Experiment Number: 8

Title: Implementation of Quick Sort.

Problem Statement: Write a C++ program to arrange the given set of numbers in ascending order using Quick Sort.

Algorithm:

1. Create the array of size n which is given by the user.
2. Read the array elements from the user.
3. Choose the zero index value as pivot.
4. Take two variables to point left and right of the array excluding pivot.
5. Left points to the low index.
6. Right points to the high.
7. While value at left is less than pivot move right.
8. While value at right is greater than pivot move left.
9. If both step 5 and step 6 does not match swap left and right.
10. If left ≥ right, the point where they met is new pivot.
11. Repeat the same procedure till all the elements are sorted.

Code:

#include <iostream>

using namespace std;

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

int partition(int[],int,int);

int main()

{

int a[50],n,i;

cout<<"How many elements to be sorted?:";

cin>>n;

cout<<"\nEnter the elements :";

for(i=0;i<n;i++) //To read the array elements

cin>>a[i];

quickSort(a,0,n-1);

cout<<"\nArray after sorting : ";

for(i=0;i<n;i++) //To print the sorted array

cout<<" "<<a[i]<<" ";

return 0;

}

void quickSort(int a[],int low,int up)

{

int j,i;

if(low<up)

{

j=partition(a,low,up);

// The following 3 blocks of code is to observe the partitioning

// and intermediate results

cout<<"\n";

for(i=low;i<=j-1;i++) //To print the array

cout<<" "<<a[i]<<" ";

cout<<"\n";

cout<<"\n";

for(i=j+1;i<=up;i++) //To print the array

cout<<" "<<a[i]<<" ";

cout<<"\n";

cout<<"\n";

for(i=0;i<=7;i++) //To print the array

cout<<" "<<a[i]<<" ";

cout<<"\n";

quickSort(a,low,j-1);

quickSort(a,j+1,up);

}

}

int partition(int a[],int low,int up)

{

int piv,i,j,temp;

piv=a[low]; //piv is the element whose final position is sought

i=low;

j=up+1;

cout<< "\n piv="<<piv<<"\n";

do

{

do

i++; // move up the array

while(a[i]< piv && i<=up);

do

j--; // move dpwn the array

while(piv<a[j]);

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}while (i<j);

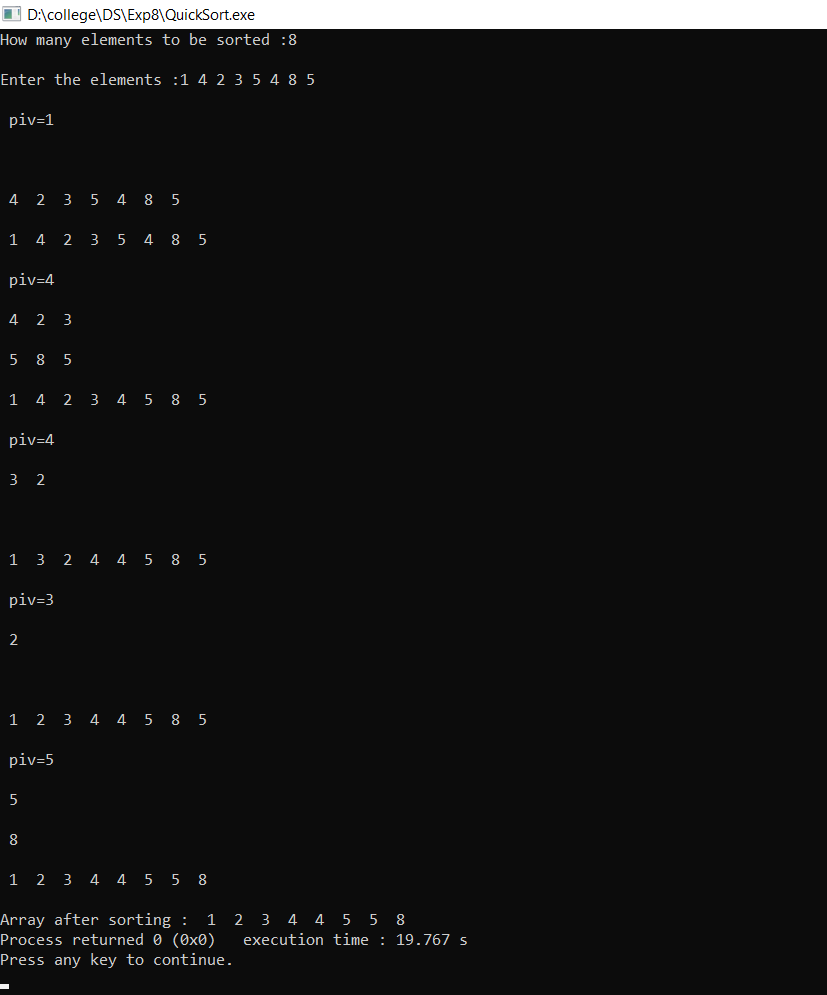
a[low]=a[j];

a[j]=piv;

return(j);

}

Sample Input/Output:



Analysis:

In this experiment we were able to execute a recursive approach to perform quick sort algorithm but quick sort has a following limitations,

1. Recursion takes a lot of stack space, usually not considerable when the program is small and running on a PC.
2. Recursion uses more processor time.
3. Recursive solution is always logical and it is very difficult to trace (debug and understand).
4. Calculating the pivot is the challenging task.
5. It requires quadratic time in worst cases.
6. It is fragile, i.e. a simple mistake in the implementation can go unnoticed and cause it to perform badly.