Asymptotic Analysis

Why bother analyzing (e.g. runtime mathematically (as opposed to just measuring runtime))?

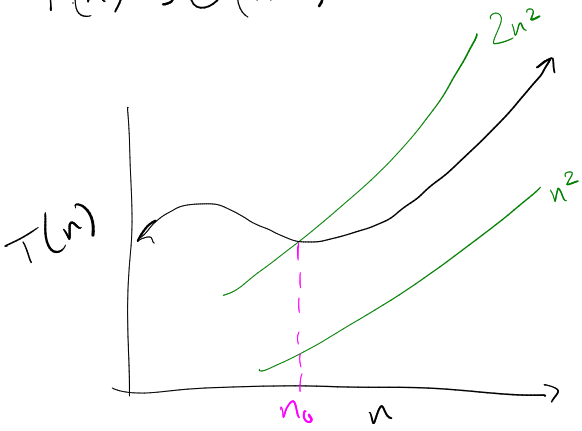
- Can save effort — analysis before actual implementation
- Independent of specific hardware etc.
- Independent of input size

Big-O $\approx$ "$\leq$" Big-Ω $\approx$ "$\geq$" Big-Θ $\approx$ "$=$"

Definitions of these?

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

↑ for big enough n.



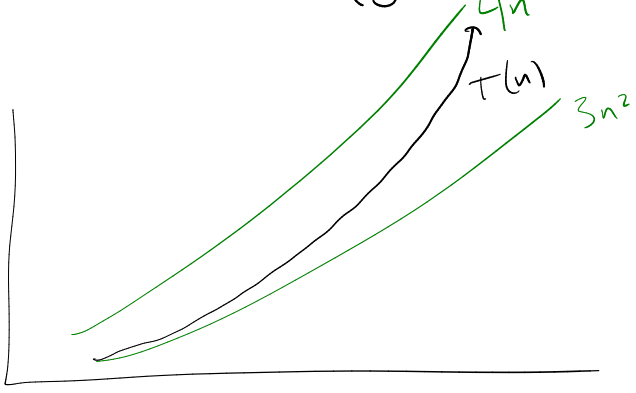
does not depend on n.

Defn $T(n)$ is $O(g(n))$ iff there exists a natural # n_0 and a real constant c such that for all $n \geq n_0$,

$$T(n) \leq c \cdot g(n).$$

$\geq$ → depth of $\Omega(g(n))$.

Defn $T(n)$ is $\Theta(g(n))$ iff $T(n)$ is $O(g(n))$ and $\Omega(g(n))$.



Defn

$T(n)$ is $o(g(n))$ iff $T(n)$ is $O(g(n))$ and not $\Omega(g(n))$.

e.g. $T(n) = 3n^2 + 5n - 7$

$T(n)$ is $\Theta(n^2)$. In practice, applying the definition is very tedious! No one does this.

Formally, we can look at the limit $\frac{T(n)}{n^2}$ as $n \to \infty$.

Theorem.

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

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

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

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

e.g. $T(n) = 3n^2 + 5n - 7$

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

$$= 3 + 0 = 3.$$

Hence $T(n)$ is $\Theta(n^2)$.

Arithmetic properties of big-O/Ω/Θ.

- $k \cdot O(g(n)) = O(g(n))$.
- $O(f(n)) + O(g(n)) = O(f(n) + g(n)) = O(\max(f(n), g(n)))$

  e.g. $O(n^2) + O(n) = O(n^2)$.

  → running algorithms in sequence

- $O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$.

  e.g. $O(n^2) \cdot O(n^3) = O(n^5)$.

  → running nested algorithms.