**Superset id:- 6363303**

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

**Code:-**

**using System;**

**using System.Linq;**

**public class Product**

**{**

**private int productId;**

**private string productName;**

**private string category;**

**public int ProductId**

**{**

**get**

**{**

**return productId;**

**}**

**set**

**{**

**productId=value;**

**}**

**}**

**public string ProductName**

**{**

**get**

**{**

**return productName;**

**}**

**set**

**{**

**productName=value;**

**}**

**}**

**public string Category**

**{**

**get**

**{**

**return category;**

**}**

**set**

**{**

**category=value;**

**}**

**}**

**public Product(int id,string name,string cat)**

**{**

**productId=id;**

**productName=name;**

**category=cat;**

**}**

**public override string ToString()**

**{**

**return $"{productId}:{productName} ({category})";**

**}**

**}**

**public class ECommerceSearch**

**{**

**public static Product? LinearSearch(Product[] products,string name)**

**{**

**foreach(var prod in products)**

**{**

**if(prod.ProductName.Equals(name,StringComparison.OrdinalIgnoreCase))**

**return prod;**

**}**

**return null;**

**}**

**public static Product? BinarySearch(Product[] products, string name)**

**{**

**int si=0;**

**int ei=products.Length-1;**

**while(si<=ei)**

**{**

**int mid=si+(ei-si)/2;**

**int check=string.Compare(products[mid].ProductName,name,StringComparison.OrdinalIgnoreCase);**

**if(check==0)**

**{**

**return products[mid];**

**}**

**else if(check<0)**

**{**

**si=mid+1;**

**}**

**else**

**{**

**ei=mid-1;**

**}**

**}**

**return null;**

**}**

**public static void Main()**

**{**

**Product[] products ={**

**new Product(1,"Dairy Milk","Food"),**

**new Product(2,"Mobile","Electronics"),**

**new Product(3,"Book","Education"),**

**new Product(4,"Chips","Food"),**

**};**

**Console.WriteLine("Linear Search:- ");**

**var result1 = LinearSearch(products, "Dairy Milk");**

**Console.WriteLine(result1!= null?$"Found:{result1}":"Not Found");**

**Console.WriteLine("\nBinary Search:-");**

**var sortedProducts=products.OrderBy(p=>p.ProductName).ToArray();**

**var result2 = BinarySearch(sortedProducts,"apple");**

**Console.WriteLine(result2!=null?$"Found:{result2}":"Not Found");**

**Console.WriteLine("\nBinary Search:-");**

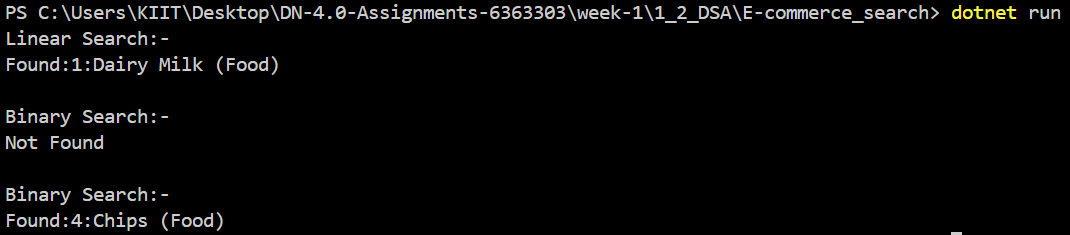
**var result3=BinarySearch(sortedProducts,"Chips");**

**Console.WriteLine(result3!=null?$"Found:{result3}":"Not Found");**

**}**

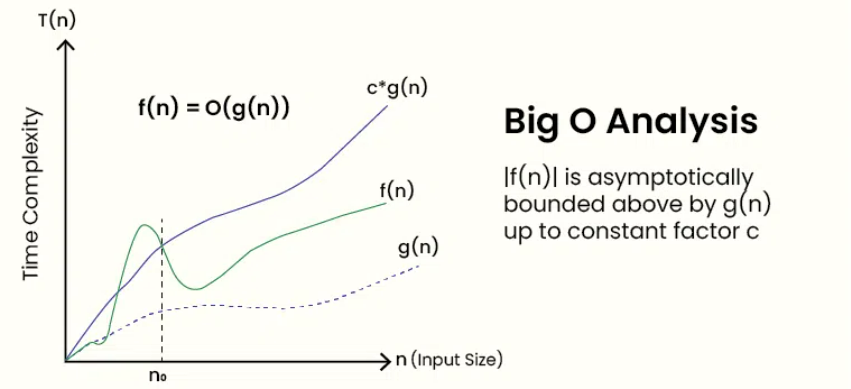
**}**

**Output:-**

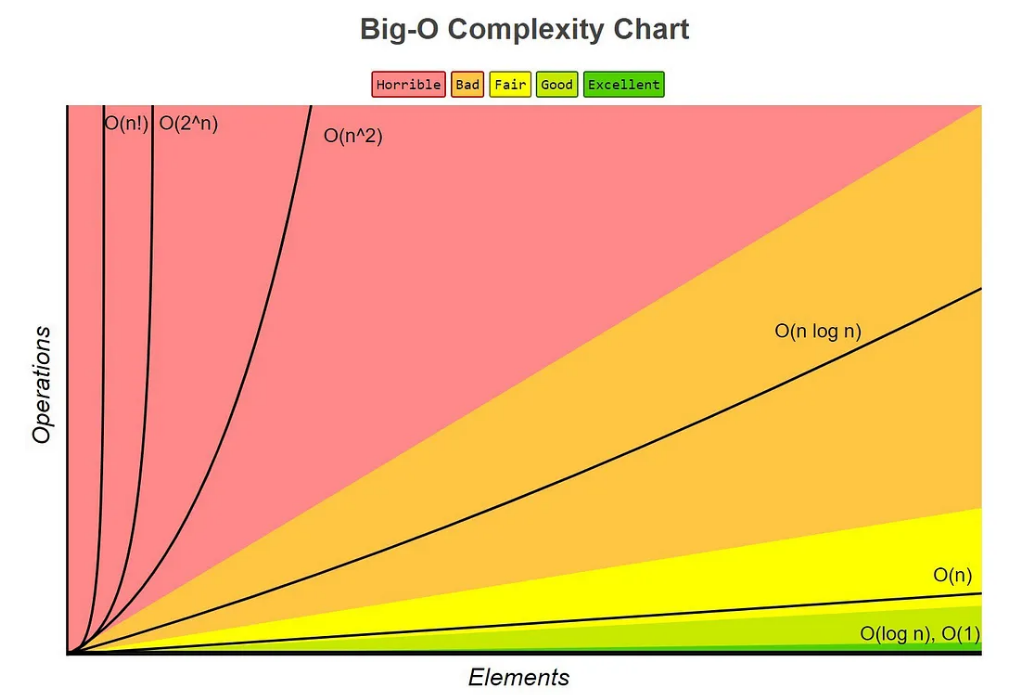
****

**Big O Notation:-**

Big O Notation is a way to describe how fast or slow an algorithm is. It helps us understand how the time or memory needed by a program grows when the amount of data increases.



**Complexity Chart:-**

****

**Best, Average, Worst Case:-**

Best Case – The fastest it can run (e.g., the item you're searching for is the first one).

Average Case – What usually happens with random inputs.

Worst Case – The slowest it can run (e.g., the item is last or not there at all).

**For Linear Search:-**

* **Best Case -> O(1)**
* **Average Case -> O(N)**
* **Worst Case -> O(N)**

**For Binary Search:-**

* **Best Case -> O(1)**
* **Average Case -> O(Log N)**
* **Worst Case -> O(Log N)**

**Which is Better For E-commerce Platform?**

* **For an e-commerce platform, binary search is more suitable because it's faster (O(Log N)) and works well with sorted data, which is common in product lists. It provides quick results especially when there are many products.**
* **Linear search (O(N)) can still be useful for small or unsorted data, but it's slower for large datasets.**
* **Overall, binary search is better for performance and is ideal when the product list is sorted.**

**Ayush Kumar (6363303)**