Dear reader, welcome to the article on the problem named **‘**[**Buy & Sell Stock - One Transaction Allowed**](https://www.pepcoding.com/resources/online-java-foundation/dynamic-programming-and-greedy/buy-and-sell-stocks-ota-official/ojquestion)**’.**

It is the 1st problem in the series Buy & Sell Stock. This series of 6 problems are very interesting and should be practiced in a group. 3 problems in this series are based on Dynamic Programming and other 3 problems are based on Greedy Algorithms.

***Problem Statement:***

* You are given an array of n elements. Element at index i represents a stock value on ith day.
* You need to print the maximum profit you can make if you are allowed a ***single transaction***.
* A transaction refers to buying a stock on a given day, and selling it on a day in future.
* Thus the ***profit*** of buying the stock will be the price of stock on the day of selling minus the price of stock on the day of buying.
* Nobody will laugh at you, if you didn't know this! It is very trivial, but still, I want to let you know that you cannot sell a stock before buying it! You cannot have a state “SB”.

If you are not able to understand the problem statement completely, I request you to watch  [this](https://www.youtube.com/watch?v=4YjEHmw1MX0) video from *[0:00, 0:30]* to have a better understanding about the question.

***Example***:

Consider the array of days as : *[7,1,5,3,6,4].* The maximum profit can be achieved by:

Buying on day with index 1 at price 1 and selling it on day with index 4 at price 6. Thus the maximum profit of this transaction will be (6 - 1) = 5.

If you are not able to make any profit, then return 0. For example, consider the array of days as: *[5, 4, 3, 2, 1]*. You will always have to buy expensive stock and sell it for a cheap price. Hence, the maximum profit will be 0 in this case.

***Solution: O(N) Time & O(1) Space***

To maximize profit, we need to find a pair so that we can buy the stock at the least possible price and sell it on a future day at the maximum price possible.

But we cannot make a pair of the minimum element of array with the maximum element of array, because it may be possible that minimum element occurs after maximum, i.e. index of minimum element is greater than index of maximum element. Thus, the condition of a valid complete transaction will be violated.

Hence, everyday can be a potential answer as a selling point or a buying point. We will apply our ***dynamic programming*** approach and try to solve this problem, by ***Travel & Solve Methodology***.

***MEANING***

So, what we can do is consider ***each day*** as our ***potential selling point***. After considering a given day as the potential selling day, we will have to find a day with least price in the left part (index smaller than the selling point), which will act as the buying point.

Thus the profit we can make with the necessary condition that today is the selling day can be calculated as arr[selling point] - arr[buying point].

We will consider these profits for each day and find the maximum profit among them.

***DIRECTION***

What can be the smallest problem?

I think the first day (0 index) is the smallest problem, where it has no buying point before it. Hence, the maximum profit will be 0 considering this day as a potential selling point.

We will keep the current maximum profit = 0. Now, we will traverse the array from ***left to right***, and will keep on updating the maximum profit, if we find a greater profit for the current day.

***TRAVEL & SOLVE***

But, how to find the ***buying point pair***?

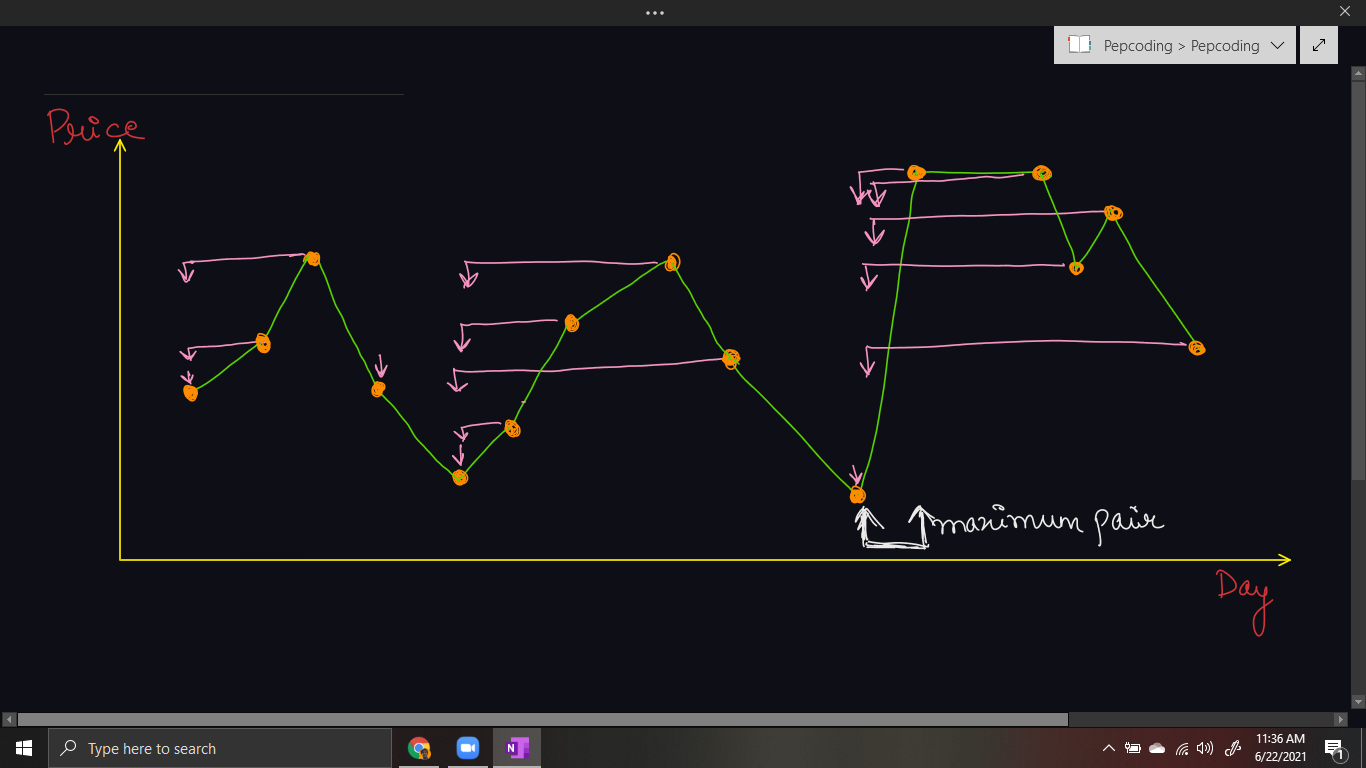
One idea is to store the minimum element encountered so far (from days on the left). On each day, if the price is less than the *msf* (Minimum So Far), we will update its value. Thus, while travelling from left to right, we are able to find the buying point pair.

***Pseudo Code/ Algorithm***

* Initialize maximum profit (op) = 0.
* Initialize minimum so far (msf) = 0.
* Run a loop i from index 0 to n-1.
  + If arr[i] < msf
    - Update msf = arr[i]
  + Calculate profit by considering current day as potential selling point
    - cp = arr[i] - msf
  + If the current profit cp is greater than op (maximum profit)
    - Update op = cp
* Return the value of op (maximum profit)

***Analysis***

Let us analyze the algorithm discussed using an example, plotted on a Price vs Day graph.



***Implementation (Java)***

*Note*: Before reading the Code, we recommend that you must try to come up with the solution on your own. Now, hoping that you have tried by yourself, here is the Java code.

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.parseInt(br.readLine());

int[] arr = new int[n];

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

arr[i] = Integer.parseInt(br.readLine());

}

int msf = arr[0];

int op = 0;

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

if(arr[i] < msf){

msf = arr[i];

}

int cp = arr[i] - msf;

if(cp > op){

op = cp;

}

}

System.out.println(op);

}

}

Java Code is written and explained by our team in the [solution video](https://www.youtube.com/watch?v=4YjEHmw1MX0) from timestamp *[10:17 , 14:30]*.

* What is the ***time complexity*** of the above code?

Since, we are running a loop on a 1d array of size n, and calculating the minimum so far(which is constant work), hence time complexity is ***O(n)***.

* What is the ***space complexity*** of the above code?

We are not using any extra space, but just a few integer variables, hence ***O(1)*** auxiliary space required.

***Follow Up:***

**Q)** We have calculated the maximum profit, but can you modify the algorithm to print the pair of buying & selling points, due to which the maximum profit occurred?

**R)** It is very simple, just a few tweaks in the implementation/code. We have to store the index of the minimum so far (msf), along with the value and then update the result (pair of integers) with the same logic. Try to code this, it is not hard, trust me!

***Asked in Companies***: *Accolite, Amazon, Directi, Flipkart, Goldman Sachs, Hike, MakeMyTrip, Microsoft, Morgan Stanley, Ola Cabs, Oracle, Paytm, Pubmatic, Quikr, Samsung, SAP Labs, Sapient, Swiggy, Walmart*

Hope that you liked the article on *‘Buy And Sell Stock - One Transaction Allowed’*. Next problem will be the 2nd problem in this series ‘[*Buy And Sell Stock - Infinite Transactions Allowed*](https://www.pepcoding.com/resources/online-java-foundation/dynamic-programming-and-greedy/buy-and-sell-stocks-ita-official/ojquestion)*’*’.

Subscribe to Pepcoding’s youtube channel for more such amazing content on Data Structures & Algorithms and follow the resources available for all students in the resources section of Pepcoding’s website.

You can suggest any improvements to the article on our telegram channel, or on the youtube channel’s comment section.

Article Contributed by:

Archit Aggarwal