Space & Time Complexity Big O Notation

By Wisnu Yudhosantoso 5221600063.

Table Of Content

1 | S&T Complexity

A fine programmer would use the most efficient method to find a solution. In that case, a complexity of an algorithm would need to be minimalized. Complexity in a algorithm is divided into two,

a. **Space**, It observe how much memory would it need to process a certain algorithm
b. **And, Time**. Distinguish how long would it need to run a certain algorithm.

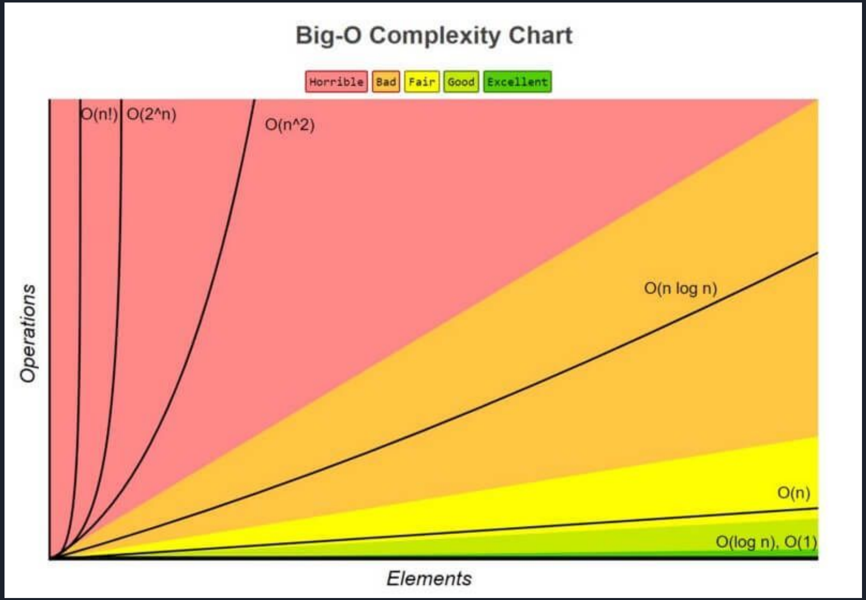
2 | Time Complexity Analysis

Since the topic is heading to the **big O notation**, time complexity is the running point and an analysis occurs.

A certain simple way to know how long would it need to run an algorithm. That is the method of **Time complexity analysis**, also known as the **big O notation**.

3 | Big O Notation

In computer science, **big O notation** is used to classify algorithms according to how their run time or space requirements grow as the input size grows.



4 | Types of Big O

01 **Constant time**, probably at your best that you can create an algorithm in a form of O(1).

02 **O(log n) or Logarithmic Time**, a good complexity for sorting algorithms.

03 **O(n) or Linear Time**, a fair complexity with running time increases at most linearly with the size of the input.

04 **O(n²) or Quadratic time**, a somewhat horrible complexity. This can happen when we run a linear function inside a linear function (n*n).

05 **O(2^n) or Exponential time**, where we don't know the problem at hand, thus it's to try every combination and permutation of all possibilities.

06 **O(n!) or factorial time**. To sum it up, it helps us identify the worst-case scenario for our algorithms, O(n!) clearly is the worst of the worst.

Thank you for listening!