Experiment: 2

Select any two arrays and try to merge these arrays after sorting. Also, compute the time complexity.

Student Name: Nitish Rai

UID: 23MCA20326

Branch: Computers Application

Section/Group: 4(A)

Semester: 2nd

Date of Performance: 29/01/2024

Subject Name: Design and Analysis of Algorithms

Subject Code: 23CAH-511

A. The task is to:

Select two arrays, sort them individually, and merge them into a single sorted array. Measure and analyse the time complexity of the merging process. Provide a concise report detailing the steps and the computed time complexity.

B. Steps of Experiment:

- Understand the problem and divide it into parts.
- Install JDK: If Java isn't installed on your system, download and install it.
- Open IDE(Integrated development environment) like VS code.
- Start by creating a new Java file with a .java extension (e.g., program.java).
- Write your code in a structural manner taking care of indentation to maintain readability.
- Execute your code

C. Practical Code:

```
import java.util.Scanner;

public class MergeSort_College_ExperimentOne {

  public static void divideIntoSubParts(int arr[], int startingIndex, int endIndex) {
    if (startingIndex >= endIndex) {
      return;
    }
    int mid = startingIndex + (endIndex - startingIndex) / 2;
    divideIntoSubParts(arr, startingIndex, mid);
    divideIntoSubParts(arr, mid + 1, endIndex);
    mergeDivide(arr, startingIndex, mid, endIndex);
  }

  public static void mergeDivide(int arr[], int startingIndex, int mid, int endIndex) {
    int merged[] = new int[endIndex - startingIndex + 1];
    int index = startingIndex;
    int index2 = mid + 1;
    int x = 0;
    while (index <= mid && index2 <= endIndex) {
      if (arr[index] <= arr[index2]) {
        merged[x++] = arr[index++];
      } else {merged[x++] = arr[index2++];}
    }
      while (index <= mid) {
        merged[x++] = arr[index++];
      }
      while (index2 <= endIndex) {
        merged[x++] = arr[index2++];
      }
      for (int i = 0, j = startingIndex; i < merged.length; i++, j++) {
        arr[j] = merged[i];
      }
    }
```

```
  public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    System.out.println("Enter the size of array:");
    int n = scn.nextInt();
    int ary[] = new int[n];
    for (int i = 0; i < ary.length; i++) {
      ary[i] = scn.nextInt();
    }
    divideIntoSubParts(ary, 0, n - 1);
    for (int i : ary) {
        System.out.print(i+ " ");
    }
    scn.close();
  }
}
```

D. Output:

PS C:\Users\Nitish\Desktop\JAVA\javaComeBack> cd "c:\Users\Nitish\Desktop\JAVA\javaComeBack\" ; if ($?) { javac MergeSort_College_ExperimentOne.java } ; if ($?) { java MergeSort_College_ExperimentOne }
Enter the size of array:
7
25 96 1 36 74 2 6 4
1 2 6 25 36 74 96

E. Time Complexity:

Worst: O(n log n)

Learning outcomes:

- Understand the concepts of functions, recursion and merge sort.
- Understand how to take user inputs.
- Understand the concept of the creation of an array.
- Understanding the concept of printing the output with the help of for each loop.
- Understanding how to deal with errors regarding array index out of bound.