**Mod 5: Critical Thinking**

Alex Mehler

CSU Global

CSC400-1

Santosh Gottipamula

December 22, 2024

**Code:**

public class bigOh {

public static void radixSort(int[] array) {

// Find maximum to determine number of digits

int max = max(array);

// counting sort for each digit

for (int exp = 1; max / exp > 0; exp \*= 10) {

sort(array, exp);

}

}

private static int max(int[] array) {

int max = array[0];

for (int num : array) {

if (num > max) {

max = num;

}

}

return max;

}

private static void sort(int[] array, int exp) {

int n = array.length;

int[] output = new int[n]; // Output array

int[] count = new int[10]; // Count array for digits

// Count occurrences of each digit

for (int num : array) {

int digit = (num / exp) % 10;

count[digit]++;

}

// Update count, stores the actual position of digits

for (int i = 1; i < 10; i++) {

count[i] += count[i - 1];

}

// output array, places elements in sorted order

for (int i = n - 1; i >= 0; i--) {

int digit = (array[i] / exp) % 10;

output[count[digit] - 1] = array[i];

count[digit]--;

}

// Copy sorted array back to original array

System.arraycopy(output, 0, array, 0, n);

}

private static void printArray(int[] array) {

for (int num : array) {

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

}

System.out.println();

}

public static void main(String[] args) {

int[] array = {783, 99, 472, 182, 264, 543, 356, 295, 692, 491, 94};

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

printArray(array);

radixSort(array);

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

printArray(array);

}

}

**Screenshot:**

**A screenshot of a computer

Description automatically generated**

**Analysis:**

**Radix sort:**

The sort takes the maximum number by analyzing each digit starting from the least to the most significant digit.

**Big-O:**

The time complexity of the algorithm takes O(n+k), where k is the range, and multiplies it by the max number in the list. The max number is represented as d = log10(max). k is a constant of 10 in this scenario and can be thrown out of the Big-Oh. Therefore, the Big-Oh is O(d\*n) with a space complexity of O(n).

**Git:**

https://github.com/SevRnce/csc400\_mod\_4