}

return profit[prices.length-1];

}
```

1 ^ v • Reply • Share ›

Eduardo Carrillo ➚ Burhan COKCA • 4 months ago

Probably the cleanest solution I have seen.

^ v • Reply • Share ›

Hu Zhang • a year ago

This is good solutions, me put one here too https://www.youtube.com/edi...

^ v • Reply • Share ›

amala rangnekar • 2 years ago

Inner for loop of naive solution must start from i and not 0. When buying a stock you can only see future values to sell it :)

^ v • Reply • Share ›

Holden • 3 years ago

Thanks for the nice post. But first solution fails at `int[] price2 = {100, 90, 80, 70, 60};`
It should be:

```
public static int maxProfit5(int[] array) {
if(array == null || array.length < 2){
return 0;
}
int maxProfit = 0;
for(int i = 0; i < array.length-1; i++){
for(int j = i+1; j array[i] && maxProfit < array[j] - array[i]){
maxProfit = array[j] - array[i];
}
}
}
return maxProfit;
}
```

Reply • Share ›

Hyde Zhang ➔ Holden • 2 years ago

Your "fails" can be fixed by simply setting the initial value of profit to prices[1] - prices[0] to get the maxProfilt / smallest lost

Reply • Share ›

Xiang Li • 3 years ago

See my solution:

public int maxProfit(int[] p) {

if(p == null || p.length<=1) return 0;

int len = p.length;

int max = p[0];

int min = p[0];

int profit = 0;

for(int i=1; i max) {

max = p[i];

}

if(p[i] < min) {

see more

Reply • Share ›

Albert • 3 years ago

The efficient approach wont' calculate correctly if the fist element in the array has the highest value

Reply • Share ›

Hyde Zhang ➚ Albert • 2 years ago

Not the first element has the highest value but elements in the array are in descending order, and this can be solved by setting the initial value of profit to prices[1] - prices[0] to get the maxProfilt / smallest lost

Reply • Share ›

Internet Hero ➚ Albert • 3 years ago

But since it is a time series, will you buy when high and sell when low?

Reply • Share ›