Big-O-Notation

Big O notation is also called Landau’s symbol. Big O notation is used in Computer Science to describe the performance or complexity of an algorithm. Big O specifically describes the **worst-case** scenario. It can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm. The letter O is used because the rate of growth of a function is also called its order. Efficiency covers CPU (time) usage, Memory usage, Disk usage, Network usage. However, Time complexity (CPU usage) is most important. Complexity affects performance but performance does not affect complexity. Different types of efficiency are:

O (1): Constant. O (1) describes an algorithm that will always execute in the same time (or space) regardless of the size of the input data set.

O (N): Linear. O (N) describes an algorithm whose performance will grow linearly and in direct proportion to the size of the input data set.

O (N2): Quadratic. O (N2) represents an algorithm whose performance is directly proportional to the square of the size of the input data set. This is common with algorithms that involve nested iterations over the data set. Deeper nested iterations will result in O (N3), O (N4) etc.

O (2N): O (2N) denotes an algorithm whose growth will double with each additional element in the input data set. The execution time of an O (2N) function will quickly become very large.

O (log (n)): Logarithmic. The operation will take longer as the input size increases, but once the input gets fairly large it won’t change enough to worry about.

O (n log n): Operations run in log linear time - increasing the input size hurts, but may still be manageable.

O (n!): Factorial. It involves doing something for all possible permutations of the n elements. The operation is impractical for any reasonably large input size n*.*

Factorial(n) = n\*factorial(n-1) if n >0 and 1 if n=0

Example:

Factorial (n) = n \* (n-1) \* (n-2) \* ... \* 1; for any integer n > 0

Factorial (0) = 1

For the problem of size N, we can use A constant-time algorithm is “order 1”: O (1), A linear- time algorithm is “order N”: O (N), A quadratic time algorithm is “order N “squared”: O (N2)