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, by appt.

**TA:**
TBA

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

- Lecture
  - B: Flex@Pitt, M/W 5:00-6:15
  - C: Flex@Pitt, T/H 4:30-5:45

- Recitations
  - Wednesday or Friday (see website)

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

- Overall breakdown:
  - 30% Midterm exam
  - 30% Final exam
  - 20% Homework
  - 10% Recitation exercises
  - 10% Lecture participation (Top Hat)
  - 100%
Recitation exercises

- Covers the previous week’s material (W-T)
- Assigned the day before recitation, due the day of your recitation at 11:59 PM via Gradescope
- Late submissions are not accepted—submit early and often
- Two lowest exercise grades will be dropped

Recitations are collaborative

- You should try the exercises in advance of recitation
- You’ll go over much of the solutions in recitations
- Submit solutions that you understand—don’t just copy what your TA writes or copy a friend’s submission
Weekly homework assignments

- Assigned on Friday, due the next Tuesday at **11:59 PM**
- Late homework is **not accepted**—submit early and often
- Two lowest homework grades will be dropped

Homework may be **discussed** with others, but must be solved and written up **individually**

- Limit discussion to understanding problems and developing high-level solution tactics
  - Canvas is a good place to do this
- Identify collaborators on your homework cover sheet
- No online resources (except Canvas)
- Failure to comply with this policy is a violation of academic integrity (F in the course)
Check the web page and Canvas frequently. Announcements, homework, and lecture slides will be posted there.

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

Regrade requests accepted within 2 weeks of hand-back. Grade may increase, decrease, or stay the same.

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.
Why study discrete math?

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

Analog (continuous) input → Sampling and discretization → Digital (discrete) output
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 | ... |

Read
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
- Integers and modular arithmetic
- Counting
- Probability and expectation
- Relations

Are these topics really useful?
Logic and proofs

Algorithm and protocol analysis

Automated reasoning, AI, security

Verifying data structures and hardware

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

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

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() + 3.1415
* 4
3 2

Parsing expressions
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

Binary arithmetic and bitwise operations

AT\text{ACK}\text{ AT} D\text{AWN}

\begin{align*}
C &= P + 6 \pmod{26} \\
012020010311012004012314 \\
062525060916062610060320 \\
FYYFIPFZJFCU \\
\text{Cryptography}
\end{align*}
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
- Risk assessment
- Spam classification
## 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

- Next: Propositional logic