Best Time to Buy and Sell Stock I

You are given an array `prices` where `prices[i]` is the price of a given stock on the `i`th day.
You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.
Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return `0` .

Examples

Example 1:

`Input:` prices = [7,1,5,3,6,4]

`Output:` 5

`Explanation:` Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6 – 1 = 5.

Example 2:

`Input:` prices = [7,6,4,3,1]

`Output:` 0

`Explanation:` In this case, no transactions are done and the max profit = 0.

Constraints

1 <= prices.length <= 10^5
0 <= prices[i] <= 10^4

Approach 1 (Brute Force)

Initialize `maxProfit = 0` .

Use two nested loops:

Outer loop picks a day `i` to buy the stock.
Inner loop picks a day `j > i` to sell the stock.
For every pair `(i, j)` , calculate the profit: `prices[j] - prices[i]` .
If this profit is greater than `maxProfit` , update `maxProfit` .

Return `maxProfit` after all iterations.

Dry Run

```
Input: prices = [7, 1, 5, 3, 6, 4]

i = 0, prices[i] = 7
   j = 1 → 1 - 7 = -6 → maxProfit = 0
   j = 2 → 5 - 7 = -2 → maxProfit = 0
   j = 3 → 3 - 7 = -4 → maxProfit = 0
   j = 4 → 6 - 7 = -1 → maxProfit = 0
   j = 5 → 4 - 7 = -3 → maxProfit = 0

i = 1, prices[i] = 1
   j = 2 → 5 - 1 = 4 → maxProfit = 4
   j = 3 → 3 - 1 = 2 → maxProfit = 4
   j = 4 → 6 - 1 = 5 → maxProfit = 5
   j = 5 → 4 - 1 = 3 → maxProfit = 5

i = 2, prices[i] = 5
   j = 3 → 3 - 5 = -2 → maxProfit = 5
   j = 4 → 6 - 5 = 1 → maxProfit = 5
   j = 5 → 4 - 5 = -1 → maxProfit = 5

... and so on.

Final maxProfit = 5 (buy at 1, sell at 6)
```

Time and Space Complexity

Time Complexity: $O(n^2)$
Two nested loops. For every element i, check all j > i. Total comparisons = n(n-1)/2 → $O(n^2)$
Space Complexity: O(1)
No extra data structures used. Only uses a variable `maxProfit` .

JavaScript | C++ | C | Java | Python | C#

```javascript
var maxProfit = function(prices) {
    let maxProfit = 0;
    for (let i = 0; i < prices.length; i++) {
```

```
        for (let j = i + 1; j < prices.length; j++) {
            if ((prices[j] - prices[i]) > maxProfit) {
                maxProfit = prices[j] - prices[i];
            }
        }
    }
    return maxProfit;
};
```

Approach 2 (Optimal)

Initialize `min` as the first price.
Initialize `maxProfit` as 0.
Loop through the prices from index 1 to the end:
If the current price minus min is greater than maxProfit, update maxProfit.
If the current price is less than min, update min to this new lower value.
Return `maxProfit` at the end.

Dry Run

```
prices = [7, 1, 5, 3, 6, 4]
min = 7, maxProfit = 0

i = 1 → prices[1] = 1
1 < 7 → update min = 1

i = 2 → prices[2] = 5
5 - 1 = 4 > 0 → update maxProfit = 4

i = 3 → prices[3] = 3
3 - 1 = 2 < 4 → no change

i = 4 → prices[4] = 6
6 - 1 = 5 > 4 → update maxProfit = 5

i = 5 → prices[5] = 4
4 - 1 = 3 < 5 → no change
```

➡️ `Final maxProfit = 5` ✅

Time and Space Complexity

Time Complexity: O(n)

One loop through the prices array.

Space Complexity: O(1)

Only a few variables used (`min` , `maxProfit`).

JavaScript | C++ | C | Java | Python | C#

```javascript
var maxProfit = function(prices) {
    let min = prices[0];
    let maxProfit = 0;
    for (let i = 1; i < prices.length; i++) {
        if (prices[i] - min > maxProfit) {
            maxProfit = prices[i] - min;
        }
        if (prices[i] < min) {
            min = prices[i];
        }
    }
    return maxProfit;
};
```

Video Course Discuss doubts Certificate

Best Time to Buy and Sell Stocks I - DSA Notes

Best Time to Buy and Sell Stocks I - DSA Notes

19 of 186 lessons 10% complete

Remove Element

22m 46s Resources

Reverse String

26m 45s Resources

Best Time to Buy and Sell Stocks

28m 16s Resources

Merge Sorted Arrays

41m 33s Resources

Move Zeros

26m 31s Resources

Max Consecutive Ones

16m 29s Resources

Missing Number

16m 55s Resources

Single Number

19m 22s Resources

Recursion - Easy/Medium

Searching & Sorting - Easy/Medium

Linked List - Easy/Medium

Strings - Easy/Medium