Exercise 2: E-commerce Platform Search Function

1. Understand Asymptotic Notation

Big O notation is a mathematical way to describe how the performance of an algorithm changes as the size of the input increases. It allows for clear performance comparisons between algorithms by focusing on their behavior as inputs grow, independent of hardware or programming language specifics. In the context of search algorithms:

* **Linear Search:**
  + **Best Case:** O(1) — when the product is found at the very beginning.
  + **Average Case:** O(n) — typically, about half of the list is checked before finding the product.
  + **Worst Case:** O(n) — when the product is at the end or not present.
* **Binary Search:** (applicable only to sorted data)
  + **Best Case:** O(1) — when the middle element matches the target.
  + **Average Case:** O(log n)
  + **Worst Case:** O(log n)

1. Setup

public class Product {  
 private int productId;  
 private String productName;  
 private String category;  
  
 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; }  
  
 @Override  
 public String toString() {  
 return "Product ID: " + productId + ", Name: " + productName + ", Category: " + category;  
 }  
}

1. Implementation

import java.util.\*;  
  
public class ProductSearchDemo {  
  
 public static class Product {  
 private int productId;  
 private String productName;  
 private String category;  
  
 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; }  
  
 @Override  
 public String toString() {  
 return "Product ID: " + productId + ", Name: " + productName + ", Category: " + category;  
 }  
 }  
  
 public static Product linearSearch(Product[] products, int targetId) {  
 for (Product product : products) {  
 if (product.getProductId() == 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 - left) / 2;  
 if (products[mid].getProductId() == targetId) {  
 return products[mid];  
 } else if (products[mid].getProductId() < targetId) {  
 left = mid + 1;  
 } else {  
 right = mid - 1;  
 }  
 }  
 return null;  
 }  
  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(101, "Laptop", "Electronics"),  
 new Product(205, "Shoes", "Footwear"),  
 new Product(302, "Washing Machine", "Appliances"),  
 new Product(150, "Smartphone", "Electronics")  
 };  
  
   
 Arrays.sort(products, Comparator.comparingInt(Product::getProductId));  
  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Enter Product ID to search: ");  
 int targetId = scanner.nextInt();  
  
 System.out.println("\nSearching using Linear Search...");  
 Product foundLinear = linearSearch(products, targetId);  
 if (foundLinear != null) {  
 System.out.println("Product Found (Linear): " + foundLinear);  
 } else {  
 System.out.println("Product not found using Linear Search.");  
 }  
  
 System.out.println("\nSearching using Binary Search...");  
 Product foundBinary = binarySearch(products, targetId);  
 if (foundBinary != null) {  
 System.out.println("Product Found (Binary): " + foundBinary);  
 } else {  
 System.out.println("Product not found using Binary Search.");  
 }  
 }  
}

1. Analysis

Performance comparison of the two search methods:

| 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) |

**Conclusion:** For e-commerce platforms handling large product datasets, binary search provides greater efficiency due to its logarithmic performance on sorted data. Although linear search is straightforward, its performance degrades as the dataset grows.

