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

**1. Understanding Asymptotic Notation**

**Big O Notation**

* Big O notation is used to describe the upper bound of an algorithm's running time or space usage in terms of input size n.
* It helps us understand how the algorithm scales and behaves as input grows large.
* This is crucial in optimizing systems like e-commerce platforms where millions of products might be searched.

**Best, Average, and Worst-Case Scenarios**

* Best Case (Omega): Fewest operations (e.g., finding item at first index).
* Average Case (Theta): Average number of operations over all cases (e.g., item in the middle).
* Worst Case ( Big O): Maximum steps needed (e.g., item not found or at the last index).

**2. Setup**

**Create a Product Class**

class Product {

int productId;

String productName;

String category;

}

**3. Implementation**

**CODE:**

import java.util.\*;

class Product {

    int productId;

    String productName;

    String category;

}

public class Ecommerce{

    static void linearSearch(Product[] products, int targetId) {

        boolean found = false;

        for (int i = 0; i < products.length; i++) {

            if (products[i].productId == targetId) {

                System.out.println("Linear Search: Product found at index " + i);

                found = true;

                break;

            }

        }

        if (!found) {

            System.out.println("Linear Search: Product not found");

        }

    }

    static void binarySearch(Product[] products, int targetId) {

        // Create a copy of the original array to avoid modifying it

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

        Arrays.sort(sortedProducts, Comparator.comparingInt(p -> p.productId));

        int low = 0, high = sortedProducts.length - 1;

        boolean found = false;

        while (low <= high) {

            int mid = (low + high) / 2;

            if (sortedProducts[mid].productId == targetId) {

                // Find and print the original index

                for (int i = 0; i < products.length; i++) {

                    if (products[i].productId == targetId) {

                        System.out.println("Binary Search: Product found at original index " + i);

                        break;

                    }

                }

                found = true;

                break;

            } else if (sortedProducts[mid].productId < targetId) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        if (!found) {

            System.out.println("Binary Search: Product not found");

        }

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        Product[] products = new Product[3];

        for (int i = 0; i < 3; i++) {

            products[i] = new Product();

            System.out.println("Enter details for Product " + (i + 1));

            System.out.print("Product ID: ");

            products[i].productId = scanner.nextInt();

            scanner.nextLine(); // consume newline

            System.out.print("Product Name: ");

            products[i].productName = scanner.nextLine();

            System.out.print("Category: ");

            products[i].category = scanner.nextLine();

            System.out.println();

        }

        System.out.print("Enter Product ID to search: ");

        int searchId = scanner.nextInt();

        linearSearch(products, searchId);

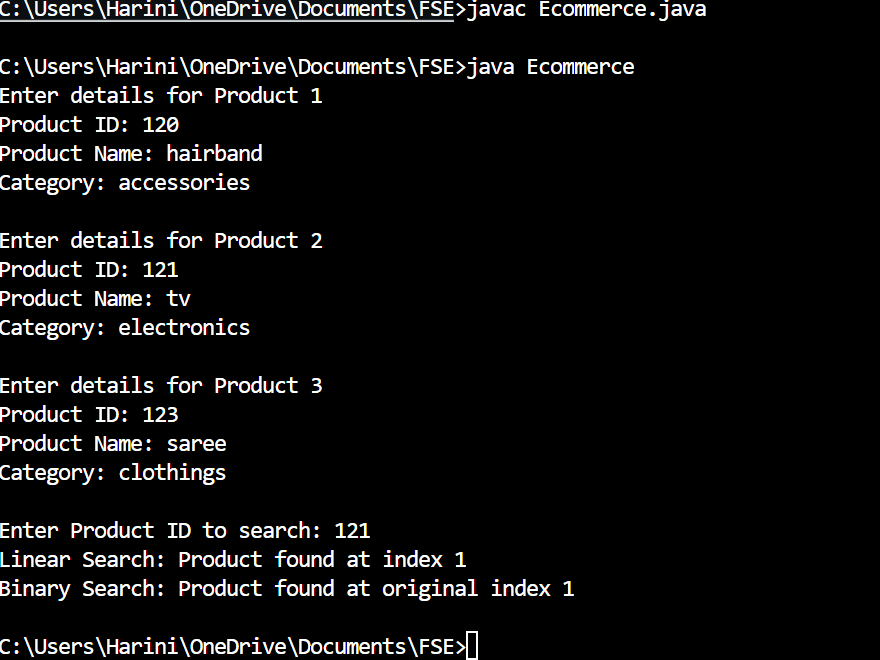
        binarySearch(products, searchId);

        scanner.close();

    }

}

**OUTPUT:**

****

**4. Analysis**

| **Search Algorithm** | **Best Case** | **Average Case** | **Worst Case** |  |
| --- | --- | --- | --- | --- |
| Linear Search | O(1) | O(n) | O(n) |  |
| Binary Search | O(1) | O(log n) | O(log n) |  |

**Which Algorithm is More Suitable?**

* Binary Search is more suitable for an e-commerce platform, especially when:
  + The product list is pre-sorted by productId.
  + Performance and speed are crucial for user satisfaction.
  + Large datasets are involved.
* Linear Search is only helpful:
  + For small or unsorted datasets.
  + During temporary states when data is being dynamically updated.