Reursive

Pseudo code

____

```
Function maxSubArray(arr, start, end):
  if start equals end:
     return arr[start]
  mid = (start + end) / 2
  left = maxSubArray(arr, start, mid)
  right = maxSubArray(arr, mid+1, end)
  max= arr[mid]
  sum = arr[mid]
  for i from mid-1 down to start:
     sum += arr[i]
     if sum is greater than max:
        max = sum
  sum = max
  for i from mid+1 up to end:
     sum += arr[i]
     if sum is greater than max:
        max = sum
  if left is greater than right and left is greater than max:
     return leftMax
  else if right is greater than left and right is greater than max:
     return rightMax
  else:
     return max
```

Main program:

```
arr = {-2, -3, 4, -1, -2, 1, 5, -3}
n = size of arr
maxSum = maxSubArray(arr, 0, n-1)
print "Maximum contiguous sum is ", maxSum
```

Time complexity

____

T(n)=2t(n/2)-4n

T(n/2) =2 t(n/4)-4n/2------T(n)=2(2t(n/4)-2n)-4n

T(n/4) =2t(n/8)-4n/4----------T(n)=2(2(2t(n/8)-n))-4n

T(n/8) =2t(n/16)-4n/8----------T(n)=2(2(2(2t(n/16)-n/2))-4n

T(n)=2^4 t(n/2^4)-2*4n

T(n)=2^kt(n/2^k)-2*k*n

T(n)=1

t(n/2^k) =1

n/2^k=1

K=log n

T(n)=t (1)-2*n*log n

T(n)=O (n log n)

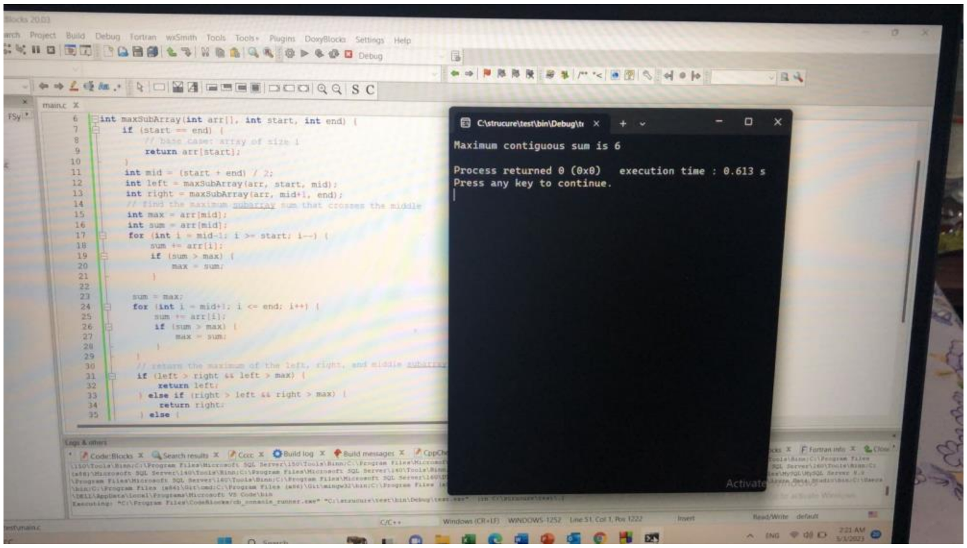
Divided and conquer =O (log n)

Maximum subArray= O(n)

Total T(n)=O (n log n)

Output

____



Non-Reursive

Pseudo-code

```
Function maximal_subarray(n,arr)
{
   Max <- arr[0]
   For i<-0 to n
   {
      Sum <- arr[i]
      For j <- i+1 to n
      {
         Sum <- sum + arr[j]
         If max < sum
            Max = sum
      }
```

```
        If I equals n-1 and max < sum
                Max = sum
    }
    Return max
}
Main program:
   arr = {-2, -1 , -3 , 4 , -1 , 2 , 1 , -5 , 4}
   n = size of arr
If n is equal 0
Print “Enter at least one item”
else
   LargestSum = maximal_subarray(n , arr)
   print "Largest sum = ", LargestSum
```

Time Complexity

____

$$\sum_{i=0}^{n} \sum_{j=i+1}^{n} 1$$

$$\sum_{i=0}^{n} [n-(i+1)+1]$$

$$= \sum_{i=0}^{n} n-i-\cancel{1}+\cancel{1}$$

$$= \sum_{i=0}^{n} n-i$$

$$= \sum_{i=0}^{n} n - \sum_{i=0}^{n} i$$

$$= n\sum_{i=0}^{n} 1 - \sum_{i=0}^{n} i$$

$$n(n-0+1) - \frac{n(n+1)}{2}$$

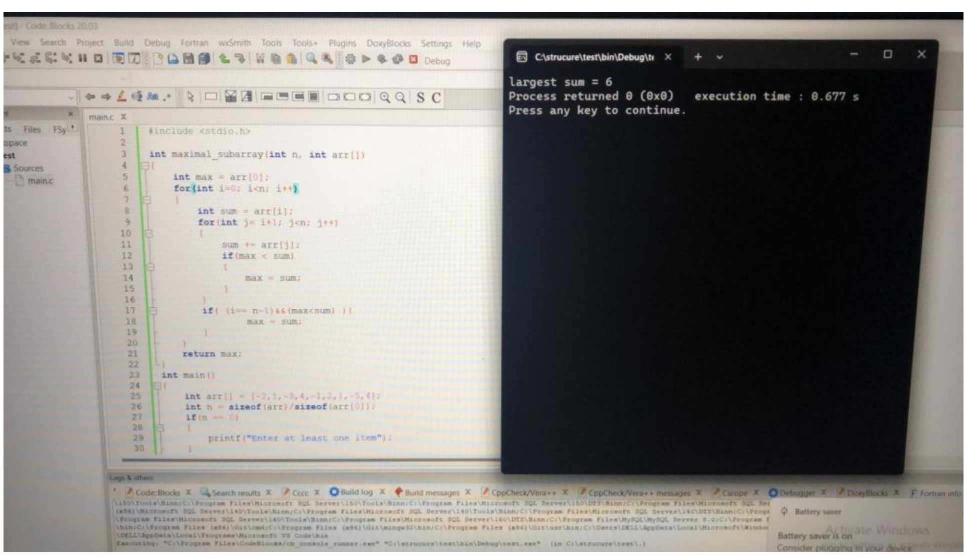
$$n(n+1) - \frac{n}{2}(n+1)$$

$$n^2 + n - \frac{1}{2}n^2 + \frac{1}{2}n$$

$$\underline{\Theta(n^2)} \quad \Omega(n^2) \quad O(n^2)$$

Output

____



Comparison between codes

	None-Recursive	Recursive
Best-Case	Sqr(n)	1
Average-Case	Sqr(n)	nlogn
Worst-Case	Sqr(n)	nlogn

Recursive is best solution

-> because complexity of None-Recursive is Quadratic and complexity of Recursive is log linear

-> algorithms with Quadratic time complexity are much slower than those with log-linear time complexity, especially for large input sizes.