Optional Question.

Can we design a Divide & conquer algorithm in linear run time?

Yes, by slightly modifying the Divide & conquer algorithm. We reduce the time complexity of the combining phase of the algorithm to O(1) thereby making the overall complexity of the algorithm as O(n) i.e; linear run time. We return additional information about (sub array) in recursive calls to reduce the combine phase to O(1).

Another Approach for Linear run time in MSS:

* The whole array is scanned just once there making the run time as O(n).
* Initially we have an empty sub array.
* There are two variables sum & answer. Sum decides which elements are to be added to the sub array where as answer returns the maximum sum of the sub array.
* If the sum of the elements is positive then the elements are added to the sub array. and if the sum of the elements is negative then the sum is reset to the initial empty sub array.
* The pseudo Answer returns the maximum sub array sum.

Code

```
int Maximum_sum_sub array (int a[], int n)
 {
int answer=0, sum_of_array=0;
for (i=0; i<n; i++)
{
if ( sum_of_array +a[i]>0)
 sum_of_array += arr[i];
else
sum_of_array =0;
answer = max(answer, sum_of_array);
}
return answer;
}.
```

Example

a= {1, -5, -2, 3, 4, 7, 8, -20}

1	-5	-2	3	4	7	8	-20
i							
0	1	2	3	4	5	6	7

Initially sub array is empty

sum stores the sub array

For the first iteration

i=0

sum = 1

answer=1

In the second iteration

i=1

sum = 1-5 = -4 as it is negative

sum is reset to initial empty sub array.

sum = {3} till i=2

when i=3

sum = [3]

answer = 3

when i=4

sum = [3|4]

answer = 7

when i=5

sum = [3|4|7]

answer = 14

when i=6

sum = [3|4|7|8]

answer = 22

when i=7

sum = [3|4|7|8|-20]

answer = 2

The above program returns the maximum sub array with maximum answer

ie [3|4|7|8]

answer = 22.

Correctness

-> Before the first iteration starts the initial subarray {3} is empty and does not contain any value.

-> Assume that before the i^{th} iteration subarray A[1.... (i-1)] does not contain maxsum. If sum of array a[i] < 0 then the subarray is initialized to empty subarray. If sum of array < 0 then answer is equal to empty subarray where i = 1, 2, 3... i where the invariant will remain true.

-> It terminates in two conditions. when maximum subarray is the whole array and the whole array is returned. In the next case it terminates after iteration through all 'n' elements.