**Exercise 2: E-commerce Platform Search Function**

Solution:

1)Product Class:

**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;

}

}

2)Implementation of Linear Search And Binary Search

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** Search {

**public** **static** String linearSearch(Product[] arr, String target) {

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

**if** (arr[i].productName.equalsIgnoreCase(target)) {

**return** "Found";

}

}

**return** "Not Found";

}

**public** **static** String binarySearch(Product[] arr, String target) {

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

**int** left = 0;

**int** right = arr.length - 1;

**while** (left <= right) {

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

String midName = arr[mid].productName;

**int** compare = target.compareToIgnoreCase(midName);

**if** (compare == 0) {

**return** "Found";

} **else** **if** (compare > 0) {

left = mid + 1;

} **else** {

right = mid - 1;

}

}

**return** "Not Found";

}

}

3)Test Class:

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Product[] stock = **new** Product[4];

stock[0] = **new** Product(1, "Mobile", "Electronics");

stock[1] = **new** Product(2, "Tablet", "Electronics");

stock[2] = **new** Product(3, "Shirt", "Clothing");

stock[3] = **new** Product(4, "Pant", "Clothing");

String targetA = "Shirt";

String targetB = "Tablet";

//first product

String lA = Search.*linearSearch*(stock, targetA);

String bA = Search.*binarySearch*(stock, targetA);

System.***out***.println(lA);

System.***out***.println(bA);

//second product

String lB = Search.*linearSearch*(stock, targetB);

String bB = Search.*binarySearch*(stock, targetB);

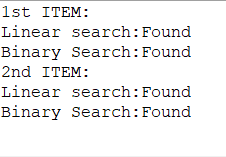
System.***out***.println(lB);

System.***out***.println(bB);

}

}

OUTPUT:



Analysis: Binary Search is more effective as the time complexity of the Binary Search is O(log n) given that the array is sorted in this case Product[],whereas Linear Search has time complexity of O(n).

**Exercise 7: Financial Forecasting**

**Solution:**

We know Recursion occurs when a function calls itself, here to calculate the future value we know

FV=PV×(1+r)^n, Hence Recursion can be implemented here to simplify the problem.

Implementation of Recursive Algorithm:

**public** **class** Financial\_Forecast {

**public** **static** **double** forecast(**double** amount, **double** rate, **int** years) {

**if** (years == 0) {

**return** amount;

}

**return** *forecast*(amount,rate, years-1)\*(1 + rate);

}

**public** **static** **void** main(String[] args) {

**double** p = 1000;

**double** rate = 0.8;

**int** time = 5;

**double** result = *forecast*(p, rate, time);

System.***out***.println("Future Value(in Rs) after "+time+ " years: "+result);

}

}

OUTPUT:



Here’s, the Time complexity of the recursive code is O(n), but in recursion extra stack space is also used hence space complexity is O(n), therefore the code can be optimised by using iterative approach which will reduce the space complexity to O(1), but time complexity will remain O(n).