**E-COMMERCE PLATFORM SEARCH FUNCTION**

SCENARIO

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

UNDERSTANDING ASYMPTOTIC NOTATION

**Big O notation**is a tool used to describe the time complexity or space complexity of algorithms. **Big-O** is a way to express the **upper bound**of an algorithm’s time or space complexity. It can be used to compare the efficiency of different algorithms or data structures.

Search operations:

Best Case scenario: Immediate match

Average Case scenario: Match found mid-way

Worst Case scenario: Last element or not found

IMPLEMENTATION

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: " + productId + ", Name: " + productName + ", Category: " + category;

    }

}

Search.java

Public class Search{

    public Product linearSearch(Product[] arr, String key){

        for(int i = 0 ; i < arr.length ; i++){

            if(arr[i].productName.equalsIgnoreCase(key)){

                return arr[i];

            }

        }

        return null;

    }

    public Product binarySearch(Product[] arr, String key){

        int low = 0;

        int high = arr.length - 1;

        int mid;

        while(low <= high){

            mid = (low + high) / 2;

            if(arr[mid].productName.equalsIgnoreCase(key)){

                return arr[mid];

            }

            else if(arr[mid].productName.compareToIgnoreCase(key) > 0){

                high = mid - 1;

            }

            else{

                low = mid + 1;

            }

        }

        return null;

    }

    public static void main(String[] args){

        Search search = new Search();

        Product p1 = new Product(01, "Laptop", "Electronics");

        Product p2 = new Product(02, "Scissors", "Stationary");

        Product p3 = new Product(03, "Headset", "Electronics");

        Product p4 = new Product(04, "Lamp", "Home Decor");

        Product p5 = new Product(05, "Lip balm", "Beauty");

        Product[] productList = {p1, p2, p3, p4, p5};

        System.out.println(search.linearSearch(productList, "laptop"));

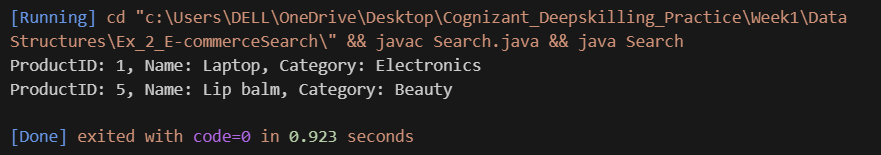
        Arrays.sort(productList, Comparator.comparing(p -> p.productName));

        System.out.println(search.binarySearch(productList, "lip balm"));

    }

}

OUTPUT



ANALYSIS

* Time complexity comparison:

Linear search: Best case - O(1)

Average case - O(n)

Worst case – O(n)

Binary search: Best case - O(1)

Average case - O(log n)

Worst case – O(log n)

|  |
| --- |
|  |

* Binary search is more suitable because E-commerce platforms may have thousands of items and binary search scales better than linear search.