BCSL404 : Analysis and Design of Algorithms Lab

Experiment No: 11

Design and implement C/C++ Program to sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>

void printArray( long int A[],int n)
 {
 for(int i=0;i<n;i++)
 printf("%d \t", A[i]);
 }

void Merge(long int a[], long int b[],long int c[], int n1,int n2)
{
    int i=0, j=0, k=0;
    int p=n1;
    int q=n2;
    while(i<p && j<q)
    {
      if(b[i]<=c[j])
        a[k++]=b[i++];
       else
        a[k++]=c[j++];
    }
    while(i<p) a[k++]=b[i++];
    while(j<q) a[k++]=c[j++];
}

void MergeSort(long int a[], int n)
{
int i,n1=0, n2=0;
long int b[n],c[n];
if(n>1)
{
//---------------------copy left half into array B---------------
for(i=0;i<(n/2);i++)
{
b[i]=a[i];
n1++;
}
//-----------------------copy right half into array C------------
for(i=n/2;i<n;i++)
{
c[i-(n/2)]=a[i];
n2++;
}
MergeSort(b,n1);
MergeSort(c,n2);
Merge(a,b,c, n1,n2);
}
}
```

```
int main()
{
    int i,n;
    double tim[10];
    printf("Merge Sort Demo :\n ");
    printf("Enter the array size :  ");
    scanf("%d",&n);

    long int a[n];

    printf("Enter the array elements:");
    for(i=0;i<n;i++)
     scanf("%d",&a[i]);

    printf("\n\n Before sorting : \n ");
    printArray(a, n);
    MergeSort(a,n);
    printf("\n\n After sorting : \n ");
    printArray(a, n);
    printf( "\n\n------Merge Sort Demo for  N > 5000 -------------");
    printf( "\n Value of N   \t   Time (in Milli Seconds)");
    printf( "\n----------------------------------------------------------------\n");

     long int N=6000;

    for(int count=0;count<10;count++)
      {
        long int a[N];
     //Generate n random numbers and store them in array a
      for (int i = 1; i <= N; i++)
         {
          a[i] = rand() % N+1;
               }

     // using clock_t to store time
      clock_t start, end;

      start = clock();
      MergeSort(a, N);
      end = clock();

      tim[count] = ((double)(end - start));
      printf("%li \t\t\t %li\n",  N, (long int)tim[count]);
      N += 1000;
    }
  printf( "\n----------------------------------------------------------------\n");
   return 0;
}
```

Output:

*** Show demo for small values of N
*** Demonstrate for N > 5000
*** Plot the graph