**Experiment No 6:**

**A program for Quick Sort**

**Aim:** Write a program to implement Quick Sort.

**Theory:**

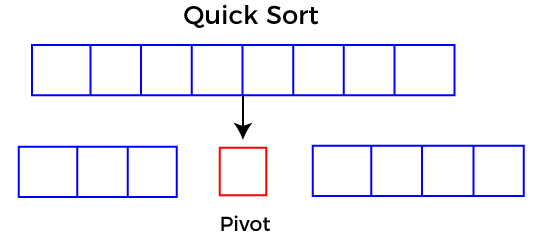
* Quicksort is the widely used sorting algorithm that makes n log n comparisons in average case for sorting an array of n elements. It is a faster and highly efficient sorting algorithm.
* This algorithm follows the divide and conquer approach.
* Divide and conquer is a technique of breaking down the algorithms into sub problems, then solving the sub problems, and combining the results back together to solve the original problem.

**Divide:** In Divide, first pick a pivot element. After that, partition or rearrange the array into two sub-arrays such that each element in the left sub-array is less than or equal to the pivot element and each element in the right sub-array is larger than the pivot element.

**Conquer:** Recursively, sort two subarrays with Quicksort.

**Combine:** Combine the already sorted array.

* Quicksort picks an element as pivot, and then it partitions the given array around the picked pivot element.
* In quick sort, a large array is divided into two arrays in which one holds values that are smaller than the specified value (Pivot), and another array holds the values that are greater than the pivot.
* After that, left and right sub-arrays are also partitioned using the same approach. It will continue until the single element remains in the sub-array.



**\*Any Solved Example with all passes**

**Algorithm:**

Algorithm Insertion sort:

**QUICK(A,N,BEG,END,LOC)**

Here A is an array with N elements. Parameters BEG and END contain the boundary values of the sublist of A to which this procedure applies. LOC keeps track of the position of the first element A[BEG] of the sublist during the procedure. The local variables LEFT and RIGHT will contain the boundary values of the list of elements that have not be scanned.

1. [Initialize.] Set LEFT := BEG, RIGHT := END and LOC := BEG.

2. [Scan from right to left.]

(a) Repeat while A[LOC] ≤ A[RIGHT] and LOC ≠ RIGHT:

RIGHT := RIGHT – 1.

[End of loop.]

(b)If LOC=Right, then: Return.

(c)If A[LOC]>A[RIGHT],then:

(i)[Interchange A[LOC] and A[RIGHT].]

TEMP:=A[LOC],A[LOC]:=A[RIGHT]

A[RIGHT]:=TEMP.

(ii)Set LOC:=RIGHT.

(iii)Go to step 3.

[End of If structure.]

3. [Scan from left to Right.]

(a) Repeat while A[LEFT] ≤ A[LOC] and LEFT ≠ LOC:

LEFTT := LEFT + 1.

[End of loop.]

(b)If LOC=LEFT, then: Return.

(c)If A[LEFT]>A[LOC],then:

(i)[Interchange A[LEFT] and A[LOC].]

TEMP:=A[LOC],A[LOC]:=A[LEFT]

A[LEFTT]:=TEMP.

(ii)Set LOC:=RIGHT.

(iii)Go to step 2.

[End of If structure.]

**(Quicksort) This algorithm sorts an array A with N elements.**

1.[Initialize.]TOP:=NULL.

2.[Push boundary values of A onto stacks when A has 2 or more elements.]

If N>1, then: TOP:=TOP+1, LOWER[1]:=1, UPPER[1]:=N.

3.Repeat Steps 4 to 7 while TOP≠NULL.

4. [Pop sublist from Stacks.]

Set BEG:=LOWER[TOP], END:=UPPER[TOP],

TOP=TOP-1.

5. Call QUICK(A,N,BEG,END,LOC) [procedure a]

6.[Push left sublist onto stacks when it has 2 or more elements.]

If BEG<LOC-1, Then:

TOP:=TOP+1,LOWER[TOP]:=BEG,

UPPER[TOP]=loc-1.

[End of If structure.]

7.[Push Right sublist onto stacks when it has 2 or more elements.]

If LOC+1<END, then:

TOP:=TOP+1,LOWER[TOP]:=LOC+1,

UPPER[TOP]=END.

[End of If structure.]

[End of Step 3 loop.]

8.Exit

**PROGRAM: [Write program for Quick sort]**

**OUTPUT**

**CONCLUSION:**