**E-commerce Platform Search Function**

## **Asymptotic Notation**

### **➤ Big O Notation:**

Big O notation describes the **upper bound of time or space complexity** in terms of input size n.

For example:

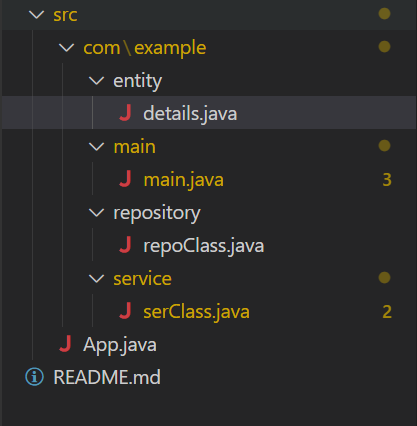
* O(1) – constant time
* O(n) – linear time
* O(log n) – logarithmic time (e.g., binary search)
* O(n log n) – linearithmic time (e.g., merge sort)
* O(n^2) – quadratic time

### **➤ Search Scenarios:**

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

* **Linear Search** scans each element one by one.
* **Binary Search** divides the array and works only on **sorted arrays**.

**CODE:**



details.java

package com.example.entity;

public class details {

int productId;

public String productName;

String category;

public details(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public void setProductId(int productId) {

this.productId = productId;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

@Override

public String toString() {

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

}

}

main.java

package com.example.main;

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

import com.example.entity.details;

import com.example.service.serClass;

public class main {

private static serClass ser = new serClass();

private static Scanner sc = new Scanner(System.in);

public static void main(String args[]) {

details[] product = {

new details(1, "lehenga", "dress"),

new details(2, "apsara", "pencil"),

new details(3, "ring", "ornaments"),

new details(4, "bat", "sports"),

new details(5, "frock", "dress")

};

System.out.println("Linear Search : ");

System.out.print("Enter the product name to search : ");

String pro1 = sc.nextLine();

details result1 = ser.linear(product, pro1);

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

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

System.out.println("Binary Search : ");

System.out.println("Enter the product name to search : ");

String pro2 = sc.nextLine();

details result2 = ser.binary(product, pro2);

System.out.println(result2 != null ? "Product Found = " + result2 : "Product not found");

}

}

repoClass.java

package com.example.repository;

import com.example.entity.details;

public class repoClass {

public static details linear(details detail[], String find) {

for (details d : detail) {

if (d.productName.equalsIgnoreCase(find)) {

return d;

}

}

return null;

}

public static details binary(details detail[],String find)

{

int left = 0;

int right = detail.length-1;

while(left <= right)

{

int mid = (left + right) / 2;

int cmp = detail[mid].productName.compareToIgnoreCase(find);

if(cmp == 0)

{

return detail[mid];

}

else if(cmp < 0)

{

left = mid + 1;

}

else

{

right = mid - 1;

}

}

return null;

}

}

serClass.java

package com.example.service;

import com.example.entity.details;

import com.example.repository.repoClass;

public class serClass {

private static repoClass repo = new repoClass();

public static details linear(details detail[], String find) {

return repo.linear(detail, find);

}

public static details binary(details detail[],String find)

{

return repo.binary(detail, find);

}

}

## **Analysis**

| **Search Method** | **Time Complexity** | **Sorted Required** | **Use Case** |
| --- | --- | --- | --- |
| Linear Search | O(n) | ❌ No | Small data, unsorted input |
| Binary Search | O(log n) | ✅ Yes | Large data, sorted list (e.g., autocomplete) |

## **Output :**

