**Experiment No. 8**

**Title :** Implementation of Quick sort in C++

**Problem Statement :** Implementing Quick sort algorithm in C++

**Algorithm :**

**S1 :** Start

**S2 :** Declare an array and loop control variables and quicksort(), partition() functions.

**S3 :** Ask for the array input from user that has to be sorted in quick sort

**S4 :** send the array to quick sort function along with first and last index.

**S5 :** In partition function move up the array and down the array using do-while loop and compare the elements and swap.

**S6 :** In quick sort function print the array from low to j-1 and j+1 to up and call quick sort function recursively with arguments of array, low,j-1 and array,j+1,up we get sorted array

**S7 :** Display the sorted array.

**S8 :** Stop

**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 down the array

while(piv<a[j]);

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}while (i<j);

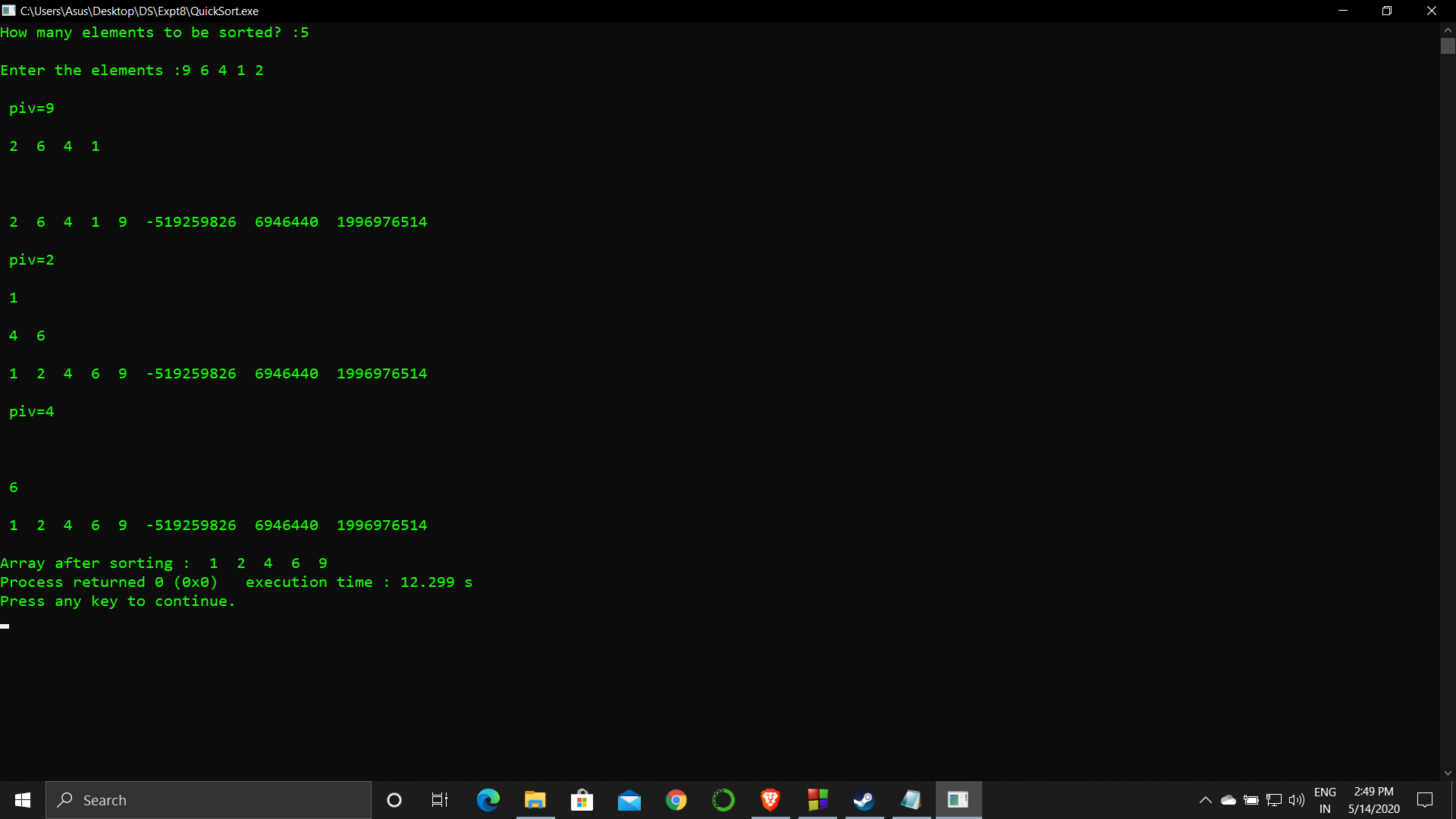
a[low]=a[j];

a[j]=piv;

return(j);

}

**Output :**

****

**Analysis :**

This sorting method is very expensive for more than average length arrays. This sorting method is highly repetitive and is more costly as compare to bubble sort.

This sorting algorithm is highly unstable and has a time complexity of O(n^2) which decreases speed.