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

Big O notation describes how an algorithm's performance (time or space) grows with input size n. It helps compare and analyze algorithms based on their efficiency, especially for large inputs.

**Why it's useful:**  
It shows how well an algorithm scales and helps choose the most efficient one.

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

The best-case scenario for a search operation occurs when the element is found immediately, such as at the beginning of a list or at the root of a tree. It represents the most efficient outcome with the least number of operations.

The average-case scenario assumes the element is located at a random position within the data structure. It represents the typical number of steps taken when the element could be anywhere, and is useful for understanding general performance.

The worst-case scenario happens when the element is either at the last position or not present at all. This requires checking every possible location, making it the most time-consuming outcome.

Compare the time complexity of linear and binary search algorithms

Linear search has a time complexity of O(n) because it checks each element one by one until it finds the target or reaches the end. Binary search has a time complexity of O(log n) because it repeatedly divides the sorted array in half to find the target. Binary search is much faster than linear search for large, sorted datasets.

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

Binary search is more suitable as it is faster with O(log n) time complexity, ideal for large, sorted product data. Linear search is slower with O(n), making it less efficient for an e-commerce platform.

Exercise 2: E-commerce Platform Search Function

