Big-O Notation	**Running Time**	**Meaning**
O(1)	Constant	Running term is independent of the input size. We would get the same run time regardless of what we put in.
*O(*log *n)*	Logarithmic	Running time grows as the log of at least one of the inputs. Typically, we assume it is $\log_2$. Every time the size of the input doubles, the algorithm performs one additional step.
O(n)	Linear	Usually algorithms dealing with lists and sequences as they touch each element of the sequence a constant number of times (i.e. comparing two strings). If you have to “touch” every element (including reindexing).
O(n log *n)*	Log-linear	Many practical algorithms are log linear. In the worst-case scenario, *sorting* is log-linear.
$O(n^c)$	Polynomial	Run time is the power of the input. Can be quadratic $O(n^2)$ or cubic $O(n^3)$. This is what happens in nested loops, and why they are so inefficient.
$O(c^n)$	Exponential	Dangerous territory. Generally considered impractical as they are too slow. Many important problems are inherently exponential i.e. Fibonacci.