AIM:-

Assignment 5

Implement Quick Sort to sort the given list of numbers. Display corresponding list in each pass.

OBJECTIVE:-

To study the quickest sorting technique .To know it’s implementation.

THEORY:-

Quick Sort 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 Quick Sort that pick pivot in different ways.

1. Always pick first element as pivot.

2. Always pick last element as pivot (implemented below)

3. Pick a random element as pivot.

4. Pick median as pivot.

The key process in Quick Sort 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. All this should be done in linear time.

ALGORITHM:-

Pivot Algorithm:

Step 1 − Choose the highest index value has pivot

Step 2 − Take two variables to point left and right of the list excluding pivot

Step 3 – j points to the low index

Step 4 − i points to the high

Step 5 − while value at left is less than pivot move i

Step 6 − while value at right is greater than pivot move j

Step 7 − if both step 5 and step 6 does not match swap j and i

Step 8 − if j ≥ i, the point where they met is new pivot

Quick sort main Algorithm :-

Step 1 − Make the right-most index value pivot

Step 2 − partition the array using pivot value

Step 3 − quicksort left partition recursively

Step 4 − quicksort right partition recursively

SOURCE CODE:-

#include<iostream>

void swap(int\*a,int\*b)

{

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

}

int partition(int array[],int p,int r)

{

int i=p-1;

int x=array[r];

for(int j=p;j<=r-1;j++)

{

if(array[j]<x)

{

i++;

swap(&array[i],&array[j]);

}

}

swap(&array[i+1],&array[r]);

return (i+1);

}

void quicksort(int array[],int p,int r,int n)

{

if(p<r)

{

int q=partition(array,p,r);

for(int i=0;i<n;i++)

{

std::cout<<array[i]<<"\t";

}

std::cout<<"\n";

quicksort(array,p,q-1,n);

quicksort(array,q+1,r,n);

}

}

int main()

{

using namespace std;

int n,start,end;

cout<<"Enter no of elements: ";

cin>>n;

int array[100];

cout<<"\nEnter elements: \n";

for(int i=0;i<n;i++)

{

cin>>array[i];

}

cout<<"\nElements before sorting:\n";

for(int i=0;i<n;i++)

{

cout<<array[i]<<"\t";

}

std::cout<<"\n";

start=0;

end=n-1;

quicksort(array,start,end,n);

std::cout<<"\nElements after sorting:\n";

for(int i=0;i<n;i++)

{

std::cout<<array[i]<<"\t";

}

return 0;

}

OUTPUT:-

Enter no of elements: 5

Enter elements:

4

3

6

2

1

Elements before sorting:

4 3 6 2 1

1 3 6 2 4

1 3 2 4 6

1 2 3 4 6

Elements after sorting:

1 2 3 4 6

--------------------------------

CONCLUSION:-

1. It is the quickest sorting algorithm.
2. With time complexity is O(n log(n)).