SRM
INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

20CSC501J – Data Structures and Algorithm

Unit III

Divide and Conquer

SRM
INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

UNIT III

Divide-and-Conquer- The maximum-subarray problem- Brute force solution for the maximum-subarray problem- Divide and conquer solution for the maximum-subarray problem- Multiplication of large integers- Strassen's algorithm for matrix multiplication

Divide and conquer : Convex Hull algorithm- Divide and conquer : Convex Hull algorithm Analysis- Divide and conquer : Median Finding- Divide and conquer : Median Finding Algorithm- Divide and conquer : Binary tree traversals- Binary tree traversals related properties

Closest pair problem- Algorithm of closest pair problem- Analysis of closest pair problem- Divide & Conquer: Van Emde Boas Trees, Preliminary approaches- A recursive structure- The van Emde Boas tree

Divide and conquer is an algorithm design paradigm.

A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly.

The solutions to the sub-problems are then combined to give a solution to the original problem.

The maximum sub array problem is a task to find the series of contiguous elements with the maximum sum in any given array.

-3	1	-8	4	-1	2	1	-5	5

For instance, in the above array, the highlighted sub array has the maximum sum(6):

Approaches

- Brute force
- Divide and conquer
- Greedy Algorithm
- Kadane's Algorithm (Dynamic Programming)

Brute Force Algorithm

Brute force is an iterative approach to solve a problem. In most cases, the solution requires a number of iterations over a data structure. In

Approach

Calculate the sum of every possible sub array and return the one with the maximum sum.
To begin with, calculate the sum of every sub array that starts at index 0. And similarly, find all sub arrays starting at every index from 0 to n-1 where n is the length of the array:

Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example:

Input: [-2,1,-3,4,-1,2,1,-5,4],

Output: 6

Explanation: [4,-1,2,1] has the largest sum = 6.

```
[-2,1,-3,4,-1,2,1,-5,4],
```

Complexity

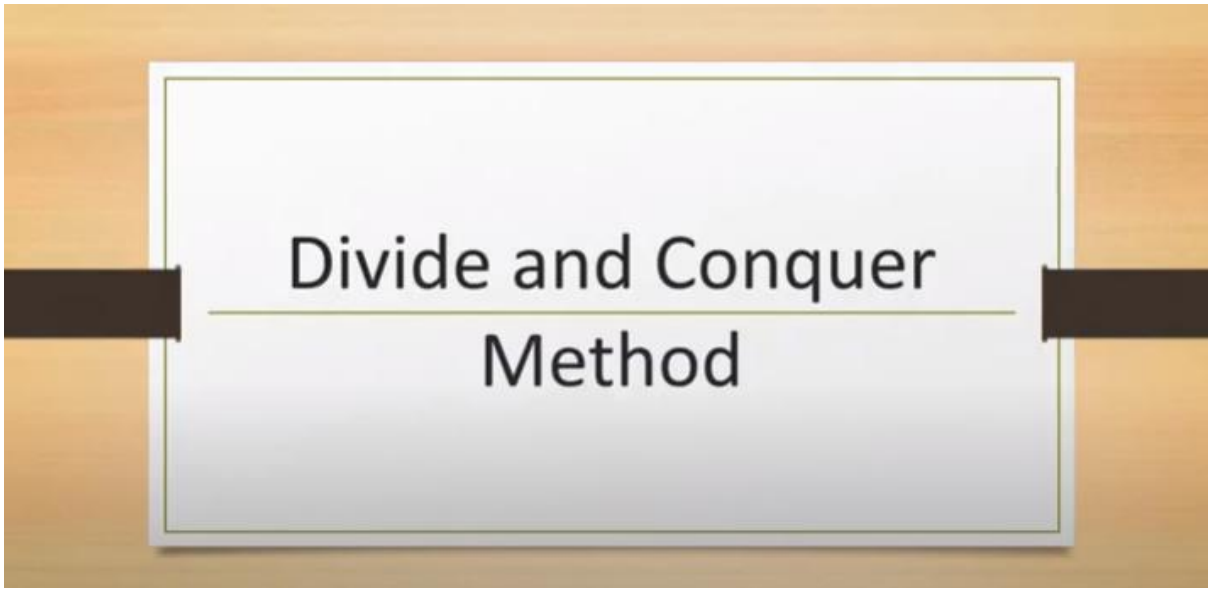
Generally speaking the brute force solution iterates over the array many times in order to get every possible solution.

The time taken by this solution grows quadratically with the number of elements in the array. This may not be a problem for arrays of a small size. **But as the size of the array grows, this solution isn't efficient.**

There are two nested *for* loops. **Therefore, we can conclude that the time complexity of this algorithm is** *O(n2)*.

SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.



Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example:

Input: [-2,1,-3,4,-1,2,1,-5,4],

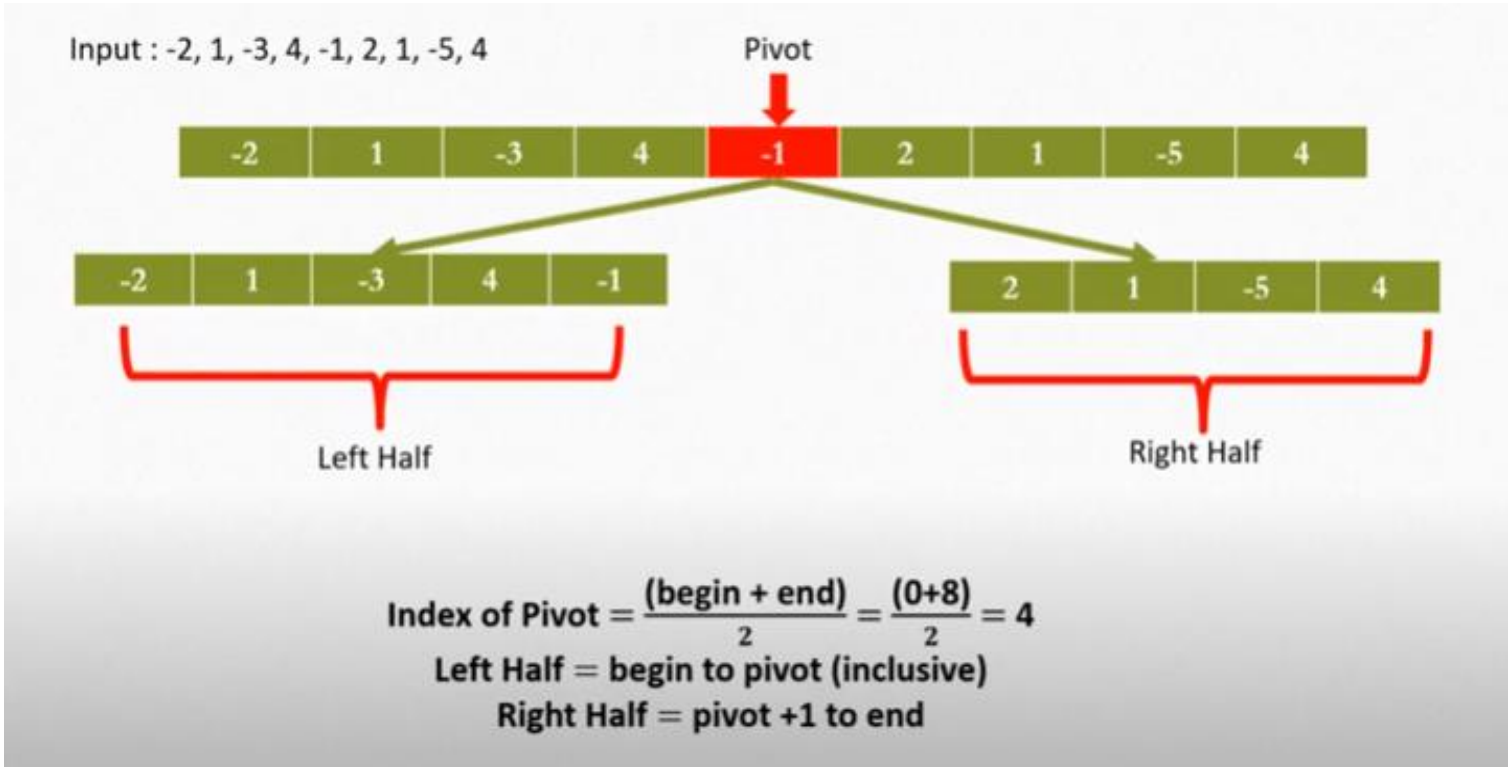
Output: 6

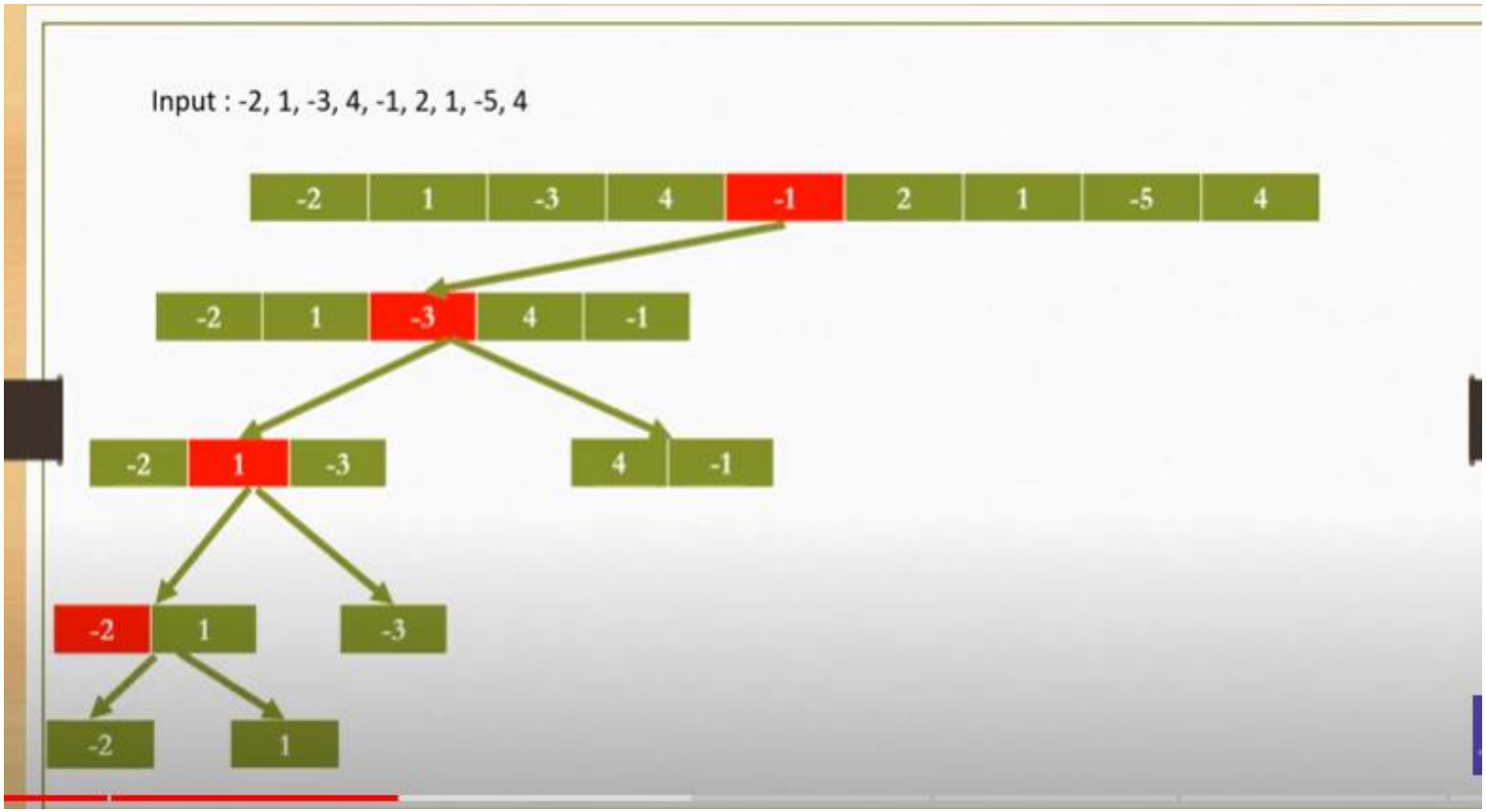
Explanation: [4,-1,2,1] has the largest sum = 6.

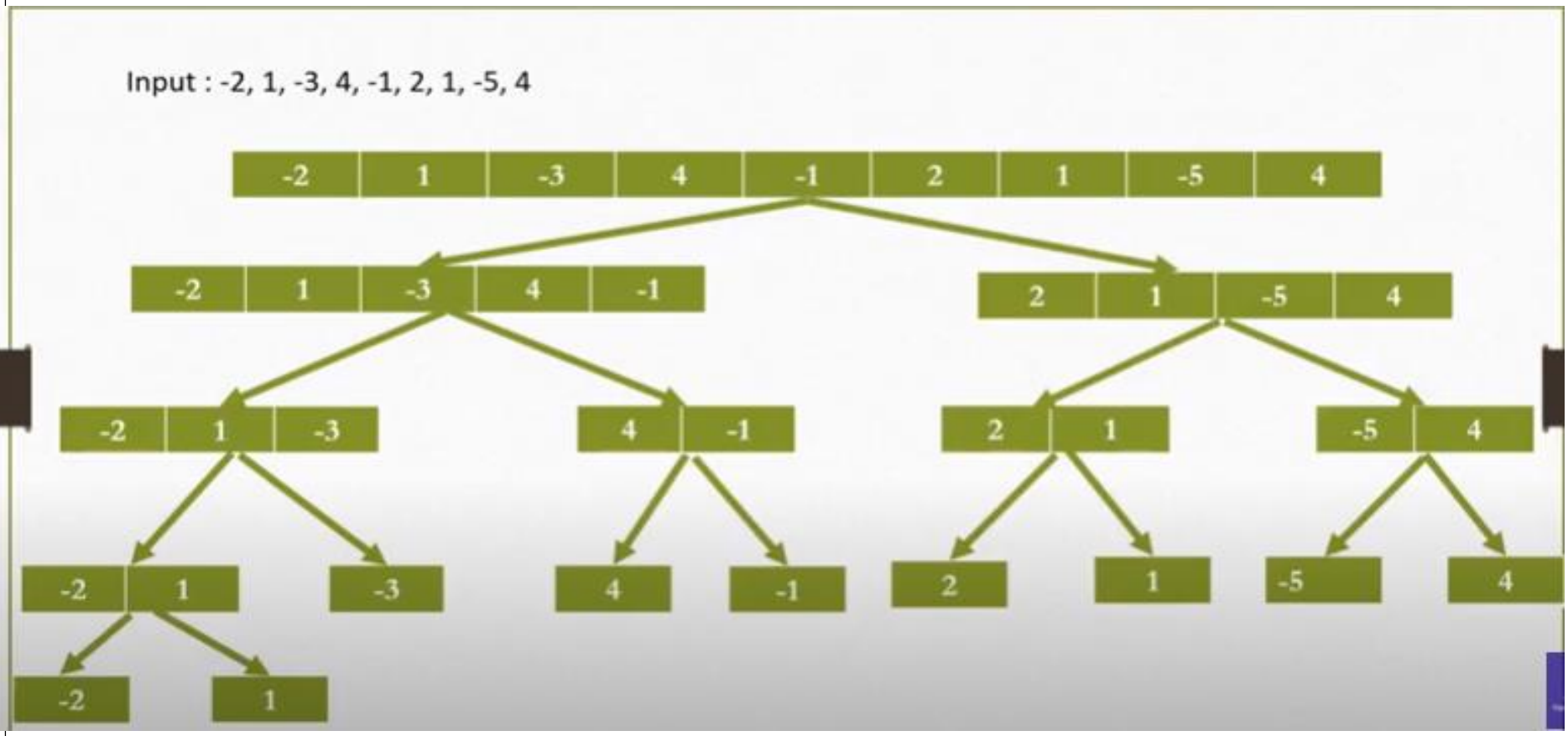
Algorithm

1) Divide the given array in two halves
2) Return the maximum of following three
    a) Recursively calculate the Maximum subarray sum in left half
    b) Recursively calculate the Maximum subarray sum in right half
    c) Recursively calculate the Maximum subarray sum such that the subarray crosses the midpoint.
        i. Find the maximum sum starting from mid point and ending at some point on left of mid.
        ii. Find the maximum sum starting from mid + 1 and ending with some point on right of mid + 1.
        iii. Finally, combine the two and return.

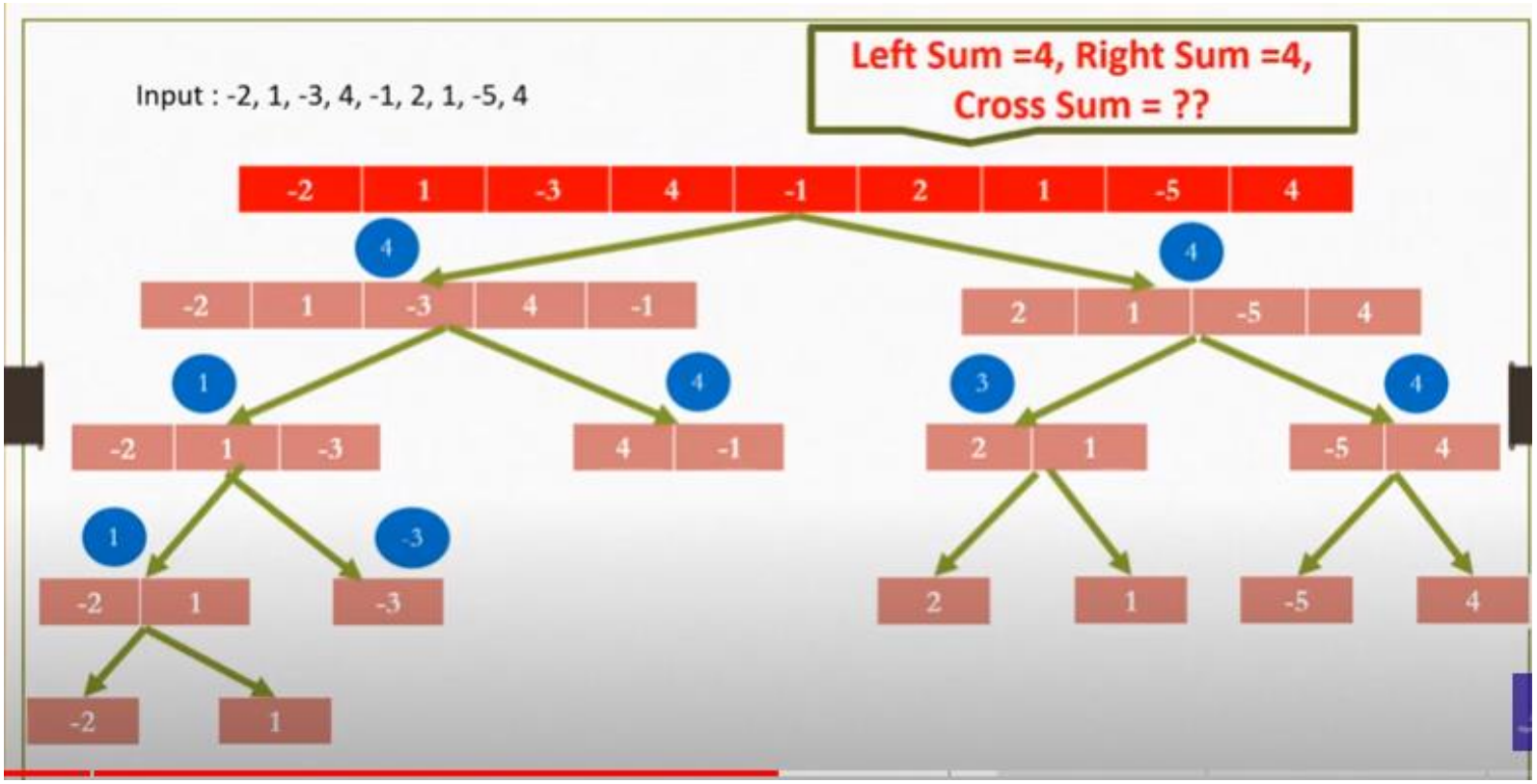
Input : -2, 1, -3, 4, -1, 2, 1, -5, 4


Input : -2, 1, -3, 4, -1, 2, 1, -5, 4
Pivot
-2
1
-3
4
-1
2
1
-5
4
-2
1
-3
4
-1
2
1
-5
4
Left Half
Right Half
Index of Pivot = (begin + end) / 2 = (0+8) / 2 = 4
Left Half = begin to pivot (inclusive)
Right Half = pivot +1 to end


Input : -2, 1, -3, 4, -1, 2, 1, -5, 4
-2 1 -3 4 -1 2 1 -5 4
-2 1 -3 4 -1
-2 1 -3 4 -1
-2 1 -3
-2 1


Input : -2, 1, -3, 4, -1, 2, 1, -5, 4
-2 1 -3 4 -1 2 1 -5 4
-2 1 -3 4 -1
2 1 -5 4
-2 1 -3
4 -1
2 1
-5 4
-2 1
-3
4
-1
2
1
-5
4
-2
1

SRM
INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.



Input : -2, 1, -3, 4, -1, 2, 1, -5, 4

Left Sum =4, Right Sum =4,
Cross Sum = ??

-2	1	-3	4	-1	2	1	-5	4

Cross Sum Calculation :

Pivot Element: -1

Left Sub Sum = Max (-1, -1+4, -1+4-3, -1+4-3+1, -1+4-3+1-2)
= Max (-1, 3, 0, 1, -2)
= 3

Right Sub Sum = Max (2, 2+1, 2+1-5, 2+1-5+4)
= Max (2, 3, -2, 2)
= 3

Cross Sum = Left Sub Sum + Right Sub Sum = (3+3) = 6

Cross Sum = 6

Maximum Sum = Max (Left Sum, Right Sum, Cross Sum) = 6

Time Complexity: maxSubArraySum() is a recursive method and time complexity can be expressed as following recurrence relation.
$T(n) = 2T(n/2) + \Theta(n)$

Using Divide and Conquer approach, we can find the maximum subarray sum in O(nLogn) time.

SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

Thank You.