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

## **Understand Asymptotic Notation:**

### **Explain Big O notation and how it helps in analysing algorithms.**

Big O notation is, in sum, a theoretical concept that helps one to grasp the worst-case scenario of an algorithm's growth rate. It is widely used to label algorithms as having different growth rates of time and space. The Big O notation is focusing on the most important factors which are the main factors and those with the highest order.

* **Performance Prediction**:Big O notation enables us to predict the performance of an algorithm based on the given data. It allows us to get a deep understanding of how the algorithm will grow with input size. This is the most important criterion, whether an algorithm will be fast enough for big sizes of data, or not.
* **Comparison**:It serves the purpose of comparing algorithms by taking into account both time and space complexity. This is the main demand when there are multiple solutions to the same problem viz., selection of the best algorithm.
* **Identifying Bottlenecks**:Developers can come up with strategies of how to optimize the major part of the software by analyzing different parts of the software and determining the complexity of the software. Bottlenecks about the previous one are concerned.
* **Scalability**:Big O notation is a standard method to measure the scalability of an algorithm with a new size of input data, and this is a good practice for OS programming projects. Programs that are designed to handle large data sets or to execute multiple times very fastly are the best examples.

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

When analyzing search operations, it's important to consider the best-case, average-case, and worst-case scenarios. These scenarios help us understand how the algorithm performs under different conditions. Here’s a breakdown for some common search operations:

1. Linear Search

* Best Case: O(1) - Target is the first element.
* Average Case: O(n) - Target is in the middle.
* Worst Case: O(n) - Target is last or not present.

1. Binary Search (on sorted arrays)

* Best Case: O(1) - Target is the middle element.
* Average Case: O(log n) - Dividing the array in half repeatedly.
* Worst Case: O(log n) - Target not found after dividing to one element.

1. Hash Table Search

* Best Case: O(1) - Immediate access using the hash function.
* Average Case: O(1) - Uniform distribution with low load factor.
* Worst Case: O(n) - Many collisions leading to a long chain.