**Exercise 1: E-commerce Platform Search Function**

**Objective:**

Implement and analyze linear search and binary search algorithms for optimizing search functionality in an E-commerce Platform, with a focus on performance and time complexity. This helps simulate how actual online stores handle real-time product searches.

**Step 1:** Asymptotic Notation

In asymptotic notation Big O notation plays a key role.

Big O Notation helps to estimate how an algorithm’s runtime grows as the input size increases.

For Example:

* O(1): instant access like opening your fridge to grab a bottle.
* O(n): checks each item one by one like flipping every page in book.
* O(log n): narrows down possibilities quickly like guessing a number between 1 – 100 in fewer steps.

**Searching Algorithms:**

Essential tools to locate specific items within a collection of data.

1. **Linear Search:**

* This search is used for unsorted array.
* It does one-by-one element comparison.

1. **Binary Search:**

* It is applied on sorted array.
* First, compares middle element first and if key matches it return item.
* Else compares left and right half.
* Binary search is faster than linear search.

**Case Based Analysis:**

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Linear Search** | **Binary Search** |
| Best Case | O(1) | O(1) |
| Average Case | O(n) | O(logn) |
| Worst Case | O(n) | O(logn) |

**Step 2:** setup a class

Each product in e-commerce site needs to have essential attributes --- a productId, productName and category. These are properties for performing with search algorithms.

Create a class Product

i.e. **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 productId + ", " + productName + ", " + category ;  }  } |

**Step 3: Implementation of search algorithms**

Imagine searching for a productName in a list of items by reading from top to bottom.

1. Using Linear Search:

|  |
| --- |
| public static int linear\_Search (Product[] products, String name) {  for (int i = 0; i < products.length; i++) {  if (products[i].productName.equalsIgnoreCase (name)) {  return i;  }  }  return -1;  } |

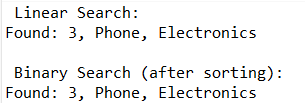
1. Using Binary Search:

|  |
| --- |
| public static int binary\_Search (Product[] products, String name) {  int low = 0, high = products.length - 1;  while (low <= high) {  int mid = (low + high) / 2;  int compare = products[mid].productName.compareToIgnoreCase(name);  if (compare == 0) return mid;  else if (compare < 0) low = mid + 1;  else high = mid - 1;  }  return -1;  } |

**Main.java:**

|  |
| --- |
| public class Main {  public static void sortProductsByName (Product[] products) {  Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));  }  public static void main (String[] args) {  Product[] products = {  new Product (1, "Laptop", "Electronics"),  new Product (2, "Shoes", "Fashion"),  new Product (3, "Phone", "Electronics"),  new Product (4, "Watch", "Accessories"),  new Product (5, "Bag", "Fashion")  };  System.out.println(" Linear Search: ");  int result1 = linear\_Search (products, "Phone");  if (result1 != -1) {  System.out.println("Found: " + products[result1]);  } else {  System.out.println("Product not found.");  }  sortProductsByName(products);  System.out.println("\n Binary Search:");  int result2 = binary\_Search (products, "Phone");  if (result2 != -1) {  System.out.println("Found: " + products[result2]);  } else {  System.out.println("Product not found.");  }  }  } |

**Expected Output:**

****

**Step 4: Analysis**

Performance comparison:

|  |  |  |
| --- | --- | --- |
| Feature | Linear Search | Binary Search |
| Requires sorting | No | Yes |
| Impact on large data | slow | Fast |
| Flexibility | No fixed order required | Needs data to be sorted |
| Usage | List is small / unsorted | List is large and sorted |

From this we can conclude that Linear Search is simple but slow for large datasets. Whereas, Binary Search is fast i.e. O(logn) but the array needs to be sorted. So, for an e-commerce platform, Binary Search is most suitable as product lists can be sorted and searched quickly. Hence user experience and performance will be improved.