Student Name:- Chaudhary Hamdan

Student Roll No.:- 1905387

Algorithm Lab. Class Assignment-7

CSE Group 1

Date: - 27th August 2021

1. **Write a program to sort a given set of elements using the Merge sort method and determine the time required 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. The elements can be read from a file or can be generated using the random number generator.**

Program

```
// Author: Chaudhary Hamdan

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

#define sf(x)            scanf("%d", &x)
#define pf               printf
#define pfs(x)           printf("%d ", x)
#define pfn(x)           printf("%d\n", x)
#define pfc(x)           printf("%d, ", x)
#define F(i,x,y)         for(int i = x; i < y; i++)
#define Fl(i,x,y,inc)     for(int i = x; i < y; i += inc)
#define RF(i,x,y)         for(int i = x; i >= y; i--)
#define pfa(i,a,n)        for(int i = 0; i < n-1; i++) printf("%d ",a[i]);
printf("%d\n", a[n-1]);
```

```
void i_o_from_file() {

#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\KIIT\\input", "r", stdin);
    freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    F(i, 0, n1) {
            L[i] = arr[l + i];
    }
    F(i, 0, n2) {
            R[i] = arr[m + 1 + i];
    }
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                    arr[k] = L[i];
                    i++;
            }
```

```
            else {
                arr[k] = R[j];
                j++;
            }
            k++;
    }
    while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
    }
    while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
    }
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
            int m = (l + r) / 2;

            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);

            merge(arr, l, m, r);
    }
}
```

```
int main() {

    i_o_from_file();

    /* ****************************************** */

    pf("n\t\t|\tbest\t\tavg\t\t\tworst\n_______|_____________________
_______________\n");

    int sizes;
    sf(sizes);

    F(i, 0, sizes) {
        int n;
        sf(n);

        pf("%d\t|\t", n);
        int arr[n];
        time_t start, end;
        double time;

        // Best
        F(j, 0, n) {
            arr[j] = j + 1;
        }

        start = clock();
        mergeSort(arr, 0, n - 1);
        end = clock();

        time = (end - start) * 1.0 / CLOCKS_PER_SEC;
```

```
        pf("%f\t", time);

        // Avg
        F(j, 0, n) {
            arr[j] = rand() % 10000;
        }

        start = clock();
        mergeSort(arr, 0, n - 1);
        end = clock();

        time = (end - start) * 1.0 / CLOCKS_PER_SEC;

        pf("%f\t", time);

        // Worst
        F(j, 0, n) {
            arr[j] = rand() % 10000;
        }

        start = clock();
        mergeSort(arr, 0, n - 1);
        end = clock();

        time = (end - start) * 1.0 / CLOCKS_PER_SEC;

        pf("%f\n", time);
    }
    return 0;
}
```

Output

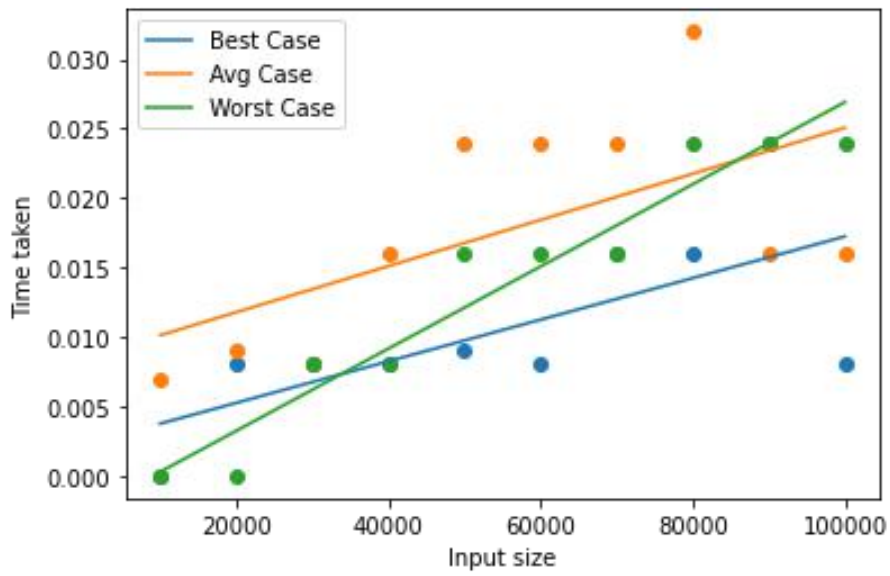
input

```
10
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
```

output

n	best	avg	worst
10000	0.000000	0.007000	0.000000
20000	0.008000	0.009000	0.000000
30000	0.008000	0.008000	0.008000
40000	0.008000	0.016000	0.008000
50000	0.009000	0.024000	0.016000
60000	0.008000	0.024000	0.016000
70000	0.016000	0.024000	0.016000
80000	0.016000	0.032000	0.024000
90000	0.024000	0.016000	0.024000
100000	0.008000	0.016000	0.024000

Graph



2. **Write a C program to implement the Tower of Hanoi problem using the Divide and Conqueror approach.**

Program

```
// Author: Chaudhary Hamdan

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

#define sf(x)          scanf("%d", &x)
#define pf             printf
#define pfs(x)         printf("%d ", x)
#define pfn(x)         printf("%d\n", x)
#define pfc(x)         printf("%d, ", x)
#define F(i,x,y)       for(int i = x; i < y; i++)
#define Fl(i,x,y,inc)   for(int i = x; i < y; i += inc)
#define RF(i,x,y)       for(int i = x; i >= y; i--)
#define pfarr(i,a,n)    for(int i = 0; i < n-1; i++) pfs(a[i]); pfn(a[n-1]);


void i_o_from_file() {

#ifndef ONLINE_JUDGE
    freopen("C:\\Users\\KIIT\\input", "r", stdin);
    freopen("C:\\Users\\KIIT\\output", "w", stdout);
#endif
}
```

```
void towerOfHanoi(int n, char* s, char* h, char* d) {

    if (n == 1) {
            pf("Move disk %d from %s to %s\n", n, s, d);
            return;
    }

    towerOfHanoi(n - 1, s, d, h);
    pf("Move disk %d from %s to %s\n", n, s, d);
    towerOfHanoi(n - 1, h, s, d);

}

int main() {

    i_o_from_file();

    /* ******************************************** */


    int n;
    sf(n);

    time_t start, end;
    double time;


    start = clock();
    towerOfHanoi(n, "STRT", "HLPR", "DEST");
    end = clock();
```

```
    time = (end - start) * 1.0 / CLOCKS_PER_SEC;

    pf("\nTime: %f\n", time);

    return 0;
}
```

Output

input

```
3
```

output

```
Move disk 1 from STRT to DEST
Move disk 2 from STRT to HLPR
Move disk 1 from DEST to HLPR
Move disk 3 from STRT to DEST
Move disk 1 from HLPR to STRT
Move disk 2 from HLPR to DEST
Move disk 1 from STRT to DEST

Time: 0.003000
```