Discrete Structures for Computer Science

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

Lecture #1: Course Introduction

Based on materials developed by Dr. Adam Lee
CS 441: Discrete Structures for Computer Science

**Instructor:**
William Garrison  
bill@cs.pitt.edu  
6311 Sennott Square  
OH: TBA

**TA:**
See course website

http://cs.pitt.edu/~bill/441
Course meeting times

- Lecture
  - SENSQ 5502, T/H 12:30-2:15

- Recitation
  - Tuesday: 2:30-3:20, SENSQ 5313
  - Thursday: 11:30-12:20, SENSQ 5313

- It is important to attend both lecture and your assigned recitation section!
  - (No recitations this week)
Grading

- Overall breakdown:
  - 35% Midterm exam
  - 35% Cumulative final exam
  - 30% Homework
  - 100%
Homework

Weekly homework assignments

- Assigned in class, due one week later at the start of lecture
- Late homework is not accepted—don’t be late to class!
- Two lowest homework grades will be dropped

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
Check the web page 2-3 times per week. Announcements, homework, and lecture slides will be posted there.
- Lecture slides are intentionally incomplete—take notes!!

We will drop your two lowest homework scores before computing your homework average—no excuses necessary!

If necessary, we will allow regrade requests. However, we reserve the right to regrade the entire assignment, not just the portion in question.

Other policies are on the web page
- Accommodating students with disabilities
- Religious observances
- 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 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.
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?

```
0 1 1 1 0 1 0 1 1 ...
```

![Diagram](image.png)
Why study discrete math?

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

```python
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
- Integers and modular arithmetic
- Counting
- Probability and expectation
- Relations

Are these topics really useful?
grant(X, projector) :- role(X, presenter), located(X, 104)
located(adam, 104)
role(adam, presenter)

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

Automated reasoning, AI, security

Verifying data structures and hardware

Algorithm and protocol analysis

Parsing expressions

Logic and proofs

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;

exp()
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
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 (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
# Relations

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>19</td>
<td>555-1234</td>
</tr>
<tr>
<td>Danielle</td>
<td>33</td>
<td>555-5353</td>
</tr>
<tr>
<td>Zach</td>
<td>27</td>
<td>555-3217</td>
</tr>
<tr>
<td>Charlie</td>
<td>21</td>
<td>555-2335</td>
</tr>
</tbody>
</table>

Relational databases

DHTs, DNS, name services

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 ➢ Solutions to odd-numbered exercises provided in textbook
  - Go to your recitation every week
  - Take advantage of office hours
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

Right now: Quick assessment quiz

Next: Propositional logic