A paper with text and images

Description automatically generated

. A screenshot of a math test

Description automatically generated

A white paper with black text

Description automatically generated

Merge Sort algorithm

import java.util.Vector;

public class MergeSort{

    public static void merge\_sort(Vector<Integer> A, int p, int r){

        if(p < r){

            int q =(p+r)/2;

            merge\_sort(A, p, q);

            merge\_sort(A, q+1, r);

            merge(A, p, q, r);

        }

    }

        public static void merge(Vector<Integer> A, int p, int q, int r){

            int n1 = q-p+1;

            int n2 = r-q;

            Vector<Integer> L = new Vector<Integer>();

            Vector<Integer> R = new Vector<Integer>();

            for(int i = 0; i < n1; i++){

                L.add(A.get(p+i));

            }

            for(int j = 0; j < n2; j++){

                R.add(A.get(q+j+1));

            }

            L.add(Integer.MAX\_VALUE);

            R.add(Integer.MAX\_VALUE);

            int i = 0;

            int j = 0;

            for(int k = p; k <= r; k++){

                if(L.get(i) <= R.get(j)){

                    A.set(k, L.get(i));

                    i++;

                }

                else{

                    A.set(k, R.get(j));

                    j++;

                }

            }

        }

        public static void main(String[] args) {

            Vector<Integer> A = new Vector<Integer>();

            A.add(3);

            A.add(41);

            A.add(52);

            A.add(26);

            A.add(38);

            A.add(57);

            A.add(9);

            A.add(49);

            System.out.println("Before sorting: " + A);

            merge\_sort(A, 0, A.size()-1);

            System.out.println("After sorting: " + A);

            }

        }

MergeSort Algorithm showing correct output

A screenshot of a computer program

Description automatically generated

MergeSort optimization “in line merge sort” using LinkedList

// this optimization introduces an "in line merge sort" that is more efficient than the original merge sort

import java.util.LinkedList;

public class MergeSortOptimization{

    public static void inLineMergeSort(LinkedList<Integer> A, int p, int r){

        if(p < r){

            int q = (p + r) / 2;

            inLineMergeSort(A, p, q);

            inLineMergeSort(A, q + 1, r);

            inLineMerge(A, p, q, r);

        }

    }

    public static void inLineMerge(LinkedList<Integer> A, int p, int q, int r){

        LinkedList<Integer> L = new LinkedList<Integer>(A.subList(p, q + 1));

        LinkedList<Integer> R = new LinkedList<Integer>(A.subList(q + 1, r + 1));

        L.add(Integer.MAX\_VALUE);

        R.add(Integer.MAX\_VALUE);

        int i = 0;

        int j = 0;

        for(int k = p; k <= r; k++){

            if(L.get(i) <= R.get(j)){

                A.set(k, L.get(i));

                i++;

            } else {

                A.set(k, R.get(j));

                j++;

            }

        }

    }

    public static void main(String[] args){

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

        A.add(3);

        A.add(41);

        A.add(52);

        A.add(26);

        A.add(38);

        A.add(57);

        A.add(9);

        A.add(49);

        System.out.println("Before sorting: " + A);

        inLineMergeSort(A, 0, A.size() - 1);

        System.out.println("After sorting: " + A);

    }

}

MergeSortOptimization with correct output:

A screenshot of a computer program

Description automatically generated

**All Code outputs properly!!**