**E-Commerce Platform Search Function**

**Understand Asymptotic Notation**

* **Explain Big O Notation and how it helps in analyzing algorithms.**
* Big O notation provides an upper bound on the growth rate of a function, helping us understand the worst-case scenario in terms of time or space complexity.
* It is represented as , meaning that for sufficiently large ,

, will not grow faster than multiplied by some constant factor.

* Big O notation is a critical tool in computer science for asymptotic analysis, as it focuses on the behaviour of algorithms as the input size nnn becomes large, ignoring constant factors and lower-order terms. By providing a standard measure of growth rates, Big O helps compare the efficiency of different algorithms, allowing us to understand their performance and scalability in the worst-case scenario.
* **Describe the best, average and worstcase scenarios for search algorithms.**

**Best case**:

○ Defines the input for which the algorithm takes the least time (fastest time to complete).

○ Input is the one for which the algorithm runs the fastest.

**Average case**

○ Provides a prediction about the running time of the algorithm.

○ Run the algorithm many times, using many different inputs that come from some distribution that generates these inputs, compute the total running time (by adding the individual times), and divide by the number of trials.

○ Assumes that the input is random.

**Worst case**

○ Defines the input for which the algorithm takes a long time (slowest time to complete).

○ Input is the one for which the algorithm runs the slowest.

**Analysis**

* **Compare the time complexity of linear and binary search algorithms.**
* **Definition**:
* **Linear Search**: Scans each element of the list sequentially until the target element is found or the list ends.
* **Binary Search**: Efficiently narrows down the search space by repeatedly dividing the sorted list in half until the target element is found or the search space is empty.
* **Time Complexity**:
* **Linear Search**:
  + Worst-case:
  + Best-case: (if the target is the first element)
  + Average-case:
* **Binary Search**:
  + Worst-case:
  + Best-case: (if the target is the middle element)
  + Average-case:
* **Use Cases**:
* **Linear Search**: Suitable for small or unsorted lists, or when quick implementation is needed without sorting.
* **Binary Search**: Ideal for large sorted lists where search efficiency is critical.
* **Discuss which algorithm is more suitable for your platform and why.**

Linear search is ideal for small or frequently changing datasets and does not require the data to be sorted, making it simple and easy to implement. However, it can be less efficient for large, sorted datasets compared to binary search. On the other hand, binary search is highly effective for large, sorted lists and is particularly useful when the data remains static and is frequently queried. It achieves greater speed by rapidly narrowing down the search range, offering significant performance advantages over linear search in such scenarios.