CS 325

**Divide and Conquer:-**

**Theoretical analysis**

1. **Pseudo code**

The Max-SubArray is the initial call function, the base case is low=high. If a cross occurs then cross-SubArray is called.

**Cross-SubArray(low, mid, high, A)**

//Sub-Array for A[i….mid]

left-sum = [Infinity]

for i = mid to low

sum= sum + A[i]

if sum>left-sum

left-sum=sum

max-left=i

//Sub-Array for A[mid+1….high]

right-sum = [Infinity]

for j = mid+1 to high

sum= sum + A[j]

if sum>right-sum

right-sum=sum

max-right=j

return(max-left, max-right, left-sum+right-sum)

//Initial call Max-SubArray

**Max-SubArray(low, high, A)**

For i to array\_size

If high==low

Return (A[low])

Else mid= [low + high)/2]

(Low-low, Low-high, Low-sum)=

**Max-SubArray (A, low, mid)**

(Right-low, Right-high, Right-sum)=

**Max-SubArray (A, mid+1, high)**

(Cross-low, Cross-high, Cross-sum)=

**Cross-SubArray(low, mid, high, A)**

If left-sum>= right-sum && left-sum>= cross-sum

Return(left-low, left-high, left-sum)

ElseIf right-sum>= left-sum && right-sum>= cross-sum

Return (right-low, right-high, right-sum)

EsleReturn (cross-low, cross-high, cross-sum)

1. **Asymptotic run time analysis**

The original problem size is a power of 2, so all sub sizes are integers. So the run time method of the program will be O(n lg n).

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

1. **Theoretical correctness analysis**

MaxSubarray(A, n) will return the sum of the maximum subarray of an array A of size n.

Proof:

Base Case,:- If n = 0. Then max = current = 0. Then subarray won’t be possible and the max will be zero for array of zero elements.

Case n=1: If n=1 meaning that there is one single element then if the value is positive it will be considered as array and max value will be assigned with the value of the array element.

Inductive hypothesis:- Considering array contain more elements say ‘n’.

The method will find the sum of the whole array and save it in a variable max. and then it will divide the array in such a way that all the possible combinations of the array will be find and computed such that every loop execution the sum is compared with the max value if sum is greater then the value will be stored.

So while n>1 (meaning that more than one element in the array) the array will recursively divided and computes sum for all possible subarrays and compares with the max which will ultimately considered as the maximum value of a subarray.