Big O notation is a mathematical concept used in computer science in order to understand the performance and complexity of an algorithm, it expresses the **upper bound** or the worst case of an algorithm   
It is denoted by **O (f (n))**, where f (n) represents how the algorithm scales with input size, For example, if an algorithm has a time complexity of O (n), its execution time grows linearly with the input size. This helps programmers:

* Analysis of worst case scenarios.
* Selection of best algorithm.
* Ease in comparison of algorithm.

By understanding of Big O notation programmers can write efficient, fast and optimised algorithms for the real world application scenarios.  
  
The two prominently used searching algorithms used are   
1. Linear Search.

2. Binary Search

**Best Case scenario:**Linear search: O (1) (if we find the target at first place or directly look up to the specific index).

Binary Search: O (1) (if we find target at first place or look up to a specific index).  
  
**Average Case Scenario**:   
Linear Search: O (n/2) (if the target is present at middle of the input structure).

Binary Search: O (log n) (as this works in sorted array and half and conquer it always require same amount of time)   
  
**Worst Case Scenario:**   
Linear Search: O (n) (if target element is present at the last index, in order to find it we have to traverse the whole array).

Binary Search: O (log n) (doesn’t matter where the target is, as the array is sorted it is easier to find the element in same time complexity).