**Experiment-2**

**Aim-** **⁠**Sort a given set of elements using Quick sort method and Merge Sort method, and determine the time taken to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.

**Theory-**

1. Quick Sort:
   * Quick Sort is a divide-and-conquer algorithm. It selects a pivot element from the array and partitions the other elements into two sub-arrays, according to whether they are less than or greater than the pivot.
   * Quick Sort is typically faster in practice compared to other sorting algorithms, especially for large datasets.
2. Merge Sort:
   * Merge Sort also uses a divide-and-conquer strategy. It recursively divides the array into two halves, sorts each half, and then merges them back together.
   * It is a stable sort and works well on large datasets, but it uses more memory due to the need for temporary arrays during the merge process.

**Software Used –** Visual Studio Code

**Code-**

**Merge Sort**

#include<iostream>

#include<vector>

using namespace std;

void merge(int arr[],int si,int mid, int ei){

vector<int> temp;

int i=si;

int j=mid+1;

while(i<=mid && j<=ei){

if(arr[i]<=arr[j]){

temp.push\_back(arr[i++]);

}else{

temp.push\_back(arr[j++]);

}

}

while(i<=mid) temp.push\_back(arr[i++]);

while(j<=ei) temp.push\_back(arr[j++]);

for(int idx=si,x=0;idx<=ei;idx++){

arr[idx]=temp[x++];

}

}

void mergeSort(int arr[],int si, int ei){

if(si>=ei){

return ;

}

int mid = si +(ei-si)/2;

mergeSort(arr,si,mid);//left half

mergeSort(arr,mid+1,ei);//right half

merge(arr,si,mid,ei);

}

void Print(int arr[],int n){

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

int main() {

// code here

cout<<"Enter the no of elements in the array"<<endl;

int n; cin>>n;

int arr[n];

cout<<"Enter the elements"<<endl;

for(int i=0;i<n;i++) cin>>arr[i];

mergeSort(arr,0,n-1);

Print(arr,n);

return 0;

}

**Quick Sort**

#include<iostream>

using namespace std;

int partition(int arr[],int si,int ei){

int i=si-1,pivot=arr[ei];

for(int j=si;j<ei;j++){

if(arr[j]<=pivot)

{

i++;

swap(arr[i],arr[j]);

}

}

i++;

swap(arr[i],arr[ei]);

return i;

}

void QiuckSort(int arr[],int si,int ei){

if(si>=ei) return ;

int partition\_idx=partition(arr,si,ei);

QiuckSort(arr,si,partition\_idx-1);

QiuckSort(arr,partition\_idx+1,ei);

}

void Print(int arr[],int n){

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

int main() {

// code here

cout<<"Enter the no of elements in the array"<<endl;

int n; cin>>n;

int arr[n];

cout<<"Enter the elements"<<endl;

for(int i=0;i<n;i++) cin>>arr[i];

QiuckSort(arr,0,n-1);

Print(arr,n);

return 0;

}

**Output-**





