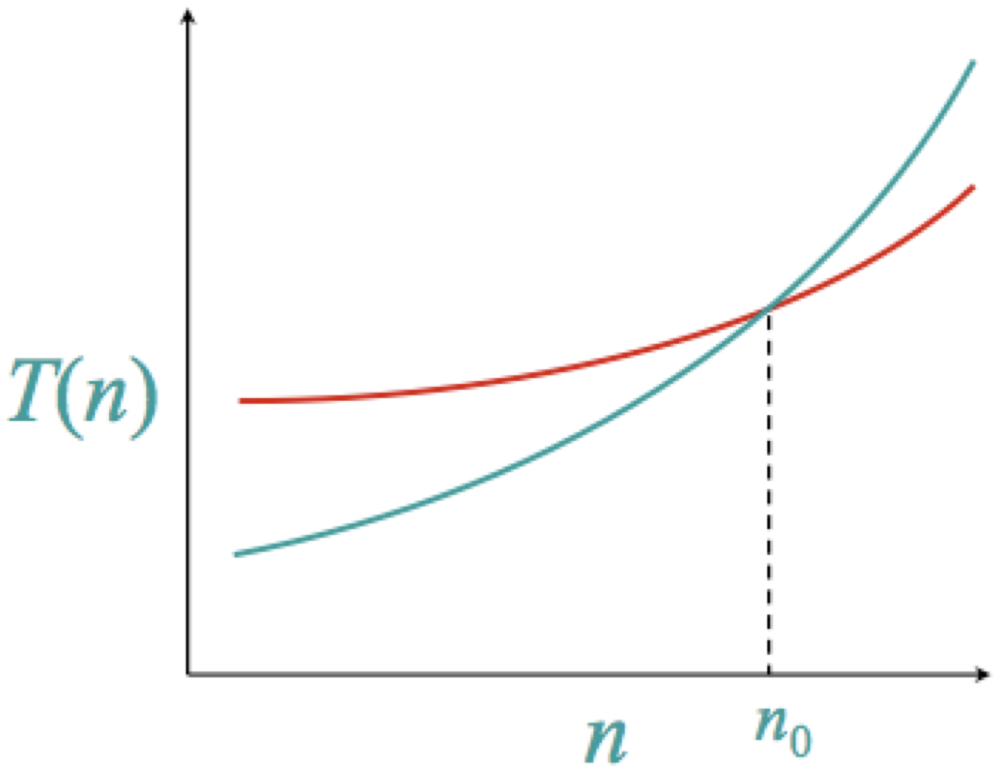
Asymptotic Notation

- Big Θ

- **Definition**: $f(n)$ is in $\Theta(g(n))$ if $f(n)$ is bounded above and below by $g(n)$ (within constant multiple)
  - there exist positive constant c_1 and c_2 and non-negative integer n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for every $n \geq n_0$

- **Examples**:
  - $\frac{1}{2}n(n-1) \in \Theta(n^2)$
    - why?
  - $2n - 51 \in \Theta(n)$
    - why?

Asymptotic analysis



- Sometimes asymptotically slower algorithms work well for small inputs
- Overall we are interested in running time as n gets large

Order of Growth

- Theoretical analysis focuses on ``order of growth'' of an algorithm
- How the algorithm behaves as $n \to \infty$
- Some common order of growth

$$n, n^2, n^3, n^d, \log n, \log^* n, \log\log n, n\log n, n!, 2^n, 3^n, n^n, \sqrt{n}$$

Asymptotic Notation

- Big $O, \Omega . \Theta$
- upper, lower, tight bound (when input is sufficiently large and remain true when input is infinitely large)
- defines a set of similar functions

Big O

- **Definition**: $f(n)$ is in $O(g(n))$ if "order of growth of $f(n)$" $\leq$ "order of growth of $g(n)$" (within constant multiple)
  - there exist positive constant c and non-negative integer n_0 such that $f(n) \leq cg(n)$ for every $n \geq n_0$
- **Examples**:
  - $10n \in O(n^2)$
    * why?
  - $5n + 20 \in O(n)$
    * why?
  - $2n + 6 \notin O(\log n)$
    * why?
- g(n) is an upper bound

Big Θ

- **Definition**: $f(n)$ is in $\Theta(g(n))$ if $f(n)$ is bounded above and below by $g(n)$ (within constant multiple)
  - there exist positive constant c_1 and c_2 and non-negative integer n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for every $n \geq n_0$
- **Examples**:
  - $\frac{1}{2}n(n-1) \in \Theta(n^2)$
    - why?
  - $2n - 51 \in \Theta(n)$
    - why?

- g(n) is a tight bound

Big Ω

For a given function g(n) $\Omega(g(n)) = f(n)$
There exist constant c and n_0 such that:

$$0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0$$

f(n) **grows at least as fast as** g(n); g(n) is asymptotically lower bound.

Example:

$$\sqrt{n} = \Omega(\log n); c = 1, n_0 = 16$$

Asymptotic Notation

- Asymptotic notation has been developed to provide a tool for studying order of growth
  - $O(g(n))$: a set of functions with the same or smaller order of growth as $g(n)$
    * $2n^2 - 5n + 1 \in O(n^2)$
    * $2^n + n^{100} - 2 \in O(n!)$
    * $2n + 6 \notin O(\log n)$
  - $\Omega(g(n))$: a set of functions with the same or larger order of growth as $g(n)$
    * $2n^2 - 5n + 1 \in \Omega(n^2)$
    * $2^n + n^{100} - 2 \notin \Omega(n!)$
    * $2n + 6 \in \Omega(\log n)$
  - $\Theta(g(n))$: a set of functions with the same order of growth as $g(n)$
    * $2n^2 - 5n + 1 \in \Theta(n^2)$
    * $2^n + n^{100} - 2 \notin \Theta(n!)$
    * $2n + 6 \notin \Theta(\log n)$

Useful conventions

- Set in a formula represents anonymous function in the set

$$n^2 + O(n) = O(n^2)$$

$$f(n) = n^3 + O(n^2)$$

Function Comparison

- Verify the notation by compare the order of growth

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & t(n) \text{ has a smaller order of growth than } g(n) \\ c > 0 & t(n) \text{ has the same order of growth as } g(n) \\ \infty & t(n) \text{ has a larger order of growth than } g(n) \end{cases}$$

- useful tools for computing limits

  - L'Hôpital's rule

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

  - Stirling's formula

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

Bounding Functions

- non-recursive algorithms
- set up a sum for the number of times the basic operation is executed simplify the sum
- determine the order of growth (using asymptotic notation)

1. $\sum_{1=1}^{n} 1 = 1 + 1 + \cdots + 1 = n \in \Theta(n)$
2. $\sum_{1=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \approx \frac{n^2}{2} \in \Theta(n^2)$
3. $\sum_{1=1}^{n} i^2 = 1 + 4 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{n^3}{3} \in \Theta(n^3)$
4. $\sum_{1=0}^{n} a^i = 1 + a^1 + \cdots + a^n = \frac{a^{n+1} - 1}{a - 1}, \forall a \neq 1, \in \Theta(a^n)$
5. $\sum a_i + b_i = \sum a_i + \sum b_i$
6. $\sum c a_i = c \sum a_i$
7. $\sum_{1=0}^{n} a_i = \sum_{1=0}^{m} a_i + \sum_{1=m+1}^{n} a_i$