Big-Oh Notation
Informally, we say that an algorithm that operations on input of size $n$ is worst-case $O(f(n))$ (e.g. $O(n^2)$ ) if for large values of $n$ the algorithm uses no more than a fixed constant times $f(n)$ basic operations. For example, we will show that the BubbleSort algorithm uses no more than $2n(n-1)$ operations; this is certainly no more than $2n^2$, which is a multiple of $n^2$, so BubbleSort is worst-case $O(n^2)$.
Stylistically, we try to keep the orders as simple as possible. Since there is an arbitrary constant factor $O(n^2)$ is the same as $O(23n^2)$, which in turn is the same as $O(23n^2+5n+6)$. We represent all of these as $O(n^2)$. 
There are several symbols like $O$:

- $O( f(n) )$ (Big-Oh) represents an upper bound -- for large $n$ the algorithm is no worse than a constant times $f(n)$, but it might be better.
- $\Omega( f(n) )$ (Big-Omega) represents a lower bound -- for large $n$ the algorithm is no better than a constant times $f(n)$.
- $\Theta( f(n) )$ (Big-Theta) represents both an upper and a lower bound.
- $o( f(n) )$ (Little-Oh) is a strict upper bound: $O(f(n))$ and not $\Omega( f(n) )$

This semester we will use Big-Oh almost exclusively.
Here are a few formal definitions for the mathematically inclined:

- We say function $T(n)$ is $O( f(n) )$ if there are constants $k$ and $N$ so that for every $n \geq N$ we have $T(n) \leq k*f(n)$.

- We say $T(n)$ is $\Omega( f(n) )$ if there are constants $k$ and $N$ so that for every $n \geq N$ we have $T(n) \geq k*f(n)$.

Don’t worry about these for now.
Here is an addition rule: if \( T_1(n) \) is \( O( f(n) ) \) and \( T_2(n) \) is \( O( f(n) ) \) then \( T_1(n) + T_2(n) \) is \( O( f(n) ) \).

For example, \( 3n^3 - 2n^2 + 27 \) is \( O(n^3) \) because each of its terms is \( O(n^3) \).

In general, a polynomial of degree \( k \) is \( O( n^k ) \) because each of its terms is \( O(n^k) \).
Note that all logarithms are proportional -- if \(a\) and \(b\) are any two bases, then
\[
\log_a(x) = \log_b(x) \cdot \log_a(b)
\]

If we are only talking about orders of growth, it doesn't matter if we interpret "log" as meaning base-2 logs or base-10 logs or natural logs; each is a constant times each of the others.