Ques 1

Analysis of time complexity of any list in insertion sort.

for best case ~~sort~~ 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;
     }
  }
```

consider List is {7, 9, 11, 13, 16}

Loop-1 →

temp = ~~7~~9

Loop 2 {(9 < 7) false else break;

means at $x=1$ Loop-2 has been called 1 time

Similiarly

(ii) for $x=2$

temp = 11

Loop2 $\boxed{y=1}$ ✓ $y>0$ true

if ($11 < arr[1]$) than break

False

means at $x=2$ Loop2 has been called once again -

So we can say - [In Ascending order

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

Total loop call $n-1$

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

Ques ⇒ 2 ~~Merge Sort~~:-

* Bubble sort:-

Algorithm:-

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

Time Complexity :-

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

<u>Loop-1</u>

Loop-2 operates $(n-1)$ times

Loop-2

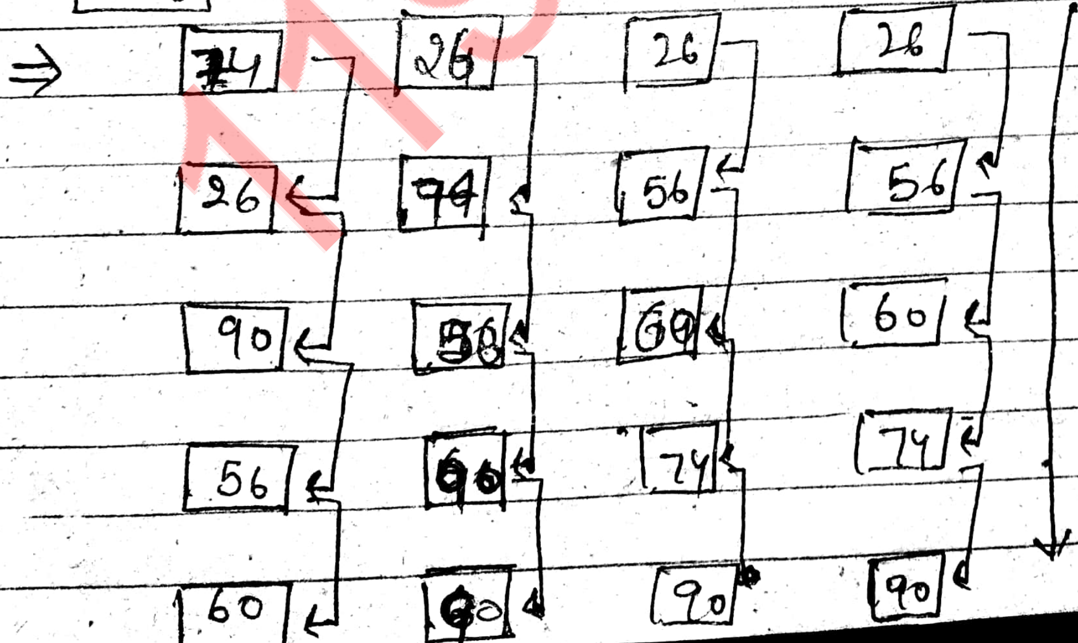
Loop-2 operates $(n-1)$ times

means that Loop-1 operates $(n-1)$
——— Loop-2 ——— $(n-1)$

Net $T(n) = O(n-1)^2$
$\approx O(n^2)$

For $\boxed{n=5}$ ⟶ $(n-1)$

⇒



$\times$
$(n-1) = O(n-1)^2$

Space Complexity:-

$\underline{O(1)} \Rightarrow$ Constant

Merge sort Algorithm:-

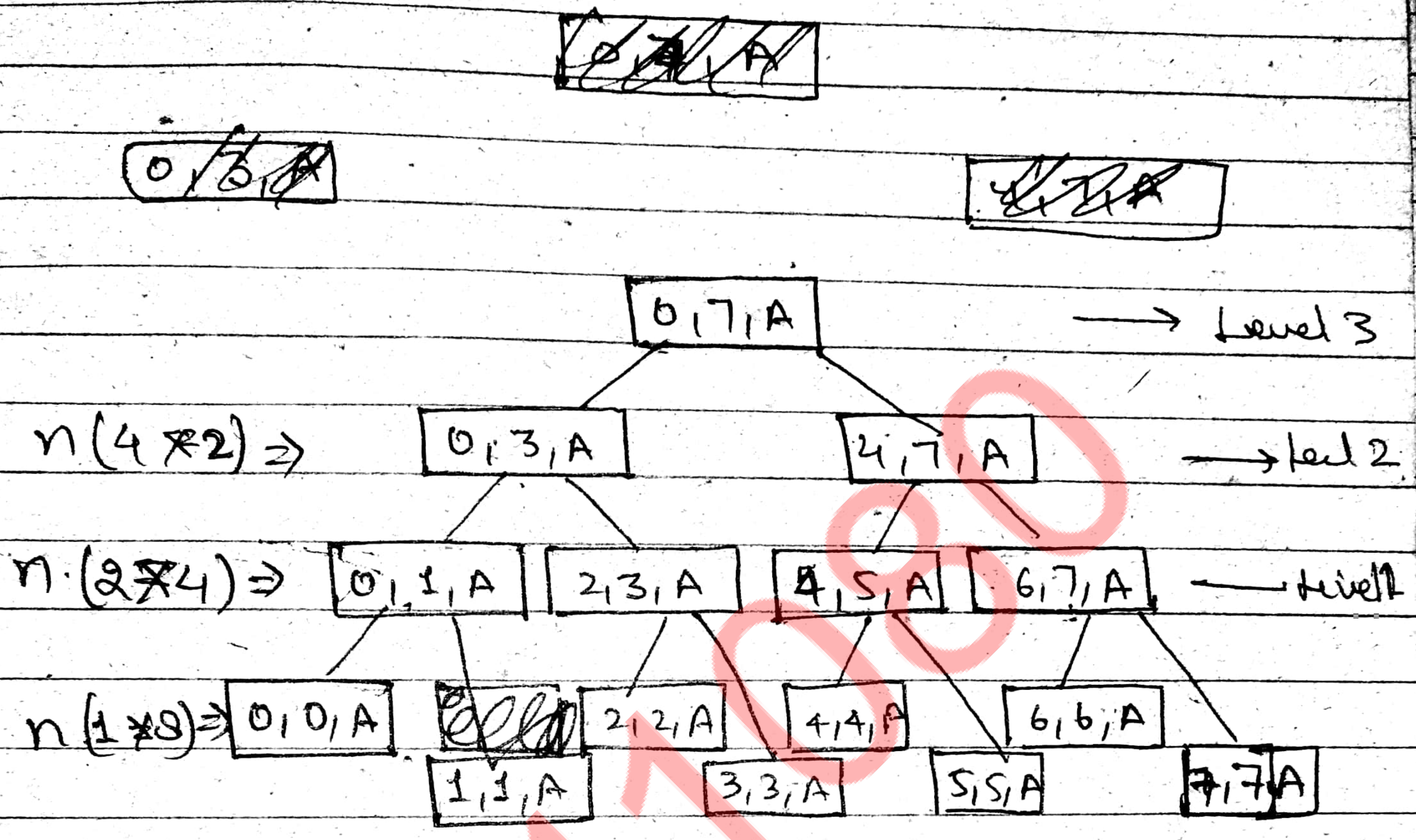
```
void merge_sort(int l, int r, int *a)
{
    if (l<r)
    {
        int m = (l+r)/2;

        mergesort(l, m, a);
        mergesort(m+1, r, a);
        merge(l, m, r, a);
    }
}

void merge(int l, int m, int r int *p)
{
    int n1 = m-l+1;
    int n2 = r-m;
```

\# array1[n1]; stores initial Data

\# array2[n2]; stores final part of Data

than p will store in array1 & array2 to get by sorting with $O(n)$

Lets consider 8 element



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

Time Complexity $[T(n)] \rightarrow n \log n$

Merge sort space Complexity = $O(n)$

* Insertion sort Algorithm:-

```
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;
}

time complexity at worst case = $O(n^2)$
best case = $O(n)$

Space Complexity = $O(1)$

* Quick Sort:-

Pivot declaration important
Pivot may be any variable from its original array.

it is used to by part array into two array ~~which~~ are in which first sub array contains all values less than pivot &

Other sub array will contain all values ~~of~~ greater than pivot

```
partition (l, m, array)   // l=0 ; m=n-1  initially
{
    int start = l
    int 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;
```