A document with text and images

Description automatically generated

A screenshot of a math problem

Description automatically generated

A white paper with blue text and black text

Description automatically generated

CODE:

public class TailRecursiveQuicksort {

    public static void tailRecursiveQuickSort(int[] A, int p, int r) {

        while (p < r) {

            int[] q = partition(A, p, r);

            tailRecursiveQuickSort(A, p, q[0] - 1);

            p = q[1] + 1;

        }

    }

    public static int[] partition(int[] A, int p, int r) {

        int x = A[r];

        int i = p - 1;

        for (int j = p; j < r; j++) {

            if (A[j] <= x) {

                i++;

                int temp = A[i];

                A[i] = A[j];

                A[j] = temp;

            }

        }

        int temp = A[i + 1];

        A[i + 1] = A[r];

        A[r] = temp;

        return new int[]{i + 1, i + 1}; // returning both indices

    }

    public static void main(String[] args) {

        int[] arr = {3, 6, 8, 10, 4, 2, 1};

        tailRecursiveQuickSort(arr, 0, arr.length - 1);

        for (int num : arr) {

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

        }

    }

}

OUTPUT:

A screenshot of a computer program

Description automatically generated

OPTIMIZATION:

public class TailRecursiveQSoptimization {

    private static final int INSERTION\_SORT\_THRESHOLD = 10;

    public static void tailRecursiveQuickSort(int[] A, int p, int r) {

        while (p < r) {

            if (r - p < INSERTION\_SORT\_THRESHOLD) {

                insertionSort(A, p, r);

                return;

            }

            int[] q = partition(A, p, r);

            tailRecursiveQuickSort(A, p, q[0] - 1);

            p = q[1] + 1;

        }

    }

    public static int[] partition(int[] A, int p, int r) {

        int x = A[p]; // pivot

        int i = p - 1;

        int j = r + 1;

        while (true) {

            do {

                i++;

            } while (A[i] < x);

            do {

                j--;

            } while (A[j] > x);

            if (i >= j)

                return new int[]{j, i}; // return indices of the partition

            swap(A, i, j);

        }

    }

    public static void insertionSort(int[] A, int p, int r) {

        for (int i = p + 1; i <= r; i++) {

            int key = A[i];

            int j = i - 1;

            while (j >= p && A[j] > key) {

                A[j + 1] = A[j];

                j--;

            }

            A[j + 1] = key;

        }

    }

    public static void swap(int[] A, int i, int j) {

        int temp = A[i];

        A[i] = A[j];

        A[j] = temp;

    }

    public static void main(String[] args) {

        int[] arr = {3, 6, 8, 10, 4, 2, 1};

        tailRecursiveQuickSort(arr, 0, arr.length - 1);

        for (int num : arr) {

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

        }

    }

}

OPTIMIZATION OUTPUT:

A screenshot of a computer program

Description automatically generated

All code works as intended