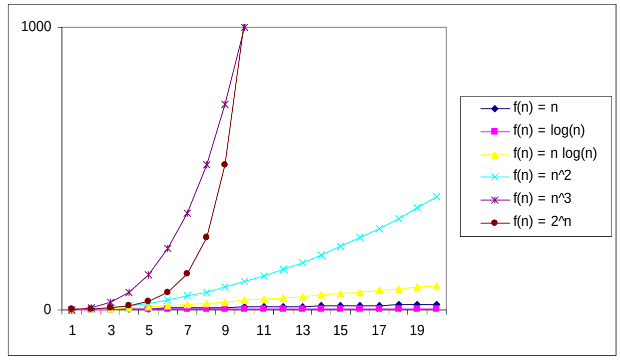
Table of Times

	10	100	300	1000
$5n$	50	500	1500	5000
$n \log n$	33	665	2469	9966
n^2	100	10,000	90,000	10^6
n^3	1000	10^6	27×10^6	10^9
2^n	1024	10^{31}	10^{91}	10^{302}
$n!$	3.6×10^6	10^{161}	10^{623}	∞
n^n	10^{10}	10^{201}	10^{744}	∞

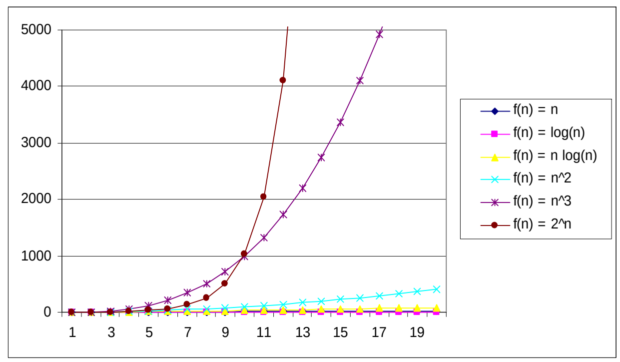
Practical Complexity



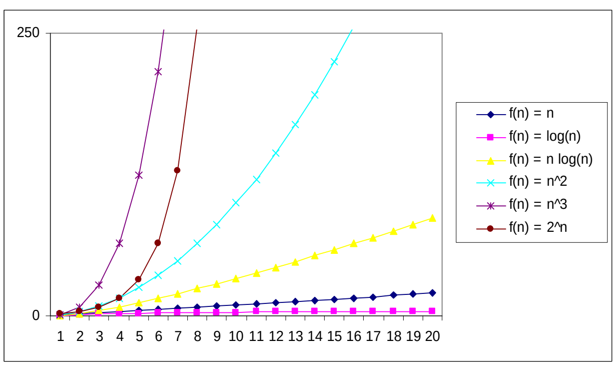
Estimate

- Number of microseconds since Big Bang has 24 digits.
- Number of protons in the universe has 126 digits.

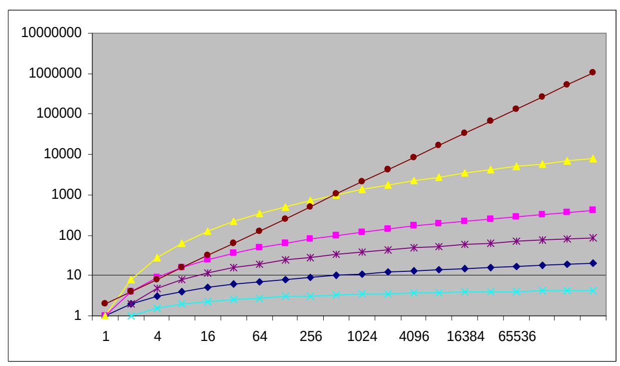
Practical Complexity



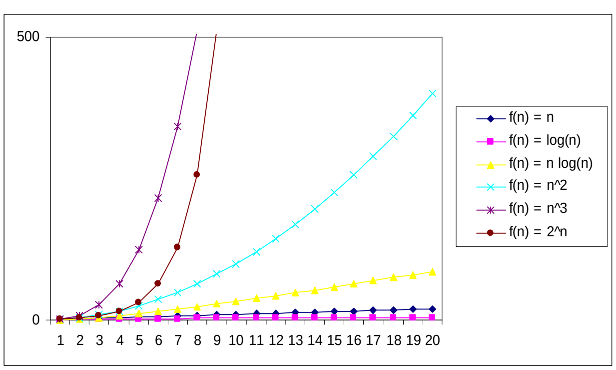
Practical Complexity



Practical Complexity



Practical Complexity



Analysis

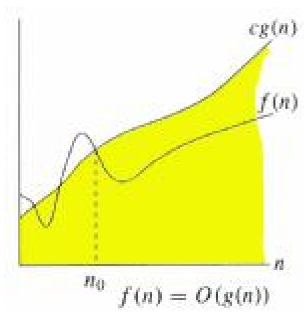
- Simplifications
  - Ignore actual and abstract statement costs
  - *Order of growth* is the interesting measure:
    - Highest-order term is what counts
      - Remember, we are doing asymptotic analysis
      - As the input size grows larger it is the high order term that dominates

Asymptotic Analysis

- Asymptotic = leading term
- 10nlogn + 2n^2-40n has leading term n^2
- Gives simplified but realistic bound
- Smarter coding often improves the constant factors, better algorithms improves the exponent of the leading term.
- E.g. an 25nlogn algorithm scales far better than an 3n^2 algorithm.

Big-Oh Notation

- Used for upper bounds. Think at most.
- f(n) = O(g(n)) if ∃ constants c, n_o such that 0 ≤ f(n) ≤ c x g(n), ∀ n ≥ n_o
- cg(n) dominates f(n) to right of n_o
- We say g(n) is an asymptotic upper bound for f(n).

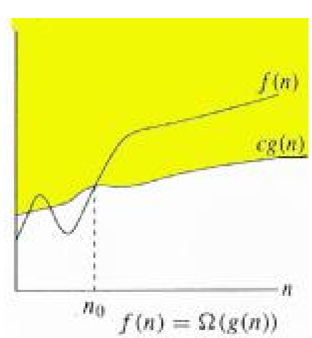


Big-Oh Notation

- Example. 3/2 n^2 + 7/2 n – 4 is O(n^2). Choose c = 2, n_0 = 6. c = 3, n_0 = 1 also work.
- Big Oh is an upper bound estimate, so its okay to say " running time is O(nlogn).
- But it is not okay to say "running time is at least O(nlogn)". Instead, we use Ω notation.

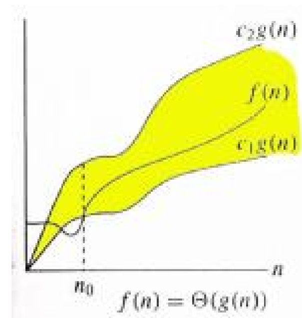
Big Omega Notation

- Used for lower bounds. Think "at least".
- f(n) = Ω(g(n)) if ∃ constants c, n_o such that 0 ≤ c x g(n) ≤ f(n), ∀ n ≥ n_o
- cg(n) is dominated by f(n) to the right of n_o.
- We say g(n) is an asymptotic lower bound for f(n)



Big Theta Notation

- Tight bounds
- f(n) = Θ(g(n)) means that f(n) = O(g(n)) and f(n) = Ω(g(n)).
- We say g(n) is an asymptotically tight bound for f(n).



Relations Between Θ, Ω, *O*

- For any two functions $g(n)$ and $f(n)$, $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
- I.e., $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$

Asymptotic Notations

- f(n) = Θ(g(n)) means f(n) and g(n) grow at the same rate.
- f(n) = O(g(n)) but f(n) ≠ Ω(g(n)), then f(n) is slower growing than g(n).
- f(n) = Ω (g(n)) but f(n) ≠ O(g(n)), then f(n) is faster growing than g(n).

Asymptotic Notations

- Mathematically, use lim n®¥ f(n)/g(n)
- If lim n®¥ f(n)/g(n) = 0, f(n) = O(g(n))
- If lim n®¥ f(n)/g(n) = ∞, f(n) = Ω(g(n))
- If lim n®¥ f(n)/g(n) = c, f(n) = Θ(g(n))

Other Asymptotic Notations

- A function $f(n)$ is $o(g(n))$ if $\exists$ positive constants c and n_0 such that
  $f(n) < c\, g(n) \;\forall\; n \geq n_0$
- A function $f(n)$ is $\omega(g(n))$ if $\exists$ positive constants c and n_0 such that
  $c\, g(n) < f(n) \;\forall\; n \geq n_0$
- Intuitively,
  - $o()$ is like $<$
  - $\omega()$ is like $>$
  - $\Theta()$ is like $=$
  - $O()$ is like $\leq$
  - $\Omega()$ is like $\geq$