# Discrete Structures

## CSE 2315 (Spring 2014)

## Lecture 1 Introduction

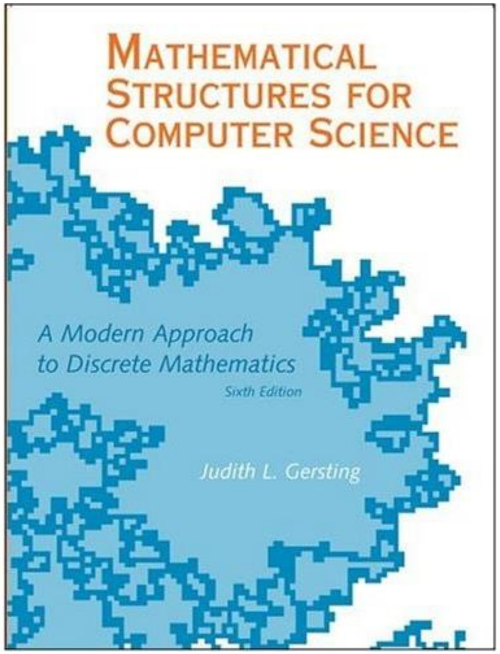
Heng Huang, Ph.D.
Department of Computer Science and Engineering

# Administration

- Course meeting times
  - When: Tue. & Thur. 3:30 ~ 4:50pm
  - Where: GACB 105
  - Lecturer: Heng Huang (Office ERB 533)
    heng@uta.edu
  - Office hour: Tue. & Thur. 1:50pm-3:30pm, 5pm-5:30pm
  - Home page: http://ranger.uta.edu/~heng/CSE2315.html
- Lecturer
  - PhD in CS from Dartmouth College
  - Research areas: algorithms, machine learning, bioinformatics, computer vision

# Study Materials

- Prerequisites:
  - Intermediate Programming (CSE 1320)
  - Calculus I (Math 1426)
- Textbook:
  - *Judith L. Gersting, Mathematical Structures for Computer Science, 6th Edition, W.H. Freeman and Company, 2006.*



# Study Materials

- Textbook
  - We will not cover all the chapters of the book
  - We will not cover all sections of all covered chapters
  - We will not fully follow the order of the book
  - In few cases the notation and terminology will be different from that of the book
  - The contents uncovered in slides/lectures are not required

- Other book for reading:
  - *Kenneth Rosen, Discrete Mathematics and Its Applications, 6th Edition, McGraw Hill Publishing Company, 2007.*
  - More mathematics, a little bit boring
  - Our textbook is better for computer science major

# Grading

- Overall breakdown
  - 25%   Homework sets
  - 25%   Quizzes (3 best from 4 will be counted)
  - 20%   Midterm
  - 25%   Final exam
  - 5%   Class Participation

    100%

- Four quizzes
  - The best three will be counted to allow for illness, excused absences, or other reasons

- Note: Homework is as important as **any** other aspect of your grade!

# Homework

- Homework assignments
  - Assigned in class, typically due one week later at the **start** of lecture
  - Automatic 10% deduction for each day late
  - Homework is not accepted more than 2 days late

- Homework may be discussed with others, but must be written up **individually**
  - Limit discussion to understanding problems and developing solution tactics
  - Identify collaborators on your homework cover sheet
  - Failure to comply with this policy is a violation of academic integrity

- Start early, Start early, Start early, Start early, Start early

# Policies

- Check the web page 2-3 times per week. Announcements, homework, and lecture notes will be posted there.

- If necessary, we will allow regrade requests. However, we reserve the right to regrade the *entire* assignment, not just the portion in question. TA will grade homework, quizzes, and exams.

- Other policies are on the web page
  - Accommodating students with disabilities
  - Student Support Services
  - Etc.

# 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 Mathematics, really?

- It's *not* just about numbers!
- Mathematics is much more than that:

> Mathematics is, most generally, the study of any and all *absolutely certain* truths about any and all *perfectly well-defined* concepts.

- But, the concepts can relate to numbers, symbols, visual patterns, or *anything*!

# What is discrete mathematics?

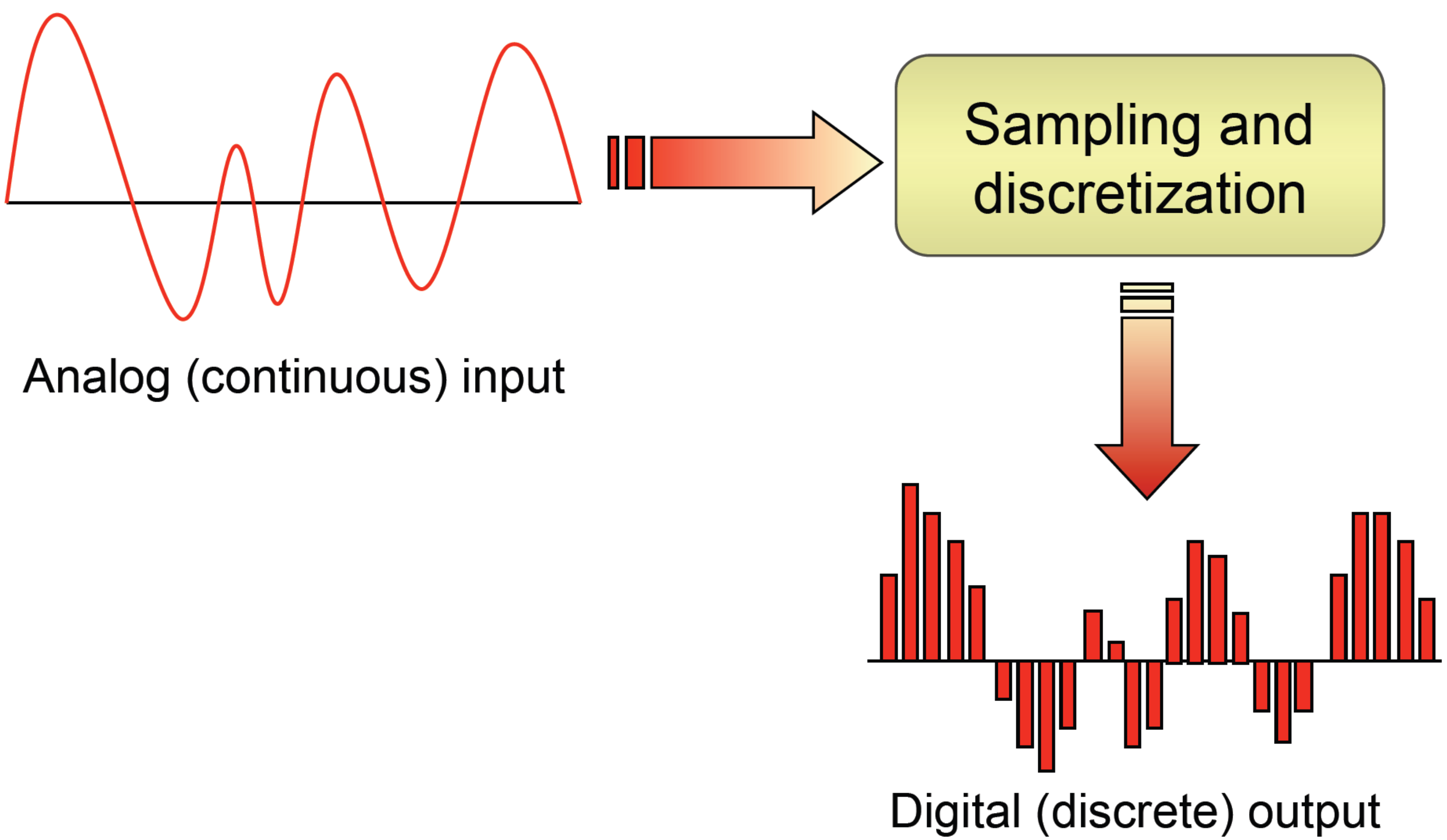
- Discrete mathematics is the study of *distinct* objects or structures and their relationships to one another

- 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?

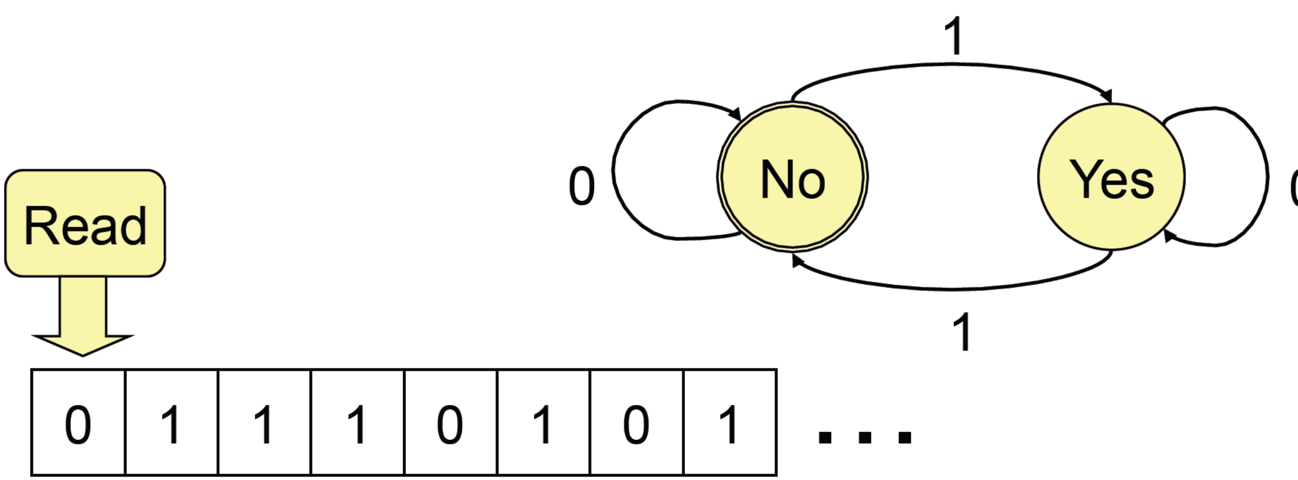
- **Reason 1:** Computers do not process continuous data



# Why study discrete math?

- **Reason 2:** Computers aren't actually all that smart, they are really 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;
```

# Why study discrete math?

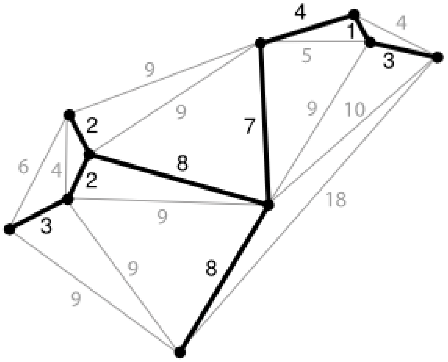
- Computers use discrete structures to represent and manipulate data.
- CSE 2315 is the basic building block for becoming a Computer Scientist
- Computer Science is not Programming; Computer Science is not Software Engineering
- Edsger Dijkstra: “Computer Science is no more about computers than Astronomy is about telescopes.”
- Computer Science is about **problem solving**.

# Why study discrete math?

- Mathematics is at the heart of problem solving
- Defining a problem requires mathematical rigor
- Use and analysis of models, data structures, algorithms requires a solid foundation of mathematics
- To justify why a particular way of solving a problem is correct or efficient (i.e., better than another way) requires analysis with a well-defined mathematical model.

# Problem Solving requires mathematical rigor

- Your boss is not going to ask you to solve an MST (Minimal Spanning Tree) or a TSP (Travelling Salesperson Problem)





- However, he/she may ask you to build a rotation of the company's delivery trucks to minimize fuel usage

- It is up to you to determine
  - a proper model for representing the problem and
  - a correct or efficient algorithm for solving it.

# Scenario I

- A limo company has hired you/your company to write a computer program to automate the following tasks for a large event

- **Task1**: In the first scenario, businesses request
  - limos and drivers
  - for a fixed period of time, specifying a start data/time and end date/time and
  - a flat charge rate.

- The program must generate a schedule that accommodates the maximum number of customers

# Scenario II

- **Task 2**: In the second scenario, the limo service allows customers to bid on a ride to that the highest bidder get a limo when there aren't enough limos available.

- The program should thus make a schedule that
  - Is feasible (no limo is assigned to two or more customers at the same time)
  - While maximizing the total profit

# Scenario III

- Task 3: Here each customer is allowed to specify a set of various times and bid an amount for the entire event. The limo service must choose to accept the entire set of times or reject it.

- The program must again maximize the profit.

# What's your job?

- Build a mathematical model for each scenario
- Develop an algorithm for solving each task.
- Justify that your solutions work
  - Prove that your algorithms terminate. Termination
  - Prove that your algorithms find a solution when there is one. Completeness
  - Prove that the solution of your algorithms is correct Soundness
  - Prove that your algorithms find the best solution (i.e., maximize profit). Optimality (of the solution)
  - Prove that your algorithms finish before the end of life on earth. Efficiency, time and space complexity

# The goal of this course

- The goal of this course is to give you the foundations that you will use to eventually solve these problems.
  - Task1 is easily (i.e., efficiently) solved by a greedy algorithm
  - Task2 can also be (almost) easily solved, but requires a more involved technique, dynamic programming
  - Task3 is not efficiently solvable (it is NP-hard) by any known technique. It is believed today that to guarantee an optimal solution, one needs to look at all (exponentially many) possibilities.

# Tentative Syllabus

- Logic and proofs
- Sets
- Functions
- Integers and modular arithmetic
- Counting
- Probability and expectation
- Relations

**Are these topics really useful?**

# Logic and proofs

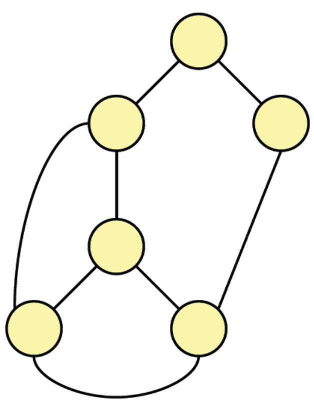
```
grant(X, projector) :- role(X, presenter), located(X, 104)
located(adam, 104)
role(adam, presenter)

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

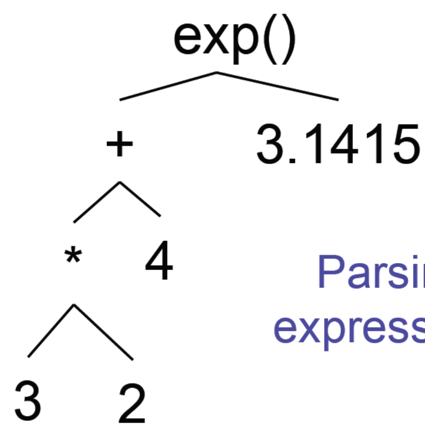
Automated reasoning, AI, security

```
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

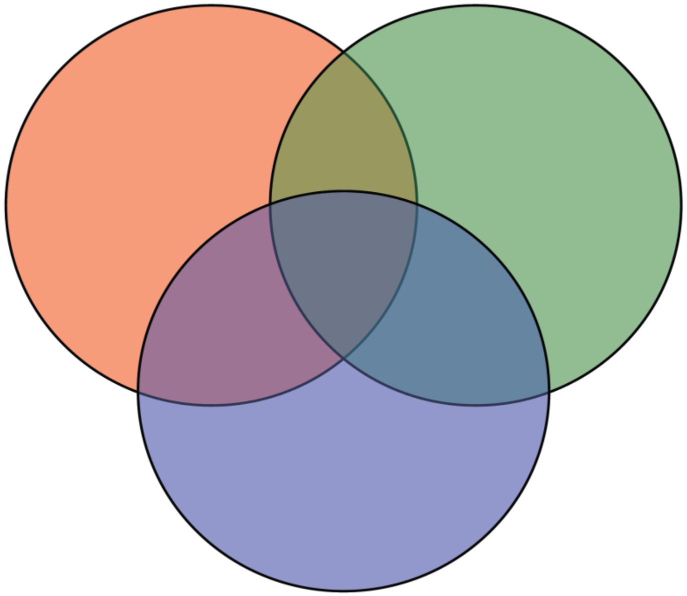


Verifying data structures and hardware



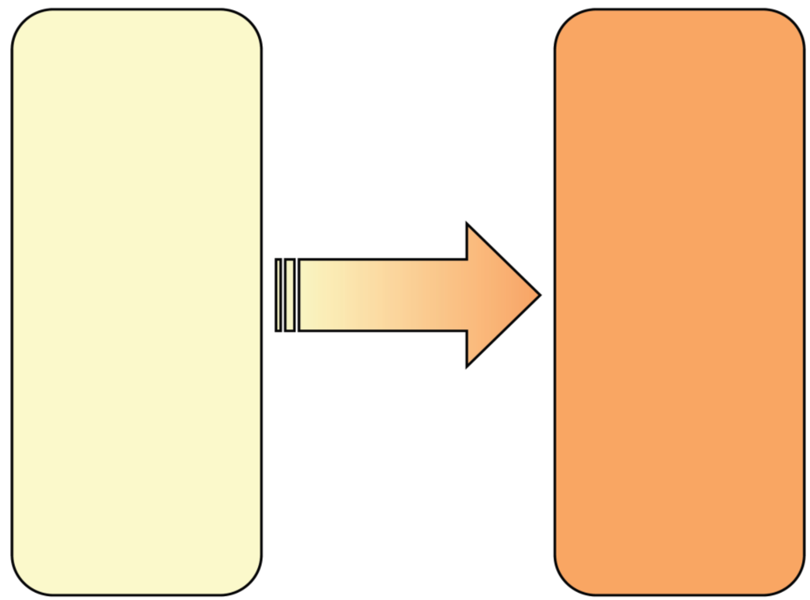

Parsing expressions

# Sets



Sets define collections of objects…

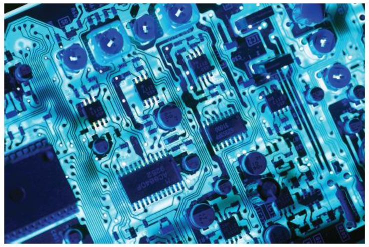
… and give us a means of reasoning about the relationships between objects



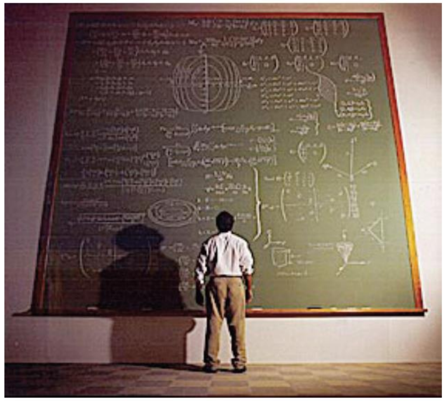
# Some Notations We'll Learn

| | | | | | |
|---|---|---|---|---|---|
| $\neg p$ | $p \wedge q$ | $p \oplus q$ | $p \rightarrow q$ | $p \Leftrightarrow q$ | $\forall x\, P(x)$ |
| $\exists x\, P(x)$ | $\{a_1, \cdots, a_n\}$ | $\boldsymbol{Z}, \boldsymbol{N}, \boldsymbol{R}$ | $\therefore$ | $\{x \mid P(x)\}$ | $x \notin S$ |
| $\varnothing$ | $S \subseteq T$ | $\lvert S \rvert$ | $A \cup B$ | $\overline{A}$ | $\bigcap_{i=1}^{n} A_i$ |
| $f : A \rightarrow B$ | $f^{-1}(x)$ | $f \circ g$ | $\lfloor x \rfloor$ | $\sum_{\alpha \in S} a_\alpha$ | $\prod_{i=1}^{n} a_i$ |
| $O, \Omega, \Theta$ | min, max | $a \nmid b$ | gcd, lcm | mod | $a \equiv b \pmod{m}$ |
| $(a_k \cdots a_0)_b$ | $[a_{ij}]$ | $\boldsymbol{A}^{\mathrm{T}}$ | $\boldsymbol{A} \odot \boldsymbol{B}$ | $\boldsymbol{A}^{[n]}$ | $\binom{n}{r}$ |
| $C(n; n_1, \cdots, n_m)$ | $p(E \mid F)$ | $R^*$ | $\Delta$ | $[a]_R$ | $\deg^+(v)$ |

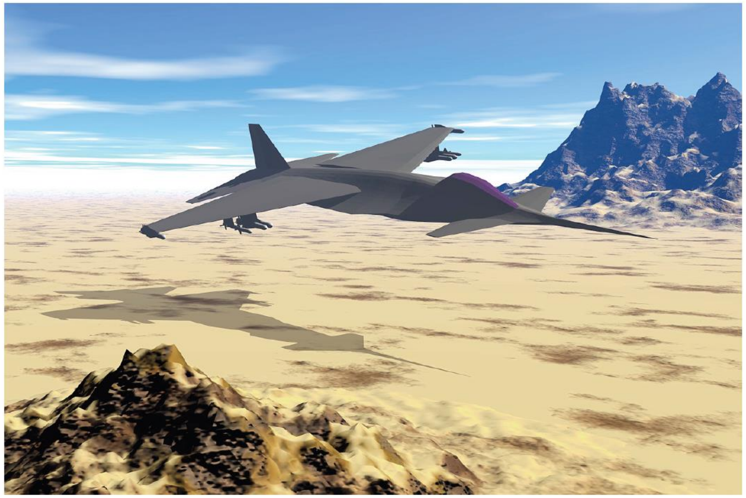
# Functions



Hardware design



Theory of computation



Computer graphics

# Integers and Modular Arithmetic

$$
\begin{array}{r}
0111\ 0101\ 0110\ 1011 \\
+\ \ 0101\ 1001\ 1110\ 0001 \\
\hline
1100\ 1111\ 0100\ 1100
\end{array}
$$

Binary arithmetic and bitwise operations

ATTACK AT DAWN

↓

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

C = P+ 6 (mod 26)

↓

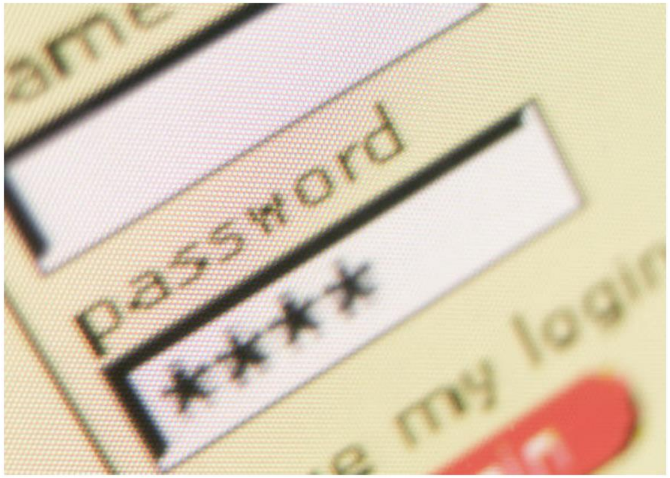
06 25 25 06 09 16 06 26 10 06 03 20

↓

FYYFIPFZJFCU

Cryptography

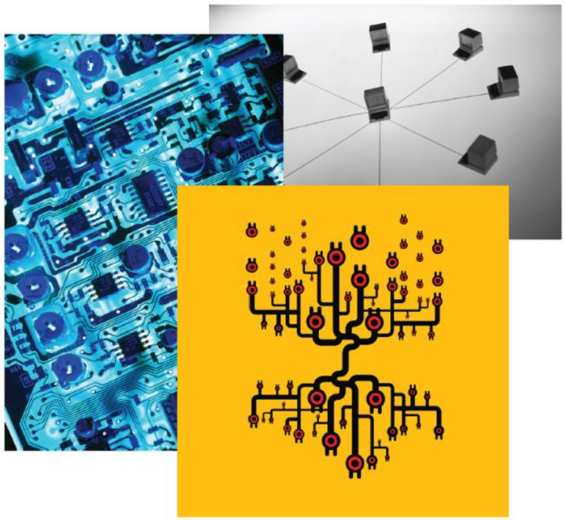
# Counting




How many valid passwords exist for a given set of rules?

How many IP addresses can be assigned within a network segment? Will we run out?

# Probability and Expectation



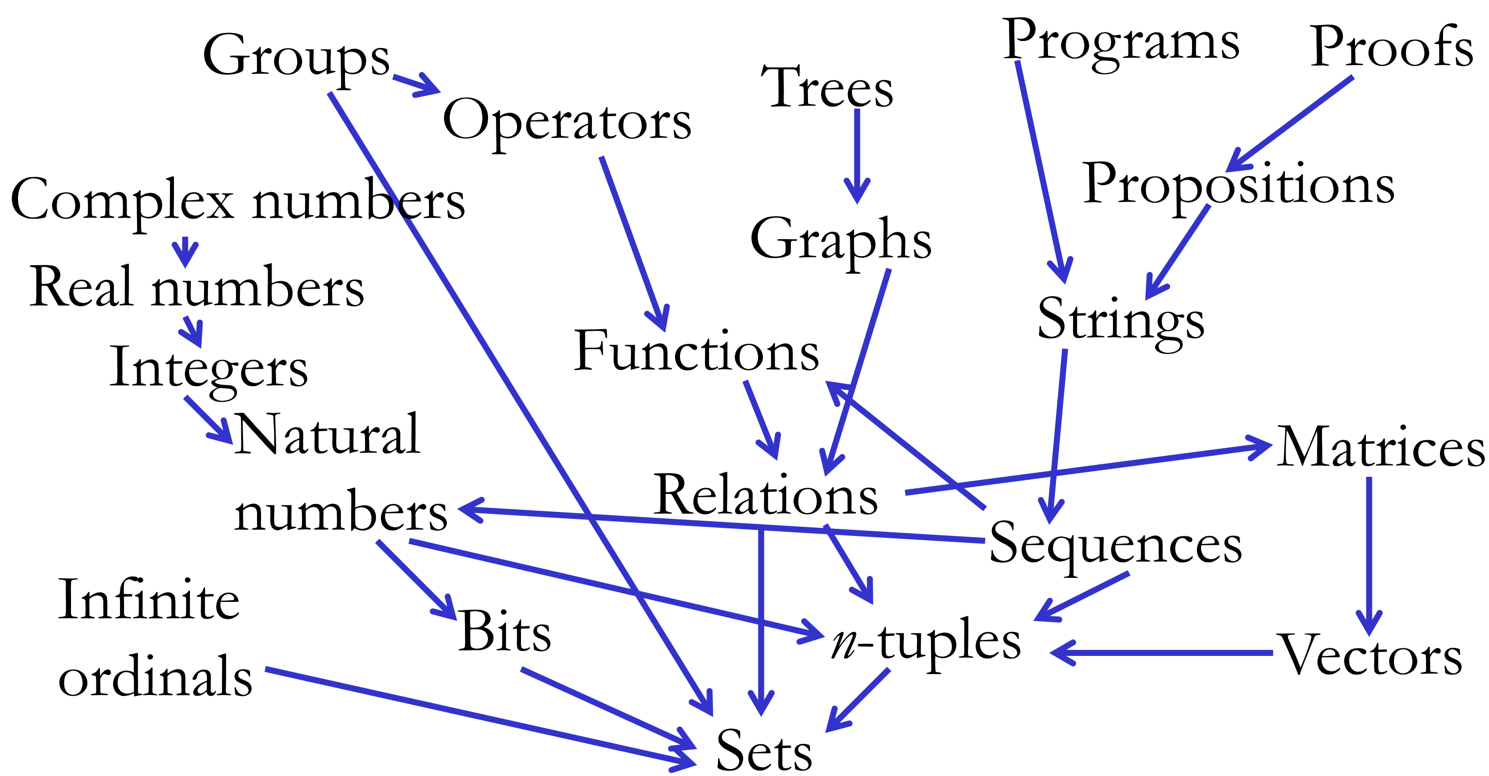
Hardware, software, and network simulation

Spam classification

Risk assessment

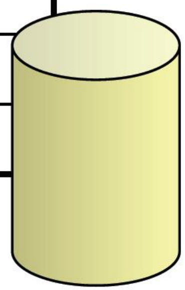
# Relationships Between Structures

- "→" :≡ "Can be defined in terms of"

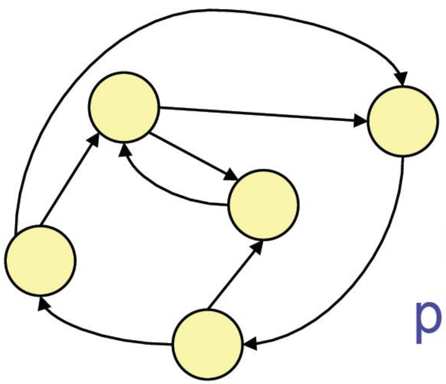


# Relations

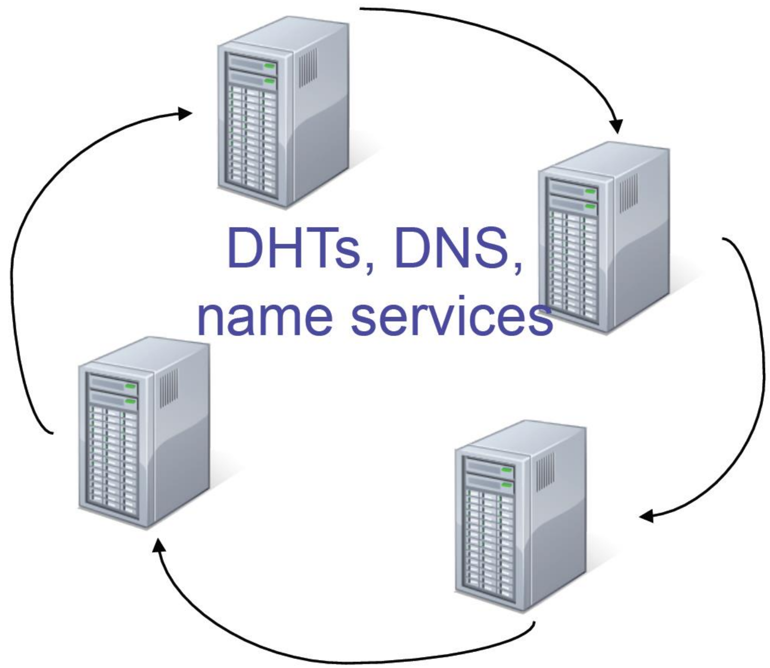
| Name | Age | Phone |
|---|---|---|
| Alice | 19 | 555-1234 |
| Danielle | 33 | 555-5353 |
| Zach | 27 | 555-3217 |
| Charlie | 21 | 555-2335 |



Relational databases



Route planning



# Syllabus Redux

- Logic and proofs
- Sets
- Functions
- Integers and modular arithmetic
- Counting
- Probability and expectation
- Relations

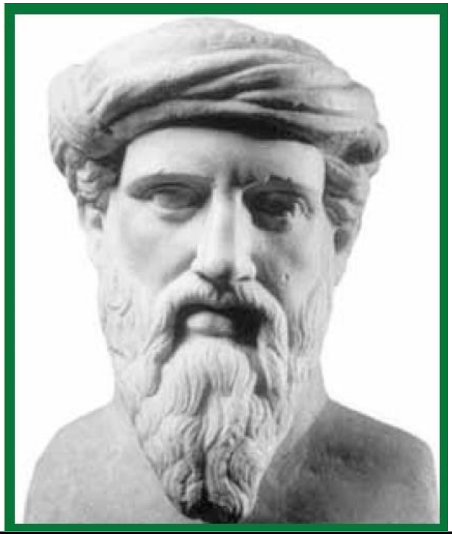
**Are these topics really useful? Yes!**

# Mastering discrete mathematics requires practice!

- Succeeding in this class requires practicing the skills that we will acquire, thinking critically, and asking questions

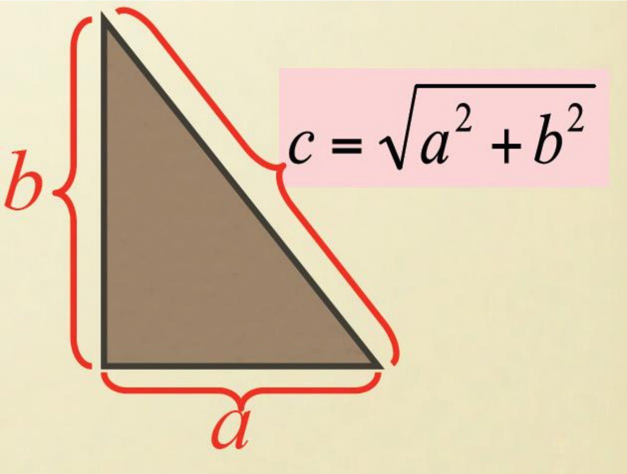
- Keys to success:
  – Attend class and take notes
  – Do your homework
  – Work extra problems when you're unsure
  – Prepare for quizzes and exams
  – Take advantage of office hours

# A Proof Example



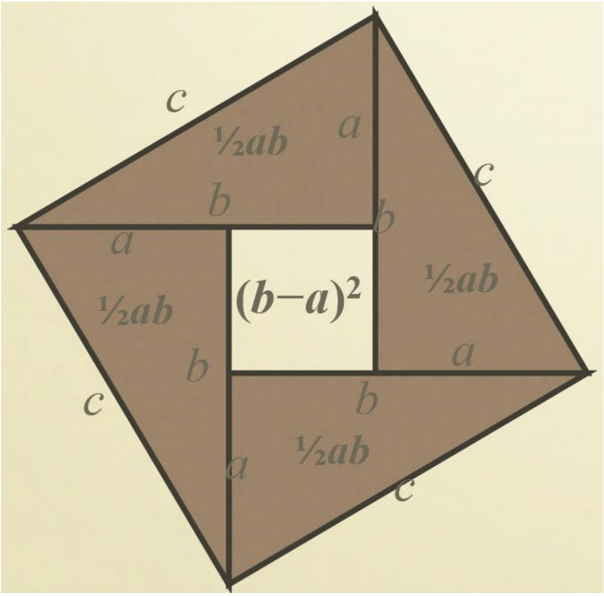
Pythagoras of Samos
(ca. 569-475 B.C.)

**Theorem:** *(Pythagorean Theorem of Euclidean geometry)* For any real numbers $a$, $b$, and $c$, if $a$ and $b$ are the base-length and height of a right triangle, and $c$ is the length of its hypotenuse, then $a^2 + b^2 = c^2$.



# Proof of Pythagorean Theorem

- **Proof.** Consider the below diagram:
  - Exterior square area = $c^2$, the sum of the following regions:
  - The area of the 4 triangles = $4(1/2\ ab) = 2ab$
  - The area of the small interior square = $(b-a)^2 = b^2-2ab+a^2$.
  - Thus, $c^2 = 2ab + (b^2-2ab+a^2) = a^2 + b^2$. ■




**Note:** It is easy to show that the exterior and interior quadrilaterals in this construction are indeed squares, and that the side length of the internal square is indeed $b-a$ (where $b$ is defined as the length of the longer of the two perpendicular sides of the triangle). These steps would also need to be included in a more complete proof.

# Final thoughts

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

- Next lecture: Propositional logic