Objections to and misconceptions of big-oh notation take forms such as

- Big-oh notation specifies only an upper bound of running time, which might be widely imprecise.
- Big-oh notation measures only the worst case, when the best case or the typical case might be much better.
- Big-oh ignores constants, which can greatly affect running time in practice.
- Algorithms that have the same big-oh category can have widely different running times in practice.
- Big-oh considers only the size of the input, when in fact other attributes of the input can greatly affect running time.
\( g(n) \sim f(n) \) means the functions are asymptotically equal, that is, that 
\[
\lim_{n \to \infty} \frac{g(n)}{f(n)} = 1.
\]
Thus \( \frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3} \sim \frac{n^3}{6} \).

\( g(n) = O(f(n)) \), which really should be written \( g(n) \in O(f(n)) \), means that a scaled version of \( f(n) \) asymptotically bounds \( g \) above. It means there exists a \( c \) such that when \( n \) is large enough, \( g(n) \leq cf(n) \). Thus \( \frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3} = O\left(\frac{n^3}{6}\right) \) but also
\[
\frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3} = O(n^3) \text{ and } \frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3} = O(n^4).
\]

With big-oh, you can throw away the lower ordered terms and throw away the constant factor of the highest order term and overshoot.

With tilde, you only can throw away the lower ordered terms.