A white paper with text and blue text

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated

BucketSort Code:

public class BucketSort {

    public static void bucketSort(int[] A) {

        int n = A.length;

        int max = getMax(A); // Find the maximum value in A

        int[] B = new int[max + 1]; // B needs to accommodate values from 0 to max

        for (int i = 0; i < n; i++) {

            B[A[i]]++;

        }

        int index = 0;

        for (int i = 0; i < B.length; i++) {

            for (int j = 0; j < B[i]; j++) {

                A[index++] = i;

            }

        }

    }

    public static int getMax(int[] A) {

        int max = A[0];

        for (int i = 1; i < A.length; i++) {

            if (A[i] > max) {

                max = A[i];

            }

        }

        return max;

    }

    public static void main(String[] args) {

        int[] A = {56, 23, 90, 51, 29, 34, 55};

        bucketSort(A);

        for (int i = 0; i < A.length; i++) {

            System.out.println(A[i]);

        }

    }

}

BucketSort output:

A screenshot of a computer program

Description automatically generated

String BucketSort

/\*Create a new version of the given algorithm that handles String input

of length 32 characters. Implement this

in Java

 \* After sorting, the strings are ordered based on their lengths,

 \* with shorter strings appearing before longer ones.

 \* If two strings have the same length, their order remains the same

 \*  as in the original array.

 \*/

import java.util.ArrayList;

import java.util.List;

public class StringBucketSort {

    public static void stringBucketSort(String[] A) {

        int n = A.length;

        int maxStringLength = getMaxStringLength(A); // Find the maximum string length in A

        List<List<String>> buckets = new ArrayList<>();

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

            buckets.add(new ArrayList<>());

        }

        // Distribute strings into buckets based on their lengths

        for (int i = 0; i < n; i++) {

            String str = A[i];

            int strLength = str.length();

            buckets.get(strLength - 1).add(str);

        }

        // Concatenate the strings from the buckets to form the sorted array of strings

        int index = 0;

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

            List<String> bucket = buckets.get(i);

            for (String str : bucket) {

                A[index++] = str;

            }

        }

    }

    public static int getMaxStringLength(String[] A) {

        int max = 0;

        for (String str : A) {

            if (str.length() > max) {

                max = str.length();

            }

        }

        return max;

    }

    public static void main(String[] args) {

        String[] A = {"abcdefg", "zxy", "mnopqrstuvwxyz", "pqrstu", "ijk"};

        stringBucketSort(A);

        for (String str : A) {

            System.out.println(str);

        }

    }

}

String BucketSort output:

A screenshot of a computer program

Description automatically generated

ALL CODE RUNS PROPERLY