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

**1. Understanding Asymptotic Notation**

**Big O Notation:** Big O notation is a way to describe how an algorithm's time or space grows as input size increases. It's useful in analyzing and comparing algorithm efficiency, especially for large datasets.

Examples:

* O(1) – Constant time
* O(n) – Linear time
* O(log n) – Logarithmic time
* O(n^2) – Quadratic time

|  |  |  |  |
| --- | --- | --- | --- |
| Search Type | Best Case | Average Case | Worst Case |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

**2. Setup – Product Class**

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 productId + " - " + productName + " (" + category + ")";

}

}

**3. Full Implementation with Output Time Complexity**

import java.util.\*;

public class Ecommerce {

public static void main(String[] args) {

Product[] products = {

new Product(105, "Shoes", "Footwear"),

new Product(102, "Laptop", "Electronics"),

new Product(108, "Watch", "Accessories"),

new Product(101, "Phone", "Electronics"),

new Product(103, "Bag", "Travel")

};

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

Product result1 = linearSearch(products, 103);

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

System.out.println("Time Complexity: O(n)\n");

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

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

Product result2 = binarySearch(products, 103);

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

System.out.println("Time Complexity: O(log n)");

}

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

for (Product p : products) {

if (p.productId == id) {

return p;

}

}

return null;

}

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

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

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

return products[mid];

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

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

}

**Which is Better and Why?**

* Use **Linear Search** for simplicity when data is small or unsorted.
* Use **Binary Search** when data is large and already sorted, as it performs faster (logarithmic time).

**Conclusion**

Choosing between linear and binary search depends on data size and whether the data is sorted. In real-world e-commerce platforms, binary search is preferred for performance, but more advanced structures (like hash maps or databases with indexes) are used in production systems.

