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

**Big O Notation** is used to describe the upper bound of an algorithm's time and space complexity relative to the input size n. It helps evaluate how efficiently an algorithm performs, especially as the input grows. Big O focuses on three common scenarios:

* **Best Case**: The element is found immediately (e.g., first position).
* **Average Case**: The element is found somewhere in the middle.
* **Worst Case**: The element is found at the end or not found at all.

**Code:**

import java.util.Arrays;

import java.util.Comparator;

public class EcommerceExample {

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

for(Product product:products) {

if(product.productName==targetName) {

return product;

}

}

return null;

}

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

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

while(left<=right) {

int mid=left+(right-left)/2;

int result=products[mid].productName.compareTo(targetName);

if(result==0) return products[mid];

else if(result<0)left=mid+1;

else right=mid-1;

}

return null;

}

public static void main(String[] args) {

Product[] products={ new Product(1,"Mobile","Electronics"),new

Product(2,"Laptop","Electronics"),new

Product(3,"Shirt","Apparel"),new

Product(4,"Shoes","Footwear"),new

Product(5,"Watch","Accessories")};

Product result1=linearSearch(products,"Phone");

if(result1!=null) System.out.println("Linear Search Result: "+result1.productName+" -Category: "+result1.category);

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

Product result2=binarySearch(products,"Phone");

if(result2!=null) System.out.println("Binary Search Result: "+result2.productName+" -Category: "+result2.category);

}

}

class Product {

int productId;

String productName;

String category;

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

{

this.productId=productId;

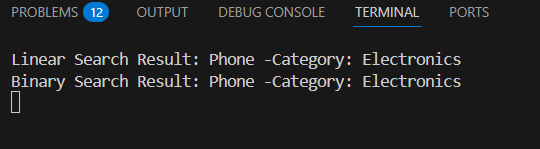
this.productName=productName;

this.category=category;

}

}

**Output:**

****

**Time Complexity of linear search and binary search**

**Linear Search:**

Best case: O(1)

Average case: O(n/2)

Worst case: O(n)

**Binary Search:**

Best case: O(1)

Average case: O(log n)

Worst case: O(log n)

**Best Algorithm to use :binary search** is the preferred algorithm. While **linear search** works well for small datasets, it becomes inefficient as the application scales and the volume of data increases. Since e-commerce platforms are dynamic and data-heavy, **binary search offers better performance and scalability**—making it the optimal choice for large datasets due to its logarithmic time complexity.