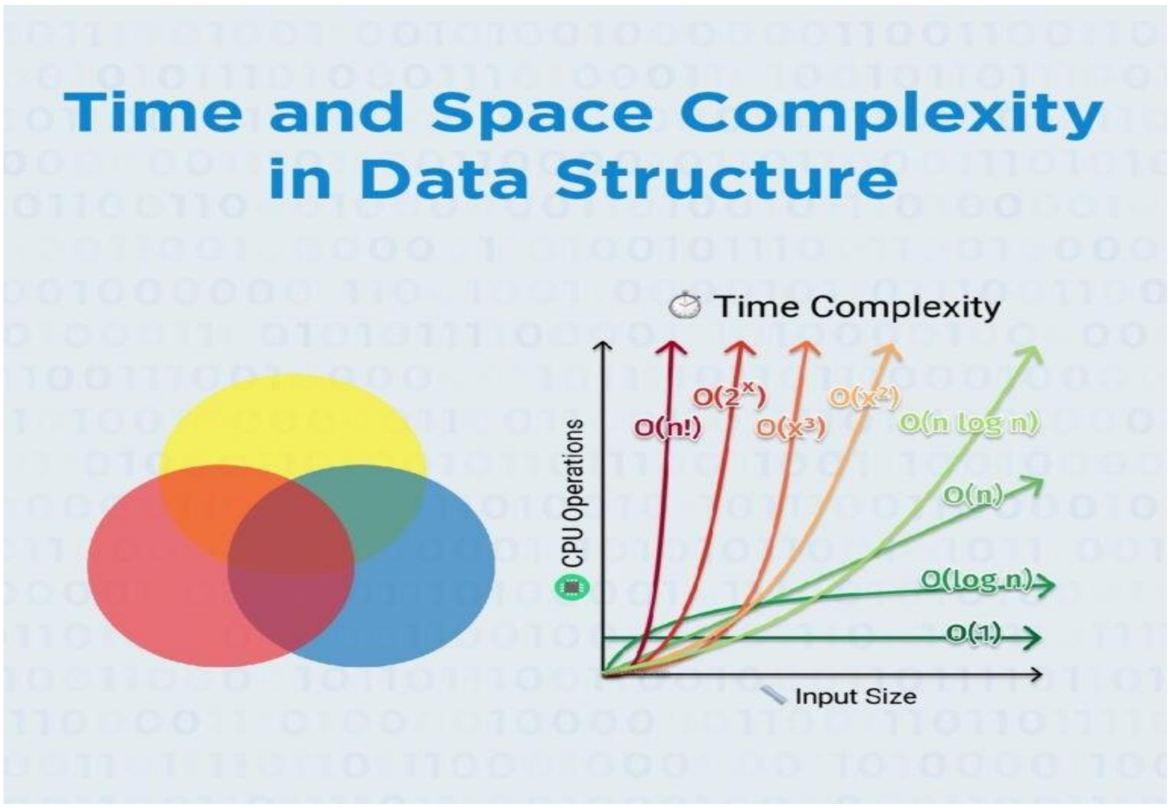
TIME SPACE COMPLEXITY

The time complexity is the computational complexity that describes the amount of computer time it takes to run an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that each elementary operation takes a fixed amount of time to perform. Thus, the amount of time taken and the number of elementary operations performed by the algorithm are taken to be related by a constant factor.

Since an algorithm's running time may vary among different inputs of the same size, one commonly considers the worst-case time complexity, which is the maximum amount of time required for inputs of a given size. Less common, and usually specified explicitly, is the average-case complexity, which is the average of the time taken on inputs of a given size (this makes sense because there are only a finite number of possible inputs of a given size). In both cases, the time complexity is generally expressed as a function of the size of the input. Since this function is generally difficult to compute exactly, and the running time for small inputs is usually not consequential, one commonly focuses on the behaviour of the complexity when the input size increases—that is, the asymptotic behaviour of the complexity. Therefore, the time complexity is



The space complexity of an algorithm or a computer program is the amount of memory space required to solve an instance of the computational problem as a function of characteristics of the input. It is the memory required by an algorithm until it executes completely.[1] This includes the memory space used by its inputs, called input space, and any other (auxiliary) memory it uses during execution, which is called auxiliary space.

commonly expressed using big O notation, typically $O(n)$, $O(n\log n)$, $O(n^{\infty})$, $O(2^{n})$, etc., where n is the size in units of bits needed to represent the input.

Algorithmic complexities are classified according to the type of function appearing in the big O notation. For example, an algorithm with time complexity $O(n)$ is a *linear time algorithm* and an algorithm with time complexity $O(n^{\infty})$ for some constant $\infty>1$ is a *polynomial time algorithm*.