Exercise 3: Sorting Customer Orders Overview

building a feature for an e-commerce platform to sort customer orders based on their total price. This functionality helps in prioritizing high-value orders for efficient processing and delivery.

In this exercise,

Implement Bubble Sort and Quick Sort.

Compare their performance and usage in real scenarios. import java.util.Scanner;

class Order {

int orderId;

String customerName; double totalPrice;

public Order(int id, String name, double price) { orderId = id;

customerName = name; totalPrice = price;

}

public String toString() {

return orderId + " - " + customerName + " -

₹" + totalPrice;

}

}

public class SortCustomerOrders {

// Bubble Sort implementation

public static void bubbleSort(Order[] orders) { int n = orders.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (orders[j].totalPrice > orders[j + 1].totalPrice) { Order temp = orders[j];

orders[j] = orders[j + 1]; orders[j + 1] = temp;

}

}

}

}

// Quick Sort implementation

public static void quickSort(Order[] orders, int low, int high) { if (low < high) {

int pi = partition(orders, low, high); quickSort(orders, low, pi - 1); quickSort(orders, pi + 1, high);

}

}

private static int partition(Order[] orders, int low, int high) { double pivot = orders[high].totalPrice;

int i = low - 1;

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

if (orders[j].totalPrice < pivot) { i++;

Order temp = orders[i];

orders[i] = orders[j]; orders[j] = temp;

}

}

Order temp = orders[i + 1]; orders[i + 1] = orders[high]; orders[high] = temp;

return i + 1;

}

// Display orders

public static void displayOrders(Order[] orders, String title) { System.out.println("\n" + title);

for (Order order : orders) { System.out.println(order);

}

}

public static void main(String[] args) { Scanner sc = new Scanner(System.in); Order[] orders = new Order[5];

System.out.println("Enter details for 5 orders (id name price):"); for (int i = 0; i < 5; i++) {

int id = sc.nextInt(); String name = sc.next();

double price = sc.nextDouble(); orders[i] = new Order(id, name, price);

}

// Copy original array for both sorting Order[] bubbleSorted = orders.clone(); Order[] quickSorted = orders.clone();

// Apply Bubble Sort bubbleSort(bubbleSorted);

displayOrders(bubbleSorted, "Bubble Sort by Total Price");

// Apply Quick Sort

quickSort(quickSorted, 0, quickSorted.length - 1); displayOrders(quickSorted, "Quick Sort by Total Price");

}

}

Analysis

Time and Space Complexity Comparison:

Algorithm Best Case Average Case Worst Case Space Complexity Bubble Sort O(n) O(n²) O(n²) O(1)

Quick Sort O(n log n) O(n log n) O(n²) O(log n)

sample input:

Enter details for 5 orders (id name price):

101 Alice 550.50 102 Bob 300.00

103 Carol 900.25 104 Dave 450.75 105 Eva 120.00

sample output:

Bubble Sort by Total Price

105 - Eva -

102 - Bob -

₹120.0

₹300.0

104 - Dave -

₹450.75

101 - Alice -

103 - Carol -

₹550.5

₹900.25

Quick Sort by Total Price

105 - Eva -

102 - Bob -

₹120.0

₹300.0

104 - Dave -

₹450.75

101 - Alice -

103 - Carol -

₹550.5

₹900.25