Bing

Time complexity functions are functions that describe how the running time of an algorithm depends on the size of the input. Some common time complexity functions are:

- O (1): Constant time. The running time does not depend on the input size. For example, accessing an array element by index takes constant time.

- O (log n): Logarithmic time. The running time grows proportionally to the logarithm of the input size. For example, binary search takes logarithmic time to find an element in a sorted array.

- O(n): Linear time. The running time grows proportionally to the input size. For example, traversing an array or a linked list takes linear time.

- O (n log n): Quasilinear time. The running time grows faster than linear but slower than quadratic. For example, merge sort and quick sort take quasilinear time to sort an array.

- O(n^2): Quadratic time. The running time grows proportionally to the square of the input size. For example, bubble sort and insertion sort take quadratic time to sort an array.

- O(2^n): Exponential time. The running time grows exponentially with the input size. For example, finding all subsets of a set takes exponential time.

- O(n!): Factorial time. The running time grows faster than exponential with the input size. For example, finding all permutations of a set takes factorial time.

There are other time complexity functions as well, such as O(n^k), O(k^n), O(log log n), O(n log log n), etc., but they are less common in practice.