Hi, I'm Kyle!

(he/him)

Office Hours

Tuesdays 3-4 (not today)

Wed. 10-11am

https://personal.utdallas.edu/~kyle.fox/courses/cs6363.003.23s/
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 2
- Turn in one copy on eLearning.
- Still get full credit with outside sources if you cite and write in your own words.
- Automatic 24-hour deadline extension if you email me.
Algorithms:

Algorithm: explicit, precise, unambiguous, mechanically executable sequence of elementary instructions

Sing "n bottles of beer for any int n > 0"

BottlesOfBeer(n):

For i ← 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 \( x \) and \( y \) as \( x[0..m-1] \) and \( y[0..n-1] \)

where

\[
x = \sum_{i=0}^{m-1} x[i].10^i
\]

\[
y = \sum_{j=0}^{n-1} y[j].10^j
\]

want \( z = x \cdot y \) as

\[
z[0..m+n-1] \text{ where } z[k] = \sum_{i=0}^{m} x[i] \cdot y[k-i].10^k
\]
A standard CS student should be able to code up the algo you described.
Describing an Algorithm:

What: Specify the input, output, and what it accomplishes.

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.

\( f(n) : \mathbb{N} \rightarrow \mathbb{R}^+ \)
\( g(n) : \mathbb{N} \rightarrow \mathbb{R}^+ \)
$s(n) \in O(g(n))$ if $s(n)$ grows no quicker than $g(n)$ "up to constant factors"

$O(g(n)) = \{ f(n) : \text{there exist pos. constants } c, n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}$
constant \( c \in O(1) \) for any \( c > 0 \). “Loose upper bound”

\[
256n \in O(n)
\]

\[
256n \in O(n^2)
\]
Suppose \( f_1(n) \in \Theta(g_1(n)) \)
\( f_2(n) \in \Theta(g_2(n)) \)
\( c \cdot f_1(n) \in \Theta(f_1(n)) \) for any constant \( c > 0 \)
\( f_1(n) + f_2(n) \in \Theta(g_1(n) + g_2(n)) \)
\( f_1(n) + f_2(\log n) \in \Theta(\max \{g_1(n), g_2(n)\}) \)
\( f_1(n) \cdot f_2(n) \in \Theta(g_1(n), g_2(n)) \)

looping
Often write

\[ f(n) = O(g(n)) \]