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

This notation provides an upper bound on the growth rate of an algorithm’s running time or space usage. It represents the worst-case scenario, i.e., the maximum amount of time or space an algorithm may need to solve a problem. It helps by comparing the efficiency of different algorithms and understanding their behavior with large inputs.

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

Best Case: The scenario where the algorithm performs the minimum possible number of operations.

Average Case: The scenario that represents the expected performance of the algorithm over a large number of inputs.

Worst Case: The scenario where the algorithm performs the maximum number of operations, such as searching an element that is not present in the array.

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

Linear Search:

Best Case: O(1) (element is at the first position)

Average Case: O(n) (element is in the middle)

Worst Case: O(n) (element is at the last position or not present)

Binary Search:

Best Case: O(1) (element is at the middle position initially)

Average Case: O(log n) (element is anywhere in the sorted array)

Worst Case: O(log n) (element is not present)

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

For an e-commerce platform with a potentially large number of products, binary search is more suitable due to its logarithmic time complexity, provided the product list is maintained in a sorted order.