CSE3004 Design and Analysis of Algorithms ELA Winter 2023-2024 Semester

Lab sheet - (L15+L16)

Name : Koya Madhusudhana Rao

Reg.No : 21BCE9905

Git Hub Repository :- https://github.com/Koya-Madhusudhana-Rao/DAA-Lab

1. Write a programs to implement the following:

(a) Quick Sort

```java
//import java.util.Scanner;
public class quicksort {
    public static void main(String args[]) {

        //Scanner sc = new Scanner(System.in);

        int arr[] = { 1, 2, 3, 8, 7, 6, 4 };
        int l = 0;
        int r = arr.length;
        // Before Sorting of arrya
        System.out.println("Before sorting");
        for (int n : arr) {
            System.out.print(n + " ");
        }
        // Performing quicksort

        quicksorte(arr, 0, arr.length - 1);

        // after sorting
        System.out.println("");

        System.out.println("After sorting");
        for (int n : arr) {
            System.out.print(n + " ");
        }

    }

    // Method to qucksort recursively
    public static void quicksorte(int[] arr, int l, int r) {
        if (l < r) {
```

```
            int pi = partition(arr, l, r);

            quicksorte(arr, l, pi - 1);
            quicksorte(arr, pi + 1, r);

        }

    }

    // method for finding pivot element and swaping with left most element if
it was
    // low
    public static int partition(int[] arr, int l, int r) {
        int pivot = arr[r];
        int i = (l - 1);

        for (int j = l; j <= r - 1; j++) {

            // If current element is smaller than the pivot
            if (arr[j] < pivot) {

                // Increment index of smaller element
                i++;
                swap(arr, i, j);
            }

        }
        swap(arr, i + 1, r);
        return (i + 1);
    }

    // Performing swaping

    public static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;

    }

}
```

(b) Merge Sort

```java
public class mergesort {
    public static void main(String[] args) {
        int arr[] = { 2, 5, 1, 3, 4, 7, 8, 9, 6 };

        // Before Sorting of Array
        System.out.println("Befor Sorting");
        for (int n : arr) {
            System.out.print(n + " ");
        }
        // performing Merge sort
        mergesort(arr, 0, arr.length - 1);

        System.out.println("");

        System.out.println("After Sorting");
        for (int n : arr) {
            System.out.print(n + " ");
        }

    }

    // Method for dividing array
    public static void mergesort(int[] arr, int l, int r)

    {
        if (l < r) {
            int mid = (l + r) / 2;
            mergesort(arr, l, mid);
            mergesort(arr, mid + 1, r);
            merge(arr, l, mid, r);
        }
    }
    // Actually merging elements

    public static void merge(int[] arr, int l, int m, int r)

    {
        int n1 = m + 1 - l;
        int n2 = r - m;
```

```java
        int[] larr = new int[n1];
        int[] rarr = new int[n2];

        for (int x = 0; x < n1; x++) {
            larr[x] = arr[l + x];
        }
        for (int x = 0; x < n2; x++) {
            rarr[x] = arr[m + 1 + x];
        }

        int i = 0, j = 0, k = l;
        while (i < n1 && j < n2) {
            if (larr[i] <= rarr[j]) {
                arr[k] = larr[i];
                i++;
            } else {
                arr[k] = rarr[j];
                j++;
            }
            k++;
        }

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

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

}
```