

Stock Buy Sell to Maximize Profit

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.

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.

```
// 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
```

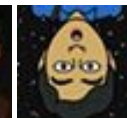
Google™ Custom Search



GeeksforGeeks



53,519 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

```

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 profit")
else
{
    for (int i = 0; i < count; i++)
        printf("Buy on day: %d\t Sell on day: %d\n", sol[i].buy, sol[i].sell);
}

return;
}

```

Deploy Early. Deploy Often.

DevOps from Rackspace:

Automation

[FIND OUT HOW ►](#)



Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and](#)

[Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

```
// 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;
}
```

Output:

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

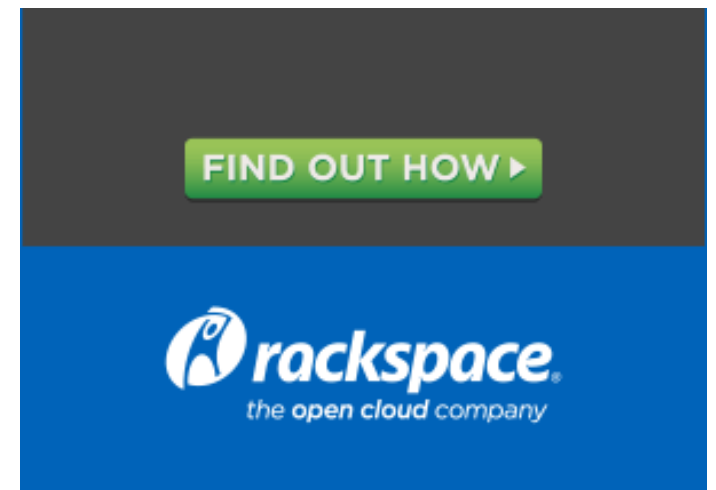
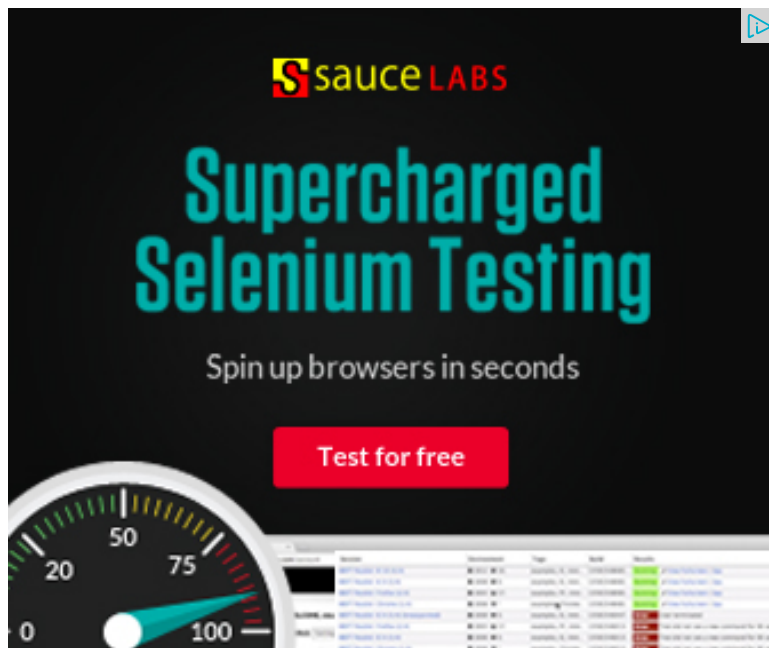
Time Complexity: The outer loop runs till i becomes n-1. The inner two loops increment value of i in every iteration. So overall time complexity is $O(n)$

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.

Deploy Early. Deploy Often.

DevOps from
Rackspace:

Automation



Related Topics:

- Remove minimum elements from either side such that $2 \times \text{min}$ becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3



Writing code in comment? Please use ideone.com and share the link here.

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 2 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 5 minutes ago

Sanjay Agarwal bool

tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 30 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 32 minutes ago

newCoder3006 If the array contains negative


numbers also. We...

[Find subarray with given sum · 57 minutes ago](#)

newCoder3006 Code without using while

loop. We can do it...

[Find subarray with given sum · 1 hour ago](#)

AdChoices 

[▶ C++ Code](#)

[▶ Programming C++](#)


[▶ Stock and Stocks](#)

AdChoices 

[▶ Pricing Stock](#)

[▶ A Stock Price](#)

[▶ Stock Pair](#)

AdChoices 

[▶ Stock Operator](#)

[▶ Stock Cases](#)

[▶ Stock Gain](#)

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team