**SKILL : DATA STRUCTURES AND ALGORITHM**

**Qn 1: E-commerce Platform Search Function**

**Program:**

import java.util.\*;

public class EcommercePlatformExample {

**//Linear Search**

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

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

            if (products[i].productName.equalsIgnoreCase(targetvalue)) {

                return i;

            }

        }

        return -1;

    }

**//Binary Search**

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

        int first = 0, last = products.length - 1;

        while (first <= last) {

            int mid = (first+last) / 2;

            int Value = products[mid].productName.compareToIgnoreCase(targetvalue);

            if (Value = = 0){

                return mid;

            }

            else if (Value < 0) {

                first = mid + 1;

            }

else {

last = mid - 1;

            }

        }

        return -1;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter number of products: ");

        int n = sc.nextInt();

        sc.nextLine();

        Product[] products = new Product[n];

        for (int i = 0; i < n; i++) {

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

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

            int id = sc.nextInt();

            sc.nextLine();

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

            String name = sc.nextLine();

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

            String category = sc.nextLine();

            products[i] = new Product(id, name, category);

        }

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

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

        System.out.println("\nEnter product name to search: ");

        String target = sc.nextLine();

**// Linear Search**

        int linearIndex = linearSearch(products, target);

        if (linearIndex != -1)

            System.out.println("\nLinear Search: Found at index " + linearIndex + ": \n" + products[linearIndex]);

        else

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

**// Binary Search**

        int binaryIndex = binarySearch(sortedProducts, target);

        if (binaryIndex != -1)

            System.out.println("\nBinary Search: Found at index " + binaryIndex + ": \n" + sortedProducts[binaryIndex]);

        else

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

        sc.close();

    }

}

class Product {

    int productId;

    String productName;

    String category;

    public Product(int id, String name, String category) {

        this.productId = id;

        this.productName = name;

        this.category = category;

    }

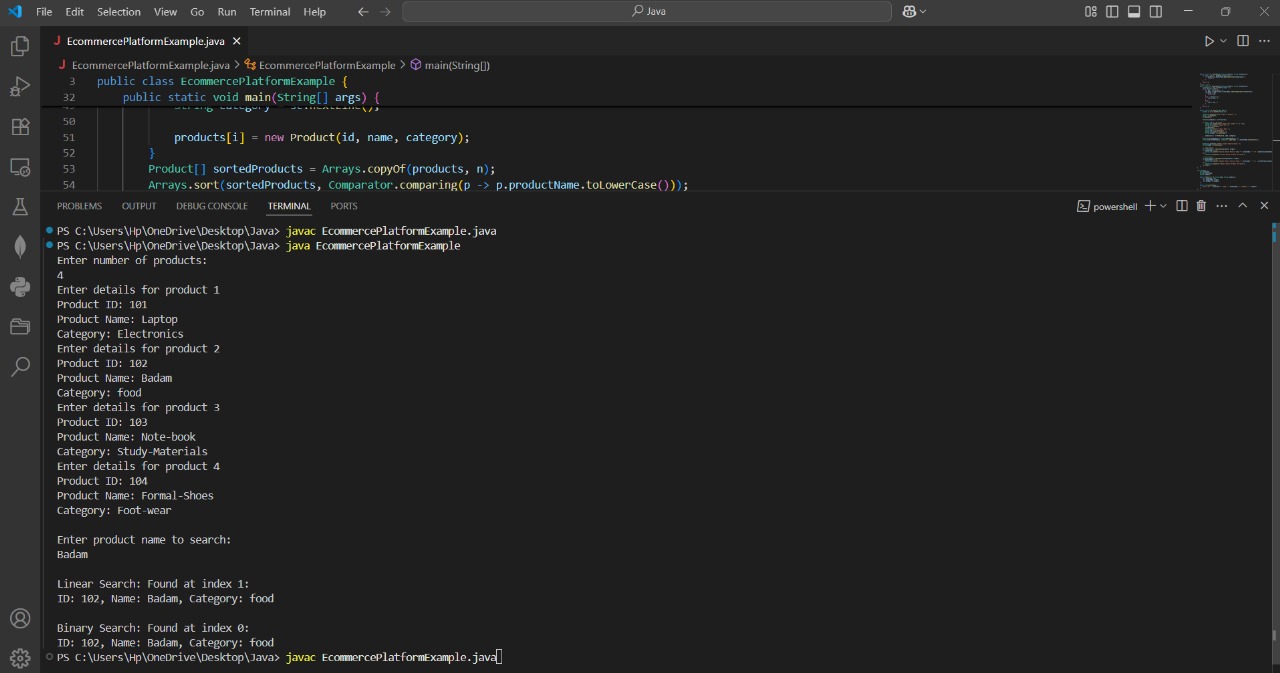
    public String toString() {

        return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

    }

}

**Output:**



**Comparison: Time complexity of Linear and Binary search algorithms.**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Linear Search (linearSearch())** | **Binary Search (binarySearch())** |
| Best Case | O(1)  Target at the first position | O(1)  Target at the middle position |
| Average Case | O(n) | O(log n) |
| Worst Case | O(n)  Target not found | O(log n)  Target not found |
| Used In Code | linearSearch(products, target) | binarySearch(sortedProducts,target) |

**Discuss which algorithm is more suitable for your platform and why?**

Binary Search is More Suitable for an E-Commerce Platform.

Reason:

* Binary search offers a time complexity of O(log n), making it faster than linear search (O(n)) as the number of products increases.
* E-commerce platforms typically display products in sorted order (e.g., by name, price, or rating), which aligns well with binary search requirements.
* It minimizes processing time and optimizes system resources, which is important for handling multiple user requests simultaneously.

**Qn 2: Financial Forecasting**

**Program:**

import java.util.\*;

public class FinancialForecastingexample {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the present value: ");

        double presentvalue=sc.nextDouble();

        // Give the decimal value or the rate for eg:if 5% then give as 0.05

        System.out.println("Enter the rate in decimal : ");

        double rate=sc.nextDouble();

      System.out.println ("Enter the Time period to predict the future value (in years): ");

        double timeperiod=sc.nextDouble();

        Double futurevalue = financialforecasting(presentvalue, rate,timeperiod);

        System.out.println ("Future value for"+" "+timeperiod+" "+"Time period is: "+futurevalue);

        sc.close();

    }

static double financialforecasting (double presentvalue,double rate,double timeperiod) {

        if (timeperiod = = 0){

            return presentvalue;

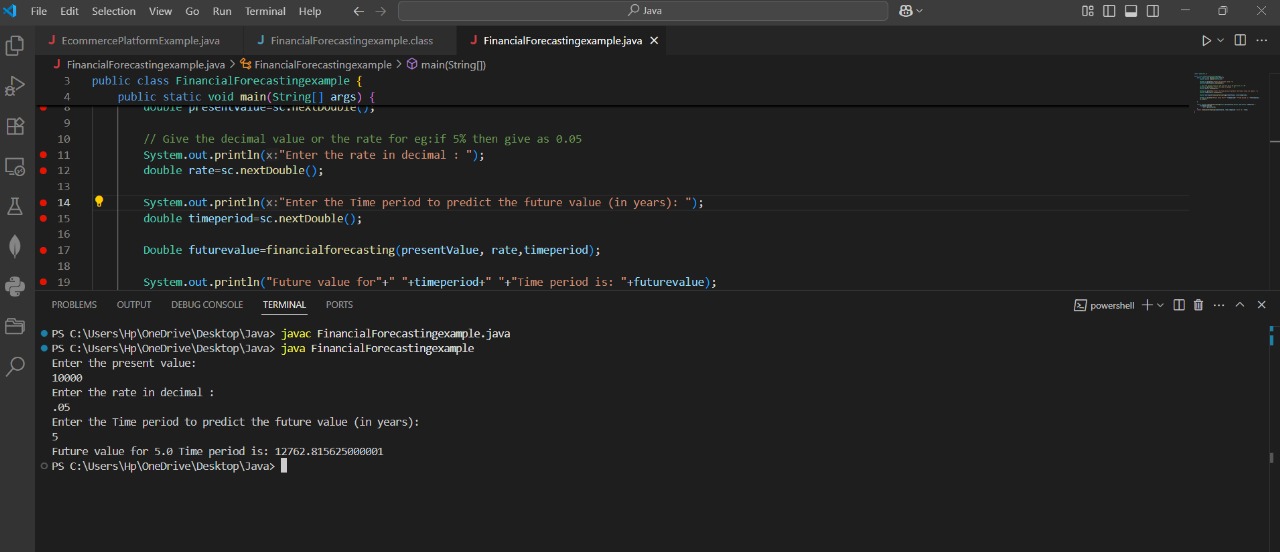
    }

    return financialforecasting (presentvalue, rate,timeperiod - 1) \* (1 + rate);

}

}

**Output:**



**Time Complexity and Space Complexity of the Recursive Algorithm:**

Time Complexity: O(n)​

Space Complexity: O(n)​