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.

      Big O describes the **upper bound** of an algorithm's running time — essentially how it grows as the input size increases.

- O(1): Constant time
- O(n): Linear time
- O(log n): Logarithmic time
- O(n^2): Quadratic time

It helps in comparing algorithms independent of hardware or programming language.

- Describe the best, average, and worst-case scenarios for search operations.

  For Linear and Binary Search:

  the scenario for Best case is O(1)

  for Average case it's O(n) for linear and O(logn) for binary search.

  for Worst case scenario it's O(n) for Linear and O(logn) for binary search.

2. **Setup:**
    - Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.

Startup Code:

```
public class Product
{
   public int ProductId { get; set; }
   public string ProductName { get; set; }
   public string Category { get; set; }

   public Product(int productId, string productName, string category)
   {
```

```
        ProductId = productId;
        ProductName = productName;
        Category = category;
    }

    public override string ToString()
    {
        return $"[{ProductId}, {ProductName}, {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.

Implementaion Code:-

```
//Code starts
using System;
using System.Linq;

public class SearchUtility
{
    // Linear search
    public static Product LinearSearch(Product[] products, string name)
    {
        foreach (var p in products)
        {
            if (p.ProductName.Equals(name, StringComparison.OrdinalIgnoreCase))
            {
                return p;
            }
```

```
        }
        return null;
    }

    // Binary search (requires sorted array)
    public static Product BinarySearch(Product[] products, string name)
    {
        Array.Sort(products, (p1, p2) => string.Compare(p1.ProductName, p2.ProductName,
StringComparison.OrdinalIgnoreCase)); // sort first

        int left = 0, right = products.Length - 1;
        while (left <= right)
        {
            int mid = (left + right) / 2;
            int cmp = string.Compare(products[mid].ProductName, name,
StringComparison.OrdinalIgnoreCase);
            if (cmp == 0) return products[mid];
            else if (cmp < 0) left = mid + 1;
            else right = mid - 1;
        }
        return null;
    }
}
```

The Main File for Test:

```
using System;

public class Main
{
    public static void Main(string[] args)
    {
        Product[] products = {
```

```
            new Product(101, "Laptop", "Electronics"),
            new Product(102, "Shoes", "Footwear"),
            new Product(103, "Watch", "Accessories"),
            new Product(104, "Smartphone", "Electronics")
        };

        // Linear Search
        Product result1 = SearchUtility.LinearSearch(products, "Watch");
        Console.WriteLine("Linear Search Result: " + result1);

        // Binary Search
        Product result2 = SearchUtility.BinarySearch(products, "Watch");
        Console.WriteLine("Binary Search Result: " + result2);
    }
}
```

Output:

```
Linear Search Result: [103, Watch, Accessories]
Binary Search Result: [103, Watch, Accessories]
```

4. **Analysis:**
    - Compare the time complexity of linear and binary search algorithms.

Linear Search- Time Complexity = O(n)

Use Case - Small Datasets, unsorted arrays

Binary Search- Time Complexity - O(logn)

Use Case- Large Datasets, sorted arrays

- Discuss which algorithm is more suitable for your platform and why.

Well the answer to this question can be really dynamic depending upon your use case so if we consider the current E-Commerce requirement the Binary Search would be the best bet because it’s ideal in categories or name based searches with sorted data besides we can combine them with Hashmap or Trie for even faster lookup in real systems.