E-commerce Platform Search Function

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

Ans. Big O notation is a mathematical notation used to describe the upper bound of an algorithm's time or space complexity. It focuses on the growth rate of the algorithm's resource usage as the input size (n) gets very large. It indicates the worst-case scenario.

It helps in algorithm analysis by:

* Comparing algorithms:

Big O allows us to quickly compare the efficiency of different algorithms. For example, an algorithm with O(n) complexity will generally be more efficient than one with O(n^2) complexity for large inputs.

* Optimizing code:

Big O analysis can help us in optimizing our code by identifying areas where we can reduce the complexity, potentially leading to significant performance improvements.

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

Ans. In search operations, the best-case scenario occurs when the target element is found very quickly, the worst-case scenario involves the most time-consuming search, and the average case represents a typical search performance.

Linear Search:

Best Case: The element is found at the first position being checked. This requires only one comparison (O(1)).

Worst Case: The element is not present in the list, or it is located at the very end of the list. This requires checking every element (n comparisons), resulting in O(n) complexity.

Average Case: On average, the element will be found somewhere in the middle of the list, requiring approximately n/2 comparisons. This is also O(n).

Binary Search:

Best Case: The element is found at the middle index (O(1)).

Worst Case: The element is not present, or it requires repeatedly dividing the search space until only one element is left. This is O(log n).

Average Case: On average, the element will be found in a similar manner to the worst case, requiring repeatedly dividing the search space. This is O(log n).

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

Ans. As we have observed the time complexity of linear search is O(n) while the time complexity of binary search is O(log(n)). As observed it took 191100 ns to find the last element by linear search while it took only 9200 ns to find the last element by binary search.

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

Ans. Between linear search and binary search, binary search is more suitable for an e-commerce platform because it offers significantly better performance, especially as the number of products grows. While linear search has a time complexity of O(n) meaning search time increases linearly with the number of products, binary search operates in O(log n) time, making it much faster by repeatedly halving the search space in a sorted product list. Although binary search requires the product list to be sorted, this can be efficiently handled during data ingestion or periodic updates.