W5 - Data Structures & Algorithms

Problem 1: Sorting Large Array of Random Integers

Overview: Generate an array of 1,000,000 random integers ranging from 1 to 1,000. Sort this array using both a custom counting sort algorithm and Java's built-in Arrays.sort method. Compare the performance of these two sorting methods in terms of running time.

Algorithm Steps:

1. **Generate Random Integers**:
    - Create an array to hold 1,000,000 random integers.
    - Populate the array with random integers ranging from 1 to 1,000.
2. **Counting Sort**:
    - Implement a counting sort algorithm to sort the array.
    - Measure the time taken to complete the sorting.
3. **Arrays.sort Method (2A)**:
    - Use Java's built-in Arrays.sort method to sort the array.
    - Measure the time taken to complete the sorting.
4. **Compare Running Times (2B)**:

- Compare the running times of both sorting algorithms.

5. **Code**:

```
import java.util.Arrays;
import java.util.Random;

public class SortingComparison {

   // Method to generate an array of random integers
   private static int[] generateRandomArray(int size, int maxValue) {
       Random random = new Random();
       int[] array = new int[size];
       for (int i = 0; i < size; i++) {
           array[i] = random.nextInt(maxValue) + 1;
       }
       return array;
   }

   // Method to perform counting sort
   private static void countingSort(int[] array, int maxValue) {
       int[] count = new int[maxValue + 1];
       int[] output = new int[array.length];

       // Count each element
       for (int value : array) {
           count[value]++;
       }

       // Modify count array
       for (int i = 1; i <= maxValue; i++) {
           count[i] += count[i - 1];
       }

       // Build the output array
       for (int i = array.length - 1; i >= 0; i--) {
           output[count[array[i]] - 1] = array[i];
           count[array[i]]--;
       }
```

```
    // Copy the sorted elements back into the original array
    System.arraycopy(output, 0, array, 0, array.length);
}

public static void main(String[] args) {
    int size = 1_000_000;
    int maxValue = 1_000;

    // Generate random integers
    int[] arrayForCountingSort = generateRandomArray(size, maxValue);
    int[] arrayForBuiltInSort = Arrays.copyOf(arrayForCountingSort, arrayForCountingSort.length);

    // Sort using counting sort and measure time
    long startTime = System.nanoTime();
    countingSort(arrayForCountingSort, maxValue);
    long countingSortTime = System.nanoTime() - startTime;

    // Sort using Java's built-in method and measure time
       startTime = System.nanoTime();
       Arrays.sort(arrayForBuiltInSort);
       long builtInSortTime = System.nanoTime() - startTime;

       // Output the time taken by both sorting algorithms
       System.out.println("Time taken by Counting Sort: " + countingSortTime / 1000000 + " ms");
       System.out.println("Time taken by Arrays.sort: " + builtInSortTime / 1000000 + " ms");
   }
}
```

Problem 2: Implement Hash Table for RMIT Student Information

↳ 1 cell hidden

Problem 3: Extended - Implement Remove Operation in Hash Table for RMIT Student Information

↳ 1 cell hidden