Big O is the way of measuring the efficiency of an algorithm and how well it scales based on the size of the dataset.

Big O references how complex an algorithm is.

Big O is represented using something like O(n).

n is the size of a collection.

O(1)/Constant Complexity: Constant.

This means irrelevant of the size of the data set the algorithm will always take a constant time. 1 item takes 1 second, 10 items takes 1 second, 100 items takes 1 second. It always takes the same amount of time.

O(n)/Linear Complexity:

The larger the data set, the time taken grows proportionately. 1 item takes 1 second, 10 items takes 10 seconds, 100 items takes 100 seconds.

O(log n)/Logarithmic Complexity:

The time taken increases with the size of the data set, but not proportionately

O(n log n): A nice combination of the previous two.

Normally there’s 2 parts to the sort, the first loop is O(n), the second is O(log n), combining to form O(n log n) 1 item takes 2 seconds, 10 items takes 12 seconds, 100 items takes 103 seconds.

O(n^2)/Quadratic Complexity: Things are getting extra slow. 1 item takes 1 second, 10 items takes 100, 100 items takes 10000.