INDEX

Ex. No.: 11

Date :

Implementation of Quick Sort

Aim: To write a C-program to implement Quick Sort.

Quick Sort:

QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

- Always pick first element as pivot.
- Always pick last element as pivot
- Pick a random element as pivot.
- Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x.

Algorithm

```
To sort a[left...right]:

1. if left < right:

    1.1. Partition a[left...right] such that:

            all a[left...p-1] are less than a[p], and

            all a[p+1...right] are >= a[p]

    1.2. Quicksort a[left...p-1]

    1.3. Quicksort a[p+1...right]

2. Terminate
```

Best and Average Case Time Complexity: O(nlogn)
Worst Case Time Complexity: O(n^2)
Space Complexity: O(n) total, O(1) auxiliary
Sorting In Place: Yes
Stable: No
Adaptive: Yes
Online: No

Program

```
/*This program demonstrate Quick Sort */

#include<stdio.h>
#include<conio.h>

void quickSort(int [10],int,int);

void main(){
  int list[20],size,i;

  printf("Enter size of the list: ");
  scanf("%d",&size);

  printf("Enter %d integer values: ",size);
  for(i = 0; i < size; i++)
    scanf("%d",&list[i]);

  quickSort(list,0,size-1);

  printf("List after sorting is: ");
  for(i = 0; i < size; i++)
    printf(" %d",list[i]);

  getch();
}

void quickSort(int list[10],int first,int last){
    int pivot,i,j,temp;

     if(first < last){
         pivot = first;
         i = first;
         j = last;

         while(i < j){
             while(list[i] <= list[pivot] && i < last)
                 i++;
             while(list[j] > list[pivot])
                 j--;
             if(i < j){
                  temp = list[i];
                  list[i] = list[j];
                  list[j] = temp;
             }
         }

         temp = list[pivot];
         list[pivot] = list[j];
         list[j] = temp;
         quickSort(list,first,j-1);
         quickSort(list,j+1,last);

    }
}
```