**Exercise 2: E-commerce Platform Search Function**

**Understanding Asymptotic Notation**

**Big O Notation**

**Big O notation** is a way to describe how the performance of an algorithm changes as the size of the input grows. It's like a measuring tool that helps us understand the efficiency of an algorithm in the worst-case scenario. Here’s a quick rundown of what each notation means:

* **O(1) - Constant Time**: No matter how big your input is, the time it takes to complete the task stays the same. Imagine a light switch; flipping it on or off takes the same amount of time no matter how many switches you have.
* **O(n) - Linear Time**: As your input grows, the time it takes grows proportionally. If you're searching through a list of items, checking each item one by one takes longer if the list is longer.
* **O(log n) - Logarithmic Time**: The time it takes grows much slower compared to the size of the input. For instance, if you’re searching in a sorted list using binary search, each step cuts the search area in half, so it takes much fewer steps even as the list grows.
* **O(n log n) - Linearithmic Time**: This is a bit of a middle ground. The time grows faster than linear time but slower than quadratic time. It’s common in efficient sorting algorithms like quicksort or mergesort.
* **O(n^2) - Quadratic Time**: The time it takes grows with the square of the input size. For example, if you need to compare each item in a list with every other item, the time increases dramatically as the list gets longer.

Understanding these notations helps you figure out how well an algorithm will perform, especially as the amount of data increases. It’s like knowing how different recipes will scale when you cook for a larger group—some might scale well, and others might not.

**Best, Average, and Worst-Case Scenarios**

When analyzing how an algorithm performs, it’s useful to consider different scenarios:

* **Best-Case Scenario**: This is the most optimistic outcome. For example, if you’re searching through a list and find the item you’re looking for on the very first try, that’s the best case. For a linear search, this would be O(1).
* **Average-Case Scenario**: This is a more realistic estimate of how the algorithm will perform under normal circumstances. For linear search, you’d expect to check about half of the items on average, which makes its time complexity O(n).
* **Worst-Case Scenario**: This represents the least favorable outcome. If you’re searching and the item is either at the very end of the list or not in the list at all, you’ll have to check every single item. For linear search, this worst-case scenario is O(n).

In essence, understanding these scenarios helps you prepare for different possible outcomes and choose the right algorithm for your needs.