Week 10 at a glance

We will be learning and practicing to:

- Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.
  - Defining functions, predicates, and binary relations using multiple representations
  - Determining whether a given binary relation is symmetric, antisymmetric, reflexive, and/or transitive
  - Determining whether a given binary relation is an equivalence relation and/or a partial order
  - Drawing graph representations of relations and functions e.g. Hasse diagram and partition diagram
- Know, select and apply appropriate computing knowledge and problem-solving techniques. Reason about computation and systems. Use mathematical techniques to solve problems. Determine appropriate conceptual tools to apply to new situations. Know when tools do not apply and try different approaches. Critically analyze and evaluate candidate solutions.
  - Using the definitions of the div and mod operators on integers
  - Using divisibility and primality predicates
  - Applying the definition of congruence modulo n and modular arithmetic
- Apply proof strategies, including direct proofs and proofs by contradiction, and determine whether a proposed argument is valid or not.
  - Using proofs as knowledge discovery tools to decide whether a statement is true or false

TODO:

Review quiz based on Week 9 class material (due Monday 03/09/2026)

Homework 5 due on Gradescope https://www.gradescope.com/ (Thursday 03/12/2026)

Second attempt tests for Test 1 and/or Test 2 in the CBTF at your scheduled time.

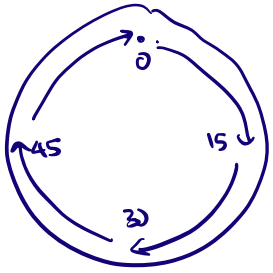
Review for Final exam. The final exam is in person and synchronous Friday 03/20/26 11:30A - 2:29P in the Rec Gym.

Week 10 Monday: Equivalence Relations and Partial Orders

Application: Cycling

How many minutes past the hour are we at? *Model with* $+15$ **mod** 60

Time:	12:00pm	12:15pm	12:30pm	12:45pm	1:00pm	1:15pm	1:30pm	1:45pm	2:00pm
"Minutes past":	0	15	30	45	0	15	30	45	0



Encryption

Replace each English letter by a letter that's fifteen ahead of it in the alphabet *Model with* $+15$ **mod** 26

Original index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Original letter:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Shifted letter:	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Shifted index:	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

To send WINTER, encrypt 22-08-13-19-04-17

as 11-23-02-08-19-06 and then send.

Decryption

$-15 \bmod 26$.

"The Code Book"

-Singh

Application: Cryptography

Definition: Let a be a positive integer and p be a large[1] prime number, both known to everyone. Let k_1 be a secret large number known only to person P_1 (Alice) and k_2 be a secret large number known only to person P_2 (Bob). Let the **Diffie-Helman shared key** for a, p, k_1, k_2 be $(a^{k_1 \cdot k_2} \textbf{ mod } p)$.

Idea: P_1 can quickly compute the Diffie-Helman shared key knowing only a, p, k_1 and the result of a^{k_2} **mod** p (that is, P_1 can compute the shared key without knowing k_2, only a^{k_2} **mod** p). Similarly, P_2 can quickly compute the Diffie-Helman shared key knowing only a, p, k_2 and the result of a^{k_1} **mod** p (that is, P_2 can compute the shared key without knowing k_1, only a^{k_1} **mod** p). But, any person P_3 who knows neither k_1 nor k_2 (but may know any and all of the other values) cannot compute the shared secret efficiently.

Key property for *shared* secret:

$$\forall a \in \mathbb{Z} \forall b \in \mathbb{Z} \forall g \in \mathbb{Z}^+ \forall n \in \mathbb{Z}^+ ((g^a \textbf{ mod } n)^b, (g^b \textbf{ mod } n)^a) \in R_{(\textbf{mod } n)}$$

Key property for shared *secret*:

There are efficient algorithms to calculate the result of modular exponentiation but there are no (known) efficient algorithms to calculate discrete logarithm.

P_1 publishes $a^{k_1} \bmod p$

P_2 publishes $a^{k_2} \bmod p$

P_1 calculates $(a^{k_2} \bmod p)^{k_1} \bmod p$

P_2 calculates $(a^{k_1} \bmod p)^{k_2} \bmod p$.

[1] We leave the definition of "large" vague here, but think hundreds of digits for practical applications. In practice, we also need a particular relationship between a and p to hold, which we leave out here.

Week 10 Wednesday: Applications

Last time, we saw that partitions associated to equivalence relations were useful in the context of modular arithmetic. Today we'll look at a different application.

Recall that in a movie recommendation system, each user's ratings of movies is represented as a n-tuple (with the positive integer n being the number of movies in the database), and each component of the n-tuple is an element of the collection $\{-1, 0, 1\}$.

We call Rt_5 the set of all ratings 5-tuples.

Define $d : Rt_5 \times Rt_5 \to \mathbb{N}$ by

$$d(\ (\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ )\ ) = \sum_{i=1}^{5} |x_i - y_i|$$

Consider the following binary relations on Rt_5.

$$E_{proj} = \{(\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ ) \in Rt_5 \times Rt_5 \ \mid\ (x_1 = y_1) \wedge (x_2 = y_2) \wedge (x_3 = y_3)\}$$

Example ordered pair in E_{proj}:

Reflexive? Symmetric? Transitive? Antisymmetric?

$$E_{dist} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid\ d(\ (u, v)\ ) \leq 2\}$$

Example ordered pair in E_{dist}:

Reflexive? Symmetric? Transitive? Antisymmetric?

$$E_{circ} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid \ d(\ (\ (0,0,0,0,0)\ , u)\ ) = d(\ (\ (0,0,0,0,0), v\ )\ )\}$$

Example ordered pair in E_{circ}:

Reflexive? Symmetric? Transitive? Antisymmetric?

The partition of Rt_5 defined by ____ is

{ { $(-1,-1,-1,-1,-1), (-1,-1,-1,-1,0), (-1,-1,-1,-1,1), (-1,-1,-1,0,-1), (-1,-1,-1,0,0), (-1,-1,-1,0,1), (-1,-1,-1,1,-1), (-1,-1,-1,1,0), (-1,-1,-1,1,1)$ },
{ $(-1,-1,0,-1,-1), (-1,-1,0,-1,0), (-1,-1,0,-1,1), (-1,-1,0,0,-1), (-1,-1,0,0,0), (-1,-1,0,0,1), (-1,-1,0,1,-1), (-1,-1,0,1,0), (-1,-1,0,1,1)$ },
{ $(-1,-1,1,-1,-1), (-1,-1,1,-1,0), (-1,-1,1,-1,1), (-1,-1,1,0,-1), (-1,-1,1,0,0), (-1,-1,1,0,1), (-1,-1,1,1,-1), (-1,-1,1,1,0), (-1,-1,1,1,1)$ },
{ $(-1,0,-1,-1,-1), (-1,0,-1,-1,0), (-1,0,-1,-1,1), (-1,0,-1,0,-1), (-1,0,-1,0,0), (-1,0,-1,0,1), (-1,0,-1,1,-1), (-1,0,-1,1,0), (-1,0,-1,1,1)$ },
{ $(-1,0,0,-1,-1), (-1,0,0,-1,0), (-1,0,0,-1,1), (-1,0,0,0,-1), (-1,0,0,0,0), (-1,0,0,0,1), (-1,0,0,1,-1), (-1,0,0,1,0), (-1,0,0,1,1)$ },
{ $(-1,0,1,-1,-1), (-1,0,1,-1,0), (-1,0,1,-1,1), (-1,0,1,0,-1), (-1,0,1,0,0), (-1,0,1,0,1), (-1,0,1,1,-1), (-1,0,1,1,0), (-1,0,1,1,1)$ },
{ $(-1,1,-1,-1,-1), (-1,1,-1,-1,0), (-1,1,-1,-1,1), (-1,1,-1,0,-1), (-1,1,-1,0,0), (-1,1,-1,0,1), (-1,1,-1,1,-1), (-1,1,-1,1,0), (-1,1,-1,1,1)$ },
{ $(-1,1,0,-1,-1), (-1,1,0,-1,0), (-1,1,0,-1,1), (-1,1,0,0,-1), (-1,1,0,0,0), (-1,1,0,0,1), (-1,1,0,1,-1), (-1,1,0,1,0), (-1,1,0,1,-1)$ },
{ $(-1,1,1,-1,-1), (-1,1,1,-1,0), (-1,1,1,-1,1), (-1,1,1,0,-1), (-1,1,1,0,0), (-1,1,1,0,1), (-1,1,1,1,-1), (-1,1,1,1,0), (-1,1,1,1,1)$ },
{ $(0,-1,-1,-1,-1), (0,-1,-1,-1,0), (0,-1,-1,-1,1), (0,-1,-1,0,-1), (0,-1,-1,0,0), (0,-1,-1,0,1), (0,-1,-1,1,-1), (0,-1,-1,1,0), (0,-1,-1,1,1)$ },
{ $(0,-1,0,-1,-1), (0,-1,0,-1,0), (0,-1,0,-1,1), (0,-1,0,0,-1), (0,-1,0,0,0), (0,-1,0,0,1), (0,-1,0,1,-1), (0,-1,0,1,0), (0,-1,0,1,1)$ },
{ $(0,-1,1,-1,-1), (0,-1,1,-1,0), (0,-1,1,-1,1), (0,-1,1,0,-1), (0,-1,1,0,0), (0,-1,1,0,1), (0,-1,1,1,-1), (0,-1,1,1,0), (0,-1,1,1,1)$ },
{ $(0,0,-1,-1,-1), (0,0,-1,-1,0), (0,0,-1,-1,1), (0,0,-1,0,-1), (0,0,-1,0,0), (0,0,-1,0,1), (0,0,-1,1,-1), (0,0,-1,1,0), (0,0,-1,1,1)$ },
{ $(0,0,0,-1,-1), (0,0,0,-1,0), (0,0,0,-1,1), (0,0,0,0,-1), (0,0,0,0,0), (0,0,0,0,1), (0,0,0,1,-1), (0,0,0,1,0), (0,0,0,1,1)$ },
{ $(0,0,1,-1,-1), (0,0,1,-1,0), (0,0,1,-1,1), (0,0,1,0,-1), (0,0,1,0,0), (0,0,1,0,1), (0,0,1,1,-1), (0,0,1,1,0), (0,0,1,1,1)$ },
{ $(0,1,-1,-1,-1), (0,1,-1,-1,0), (0,1,-1,-1,1), (0,1,-1,0,-1), (0,1,-1,0,0), (0,1,-1,0,1), (0,1,-1,1,-1), (0,1,-1,1,0), (0,1,-1,1,1)$ },
{ $(0,1,0,-1,-1), (0,1,0,-1,0), (0,1,0,-1,1), (0,1,0,0,-1), (0,1,0,0,0), (0,1,0,0,1), (0,1,0,1,-1), (0,1,0,1,0), (0,1,0,1,-1)$ },
{ $(0,1,1,-1,-1), (0,1,1,-1,0), (0,1,1,-1,1), (0,1,1,0,-1), (0,1,1,0,0), (0,1,1,0,1), (0,1,1,1,-1), (0,1,1,1,0), (0,1,1,1,1)$ },
{ $(1,-1,-1,-1,-1), (1,-1,-1,-1,0), (1,-1,-1,-1,1), (1,-1,-1,0,-1), (1,-1,-1,0,0), (1,-1,-1,0,1), (1,-1,-1,1,-1), (1,-1,-1,1,0), (1,-1,-1,1,1)$ },
{ $(1,-1,0,-1,-1), (1,-1,0,-1,0), (1,-1,0,-1,1), (1,-1,0,0,-1), (1,-1,0,0,0), (1,-1,0,0,1), (1,-1,0,1,-1), (1,-1,0,1,0), (1,-1,0,1,1)$ },
{ $(1,-1,1,-1,-1), (1,-1,1,-1,0), (1,-1,1,-1,1), (1,-1,1,0,-1), (1,-1,1,0,0), (1,-1,1,0,1), (1,-1,1,1,-1), (1,-1,1,1,0), (1,-1,1,1,1)$ },
{ $(1,0,-1,-1,-1), (1,0,-1,-1,0), (1,0,-1,-1,1), (1,0,-1,0,-1), (1,0,-1,0,0), (1,0,-1,0,1), (1,0,-1,1,-1), (1,0,-1,1,0), (1,0,-1,1,1)$ },
{ $(1,0,0,-1,-1), (1,0,0,-1,0), (1,0,0,-1,1), (1,0,0,0,-1), (1,0,0,0,0), (1,0,0,0,1), (1,0,0,1,-1), (1,0,0,1,0), (1,0,0,1,1)$ },
{ $(1,0,1,-1,-1), (1,0,1,-1,0), (1,0,1,-1,1), (1,0,1,0,-1), (1,0,1,0,0), (1,0,1,0,1), (1,0,1,1,-1), (1,0,1,1,0), (1,0,1,1,1)$ },
{ $(1,1,-1,-1,-1), (1,1,-1,-1,0), (1,1,-1,-1,1), (1,1,-1,0,-1), (1,1,-1,0,0), (1,1,-1,0,1), (1,1,-1,1,-1), (1,1,-1,1,0), (1,1,-1,1,1)$ },
{ $(1,1,0,-1,-1), (1,1,0,-1,0), (1,1,0,-1,1), (1,1,0,0,-1), (1,1,0,0,0), (1,1,0,0,1), (1,1,0,1,-1), (1,1,0,1,0), (1,1,0,1,-1)$ },
{ $(1,1,1,-1,-1), (1,1,1,-1,0), (1,1,1,-1,1), (1,1,1,0,-1), (1,1,1,0,0), (1,1,1,0,1), (1,1,1,1,-1), (1,1,1,1,0), (1,1,1,1,1)$ }
}

The partition of Rt_5 defined by $E =$ ____ is

$$\{$$
$$[\ (0,0,0,0,0)\ ]_E$$
$$,[\ (0,0,0,0,1)\ ]_E$$
$$,[\ (0,0,0,1,1)\ ]_E$$
$$,[\ (0,0,1,1,1)\ ]_E$$
$$,[\ (0,1,1,1,1)\ ]_E$$
$$,[\ (1,1,1,1,1)\ ]_E$$
$$\}$$

How many elements are in each part of the partition?

Scenario: Good morning! You're a user experience engineer at Netflix. A product goal is to design customized home pages for groups of users who have similar interests. Your manager tasks you with designing an algorithm for producing a clustering of users based on their movie interests, so that customized homepages can be engineered for each group.

Your idea: equivalence relations!

$$E_{id} = \{(\ (x_1, x_2, x_3, x_4, x_5), (x_1, x_2, x_3, x_4, x_5)\ ) \mid (x_1, x_2, x_3, x_4, x_5) \in Rt_5\}$$

Describe how each homepage should be designed . . .

$$E_{proj} = \{(\ (x_1, x_2, x_3, x_4, x_5), (y_1, y_2, y_3, y_4, y_5)\ ) \in Rt_5 \times Rt_5 \ \mid \ (x_1 = y_1) \wedge (x_2 = y_2) \wedge (x_3 = y_3)\}$$

Describe how each homepage should be designed . . .

$$E_{circ} = \{(u, v) \in Rt_5 \times Rt_5 \ \mid \ d(\ (\ (0, 0, 0, 0, 0)\ , u)\ ) = d(\ (\ (0, 0, 0, 0, 0), v\ )\ )\}$$

Describe how each homepage should be designed . . .

Week 10 Friday: Review and Advice

Convert $(2A)_{16}$ to

- binary (base _____)
- decimal (base _____)
- octal (base _____)
- ternary (base _____)

The bases of RNA strands are elements of the set $B = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$. The set of RNA strands S is defined (recursively) by:

Basis Step:	$\texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

Each of the sets below is described using set builder notation. Rewrite them using the roster method.

- $\{s \in S \mid$ the leftmost base in s is the same as the rightmost base in s and s has length $3\}$

- $\{s \in S \mid$ there are twice as many $\texttt{A}$s as $\texttt{C}$s in s and s has length $1\}$

Certain sequences of bases serve important biological functions in translating RNA to proteins. The following recursive definition gives a special set of RNA strands: The set of RNA strands $\hat{S}$ is defined (recursively) by

Basis step:	$\texttt{AUG} \in \hat{S}$
Recursive step:	If $s \in \hat{S}$ and $x \in R$, then $sx \in \hat{S}$

where $R = \{\texttt{UUU}, \texttt{CUC}, \texttt{AUC}, \texttt{AUG}, \texttt{GUU}, \texttt{CCU}, \texttt{GCU}, \texttt{UGG}, \texttt{GGA}\}$.

Each of the sets below is described using set builder notation. Rewrite them using the roster method.

- $\{s \in \hat{S} \mid s$ has length less than or equal to $5\}$

- $\{s \in S \mid$ there are twice as many $\texttt{C}$s as $\texttt{A}$s in s and s has length $6\}$

Let $W = \mathcal{P}(\{1, 2, 3, 4, 5\})$. Consider the statement

$$\forall A \in W \ \forall B \in W \ \forall C \in W \ ((A \cap B = A \cap C) \to (B = C))$$

Translate the statement to English. Negate the statement and translate this negation to English. Decide whether the original statement or its negation is true and justify your decision.

The set of linked lists of natural numbers L is defined by

Basis step:	$[] \in L$
Recursive step:	If $l \in L$ and $n \in \mathbb{N}$, then $(n, l) \in L$

The function $length : L \to \mathbb{N}$ that computes the length of a list is

Basis step:	$length([]) = 0$
Recursive step:	If $l \in L$ and $n \in \mathbb{N}$, then $length((n, l)) = 1 + length(l)$

Prove or disprove: the function *length* is onto.

Prove or disprove: the function *length* is one-to-one.

Tips for future classes from the CSE 20 TAs and tutors

- Final exams
  - Don't forget your university card during exams. Physical version is best for ID checks.
  - Look up seating assignments for exams and go early to make sure you're in the right place (check the exits to make sure you're reading it the right way)
  - Know where your exam is being held (find it on a map at least a day beforehand). Finals are often in strange places that take a while to find
- In class
  - Go to class
  - Take notes - it's much faster and more effective to note-take in class than watch recordings after, particularly if you do so longhand
  - Resist the urge to sit in the back. You will be able to focus much better sitting near the front, where there are fewer screens in front of you to distract from the lecture content
  - If you bring your laptop to class to take notes / access class materials, sit towards the back of the room to minimize distractions for people sitting behind you!
  - Don't be afraid to talk to the people next to you during group discussions. Odds are they're as nervous as you are, and you can all benefit from sharing your thoughts and understanding of the material
  - Certain classes will podcast the lectures, just like Zoom archives lecture recordings, at podcast.ucsd.edu . If they aren't podcasted, and you want to record lectures, ask your professor for consent first
- Office hours, tutor hours, and the CSE building
  - Office hours are a good place to hang out and get work done while being able to ask questions as they come up
  - Get to know the CSE building: CSE B260, basement labs, office hours rooms
- Libraries and on-campus resources
  - Look up what library floors are for what, how to book rooms: East wing of Geisel is open 24/7 (they might ask to see an ID if you stay late), East Wing of Geisel has chess boards and jigsaw puzzles, study pods on the 8th floor, free computers/wifi
  - Know Biomed exists and is usually less crowded
  - Most libraries allow you to borrow whiteboards and markers (also laptops, tablets, microphones, and other cool stuff) for 24 hours
  - Take advantage of Dine with a prof / Coffee with a prof program. It's legit free food / coffee once per quarter.
  - When planning out your daily schedule, think about where classes are, how much time will they take, are their places to eat nearby and how you can schedule social time with friends to nearby areas
  - Take into account the distances between classes if they are back to back

- Recommendations for electives from this quarter's tutors and TAs
  - CSE 105 (if you enjoy theory)
  - CSE 140 (if you enjoyed logic/circuits)
  - Math 160A/B (if you enjoy proof foundations)
  - CSE 190 (Intro to Quantum Computing)
  - CSE 130 (if you want to think about programming differently)
  - CSE 190 (Working with Large Codebases) (software engineering focused)

