Stock Buy Sell to Maximize Profit

3.1

The cost of a stock on each day is given in an array, find the max profit that you can make by buying and selling in those days. For example, if the given array is {100, 180, 260, 310, 40, 535, 695}, the maximum profit can earned by buying on day 0, selling on day 3. Again buy on day 4 and sell on day 6. If the given array of prices is sorted in decreasing order, then profit cannot be earned at all.

Recommended: Please solve it on "*PRACTICE* " first, before moving on to the solution.

If we are allowed to buy and sell only once, then we can use following algorithm. Maximum difference between two elements. Here we are allowed to buy and sell multiple times. Following is algorithm for this problem.

1. Find the local minima and store it as starting index. If not exists, return.

2. Find the local maxima. and store it as ending index. If we reach the end, set the end as ending index.

3. Update the solution (Increment count of buy sell pairs)

4. Repeat the above steps if end is not reached.

C

```
// Program to find best buying and selling days
#include <stdio.h>

// solution structure
struct Interval
{
    int buy;
    int sell;
};

// This function finds the buy sell schedule for maximum profit
```

```
void stockBuySell(int price[], int n)
{
    // Prices must be given for at least two days
    if (n == 1)
        return;

    int count = 0; // count of solution pairs

    // solution vector
    Interval sol[n/2 + 1];

    // Traverse through given price array
    int i = 0;
    while (i < n-1)
    {
        // Find Local Minima. Note that the limit is (n-2) as we are
        // comparing present element to the next element.
        while ((i < n-1) && (price[i+1] <= price[i]))
            i++;

        // If we reached the end, break as no further solution possible
        if (i == n-1)
            break;

        // Store the index of minima
        sol[count].buy = i++;

        // Find Local Maxima.  Note that the limit is (n-1) as we are
        // comparing to previous element
        while ((i < n) && (price[i] >= price[i-1]))
            i++;

        // Store the index of maxima
        sol[count].sell = i-1;

        // Increment count of buy/sell pairs
        count++;
    }

    // print solution
    if (count == 0)
        printf("There is no day when buying the stock will make profitn");
    else
    {
       for (int i = 0; i < count; i++)
          printf("Buy on day: %dt Sell on day: %dn", sol[i].buy, sol[i].sell);
    }

    return;
}

// Driver program to test above functions
int main()
{
    // stock prices on consecutive days
    int price[] = {100, 180, 260, 310, 40, 535, 695};
    int n = sizeof(price)/sizeof(price[0]);

    // fucntion call
    stockBuySell(price, n);

    return 0;
}
```

Run on IDE

Java

```
// Program to find best buying and selling days
import java.util.ArrayList;

// Solution structure
class Interval
{
    int buy, sell;
}

class StockBuySell
{
    // This function finds the buy sell schedule for maximum profit
    void stockBuySell(int price[], int n)
    {
        // Prices must be given for at least two days
        if (n == 1)
            return;

        int count = 0;

        // solution array
        ArrayList<Interval> sol = new ArrayList<Interval>();

        // Traverse through given price array
        int i = 0;
        while (i < n - 1)
        {
            // Find Local Minima. Note that the limit is (n-2) as we are
            // comparing present element to the next element.
            while ((i < n - 1) && (price[i + 1] <= price[i]))
                i++;

            // If we reached the end, break as no further solution possible
            if (i == n - 1)
                break;

            Interval e = new Interval();
            e.buy = i++;
            // Store the index of minima


            // Find Local Maxima.  Note that the limit is (n-1) as we are
            // comparing to previous element
            while ((i < n) && (price[i] >= price[i - 1]))
                i++;

            // Store the index of maxima
            e.sell = i-1;
            sol.add(e);

            // Increment number of buy/sell
            count++;
        }

        // print solution
        if (count == 0)
            System.out.println("There is no day when buying the stock "
                                                + "will make profit");
        else
            for (int j = 0; j < count; j++)
                System.out.println("Buy on day: " + sol.get(j).buy
                        +"        " + "Sell on day : " + sol.get(j).sell);

        return;
    }

    public static void main(String args[])
    {
        StockBuySell stock = new StockBuySell();
```

```
        // stock prices on consecutive days
        int price[] = {100, 180, 260, 310, 40, 535, 695};
        int n = price.length;

        // fucntion call
        stock.stockBuySell(price, n);
    }
}

// This code has been contributed by Mayank Jaiswal
```

Run on IDE

Output:

```
Buy on day : 0   Sell on day: 3
Buy on day : 4   Sell on day: 6
```

Asked in: Accolite,Amazon,Directi,Flipkart,Goldman-Sachs,Intuit ,MakeMyTrip,Microsoft,Ola-Cabs,Oracle,Paytm,Pubmatic,Quikr,Sapinet,Swiggy,Walmart labs

Time Complexity: The outer loop runs till i becomes n-1. The inner two loops increment value of i in every iteration. So overall time complexity is O(n)

This article is compiled by Ashish Anand and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

Arrays

Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Maximum difference between two elements such that larger element appears after the smaller number
Rearrange positive and negative numbers in O(n) time and O(1) extra space
Find the maximum repeating number in O(n) time and O(1) extra space
Two elements whose sum is closest to zero
Smallest subarray with sum greater than a given value
Maximum number of chocolates to be distributed equally among k students
Range Query on array whose each element is XOR of index value and previous element
Range sum query using Sparse Table
Minimum cost to make array size 1 by removing larger of pairs
Python | Check if all the values in a list that are greater than a given value

Logged in as **himanshudce**(Logout)

3.1 Average Difficulty : **3.1/5.0**
Based on **160** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!