Assignment 7: Part 1-Merge Sorting

Ryan Brinson

10/25/23

## Output:

A screenshot of a computer screen

Description automatically generated

## Code:

// Name: Ryan Brinson

// Class:CS 3305 W04

// Term: Fall 2023

//  Instructor:  Carla McManus

//  Assignment:  7-Part-1-Sorting

import java.util.Arrays;

import java.util.LinkedList;

public class MergeSort {

    public static void main(String[] args) {

        // Initilize the array to be sorted

        LinkedList<Integer> list = new LinkedList<Integer>(

            Arrays.asList(

                1009,21,3,55,2022,24,99,501,105,98,178,245,0,3305,990,76,373,1010,642,777));

        System.out.println("Unsorted list:");

        System.out.println(list);

        // Pass the list to mergeSort method

        mergeSort(list);

        // Print the final, sorted list

        System.out.println("\nFinal sorted list:");

        System.out.println(list);

        System.out.println();

    }

    // ----- Main Methods ----- //

    // Merge Sort method

    public static void mergeSort(LinkedList<Integer> list) {

        // Recursive stop condition

        if (list.size() > 1){

            // Split the list in half using a for loop

            LinkedList<Integer> firstHalf = new LinkedList<>();

            for (int i = 0; i < list.size() / 2; i++) {

                firstHalf.addLast(list.get(i));

            }

            // Call recursively until there is only one element left

            // in each partition

            mergeSort(firstHalf);

            // Repeat with the second half of the list

            LinkedList<Integer> secondHalf = new LinkedList<>();

            for (int j = list.size() / 2; j < list.size(); j++){

                secondHalf.addLast(list.get(j));

            }

            // Call recursively until there is only one element

            // left in each partition

            mergeSort(secondHalf);

            // Call the merge function to rejoin each half

            merge(firstHalf, secondHalf, list);

        }

    }

    // Merge Method

    public static void merge(LinkedList<Integer> list1, LinkedList<Integer> list2, LinkedList<Integer> temp){

        // Indexing integers

        Integer curr1 = 0;

        Integer curr2 = 0;

        Integer curr3 = 0;

        // List1 and List2 should add to the total number of elements

        // in the list so we can use that as the stop condition

        while ((curr1 < list1.size()) && (curr2 < list2.size())){

            // Set temp to the smaller element from index of list1 vs list2

            if (list1.get(curr1) < list2.get(curr2)){

                temp.set(curr3++, list1.get(curr1++));

            }

            else{

                temp.set(curr3++, list2.get(curr2++));

            }

        }

        // These two while funtions finish filling out temp

        // if there is any elements left over that haven't been

        // sorted

        while (curr1 < list1.size())

            temp.set(curr3++, list1.get(curr1++));

        while (curr2 < list2.size())

            temp.set(curr3++, list2.get(curr2++));

    }

}