Exercise 2: E-commerce Platform Search Function

1. **Understand Asymptotic Notation:**

- Explain Big O notation and how it helps in analyzing algorithms.
  Big O notation describes the **upper bound** of an algorithm's running time — essentially how it scales with input size n.

- Describe the best, average, and worst-case scenarios for search operations
  Best, Average, and Worst-Case for Search

Search Method	Best Case	Average Case	Worst Case
Linear Search	O(1) – First element match	O(n/2) ≈ O(n)	O(n) – Last/no match
Binary Search	O(1) – Middle match	O(log n)	O(log n)

2. **Analysis:**

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

Search Type	Time Complexity	Notes
Linear Search	O(n)	Easy, but slow for large data
Binary Search	O (log n)	Much faster, but needs sorting

Sorting the array takes O (n log n), but that's a **one-time cost** if the product list changes infrequently.

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

Platform Needs	Suggested Approach
Small datasets or frequent changes	Linear search (simple)
Large dataset & rarely changed	Binary search (faster)

Exercise 7: Financial Forecasting

1. **Understand Recursive Algorithms:**

- Explain the concept of recursion and how it can simplify certain problems.

What is Recursion?
Recursion is a technique where a method calls **itself** to solve a smaller sub-problem.
Recursion is useful when:
- A problem can be **divided into sub-problems** of the same type.
- There's a **base case** that stops recursion.

2. **Analysis:**

- Discuss the time complexity of your recursive algorithm.

Time Complexity:
- **Recursive Depth = years → O(n)** time
- For n years, the function is called n times

- Explain how to optimize the recursive solution to avoid excessive computation.

Solution 1: **Tail Recursion**- Modify the recursion method to pass forward the result to avoid deep nesting.
Solution 2: **Iterative Approach (for better performance)-** Iterative version uses O(1) space and is more efficient for large n.