**Week-1 Data Structures and Algorithm  
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**

**Linear Search**

**Product.java**

package LinearSearch;

public class Product {

int productid;

String productname;

String category;

public Product(int productid, String productname, String category) {

super();

this.productid = productid;

this.productname = productname;

this.category = category;

}

}

**ProductMain.java**

package LinearSearch;

import java.util.Scanner;

public class ProductMain {

public static void main(String[] args) {

Product arr[]=new Product[5];

Product p1=new Product(1, "Pen", "Stationary");

Product p2=new Product(2, "Fan", "Electronics");

Product p3=new Product(3, "Note", "Stationary");

Product p4=new Product(4, "Car", "Automobile");

Product p5=new Product(5, "pencil", "Stationary");

arr[0]=p1;

arr[1]=p2;

arr[2]=p3;

arr[3]=p4;

arr[4]=p5;

Scanner scanner=new Scanner(System.***in***);

System.***out***.println("Search Product By \n1. Id\n2. Name\n3. Category");

int searchBy=scanner.nextInt();

switch(searchBy) {

case 1:

System.***out***.println("Enter the id");

int id=scanner.nextInt();

SearchById si=new SearchById(arr,id);

break;

case 2:

System.***out***.println("Enter the name");

String name=scanner.next();

SearchByName sn=new SearchByName(arr,name);

break;

case 3:

System.***out***.println("Enter the category");

//scanner.next();

String category=scanner.next();

SearchByCategory sc=new SearchByCategory(arr,category);

break;

}

scanner.close();

}

}

**SearchById.java**

package LinearSearch;

public class SearchById {

public SearchById(Product arr[],int id) {

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

if(arr[i].productid ==id) {

System.***out***.println("Id: "+arr[i].productid +" Name: "+ arr[i].productname+ " Category: "+ arr[i].category);

}

}

}

}

**SearchByCategory.java**

package LinearSearch;

public class SearchByCategory {

public SearchByCategory(Product[] arr, String category) {

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

if(arr[i].category.equalsIgnoreCase(category)) {

System.***out***.println("Id: "+arr[i].productid +" Name: "+ arr[i].productname+ " Category: "+ arr[i].category);

}

}

}

}

**SearchByName.java**

package LinearSearch;

public class SearchByName {

public SearchByName(Product[] arr, String name) {

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

if(arr[i].productname.equalsIgnoreCase(name)) {

System.***out***.println("Id: "+arr[i].productid +" Name: "+ arr[i].productname+ " Category: "+ arr[i].category);

}

}

}

}

**BINARY SEARCH**

**Product.java**

package BinarySearch;

public class Product implements Comparable<Product>{

int productid;

String productname;

String category;

public Product(int productid, String productname, String category) {

super();

this.productid = productid;

this.productname = productname;

this.category = category;

}

*@Override*

public int compareTo(Product other) {

return Integer.*compare*(this.productid, other.productid); // Sort by ID

}

}

**ProductMain.java**

package BinarySearch;

import java.util.Arrays;

import java.util.Scanner;

public class ProductMain {

public static void main(String[] args) {

Product arr[]=new Product[5];

Product p1=new Product(1, "Pen", "Stationary");

Product p2=new Product(2, "Fan", "Electronics");

Product p3=new Product(3, "Note", "Stationary");

Product p4=new Product(4, "Car", "Automobile");

Product p5=new Product(5, "pencil", "Stationary");

arr[0]=p1;

arr[1]=p3;

arr[2]=p2;

arr[3]=p5;

arr[4]=p4;

Scanner scanner=new Scanner(System.in);

System.out.println("Search Product By \n1. Id\n2. Name\n3. Category");

int searchBy=scanner.nextInt();

switch(searchBy) {

case 1:

System.out.println("Enter the id");

int id=scanner.nextInt();

Arrays.sort(arr);

SearchById si=new SearchById(arr,id);

break;

case 2:

System.out.println("Enter the name");

String name=scanner.next();

SearchByName sn=new SearchByName(arr,name);

break;

case 3:

System.out.println("Enter the category");

//scanner.next();

String category=scanner.next();

SearchByCategory sc=new SearchByCategory(arr,category);

break;

}

scanner.close();

}

}

**SearchById.java**

package BinarySearch;

public class SearchById {

public SearchById(Product arr[],int id) {

int low=0;

int high=arr.length-1;

while(low<=high) {

int mid=(low+high)/2;

if(arr[mid].productid==id) {

System.***out***.println("ID: "+arr[mid].productid+" Name: "+arr[mid].productname+" Category: "+arr[mid].category);

return;

}

else if(arr[mid].productid<id) {

low=mid+1;

}

else {

high=mid-1;

}

}

}

}

**SearchByName.java**

package BinarySearch;

import java.util.Arrays;

import java.util.Comparator;

public class SearchByName {

public SearchByName(Product[] arr, String name) {

Arrays.*sort*(arr,new Comparator<Product>() {

public int compare(Product p1,Product p2) {

return p1.productname.compareToIgnoreCase(p2.productname);

}

});

int low=0;

int high=arr.length-1;

while(low<=high) {

int mid=(low+high)/2;

if(arr[mid].productname.compareToIgnoreCase(name)==0) {

System.***out***.println("ID: "+arr[mid].productid+" Name: "+arr[mid].productname+" Category: "+arr[mid].category);

return;

}

else if(arr[mid].productname.compareToIgnoreCase(name)<0) {

low=mid+1;

}

else {

high=mid-1;

}

}

}

}

**SearchByCategory.java**

package BinarySearch;

import java.util.Arrays;

import java.util.Comparator;

public class SearchByCategory {

public SearchByCategory(Product[] arr, String category) {

Arrays.*sort*(arr, new Comparator<Product>() {

*@Override*

public int compare(Product p1, Product p2) {

return p1.category.compareToIgnoreCase(p2.category);

}

});

int low=0;

int high=arr.length-1;

while(low<=high) {

int mid=(low+high)/2;

if(arr[mid].category.compareToIgnoreCase(category)==0) {

System.***out***.println("ID: "+arr[mid].productid+" Name: "+arr[mid].productname+" Category: "+arr[mid].category);

return;

}

else if(arr[mid].category.compareToIgnoreCase(category)<0) {

low=mid+1;

}

else {

high=mid-1;

}

}

}

}

**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**

**FutureRecc.java**

package forecasting;

import java.util.Scanner;

public class FutureRecc {

public double futureVal(double initial,double growth, int year) {

if(year==0) {

return initial;

}

return futureVal(initial, growth, year-1) \*(1+growth);

}

public static void main(String[] args) {

Scanner scanner=new Scanner(System.***in***);

System.***out***.println("Enter the Initial value ");

double initial=scanner.nextDouble();

System.***out***.println("Enter the Growth value ");

double growth=scanner.nextDouble();

System.***out***.println("Enter the year ");

int year=scanner.nextInt();

FutureRecc futureRecc=new FutureRecc();

double val=futureRecc.futureVal(initial, growth, year);

System.***out***.println("Future value after "+year+" "+val);

}

}