Q1 - Perform analysis on the time complexity of insertion sort algorithm in best case.

Sol^n - For best case, the list should be in ascending order.

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

   }
 }
```

Amy

given list is {4 5 7 9 11}.

Loop 1 -

temp = 5

Loop 2 - (5<4)

False

break

for i=2

temp = 7

Loop 2 - j=1

if (7 < 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 $\Rightarrow$ $O(n-1)$

$\approx O(n)$

Q2 - Bubble Sort -

```
for(i=0; i<n-1; i++)  ]--> It runs (n-1) times
{
  for(j=0; j<n-1; j++) ]--> (n-1) times
  {
    if (i[j] > i[j+1])
    {
      temp = i[j];
      i[j] = i[j+1];
      i[j+1] = temp;
    }
  }
}
```

So, total time complexity is of $\boxed{O(n^2)}$

Anuj

Quick Sort —

```
void swap (int *x, int *y)
{
  int temp = *x;
  *x = *y;
  *y = temp;
}
int partition (int A[], int l, int h)
{
  int pivoit = A[l];
  int i = l;
  int j = h;
  do
  {
    do {i++;} while (A[i] <= pivot);
    do {j--;} while (A[j] > pivoit);
    if (i<j) swap (&A[i], &A[j]);
  }
  while (i<j);
  swap (&A[l], &A[j]);
  return j;
}
void sort( int A[], int l, int h)
{
  int j;
  if (l<h)
  {
    j = partition (A,l,h);
    sort (A,l,j);
    sort (A,j+1,h);
  }
```

Anuj

3.

Merge Sort -

```
void Merge(int A[], int l, int mid, int h)
{
  int i=l, j= mid+1, k=l;
  int B[100];
  while (i <= mid && j <= h)
  {
    if (A[i] < A[j])
        B[k++] = A[i++];
    else
        B[k++] = A[j++];
  }
  for(; i <= mid ; i++)
      B[k++] = A[i];
  for(; j <= h; j++)
      B[k++] = A[j];
  for(i=l; i<=h; i++)
      A[i] = B[i];
}
void sort(int A[], int n)
{
  int p, l, h, mid, i;
  for(p=2; p<=n; p=p*2)
  {
    for(i=0; i+p-1 <= n; i=i+p)
    {
      l = i;
      h = i+p-1;
      mid = (l+h)/2;
      Merge(A, l, mid, h);
    }
  }
  if (p/2 < n)
      Merge(A, 0, p/2-1, n);
}
```

Its time complexity is of O(n log n)