Data Structures and Algorithms

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:

```
import java.util.Arrays;

public class EcommerceSearch {

    static class Product implements Comparable<Product> {
        String id;
        String name;
        String category;

        public Product(String id, String name, String category) {
            this.id = id;
            this.name = name;
            this.category = category;
        }

        @Override
        public int compareTo(Product other) {
            return this.id.compareTo(other.id);
        }
```

```
}

public static void main(String[] args) {

    Product[] products = {
        new Product("P100", "Laptop", "Electronics"),
        new Product("P200", "Phone", "Electronics"),
        new Product("P300", "Chair", "Furniture")
    };


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


    String searchId = "P200";

    System.out.println("Linear Search Results:");
    Product linearResult = linearSearch(products, searchId);
    System.out.println(linearResult != null ? "Found: " + linearResult.name : "Not found");

    System.out.println("\nBinary Search Results:");
    Product binaryResult = binarySearch(sortedProducts, searchId);
    System.out.println(binaryResult != null ? "Found: " + binaryResult.name : "Not found");
}


public static Product linearSearch(Product[] products, String id) {
    for (Product p : products) {
        if (p.id.equals(id)) {
            return p;
        }
    }
    return null;
}


public static Product binarySearch(Product[] products, String id) {
    int left = 0;
    int right = products.length - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        int comparison = products[mid].id.compareTo(id);
```

```
            if (comparison == 0) {
                return products[mid];
            } else if (comparison < 0) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return null;
    }
}
```

```
Output

Linear Search Results:
Found: Phone

Binary Search Results:
Found: Phone

=== Code Execution Successful ===
```

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 SimpleFinancialForecast {


   public static double calculateFutureValue(double currentValue,  double growthRate,int years)
   {

      if (years == 0) {
         return currentValue;
      }

            double nextValue = currentValue * (1 + growthRate);


      return calculateFutureValue(nextValue, growthRate, years - 1);
   }

   public static void main(String[] args) {
      double initialAmount = 1000.0;
      double annualRate = 0.05;
      int investmentYears = 10;


      double result = calculateFutureValue(initialAmount, annualRate, investmentYears);


      System.out.println("Simple Financial Forecast");
      System.out.println("-------------------------");
      System.out.printf("Initial amount: $%.2f%n", initialAmount);
      System.out.printf("Annual growth rate: %.1f%%%n", annualRate * 100);
      System.out.printf("Investment period: %d years%n", investmentYears);
      System.out.printf("Future value: $%.2f%n", result);
   }
}
```

Output:

```
Simple Financial Forecast
-------------------------
Initial amount: $1000.00
Annual growth rate: 5.0%
Investment period: 10 years
Future value: $1628.89

=== Code Execution Successful ===
```