ASSIGNMENT NO :6

import java.util.Random;

public class QuickSortAnalysis {

// Deterministic Quick Sort (using the last element as the pivot)

public static void quickSortDeterministic(int[] array, int low, int high) {

if (low < high) {

int pivotIndex = partitionDeterministic(array, low, high);

quickSortDeterministic(array, low, pivotIndex - 1);

quickSortDeterministic(array, pivotIndex + 1, high);

}

}

// Partition function for deterministic quick sort

private static int partitionDeterministic(int[] array, int low, int high) {

int pivot = array[high]; // last element as pivot

int i = low - 1;

for (int j = low; j < high; j++) {

if (array[j] < pivot) {

i++;

swap(array, i, j);

}

}

swap(array, i + 1, high);

return i + 1;

}

// Randomized Quick Sort

public static void quickSortRandomized(int[] array, int low, int high) {

if (low < high) {

int pivotIndex = partitionRandomized(array, low, high);

quickSortRandomized(array, low, pivotIndex - 1);

quickSortRandomized(array, pivotIndex + 1, high);

}

}

// Partition function for randomized quick sort

private static int partitionRandomized(int[] array, int low, int high) {

Random random = new Random();

int pivotIndex = low + random.nextInt(high - low + 1);

swap(array, pivotIndex, high); // Move pivot to end

return partitionDeterministic(array, low, high); // Reuse the deterministic partitioning

}

// Swap helper method

private static void swap(int[] array, int i, int j) {

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

// Method to print the array

private static void printArray(int[] array) {

for (int num : array) {

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

}

System.out.println();

}

public static void main(String[] args) {

int[] dataDeterministic = {34, 7, 23, 32, 5, 62};

int[] dataRandomized = dataDeterministic.clone(); // Clone for fair comparison

System.out.println("Original Array:");

printArray(dataDeterministic);

// Analyze Deterministic Quick Sort

long startTime = System.nanoTime();

quickSortDeterministic(dataDeterministic, 0, dataDeterministic.length - 1);

long endTime = System.nanoTime();

System.out.println("Sorted Array (Deterministic):");

printArray(dataDeterministic);

System.out.println("Time taken (Deterministic): " + (endTime - startTime) + " ns");

// Analyze Randomized Quick Sort

startTime = System.nanoTime();

quickSortRandomized(dataRandomized, 0, dataRandomized.length - 1);

endTime = System.nanoTime();

System.out.println("Sorted Array (Randomized):");

printArray(dataRandomized);

System.out.println("Time taken (Randomized): " + (endTime - startTime) + " ns");

}

}

**OUTPUT :**

Original Array:

34 7 23 32 5 62

Sorted Array (Deterministic):

5 7 23 32 34 62

Time taken (Deterministic): 11760 ns

Sorted Array (Randomized):

5 7 23 32 34 62

Time taken (Randomized): 577520 ns