# Cognizant Digital Nurture 4.0 Deep Skilling

## Algorithms Data Structures

**Exercise 2: 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.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**Solution :**

import java.util.Arrays;

import java.util.Comparator;

public class ECommerceSearch {

static class Product {

int productId;

String productName;

String category;

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

this.productId = productId;

this.productName = productName;

this.category = category;

}

@Override

public String toString() {

return "[" + productId + ", " + productName + ", " + category + "]";

}

}

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

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

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

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

while (low <= high) {

int mid = (low + high) / 2;

int comparison = products[mid].productName.compareToIgnoreCase(targetName);

if (comparison == 0) {

return products[mid];

} else if (comparison < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(101, "Shoes", "Footwear"),

new Product(102, "Laptop", "Electronics"),

new Product(103, "Shirt", "Clothing"),

new Product(104, "Headphones", "Electronics"),

new Product(105, "Phone", "Electronics")

};

String searchTerm = "Phone";

Product result1 = linearSearch(products, searchTerm);

System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Not Found"));

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

Product result2 = binarySearch(products, searchTerm);

System.out.println("Binary Search Result: " + (result2 != null ? result2 : "Not Found"));

}

}

**Output :**

Linear Search Result: [105, Phone, Electronics]

Binary Search Result: [105, Phone, Electronics]

**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

**Solution :**

public class FinancialForecast {

public static double futureValueRecursive(double initialAmount, double growthRate, int years) {

if (years == 0) {

return initialAmount;

}

return futureValueRecursive(initialAmount, growthRate, years - 1) \* (1 + growthRate);

}

public static double futureValueIterative(double initialAmount, double growthRate, int years) {

double result = initialAmount;

for (int i = 1; i <= years; i++) {

result \*= (1 + growthRate);

}

return result;

}

public static void main(String[] args) {

double initialAmount = 10000;

double annualGrowthRate = 0.08;

int years = 5;

double fvRecursive = futureValueRecursive(initialAmount, annualGrowthRate, years);

double fvIterative = futureValueIterative(initialAmount, annualGrowthRate, years);

System.out.printf("Future Value (Recursive): ?%.2f%n", fvRecursive);

System.out.printf("Future Value (Iterative): ?%.2f%n", fvIterative);

}

}

**Output :**

Future Value (Recursive): ₹14693.28

Future Value (Iterative): ₹14693.28