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

Big O Notation expresses the upper bound of an algorithm's running time in terms of input size n. It helps developers to predict performance.Choose the most efficient algorithm as n grows**.**

For Linear Search:

Best Case:O(1)

Worst Case:O(n)

Average Case: O(n/2) ≈ O(n)

For Binary Search:

Best Case:O(1)

Worst Case:O(log n)

Average Case: O(log n)

**CODE:**

import java.util.\*;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product("P101","Shoes","Footwear"),

new Product("P102","T-shirt","Clothing"),

new Product("P103","Laptop","Electronics"),

new Product("P104","Book","Stationery")

};

if(linearSearch(products,"Laptop")) System.out.println("Product exists from linear Search");

else System.out.println("Product does not exists from linear Search");

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

if(binarySearch(products,"Laptop")) System.out.println("Product exists from binary Search");

else System.out.println("Product does not exists from binary Search");

}

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

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

if(products[i].productName.equalsIgnoreCase(name)==true) return true;

}

return false;

}

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

int low=0,high=products.length-1;

while (low <= high) {

int mid=(low+high)/2;

int cmp=name.compareToIgnoreCase(products[mid].productName);

if (cmp==0) return true;

else if(cmp<0)high=mid-1;

else low=mid+1;

}

return false;

}

}

class Product {

String productId;

String productName;

String category;

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

this.productId = productId;

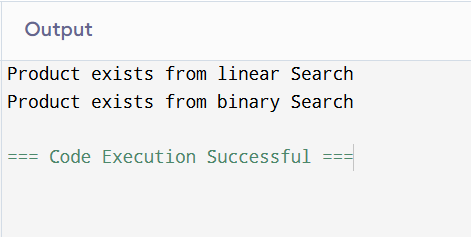
this.productName = productName;

this.category = category;

}

}

OUTPUT:



Analysis:

Time Complexity:

**For Linear Search:**

Best Case:O(1)

Worst Case:O(n)

Average Case: O(n/2) ≈ O(n)

**For Binary Search:**

Best Case:O(1)

Worst Case:O(log n)

Average Case: O(log n)

Linear Search is compatible for small data sets and does not require sorted data set.Whereas,Binary search require sorting and is very much useful for large data set.It is fast efficient.Therefore Bianry search is more compatible.

**Exercise 7: Financial Forecasting**

Recursion is a programming technique where a method calls itself to solve smaller instances of a problem.

For Future Financial Forecasting,

Recurrence relation:F(n)=F(n-1)\*(1+rate)

Here n is no of years

**Code:**

import java.util.\*;

public class Main {

public static void main(String[] args) {

double principal=10000;

double rate=0.08;

int years=10;

System.out.println("The Financial Forecast value after "+years+" years is "+financialforecast(principal,rate,years));

}

public static double financialforecast(double principal,double rate,int years){

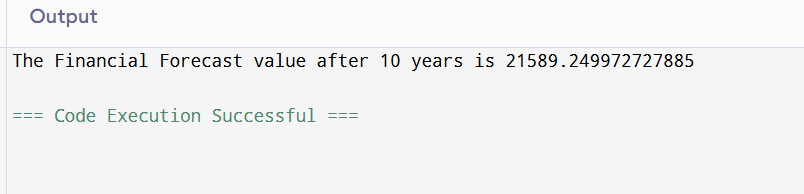
if(years==0) return principal;

return financialforecast(principal,rate,years-1)\*(1+rate);

}

}

**Output:**



**Analysis:**

Time Complexity:O(n)

Space Complexity:O(n) // Auxilliary Stack Space

To reduce the auxiliary stack space we can switch to tabulation or iterative space optimized methods