Exercise 2: E-commerce Platform Search Function

1. Understand Asymptotic Notation

What is Big O Notation?

Big O notation describes the upper bound of an algorithm’s running time in terms of input size n.It helps you understand how an algorithm scales and performs with large data sets.

Search Case Scenarios

Scenario Linear Search Binary Search

Best Case O(1) — first match O(1) — mid element match

Average Case O(n/2) ⇒ O(n) O(log n)

Worst Case O(n) — last or no match O(log n) — not found

2. Setup

Create a Product class

3. Implementation

Product.java

public class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

@Override

public String toString() {

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

}

}

SearchUtil.java

import java.util.Arrays;

import java.util.Comparator;

public class SearchUtil {

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

for (Product product : products) {

if (product.productId == targetId) {

return product;

}

}

return null;

}

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

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

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

return products[mid];

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

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void sortProducts(Product[] products) {

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

}

}

Main.java

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(103, "Mobile", "Electronics"),

new Product(101, "Shirt", "Clothing"),

new Product(102, "Book", "Education"),

new Product(104, "Laptop", "Electronics")

};

System.out.println("Linear Search:");

Product result1 = SearchUtil.linearSearch(products, 102);

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

System.out.println("\nBinary Search:");

SearchUtil.sortProducts(products);

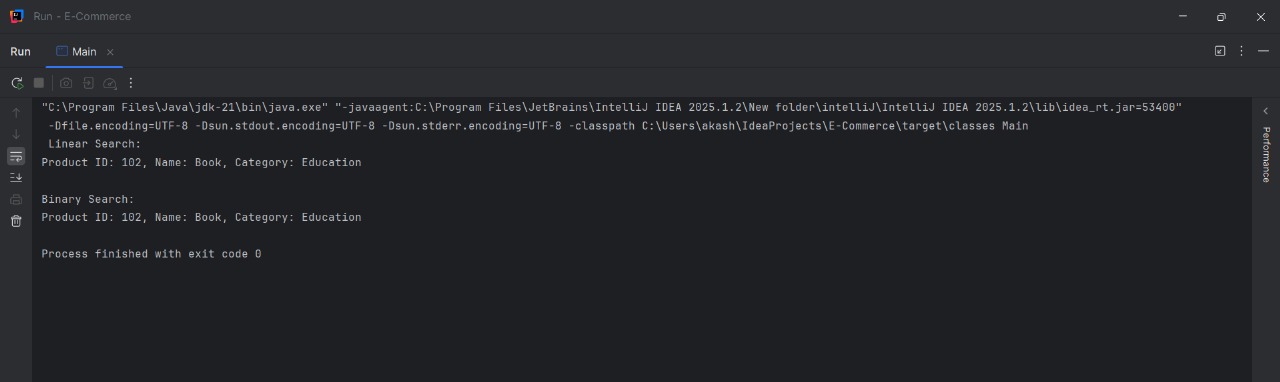
Product result2 = SearchUtil.binarySearch(products, 102);

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

}

}

OUTPUT:



4. Analysis

Time Complexity Comparison:

Algorithm Best Case Average Case

Linear Search O(1) O(n)

Binary Search O(1) O(log n)