Step 1: Understand Sorting Algorithms

Different sorting algorithms have different time complexities and are suitable for different scenarios. Here's a brief overview of four common sorting algorithms:

Bubble Sort: Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

Insertion Sort: Insertion sort is a simple sorting algorithm that works by taking elements from the list one by one and inserting them at their correct position into a new sorted list.

Quick Sort: Quick sort is a divide-and-conquer algorithm that selects a pivot element, partitions the list around the pivot, and recursively sorts the sublists.

Merge Sort: Merge sort is a divide-and-conquer algorithm that divides the list into smaller sublists, sorts each sublist, and then merges the sorted sublists back together.

Step 2: Setup

Create a Order class with attributes like orderId, customerName, and totalPrice:

public class Order {

private int orderId;

private String customerName;

private double totalPrice;

public Order(int orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

// Getters and setters

public int getOrderId() {

return orderId;

}

public void setOrderId(int orderId) {

this.orderId = orderId;

}

public String getCustomerName() {

return customerName;

}

public void setCustomerName(String customerName) {

this.customerName = customerName;

}

public double getTotalPrice() {

return totalPrice;

}

public void setTotalPrice(double totalPrice) {

this.totalPrice = totalPrice;

}

}

Step 3: Implementation

Implement Bubble Sort to sort orders by totalPrice:

public class BubbleSort {

public static void sortOrders(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].getTotalPrice() < orders[j + 1].getTotalPrice()) {

// Swap orders[j] and orders[j + 1]

Order temp = orders[j];

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

orders[j + 1] = temp;

}

}

}

}

}

Implement Quick Sort to sort orders by totalPrice:

public class QuickSort {

public static void sortOrders(Order[] orders) {

quickSort(orders, 0, orders.length - 1);

}

private 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) {

Order pivot = orders[high];

int i = (low - 1);

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

if (orders[j].getTotalPrice() > pivot.getTotalPrice()) {

i++;

// Swap orders[i] and orders[j]

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

// Swap orders[i + 1] and orders[high]

Order temp = orders[i + 1];

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

orders[high] = temp;

return i + 1;

}

}

Step 4: Analysis

Performance Comparison:

Bubble Sort: The time complexity of Bubble Sort is O(n^2) in the worst case, where n is the number of orders.

Quick Sort: The time complexity of Quick Sort is O(n log n) on average, where n is the number of orders.

Why Quick Sort is Generally Preferred:

Quick Sort is generally preferred over Bubble Sort because of its better performance. Quick Sort has an average time complexity of O(n log n), which is much faster than Bubble Sort's O(n^2) time complexity for large datasets. Additionally, Quick Sort is a more efficient sorting algorithm in practice, especially for large datasets.