**Exercise 2: E-commerce Platform Search Function**

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**Code:**public class Product {

int productId;

String productName;

String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

@Override

public String toString() {

return "[" + productId + ", " + productName + ", " + category + "]";

}

}  
public class Search {

// Linear Search

public static Product linearSearch(Product[] products, String targetName) {

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

// Binary Search - assumes sorted by productName

public static Product binarySearch(Product[] products, String targetName) {

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int cmp = products[mid].productName.compareToIgnoreCase(targetName);

if (cmp == 0)

return products[mid];

else if (cmp < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

}

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1, "Shoes", "Fashion"),

new Product(2, "Laptop", "Electronics"),

new Product(3, "Watch", "Accessories"),

new Product(4, "Shirt", "Fashion"),

new Product(5, "Phone", "Electronics")

};

// Sort for Binary Search

Product[] sortedProducts = Arrays.copyOf(products, products.length);

Arrays.sort(sortedProducts, Comparator.comparing(p -> p.productName.toLowerCase()));

// Linear Search Test

String target1 = "Phone";

String target2 = "Watch";

Product result1 = Search.linearSearch(products, target1);

System.out.println("Linear Search Result: " + result1);

// Binary Search Test

Product result2 = Search.binarySearch(sortedProducts, target2);

System.out.println("Binary Search Result: " + result2);

}

}

**Output:**A black screen with white text

AI-generated content may be incorrect. **Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

**Code:**public class FinancialForecast {

// Recursive method to calculate future value

public static double forecast(double initialValue, double rate, int years) {

if (years == 0) {

return initialValue;

}

return forecast(initialValue, rate, years - 1) \* (1 + rate);

}

}  
public class Main {

public static void main(String[] args) {

double initialValue = 1000.0;

double annualGrowthRate = 0.10;

int years = 5;

double predictedValue = FinancialForecast.forecast(initialValue, annualGrowthRate, years);

System.out.printf("Predicted future value after %d years: ₹%.2f\n", years, predictedValue);

}

}

**Output:**

A black screen with white text

AI-generated content may be incorrect.