18/11/2024 DSA PRACTICE 5

1.Bubble Sort

```
import java.util.Scanner;


class BubbleSort{
    public static void bubbleSort(int arr[]) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            boolean swapped = false;
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            if (!swapped) break;
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
```

```
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        bubbleSort(arr);
        System.out.println("Sorted array:");
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
}
```

Time Complexity : O(n^2)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice5>javac BubbleSort.java

C:\Users\POOJA\Documents\SDE\DSA_Practice5>java BubbleSort
Enter the size of the array: 5
Enter the elements of the array:
4 1 3 9 7
Sorted array:
1 3 4 7 9
```

2.Quick Sort

```
import java.util.Scanner;


class QuickSort{
    static int partition(int[] arr, int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
```

```
        for (int j = low; j <= high - 1; j++) {
                if (arr[j] < pivot) {
                        i++;
                        swap(arr, i, j);
                }
        }
        swap(arr, i + 1, high);
        return i + 1;
}

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

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) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();
```

```
        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        quickSort(arr, 0, n - 1);
        System.out.println("Sorted array:");
        for (int val : arr) {
            System.out.print(val + " ");
        }
    }
}
```

Time Complexity : O(n^2)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice5>javac QuickSort.java

C:\Users\POOJA\Documents\SDE\DSA_Practice5>java QuickSort
Enter the size of the array: 6
Enter the elements of the array:
10 7 8 9 1 5
Sorted array:
1 5 7 8 9 10
```

3.Non Repeating Character

```
import java.util.Scanner;


class NonRepeatingChar{


    static final int MAX_CHAR = 26;


    static char nonRepeatingChar(String s) {
        int[] freq = new int[MAX_CHAR];
```

```
        for (char c : s.toCharArray())
            freq[c - 'a']++;
        for (int i = 0; i < s.length(); ++i) {
            if (freq[s.charAt(i) - 'a'] == 1)
                return s.charAt(i);
        }
        return '$';
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        System.out.println(nonRepeatingChar(s));
    }
}
```

Time Complexity:O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice5>javac NonRepeatingChar.java

C:\Users\POOJA\Documents\SDE\DSA_Practice5>java NonRepeatingChar
racecar
e
```

4.Edit Distance

```
import java.util.Scanner;


public class EditDistance {
    private static int minDisRec(String s1, String s2, int m, int n, int[][] memo) {
        if (m == 0) return n;
        if (n == 0) return m;
```

```
        if (memo[m][n] != -1) return memo[m][n];
        if (s1.charAt(m - 1) == s2.charAt(n - 1)) {
                memo[m][n] = minDisRec(s1, s2, m - 1, n - 1, memo);
        } else {
                int insert = minDisRec(s1, s2, m, n - 1, memo);
                int remove = minDisRec(s1, s2, m - 1, n, memo);
                int replace = minDisRec(s1, s2, m - 1, n - 1, memo);
                memo[m][n] = 1 + Math.min(insert, Math.min(remove, replace));
        }
        return memo[m][n];
}

public static int minDis(String s1, String s2) {
        int m = s1.length(), n = s2.length();
        int[][] memo = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) {
                for (int j = 0; j <= n; j++) {
                        memo[i][j] = -1;
                }
        }
        return minDisRec(s1, s2, m, n, memo);
}

public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1 = sc.nextLine();
        String s2 = sc.nextLine();
        System.out.println(minDis(s1, s2));
```

```
    }
}
```

Time Complexity : O(m x n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice5>javac EditDistance.java

C:\Users\POOJA\Documents\SDE\DSA_Practice5>java EditDistance
GEEXSFRGEEKKS
GEEKSFORGEEKS
3
```

5.K Largest Elements

```
import java.util.*;


class KLargest{
    static ArrayList<Integer> kLargest(int[] arr, int k) {
        int n = arr.length;
        Integer[] arrInteger = Arrays.stream(arr).boxed().toArray(Integer[]::new);
        Arrays.sort(arrInteger, Collections.reverseOrder());
        ArrayList<Integer> res = new ArrayList<>();
        for (int i = 0; i < k; i++)
            res.add(arrInteger[i]);
        return res;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt();
```

```
        int k = sc.nextInt();
        ArrayList<Integer> res = kLargest(arr, k);
        for (int ele : res)
            System.out.print(ele + " ");
    }
}
```

Time Complexity : O(n * log(n))

```
C:\Users\POOJA\Documents\SDE\DSA_Practice5>javac KLargest.java

C:\Users\POOJA\Documents\SDE\DSA_Practice5>java KLargest
7
1 23 12 9 30 2 50
3
50 30 23
C:\Users\POOJA\Documents\SDE\DSA_Practice5>
```

6.Form the largest number

```
import java.util.*;


class LargestNumber{


    static boolean myCompare(String s1, String s2) {
        return (s1 + s2).compareTo(s2 + s1) > 0;
    }


    static String findLargest(int[] arr) {
        ArrayList<String> numbers = new ArrayList<>();
        for (int ele : arr) {
            numbers.add(Integer.toString(ele));
        }
```

```
        Collections.sort(numbers, (s1, s2) -> myCompare(s1, s2) ? -1 : 1);
        if (numbers.get(0).equals("0")) {
            return "0";
        }
        StringBuilder res = new StringBuilder();
        for (String num : numbers) {
            res.append(num);
        }
        return res.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];
        for (int i = 0; i < n; i++)
            arr[i] = sc.nextInt();
        System.out.println(findLargest(arr));
    }
}
```

Time Complexity : O(n·klogn)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice5>javac LargestNumber.java

C:\Users\POOJA\Documents\SDE\DSA_Practice5>java LargestNumber
5
3 30 34 5 9
9534330
```