**Product Search using Linear and Binary Search**

Program:

import java.util.Scanner;

class Product

{

    int productId;

    String productName , category;

    Product(int productId , String productName , String category)

    {

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

}

class ProductSearch

{

    public int linearSearch(Product[] p , int id)

    {

        int index= 0;

        for(Product p1 : p)

        {

            if(p1.productId == id)

                return index;

            else

                index++;

        }

        return -1;

    }

    public int binarySearch(Product[] p , int id)

    {

        int mid = p.length /2;

        if(p[mid].productId == id)

            return mid;

        else if(p[mid].productId < id)

        {

            for(int i = mid + 1 ; i < p.length ;i++)

            {

                if(p[i].productId == id)

                    return i;

            }

        }

        else if (p[mid].productId > id)

        {

            for(int i = 0 ; i< mid-1 ;i++)

            {

                if(p[i].productId == id)

                    return i;

            }

        }

        return -1;

    }

}

public class ProductSearchDriver {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        Product[] p = {

            new Product(1, "Book", "Education"),

            new Product(2, "Bat", "Sport"),

            new Product(3, "Ball", "Sport"),

            new Product(4, "Battery", "Electrical")

        };

        ProductSearch psearch = new ProductSearch();

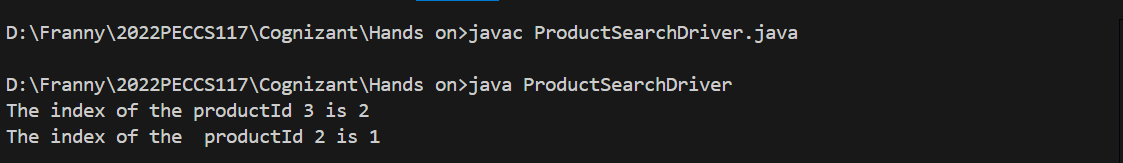
        System.out.println("The index of the productId 3 is " + psearch.binarySearch(p, 3));

        System.out.println("The index of the  productId 2 is " + psearch.linearSearch(p, 2));

    }

}

**Output:**

****

**ANALYSIS:**

**Time complexity of Linear Search:**

O (n): As the loop runs n times during worst case scenario.

**Time complexity of Binary Search:**

O (log n): As the loop splits in half during the worst case and from there on it searches only half the length of array.

And that is exactly why binary search is more efficient.