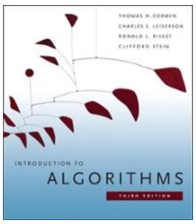
CS146 Data Structures and Algorithms



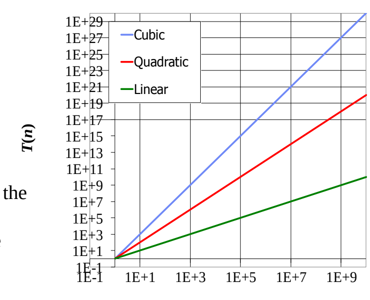
Chapter 3: Growth of Functions

Asymptotic Performance

- *Asymptotic performance*: How does algorithm behave as the problem size gets very large?
  - Running time
  - Memory/storage requirements
  - *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

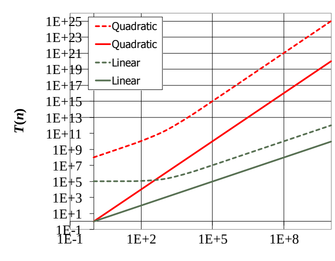
Growth Rates

- Growth rates of functions:
  - Linear $\approx n$
  - Quadratic $\approx n^2$
  - Cubic $\approx n^3$
- In a log-log chart, the slope of the line corresponds to the growth rate of the function

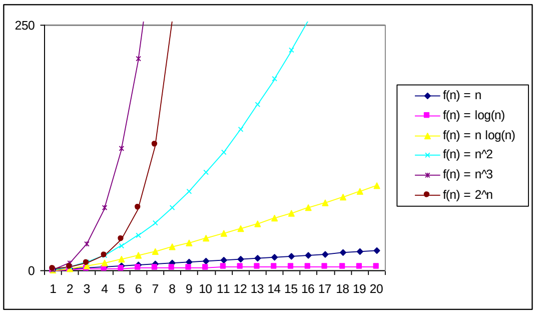


Constant Factors

- The growth rate is not affected by
  - constant factors or
  - lower-order terms
- Examples
  - $10^2 n + 10^5$ is a linear function
  - $10^5 n^2 + 10^8 n$ is a quadratic function

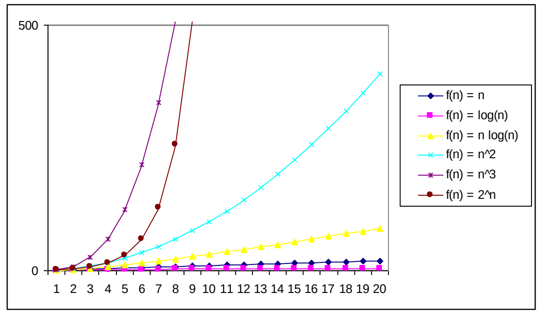


Practical Complexity



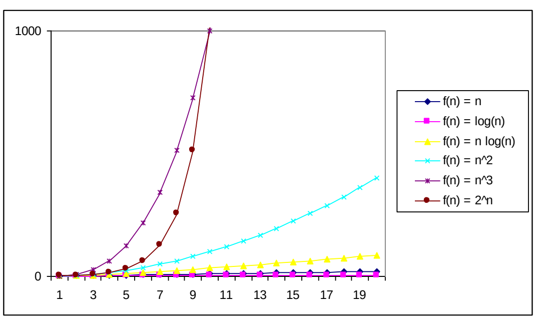
L3.5

Practical Complexity



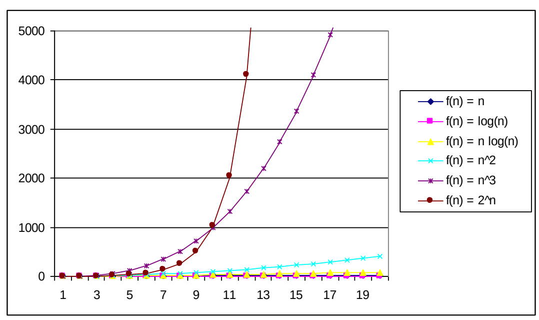
L3.6

Practical Complexity



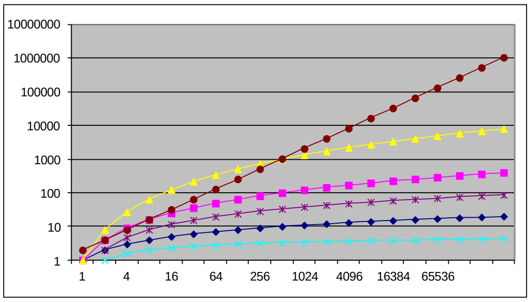
L3.7

Practical Complexity



L3.8

Practical Complexity



When *n* grows to larger number, what is the f(n) for each curve line?

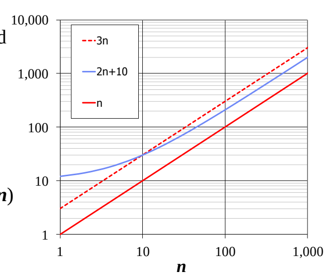
Asymptotic Notation

- Upper Bound Notation:
  - f(n) is O(g(n)) if there exist positive constants c and n_0 such that f(n) ≤ c · g(n) for all n ≥ n_0
  - Formally, O(g(n)) = { f(n): ∃ positive constants c and n_0 such that f(n) ≤ c · g(n) ∀ n ≥ n_0
- Big O fact:
  - A polynomial of degree k is O(n^k)

Big-Oh Notation

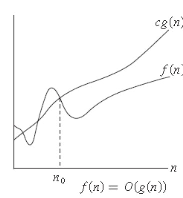
- Given functions ***f*(*n*)** and ***g*(*n*)**, we say that ***f*(*n*)** is ***O*(*g*(*n*))** if there are positive constants ***c*** and $\boldsymbol{n_0}$ such that

  $f(n) \le cg(n)$ for $n \ge n_0$
- Example: $2n + 10$ is $O(n)$
  - $2n + 10 \le cn$
  - $(c - 2)\, n \ge 10$
  - $n \ge 10/(c - 2)$
  - Pick $c = 3$ and $n_0 = 10$



Upper Bound Notation

- We say InsertionSort's run time is *O(n^2)*
  - Properly we should say run time is *in* O(n^2)
  - Read O as "Big-O" (you'll also hear it as "order")
- In general a function
  - f(n) is O(g(n)) if there exist positive constants c and n_0 such that f(n) ≤ c · g(n) for all n ≥ n_0
- Formally
  - O(g(n)) = { f(n): ∃ positive constants c and n_0 such that f(n) ≤ c · g(n) ∀ n ≥ n_0



Insertion Sort Is $O(n^2)$

- Proof
  - Suppose runtime is $an^2 + bn + c$
    - If any of a, b, and c are less than 0 replace the constant with its absolute value
  - $an^2 + bn + c \le (a + b + c)n^2 + (a + b + c)n + (a + b + c)$
  - $\le 3(a + b + c)n^2$ for $n \ge 1$
  - Let $c' = 3(a + b + c)$ and let $n_0 = 1$
- Question
  - Is InsertionSort $O(n^3)$?
  - Is InsertionSort $O(n)$?