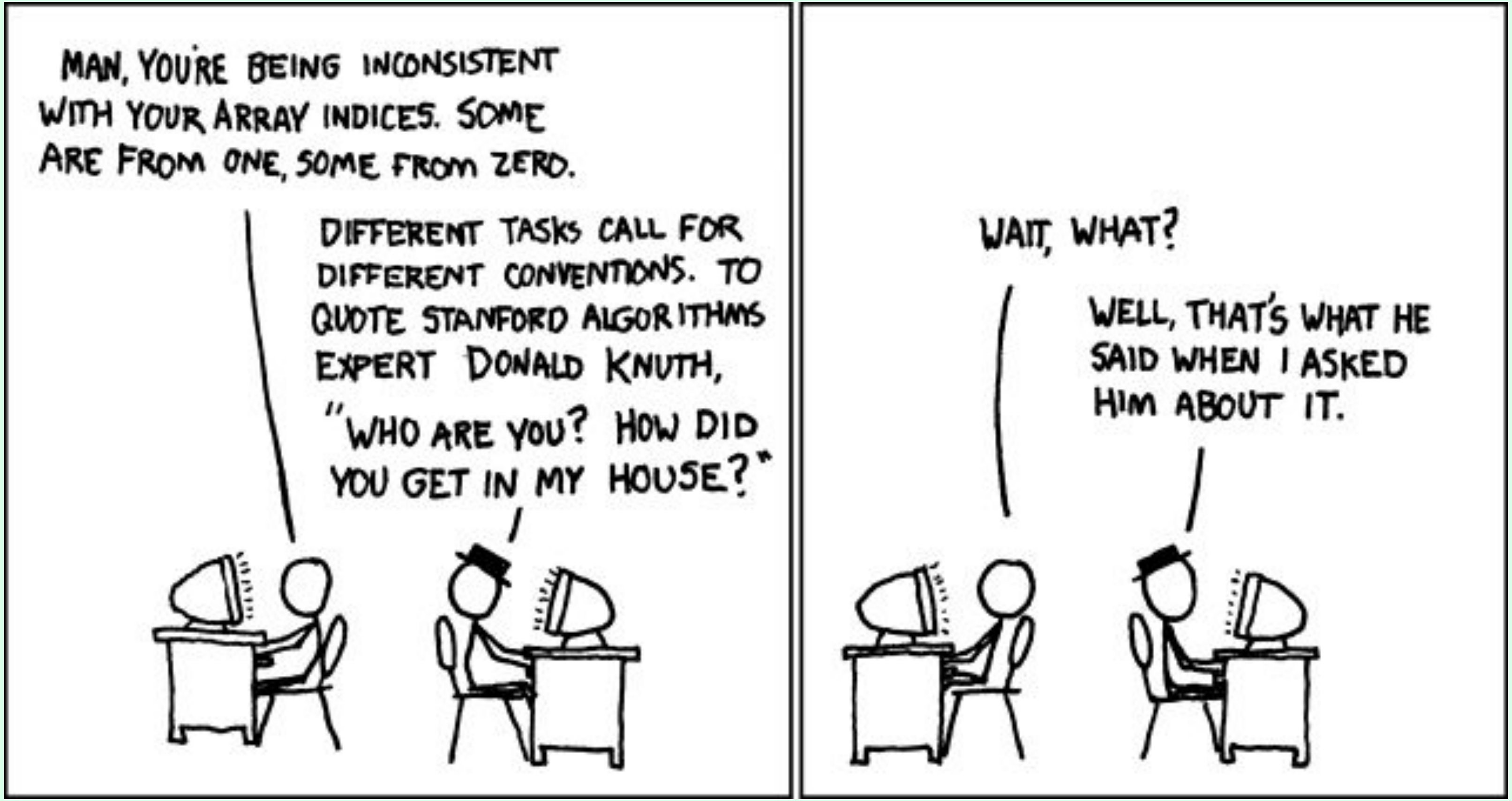
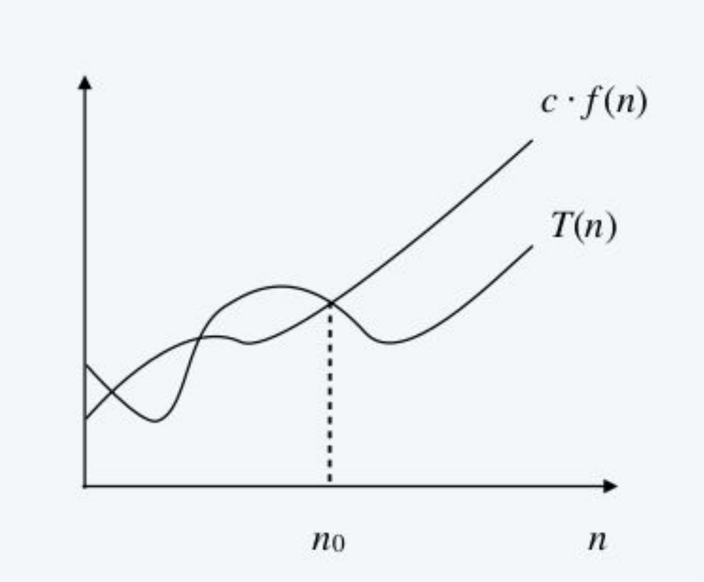
CS 310


MAN, YOU'RE BEING INCONSISTENT WITH YOUR ARRAY INDICES. SOME ARE FROM ONE, SOME FROM ZERO.
DIFFERENT TASKS CALL FOR DIFFERENT CONVENTIONS. TO QUOTE STANFORD ALGORITHMS EXPERT DONALD KNUTH, "WHO ARE YOU? HOW DID YOU GET IN MY HOUSE?"
WAIT, WHAT?
WELL, THAT'S WHAT HE SAID WHEN I ASKED HIM ABOUT IT.

O-notation (big-Oh)

Upper bounds.

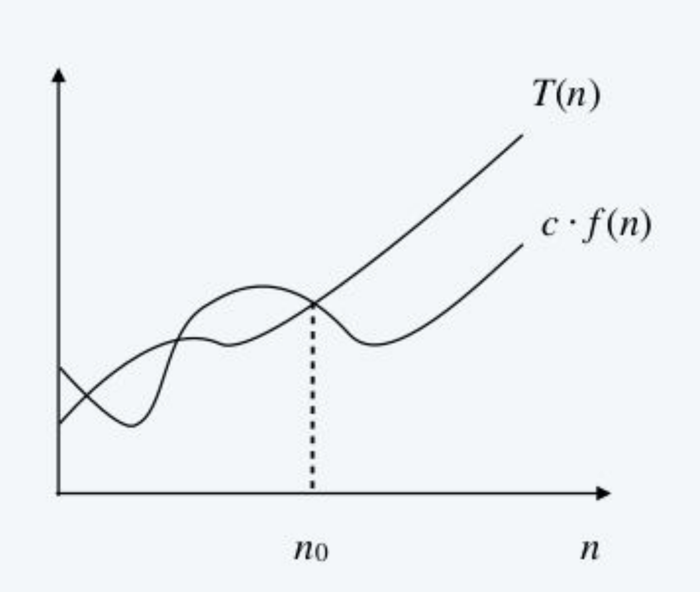
$O(f(n)) = \{T(n)$: there exist positive constants c and n_0 such that $0 \leq T(n) \leq cf(n)$ for all $n \geq n_0\}$



Big-Omega notation

Lower bounds.

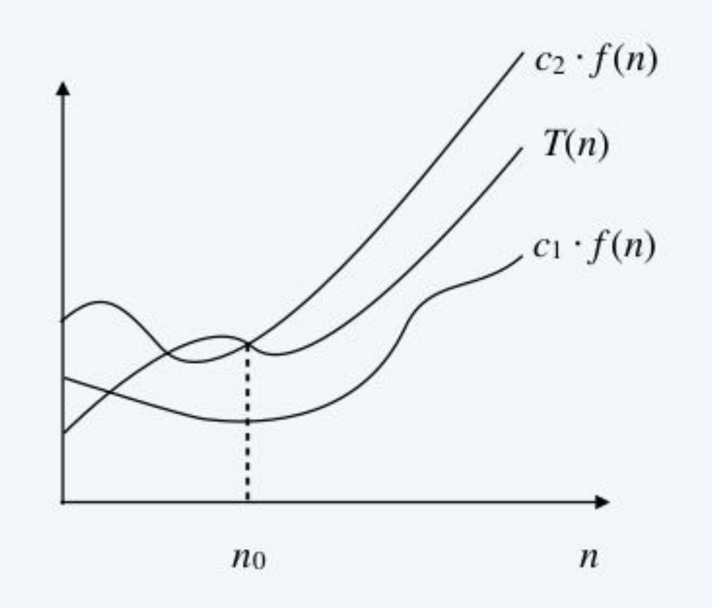
$\Omega(f(n))$ = {T(n) : there exist positive constants c and n_0 such that $0 \leq cf(n) \leq T(n)$ for all $n \geq n_0$}



Big-Theta notation

Tight bounds.

$\Theta(f(n)) = \{T(n)$: there exist positive constants c_1, c_2, and n_0 such that $0 \le c_1 f(n) \le T(n) \le c_2 f(n)$ for all $n \ge n_0\}$



If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$ then $f(n)$ is?

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$ then $f(n)+g(n)$ is?

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$ then $f(n)$ is $O(h(n))$

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$ then $f(n)+g(n)$ is $O(h(n))$

Logarithms

$\log_a n$? $\log_b n$

Logarithms

$\Theta(\log_a n)$ is $\Theta(\log_b n)$ for any constants a, b > 0.

We say that an algorithm is efficient if has a polynomial running time.

Especially those with small constants and small exponents.

Analyzing recursive programs