To assess the time complexity, one should not use real time measures like milli-seconds or micro-seconds. Instead, one must use logical units that express the relation between the size n of the data and the amount of time t required to process the data.

Total execution time is derived from $\sum op_i(n) * t$. $op_i(n)$ is the total number of instances of operation op_i and t_i is the time taken

The *Operation Count* and *Step Count* are two commonly used measures to compare and assess algorithms.

Commonly Used functions in Time complexity Analysis

Before moving onto the analysis of individual algorithms, let's look at some of the functions that are commonly used in analysis of time complexity.

f(n)	Performance
1	*Constant Performance*. Does not vary with n.
log n	*Logarithmic*: The execution time increases when n increases, but much slower.
n	*Linear*: Run time varies directly with n.
n log n	When n doubles, run time slightly more than doubles.
n^2	*Quadratic*: when n doubles, the execution time increases fourfold. Practical only for small problems.
n^3	*Cubic*: when n doubles, the execution time increases eightfold.
2^n	*Exponential*: when n doubles, the execution time squares.

The same data has been graphically represented in Figure-8. As can be seen, the curves reflect the increase in execution time with increasing size of n.

‹ Asymptotic Analysis - Part 1 (/?q=MULNCourseBook/listDetailedCBLearningContents/178099/cidb/full/view/153678/431532/431528/39730)
Searching Algorithms - Linear and Binary Search › (/?q=MULNCourseBook/listDetailedCBLearningContents/178099/cidb/full/view/153734/431536/431534/39731)