Explain Big O notation and how it helps in analyzing algorithms. Describe the best, average, and worst-case scenarios for search operations.

Big O notation is a mathematical notation that describes the asymptotic behavior (growth rate) of a function in terms of how it scales with respect to an input. It helps in analyzing algorithms by providing a standardized way to express the worst-case time complexity. Key points include:

* Best Case: The minimum time complexity required by an algorithm for any input of size n.
* Average Case: The expected time complexity for a randomly chosen input of size n.
* Worst Case: The maximum time complexity required by an algorithm for any input of size n.

Compare the time complexity of linear and binary search algorithms.

* + Linear Search:
  + Best Case: O(1) (when the first element is the target).
  + Average Case: O(n) (when the target is equally likely to be any element).
  + Worst Case: O(n) (when the target is at the end or not present).
  + Binary Search:
  + Best Case: O(1) (when the target is the middle element).
  + Average Case: O(log n) (due to the halving of the search space).
* Worst Case: O(log n) (when the target is at the extremes or not present).

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

Binary Search is more suitable for the e-commerce platform for the following reasons:

* Efficiency: Binary search has a significantly better worst-case time complexity (O(log n)) compared to linear search (O(n)). This means as the number of products grows, binary search will scale more efficiently in terms of search time.
* Sorted Data Requirement: Binary search requires the data to be sorted by productId, which can be efficiently managed during product insertion or update operations to ensure the array remains sorted.
* User Experience: Faster search times improve user experience on the platform, especially as the number of products increases.