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

**Problem Statement**: You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Big O Notation**:

* Big O notation is used to describe the time or space complexity of an algorithm in terms of input size n.
* It provides an upper bound on the runtime, helping evaluate how an algorithm performs as data grows.
* It helps choose the most efficient algorithm for a problem.

**IMPLEMENTATION:**

import java.util.Arrays;

import java.util.Comparator;

class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return "[" + productId + ", " + productName + ", " + category + "]";

}

}

public class ECommerceSearch {

**// Linear search method**

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

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

**// Binary search method (needs sorted array)**

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

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = products[mid].productName.compareToIgnoreCase(name);

if (cmp == 0) {

return products[mid];

} else if (cmp < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(1, "Phone", "Electronics"),

new Product(2, "Shirt", "Clothing"),

new Product(3, "Watch", "Accessories"),

new Product(4, "Shoes", "Footwear"),

new Product(5, "Laptop", "Electronics")

};

**// Linear Search (no need to sort)**

Product result1 = linearSearch(products, "Phone");

System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Product not found"));

**// Sorting the array by product name before binary search**

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

**// Binary Search**

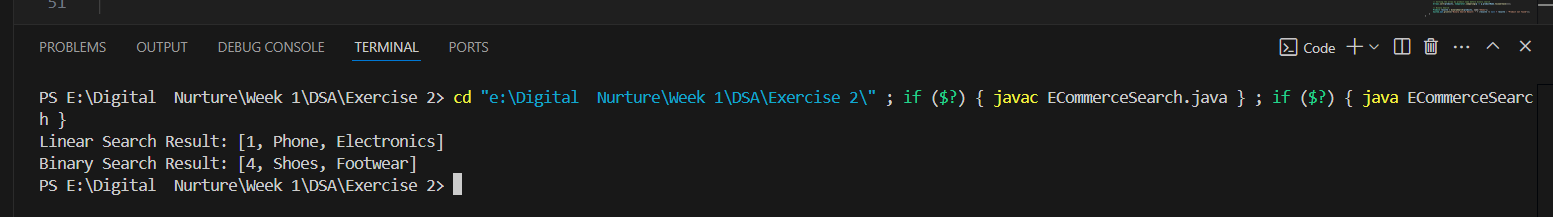
Product result2 = binarySearch(products, "Phone");

System.out.println("Binary Search Result: " + (result2 != null ? result2 : "Product not found"));

}

}

**OUTPUT:**



**ANALYSIS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Time Complexity** |
| Linear Search | O(1) | O(n) | O(n) | Slower on large data |
| Binary Search | O(1) | O(log n) | O(log n) | Faster but only works on sorted data |

**Linear search** is easy and works on **any data**, even if it's unsorted.  
But when the number of products becomes large, it becomes slow.

**Binary search** is much faster, but the data **must be sorted first**.

So for an e-commerce platform where speed matters a lot and products are already sorted (by name, price, etc.), **binary search** is the better choice.