DATA STRUCTURE ASSIGNMENT

TOPIC :- BST AND SORTING.

MADE BY :- GARV SINGHAL.

BRANCH :- CSE.

ROLL NO :- 11911070.

Made by
Garv Singhal
11911070

D-Structure
Assignment
Submission by:
GARV SINGHAL

Ques-1:-
Analysis of time Complexity of any list in Insertion Sort.

For best case list should be in ascending order as we know Algorithm of Insertion Sort:-

```
for(int x = 1 ; x < n ; x++)
    {
      temp = arr[x]
      for(int y = x-1 ; y>=0 ; y--)
      { if (temp < arr[y])
          {
             temp[y+1] = temp[y];
             temp[y] = temp;
          }
        else
        break;
      }
    }
```

ex:- Consider list { 7, 9, 11, 13, 16 }

```
Loop1:-     temp = 9
    Loop2:  (9<7)   else
            false   break;
```

means at $n = 1$ loop 2 has been called 1 time.
Similarly for all cases.

Made by GARU SINGHAL
11911070

(ii) for $x = 2$
temp = 11
Loop 2 $y = 1$ $y > 0$ true
if (11 < arr [1]) then break
False

means at $x = 2$ loop has been called once

So, to sum up:-
For asscending ~~for~~ order

Loop 1	Loop -2	No. of call
$x = 1$	$y = 0$	1
$= 2$	$= 1$	1
$= 3$	$= 2$	1
$= 4$	$= 3$	1
$= 5$	$= 4$	1
$x = n-1$	$y = n-2$	1

[Total loop call = $n-1$]

Time complexity = $O(n-1) \approx O(n)$ Ans

Made by
GARU SINGHAL
11911070

Ques-2

* Bubble Sort
  Algo :-

```
for ( int x = 0 ; x < n-1 ; x++ )
  {
    for (int y = 0 ; y < n ; y++ )
     {
        ~~if(x~~
        if(x[ y ] > x[y+1 ])
          {
            temp = x[y];
            x[y] = x[y+1];
            x[y+1] = temp ;
          }
     }
  }
```

time complexity :-

$T(n) = O(n-1) \times O(n-1)$
$= O[(n-1)^2] \approx O(n^2)$

loop-1 →
loop 2 operates n-1 times
loop-2 →
loo 2 operates n-1 times

Means that loop-1 operates (n-1)
" " loop-2 " (n-1)
$T(n) \approx O(n^2)$

Made by GARV SINGHAL
11911070

For $\boxed{n=5}$

=>	14	26	26	26
	26	14	56	56
	90	56	60	60
	56	90	74	74
	60	90	90	90

Space-Complexity: (O(1) : const)

* Merge Sort Algo:-

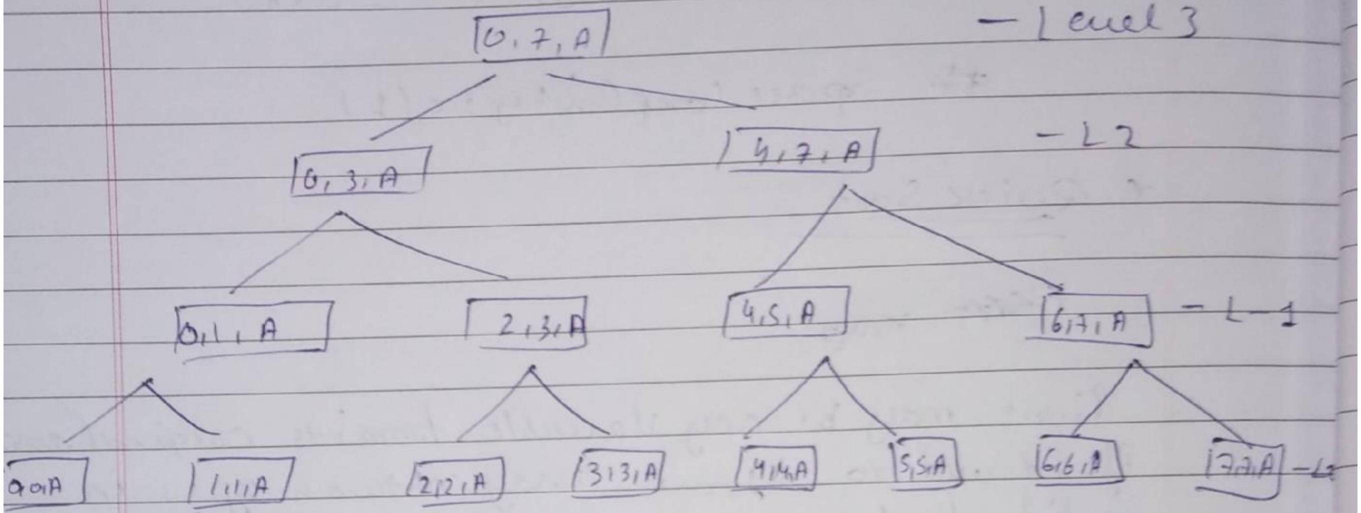
```
Void merge-sort ( int l, int r, int * a )
        {
            if(l<r)
            {
                int m = (l+r)/2 ;

                merge sort(l,m,a);
                merge sort(m+1,r,a);
                merge sort (l,m,r,a);

Void merge(int l, int m, int r, int*p)
      { int n1 = m-l+1;
        int n2 = r-m;
      # array [n1]; - storing initial Data
      # array [n2]; - storing final Data
        then p will store in array 1 & array 2 to
        by sorting with (O(n)).
```

Made by GARU SINGHAL
11911070

Lets consider 8 elements:-



Number of Level O(n) = $\log_2(n) = \log_2(8) = 3$

Time Complexity (T(n)) = n log n

Merge Sort Space Complexity = O(n)

* Insertion Sort Algo:-

```
For(int a=1 ; a<n ; a++)
{
    for(int b=a-1 ; b>=0 ; b++)
    {   if(array[b] >array[b+1])
        {  temp = array[b];
           array[b] = array[b+1];
           array[b+1] = temp;
        }
        else
        break;
    }
}
```

Made by GARV SINGHAL
11911070

time Complexity at worst case :- $O(n^2)$
'' '' best '' $= O(n)$

Space complexity $= O(1)$

* Quick Sort :-

~~Pivot may~~

Pivot may be any variable from its original array. it used to bypart array into two way :- in which first subarray contains all values less than pivot & other sub array will contain all values greater than pivot.

```
Partition(l,m,array) //l=0; m=n-1 (Initially)
{  int start=l, end=m;
     pivot=a[l]
 while( start < end){
      while((a[start <=pivot) && (start <= m) ) {
           start++ ; }
           while( a[end>pivot) && (end >= l) ) end--;
           if (start< end ){
             swaping (array[start], array[end]) ; }
           }
  Swaping(array[l], array[end])
  return end;
  }
```