As another example, suppose that for some algorithm, the exact number of steps is $T(n) = 5n^2 + 27n + 1005$. When *n* is small, say 1 or 2, the constant 1005 seems to be the dominant part of the function. However, as *n* gets larger, the n^2 term becomes the most important. In fact, when *n* is really large, the other two terms become insignificant in the role that they play in determining the final result. Again, to approximate $T(n)$ as *n* gets large, we can ignore the other terms and focus on $5n^2$. In addition, the coefficient 5 becomes insignificant as *n* gets large. We would say then that the function $T(n)$ has an order of magnitude $f(n) = n^2$, or simply that it is $O(n^2)$.

Although we do not see this in the summation example, sometimes the performance of an algorithm depends on the exact values of the data rather than simply the size of the problem. For these kinds of algorithms we need to characterize their performance in terms of best case, **worst case**, or **average case** performance. The worst case performance refers to a particular data set where the algorithm performs especially poorly. Whereas a different data set for the exact same algorithm might have extraordinarily good performance. However, in most cases the algorithm performs somewhere in between these two extremes (average case). It is important for a computer scientist to understand these distinctions so they are not misled by one particular case.

A number of very common order of magnitude functions will come up over and over as you study algorithms. These are shown in Table 1. In order to decide which of these functions is the dominant part of any $T(n)$ function, we must see how they compare with one another as *n* gets large.

Table 1: Common Functions for Big-O

f(n)	Name
1	Constant
$\log n$	Logarithmic
n	Linear
$n \log n$	Log Linear
n^2	Quadratic
n^3	Cubic
2^n	Exponential

Figure 1 shows graphs of the common functions from Table 1. Notice that when *n* is small, the functions are not very well defined with respect to one another. It is hard to tell which is dominant. However, as *n* grows, there is a