**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.

**Understanding and analyzing :**

To create a powerful search function for an e-commerce site, grasping Big O notation is essential. It allows us to assess how the performance of algorithms changes as the amount of data increases. In terms of searches, the best-case scenario is when the item is found right away, the average case reflects a typical search experience, and the worst-case scenario occurs when the item is either last on the list or not found at all.

You can utilize a Product class that includes attributes like productId, productName, and category. A linear search goes through each item one by one, which has a time complexity of O(n) and can become sluggish as the dataset expands. In contrast, binary search operates on sorted data and boasts a quicker O(log n) time complexity by cutting the search area in half with each step.

While linear search is straightforward and works well with unsorted or complex data, binary search shines when dealing with large, sorted datasets where speed is of the essence. For platforms that demand high performance, opting for binary search or even more sophisticated techniques like indexing is the way to go.

**Code :**

using System;

using System.Collections.Generic;

namespace ECommerceSearch

{

class Product

{

public int ProductId;

public string ProductName;

public string Category;

public Product(int id, string name, string category)

{

ProductId = id;

ProductName = name;

Category = category;

}

}

class Program

{

static void Main()

{

List<Product> products = new List<Product>

{

new Product(105, "Laptop", "Electronics"),

new Product(101, "Shoes", "Fashion"),

new Product(103, "Smartphone", "Electronics"),

new Product(102, "Watch", "Accessories"),

new Product(104, "Backpack", "Travel")

};

Console.WriteLine("Linear Search: Searching for 'Watch'");

Product foundLinear = LinearSearch(products, "Watch");

if (foundLinear != null)

PrintProduct(foundLinear);

else

Console.WriteLine("Product not found.");

Console.WriteLine("\nBinary Search: Searching for Product ID 103");

products.Sort((p1, p2) => p1.ProductId.CompareTo(p2.ProductId));

Product foundBinary = BinarySearch(products, 103);

if (foundBinary != null)

PrintProduct(foundBinary);

else

Console.WriteLine("Product not found.");

}

static Product LinearSearch(List<Product> list, string name)

{

foreach (var product in list)

{

if (product.ProductName.Equals(name, StringComparison.OrdinalIgnoreCase))

return product;

}

return null;

}

static Product BinarySearch(List<Product> list, int targetId)

{

int left = 0;

int right = list.Count - 1;

while (left <= right)

{

int mid = (left + right) / 2;

if (list[mid].ProductId == targetId)

return list[mid];

else if (list[mid].ProductId < targetId)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

static void PrintProduct(Product p)

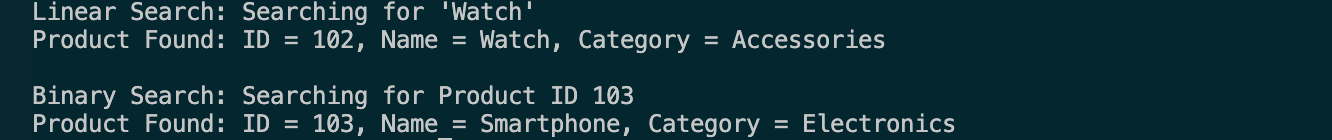
{

Console.WriteLine($"Product Found: ID = {p.ProductId}, Name = {p.ProductName}, Category = {p.Category}");

}

}

}

**Output :**