Q-1 Perform analysis on the time complexity of insertion sort algorithm in best case.

Ans- For best case, the list should in ascending order.

Program :-

```
for ( int i=1 ; i <n; i++)
{
    a = arr[i]
  for ( j = i-1 ; j >=0 ; j--).
{
    if ( a < arr[j])
{
    a[j+1] = a[j]
    a[j] = a
    }
else
  break;
} }
```

List is { 2, 6, 9, 15, 20 }

Loop 1 -

a = 6

Loop 2 - (6 < 2)

↓ False

Break

P.T.O →

for n = 2

a = 9

Loop 2 j = 1

if (9 < arr[i])

break

Loop 1	Loop 2	Cost
i = 1	j = 0	1
i = 2	j = 1	1
i = 3	j = 2	1
i = 4	j = 3	1
i = 5	j = 4	1
i = n-1	j = n-2	1

Time complexity ⇒ O(n-1)

≅ O(n).

Dipanshu Garg

Q-2 Bubble sort :-

It runs (n-1) times

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

So; total time complexity is $O((n-1)^2) \approx O(n^2)$.

Quick sort :-

```
void swap (int *a, int *b)
{
  int x = *a;
  *a = *b;
  *b = x; }
```

Dipanshu Garg

```
int part (int R[], int a, int b)
{
  int pro = R[a];
  int i = a;
  int j = b;
  do
  {
    do {i++;} while (R[i] <= pro);
    do {j--;} while (R[j] > pro);
    if (i<j) swap (&R[i]; &R[j]);
  }
void sort (int R[], int a, int b)
{
  int j;
  if (a<b)
{
```

```
b= part (R, a, b);
sort (R, A, j);
sort (R, j+1, b);
  }  }
```

Dipanshu Garg

Merge sort

```
Void Merge (int R[], int a, int mid,
                    int b).
{
int x = a; y = mid+1; z = a;
int B[100];
while (x <= mid && y <= b)
{
  if (R(x) < R(y))
    B[z++] = R[x++];
else
  B[z++] = R[y++];
}
for (; x <= mid; x++)
    B[z++] = R[x];
for (; y <= b; y++)
    B[z++] = R[y];
for (x = a; x <= b; x++)
    B[x] = R[x];
}
```

```
void sort (int R[], int n)
{
int p, q, r, mid, i;

for (p=2; p<=n; p=p*2)
{
 for (i=0; i+p-1<=n; i=i+p)
 {
   q = i;
   r = i+p-1;
  mid = (q+r)/2;
  Merge (R, q, mid, r);
   } }
 if (p/2<n)
    Merge (R, 0, p/2-1, n);
}
```

So, It's time complexity is of O(n log n) &.

Dipanshu Garg
IT
11912012