# CS 6363.003.23S

# Algorithms

Hi, I'm Kyle!
(he/him)

https://personal.utdallas.edu/~kyle.fox/courses/cs6363.003.23s/

Office Hours
Tuesdays 3-4 (not today)
Wed. 10-11am

# Reading

"Required": Algorithms
Erickson

Recommended: Cormen et al.
Intro to Algorithms
(CLRS)

I'll never make you buy the book.

30% homework

(5 assignments)

Consider LaTeX

Two midterms 20% each

Final exam 30%

cumulative

closed notes/book

Homework:

Groups of $\le 2$

- Turn in one copy on elearning.
- Still get full credit with outside sources **if** you cite & write in your own words
- Automatic 24-hour deadline extension if you email me.

# Algorithms:

<u>algorithm</u>: explicit, precise, unambiguous, mechanically-execatable sequence of elementary instructions

Sing "n bottles of beer for any int $n \geq 0$"

**BottlesOfBeer($n$):**
For $i \leftarrow n$ down to 1
  Sing "*$i$ bottles of beer on the wall, $i$ bottles of beer,*"
  Sing "*Take one down, pass it around, $i-1$ bottles of beer on the wall.*"

Sing "*No bottles of beer on the wall, no bottles of beer,*"
Sing "*Go to the store, buy some more, $n$ bottles of beer on the wall.*"

Lattice multiplication:

Given two non-negative ints $x$ & $y$ as

$X[0 \dots m-1]$ & $Y[0 \dots n-1]$

where

$$x = \sum_{i=0}^{m-1} X[i] \cdot 10^i$$

$$y = \sum_{j=0}^{n-1} Y[j] \cdot 10^j$$

want $z = x \cdot y$ as

$Z[0 \dots m+n-1]$ where

$$z = x \cdot y = \sum_{k=0}^{m+n-1} Z[k] \cdot 10^k$$

```
FIBONACCIMULTIPLY(X[0..m−1], Y[0..n−1]):
    hold ← 0
    for k ← 0 to n+m−1
        for all i and j such that i+j = k
            hold ← hold + X[i]·Y[j]
        Z[k] ← hold mod 10
        hold ← ⌊hold/10⌋
    return Z[0..m+n−1]
```

A standard CS student should be able to code up the algo you describe.

# Describing an Algorithm:

What: Specify the input, output, what it <u>accomplishes</u>.

How: Precise description of the algorithm.

Why: Argue correctness (a proof!)

How fast: Big-Oh notation

Know your audience!

use pseudocode + precise English descriptions

"skeptical novice"

What: Specify the exact (mathy) problem you're being asked to solve.

Specify: input & output variables, types, etc.

How: Pseudocode is nice but not strictly necessary.

Why: Prove it works for any input

How fast...

Analysis:

Only care about big inputs.

Constants don't matter (much)

Big-oh notation.

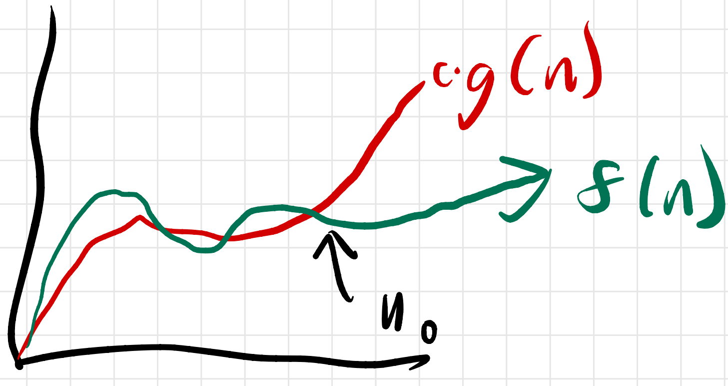
$f(n) : \mathbb{N} \to \mathbb{R}^+$

naturals    reals

$g(n) : \mathbb{N} \to \mathbb{R}^+$

$f(n) \in O(g(n))$ if $f(n)$ grows no quicker than $g(n)$ "up to constant factors"

$O(g(n)) = \{ f(n)$ : there exist pos. constants $c$ & $n_0$ such that $0 \le f(n) \le c g(n)$ for all $n \ge n_0 \}$



constant $c \in O(1)$ for any $c > 0$.

"loose upper bound"

$256n \in O(n)$

$256n \in O(n^2)$

Suppose $f_1(n) \in O(g_1(n))$

$f_2(n) \in O(g_2(n))$

$c \cdot f_1(n) \in O(f_1(n))$ for any constant $c > 0$

sequence of instructions

$f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$

$f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$

$f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$

looping

Often write

$$f(n) = O(g(n))$$