**Week 1 - Algorithms and Data Structures - Hands-On Exercise**

**Student Name: Jitesh Sen  
Email:** [**jiteshsen2002@gmail.com**](mailto:jiteshsen2002@gmail.com) **Superset ID: 6387362**

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

**Solution:**

**1. Understanding Asymptotic Notation**

**Big O Notation Explanation:** Big O notation helps us understand how our search algorithms perform as the number of products increases. It shows the worst-case scenario of how long an algorithm takes to run.

**Key Concepts:**

* **O(1)** - Constant time: Same speed regardless of data size
* **O(n)** - Linear time: Time increases directly with data size
* **O(log n)** - Logarithmic time: Time increases slowly even with large data

**Search Operation Scenarios:**

* **Best Case**: Product found immediately (first item checked)
* **Average Case**: Product found somewhere in the middle
* **Worst Case**: Product is the last item or doesn't exist

**2. Setup - Product Class**

Created a Product class with three main attributes needed for e-commerce search:

* **productId**: Unique identifier for each product
* **productName**: Name customers search for
* **category**: Product classification for filtered searches

**3. Implementation**

**Linear Search Algorithm**

* **Approach**: Check each product one by one from start to finish
* **Data Requirement**: Works with any array arrangement
* **Process**: Compare search term with each product name until match found
* **Time Complexity**: O(n) - might need to check all n products

**Binary Search Algorithm**

* **Approach**: Divide and conquer method
* **Data Requirement**: Array must be sorted first
* **Process**:
  + Compare search term with middle element
  + If match found, return result
  + If search term is smaller, check left half
  + If search term is larger, check right half
* **Time Complexity**: O(log n) - eliminates half the data each step

**Testing Setup**

* I created sample product arrays for both algorithms
* I implemented comparison counters to track performance
* Tested with both small (8 products) and large (1000 products) datasets

**4. Analysis and Comparison**

**Performance Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time Complexity** | **Space Complexity** | **Data Requirement** |
| Linear Search | O(n) | O(1) | Unsorted OK |
| Binary Search | O(log n) | O(1) | Must be sorted |

**Practical Test Results (1000 products):**

* **Linear Search**: Average 500 comparisons
* **Binary Search**: Average 10 comparisons

**Algorithm Suitability for E-commerce Platform**

**Linear Search is better when:**

* Small product catalogue (under 100 items)
* Products are added/removed frequently
* Simple implementation is priority
* Quick development needed

**Binary Search is better when:**

* Large product catalogue (1000+ items)
* Search speed is critical for user experience
* Products can be pre-sorted
* Multiple searches performed on same data

**Recommendation for Our E-commerce Platform**

**I recommend Binary Search** for our e-commerce platform because:

1. **Scalability**: As our product catalogue grows, binary search will maintain fast performance
2. **User Experience**: Customers expect instant search results, especially on large platforms
3. **Efficiency**: With thousands of products, binary search provides significantly better performance
4. **Future-Proof**: The algorithm handles growth better than linear search

**Conclusion**

Binary search is the better choice for e-commerce platforms due to its superior performance with large datasets. While linear search is simpler, binary search provides the speed customers expect. Algorithm selection should always balance implementation complexity with performance requirements.

Program Output:



