Topic IV / 4.13 *** Quick sort ***

Unit No. : FDS 4th

- Most popular sorting Techniques.
- It possesses a very good average case behavior among all the sorting techniques.
- Developed by C.A.R Hoare.

Logic :-

This algorithm works by partitioning the array to be sorted. Then each partition is recursively sorted.

In partition, one of array element is chosen as a key i.e. pivot

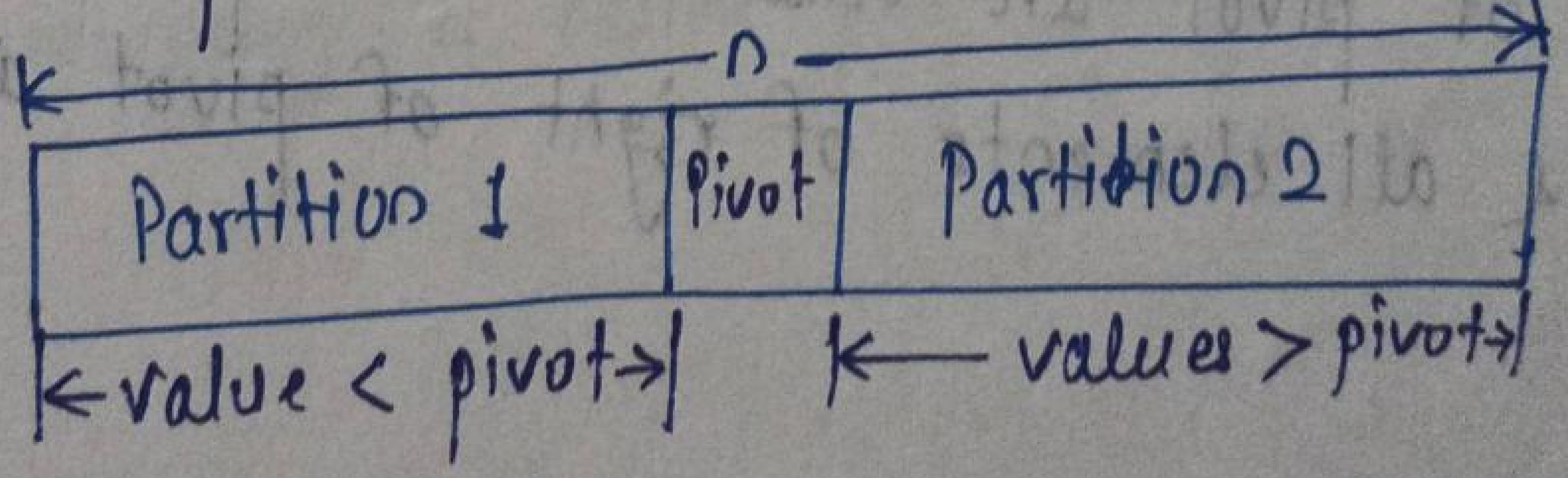
The pivot element is 1st element in array.

ie pivot = a[0] OR can be last element in array

& Rest of array elements are grouped into two partitions

(a) One partition contains elements < Pivot element

(b) Another partition contains elements > pivot element



Example

* Implementation logic / working :-

① select the pivot element

② Take two indices, low and high
low ~~as~~ indicate element (pivot + 1)
high ⇒ Indicates last element.

③ Increment low (start on left) until select element greater than pivot element.

④ Decrement high (start on right) until select element smaller than pivot element

⑤ Then these element get interchanged.

⑥ This process is repeated until all ements to left of pivot are smaller than pivot,
& all elements of right of pivot are greater than pivot.

Topic :

Unit No. :

In Quicksort, the division into two subarrays is made so that sorted subarray donot need to merge.

ie Rearrang $a[1:n]$ such that $a[i] <= a[j]$ where i ~~is~~ is bet^n 1 and 'm' ie $1 \leq i \leq m$

j bet^n $m+1$ & n $\Rightarrow 1 \leq m \leq n$

Thus, element $a[i:m]$ and $a[m+1:n]$ are sorted independently.

ie Rearranging of element is accomplished by picking some element & then reordering other elements such that

<	t	<

ie all elements appear before 't' are less than equal & all elements apper after t are greater than to 't'.

This rearranging is called partitioning.

Algorithm Partition (int a[ ], int m, int p)
{
// within a[m], a[m+1] ------ a[p-1] elements are rearranged in such a manner that if initially, v = a[m] then after completion
a[q] == v for some 'q' betn 'm' and 'p-1'
a[k] <= v for m <= k < q
& a[k] >= v for q < k < p. return 'q'

```
V = a[m],
i = m
j = p
do {
      do{
           i++;
      }while (a[i]<v);
      do
      {   j--
      }while(a[j]>v)
      if (i<j) then swap a[i] & a[j]
   }while (i<j)
a[m] = a[j];
a[j] = V; return (j)
}
```

```
Void Quicksort (int a[], int p, int q)
{
// sort the element a[p] ---- a[q] which resides in arra
into ascending order

    if (p < q)
    {
// if there are more than one element
// divide list into two sublist
    int j = partition (a, p, q+1)
// j is position of partitioning element

// solve 1st sublist
    quicksort (a, p, j-1);

// solve 2nd sublist
    Quicksort (a, j+1, q);
    }
}
```

Pass-I $\boxed{\text{Pivot}=65}$

										l	h
$\boxed{65}$	70	75	80	85	60	55	50	45		2	9

$\boxed{65}$ 45 75 80 85 60 55 50 70 — l=3, h=8

$\boxed{65}$ 45 50 80 85 60 55 75 70 — l=4, h=7

l

$\boxed{65}$ 45 50 55 85 60 80 75 70 — l=5, h=6

65 45 50 55 60 85 80 75 70 — l=6, h=5

h

60 45 50 55 $\boxed{65}$ 85 80 75 70

Pass-II
pivot = 60.

l=4, h=4

$\boxed{60}$ 45 50 55

l

$\boxed{55}$ 45 50 $\boxed{60}$

Pass-III
pivot = 55

$\boxed{55}$ 45 50 — l=2, h=3

l

~~50 50 45~~ $\boxed{50}$ 45 55 — l=3 h=2

h l

pivot = 50 50 — 45 50 55

Topic : Unit No. :

45 50 55 60 65 85 80 75 70

Pivot = 85

$\boxed{85}$ 80 75 70

↑ l → ↑ l, h

~~85 70 75 80~~

70 80 75 $\boxed{85}$

Pivot = 70

$\boxed{70}$ 80 75

↑ ← ↑

Pivot = $\boxed{80}$ 75

∴ sorted array is

~~45, 50, 55, 60, 65, 85, 80, 70,~~

45, 50, 55, 60, 65, 70, 75, 80, 85 //

Topic

Unit No. :

It is divide and conquer method/approach used.

// sorting by the partitioning //

Algorithm Quicksort (int A[10], int p, int q)

{

// sort the element a[p] .... a[q] which reside in the array a[1....n] into ascending order.

if (p<q) // If .

{

// divide the array into two parts.

j = partition(a, p, q+1)

// j is position of partitioning elements.

// solve the subproblem.

Quicksort (a, p, j-1);

Quicksort (a, j+1, q);

// there is no need for combining solutions.

}

}

Topic

Unit No. :

Algorithm Partition (int a[10], int L, int H)
{
// within a[L], a[L+1] .... a[H-1] the elements are rearranged in such manner that

V = a[L] or V = a[L]
low = L
high = H+1
repeat
{
repeat
{
low = low+1;
until (a[low] ≥ V) and Low ≤ H);
repeat
high = high-1;
until (a[high] ≤ V);
if (low < high)
then Inchange (a, i, j)
ie a[low] and a[high]
} until (low ≥ high)
a[L] = a[high]
a[high] = V
return (j)
}

Topic

Unit No. :

Examples.

① 56 -90 12 632 457 1000 -1 8 34 0

Ans:

Passes

Pass.I Pivot ~~22~~ 56

II Pivot ~~18~~ -1

III Pivot ~~26~~ 12

IV Pivot ~~45~~ 8

V Pivot ~~27~~ 1000

VI Pivot ~~56~~ 632

PASS VII