Some Commonly Encountered Functions

The following functions are often encountered in computer science complexity analysis. We'll express them in terms of BigOh just to be specific. They apply equally to BigOmega and BigTheta.

- $T(n) = O(1)$. This is called constant growth. $T(n)$ does not grow at all as a function of n, it is a constant. It is pronounced ``BigOh of one.'' For example, array access has this characteristic. The operation `A[i]` takes the same number of steps no matter how big `A` is.
- $T(n) = O(\lg(n))$. This is called logarithmic growth. $T(n)$ grows as the base 2 logarithm of n (actually, the base doesn't matter, it's just traditional to use base-2 in computer science). It is pronounced ``BigOh of log n.'' For example, binary search has this characteristic.
- $T(n) = O(n)$. This is called linear growth. $T(n)$ grows linearly with n. It is pronounced ``BigOh of n.'' For example, looping over all the elements in a one-dimensional array would be an $O(n)$ operation.
- $T(n) = O(n \lg(n))$. This is called ``n log n'' growth. $T(n)$ grows as $n$ times the base 2 logarithm of $n$. It is pronounced ``BigOh of n log n.'' For example, heapsort and mergesort have this characteristic.
- $T(n) = O(n^k)$. This is called polynomial growth. $T(n)$ grows as the k-th power of n. Computer applications with k greater than about 3 are often impractical. They just take too long to run for any reasonably large problem. Selection sort is an example of a polynomial growth rate algorithm. It is in $O(n^2)$, pronounced ``BigOh of n squared.''
- $T(n) = O(2^n)$. This is called exponential growth. $T(n)$ grows exponentially. It is pronounced ``BigOh of 2 to the n.'' Exponential growth is the most-feared growth pattern in computer science; algorithms that grow this way are basically useless for anything but very small problems. In computer science, we traditionally use the constant 2, but the any other positive constant could be used to express the same idea.

The growth patterns above have been listed in order of increasing ``size.'' That is,

$$O(1) = O(\lg(n)) = O(n) = O(n \lg(n)) = O(n^2) = O(n^3) = O(2^n)$$

It may appear strange to use the equals sign between two BigOh expressions. Since BigOh defines a *set* of functions, the notation $O(f) = O(g)$ means that the set of functions $O(f)$ is *contained* in the the set of functions $O(g)$. Note that it is not true that if $g(n) = O(f(n))$ then $f(n) = O(g(n))$. The ``$=$'' sign does not mean equality in the usual algebraic sense.

Best, Worst, and Average Cases

It's usually not enough to describe an algorithm's complexity by simply giving an expression for O, Ω, or Θ. Many computer science problems have different complexities depending on the data on which they work. Best, worst, and average cases are statements about the *data*, not about the algorithm.

The *best* case for an algorithm is that property of the data that results in the algorithm performing as well as it can. The *worst* case is that property of the data that results in the algorithm performing as poorly as possible. The*average* case is determined by averaging algorithm performance over all possible data sets. It is often very difficult to define the average data set.

Note that the worst case and best case do not correspond to upper bound and lower bound. A complete description of the complexity of an algorithm might be ``the time to execute this algorithm is in $O(n)$ in the average case.'' Both the complexity and the property of the data must be stated. A famous example of this is given by the quicksort algorithm for sorting an array of data. Quicksort is in $O(n^2)$ worst case, but is in $O(n \lg n)$ best and average cases. The worst case for quicksort occurs when the array is already sorted (*i.e.*, the data have the property of being sorted).

One of the all-time great theoretical results of computer science is that any sorting algorithm that uses comparisons must take at least $n \lg n$ steps. That means that any such sorting algorithm, including quicksort, must be in $\Omega(n \lg n)$. Since quicksort is in $O(n \lg n)$ in best and average cases, and also in $\Omega(n \lg n)$, it must be in $\Theta(n \lg n)$ in best and average cases.