UNIT-1

1

1. Q:- Explain the algorithmic steps involved in Quick sort. Discuss how it partitions elements and recursively sorts sub arrays and show the passes for quick sort for the elements 14, 2, 16, 8, 12, 10, 15

Ans:- Quick Sort: This sorting is also called as partition exchange sorting. This list is divided in to two parts based on an element called pivot element.

* The procedure of choosing a pivot and Partioning the list is applied recursively.

Algorithm:

```
        Quick Sort (A, left, right)
step 1: if left < right goto step 2 to 7 Otherwise goto step 8.
step 2: Set pivot = A[left], set i = left, j = right.
step 3: Repeat step 4 to 6 while i<j
step 4: Repeat while a[i] ≤ pivot, i++ end of while
step 5: Repeat while a[j] > pivot, j-- end of while
step 6: i<j swap a[i], a[j] end of while
step 7: i>j swap a[j], pivot
        call quicksort (a, left, j-1)
             quicksort (a, j+1, right)
step 8: Exit.
```

Passes for Quicksort of the elements 14,2,16,8,12,10,15

14	2	16	8	12	10	15
0	1	2	3	4	5	6

↳ pivot

⇒ i<j → 0<6 (true)

a[i] ≤ a[pivot]

14 ≤ 14 ✓

i++ → 0+1=1

⇒ i<j → 1<6 (true)

a[i] ≤ a[pivot]

2 ≤ 14 ✓

i++ → 1+1=2

⇒ i<j → 2<6 (true)

a[i] ≤ a[pivot]

16 ≤ 14 ✗

Comes out from loop at i=2

a[i]=16

⇒ a[j] > a[pivot]

15 > 14 ✓

j-- → 6-1=5

⇒ a[j] > a[pivot]

10 > 14 ✗

Comes out from loop at j=5

a[j]=10

⇒ i<j → 2<5 (true)

swap (a[j], a[j])

0	1	2	3	4	5	6
14	2	15	8	12	10	16

↳ pivot

⇒ i<j → 0<6 (true)

a[i] ≤ a[pivot]

14 ≤ 14 ✓

i++ → 0+1=1

→ i<j → 1<6 (true)

a[i] ≤ a[pivot]

2 ≤ 14 ✓

i++ → 1+1=2

⇒ i<j → 2<6 (true)

a[i] ≤ a[pivot]

15 ≤ 14 ✗

Comes out from loop at i=2

a[i]=15

⇒ a[j] > a[pivot]

16 > 14 ✓

j-- → 6-1=5

⇒ a[j] > a[pivot]

10 > 14 ✗

Comes out from loop at j=5

a[j] = 10

⇒ i<j

2<5 (true)

swap (a[i], a[j])

14	2	10	8	12	15	16
0	1	2	3	4	5	6

↳ pivot

⇒ i<j → 0<6 (true)

a[i] ≤ a[pivot]

14 ≤ 14 ✓

i++ → 0+1=1

⇒ $i < j \rightarrow \phi < 6$ (true)

$a[i] \le a[pivot]$

$2 \le 14$ ✓

$i{+}{+} \rightarrow 1+1=2$

⇒ $i < j \rightarrow 2 < 6$ (true)

$a[i] \le a[pivot]$

$10 \le 14$ ✓

$i{+}{+} \rightarrow 2+1=3$

⇒ $i < j \rightarrow 3 < 6$ (true)

$a[i] \le a[pivot]$

$8 \le 14$ ✓

$i{+}{+} \rightarrow 3+1=4$

⇒ $i < j \rightarrow 4 < 6$ (true)

$a[i] \le a[pivot]$

$12 \le 14$ ✓

$i{+}{+} \rightarrow 4+1=5$

⇒ $i < j \rightarrow 5 < 6$ (true)

$15 \le 14$ ✗

Comes out from loop at $i=5$

$a[i]=15$

⇒ $a[j] > a[pivot]$

$16 > 14$ (true)

$j{-}{-} \rightarrow 6-1=5$

⇒ $a[j] > a[pivot]$

$15 > 14$ ✓

$j{-}{-} \rightarrow 5-1=4$

⇒ $a[j] > a[pivot]$

$12 > 14$ ✗

Comes out from loop at $j=4$

$\Rightarrow i<j$

$5<4$ false

swap (a[j], pivot]

12	2	10	8	14	15	16

partition - ① pivot partition - ②

partition - ①

→ pivot

12	2	10	8
0	1	2	3

$\Rightarrow i<j \rightarrow 0<3$ (true)

$a[i] \leq a[pivot]$

$12 \leq 12$ ✓

$i{+}{+} \rightarrow 0+1=1$

$\Rightarrow i<j \rightarrow 1<3$ (true)

$a[i] \leq a[pivot]$

$2 \leq 14$ ✓

$i{+}{+} \rightarrow 1+1=2$

$\Rightarrow i<j \rightarrow 3<3$ (false)

Comes out from loop at $i=3$

$\Rightarrow a[j] > a[pivot]$

$8 > 12$ ✗

Comes out from loop at $j=0$

$\Rightarrow i<j \rightarrow 3<3$ (false)

swap (a[j], pivot)

8	2	10	12

partition - ③ → pivot

partition - 3

0	1	2
8	2	10

↳ pivot

⇒ $i < j \rightarrow 0 < 2$ ✓

$a[i] \le a[pivot]$

$8 \le 8$ ✓

$i{+}{+} \rightarrow 0+1 \Rightarrow 1$

⇒ $i < j \rightarrow 1 < 2$ ✓

$a[i] \le a[pivot]$

$2 \le 8$ ✓

$i{+}{+} \rightarrow 1+1 = 2$

⇒ $i < j \rightarrow 2 < 2$ ✗

comes out from loop at $i = 2$

⇒ $a[j] > pivot$

$10 > 8$

$j{-}{-} \rightarrow 2-1 = 1$

⇒ $a[j] > pivot$

$2 > 8$ ✗

comes out from loop at $j = 1$

⇒ $i < j \rightarrow 2 < 1$ ✗

Swap (a[j], pivot)

2	8	10

partititon -2

15	16
0	1

↳ pivot

$\Rightarrow i < j \rightarrow 0 < 1$ ✓

$a[i] \leq a[pivot]$

$15 \leq 15$ ✓

$i{+}{+} \rightarrow 0+1=1$

$\Rightarrow i < j \rightarrow 1 < 1$ ✗

Comes out from loop at $i=1$

$\Rightarrow a[j] > pivot$

$16 > 15$ ✓

$j{-}{-} \rightarrow 1-1=0$

$\Rightarrow a[j] > pivot$

$15 > 15$ ✗

Comes out from loop at $j=0$

$\Rightarrow i < j$

$1 < 0$ (false)

swap (a[j], pivot)

15	16

Final Sorted Array is

2	8	10	12	14	15	16

Algorithm :

2Q) Merge (A, low, mid, high)

```
Step 1: intialize i= low= mid +1, k=0
step 2: repeat while (i<=mid) and (j<=high)
          if (a[i]<a[j])
          set temp[k++] = a[i++]
          else
          set temp[k++] = a[j++]
        [end of if]
        [end of for]
Step 3: Repeat while j<= high
          set temp[k++] = a[i++]
          [end of loop]
          repeat while i<=mid
          set temp[k++] = a[i++]
          [end of loop]
step 4: [Copy the contents of temp back to a]
        repeat for k= low to high
            set a[k] = temp[k]
          [end of loop]
step 5: exit.
```

Process :

* Consider the intial list and divide the list into two sublists.

* Again these sub lists are divided into many number of sublists until each and every sub list contains single element.

* Combine these sub lists into sorted order. Finally we will get list of elements in sorted order.

Demonstrate eachstep of mergesort technique with the following initial assay of elements. 25, 16, 35, 22, 18, 34, 81, 47

25	16	35	22	18	34	81	47
0	1	2	3	4	5	6	7

→ array a

mid

16	22	25	35
0	1	2	3

18	34	47	81
4	5	6	7

i = 0

j = mid + 1

k = 0

array b ←

0	1	2	3	4	5	6	7

⇒ i ≤ mid && j ≤ high

0 ≤ 3 ✓ → 4 ≤ 7 ✓

a[i] ≤ a[j]

16 ≤ 18 ✓

b[k] = a[i] = 16

i++ → 0+1 = 1

k++ → 0+1 = 1

⇒ i ≤ mid && j ≤ high

1 ≤ 3 ✓ → 4 ≤ 7 ✓

a[i] ≤ a[j]

22 ≤ 18 ✗

b[k] = a[j] = 18

j++ → 4+1 = 5

k++ → 1+1 = 2

⇒ i ≤ mid && j ≤ high

1 ≤ 3 ✓ → 5 ≤ 7 ✓

a[i] ≤ a[j]

22 ≤ 34 ✓

b[k] = a[i] = 22

i++ → 1+1=2

k++ → 2+1=3

⇒ i ≤ mid && j ≤ high

2 ≤ 3 ✓ → 5 ≤ 7 ✓

a[i] ≤ a[j]

: 25 ≤ 34 ✓

b[k] = a[i] = 25

i++ → 2+1=3

k++ → 3+1 = 4

⇒ i ≤ mid && j ≤ high

3 ≤ 3 ✓ → 5 ≤ 7 ✓

a[j] ≤ a[j]

35 ≤ 34 ×

b[k] = a[j] = 34

j++ → 5+1=6

k++ → 4+1 = 5

⇒ i ≤ mid && j ≤ high

3 ≤ 3 ✓ → 6 ≤ 7

a[i] ≤ a[j]

35 ≤ 47 ✓

b[k] = a[j] = 35

i++ → 3+1=4

k++ → 5+1=6

→ $i \le mid$ && $j \le high$

$4 \le 3$ ✗

$i \le mid$

$4 \le 3$ ✗

$j \le high$

$6 \le 7$ ✓

$b[k] = a[j] = 47$

j++ → 6+1 = 7

k++ → 6+1 = 7

$j \le high$

$7 \le 7$ ✓

$b[k] = a[j] = 81$

j++ → 7+1 = 8

k++ → 7+1 = 8

16	18	22	25	34	35	47	81

3Q) For 3rd Question refer Question ①

4Q) For 4th Question refer Question ②

Merge Sort : 4Q) continue

merge sort is to divide the list in to two sublists.

* This sorting technique also uses Divide & Conquer paradigm.
* It separates the list into two halves and then sorts the two data sets recursively.
* finally these are merged to obtain the complete sorted list.

Algorithm :

5Q) insertion Sort(A,N)

```
Step 1: repeat Step 2 to 5 for i=1 to N
step 2: set index = A[i]
step 3: set j=i-1
step 4: repeat while j>0 && a[j-1] > index
        set a[j]= a[j-1].
        set j-- [End while loop]
step 5 : a[j]= index.
        [end of for loop]
step 6 : Exit
```

Program :

```
# include <stdio.h>
int getMax (int a[ ], int n);
void radixsort(int a[], int n);
Void Countsort (int a[], int n, int Pos);
int main()
{
 int a[10], i,n,K;
 Printf("\n Enter the size of array");
 Scanf("%d", &n);
 Printf("\n Enter the elements in to array");
 for(i=0; i<n; i++)
 {
  scanf(" %d", &a[i]);
 }
 K= getMax(a,n);
 radixesort(a,n);
 Printf("\n sorted array:");
 for(i=0; i<n; i++)
 {
 Printf("%d\t", a[i]);
 }
 return 0;
}
```

```
int getMax(int a[], int n)
{
 int max, i;
 max = a[0];
 for(i=1; i<n; i++)
 {
  if (a[i] > max)
   max = a[i];
  }
 return max;
}
Void radixsort(int a[ ], int n)
{
 int max, pos;
 max = getMax(a,n);
 for(pos=1; (max/pos) > 0; pos *=10)
 {
   Countsort(a,n,pos);
  }
}
Void Countsort(int a[ ], int n, int Pos)
 {
 int i, b[10], Count[10] = {0};
for(i=0; i<n; i++)
{
 Count[a[i]/Pos) % 10]++ ;
}
for(i=1; i<10; i++)
{
 Count[i] = Count[i] + count[i-1];
}
for(i= n-1; i>=0; i--)
{
 b[-- count[(a[i]/Pos) % 10]] = a[i];
}
for(i=0; i<n; i++)
{
 a[i]=b[i];
}
}
```

Algorithm : 6Q)

```
Selectionsort(A,N)
Step1: repeat step2 to5 for i=0 to N-1
step 2: set min=i
step 3: set j=i
step 4: repeat for i+1 to N
  if a[min] > a[j] set min=j
    end inner loop.
step 5: swap a[i] and a[min] (end for loop)
step 6: stop.
```

Program :

```
#include <stdio.h>
Void mergesort(int a[], int low, int high);
Void merge(int a[], int low, int mid, int high);
int main()
{
 int a[10], i, n, low, high
 Printf("\n Enter the size of an array");
 scanf("%d", &n);
 for(i=0; i<n; i++)
 {
  scanf("%d", &a[i]);
 }
 Printf("\n unsorted array of elements");
 for(i=0; i<n; i++)
 {
 printf("%d\t", a[i]);
 }
 low=0;
 high=n-1;
 mergesort(a, low, high);
 printf("\n Sorted elements in an array");
 for(i=0; i<n; i++)
 {
 printf("%d\t", a[i]);
 }
return 0;
}
void mergesort(int a[], int low, int high)
{
 if(low < high)
 {
  int   mid=(low+high)/2 ;
  mergesort (a, low, mid);
  mergesort ( a, mid+1, high);
  merge (a, low, mid, high);
 }
}
Void merge(int a[], int low, int mid, int high)
{
 int  i,j,k
 int  n1 = mid-low+1;
 int  n2 = high-mid;
 int  l[n1], r[n2];
```

```
for(i=0; i<n1; i++)
{
 l[i] = a[low+i];
}
for (j=0; j<n2; j++)
{
  r[j]= a[mid+1+j];
}
i=0;
j=0;
K=0; // k= low;
while (i<n1 && j<n2)
{
  if (.l[i] <= r[j])
  {
   a[K] = l[i];
   i++;
  }
  else
  {
  a[k] = r[j];
  j++;
  }
} k++;
```

```
    while (i<n1)
    {
      a[k]= l[i] ;
      i++ ;
      k++ ;
    }
  while (j<n2)
  {
  a[k] = r[j] ;
  j++;
  k++;
    }
}
```