

LAB REPORT

CSE 114 : Data Structure and Algorithms Sessional

PREPARED BY

Mehrin Farzana

ID: 2101013

Session: 2021-2022

Date: 30/07/2023

SUPERVISED BY

Suman Saha

Lecturer

Department of IRE, BDU



BANGABANDHU SHEIKH MUJIBUR

RAHMAN DIGITAL UNIVERSITY

(BDU)

List of Problems

1. Given an array `arr[]`, its starting position `l` and its ending position `r`. Sort the array using merge sort algorithm.

Constraints: $1 \leq N \leq 10^5$ $1 \leq arr[i] \leq 10^5$.

2. Analyze time complexity and space complexity of the Merge Sort Algorithm.

Problem No.: 01**Problem Statement:**

Write down a program that implements three sorting algorithms (bubble sort, selection sort, and insertion sort). Using a switch statement to choose the desired algorithm.

Code:

```
#include <stdio.h>

void merge_sort(long int arr[], long int l, long int r)
{
    if (l < r) {

        long int m = l + (r - l) / 2;

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

        long int i, j, k;
        long int n1 = m - l + 1;
        long int n2 = r - m;

        long int L[n1], R[n2];

        for (i = 0; i < n1; i++)
            L[i] = arr[l + i];
        for (j = 0; j < n2; j++)
            R[j] = arr[m + 1 + j];

        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++;
        }
    }
}
```

```

        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

}

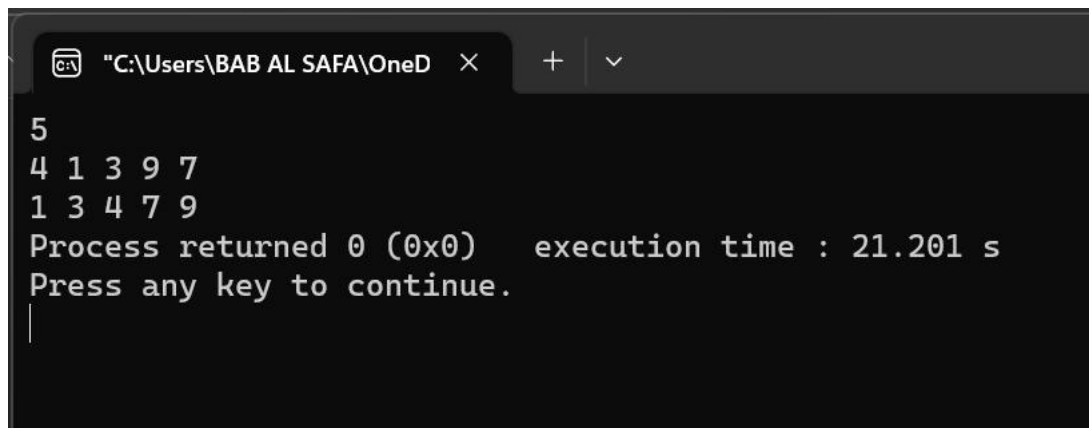
int main()
{
    long int n;
    scanf("%ld", &n);
    long int a[n];
    for(long int i=0; i<n; i++){
        scanf("%ld", &a[i]);
    }

    merge_sort(a, 0, n - 1);

    for(long int i=0; i<n; i++){
        printf("%ld ", a[i]);
    }
    return 0;
}

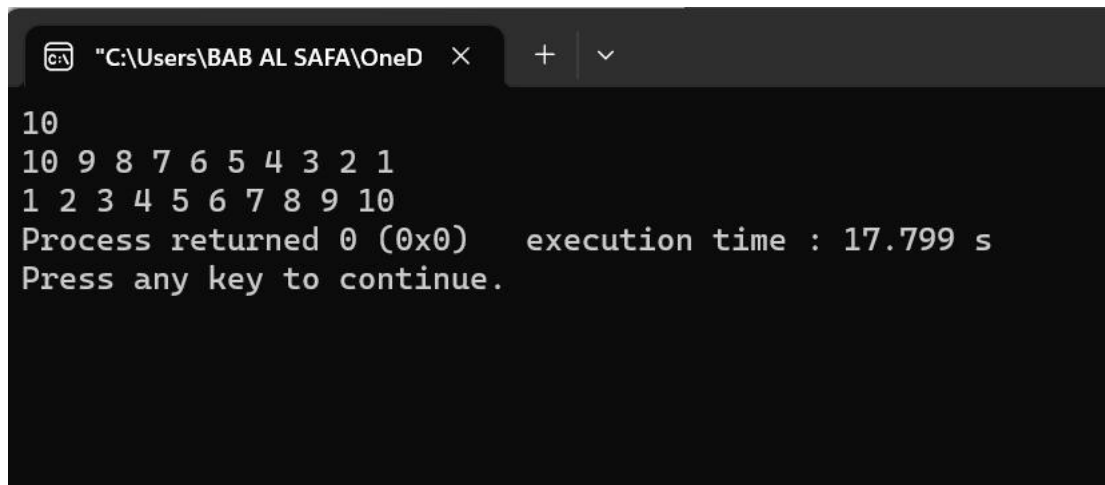
```

Output:



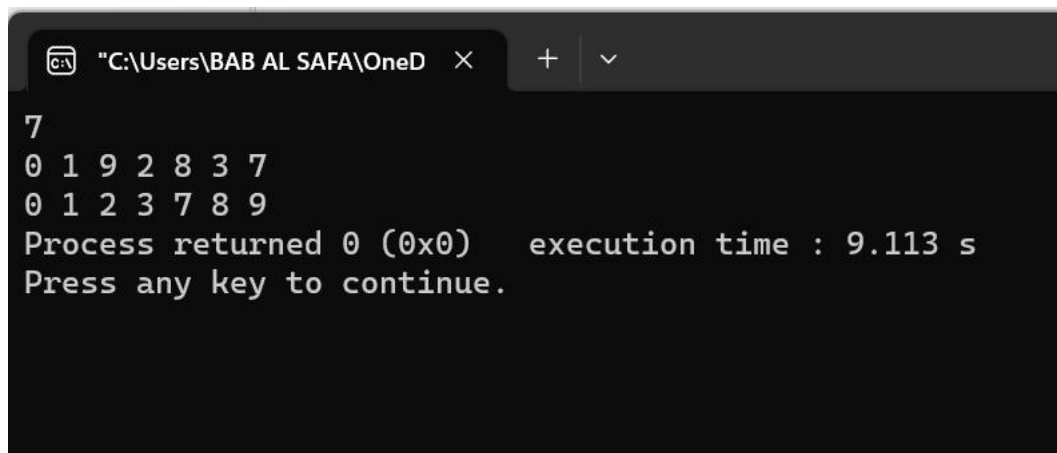
```
"C:\Users\BAB AL SAFA\OneD" × + v
5
4 1 3 9 7
1 3 4 7 9
Process returned 0 (0x0) execution time : 21.201 s
Press any key to continue.
|
```

Fig 1.1: Output on console for case 1.



```
"C:\Users\BAB AL SAFA\OneD" × + v
10
10 9 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 10
Process returned 0 (0x0) execution time : 17.799 s
Press any key to continue.
|
```

Fig 1.2: Output on console for case 2.



```
"C:\Users\BAB AL SAFA\OneD" × + v
7
0 1 9 2 8 3 7
0 1 2 3 7 8 9
Process returned 0 (0x0) execution time : 9.113 s
Press any key to continue.
|
```

Fig 1.3: Output on console for case 3.

Problem No.: 02**Problem Statement:**

Analyze time complexity and space complexity of the Merge Sort Algorithm.

Code:

```
#include <stdio.h>

void merge_sort(long int arr[], long int l, long int r)
{
    if (l < r) {

        long int m = l + (r - l) / 2;

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

        long int i, j, k;
        long int n1 = m - l + 1;
        long int n2 = r - m;

        long int L[n1], R[n2];

        for (i = 0; i < n1; i++)
            L[i] = arr[l + i];
        for (j = 0; j < n2; j++)
            R[j] = arr[m + 1 + j];

        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++;
    }
}

int main()
{
    long int n;
    scanf("%ld", &n);
    long int a[n];
    for(long int i=0; i<n; i++){
        scanf("%ld", &a[i]);
    }

    merge_sort(a, 0, n - 1);

    for(long int i=0; i<n; i++){
        printf("%ld ", a[i]);
    }
    return 0;
}

```

Analysis:

The time complexity of merge sort for best case, average case and worst case is $O(N * \log N)$. The space complexity of merge sort algorithm is $O(N)$.

Merge Sort is a sorting algorithm that uses the divide and conquer approach. It keeps on dividing the array into two parts until they cannot be divided further and then they are sorted and merged back together into sorted arrays.

Thus, The time complexity of merge sort for best case, average case and worst case is $O(N * \log N)$.

In case of merge sort, we need to build a temporary array to merge the sorted arrays. So the auxiliary space requirement is $O(N)$.

Thus, the total space required is $O(N)$.