

## Homework – 5

### Program Structures and Algorithms

#### Buying and Selling of Stocks

##### Part – 1: nsquare\_time\_constant\_space

**121. Best Time to Buy and Sell Stock** Solved

Easy Topics Companies

You are given an array `prices` where `prices[i]` is the price of a given stock on the `i`<sup>th</sup> 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`.

**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.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
```

**Example 2:**

```
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.
```

**Code**

```
Python3
# YOU CANNOT CHANGE ANYTHING
def maxProfit(self, prices: List[int]) -> int:
    if True:
        [sellday, buyday, work] = self.nsquare_time_constant_space(prices)
    if False:
        [sellday, buyday, work] = self.nlogn_time_logn_space(prices)
    if True:
        [sellday, buyday, work] = self.ntime_constant_space(prices)
    p = self._compute_profit(prices, sellday, buyday)
    return p
```

Accepted Runtime: 75 ms

Case 1 Case 2

Input

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

Output

```
5
```

**Time Limit Exceeded** 198 / 212 testcases passed

Last Executed Input

```
prices =
[10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 9986, 9985, 9984, 9
983, 9982, 9981, 9980, 9979, 9978, 9977, 9976, 9975, 9974, 9973, 9972, 9971, 9970, 9969, 9968, 9967, 996
6, 9965, 9964, 9963, 9962, 9961, 9960, 9959, 9958, 9957, 9956, 9955, 9954, 9953, 9952, 9951, 9950, 9949, 9
948, 9947, 9946, 9945, 9944, 9943, 9942, 9941, 9940, 9939, 9938, 9937, 9936, 9935, 9934, 9933, 9932, 993
1, 9930, 9929, 9928, 9927, 9926, 9925, 9924, 9923, 9922, 9921, 9920, 9919, 9918, 9917, 9916, 9915, 9914, 9
913, 9912, 9911, 9910, 9909, 9908, 9907, 9906, 9905, 9904, 9903, 9902, 9901, 9900, 9899, 9898, 9897, 989
6, 9895, 9894, 9893, 9892, 9891, 9890, 9889, 9888, 9887, 9886, 9885, 9884, 9883, 9882, 9881, 9880, 9879, 9
878, 9877, 9876, 9875, 9874, 9873, 9872, 9871, 9870, 9869, 9868, 9867, 9866, 9865, 9864, 9863, 9862, 9861]
```

**Code**

```
Python3
class Solution:
    def __init__(self):
        pass
        ## You can have what ever you want here

    ##LEETCODE INTERFACE: DO NOT CHANGE
    ## YOU CANNOT CHANGE ANYTHING
```

Accepted Runtime: 75 ms

Case 1 Case 2

Input

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

Output

```
5
```

```
def _nsquare_time_constant_space(self,a):  
    length = len(a)  
    sellday = 0  
    buyday = 0  
    work = 0  
    maxprofit = 0  
    for i in range(length):  
        for j in range(i+1,length):  
            if((a[j]-a[i])>=0 and (a[j]-a[i])>=maxprofit):  
                maxprofit = a[j]-a[i]  
                sellday = j  
                buyday = i  
                work+=1  
    return [sellday,buyday,work]
```

## Part – 2: nlogn\_time\_logn\_space

**121. Best Time to Buy and Sell Stock** Solved

Easy Topics Companies

You are given an array `prices` where `prices[i]` is the price of a given stock on the  $i^{\text{th}}$  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.

**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.  
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

**Example 2:**

**Input:** `prices = [7,6,4,3,1]`  
**Output:** 0  
**Explanation:** In this case, no transactions are done and the max profit = 0.

29.9K 279

**Code**

```
Python3
# YOU CANNOT CHANGE ANYTHING
def maxProfit(self, prices: List[int]) -> int:
    if False:
        [sellday, buyday, work] = self.nsquare_time_constant_space(prices)
    if True:
        [sellday, buyday, work] = self.nlogn_time_logn_space(prices)
    if False:
        [sellday, buyday, work] = self.ntime_constant_space(prices)
    p = self._compute_profit(prices, sellday, buyday)
    return p
#####
```

Saved to local Ln 12, Col 16

**Testcase | Test Result**

prices =  
[7,1,5,3,6,4]

Output  
5

Expected  
5

**Accepted** Editorial Solution

shahdarshil622 submitted at Feb 12, 2024 10:47

**Runtime** 1285 ms  
Beats 7.83% of users with Python3

**Memory** 27.64 MB  
Beats 12.75% of users with Python3

15%  
10%  
5%  
0% 95ms 275ms 454ms 633ms 812ms 992ms 1171ms 1350ms

Code: Python3

```
class Solution:
    def __init__(self):
        pass
    ## You can have what ever you want here
```

**Code**

```
Python3
# YOU CANNOT CHANGE ANYTHING
def maxProfit(self, prices: List[int]) -> int:
    if False:
        [sellday, buyday, work] = self.nsquare_time_constant_space(prices)
    if True:
        [sellday, buyday, work] = self.nlogn_time_logn_space(prices)
    if False:
        [sellday, buyday, work] = self.ntime_constant_space(prices)
    p = self._compute_profit(prices, sellday, buyday)
    return p
#####
```

Saved to local Ln 12, Col 16

**Testcase | Test Result**

prices =  
[7,1,5,3,6,4]

Output  
5

Expected  
5

```

def _nlogn_time_logn_space(self,a):
    def maxProfitFinder(a,start,end):

        if (start >= end) :
            return start,start,0

        mid = (start+end) // 2

        sellday_right_array,buyday_right_array,profit_right_array = maxProfitFinder(a,mid+1,end)
        sellday_left_array, buyday_left_array,profit_left_array = maxProfitFinder(a,start,mid)

        maxprofit_cross_array = max(a[mid+1:end+1])
        minprofit_cross_array = min(a[start:mid+1])

        buyday_cross_profit = a.index(minprofit_cross_array,start,mid+1)
        sellday_cross_profit = a.index(maxprofit_cross_array,mid+1,end+1)

        cross_profit = maxprofit_cross_array - minprofit_cross_array

        if (profit_left_array >= profit_right_array and profit_left_array >= cross_profit) :
            return (sellday_left_array, buyday_left_array, profit_left_array)

        if (profit_right_array >=cross_profit and profit_right_array >= profit_left_array) :
            return sellday_right_array,buyday_right_array,profit_right_array

        if (cross_profit >= profit_left_array and cross_profit >= profit_right_array) :
            return sellday_cross_profit, buyday_cross_profit, cross_profit

        return start,start,0

    sellday,buyday,max_profit = maxProfitFinder(a,0,len(a)-1)

    return [sellday,buyday,(len(a) * math.log(len(a)))]

```

## Part 3 - ntime\_constant\_space

**121. Best Time to Buy and Sell Stock** Solved

Easy Topics Companies

You are given an array `prices` where `prices[i]` is the price of a given stock on the  $i^{\text{th}}$  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`.

**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.  
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

**Example 2:**

Input: `prices = [7,6,4,3,1]`  
Output: `0`  
Explanation: In this case, no transactions are done and the max profit = 0.

29.9K 279

```
Python3
def maxProfit(self, prices: List[int]) -> int:
    if False:
        [sellday, buyday, work] = self.nsquare_time_constant_space(prices)
    if False:
        [sellday, buyday, work] = self.nlogn_time_logn_space(prices)
    if True:
        [sellday, buyday, work] = self.ntime_constant_space(prices)
    p = self._compute_profit(prices, sellday, buyday)
    return p
```

Testcase | Test Result

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

Output: `5`

Expected: `5`

Accepted

shahdarshil622 submitted at Feb 12, 2024 10:49

Runtime: 750 ms (Beats 9.62% of users with Python3)

Memory: 27.58 MB (Beats 95.99% of users with Python3)

Code: Python3

```
class Solution:
    def __init__(self):
        pass
    ## You can have what ever you want here
```

```
Python3
def maxProfit(self, prices: List[int]) -> int:
    if False:
        [sellday, buyday, work] = self.nsquare_time_constant_space(prices)
    if False:
        [sellday, buyday, work] = self.nlogn_time_logn_space(prices)
    if True:
        [sellday, buyday, work] = self.ntime_constant_space(prices)
    p = self._compute_profit(prices, sellday, buyday)
    return p
```

Testcase | Test Result

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

Output: `5`

Expected: `5`

```
def _ntime_constant_space(self,a):
    sell = 1
    buy = 0
    sellday=0
    buyday=0
    work=0
    maxprofit=0
    while(sell<len(a)):
        if((a[sell]-a[buy])>=0 and (a[sell]-a[buy])>=maxprofit):
            maxprofit = a[sell]-a[buy]
            sellday = sell
            buyday = buy
            work+=1
        if((a[sell]-a[buy])<0):
            buy = sell
            sell+=1
    return [sellday,buyday,work]
```