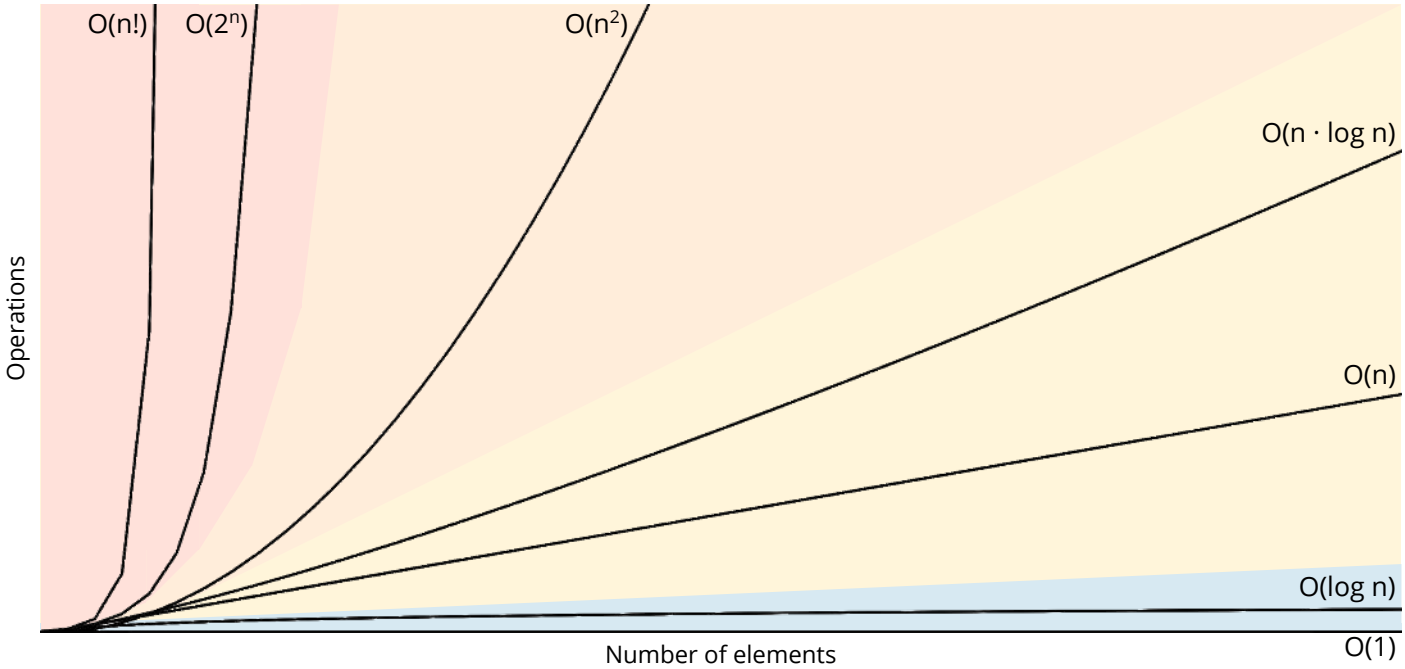
Big O Notation Cheat Sheet

www.happycoders.eu



Time Complexity Classes Explained

Complexity Class	Description	Example	
Constant Time O(1)	The number of operations stays the same, independent of the number of elements.	Accessing a specific element of an array.	Excellent
Logarithmic Time O(log n)	The number of operations increases by a constant amount whenever the number of elements doubles.	Using binary search to find an element in a *sorted* array.	
Linear Time O(n)	The number of operations grows linearly with the number of elements *n*. If *n* doubles, the number of operations doubles, too.	Finding an element in an *unsorted* array.	Fair
Quasilinear Time O(n · log n)	The number of operations grows slightly faster than linear as the linear component is multiplied with a logarithmic one.	Efficient sorting algorithms like Quicksort, Merge Sort, Heapsort.	
Quadratic Time $O(n^2)$	The number of operations grows linearly with the square of the number of elements *n*. If *n* doubles, the number of operations quadruples.	Simple sorting algorithms like insertion sort and selection sort.	Bad
Exponential Time $O(2^n)$	The number of operations grows exponentially with the number of elements. It doubles for each additional element.	Recursive Fibonacci method.	Terrible
Factorial Time O(n!)	The number of operations grows linearly with the factorial of the number of elements *n*, which is the product of all numbers up to (and including) *n*.	Brute-force solution to the traveling salesman problem.	