Ecommerce Analysis

Big O Notation

Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.

Why It Matters

In e-commerce platforms with thousands or millions of products, choosing an efficient search algorithm can significantly affect speed and user experience.

Case Scenarios in Search

Case	Description
Best Case	Search finds the element immediately (e.g., first element)
Average Case	Search finds the element somewhere in the middle
Worst Case	Search doesn't find the element or finds it at the last index

Product.cs

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

    public Product(int id, string name, string category)
    {
        ProductId = id;
        ProductName = name;
        Category = category;
    }

    public override string ToString()
    {
        return $"{ProductId} - {ProductName} - {Category}";
    }
}
```

BinarySearch.cs

```
using System;

public class BinarySearch
{
    public static Product? SearchByName(Product[] products, string name)
    {
        Array.Sort(products, (a, b) =>
            string.Compare(a.ProductName, b.ProductName,
StringComparison.OrdinalIgnoreCase));

        int low = 0, high = products.Length - 1;

        while (low <= high)
        {
            int mid = (low + high) / 2;
            int cmp = string.Compare(name, products[mid].ProductName,
StringComparison.OrdinalIgnoreCase);

            if (cmp == 0)
                return products[mid];
            else if (cmp < 0)
                high = mid - 1;
            else
                low = mid + 1;
        }

        return null;
    }
}
```

LinearSearch.cs

```
using System;

public class BinarySearch
{
    public static Product? SearchByName(Product[] products, string name)
    {
        Array.Sort(products, (a, b) =>
            string.Compare(a.ProductName, b.ProductName,
StringComparison.OrdinalIgnoreCase));

        int low = 0, high = products.Length - 1;

        while (low <= high)
        {
            int mid = (low + high) / 2;
            int cmp = string.Compare(name, products[mid].ProductName,
StringComparison.OrdinalIgnoreCase);

            if (cmp == 0)
                return products[mid];
            else if (cmp < 0)
```

```
                high = mid - 1;
            else
                low = mid + 1;
        }

        return null;
    }
}
```

Program.cs

```
using System;

class Program
{
    static void Main()
    {
        // Create a sample product list
        Product[] products = new Product[]
        {
            new Product(101, "Laptop", "Electronics"),
            new Product(102, "Shoes", "Fashion"),
            new Product(103, "Watch", "Accessories"),
            new Product(104, "Smartphone", "Electronics")
        };

        Console.WriteLine("=== E-Commerce Product Search ===");

        Console.Write("Enter product name to search: ");
        string searchName = Console.ReadLine();

        // Linear Search
        Product? linearResult = LinearSearch.SearchByName(products,
searchName);
        if (linearResult != null)
        {
            Console.WriteLine("Linear Search Result: " + linearResult);
        }
        else
        {
            Console.WriteLine("Linear Search: Product not found");
        }

        // Binary Search
        Product? binaryResult = BinarySearch.SearchByName(products, 
searchName);
        if (binaryResult != null)
        {
            Console.WriteLine("Binary Search Result: " + binaryResult);
        }
        else
        {
            Console.WriteLine("Binary Search: Product not found");
        }

        Console.WriteLine("\nPress any key to exit...");
        Console.ReadKey();
    }
}
```

Analysis

Time Complexity

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: Simple but inefficient for large datasets.

Binary Search: Much faster but requires sorting (O(n log n) upfront cost).

Which Algorithm is Better for E-commerce?

Binary Search is more suitable **if:**

- Products can be kept in sorted order.
- You are doing frequent searches with fewer updates.

Linear Search is better **if:**

- The dataset is small or rarely searched.
- You have no time to sort or can't maintain order.