Big O notation is a formal expression of an algorithm's complexity in relation to the growth of the input size. Hence, it is used to rank algorithms based on their performance with large inputs.

Rating	Notation	Name	Desc	Use Case	Explaination
Excellent	O(1)	Constant	Executes in same time regardless of size of dataset	Hashing Algorithm	Extracting data from an item in an array
Good	O(log n)	Logarithmic	Halves dataset each pass	Binary Search	
Neutral	O(n)	Linear	with growing dataset	Linear Search	A single loop through an array
Fair	O(n log n)	Linearithmic	Divides dataset	Merge Sort	
Poor	$O(n^2)$	Polynomial	Performance proportional to size of the data set	Bubble Sort	Nested loop iterating through a 2D array
Terrible	$O(2^n)$	Exponential	Doubles with each addition to dataset in each pass	Fibonacci number calc	Recursive function with 2 calls

Algorithm		Time Complexity		
		Best	Average	Worst
Searching	Binary Array	O(1)	O(log n)	O(log n)
	Binary Tree	O(1)	O(log n)	O(n)
	Linear	O(1)	O(n)	O(n)
	Breadth/Depth Firist	O(1)	O(V + E)	$O(V^2)$
Sorting	Quick	O(n log n)	O(n log n)	$O(n^2)$
	Merge	O(n log n)	O(n log n)	O(n log n)
	Insertion	O(n)	$O(n^2)$	$O(n^2)$
	Bubble	O(n)	$O(n^2)$	$O(n^2)$