**Algorithms\_Data Structures**

**Exercise 1: Inventory Management System**

In our product search program, we used two ways to find a product by its ID: Linear Search and Binary Search. These are two different methods, and this document explains how they work and how fast they are.

There are mainly three asymptotic notations for time complexity:

1) Big 'O' Notation

2) Big 'Omega' Notation

3) Big 'Theta' Notation

Most of the times, Big 'O' Notation is what's considered for measuring and analysing a program's time complexity.

O(1)- Always takes the same time, regardless of the input size

O(log n)- Input size is reduced each step

O(n)- Time grows in proportion to input size

O(n log n)- Grows faster than linear but less than quadratic

O(n²)- Time grows rapidly with input size (eg. Nested loops)

O(2ⁿ)- Time doubles with each increase in input size

O(n!)- Extremely slow as input grows

**Time Complexities of Linear Search and Binary Search:**

**Linear Search:**

Linear Search is a type of search algorithm that runs through every element of an array and ends the search once the search element is found. In each different case, it has varying time complexity. It is given as below:  
Best case: Very fast (O(1)) if the product is first element of the array.

Average case: Slower (O(n)), looks through about half the array.

Worst case: Slowest (O(n)), looks through the whole array.

**Binary Search:**

Binary Search is a type of search algorithm that takes a sorted array as an input and checks whether the search element lies in the first half of the array, is the middle positioned element or the second half of the array. Strategically, it will keep doing this partitioning until the search element is the middle element of the sub-array. In each different case, it has varying time complexity. It is given as below:  
Best case: Very fast (O(1)) if it's in the middle.

Average case: Much faster than linear (O(log n)).

Worst case: Still fast (O(log n)).

**Comparison**: In comparison to Linear Search, Binary Search is very much better in terms of Time Complexity. So, it is best to use binary search for this product search scenario, for fast retrieval of the product.

**Optimisation Techniques:**

**Optimize Linear Search**

1) Moving frequently accessed items to front: Improves average search time (called "move-to-front heuristic"), heuristic is guiding the program with few heuristic function values, that help the program to find the solution quicker.

2) Using enhanced data structures: Replace array with a HashMap<Integer, Product> if often search by product ID, then it becomes O(1).

**Optimize Binary Search**

1) Avoiding repeated sorting: Only sorting once and maintaining the sorted array as long as no modifications happen.

2) Using efficient sorting algorithms: For large datasets, using faster sort methods (Arrays.sort()) and appropriate datastructure.

3) Appropriate Data Structure: Using TreeMap if both sorted order and fast searches (O(log n))is needed.