Hands on session

Algorithms Data Structure

**E-commerce Platform Search Function**

**Definition**

An e-commerce platform search function is a feature that allows users to search for products by entering queries like product name, ID, or category.It helps users quickly find items they are interested in by scanning through the product data.

**Steps**

* Define Product Class
* Create Product Array
* Sort Array for Binary Search
* Implement Linear Search
* Implement Binary Search
* Take Input and Perform Search
* Display Results
* Time Complexity Analysis

**Code**

import java.util.Arrays;

import java.util.Scanner;

public class EcommerceSearch {

static 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 "Product ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

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

for (Product p : products) {

if (p.productId == targetId) {

return p;

}

}

return null;

}

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

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

while (left <= right) {

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

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

return products[mid];

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

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(205, "Shirt", "Clothing"),

new Product(150, "Book", "Education"),

new Product(305, "Phone", "Electronics"),

new Product(110, "Shoes", "Footwear")

};

Product[] sortedProducts = Arrays.copyOf(products, products.length);

Arrays.sort(sortedProducts, (a, b) -> a.productId - b.productId);

Scanner scanner = new Scanner(System.in);

System.out.print("Enter Product ID to search: ");

int searchId = scanner.nextInt();

Product result1 = linearSearch(products, searchId);

System.out.println("\n🔍 Linear Search Result:");

if (result1 != null)

System.out.println(result1);

else

System.out.println("Product not found.");

Product result2 = binarySearch(sortedProducts, searchId);

System.out.println("\n🔍 Binary Search Result:");

if (result2 != null)

System.out.println(result2);

else

System.out.println("Product not found.");

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

System.out.println("Linear Search: O(n) - checks every product.");

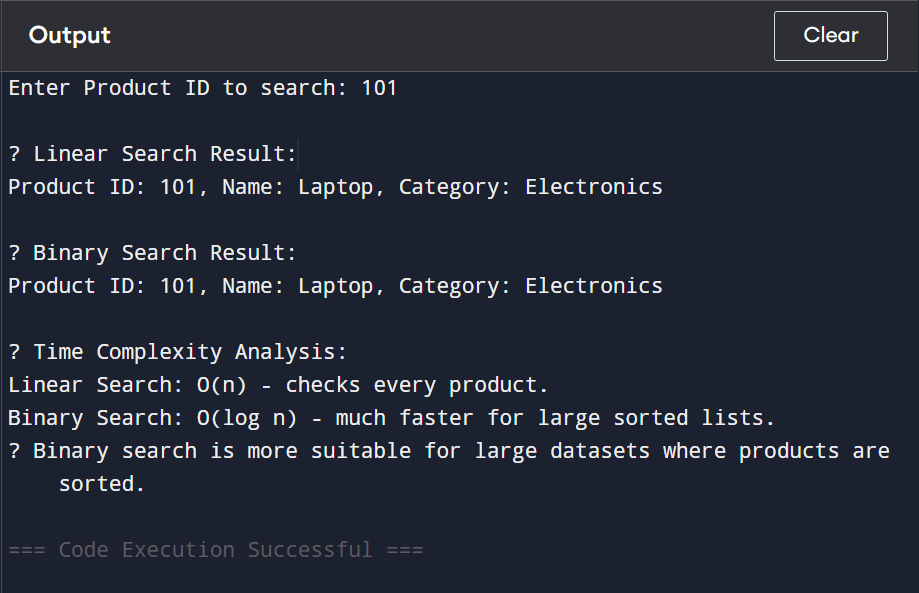
System.out.println("Binary Search: O(log n) - much faster for large sorted lists.");

System.out.println("✅ Binary search is more suitable for large datasets where products are sorted.");

}

}

**Output**



**Summary**

* **Linear Search** and **Binary Search** — to find a product by its productId.
* Products are stored in an array.
* Linear search scans each item one-by-one.
* Binary search works on a **sorted array** and finds items faster
* The user enters a product ID, and the program searches using both methods
* It displays the search results and compares the **time complexity** of both algorithms.

**Financial Forecasting**

**Definition**

Financial forecasting is the process of estimating or predicting a business's future financial outcomes based on historical data, such as past revenue growth, interest rates, or investment returns.

We use Recursion Method to solve

Recursion is a programming technique where a method calls itself to solve a problem by breaking it down into smaller subproblems, until a base case is reached.

**Steps**

* Understand Recursive Algorithms
* Create a method predictFutureValue() with inputs
* Implementation
* Analysis of Time Complexity and Optimization

**Code**

public class FinancialForecasting {

public static double predictFutureValue(double presentValue, double growthRate, int years) {

if (years == 0) {

return presentValue; // Base case

}

return (1 + growthRate) \* predictFutureValue(presentValue, growthRate, years - 1);

}

public static void main(String[] args) {

double presentValue = 10000.0;

double growthRate = 0.10;

int years = 5;

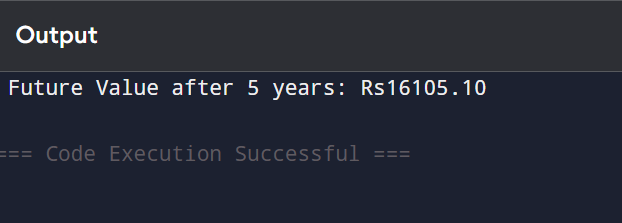
double futureValue = predictFutureValue(presentValue, growthRate, years);

System.out.println(" Future Value after " + years + " years: Rs" + String.format("%.2f", futureValue));

}

}

**Output**



**Summary**

* The code predicts future financial value using **recursion**.
* It follows the compound interest growth formula.
* Time complexity is **O(n)**, and for better efficiency, recursion can be replaced by iteration or direct formula.