G) Implement  Merge Sort or  Quick Sort algorithm using  Amortized Analysis.

Aarti Ambekar

RollNo:20121056

public class Merge\_Sort\_Amortized {

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

        int n1 = m - l + 1;

        int n2 = r - m;

        int[] L = new int[n1];

        int[] R = new int[n2];

        for (int i = 0; i < n1; i++) {

            L[i] = arr[l + i];

        }

        for (int j = 0; j < n2; j++) {

            R[j] = arr[m + 1 + j];

        }

        int 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++;

        }

    }

    public static void mergeSort(int[] arr, int l, int r) {

        if (l < r) {

            int m = l + (r - l) / 2;

            mergeSort(arr, l, m);

            mergeSort(arr, m + 1, r);

            merge(arr, l, m, r);

        }

    }

    public static void main(String[] args) {

        int[] arr = {12, 11, 13, 5, 6, 7};

        System.out.println("Original Array:");

        for (int num : arr) {

            System.out.print(num + " ");

        }

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

        System.out.println("\nSorted Array:");

        for (int num : arr) {

            System.out.print(num + " ");

        }

    }

}

Output:

Original Array:

12 11 13 5 6 7

Sorted Array:

5 6 7 11 12 13

public class Quik\_Sort\_Amortized{

    public static int partition(int[] arr, int low, int high) {

        int pivot = arr[high];

        int i = (low - 1);

        for (int j = low; j < high; j++) {

            if (arr[j] < pivot) {

                i++;

                int temp = arr[i];

                arr[i] = arr[j];

                arr[j] = temp;

            }

        }

        int temp = arr[i + 1];

        arr[i + 1] = arr[high];

        arr[high] = temp;

        return i + 1;

    }

    public static void quickSort(int[] arr, int low, int high) {

        if (low < high) {

            int pi = partition(arr, low, high);

            quickSort(arr, low, pi - 1);

            quickSort(arr, pi + 1, high);

        }

    }

    public static void main(String[] args) {

        int[] arr = {12, 11, 13, 5, 6, 7};

        System.out.println("Original Array:");

        for (int num : arr) {

            System.out.print(num + " ");

        }

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

        System.out.println("\nSorted Array:");

        for (int num : arr) {

            System.out.print(num + " ");

        }

    }

}

