**E-commerce Platform Search Function**

**Big O Notation**

Big O notation describes the upper bound of an algorithm's running time as the input size grows. It helps analyze how efficiently an algorithm scales and allows comparison independent of hardware.

For example:

* O(1) — Constant time
* O(n) — Linear time
* O(log n) — Logarithmic time

Best, Average, and Worst Case in Search

| Case | Description |
| --- | --- |
| Best | The item is found early (e.g., first element). |
| Average | The item is somewhere in the middle. |
| Worst | The item is not present or at the last position. |

**Setup**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;

}

}

**Implementation**

Linear Search  
public static Product linearSearch(Product[] products, String targetName) {

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null; }

Binary Search  
import java.util.Arrays;

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

// Binary Search assumes sorted array

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

int compare = products[mid].productName.compareToIgnoreCase(targetName);

if (compare == 0)

return products[mid];

else if (compare < 0)

low = mid + 1;

else

high = mid - 1;

}

return null;

}

Main method

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Shoes", "Footwear"),

new Product(103, "Book", "Stationery"),

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

new Product(105, "Watch", "Accessories")

};

// Sort the array by productName

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

// Display sorted products

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

for (Product p : products) {

System.out.println(p);

}

System.out.println("\n--- Search Results ---");

// Linear Search Test

String searchName = "Book";

Product result1 = linearSearch(products, searchName);

System.out.println("\nLinear Search Result for \"" + searchName + "\":");

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

// Binary Search Test

String searchName2 = "Watch";

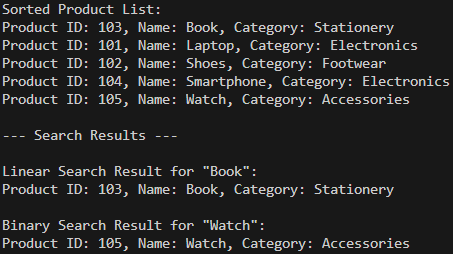
Product result2 = binarySearch(products, searchName2);

System.out.println("\nBinary Search Result for \"" + searchName2 + "\":");

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

}

**Output**

****

**Analysis**

| Algorithm | Time Complexity | Description |
| --- | --- | --- |
| Linear Search | Best: O(1), Avg/Worst: O(n) | Checks each element one by one |
| Binary Search | Best: O(1), Avg/Worst: O(log n) | Efficient on sorted arrays |