**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.

## Answer:-

Big O notation expresses an algorithm's performance in terms of input size, highlighting its efficiency. It helps compare algorithms by indicating time or space complexity growth trends.

In search operations, best-case is finding the item immediately, average-case assumes a typical input distribution, and worst-case is searching through all elements without success or finding it last.

Main.java

import java.util.Arrays;

public class Main {

    public static void main(String[] args) {

        Product[] prod = {

            new Product(1, "T-Shirt", "cloth"),

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

            new Product(3, "Shoes", "Footware"),

            new Product(4, "Pencil", "Stationery")

        };

        String target = "Laptop";

        int i1 = SearchProduct.linearSearch(prod, target);

        System.out.println("Linear search: \n");

        System.out.println("Product List:");

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

            System.out.print(prod[i].getProductName());

            if (i < prod.length - 1) {

                System.out.print(", ");

            }

        }

        System.out.println();

        System.out.println("Product found at index: "+i1);

        Arrays.sort(prod, (a, b) ->

            a.getProductName().compareToIgnoreCase(b.getProductName())

        );

        int i2 = SearchProduct.binarySearch(prod, target);

        System.out.println("\nBinary search: \n");

        System.out.println("Product List after sorting:");

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

            System.out.print(prod[i].getProductName());

            if (i < prod.length - 1) {

                System.out.print(", ");

            }

        }

        System.out.println();

        System.out.println("Product found at index: " + i2);

    }

}

Product.java

public class Product {

    private int productId;

    private String productName;

    private String category;

    public Product (){

    }

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    public int getProductId(){

        return productId;

    }

    public String getProductName(){

        return productName;

    }

    public String getcategory(){

        return category;

    }

    public String toString() {

        return "Product Id: " + productId + ", Product Name: " + productName + ", Category: " + category;

    }

}

SearchProduct.java

public class SearchProduct {

    public static int linearSearch(Product[] prod, String target){

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

            if (prod[i].getProductName().equalsIgnoreCase(target)) {

                return i;

            }

        }

        return -1;

    }

    public static int binarySearch(Product[] sortedProd, String target) {

        int low = 0;

        int high = sortedProd.length - 1;

        int mid = low + (high - low) / 2;

        while (low <= high) {

            String currName = sortedProd[mid].getProductName();

            int cmp = currName.compareToIgnoreCase(target);

            if (cmp == 0) {

                return mid;

            } else if (cmp < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

            mid = low + (high - low) / 2;

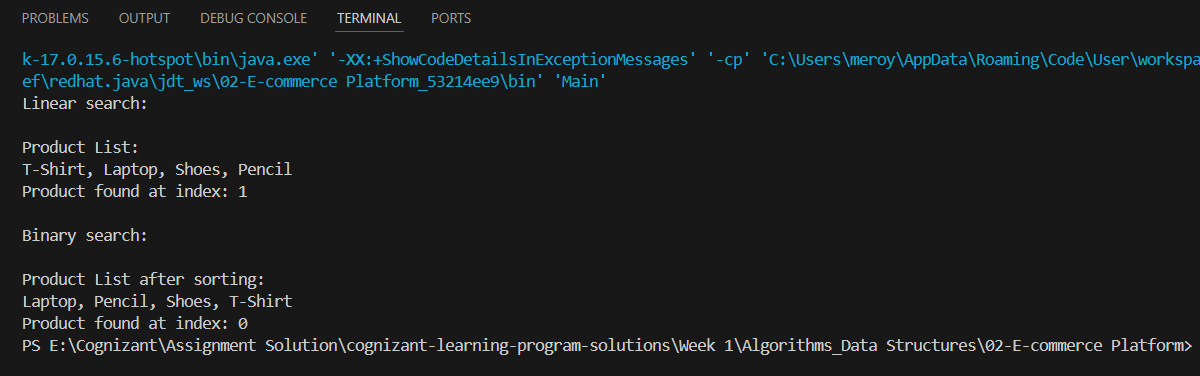
        }

        return -1;

    }

}

Output:-



Linear search has O(n) time complexity, scanning each element. Binary search is O(log n), dividing the dataset repeatedly. Binary search is faster but needs a sorted array to function efficiently.

* Linear Search is Best Suited for Small datasets..
* Binary search is best suited for Large datasets..And it also requires Sorted inputs.