Big O Notation Compare-o-Matic

	O(1)	O(log n)	O(n)	O(n log n)	O(n^2)	O(2^n)	O(n!)
Rate of Growth:	Constant	Logarithmic	Linear	Linearithmic	Quadratic	Exponential	Factorial
…which means:	The program takes the same time to run, no matter how big the input is.	The program runtime increases only slowly, even with big increases in the size of the input.	The program runtime increases proportionally to the size of the input.	It takes linear time *plus* logarithmic time	Processing time increases at a faster rate as the site of the input increases.	The program runtime increases very quickly even with moderate increases in the input size.	Doing all possible permutations for all inputs.
Input Example:	Even if I double/triple/quadruple/quintuple…the input size, the runtime does not increase.	If I double the input size, the runtime increases by 1.	If I double the input size, the runtime doubles.	It remains around linear time until the input reaches “advanced” size. But it is still slower than logarithmic time.	If I double the input size, the runtime quadruples.	If I triple the input size, the runtime is cubed.	If the input is 5, then 5! is: (5 * 4 * 3 * 2 * 1) = 120
This is considered…	Excellent	Good	Fair	Bad	Horrible	Horrible	Horrible
Real World Example	Nightcrawler teleporting from Point A to Point B. It will take the same amount of time, regardless of the distance between A and B.	The Richter scale: A “2” on the Richter scale is *ten times* the intensity of a “1” on the Richter scale.	Since a Roomba has only 1 speed, it will take twice as much time to cross a room that’s twice as long.	The task of compiling a telephone or address book from a list of names/numbers.	Nightcrawler on a giant checkerboard, teleporting square-by-square looking for his wallet he dropped somewhere.	Compound Interest	A deck of 52 cards has more possible arrangements than there are *atoms* in the friggin’ *galaxy.*
Programming Example	Looking up an element of an array by its index will take the same time regardless of the length of the array.	An algorithm which cuts the problem in half each time, such as Binary Search.	Simple Search: Finding an item in an unsorted list where the size of the list is N.	A fast sporting algorithm like Quicksort, Mergsort, or Heapsort	A slow sorting algorithm, like Selection Sort, Bubble Sort, or Insertion Sort	Solving matrix chain multiplication via brute-force search	A really slow algorithm, like the Traveling Salesman problem where you’re exploring every possible permutation.