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.

Solution of Exercise 2

- Big O notation helps us efficiently check out the performance of an algorithm as input size increases. It focuses on how the algorithm scales and gives us an upper bound — that is, the worst-case performance.
- Common asymptotic notations are:
  - the Big-Oh notation (for worst case analysis),
  - the Theta notation (for average case analysis) and
  - the Omega notation (for best case analysis).
- Time Complexity analysis:

Algorithm	Best Case	Average Case	Worst Case
Linear Search	Ω (1)	Θ (n)	O (n)
Binary Search	Ω (1)	Θ (logn)	O (logn)

- As more queries occur in an e-commerce platform and many products will be stored, it would be better to sort and store our products such that Binary Search can be applied as it would ensure a better searching performance compared to Linear Search.

Code

Product.java

```
package M2;

public class Product {
    private String productID;
    private String productName;
    private double price;

    public Product(String productID, String productName, double price) {
        this.productID = productID;
        this.productName = productName;
        this.price = price;
    }

    public String getProductID() {
        return productID;
    }

    public String getProductName() {
        return productName;
    }

    public double getPrice() {
        return price;
    }

    public void printDetails() {
        System.out.println("Product ID: " + productID);
        System.out.println("Product Name: " + productName);
        System.out.println("Product Price: " + price);
    }
}
```

Searcher.java

```
package M2;
import java.util.Scanner;

public class Searcher {
```

```java
    public static Product linearProductSearch(Product[] products, String
searchID) {
        for(int i = 0; i < products.length; i++) {
            if(products[i].getProductID().equals(searchID)){
                return products[i];
            }
        }
        return null;
    }

    public static Product binaryProductSearch(Product[] products, String
searchID) {
        int low = 0, high = products.length - 1;

        while(low <= high) {
            int mid = (low + high) / 2;
            if(products[mid].getProductID().equals(searchID)) {
                return products[mid];
            } else if(products[mid].getProductID().compareTo(searchID) < 0)
{
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return null;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Product products[] = new Product[5];

        products[0] = new Product("P001", "Laptop", 80000);
        products[1] = new Product("P002", "Perfume", 500.00);
        products[2] = new Product("P003", "Microphone", 1300.00);
        products[3] = new Product("P004", "Y2K Hat", 20000);
        products[4] = new Product("P005", "Jelly Beans", 100.00);

        // for(int i = 0; i < 5; i++) {
        //     System.out.println("Enter product ID: ");
        //     String productID = sc.nextLine();

        //     System.out.println("Enter product name: ");
        //     String prodName = sc.nextLine();

        //     System.out.println("Enter product price: ");
        //     double price = sc.nextDouble();
```

```
        //     sc.nextLine();

        //     products[i] = new Product(productID, prodName, price);
        // }

        System.out.println("Enter product ID to search for a product [LINEAR
SEARCH]: ");
        String searchID = sc.nextLine();

        Product searchedProduct = linearProductSearch(products, searchID);

        if(searchedProduct == null) {
            System.out.println("Product with ID " + searchID + " not found")
;
        } else {
            System.out.println("Here is the requested product: " );
            searchedProduct.printDetails();
        }
        System.out.println();

System.out.println("*********************************************************
********************");
        System.out.println();

        System.out.println("Enter product ID to search for a product [BINARY
SEARCH]: ");
        String searchID2 = sc.nextLine();

        Product searchedProduct2 = binaryProductSearch(products, searchID2);

        if(searchedProduct2 == null) {
            System.out.println("Product with ID " + searchID + " not found")
;
        } else {
            System.out.println("Here is the requested product: ");
            searchedProduct2.printDetails();
        }

        sc.close();
    }
}
```

Output Screenshots

```
Enter product ID to search for a product [LINEAR SEARCH]:
P005
Here is the requested product:
Product ID: P005
Product Name: Jelly Beans
Product Price: 100.0

****************************************************************************

Enter product ID to search for a product [BINARY SEARCH]:
P004
Here is the requested product:
Product ID: P004
Product Name: Y2K Hat
Product Price: 20000.0
PS D:\DN-4.0-Java>
```

```
Enter product ID to search for a product [LINEAR SEARCH]:
P006
Product with ID P006 not found

****************************************************************************

Enter product ID to search for a product [BINARY SEARCH]:
P0001
Product with ID P006 not found
```

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.

Solution of Exercise 7

- Recursion is the process of a function calling itself until a given condition is fulfilled to solve smaller subproblems.
- It can simplify repetitive tasks that can naturally be broken down into smaller subproblems of same format.
- Time complexity: O(n)
  Space complexity: O(n) stack space
- To optimise recursive solutions:
    - If the problem calls for “remembering” values, go for *memoization* technique.
    - If not, consider an iterative approach.

Code

`FinancialForecasting.java`

```
package M2;

public class FinancialForecasting {
    public static double predictFutureValue(double initialVal, double rate,
```

```
int years) {
        if(years == 0) {
            return initialVal;
        }

        return predictFutureValue(initialVal, rate, years - 1) * (1 + rate);
    }
    public static void main(String[] args) {
        double initialVal = 50000.00;
        double rate = 0.1;
        int years = 3;

        System.out.println("Initial val: " + initialVal);
        System.out.println("Rate of interest: " + rate);
        System.out.println("Years: "+ years);
        System.out.println();
        System.out.println("Future value after " + years + " years is:  " +
predictFutureValue(initialVal, rate, years));
    }
}
```

Output Screenshots

```
Initial val: 50000.0
Rate of interest: 0.1
Years: 3

Future value after 3 years is:  66550.00000000001
PS D:\DN-4.0-Java>
```

```
Initial val: 2000.0
Rate of interest: 0.08
Years: 2

Future value after 2 years is:  2332.8
PS D:\DN-4.0-Java>
```