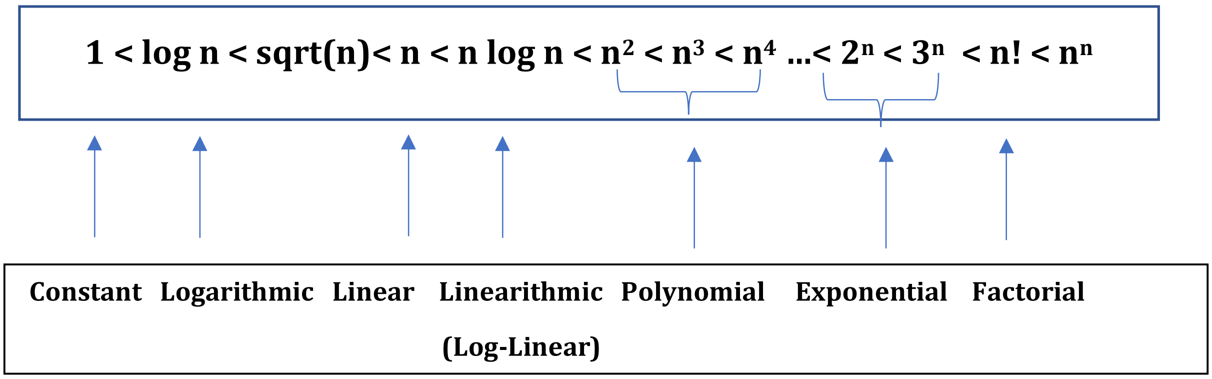
Time Complexity - Growth of Functions (on input size)

Complexity Grows along this direction.

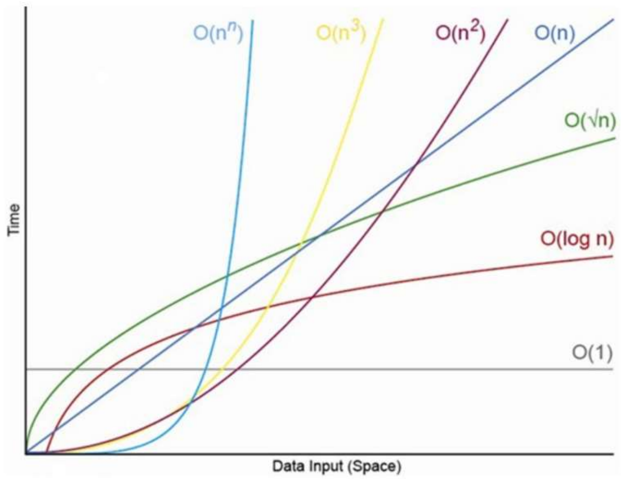
Best → Worst

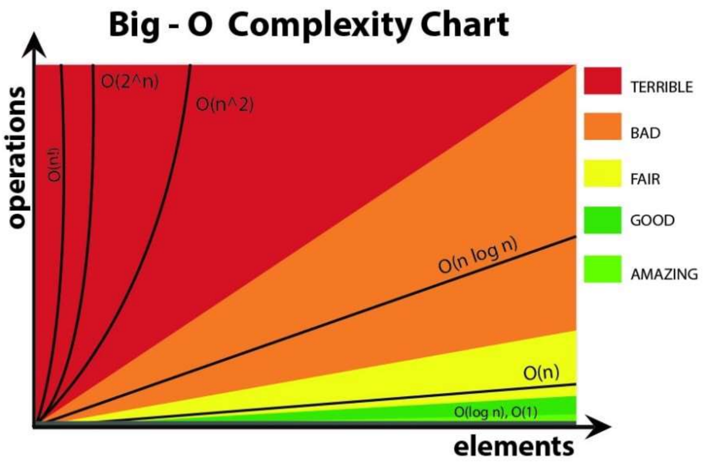


$1 < \log n < \text{sqrt}(n) < n < n \log n < \underbrace{n^2 < n^3 < n^4}\ \ldots < \underbrace{2^n < 3^n}\ < n! < n^n$

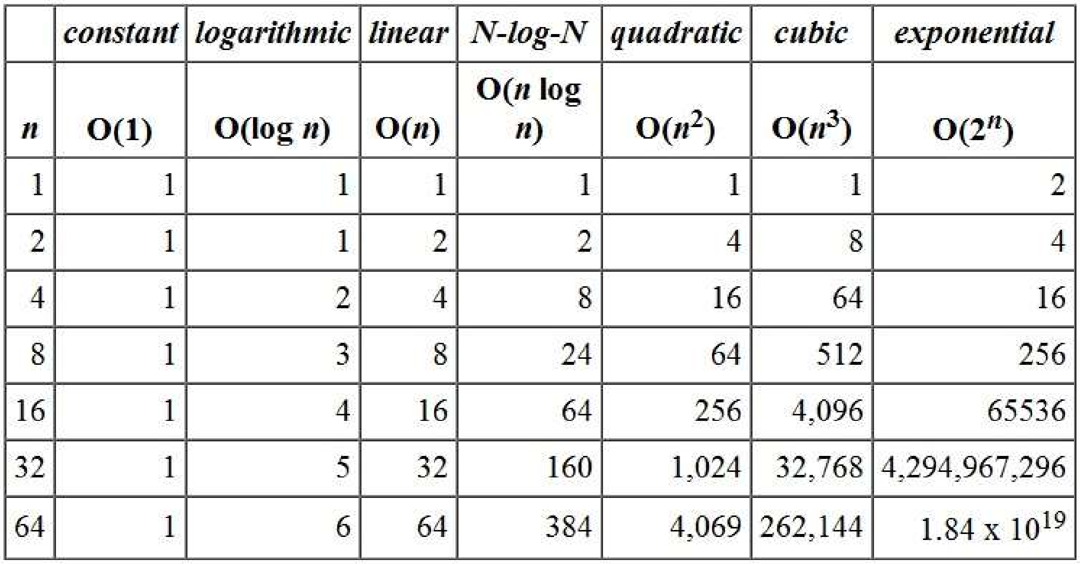
Constant **Logarithmic** **Linear** **Linearithmic (Log-Linear)** **Polynomial** **Exponential** **Factorial**

Growth of Functions

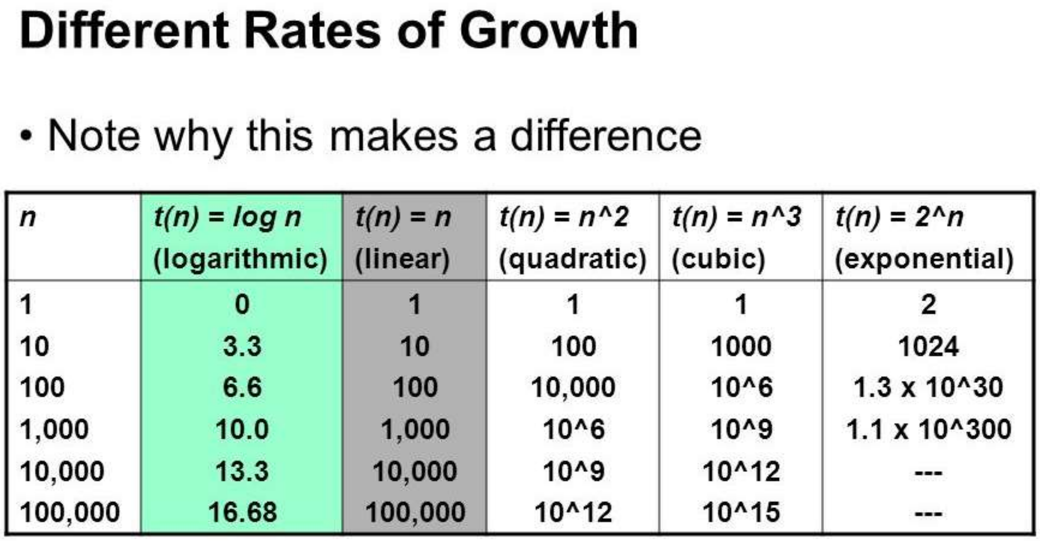




Number of Comparisons



	constant	*logarithmic*	*linear*	*N-log-N*	*quadratic*	*cubic*	*exponential*
n	**O(1)**	**O(log *n*)**	**O(*n*)**	**O(*n* log *n*)**	**O(n^2)**	**O(n^3)**	**O(2^n)**
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84×10^{19}

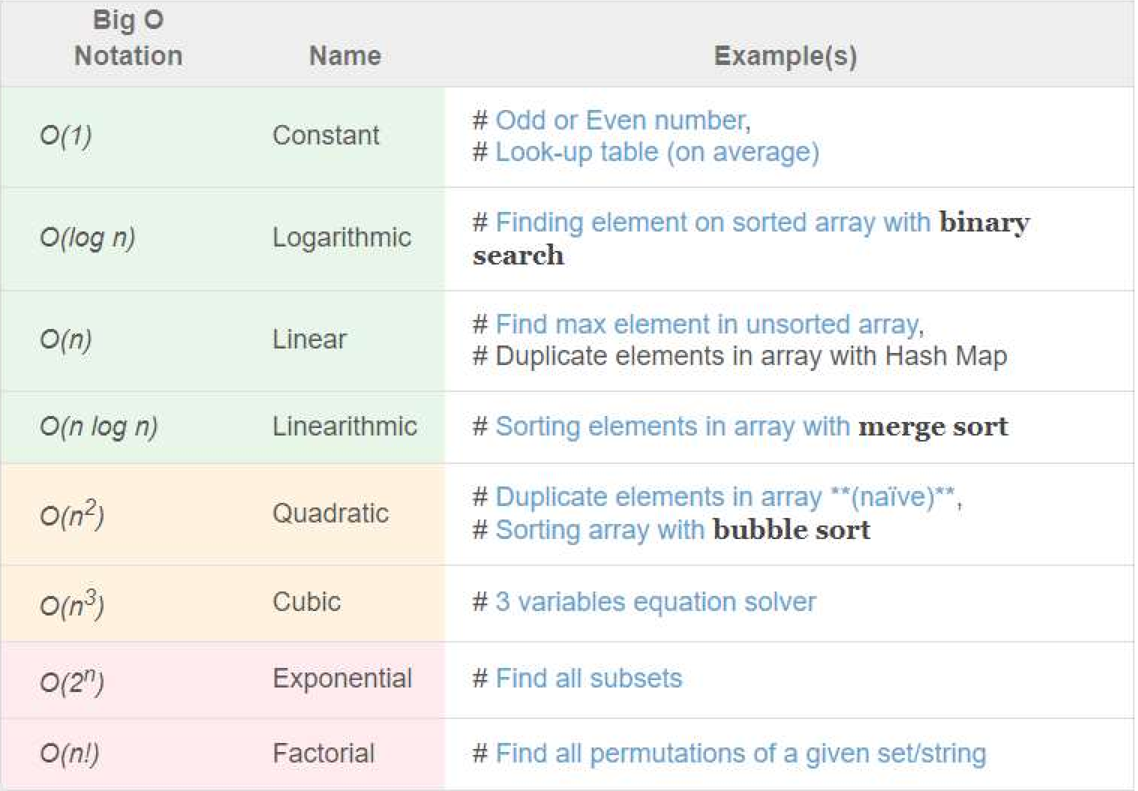


Different Rates of Growth

- Note why this makes a difference

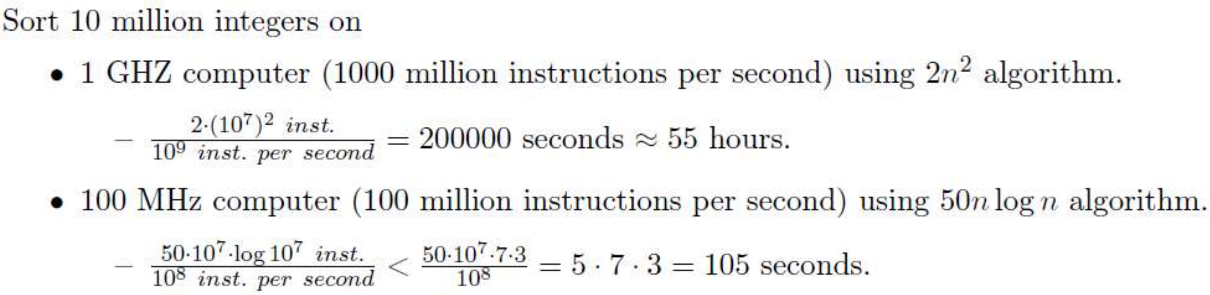
n	*t(n) = log n* (logarithmic)	*t(n) = n* (linear)	*t(n) = n^2* (quadratic)	*t(n) = n^3* (cubic)	*t(n) = 2^n* (exponential)
1	0	1	1	1	2
10	3.3	10	100	1000	1024
100	6.6	100	10,000	10^6	1.3 x 10^30
1,000	10.0	1,000	10^6	10^9	1.1 x 10^300
10,000	13.3	10,000	10^9	10^12	---
100,000	16.68	100,000	10^12	10^15	---

Algorithm Complexity Classes and Examples



Big O Notation	Name	Example(s)
O(1)	Constant	# Odd or Even number, # Look-up table (on average)
O(log n)	Logarithmic	# Finding element on sorted array with **binary search**
O(n)	Linear	# Find max element in unsorted array, # Duplicate elements in array with Hash Map
O(n log n)	Linearithmic	# Sorting elements in array with **merge sort**
$O(n^2)$	Quadratic	# Duplicate elements in array **(naïve)**, # Sorting array with **bubble sort**
$O(n^3)$	Cubic	# 3 variables equation solver
$O(2^n)$	Exponential	# Find all subsets
O(n!)	Factorial	# Find all permutations of a given set/string

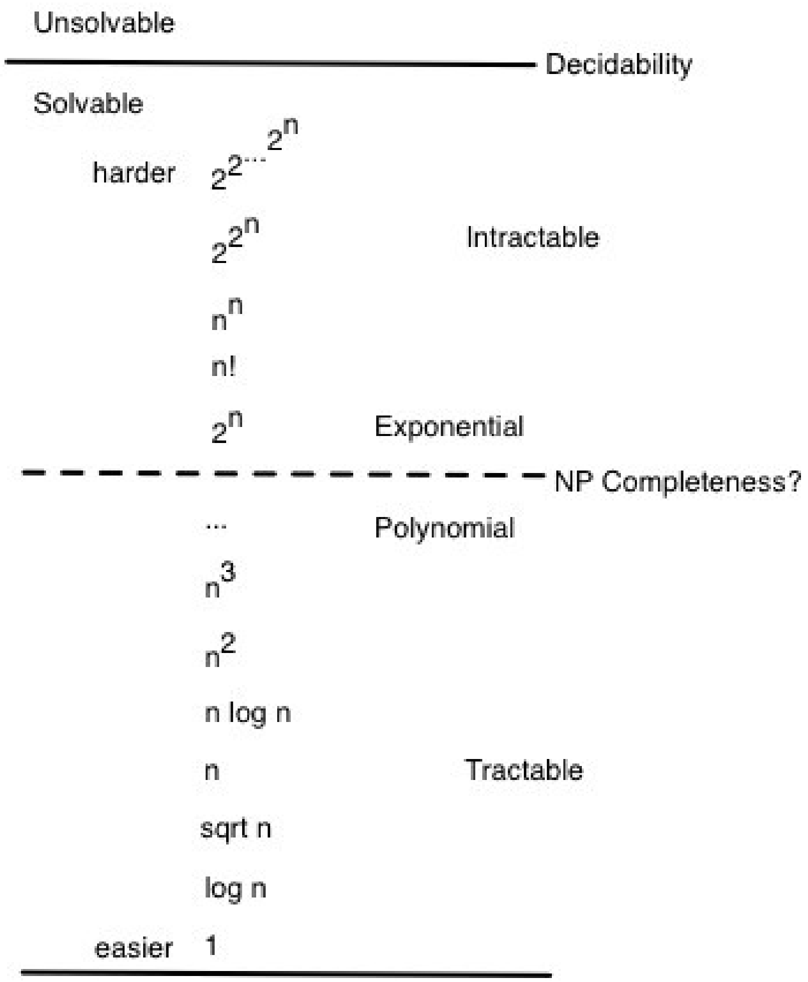
Why does it Matter?



Sort 10 million integers on

- 1 GHZ computer (1000 million instructions per second) using $2n^2$ algorithm.
  - $\frac{2 \cdot (10^7)^2 \; inst.}{10^9 \; inst. \; per \; second} = 200000$ seconds ≈ 55 hours.
- 100 MHz computer (100 million instructions per second) using $50n \log n$ algorithm.
  - $\frac{50 \cdot 10^7 \cdot \log 10^7 \; inst.}{10^8 \; inst. \; per \; second} < \frac{50 \cdot 10^7 \cdot 7 \cdot 3}{10^8} = 5 \cdot 7 \cdot 3 = 105$ seconds.

Complexity Classes


Unsolvable
Decidability
Solvable
harder
$2^{2^{\cdots^{2^n}}}$
2^{2^n}
Intractable
n^n
n!
2^n
Exponential
NP Completeness?
...
Polynomial
n^3
n^2
n log n
n
Tractable
sqrt n
log n
easier
1