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 Zoom  
OH: TBA, by appt. for this week

**TAs:**
See syllabus site

https://bill-computer.science/441
Course meeting times

- **Lecture**
  - B: CHVRN 154, T/H 11:00-12:15
  - C: SENSQ 5502, T/H 2:30-3:45

- **Recitations**
  - Thursday or Friday (check Peoplesoft)

- It is important to attend *both* lecture and your assigned recitation section!
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

- For now, the “board” is a tablet 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!
- TH questions allow me to gauge what everyone is understanding in the moment
- 15% of points are dropped

After lecture, reflect on the topic using CourseMirror

- This is an optional research system that asks quick questions
- For up to 2 extra participation points, complete the consent form and submit reflections for each lecture (within 8 hours)
Recitation

Recitation exercises

- Covers the previous week’s material
- Assigned mid-week, due the day of your recitation at 11:59 PM
- Submit your own work; *primarily* graded for completion
- Late submissions are *not accepted*—submit early and often
- Two recitation exercise grades will be dropped

Recitations are collaborative

- You should attempt the exercises in advance of recitation
- 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
Weekly homework assignments

- Assigned on Friday, due the next Wednesday at 11:59 PM
- Late homework is not accepted—submit early and often
- Two homework grades will be dropped
- Optional challenge problem(s) in each

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 so)
- 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 problems will be submitted for participation on TH
- Next, recitation is collected, mainly for completion
- Finally, homework is graded carefully

<table>
<thead>
<tr>
<th></th>
<th>Week 1 material</th>
<th>Week 2 material</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecture and TH</td>
<td>8/31 and 9/02</td>
<td>9/07 and 9/09</td>
</tr>
<tr>
<td>Recitation</td>
<td>9/02 or 9/03</td>
<td>9/09 or 9/10</td>
</tr>
<tr>
<td>Homework</td>
<td>9/03 through 9/08</td>
<td>9/10 through 9/15</td>
</tr>
</tbody>
</table>
Review Sessions

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

- The first is planned for Lecture 6, 9/16

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...
Check the syllabus site 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 documentation 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, where we’ll go next
- 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
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 ...
```
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);
end function
```

```
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;
end function
```
Tentative Syllabus

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

Are these topics really useful?
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;
end function

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

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

Automated reasoning, AI, security

Parsing expressions

Verifying data structures and hardware

Algorithm and protocol analysis

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

Binary arithmetic and bitwise operations

\[
\begin{array}{cccccccc}
0111 & 0101 & 0110 & 1011 \\
+ & 0101 & 1001 & 1110 & 0001 \\
\hline
1100 & 1111 & 0100 & 1100 \\
\end{array}
\]

ATTACK AT DAWN

\[
C = P + 6 \pmod{26}
\]

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
  - 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
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
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
5. Watch for a Gradescope invitation, where you’ll find recitation and homework
6. (Optional) Sign up for CourseMirror and complete the pre-survey and consent form