Discrete Structures for Computer Science

William Garrison
bill@cs.pitt.edu
6311 Sennott Square

Lecture #1: Course Introduction

University of Pittsburgh

Administrivia

CS 441: Discrete Structures for Computer Science

Instructor:

William Garrison
bill@cs.pitt.edu
6311 Sennott Square (hybrid)
OH: Appts. on Canvas
(by email, if none posted work)

TAs:

See syllabus site

https://bill-computer.science/441

Course meeting times

- Lecture
  - B: SENSQ 5502, T/H 11:00–12:15
  - C: SENSQ 5502, T/H 2:30–3:45
- Recitations
  - Wednesdays (check Peoplesoft for time and place)
  - No recitations this week (Jan 10)
- It is important to attend ***both*** lecture and your assigned recitation section regularly!

Grading

- Overall breakdown:
  - 30% Midterm exam
  - 30% Final exam
  - 20% Homework
  - 10% Recitation exercises
  - 10% Lecture participation (Top Hat)

  - 100%

Note: Course details are subject to change quickly.
Check your Canvas notifications so you catch all announcements!

Lecture

Slides, with breaks to work on problems individually and/or together on the board

- I'll continue using a tablet as the "board" so everyone can see

Some problems will be submitted via Top Hat

- You will be added to Top Hat automatically before Week 2; install the mobile app or bookmark https://app.tophat.com
- TH questions allow me to gauge what everyone is understanding in the moment
- Opportunities later in the term to make up missed questions
- I will usually start with a question or two recapping the assigned reading at a high level

Recitation

Recitation exercises

- Covers the previous week's material
- Assigned Tuesday, due Thursday at 11:59 PM
- Submit your own work; *primarily* graded for completion
- Late submissions are not accepted—submit early and often
- One recitation exercise grade will be dropped
  - To accommodate undocumented minor illness or personal conflict

Recitations are collaborative

- You should attempt the exercises in advance of recitation
- Discuss solution techniques and details with classmates
- You'll go over most of the solutions in your recitation meeting
- Submit solutions that you understand—do not just copy what your TA shows or copy a friend's submission

Homework

Weekly homework assignments (from the textbook)

- Assigned on Thursday, due the next Tuesday at 11:59 PM
- Late homework is not accepted—submit early and often
- Optional challenge problem(s) in each
- One homework grade will be dropped
  - Again, as an accommodation for undocumented circumstances

Homework may be **discussed** with others, but details must be worked through and written up **individually**

- Limit discussion to understanding problems and developing high-level solution tactics (Do we want a discussion board for this?)
- Identify collaborators on your homework submission
- No outside resources permitted when writing your solutions
- Failure to comply with this policy is a violation of academic integrity (F in the course)

This gives you a lot of opportunities to practice!

Each problem you see on a topic will be slightly higher pressure

- First, read the chapter and think about the study questions as you go
- Then, come to lecture and complete pen-and-paper problems
- Some shorter problems will be submitted for participation on TH
- Next, (collaborative) recitation is collected, mainly for completion
- Finally, (individual) homework is graded carefully
- To give you a chance to get familiar with the standard, the first two recitations and homework will not count toward your grade

	Week 1* material	**Week 2 material**
Lecture and TH	1/11 and 1/16	1/18 and 1/23
Recitation	1/17	1/24
Homework	1/18 through 1/23	1/25 through 1/30

Flipped Review Sessions

To help you study as we go, I'd like to offer review sessions every 2–3 weeks

- The first is planned around Lecture 6

To avoid losing class time, I'll post asynchronous videos of the main material I would usually present in lecture

- You'll be responsible for watching this **before** the review session
- In the recordings, I'll skip the "hands on" parts and we'll do them together

These review sessions are your chance to ask longer questions, work through hard problems from the text, review already-submitted homework problems…

Policies

- Check the syllabus site and Canvas frequently. Announcements, homework, and lecture slides will be posted there.
- We will drop your lowest homework and recitation score before computing your homework average—no documentation necessary!
  - These are "sick days" you can take anytime
  - Reach out if you're impacted by a documented emergency
- Regrade requests accepted within 2 weeks of hand-back. Grade may increase, decrease, or stay the same.
- Other policies are on the web page, where we'll go next
  - Accommodating students with disabilities
  - Religious observances
  - …

Questions?

Course overview

- What *is* discrete mathematics?
- Why is a math course part of the computer science curriculum?
- Will I really ever use this stuff again?
- How to succeed in this course

What is discrete mathematics?

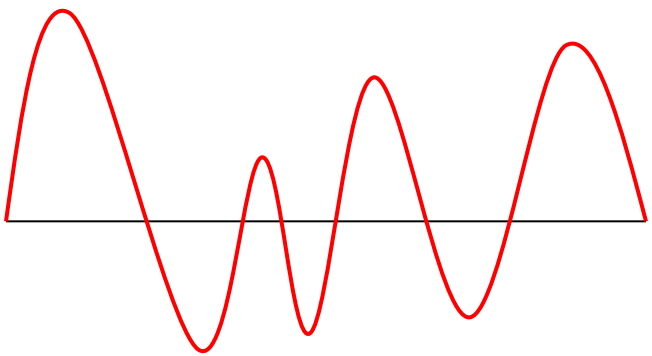
- Discrete mathematics is the study of *distinct* objects or structures and their relationships to one another

- For example:
  - How many ways can a valid password be chosen?
  - Can traffic flow between two computers in a network?
  - How can we transform messages to hide their contents?
  - How do we parse a given sequence of commands?

- By contrast, continuous mathematics (e.g., calculus) studies objects and relationships that vary continuously
  - e.g., position, velocity, and acceleration of a projectile

Why study discrete math?

Reason 1: Computers do not process continuous data



Analog (continuous) input

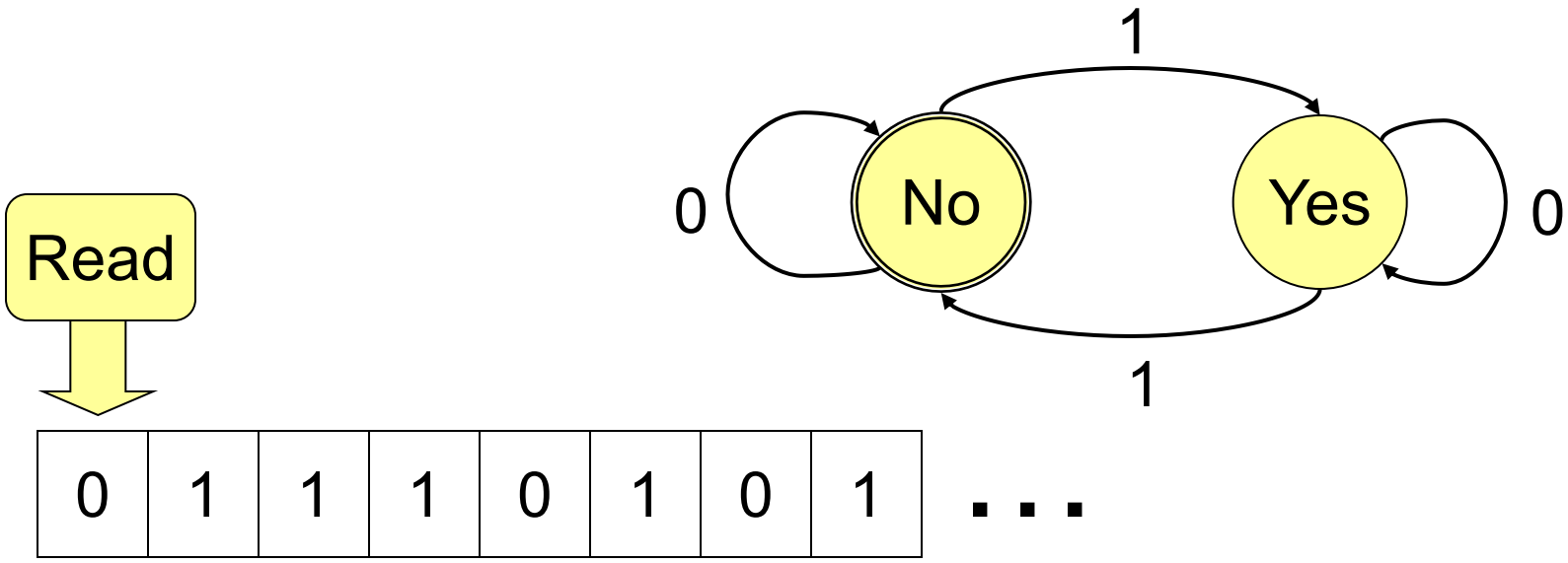
Sampling and discretization

Digital (discrete) output

Why study discrete math?

Reason 2: Computers aren't actually all that smart, they are just deterministic functions that map discrete inputs to discrete outputs

Example: Does a given string contain an odd number of 1s?



Why study discrete math?

In general: Discrete mathematics allows us to better understand computers and algorithms

```
function fib(int n)
  if(n == 0 || n == 1)
    return 1;
  else
    return fib(n-1) + fib(n-2);
```

```
function fib(int n)
  int first = 0;
  int second = 1;
  int tmp;
  for(i = 1 to n)
    tmp = first + second;
    first = second;
    second = tmp;
  end for
  return first;
```

Tentative Syllabus

- Logic and proofs
- Sets
- Functions
- Algorithms and analysis
- Integers, modular arithmetic, cryptography
- Induction
- Relations

Are these topics really useful?

Logic and proofs

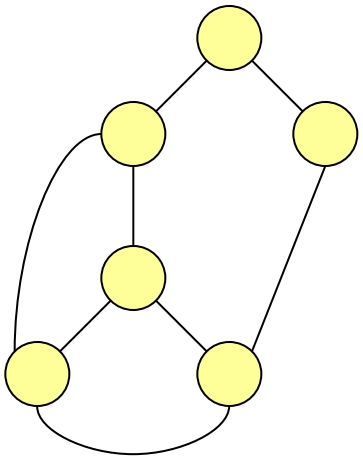
```
grant(X, projector) :- role(X, presenter), located(X, 104)
located(adam, 104)
role(adam, presenter)

=> ?grant(adam, projector)
=> true
```

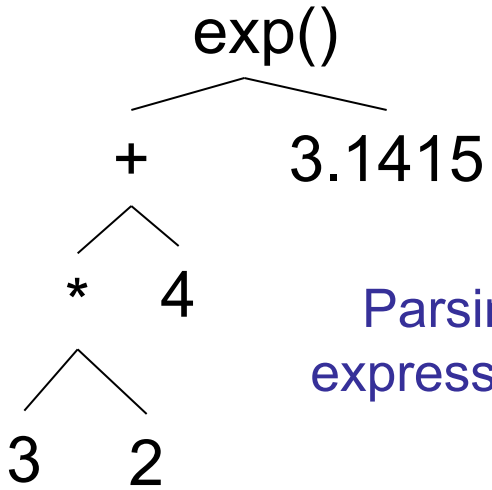
Automated reasoning, AI, security

```
function fib(int n)
  int first = 0;
  int second = 1;
  int tmp;
  for(i = 1 to n)
    tmp = first + second;
    first = second;
    second = tmp;
  end for
  return first;
```

Algorithm and protocol analysis

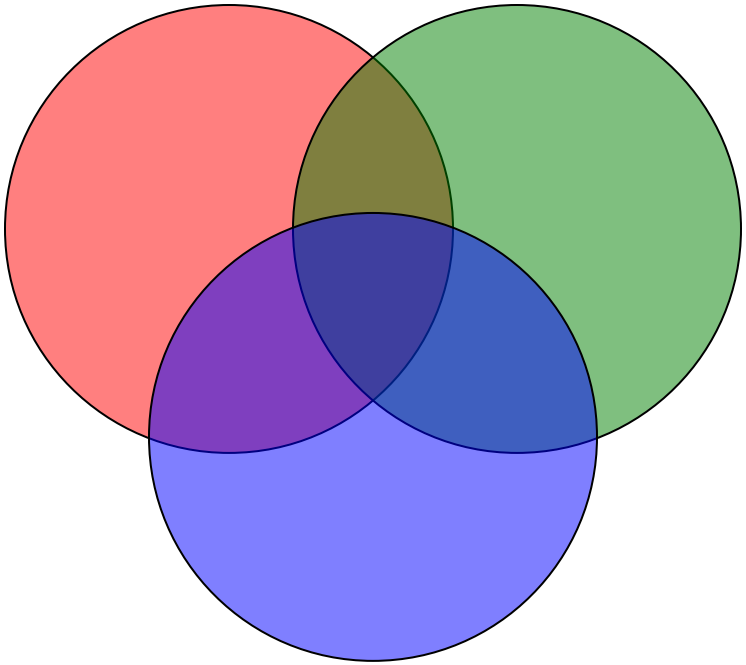


Verifying data structures and hardware



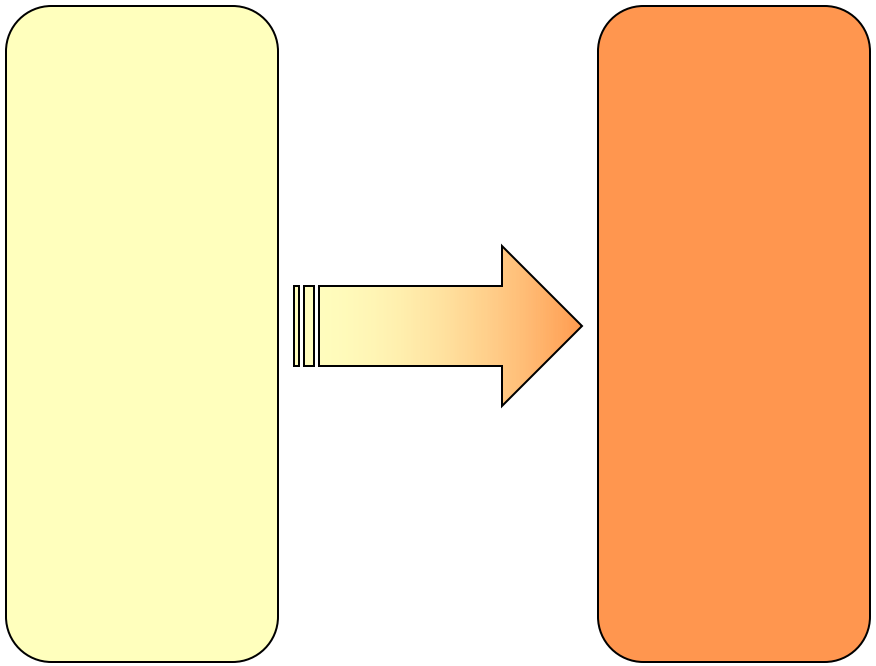
Parsing expressions

Sets

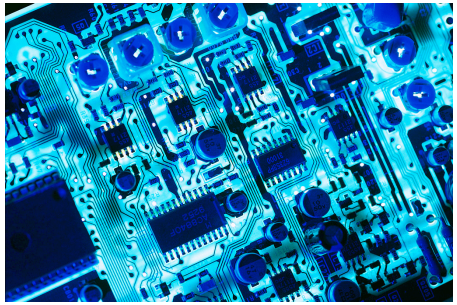


Sets define collections of objects…

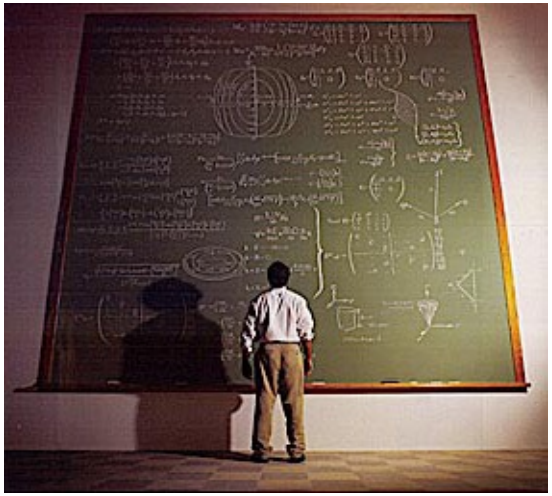
… and give us a means of reasoning about the relationships between objects



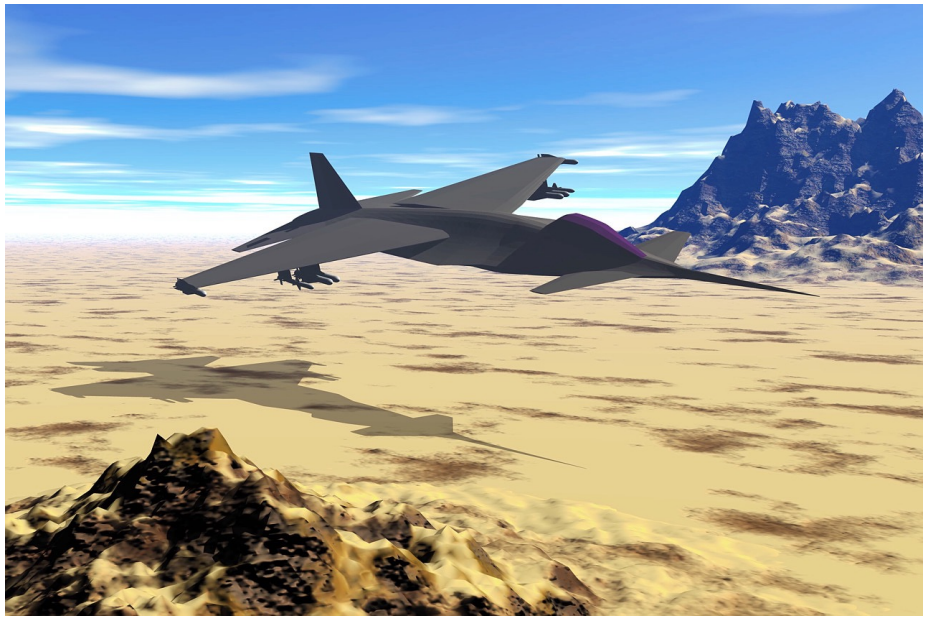
Functions



Hardware design

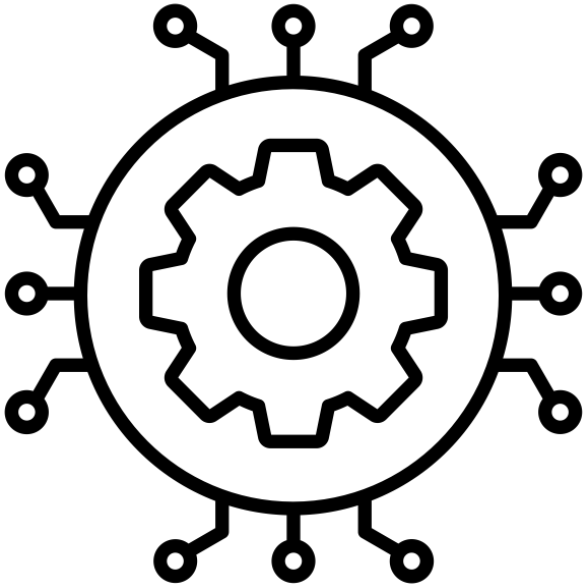


Theory of computation



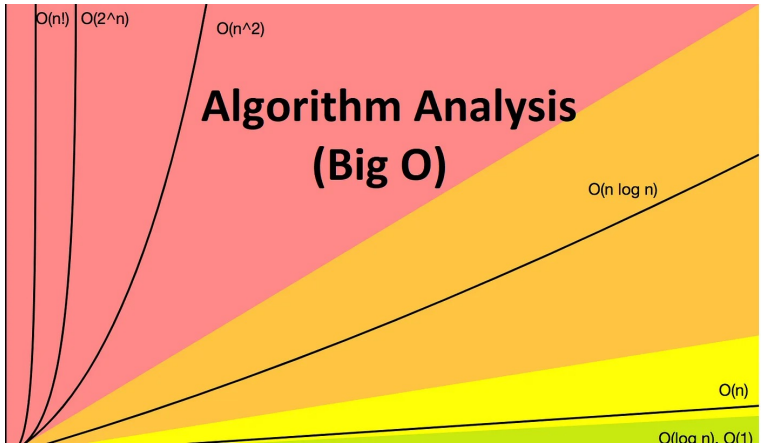
Computer graphics

Algorithms and analysis



Studying algorithms helps us write better code…

… and algorithm analysis helps us determine which approaches scale best



Integers and Modular Arithmetic

```
  0111 0101 0110 1011
+ 0101 1001 1110 0001
-------------------
  1100 1111 0100 1100
```

Binary arithmetic and bitwise operations

ATTACK AT DAWN

01 20 20 01 03 11 01 20 04 01 23 14

$C = P + 6 \pmod{26}$

06 25 25 06 09 16 06 26 10 06 03 20

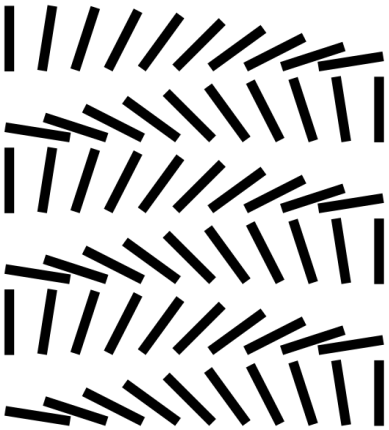
FYYFIPFZJFCU

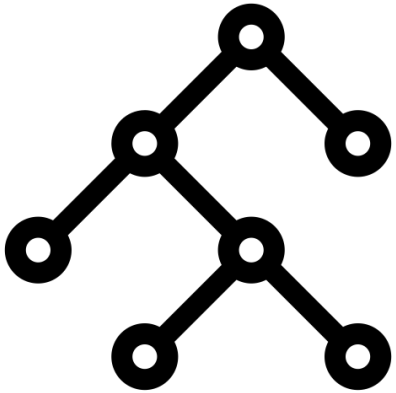
Cryptography

Induction

Induction is a proof technique that helps us reason about infinite objects...

... and processes...

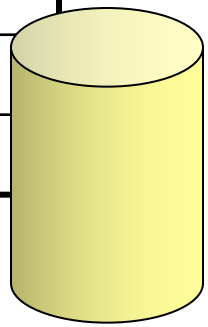




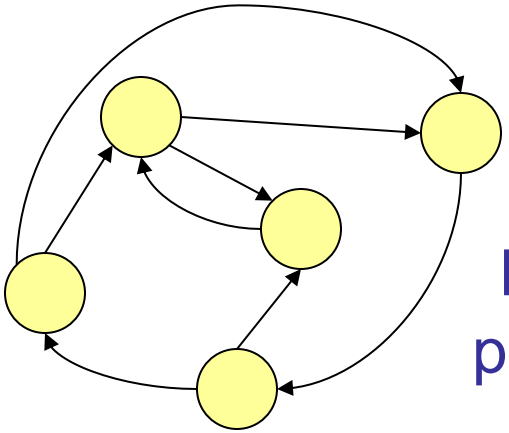
... and data structures!

Relations

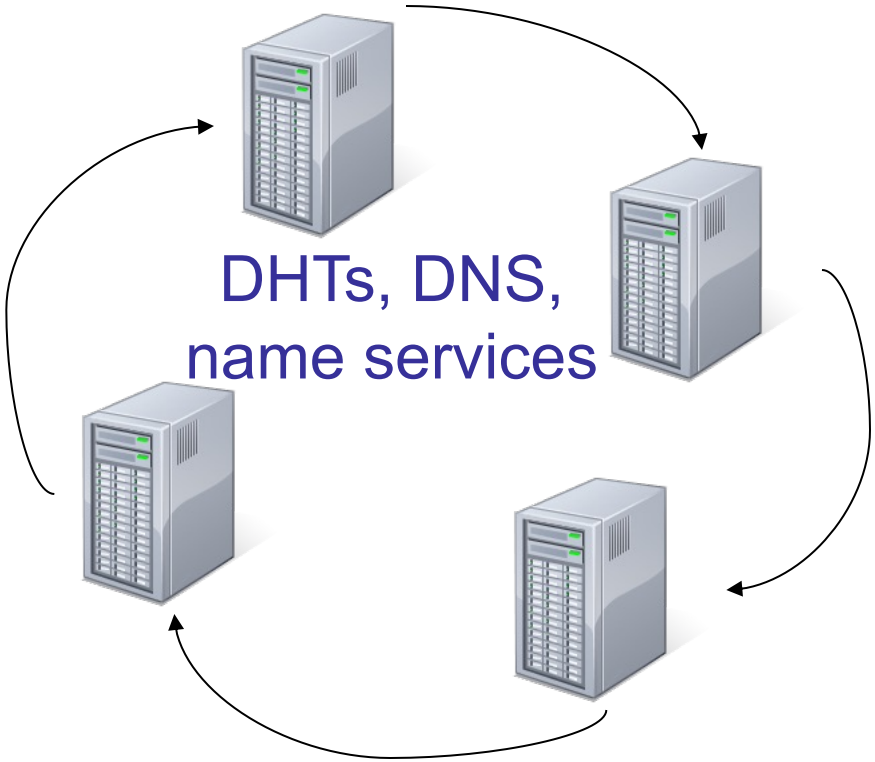
Name	Age	Phone
Alice	19	555-1234
Danielle	33	555-5353
Zach	27	555-3217
Charlie	21	555-2335



Relational databases



Route planning



DHTs, DNS, name services

Syllabus, redux

- Logic and proofs
- Sets
- Functions
- Algorithms and analysis
- Integers, modular arithmetic, cryptography
- Induction
- Relations

Are these topics really useful?

Yes

Mastering discrete mathematics requires practice!

- Succeeding in this class requires practicing the skills that we will acquire, thinking critically, and asking questions
  - We are practicing clear and precise communication in the language of mathematics and logic—be specific!

- Keys to success:
  - Attend class and take notes
  - Do your homework
  - Work extra problems when you're unsure
    - Solutions to odd-numbered exercises provided in textbook
  - Go to your recitation every week
  - Take advantage of office hours

What should I do now?

1. Check your Canvas notification settings
2. Read the chapter for next lecture
3. Decide if you need Inclusive Access, and opt out if not
4. Install Top Hat if you plan to use the mobile app
   - Then, wait to be added to the course on TH
5. Watch for a Gradescope invitation, where you'll submit recitation and homework assignments
6. Email me if you have any special circumstances that you may need accommodated

Final thoughts

- Our goal is to prepare you to be stronger computer scientists by:
  - Exploring the formal underpinnings of computer science
  - Developing critical thinking skills
  - Articulating ties between theory and practice

- Next: Propositional logic