**Analysis**

**Abstract**

Given Information about the price of stock, the maximum-subarray problem can be solved in linear time by a simple incremental algorithm that scans the input array from left to right. On the other hand, the divide & conquer solution for this problem has super-linear running time. And, we present a non-recursive algorithm that takes only linear time and thus is optimal.

Maximum subarray using divide & conquer

The well-known divide & conquer approach to solve the maximum-subarray problem involves splitting the array in half by the median index and making recursive calls on each of the two subarrays to find the maximum subarray on the left half and the maximum subarray on the right half. The combine step searches for the maximum subarray that begins in the left half of the array and ends in the right half. An overall maximum is then reported as the maximum of the three (left, right, and cross). Since the combine step requires a scan from the middle index of A to the left and to the right, a linear term to the recurrence solution is added due to this step, resulting in a final recurrence of T(n) = 2T(n/2) + cn with time complexity of Θ(n log n).

Step1. Select the middle element of the array.  
So the maximum subarray may contain that middle element or not.

Step 2.1 If the maximum subarray does not contain the middle element, then we can apply the same algorithm to the the subarray to the left of the middle element and the subarray to the right of the middle element.

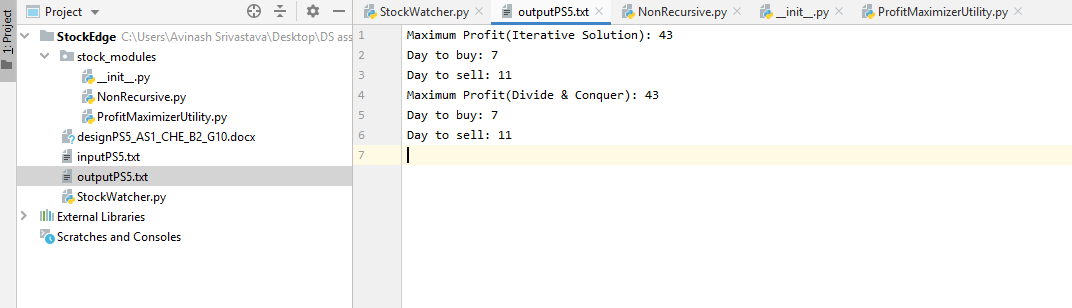
Step 2.2 If the maximum subarray does contain the middle element, then the result will be simply the maximum suffix subarray of the left subarray plus the maximum prefix subarray of the right subarray

Step 3 return the maximum of those three answers.

A close up of a piano

Description automatically generated

**Output :**



**Time Complexity for Maximum Subarray using Divide and conquer Algorithm**

The time complexity of above Divide and Conquer solution is o(nlogn) as for given

array of size n, we make two recursive calls on input size n/2 and finding maximum

subarray that crosses midpoint takes 0(n) time in worst case.

So,

**T(n)=2T(n/2)+O(n)**

**T(n)=O(nlogn)**

We can solve this problem in O(n) time using Kadane's Algorithm

Iterative Approach

Using this approach the Time Complexity is : **O(n^2)**