Asymptotic analysis (big-O, etc.)

Why bother analyzing runtime analytically? why not just use a stopwatch?

- Faster!
- More general, not specific to language, hardware, inputs, etc.
- Computers are finicky

---

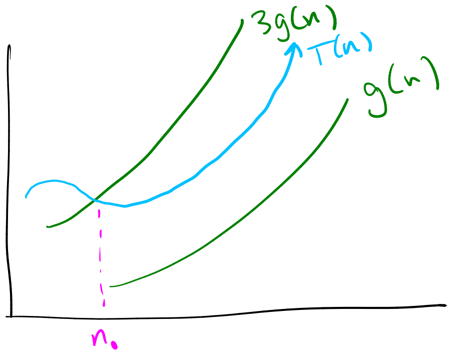
big-O $\approx \leq$, big-$\Omega \approx \geq$, big-$\Theta \approx =$.

"$T(n)$ is $O(n^2)$ if $T(n) \leq c \cdot n^2$ eventually."

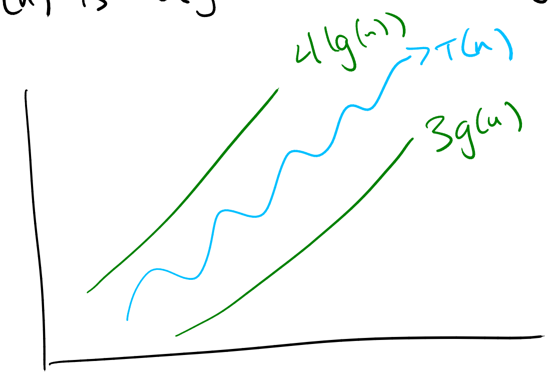
Def'n. If there exists an integer $n_0 \geq 0$ and a real number $c > 0$ such that for all $n \geq n_0$, $T(n) \leq c \cdot g(n)$, then we say

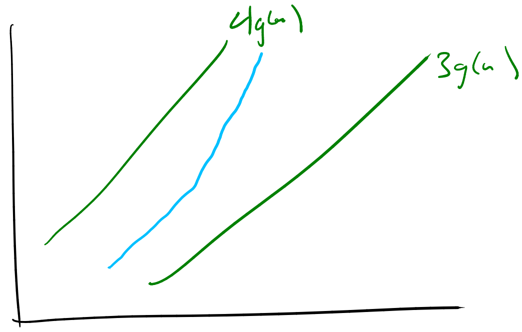
$T(n)$ is $O(g(n))$.

$\geq$ → def'n of $\Omega(g(n))$.



Def'n. If $T(n)$ is $O(g(n))$ and $\Omega(g(n))$, then $T(n)$ is $\Theta(g(n))$.





$3n^2 + 5n - 7$ — intuitively $\Theta(n^2)$, but applying def'ns is tedious.

Use limits — idea is to look at $\lim_{n \to \infty} \frac{T(n)}{g(n)}$.

Theorem. If $0 \le \lim_{n\to\infty} \frac{T(n)}{g(n)} < \infty$, then $T(n)$ is $O(g(n))$.

If $0 < \lim_{n\to\infty} \frac{T(n)}{g(n)} \le \infty$, then $T(n)$ is $\Omega(g(n))$.

If $0 < \lim_{n\to\infty} \frac{T(n)}{g(n)} < \infty$, then $T(n)$ is $\Theta(g(n))$.

eg.

$$\lim_{n\to\infty} \frac{3n^2+5n-7}{n^2} = \lim_{n\to\infty} \frac{3\cancel{n^2}}{\cancel{n^2}} + \lim_{n\to\infty} \frac{5n}{n^2} + \lim_{n\to\infty} \frac{-7}{n^2}$$

$$= 3 + 0 + 0 = 3.$$

Properties of $O/\Omega/\Theta$.

same for Ω, Θ.

- $k \cdot O(f(n)) = O(f(n))$. [if k is a constant, ie. doesn't depend on n.]
  - ie running the same algorithm k times.
- $O(f(n)) + O(g(n)) = O(f(n)+g(n)) = O(\max(f(n), g(n)))$
  - eg $O(n^2) + O(n) = O(n^2)$.
  - ie. running algorithms in sequence.
- $O(f(n)) \times O(g(n)) = O(f(n) \times g(n))$
  - eg. $O(n^2) \times O(n) = O(n^3)$.
  - ie. running nested algorithms.