A paper with text and images

Description automatically generated

A screenshot of a math test

Description automatically generated

A white paper with blue text

Description automatically generated

CODE:

public class RandomizedQuicksort {

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

        int i = (int) (Math.random() \* (r - p + 1)) + p;

        int temp = A[i];

        A[i] = A[r];

        A[r] = temp;

        return partition(A, p, r);

    }

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

        if (p < r) {

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

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

            randomizedQuicksort(A, q[1] + 1, r);

        }

    }

    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;

        int[] indices = {i + 1, i + 1}; // Initialize both indices to i + 1

        // Adjust the indices to cover equal elements

        for (int k = i; k >= p; k--) {

            if (A[k] == A[i + 1]) {

                indices[0] = k;

            } else {

                break;

            }

        }

        for (int k = i + 2; k <= r; k++) {

            if (A[k] == A[i + 1]) {

                indices[1] = k;

            } else {

                break;

            }

        }

        return indices;

    }

    public static void main(String[] args) {

        int[] A  = {1,3,10,2,7,5,4,6,9,8};

        randomizedQuicksort(A, 0, A.length - 1);

        for (int i : A) {

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

        }

    }

}

CODE OUTPUT:

A screen shot of a computer program

Description automatically generated

OPTIMIZATION CODE:

import java.util.concurrent.ThreadLocalRandom;

public class optimization {

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

        if (p < r) {

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

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

            randomizedQuicksort(A, q[1] + 1, r);

        }

    }

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

        int i = ThreadLocalRandom.current().nextInt(p, r + 1);

        swap(A, i, r);

        return partition(A, p, r);

    }

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

        int x = A[r];

        int i = p - 1;

        int j = p - 1;

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

            if (A[k] < x) {

                i++;

                j++;

                swap(A, i, j);

            } else if (A[k] == x) {

                j++;

                swap(A, k, j);

            }

        }

        swap(A, i + 1, r);

        return new int[]{i, j};

    }

    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[] A = {1, 3, 10, 2, 7, 5, 4, 6, 9, 8};

        randomizedQuicksort(A, 0, A.length - 1);

        for (int i : A) {

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

        }

    }

}

OPTIMIZATION OUTPUT:

A screenshot of a computer program

Description automatically generated

All code works as intended