Problem Solving 18/11/2024

1. Bubble Sort
   Time complexity: O(n^2)
   Space complexity: O(1)

```java
package program18thNov;
import java.util.Scanner;
public class BubbleSort {
    static void bubbleSort(int arr[], int n) {
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++) {
            swapped = false;
            for (j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            if (!swapped) {
                break;
            }
        }
    }
    static void printArray(int arr[], int size) {
        for (int i = 0; i < size; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }
        bubbleSort(arr, n);
        printArray(arr, n);
        scanner.close();
    }
}

```

```
<terminated> BubbleSort [Java Application] C:\Program Files\Java\bin\javaw.exe (18
Enter the number of elements in the array: 8
Enter the elements:
8 5 4 1 2 6 7 9
1 2 4 5 6 7 8 9
```

2. Quick Sort

Time complexity: O(n^2)

Space complexity: O(log n)

```java
package program18thNov;
import java.util.Scanner;
public 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 number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter the elements:");
        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 + " ");
        }
        scanner.close();
    }
```

```
<terminated> QuickSort [Java Application] C:\Program Files\Java\bin\javaw.exe (18-
Enter the number of elements in the array: 10
Enter the elements:
8 4 5 6 1 2 3 7 9 6
Sorted array:
1 2 3 4 5 6 6 7 8 9
```

3. Non Repeating Character
   Time complexity: O(n)
   Space complexity: O(1)

```java
package program18thNov;
import java.util.Scanner;
public class NonRepeatingCharacter {
    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 scanner = new Scanner(System.in);
        System.out.print("Enter the string: ");
        String s = scanner.nextLine();
        System.out.println(nonRepeatingChar(s));
        scanner.close();
    }
}
```

```
<terminated> NonRepeatingCharacter [Java Application] C:\Program F
Enter the string: geeksforgeeks
f
```

4. Edit Distance

   Time complexity: O(m * n)

   Space complexity: O(m * n)

```java
package program18thNov;
import java.util.Scanner;
public class EditDistance {
    public static int editDistDP(String s1, String s2) {
        int m = s1.length();
        int n = s2.length();
        int[][] dp = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) {
            dp[i][0] = i;
        }
        for (int j = 0; j <= n; j++) {
            dp[0][j] = j;
        }
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = 1 + Math.min(dp[i - 1][j - 1], Math.min(dp[i][j - 1], dp[i - 1][j]));
                } }  }
        return dp[m][n];
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the first string:");
        String s1 = scanner.nextLine();
        System.out.println("Enter the second string:");
        String s2 = scanner.nextLine();
        int result = editDistDP(s1, s2);
        System.out.println(result);
    }
}

```

```
<terminated> EditDistance [Java Application] C:\Program Files\Java\bi
Enter the first string:
geeks
Enter the second string:
geexs
1
```

5. Kth Largest Number

Time complexity: O(n log k)

Space complexity: O(k)

```java
package program18thNov;
import java.util.*;
public class KthLargestNumber {
    public static List<Integer> kLargest(int[] arr, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
        for (int i = 0; i < k; i++) {
            minHeap.add(arr[i]);
        }
        for (int i = k; i < arr.length; i++) {
            if (arr[i] > minHeap.peek()) {
                minHeap.poll();
                minHeap.add(arr[i]);
            }
        }
        List<Integer> result = new ArrayList<>();
        while (!minHeap.isEmpty()) {
            result.add(minHeap.poll());
        }
        Collections.reverse(result);
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("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();        }
        System.out.println("Enter the value of k:");
        int k = scanner.nextInt();
        List<Integer> result = kLargest(arr, k);
        System.out.println("The " + k + " largest elements are:");
        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}
```

```
<terminated> KthLargestNumber [Java Application] C:\Program Files\Ja
Enter the size of the array:
10
Enter the elements of the array:
2 1 5 3 4 6 7 8 9 5
Enter the value of k:
5
The 5 largest elements are:
9 8 7 6 5
```

6. Form The Largest Number

Time complexity: O(n log n)

Space complexity: O(n)

```java
package program18thNov;
import java.util.*;
public class FormtheLargestNumber {
    public static String largestNumber(String[] arr) {
        Comparator<String> myCompare = (X, Y) -> (X + Y).compareTo(Y + X);
        Arrays.sort(arr, myCompare.reversed());
        if (arr[0].equals("0")) {
            return "0";
        }
        StringBuilder result = new StringBuilder();
        for (String num : arr) {
            result.append(num);
        }
        return result.toString();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = scanner.nextInt();
        String[] arr = new String[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.next();
        }
        System.out.println("The largest number formed is: " + largestNumber(arr));
    }
}

```

```
<terminated> FormtheLargestNumber [Java Application] C:\Program Files\Java\bin\javaw.exe (18-Nov-2024, 7:41
Enter the number of elements:
10
Enter the elements:
55 44 385 15 98 21 75 02 56 45
The largest number formed is: 987556554544385211502
```