**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.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **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.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**SOLUTION**

**Understand Asymptotic Notation**

* Explain Big O notation and how it helps in analyzing algorithms.

Big O Notation is a way to describe how the time or space needed by an algorithm grows as the size of the input increases. When it is said that a function f(n) is O(g(n)), it means that f(n) will not grow faster than a certain multiple of another function g(n) once the input size is large enough. In other words, f(n) is bounded by g(n) scaled up by some constant value for large inputs.

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

Best, worst, and average case analysis is a method of analyzing algorithms that takes into account the amount of time it takes for an algorithm to run in the best case, worst case, and average case scenarios.

It is a tool used to evaluate the performance of algorithms given certain input data. By analyzing the running time or memory usage of an algorithm under different conditions, one can gain insights into its effectiveness for solving a particular problem. This article will provide an overview of best, worst, and average case analysis by exploring how it works and when it should be applied.

**Analysis**

* Compare the time complexity of linear and binary search algorithms.

Time Complexity of linear search:

The time complexity of linear search is O(n), where n is the number of elements in the array. In the worst-case scenario, the algorithm has to check all elements before finding the target value, making it inefficient for large datasets.

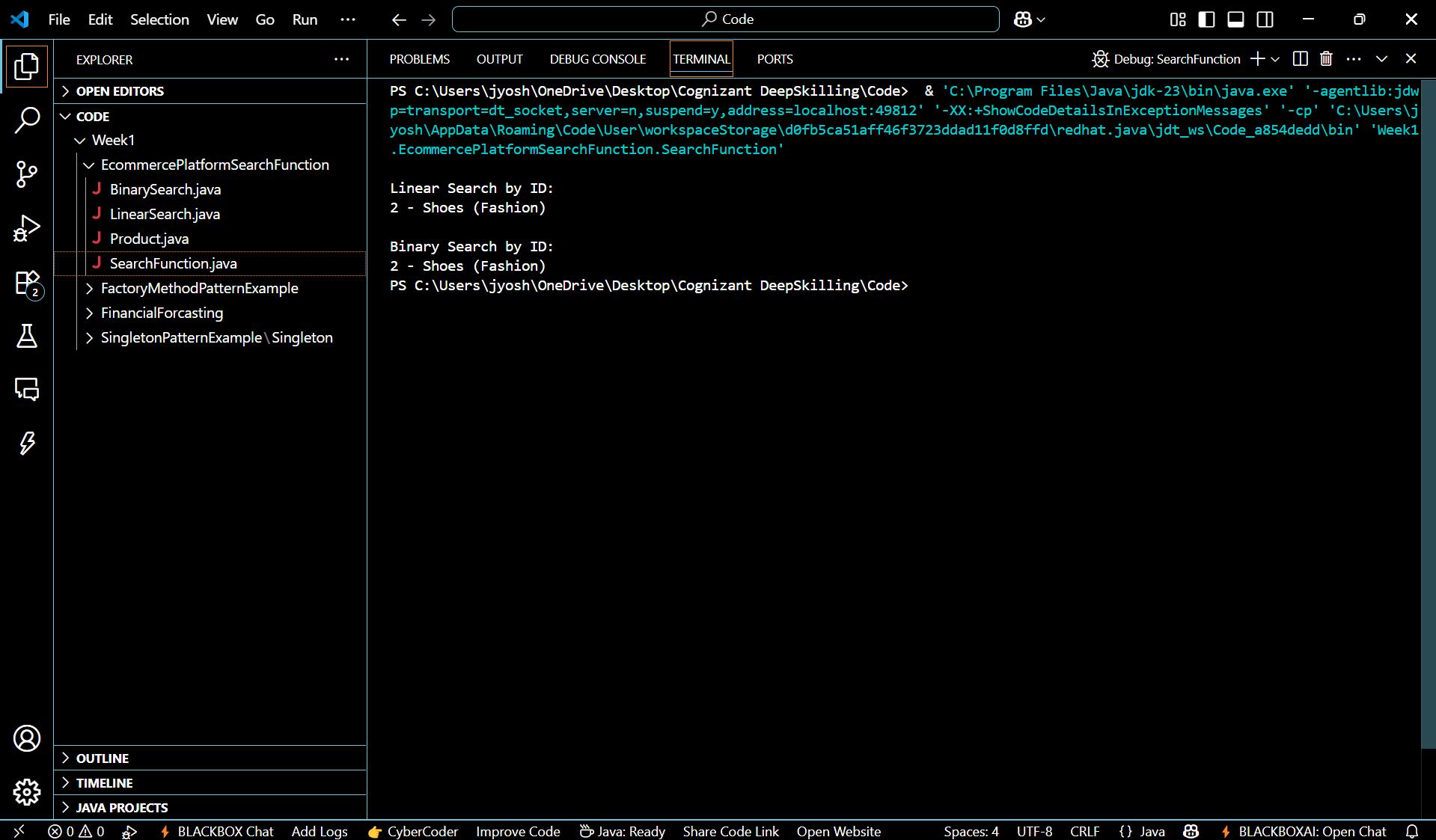
Time Complexity of binary search:

The time complexity of binary search is O(log n), where n is the number of elements in the array. With each comparison, the search space is reduced by half, making it a highly efficient algorithm for searching large datasets.

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

Binary search works well for an e-commerce platform because it is scalable, is efficient, and is fast. This is especially true when the data is kept in a sorted order. However, in real-life situations where users search for partial matches, misspelled names, or keywords, platforms often use better solutions like Elasticsearch or database indexing to allow for full-text and fuzzy searching. Still, for basic search operations using algorithms, binary search provides the best balance between performance and simplicity.

**Output**

****