CS251 Discrete Mathematics

Weekly Tutorial Notes

Based on Epp, *Discrete Mathematics with Applications*

Abstract

These notes provide supplementary material for CS251 Discrete Mathematics. Each week covers key definitions, theorems, worked examples, and practice problems aligned with the Epp textbook. The emphasis is on developing proof techniques and problem-solving skills essential for computer science.

Contents

1 Chapter 0: Proof Refresher

This chapter reviews the proof techniques and logical foundations from the first quarter. Use it as a reference throughout the course.

Propositional Logic Review

Definition 1.1 (Logical connectives)**.** Let P and Q be propositions (statements that are either true or false).

Symbol	**Name**	**True when...**
$\neg P$	Negation (NOT)	P is false
$P \wedge Q$	Conjunction (AND)	Both P and Q are true
$P \vee Q$	Disjunction (OR)	At least one of P, Q is true
$P \to Q$	Implication (IF-THEN)	P is false, or Q is true
$P \leftrightarrow Q$	Biconditional (IFF)	P and Q have the same truth value

Theorem 1.1 (Key equivalences)**.** *The following are logically equivalent (have the same truth table):*

$$P \to Q \equiv \neg P \vee Q \equiv \neg Q \to \neg P \quad \textit{(contrapositive)}$$
$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q \quad \textit{(De Morgan)}$$
$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q \quad \textit{(De Morgan)}$$
$$P \to Q \not\equiv Q \to P \quad \textit{(converse is NOT equivalent!)}$$

Predicate Logic Review

Definition 1.2 (Quantifiers)**.** Let $P(x)$ be a predicate (a statement depending on variable x) over domain D.

Universal quantifier: $\forall x \in D.\, P(x)$ means "for all x in D, $P(x)$ holds."
Existential quantifier: $\exists x \in D.\, P(x)$ means "there exists some x in D such that $P(x)$ holds."

Theorem 1.2 (Negating quantifiers)**.**

$$\neg(\forall x.\, P(x)) \equiv \exists x.\, \neg P(x)$$
$$\neg(\exists x.\, P(x)) \equiv \forall x.\, \neg P(x)$$

To negate "all cats are black," say "there exists a cat that is not black."

Theorem 1.3 (Quantifier order matters)**.**

$$\forall x.\, \exists y.\, P(x,y) \quad \not\equiv \quad \exists y.\, \forall x.\, P(x,y)$$

Example. *Let $P(x,y)$ mean "$y > x$" over $\mathbb{R}$.*

- $\forall x.\, \exists y.\, (y > x)$*: For every number, there's a larger one.* ***True.***
- $\exists y.\, \forall x.\, (y > x)$*: There's a number larger than all others.* ***False.***

Proof Techniques

Proof Strategy

[Direct proof] To prove $P \to Q$: Assume P is true, then show Q follows.
Template:

Assume P. [Reasoning...] Therefore Q.

Example. Prove: If n is even, then n^2 is even.
Proof. Assume n is even. Then $n = 2k$ for some integer k. So $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$. Since $2k^2$ is an integer, n^2 is even. □

Proof Strategy

[Proof by contrapositive] To prove $P \to Q$: Prove $\neg Q \to \neg P$ instead (they're equivalent).
Template:

We prove the contrapositive. Assume $\neg Q$. [Reasoning...] Therefore $\neg P$.

Example. Prove: If n^2 is odd, then n is odd.
Proof. We prove the contrapositive: if n is even, then n^2 is even. Assume n is even. Then $n = 2k$, so $n^2 = 4k^2 = 2(2k^2)$, which is even. □

Proof Strategy

[Proof by contradiction] To prove P: Assume $\neg P$ and derive a contradiction.
Template:

Suppose for contradiction that $\neg P$. [Reasoning...] This contradicts [known fact]. Therefore P.

Example. Prove: $\sqrt{2}$ is irrational.
Proof. Suppose for contradiction that $\sqrt{2} = a/b$ where a, b are integers with no common factors. Then $2 = a^2/b^2$, so $a^2 = 2b^2$. Thus a^2 is even, so a is even; write $a = 2c$. Then $4c^2 = 2b^2$, so $b^2 = 2c^2$, meaning b is also even. But then a and b share factor 2, contradicting our assumption. □

Proof Strategy

[Proof by cases] To prove P: Partition into exhaustive cases and prove each.
Template:

Case 1: [Condition]. [Proof of P in this case.]
Case 2: [Condition]. [Proof of P in this case.]
These cases are exhaustive, so P holds.

Example. Prove: For any integer n, $n^2 + n$ is even.
Proof. Case 1: n is even. Then n^2 is even, so $n^2 + n$ is even + even = even.
Case 2: n is odd. Then n^2 is odd, so $n^2 + n$ is odd + odd = even.
Every integer is either even or odd, so $n^2 + n$ is always even. □

Mathematical Induction

Proof Strategy

[Weak induction] To prove $\forall n \geq n_0.\, P(n)$:

1. **Base case:** Prove $P(n_0)$.
2. **Inductive step:** Prove $P(k) \to P(k+1)$ for arbitrary $k \geq n_0$.

Why it works: Base case gives $P(n_0)$. Inductive step gives $P(n_0) \to P(n_0+1)$, so $P(n_0+1)$. Inductive step again gives $P(n_0+1) \to P(n_0+2)$, so $P(n_0+2)$. And so on forever.

Example 1.1. Prove: $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ for all $n \geq 1$.

Proof. By induction on n.

Base case ($n = 1$): $\sum_{i=1}^{1} i = 1 = \frac{1\cdot 2}{2}$. ✓

Inductive step: Assume $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$ for some $k \geq 1$. Then:

$$\sum_{i=1}^{k+1} i = \left(\sum_{i=1}^{k} i\right) + (k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{k(k+1)+2(k+1)}{2} = \frac{(k+1)(k+2)}{2}$$

This is the formula with $n = k+1$. ✓

By induction, the formula holds for all $n \geq 1$. □

Proof Strategy

[Strong induction] To prove $\forall n \geq n_0.\, P(n)$:

1. **Base case(s):** Prove $P(n_0)$ (and possibly $P(n_0+1), \ldots$).
2. **Inductive step:** Prove $\big[P(n_0) \land P(n_0+1) \land \cdots \land P(k)\big] \to P(k+1)$.

Use strong induction when proving $P(k+1)$ requires cases smaller than k (not just k itself).

Example 1.2. Prove: Every integer $n \geq 2$ can be written as a product of primes.

Proof. By strong induction on n.

Base case ($n = 2$): 2 is prime, so it's a product of primes (just itself). ✓

Inductive step: Assume every integer from 2 to k can be written as a product of primes. Consider $k+1$.

Case 1: $k+1$ is prime. Then $k+1$ is a product of primes (itself). ✓

Case 2: $k+1$ is composite. Then $k+1 = ab$ where $2 \leq a, b \leq k$. By the inductive hypothesis, both a and b are products of primes. So $k+1 = ab$ is also a product of primes. ✓

By strong induction, every $n \geq 2$ is a product of primes. □

Proof Strategy

[Structural induction] To prove a property P holds for all elements of a recursively-defined set S:

1. **Base case(s):** Prove P for each base element of S.
2. **Inductive step(s):** For each recursive rule, assume P holds for the inputs and prove P for the output.

Example 1.3. Define binary trees recursively:

- **Base:** A single node $\bullet$ is a binary tree.
- **Recursive:** If T_1 and T_2 are binary trees, then the tree with a root connected to T_1 (left) and T_2 (right) is a binary tree.

Prove: Every binary tree has one more node than it has internal nodes (nodes with children).
Proof. Let $P(T)$ be: "T has one more leaf than internal node."
Base case: A single node $\bullet$ has 1 leaf and 0 internal nodes. $1 = 0 + 1$. ✓
Inductive step: Suppose T_1 has ℓ_1 leaves, i_1 internal nodes (with $\ell_1 = i_1 + 1$), and similarly T_2 has $\ell_2 = i_2 + 1$. Form tree T with root connected to T_1 and T_2.
In T: The root is a new internal node. All leaves of T_1 and T_2 are still leaves. So:

$$\begin{aligned} \text{leaves}(T) &= \ell_1 + \ell_2 \\ \text{internal}(T) &= i_1 + i_2 + 1 \quad \text{(the root)} \end{aligned}$$

Check: $\ell_1 + \ell_2 = (i_1 + 1) + (i_2 + 1) = (i_1 + i_2 + 1) + 1 = \text{internal}(T) + 1$. ✓
By structural induction, every binary tree has one more leaf than internal node. □

Common Inference Rules

Name	**Rule**	**Meaning**
Modus ponens	$P,\ P \to Q \ \vdash\ Q$	If P and $P \to Q$, conclude Q
Modus tollens	$\neg Q,\ P \to Q \ \vdash\ \neg P$	If $\neg Q$ and $P \to Q$, conclude $\neg P$
Hypothetical syllogism	$P \to Q,\ Q \to R \ \vdash\ P \to R$	Chain implications
Disjunctive syllogism	$P \lor Q,\ \neg P \ \vdash\ Q$	Eliminate one disjunct
Conjunction	$P,\ Q \ \vdash\ P \land Q$	Combine facts
Simplification	$P \land Q \ \vdash\ P$	Extract from conjunction
Addition	$P \ \vdash\ P \lor Q$	Weaken to disjunction
Resolution	$P \lor Q,\ \neg P \lor R \ \vdash\ Q \lor R$	Combine disjunctions

Existential and Universal Proofs

Proof Strategy

[Proving $\exists x.\, P(x)$] Exhibit a specific witness c and show $P(c)$ holds.
Example. Prove: There exists an integer n such that $n^2 = n$.
Proof. Take $n = 1$. Then $1^2 = 1$. □

Proof Strategy

[Proving $\forall x.\, P(x)$] Let x be an arbitrary element of the domain and prove $P(x)$.
Example. Prove: For all integers n, if n is odd, then n^2 is odd.
Proof. Let n be an arbitrary odd integer. Then $n = 2k+1$ for some integer k. So $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$, which is odd. □

Proof Strategy

[Disproving $\forall x.\, P(x)$] Find a counterexample: a specific c where $P(c)$ is false.
Example. Disprove: For all primes p, $2^p - 1$ is prime.
Counterexample. $p = 11$ is prime, but $2^{11} - 1 = 2047 = 23 \times 89$. □

Common Pitfalls

Common Mistake

Assuming what you're trying to prove. In a direct proof of $P \to Q$, you assume P, not Q. If you find yourself writing "Assume Q..." you've gone wrong.

Common Mistake

Confusing the converse. $P \to Q$ is NOT equivalent to $Q \to P$. Proving "if it's raining, the ground is wet" does not prove "if the ground is wet, it's raining."

Common Mistake

Induction: not using the hypothesis. In the inductive step, you must actually use the assumption $P(k)$ to prove $P(k+1)$. If your proof of $P(k+1)$ doesn't reference $P(k)$, either the proof is wrong or you didn't need induction.

Common Mistake

Induction: wrong base case. If your claim is $\forall n \geq 5.\, P(n)$, your base case must be $n = 5$, not $n = 1$.

Common Mistake

Proof by example. Checking $P(1), P(2), P(3)$ does not prove $\forall n.\, P(n)$. You need a general argument (or induction).

Common Mistake

Existential overgeneralization. From "there exists an x with property P," you cannot conclude that *every* x has property P.

Practice

1. Prove by induction: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.
2. Prove by induction: $n! > 2^n$ for all $n \geq 4$.
3. Prove by strong induction: Every amount of postage ≥ 12 cents can be made using 4-cent and 5-cent stamps.
4. Prove by contrapositive: If n^2 is divisible by 3, then n is divisible by 3.
5. Prove by contradiction: There are infinitely many primes.
6. Prove or disprove: For all integers a, b, c, if $a \mid bc$, then $a \mid b$ or $a \mid c$.
7. Prove by structural induction: For any arithmetic expression built from integers using $+$ and $\times$, the result is an integer. (Define the set of arithmetic expressions recursively first.)
8. Negate the following statement and determine which is true: "For every $\epsilon > 0$, there exists $\delta > 0$ such that for all x, if $|x| < \delta$ then $|f(x)| < \epsilon$."
9. Find the error in this "proof": *Claim: All horses are the same color.*

   "Proof" by induction: Base case: One horse is trivially the same color as itself. Inductive step: Assume any k horses are the same color. Given $k+1$ horses, remove one; the remaining k are the same color. Put it back and remove a different one; those k are the same color too. So all $k+1$ are the same color. $\square$
10. Prove: For all sets A and B, if $A \subseteq B$, then $\mathcal{P}(A) \subseteq \mathcal{P}(B)$.

2 Week 1: Set Theory

Reading

Epp §6.1–6.4.
Category theory companion: Week 1–2 (`category_theory_companion.pdf`).

Learning objectives

- Translate between roster and set-builder notation.
- Prove set identities with the element method.
- Apply standard set laws (commutative, associative, distributive).
- Use power sets and partitions correctly.
- Understand Cartesian products and their properties.

Key definitions and facts

Definition 2.1 (Subset and equality)**.** For sets A, B, $A \subseteq B$ means every element of A is in B. $A = B$ means $A \subseteq B$ and $B \subseteq A$.

Definition 2.2 (Set operations)**.** Let A and B be sets:

- **Union:** $A \cup B = \{x : x \in A \text{ or } x \in B\}$
- **Intersection:** $A \cap B = \{x : x \in A \text{ and } x \in B\}$
- **Difference:** $A \setminus B = \{x : x \in A \text{ and } x \notin B\}$
- **Complement:** $A^c = \{x \in U : x \notin A\}$ (relative to universal set U)

Definition 2.3 (Power set and partition)**.** The power set $\mathcal{P}(A)$ is the set of all subsets of A. If $|A| = n$, then $|\mathcal{P}(A)| = 2^n$. A partition of A is a collection of nonempty, pairwise disjoint subsets whose union is A.

Definition 2.4 (Cartesian product)**.** The Cartesian product $A \times B = \{(a, b) : a \in A, b \in B\}$. We have $|A \times B| = |A| \cdot |B|$ for finite sets.

Theorem 2.1 (Set laws)**.** *For all sets A, B, C:*

- ***Commutative:*** $A \cup B = B \cup A$, $A \cap B = B \cap A$
- ***Associative:*** $(A \cup B) \cup C = A \cup (B \cup C)$
- ***Distributive:*** $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- ***Absorption:*** $A \cup (A \cap B) = A$, $A \cap (A \cup B) = A$
- ***Identity:*** $A \cup \emptyset = A$, $A \cap U = A$

Theorem 2.2 (De Morgan's laws)**.** $(A \cup B)^c = A^c \cap B^c$ *and* $(A \cap B)^c = A^c \cup B^c$.

Remark (Boolean algebra). The set operations form a **Boolean algebra**: $\cup$ behaves like logical OR, $\cap$ behaves like AND, and complementation behaves like NOT. This correspondence allows you to translate between set identities and logical equivalences. For example, De Morgan's laws for sets correspond exactly to De Morgan's laws for logic: $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$.

Example 2.1 (Boolean-style simplification)**.** Simplify the set expression $(A \cap B) \cup (A \cap B^c)$.

Solution. Factor out A using distributivity:

$$(A \cap B) \cup (A \cap B^c) = A \cap (B \cup B^c) = A \cap U = A$$

The key insight is that $B \cup B^c = U$ (law of excluded middle for sets).

Warning (Russell's paradox). Not every description defines a valid set. Consider $R = \{x : x \notin x\}$—the "set of all sets that don't contain themselves." Is $R \in R$? If yes, then by definition $R \notin R$. If no, then $R \in R$. This contradiction shows that unrestricted set formation leads to paradoxes. Modern set theory (ZFC) avoids this by carefully axiomatizing which collections can be sets.

Worked examples

Example 2.2. Prove $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ (Distributive law).

Proof. We show mutual inclusion.

($\subseteq$) Let $x \in A \cap (B \cup C)$. Then $x \in A$ and $x \in B \cup C$. Since $x \in B \cup C$, either $x \in B$ or $x \in C$.

- If $x \in B$: then $x \in A \cap B$, so $x \in (A \cap B) \cup (A \cap C)$.
- If $x \in C$: then $x \in A \cap C$, so $x \in (A \cap B) \cup (A \cap C)$.

($\supseteq$) Let $x \in (A \cap B) \cup (A \cap C)$. Then $x \in A \cap B$ or $x \in A \cap C$. In either case, $x \in A$, and $x \in B$ or $x \in C$, so $x \in B \cup C$. Thus $x \in A \cap (B \cup C)$. □

Example 2.3. Prove De Morgan's law: $(A \cup B)^c = A^c \cap B^c$.

Proof. $x \in (A \cup B)^c$ iff $x \notin A \cup B$ iff not($x \in A$ or $x \in B$) iff ($x \notin A$ and $x \notin B$) iff $x \in A^c$ and $x \in B^c$ iff $x \in A^c \cap B^c$. □

Example 2.4. List the power set $\mathcal{P}(\{1, 2\})$ and verify that $|\mathcal{P}(\{1, 2\})| = 2^2$.

Solution. The subsets of $\{1, 2\}$ are:

$$\mathcal{P}(\{1, 2\}) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$$

We have $|\mathcal{P}(\{1, 2\})| = 4 = 2^2$. ✓

Example 2.5. Let $A = \{0, 1\}$ and $B = \{a, b, c\}$. Find $A \times B$ and $|A \times B|$.

Solution.

$$A \times B = \{(0, a), (0, b), (0, c), (1, a), (1, b), (1, c)\}$$

We have $|A \times B| = 6 = 2 \times 3 = |A| \cdot |B|$. ✓

Example 2.6. Disprove: $(A \cup B) \setminus C = A \cup (B \setminus C)$ for all sets A, B, C.

Solution. We need a counterexample where removing C from $A \cup B$ differs from keeping A intact and only removing C from B. The key is to have elements in $A \cap C$ that are not in B.

Let $A = \{1\}$, $B = \{2\}$, $C = \{1\}$:

- LHS: $A \cup B = \{1, 2\}$, so $(A \cup B) \setminus C = \{1, 2\} \setminus \{1\} = \{2\}$.

- RHS: $B \setminus C = \{2\} \setminus \{1\} = \{2\}$, so $A \cup (B \setminus C) = \{1\} \cup \{2\} = \{1, 2\}$.

Since $\{2\} \neq \{1, 2\}$, the identity fails. The issue is that the LHS removes elements of C from all of $A \cup B$, while the RHS preserves A completely. □

Example 2.7. Prove the absorption law: $A \cup (A \cap B) = A$.

Proof. ($\supseteq$) If $x \in A$, then $x \in A \cup (A \cap B)$ since x is in the first part of the union.
($\subseteq$) If $x \in A \cup (A \cap B)$, then $x \in A$ or $x \in A \cap B$. In either case, $x \in A$ (since $A \cap B \subseteq A$).
Therefore $A \cup (A \cap B) = A$. □

Going Deeper: Arrows and Diagrams

This begins a running thread through the course: learning to think in terms of *arrows* and *diagrams*. These tools will illuminate structures throughout discrete mathematics. For more detail, see the Category Theory Companion, Week 1–2.

Functions as Arrows

We've been writing $f : A \to B$ for functions. The arrow notation isn't accidental—it suggests *direction* and *connection*. Let's take this seriously.
Key observations:

- **Arrows compose:** If $f : A \to B$ and $g : B \to C$, we get $g \circ f : A \to C$.
- **Composition is associative:** $(h \circ g) \circ f = h \circ (g \circ f)$, so we can write $h \circ g \circ f$ without ambiguity.
- **Identity arrows exist:** Every set A has an identity function $\mathrm{id}_A : A \to A$ with $\mathrm{id}_A(x) = x$.
- **Identities are neutral:** $f \circ \mathrm{id}_A = f$ and $\mathrm{id}_B \circ f = f$.

Elements as Maps from 1

In **Set**, a singleton set $1 = \{*\}$ is special. Every element $a \in A$ corresponds to a unique map $1 \to A$ that sends $*$ to a. In categorical language, *elements of A are its points*, i.e., arrows $1 \to A$.

Terminal and Initial Objects

The singleton set 1 is a **terminal object**: for every set A, there is exactly one map $A \to 1$. The empty set 0 is an **initial object**: for every set A, there is exactly one map $0 \to A$. Thinking in terms of arrows makes "elements" and "emptiness" uniform across categories.

Products via Universal Properties

The Cartesian product $A \times B$ is the categorical product: for any X with maps $f : X \to A$ and $g : X \to B$, there is a unique map $\langle f, g\rangle : X \to A \times B$ making the projection triangle commute. This universal property will reappear in many disguises.

Diagrams as Visual Equations

A *commutative diagram* is a picture representing equations between composites of functions. Consider:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "h"'] & B \arrow[d, "g"] \\ & C \end{tikzcd}$$

This diagram **commutes** if $g \circ f = h$. In words: "going from A to C via B gives the same result as going directly."
A more complex example—a commutative square:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\ C \arrow[r, "k"'] & D \end{tikzcd}$$

This commutes if $g \circ f = k \circ h$. Both paths from A to D give the same composite.
Why diagrams? Complex equations become pictures you can *see*. Proofs become *path-finding*: to show two composites are equal, find paths in a commuting diagram connecting them.

Exercises: Arrows and Diagrams

1. Draw the diagram representing $h \circ g \circ f = k$ for functions $f : A \to B$, $g : B \to C$, $h : C \to D$, and $k : A \to D$. (Hint: it's a triangle with a long path.)

2. Consider this commutative square:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\ C \arrow[r, "k"'] & D \end{tikzcd}$$

   Write down the equation this diagram represents.

3. If the square above commutes, and we also have $m : D \to E$, draw the extended diagram. What new equation(s) can we derive involving m?

4. True or False: If $g \circ f = g \circ f'$, then $f = f'$. Either prove this or give a counterexample with small sets.

5. The identity law says $f \circ \mathrm{id}_A = f$. Draw this as a (degenerate) commutative triangle.

6. Associativity says $(h \circ g) \circ f = h \circ (g \circ f)$. Explain why this means we can unambiguously write $h \circ g \circ f$ without parentheses.

7. Let $f : \{1, 2\} \to \{a, b, c\}$ and $g : \{a, b, c\} \to \{x, y\}$ be given functions. How many functions $h : \{1, 2\} \to \{x, y\}$ are there such that the triangle with f, g, h commutes (i.e., $g \circ f = h$)? Is it always exactly one?

8. **Diagram chase:** Suppose triangles $g \circ f = h$ and $k \circ g = \ell$ both commute. Prove that $k \circ h = \ell \circ f$ by manipulating composites. (This is "chasing" around the combined diagram.)

Practice

1. Convert $\{x \in \mathbb{Z} : x^2 < 10\}$ into roster notation.
2. Prove $A \setminus B = A \cap B^c$ using the element method.
3. List $\mathcal{P}(\{a, b, c\})$ and verify its size.
4. Give a counterexample showing $A \cap (B \setminus C) = (A \cap B) \setminus C$ can fail.
5. Prove the absorption law: $A \cup (A \cap B) = A$.
6. If A has 4 elements and B has 3 elements, what is the maximum size of $A \cap B$? The minimum?
7. Prove that for any sets A, B, C: $(A \cap B) \cup (A \cap C) \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C) \cap (B \cup C)$.

3 Week 2: Functions and Cardinality

Reading

Epp §7.1–7.4.
Category theory companion: Week 1–2 (`category_theory_companion.pdf`).

Learning objectives

- Identify domain, codomain, and image of a function.
- Distinguish injective, surjective, and bijective functions.
- Use inverses and composition to solve functional equations.
- Compare sizes of sets using countability arguments.
- Apply the pigeonhole principle to function problems.

Key definitions and facts

Definition 3.1 (Function)**.** A function $f : A \to B$ assigns to each element $a \in A$ exactly one element $f(a) \in B$.

- **Domain:** the set A
- **Codomain:** the set B
- **Image/Range:** $\{f(a) : a \in A\} \subseteq B$

Definition 3.2 (Image and preimage of sets)**.** Let $f : A \to B$ be a function.

- For $S \subseteq A$, the **image** of S under f is $f(S) = \{f(x) : x \in S\} \subseteq B$.
- For $T \subseteq B$, the **preimage** (or inverse image) of T under f is $f^{-1}(T) = \{x \in A : f(x) \in T\} \subseteq A$.

Warning. The notation $f^{-1}(T)$ for preimage does *not* require f to have an inverse function. The preimage $f^{-1}(T)$ is always defined as a set, even when f is not bijective.

Definition 3.3 (Injective, surjective, bijective)**.** Let $f : A \to B$ be a function.

- f is **injective** (one-to-one) if $f(x) = f(y)$ implies $x = y$.
- f is **surjective** (onto) if for every $b \in B$, there exists $a \in A$ with $f(a) = b$.
- f is **bijective** if it is both injective and surjective.

Definition 3.4 (Inverse function)**.** If $f : A \to B$ is bijective, then $f^{-1} : B \to A$ exists and satisfies: $f^{-1}(f(a)) = a$ for all $a \in A$, and $f(f^{-1}(b)) = b$ for all $b \in B$.

Proposition 3.1 (Composition preserves properties)**.** *Let $f : A \to B$ and $g : B \to C$.*

- *If f and g are injective, then $g \circ f$ is injective.*
- *If f and g are surjective, then $g \circ f$ is surjective.*

- *If f and g are bijective, then $g \circ f$ is bijective with $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$.*

Proposition 3.2 (Composition tests)**.**
- *If $g \circ f$ is injective, then f is injective.*
- *If $g \circ f$ is surjective, then g is surjective.*

Definition 3.5 (Countable and uncountable)**.** A set A is **countably infinite** if there is a bijection $A \leftrightarrow \mathbb{N}$. A set is **countable** if it is finite or countably infinite. A set is **uncountable** if it is not countable.

Theorem 3.1 (Countability results)**.**
- *$\mathbb{Z}$ and $\mathbb{Q}$ are countable.*
- *The union of countably many countable sets is countable.*
- *$\mathbb{R}$ is uncountable (Cantor's diagonal argument).*
- *$|\mathcal{P}(\mathbb{N})| = |\mathbb{R}| > |\mathbb{N}|$.*

Worked examples

Example 3.1. Let $f : \mathbb{Z} \to \mathbb{Z}$ be defined by $f(n) = n^2 - 1$. Find $f(\{-2, 0, 3\})$ and $f^{-1}(\{0, 3, 8\})$.

Solution.

Image: Compute f on each element of $\{-2, 0, 3\}$:

- $f(-2) = (-2)^2 - 1 = 4 - 1 = 3$
- $f(0) = 0^2 - 1 = -1$
- $f(3) = 3^2 - 1 = 9 - 1 = 8$

Thus $f(\{-2, 0, 3\}) = \{-1, 3, 8\}$.

Preimage: Find all $n \in \mathbb{Z}$ such that $f(n) = n^2 - 1 \in \{0, 3, 8\}$.

- $n^2 - 1 = 0 \Rightarrow n^2 = 1 \Rightarrow n = \pm 1$
- $n^2 - 1 = 3 \Rightarrow n^2 = 4 \Rightarrow n = \pm 2$
- $n^2 - 1 = 8 \Rightarrow n^2 = 9 \Rightarrow n = \pm 3$

Thus $f^{-1}(\{0, 3, 8\}) = \{-3, -2, -1, 1, 2, 3\}$.

Example 3.2. Let $f : \mathbb{N} \to \mathbb{N}$ be $f(n) = 2n$. Show f is injective but not surjective.

Proof. **Injective:** Suppose $f(n) = f(m)$, i.e., $2n = 2m$. Dividing by 2, we get $n = m$. Thus f is injective.

Not surjective: Consider $1 \in \mathbb{N}$. If $f(n) = 1$, then $2n = 1$, which has no solution in $\mathbb{N}$. So 1 is not in the image of f. □

Example 3.3. Prove: If $g \circ f$ is injective, then f is injective.

Proof. Suppose $f(x) = f(y)$. Then $g(f(x)) = g(f(y))$, i.e., $(g \circ f)(x) = (g \circ f)(y)$. Since $g \circ f$ is injective, $x = y$. Thus f is injective. □

Example 3.4. Give a bijection between $\mathbb{Z}$ and $\mathbb{N}$.

Solution. Define $f : \mathbb{Z} \to \mathbb{N}$ by:

$$f(n) = \begin{cases} 2n & \text{if } n > 0 \\ -2n + 1 & \text{if } n \leq 0 \end{cases}$$

This maps: $0 \mapsto 1$, $-1 \mapsto 3$, $1 \mapsto 2$, $-2 \mapsto 5$, $2 \mapsto 4$, etc.

Example 3.5. Determine whether $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = x^3$ is bijective.

Solution.

Injective: Suppose $f(a) = f(b)$, i.e., $a^3 = b^3$. Taking cube roots (which is well-defined on $\mathbb{R}$), we get $a = b$. So f is injective.

Surjective: For any $y \in \mathbb{R}$, we need x with $x^3 = y$. Take $x = \sqrt[3]{y}$ (cube roots exist for all real numbers, including negatives). Then $f(x) = (\sqrt[3]{y})^3 = y$. So f is surjective.

Since f is both injective and surjective, f is bijective. The inverse is $f^{-1}(y) = \sqrt[3]{y}$.

Example 3.6. Prove that $(0, 1)$ is uncountable using Cantor's diagonal argument.

Proof. Suppose for contradiction that $(0, 1)$ is countable, so we can list all numbers in $(0, 1)$:

$$r_1 = 0.d_{11}d_{12}d_{13}\ldots, \quad r_2 = 0.d_{21}d_{22}d_{23}\ldots, \quad r_3 = 0.d_{31}d_{32}d_{33}\ldots, \quad \ldots$$

where each d_{ij} is a digit. Construct a new number $x = 0.e_1e_2e_3\ldots$ where:

$$e_n = \begin{cases} 5 & \text{if } d_{nn} \neq 5 \\ 6 & \text{if } d_{nn} = 5 \end{cases}$$

Then $x \in (0, 1)$ but $x \neq r_n$ for any n (they differ in the nth decimal place). This contradicts the assumption that all elements of $(0, 1)$ were listed. Therefore $(0, 1)$ is uncountable. □

Example 3.7. Let $f : A \to B$ and $g : B \to C$. Prove that if $g \circ f$ is surjective, then g is surjective.

Proof. Let $c \in C$. We need to find some $b \in B$ with $g(b) = c$.

Since $g \circ f$ is surjective, there exists $a \in A$ with $(g \circ f)(a) = c$, i.e., $g(f(a)) = c$.

Let $b = f(a) \in B$. Then $g(b) = g(f(a)) = c$.

So for every $c \in C$, we found $b \in B$ with $g(b) = c$. Thus g is surjective. □

Going Deeper: The Art of Diagram Chasing

Building on Week 1's introduction to diagrams, we now develop *diagram chasing* as a proof technique and encounter our first *universal property*. For more detail, see the Category Theory Companion, Week 1–2.

Diagram Chasing as Proof

Given that some diagram commutes, we often want to prove that certain composites are equal. The method: find two paths between the same endpoints and use commutativity.

Example. Suppose this diagram commutes:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\ C \arrow[r, "k"'] & D \end{tikzcd}$$

and we also know that $m \circ g = n \circ k$ for some m, n. Then:

$$m \circ g \circ f = n \circ k \circ h$$

Proof: $m \circ g \circ f = m \circ (g \circ f) = m \circ (k \circ h) = (m \circ k) \circ h$... wait, that's not quite right. Let's be more careful: $m \circ g \circ f = (m \circ g) \circ f = (n \circ k) \circ f$. Hmm, we need $k \circ h = g \circ f$. So: $m \circ g \circ f = m \circ (g \circ f) = m \circ (k \circ h)$. And $(m \circ g) \circ f = (n \circ k) \circ f$...

Actually, the key insight is simpler: from the square, $g \circ f = k \circ h$. So $m \circ g \circ f = m \circ k \circ h$. If additionally $m \circ k = n \circ k$... This shows why we must chase carefully!

Cancellation Properties

Recall from the main notes that injective functions are "left-cancellable":

$$f \circ g = f \circ h \implies g = h \quad \text{(when } f \text{ is injective)}$$

And surjective functions are "right-cancellable":

$$g \circ f = h \circ f \implies g = h \quad \text{(when } f \text{ is surjective)}$$

These properties can be drawn as diagrams:

$$\begin{tikzcd} X \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & A \arrow[r, "f"] & B \end{tikzcd} \quad \text{(left cancellation)} \qquad \begin{tikzcd} A \arrow[r, "f"] & B \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & X \end{tikzcd} \quad \text{(right cancellation)}$$

Isomorphisms, Sections, and Retractions

A map $f : A \to B$ is an **isomorphism** if there exists $g : B \to A$ with $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$. In **Set**, isomorphisms are exactly bijections.
A **section** (right inverse) is a map $s : B \to A$ with $f \circ s = \mathrm{id}_B$. A **retraction** (left inverse) is a map $r : B \to A$ with $r \circ f = \mathrm{id}_A$:

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[l, bend left, "s"] \end{tikzcd}$$

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[l, bend right, "r"'] \end{tikzcd}$$

Sections force f to be surjective; retractions force f to be injective.

Idempotents and Retracts

A map $p : A \to A$ is **idempotent** if $p \circ p = p$. In **Set**, idempotents correspond to "projection onto a subset": if $p = i \circ r$ with $r \circ i = \mathrm{id}_B$, then p projects A onto the retract B.

The "Unique Arrow" Pattern

A powerful pattern emerges: *there exists a unique arrow making the diagram commute.* When we have uniqueness:

- Any two arrows satisfying the condition must be equal
- This lets us prove equality by showing both arrows satisfy the same property

Universal Property of Products

The Cartesian product $A \times B$ satisfies a *universal property*: for any set X with functions $f : X \to A$ and $g : X \to B$, there exists a **unique** function $\langle f, g\rangle : X \to A \times B$ making this diagram commute:

$$\begin{tikzcd}
 & X \arrow[dl, "f"'] \arrow[d, dashed, "\langle f,g\rangle"] \arrow[dr, "g"] & \\
A & A \times B \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & B
\end{tikzcd}$$

The function is $\langle f, g\rangle(x) = (f(x), g(x))$. The universal property says this is the *only* way to "factor through" $A \times B$.
Why uniqueness matters: If someone gives you another function $h : X \to A \times B$ with $\pi_1 \circ h = f$ and $\pi_2 \circ h = g$, you can immediately conclude $h = \langle f, g\rangle$!

Exercises: Diagram Chasing

1. **Cancellation practice:** Let $f : A \to B$ be injective, and suppose $f \circ g = f \circ h$ for $g, h : X \to A$. Prove $g = h$ by considering what happens element-by-element.
2. Draw the diagram expressing "f is left-cancellable"—show g, h, and f, and indicate the conclusion $g = h$.
3. If $g \circ f$ is injective, prove that f must be injective. (Hint: Suppose $f(a) = f(b)$ and show $a = b$.)
4. If $g \circ f$ is injective, must g be injective? Prove or give a counterexample.
5. If $g \circ f$ is surjective, prove that g must be surjective.
6. If $g \circ f$ is surjective, must f be surjective? Prove or give a counterexample.
7. **Product verification:** Let $A = \{1, 2\}$, $B = \{a, b, c\}$, $X = \{*\}$ (a one-element set). Define $f(*) = 1$ and $g(*) = b$. What is $\langle f, g\rangle(*)$?
8. For $A = \{1, 2\}$, $B = \{a, b\}$, $X = \{x, y\}$, define $f(x) = 1$, $f(y) = 2$, $g(x) = a$, $g(y) = b$. Write out $\langle f, g\rangle$ explicitly.
9. **Uniqueness in action:** Suppose $h, h' : X \to A \times B$ both satisfy $\pi_1 \circ h = f$, $\pi_2 \circ h = g$ and the same for h'. Prove $h = h'$ using element-by-element reasoning.
10. **Diagram chase:** Given this commuting diagram:

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\
C \arrow[r, "k"'] & D
\end{tikzcd}$$

    Prove that $g \circ f = k \circ h$ by writing out what "the diagram commutes" means.

11. If $f : A \to B$ is a bijection with inverse $f^{-1} : B \to A$, draw the two triangles showing $f^{-1} \circ f = \mathrm{id}_A$ and $f \circ f^{-1} = \mathrm{id}_B$.

12. **Challenge:** Prove that if $f : A \to B$ has both a left inverse g (meaning $g \circ f = \mathrm{id}_A$) and a right inverse h (meaning $f \circ h = \mathrm{id}_B$), then $g = h$. (Hint: Compute $g \circ f \circ h$ two ways.)

Practice

1. Give an explicit bijection between $\mathbb{Z}$ and $\mathbb{N}$.
2. Decide whether $f(x) = x^3$ from $\mathbb{R}$ to $\mathbb{R}$ is bijective and justify.
3. Prove that if $g \circ f$ is injective, then f is injective.
4. Show that a finite set cannot be in bijection with a proper subset of itself.
5. Prove that $f : A \to B$ is injective iff there exists $g : B \to A$ with $g \circ f = \mathrm{id}_A$.
6. Prove that $f : A \to B$ is surjective iff there exists $g : B \to A$ with $f \circ g = \mathrm{id}_B$.
7. Show that the set of all finite subsets of $\mathbb{N}$ is countable.

4 Week 3: Relations and Modular Arithmetic

Reading

Epp §8.1–8.5.
Category theory companion: Week 3 (`category_theory_companion.pdf`).

Learning objectives

- Describe a relation using sets, matrices, or digraphs.
- Test whether a relation is reflexive, symmetric, or transitive.
- Form equivalence classes and connect them to partitions.
- Compute congruences and modular inverses.
- Apply the extended Euclidean algorithm.

Key definitions and facts

Definition 4.1 (Relation)**.** A **relation** from set A to set B is a subset $R \subseteq A \times B$. We write aRb or $(a, b) \in R$ to indicate that a is related to b. A relation on A is a relation from A to A.

Definition 4.2 (Properties of relations)**.** Let R be a relation on set A.

- R is **reflexive** if aRa for all $a \in A$.
- R is **symmetric** if aRb implies bRa for all $a, b \in A$.
- R is **antisymmetric** if aRb and bRa imply $a = b$ for all $a, b \in A$.
- R is **transitive** if aRb and bRc imply aRc for all $a, b, c \in A$.
- R is **irreflexive** if $\neg(aRa)$ for all $a \in A$.
- R is **total** if aRb or bRa for all $a, b \in A$.

Definition 4.3 (Equivalence relation)**.** A relation R on a set A is an **equivalence relation** if it is reflexive, symmetric, and transitive. For an equivalence relation $\sim$, the **equivalence class** of a is:

$$[a] = \{x \in A : x \sim a\}$$

Theorem 4.1 (Equivalence classes partition)**.** *If $\sim$ is an equivalence relation on A, then:*

1. *Every element belongs to exactly one equivalence class.*
2. *Two equivalence classes are either identical or disjoint.*
3. *The equivalence classes partition A: $A = \bigsqcup_{a \in A} [a]$.*

Definition 4.4 (Partial order)**.** A relation R on A is a **partial order** if it is reflexive, antisymmetric, and transitive. The pair (A, R) is called a **partially ordered set** (poset).

Definition 4.5 (Total order)**.** A partial order $\leq$ on A is a **total order** if for all $a, b \in A$, either $a \leq b$ or $b \leq a$.

Posets and Hasse diagrams

Definition 4.6 (Minimal/maximal vs least/greatest)**.** Let $(P, \leq)$ be a poset.

- $m \in P$ is **minimal** if there is no $x \in P$ with $x < m$.
- $m \in P$ is **maximal** if there is no $x \in P$ with $m < x$.
- $\ell \in P$ is the **least element** if $\ell \leq x$ for all $x \in P$.
- $g \in P$ is the **greatest element** if $x \leq g$ for all $x \in P$.

Least/greatest elements are unique if they exist; minimal/maximal elements need not be unique.

Definition 4.7 (Hasse diagram)**.** For a finite poset, the **Hasse diagram** is obtained by:

- Drawing one vertex per element.
- Drawing an edge upward from a to b if b *covers* a (i.e., $a < b$ and there is no c with $a < c < b$).
- Omitting self-loops and edges implied by transitivity.

Example 4.1. The divisibility poset on $\{1, 2, 3, 6\}$ has Hasse diagram:

$$\begin{tikzcd}
& 6 & \\
2 \arrow[ur] & & 3 \arrow[ul] \\
& 1 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Here 1 is least; 6 is greatest; 2 and 3 are incomparable.

Definition 4.8 (Bounds, meet, join, lattice)**.** Let $S \subseteq P$.

- u is an **upper bound** of S if $x \leq u$ for all $x \in S$.
- l is a **lower bound** of S if $l \leq x$ for all $x \in S$.
- The **least upper bound** (join) of $\{a, b\}$ is denoted $a \vee b$.
- The **greatest lower bound** (meet) of $\{a, b\}$ is denoted $a \wedge b$.

A poset is a **lattice** if every pair has both a meet and a join.

Remark. A **linear extension** is a total order that is consistent with the partial order. For finite posets, a linear extension can be found by repeatedly removing a minimal element (this is the same idea as topological sorting a DAG).

Definition 4.9 (Congruence modulo n)**.** For integers a, b and positive integer n, we say a is **congruent** to b modulo n, written $a \equiv b \pmod{n}$, if n divides $a - b$. Equivalently:

$$a \equiv b \pmod{n} \iff a \bmod n = b \bmod n \iff \exists k \in \mathbb{Z} : a = b + kn$$

Theorem 4.2 (Congruence is an equivalence relation). *For any positive integer n, congruence modulo n is an equivalence relation on $\mathbb{Z}$. The equivalence classes are the **residue classes**:*

$$[0], [1], [2], \ldots, [n-1]$$

The set of residue classes is denoted $\mathbb{Z}_n$ or $\mathbb{Z}/n\mathbb{Z}$.

Theorem 4.3 (Modular arithmetic). *For all $a, b, c, d \in \mathbb{Z}$ and positive integer n:*

1. *If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $a + c \equiv b + d \pmod{n}$.*
2. *If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $ac \equiv bd \pmod{n}$.*
3. *If $a \equiv b \pmod{n}$, then $a^k \equiv b^k \pmod{n}$ for all $k \geq 0$.*

Definition 4.10 (Modular inverse). For $a \in \mathbb{Z}_n$, the **modular inverse** of a modulo n is an integer b such that $ab \equiv 1 \pmod{n}$. We write $b = a^{-1} \pmod{n}$.

Theorem 4.4 (Existence of modular inverse). *The integer a has a modular inverse modulo n if and only if $\gcd(a, n) = 1$.*

Theorem 4.5 (Division algorithm). *For any integers a and b with $b > 0$, there exist unique integers q (quotient) and r (remainder) such that:*

$$a = bq + r \quad \text{and} \quad 0 \leq r < b$$

Theorem 4.6 (Euclidean algorithm). *For positive integers a and b, $\gcd(a, b)$ can be computed by repeatedly applying: $\gcd(a, b) = \gcd(b, a \bmod b)$, until one argument becomes 0.*

Theorem 4.7 (Extended Euclidean algorithm (Bézout's identity)). *For any positive integers a and b, there exist integers x and y such that:*

$$ax + by = \gcd(a, b)$$

Proof Strategy

To find the modular inverse of a modulo n when $\gcd(a, n) = 1$:

1. Use the extended Euclidean algorithm to find x, y with $ax + ny = 1$.
2. Then $a^{-1} \equiv x \pmod{n}$.

Representations of relations

Definition 4.11 (Matrix representation). A relation R on a finite set $A = \{a_1, \ldots, a_n\}$ can be represented by an $n \times n$ **relation matrix** M where:

$$M_{ij} = \begin{cases} 1 & \text{if } a_i R a_j \\ 0 & \text{otherwise} \end{cases}$$

Proposition 4.1 (Matrix properties). *For a relation matrix M:*

- *R is reflexive iff all diagonal entries are 1.*

- *R is symmetric iff $M = M^T$ (matrix is symmetric).*
- *R is antisymmetric iff $M_{ij} = 1$ and $M_{ji} = 1$ imply $i = j$.*
- *R is transitive iff M^2 (Boolean matrix product) has no 1 where M has 0.*

Definition 4.12 (Digraph representation)**.** A relation R on A can be represented as a directed graph (digraph) with vertices A and a directed edge from a to b whenever aRb.

Closures

Definition 4.13 (Closure)**.** The **reflexive closure** of R is the smallest reflexive relation containing R: $R \cup \{(a, a) : a \in A\}$.

The **symmetric closure** of R is the smallest symmetric relation containing R: $R \cup R^{-1}$ where $R^{-1} = \{(b, a) : (a, b) \in R\}$.

The **transitive closure** of R, denoted R^+, is the smallest transitive relation containing R.

Theorem 4.8 (Computing transitive closure)**.** *$R^+ = R \cup R^2 \cup R^3 \cup \cdots$ where $R^n = R \circ R^{n-1}$ (relation composition). For finite sets, $R^+ = R \cup R^2 \cup \cdots \cup R^n$ where $|A| = n$.*

Worked examples

Example 4.2. Show that congruence modulo n is an equivalence relation on $\mathbb{Z}$.

Proof.

- **Reflexive:** For any $a \in \mathbb{Z}$, we have $a - a = 0 = n \cdot 0$, so $n \mid (a - a)$. Thus $a \equiv a \pmod{n}$.
- **Symmetric:** Suppose $a \equiv b \pmod{n}$. Then $n \mid (a - b)$, so $a - b = nk$ for some integer k. Then $b - a = -nk = n(-k)$, so $n \mid (b - a)$. Thus $b \equiv a \pmod{n}$.
- **Transitive:** Suppose $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$. Then $a - b = nj$ and $b - c = nk$ for integers j, k. Adding: $a - c = (a - b) + (b - c) = nj + nk = n(j + k)$. So $n \mid (a - c)$, and $a \equiv c \pmod{n}$. $\square$

Example 4.3. Find the inverse of 3 modulo 7.

Solution. We need x such that $3x \equiv 1 \pmod{7}$.

Method 1 (Trial): Check $3 \cdot 1 = 3$, $3 \cdot 2 = 6$, $3 \cdot 3 = 9 \equiv 2$, $3 \cdot 4 = 12 \equiv 5$, $3 \cdot 5 = 15 \equiv 1 \pmod{7}$. So $3^{-1} \equiv 5 \pmod{7}$.

Method 2 (Extended Euclidean):

$$\begin{aligned} 7 &= 2 \cdot 3 + 1 \\ 1 &= 7 - 2 \cdot 3 = 1 \cdot 7 + (-2) \cdot 3 \end{aligned}$$

So $(-2) \cdot 3 + 1 \cdot 7 = 1$, meaning $3^{-1} \equiv -2 \equiv 5 \pmod{7}$.

Example 4.4. Let $A = \{1, 2, 3, 4\}$ and $R = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (3, 4), (4, 3), (4, 4)\}$. Show this is an equivalence relation and find the equivalence classes.

Solution.

- **Reflexive:** $(1, 1), (2, 2), (3, 3), (4, 4) \in R$. ✓
- **Symmetric:** For each $(a, b) \in R$ with $a \neq b$: $(1, 2) \in R$ and $(2, 1) \in R$; $(3, 4) \in R$ and $(4, 3) \in R$. ✓

- **Transitive:** Check all pairs: $(1,2),(2,1) \Rightarrow (1,1) \in R$; $(3,4),(4,3) \Rightarrow (3,3) \in R$. All other transitive requirements are satisfied. ✓

Equivalence classes: $[1] = [2] = \{1,2\}$ and $[3] = [4] = \{3,4\}$.
The partition is $\{\{1,2\},\{3,4\}\}$.

Example 4.5. Use the Euclidean algorithm to find $\gcd(252,105)$.
Solution.

$$\begin{aligned} 252 &= 2 \cdot 105 + 42 \\ 105 &= 2 \cdot 42 + 21 \\ 42 &= 2 \cdot 21 + 0 \end{aligned}$$

So $\gcd(252,105) = 21$.

Example 4.6. Use the extended Euclidean algorithm to express $\gcd(252,105)$ as a linear combination.
Solution. Working backwards:

$$\begin{aligned} 21 &= 105 - 2 \cdot 42 \\ &= 105 - 2 \cdot (252 - 2 \cdot 105) \\ &= 105 - 2 \cdot 252 + 4 \cdot 105 \\ &= 5 \cdot 105 + (-2) \cdot 252 \end{aligned}$$

So $21 = 5 \cdot 105 + (-2) \cdot 252$.

Example 4.7 (Toy RSA example)**.** This example illustrates RSA arithmetic (not secure for real use). Let $p = 5$ and $q = 11$, so $n = pq = 55$ and $\phi(n) = (p-1)(q-1) = 4 \cdot 10 = 40$.

Choose public exponent $e = 3$ (which is coprime to 40). Find the private exponent d such that $ed \equiv 1 \pmod{40}$.

Solution. We need $3d \equiv 1 \pmod{40}$. Using trial or the extended Euclidean algorithm:

$$40 = 13 \cdot 3 + 1 \implies 1 = 40 - 13 \cdot 3$$

So $d \equiv -13 \equiv 27 \pmod{40}$. Check: $3 \cdot 27 = 81 = 2 \cdot 40 + 1 \equiv 1 \pmod{40}$. ✓

Encryption: To encrypt message $m = 12$, compute $c \equiv m^e \pmod{n}$:

$$c \equiv 12^3 = 1728 \equiv 1728 - 31 \cdot 55 = 1728 - 1705 = 23 \pmod{55}$$

Decryption: To decrypt, compute $c^d \pmod{n}$. We have $23^{27} \pmod{55}$.

Using repeated squaring modulo 55: $23^2 = 529 \equiv 529 - 9 \cdot 55 = 34$, $23^4 \equiv 34^2 = 1156 \equiv 1156 - 21 \cdot 55 = 1$, so $23^{27} = 23^{24} \cdot 23^3 = (23^4)^6 \cdot 23^3 \equiv 1^6 \cdot 23^3 = 12167 \equiv 12 \pmod{55}$.

We recover the original message $m = 12$!

Example 4.8. Prove that if R is symmetric and transitive, and every element is related to some element, then R is reflexive.

Proof. Let $a \in A$. By assumption, there exists b such that aRb. By symmetry, bRa. By transitivity (using aRb and bRa), we get aRa. Since a was arbitrary, R is reflexive. □

Advanced number theory

The following theorems are powerful tools for modular arithmetic, especially in cryptography and algorithm design.

Definition 4.14 (Euler's totient function)**.** For a positive integer n, **Euler's totient function** $\phi(n)$ counts the integers from 1 to n that are coprime to n:

$$\phi(n) = |\{k : 1 \le k \le n \text{ and } \gcd(k, n) = 1\}|$$

Theorem 4.9 (Computing $\phi(n)$)**.**

- *If p is prime: $\phi(p) = p - 1$*
- *If p is prime: $\phi(p^k) = p^k - p^{k-1} = p^{k-1}(p - 1)$*
- *If $\gcd(m, n) = 1$: $\phi(mn) = \phi(m)\phi(n)$ (multiplicative)*

In general, if $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, then:

$$\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right) = n \cdot \frac{p_1 - 1}{p_1} \cdot \frac{p_2 - 1}{p_2} \cdots \frac{p_k - 1}{p_k}$$

Example 4.9. Compute $\phi(12)$.
Solution. $12 = 2^2 \cdot 3$. So:

$$\phi(12) = 12 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{3}\right) = 12 \cdot \frac{1}{2} \cdot \frac{2}{3} = 4$$

We can verify: the integers from 1 to 12 coprime to 12 are $\{1, 5, 7, 11\}$. Indeed, $|\{1, 5, 7, 11\}| = 4$.

Theorem 4.10 (Fermat's Little Theorem)**.** *If p is prime and $\gcd(a, p) = 1$, then:*

$$a^{p-1} \equiv 1 \pmod{p}$$

Equivalently, for any integer a: $a^p \equiv a \pmod{p}$.

Example 4.10. Compute $2^{100} \bmod 7$.
Solution. By Fermat's Little Theorem, $2^6 \equiv 1 \pmod 7$ (since 7 is prime and $\gcd(2, 7) = 1$).
Write $100 = 6 \cdot 16 + 4$. Then:

$$2^{100} = 2^{6 \cdot 16 + 4} = (2^6)^{16} \cdot 2^4 \equiv 1^{16} \cdot 16 \equiv 16 \equiv 2 \pmod 7$$

Theorem 4.11 (Euler's Theorem)**.** *If $\gcd(a, n) = 1$, then:*

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

This generalizes Fermat's Little Theorem (when $n = p$ is prime, $\phi(p) = p - 1$).

Example 4.11. Compute $7^{222} \bmod 12$.
Solution. Since $\gcd(7, 12) = 1$ and $\phi(12) = 4$, Euler's theorem gives $7^4 \equiv 1 \pmod{12}$.
Write $222 = 4 \cdot 55 + 2$. Then:

$$7^{222} = (7^4)^{55} \cdot 7^2 \equiv 1^{55} \cdot 49 \equiv 49 \equiv 1 \pmod{12}$$

Key Result

Euler's Theorem explains *why RSA works.* If $n = pq$ with primes p, q, and $ed \equiv 1 \pmod{\phi(n)}$, then for any message m coprime to n:

$$(m^e)^d = m^{ed} = m^{1+k\phi(n)} = m \cdot (m^{\phi(n)})^k \equiv m \cdot 1^k = m \pmod{n}$$

Decryption recovers the original message!

Theorem 4.12 (Chinese Remainder Theorem (CRT))**.** *Let $n_1, n_2, \ldots, n_k$ be pairwise coprime positive integers (i.e., $\gcd(n_i, n_j) = 1$ for $i \neq j$). Then for any integers $a_1, a_2, \ldots, a_k$, the system of congruences:*

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

has a unique solution modulo $N = n_1 n_2 \cdots n_k$.

Proof Strategy

To solve a CRT system with two moduli n_1 and n_2:

1. From $x \equiv a_1 \pmod{n_1}$, write $x = a_1 + n_1 t$ for some integer t.
2. Substitute into $x \equiv a_2 \pmod{n_2}$: solve $a_1 + n_1 t \equiv a_2 \pmod{n_2}$ for t.
3. Compute $x = a_1 + n_1 t$.

Example 4.12. Solve the system:

$$\begin{aligned} x &\equiv 2 \pmod{3} \\ x &\equiv 3 \pmod{5} \end{aligned}$$

Solution. From the first equation: $x = 2 + 3t$ for some integer t.
Substitute into the second: $2 + 3t \equiv 3 \pmod{5}$, so $3t \equiv 1 \pmod{5}$.
To find $3^{-1} \pmod{5}$: $3 \cdot 2 = 6 \equiv 1 \pmod{5}$, so $t \equiv 2 \pmod{5}$.
Thus $t = 2 + 5s$ for some s, and:

$$x = 2 + 3(2 + 5s) = 8 + 15s$$

So $x \equiv 8 \pmod{15}$.
Check: $8 = 2 + 2 \cdot 3$, so $8 \equiv 2 \pmod{3}$. ✓ $8 = 3 + 1 \cdot 5$, so $8 \equiv 3 \pmod{5}$. ✓

Example 4.13. Find the last two digits of 3^{100}.
Solution. We need $3^{100} \bmod 100$. Since $100 = 4 \cdot 25$ and $\gcd(4, 25) = 1$, we use CRT.
Modulo 4: $3^2 = 9 \equiv 1 \pmod{4}$, so $3^{100} = (3^2)^{50} \equiv 1 \pmod{4}$.
Modulo 25: $\phi(25) = 25(1 - 1/5) = 20$. By Euler, $3^{20} \equiv 1 \pmod{25}$.
Since $100 = 20 \cdot 5$: $3^{100} = (3^{20})^5 \equiv 1 \pmod{25}$.
Combine using CRT: We need x with $x \equiv 1 \pmod{4}$ and $x \equiv 1 \pmod{25}$.
Both congruences give $x \equiv 1$, so $x \equiv 1 \pmod{100}$.
The last two digits of 3^{100} are $\boxed{01}$.

Common Mistake

Forgetting the coprimality requirement. Fermat's Little Theorem requires $\gcd(a, p) = 1$. For example, $3^5 \not\equiv 1 \pmod 5$ because $\gcd(3, 5) = 1$... wait, that's wrong! Let's check: $3^4 = 81 \equiv 1 \pmod 5$. ✓
But $5^4 \not\equiv 1 \pmod 5$ because $\gcd(5, 5) = 5 \neq 1$. In fact, $5^4 \equiv 0 \pmod 5$.

Example 4.14. Write the relation matrix for "divides" on $\{1, 2, 3, 4\}$.

Solution. $a \mid b$ means a divides b.

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Row i, column j is 1 iff $i \mid j$. This is reflexive (diagonal is 1), antisymmetric (if $i \mid j$ and $j \mid i$ with $i, j \geq 1$, then $i = j$), and transitive. So divisibility is a partial order.

Going Deeper: Preorders—The Simplest Categories

This week we discover that the arrow-and-diagram language from Weeks 1–2 applies beautifully to familiar structures: preorders. This gives us concrete, easy-to-visualize categories to practice with. For more detail, see the Category Theory Companion, Week 3.

Preorders You Already Know

A *preorder* is a set with a reflexive and transitive relation. You've seen many:

- $(\mathbb{N}, \leq)$: natural numbers with "less than or equal to"
- $(\mathcal{P}(X), \subseteq)$: subsets of X ordered by inclusion
- $(Div_n, \mid)$: divisors of n ordered by divisibility
- $(\mathbb{Z}, \leq)$: integers with the usual ordering

Preorders as Categories

Here's the key insight: **a preorder IS a category**. Given $(P, \leq)$:

- **Objects:** Elements of P
- **Arrows:** There is exactly one arrow $a \to b$ if $a \leq b$, and no arrow otherwise
- **Identity:** $a \leq a$ (reflexivity) gives the identity arrow $a \to a$
- **Composition:** $a \leq b$ and $b \leq c$ imply $a \leq c$ (transitivity)

This is called a *thin category*: between any two objects, there's *at most one* arrow. The existence of an arrow $a \to b$ simply records that $a \leq b$.

Example: Divisibility. Consider $(\{1, 2, 3, 6\}, |)$:

$$\begin{tikzcd}
 & 6 & \\
2 \arrow[ur] & & 3 \arrow[ul] \\
 & 1 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

We have arrows $1 \to 2$ (since $1 \mid 2$), $2 \to 6$, etc. The composite $1 \to 2 \to 6$ equals the direct arrow $1 \to 6$ (both just say $1 \mid 6$).

What Composition Means Here

In a preorder category, composition is *invisible*—there's only one arrow between any two comparable elements anyway. But it corresponds exactly to **transitivity**:

$$\text{Arrow } a \to b \text{ and arrow } b \to c \text{ compose to give arrow } a \to c$$

This is just: $a \leq b$ and $b \leq c$ imply $a \leq c$.

Monotone Functions = Structure-Preserving Maps

In Weeks 1–2, we emphasized that the arrows between objects matter as much as the objects themselves. What are the "good" maps between preorders?
A function $f : P \to Q$ between preorders is **monotone** (or order-preserving) if:

$$a \leq b \implies f(a) \leq f(b)$$

In categorical language: *if there's an arrow $a \to b$, then there's an arrow $f(a) \to f(b)$*. The function f "preserves arrows."
Such a structure-preserving map between categories is called a **functor**.
Examples of monotone functions:

- $\lfloor \cdot \rfloor : (\mathbb{R}, \leq) \to (\mathbb{Z}, \leq)$ (floor function)
- $|S| : (\mathcal{P}(X), \subseteq) \to (\mathbb{N}, \leq)$ (cardinality, when X is finite)
- $n \mapsto 2n : (\mathbb{N}, \leq) \to (\mathbb{N}, \leq)$

Galois Connections (Adjunctions for Orders)

A **Galois connection** between preorders $(P, \leq)$ and $(Q, \leq)$ is a pair of monotone maps $F : P \to Q$ and $G : Q \to P$ such that:

$$F(p) \leq q \iff p \leq G(q)$$

This is the preorder version of an adjunction, one of the key ideas in category theory.
Example. The inclusion $\iota : \mathbb{Z} \to \mathbb{R}$ is left adjoint to floor:

$$\iota(n) \leq x \iff n \leq \lfloor x \rfloor$$

Similarly, ceiling is left adjoint to inclusion: $x \leq \iota(n) \iff \lceil x \rceil \leq n$.

Products and Coproducts in Preorders

The universal properties from Week 2 specialize nicely to preorders:

- **Product** of a and b = greatest lower bound (meet): the largest c with $c \leq a$ and $c \leq b$
- **Coproduct** of a and b = least upper bound (join): the smallest c with $a \leq c$ and $b \leq c$

Example. In $(\{1, 2, 3, 6\}, |)$:

- Product of 2 and 3 is $\gcd(2, 3) = 1$
- Coproduct of 2 and 3 is $\operatorname{lcm}(2, 3) = 6$

Exercises: Preorders as Categories

1. Draw the divisibility preorder on $\{1, 2, 4, 8\}$ as a diagram with arrows. Is this a total order (every two elements comparable)?
2. List all arrows in the preorder $(\mathcal{P}(\{1, 2\}), \subseteq)$. How many are there? (Don't forget identity arrows!)
3. Verify that transitivity is composition: In $\{1, 2, 4, 8\}$ with divisibility, we have $1 \mid 2$ and $2 \mid 4$. Draw this as composing arrows $1 \to 2 \to 4$, giving $1 \to 4$.
4. Is the floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$ monotone? Prove or disprove.
5. Is the function $f(n) = n^2$ monotone from $(\mathbb{Z}, \leq)$ to $(\mathbb{Z}, \leq)$? What about from $(\mathbb{N}, \leq)$ to $(\mathbb{N}, \leq)$?
6. In $(\{1, 2, 3, 6\}, |)$, what is the product (greatest lower bound) of 2 and 3? What is their coproduct (least upper bound)?
7. **Universal property check:** In divisibility, verify that $\gcd(6, 10)$ satisfies: for any d with $d \mid 6$ and $d \mid 10$, we have $d \mid \gcd(6, 10)$.
8. Draw the diagram expressing the universal property of $\gcd(a, b)$ using arrows in a divisibility preorder.
9. How many monotone functions are there from $(\{0, 1\}, \leq)$ to $(\{a, b, c\}, \leq)$ where $a \leq b \leq c$? (Hint: Where can 0 go? Then where can 1 go?)
10. Give an example of a function $\{1, 2, 3\} \to \{1, 2, 3\}$ that is NOT monotone for the usual $\leq$.
11. **Functors compose:** Prove that if $f : P \to Q$ and $g : Q \to R$ are both monotone, then $g \circ f : P \to R$ is monotone.
12. **Challenge:** In $(\mathcal{P}(\{1, 2, 3\}), \subseteq)$, what is the product (greatest lower bound) of $\{1, 2\}$ and $\{2, 3\}$? The coproduct (least upper bound)? Verify using the universal property.

Common Mistake

Confusing symmetric and antisymmetric. These are *not* opposites!

- Symmetric: $aRb \Rightarrow bRa$
- Antisymmetric: $aRb \wedge bRa \Rightarrow a = b$

A relation can be both (e.g., $=$), neither, or just one. The identity relation $\{(a, a) : a \in A\}$ is both symmetric and antisymmetric.

Practice

1. For $n = 5$, list the equivalence classes of $\mathbb{Z}$ modulo n.
2. Find the inverse of 3 modulo 7 using the extended Euclidean algorithm.
3. Decide whether the relation xRy iff $x - y$ is even is an equivalence relation on $\mathbb{Z}$.
4. Prove that every equivalence relation on A defines a partition of A.
5. Let $R = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} : a \leq b\}$. Which properties does R have: reflexive, symmetric, antisymmetric, transitive?
6. Find $\gcd(1071, 462)$ and express it as a linear combination of 1071 and 462.
7. Solve $17x \equiv 1 \pmod{43}$.
8. Let R be a relation on $\{1, 2, 3\}$ with matrix:

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

   Find the transitive closure R^+ and give its matrix.
9. Prove: If $a \equiv b \pmod{n}$ and $c \mid n$, then $a \equiv b \pmod{c}$.
10. Show that the intersection of two equivalence relations on A is an equivalence relation.
11. Compute $\phi(60)$ using the formula.
12. Use Fermat's Little Theorem to find $5^{302} \bmod 7$.
13. Use Euler's Theorem to find $3^{340} \bmod 11$.
14. Solve the system: $x \equiv 1 \pmod{4}$, $x \equiv 2 \pmod{5}$, $x \equiv 3 \pmod{7}$.
15. What are the last two digits of 7^{2024}?
16. Prove Fermat's Little Theorem for $a = 2$ and $p = 5$ by direct computation.
17. Draw the Hasse diagram for divisibility on $\{1, 2, 3, 6, 12\}$. Identify all minimal and maximal elements, and state whether a least or greatest element exists.
18. In $(\mathcal{P}(\{1, 2, 3\}), \subseteq)$, compute $\{1, 2\} \wedge \{2, 3\}$ and $\{1, 2\} \vee \{2, 3\}$.

19. Consider a poset with relations $a < b$, $a < c$, $b < d$, $c < d$. Give one linear extension and one different linear extension.

20. Does the poset $(\mathbb{Z}, \leq)$ have minimal or maximal elements? Does it have a least or greatest element? Justify your answers.

5 Week 4: Counting and Probability I

Reading

Epp §9.1–9.4, 9.9.
Category theory companion: Weeks 4–5 (`category_theory_companion.pdf`).

Learning objectives

- Build sample spaces and events for basic probability models.
- Apply the multiplication rule to count outcomes.
- Apply the addition rule and inclusion–exclusion for two sets.
- Use the pigeonhole principle to force collisions.
- Count permutations and arrangements with restrictions.

Key definitions and facts

Definition 5.1 (Sample space and event)**.** A **sample space** S is the set of all possible outcomes of an experiment. An **event** is a subset $E \subseteq S$. The event E **occurs** if the outcome of the experiment is in E.

Definition 5.2 (Probability (equally likely outcomes))**.** If all outcomes in a finite sample space S are equally likely, then for any event E:

$$P(E) = \frac{|E|}{|S|}$$

Theorem 5.1 (Basic probability properties)**.** *For any events A, B in sample space S:*

1. $0 \leq P(A) \leq 1$
2. $P(S) = 1$ *and* $P(\emptyset) = 0$
3. $P(A^c) = 1 - P(A)$ *where* $A^c = S \setminus A$ *is the complement of* A
4. *If* $A \cap B = \emptyset$*, then* $P(A \cup B) = P(A) + P(B)$

Theorem 5.2 (Multiplication rule (product rule))**.** *If a procedure can be broken into k successive steps, where step 1 can be done in n_1 ways, step 2 can be done in n_2 ways (regardless of step 1's outcome), ..., step k can be done in n_k ways, then the total number of ways to complete the procedure is:*

$$n_1 \times n_2 \times \cdots \times n_k$$

Theorem 5.3 (Addition rule (sum rule))**.** *If a task can be done either by method A (in n_1 ways) or by method B (in n_2 ways), and the two methods are mutually exclusive (no overlap), then the total number of ways is:*

$$n_1 + n_2$$

Theorem 5.4 (Inclusion-exclusion (two sets)). *For any finite sets A and B:*

$$|A \cup B| = |A| + |B| - |A \cap B|$$

For probabilities:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Theorem 5.5 (Inclusion-exclusion (three sets)). *For finite sets A, B, C:*

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

Theorem 5.6 (Inclusion-exclusion (general)). *For finite sets $A_1, A_2, \ldots, A_n$:*

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_i |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n-1}|A_1 \cap A_2 \cap \cdots \cap A_n|$$

The pattern alternates: add singles, subtract pairs, add triples, subtract quadruples, etc.

Example 5.1. How many integers in $\{1, \ldots, 100\}$ are divisible by 2, 3, or 5?

Solution. Let A_2, A_3, A_5 be the sets divisible by 2, 3, 5 respectively.

$$\begin{aligned} |A_2| = 50, \quad |A_3| = 33, \quad |A_5| = 20 \\ |A_2 \cap A_3| = |A_6| = 16, \quad |A_2 \cap A_5| = |A_{10}| = 10, \quad |A_3 \cap A_5| = |A_{15}| = 6 \\ |A_2 \cap A_3 \cap A_5| = |A_{30}| = 3 \end{aligned}$$

By inclusion-exclusion:

$$|A_2 \cup A_3 \cup A_5| = 50 + 33 + 20 - 16 - 10 - 6 + 3 = 74$$

Conditional Probability and Independence

Definition 5.3 (Conditional probability). If $P(B) > 0$, the conditional probability of A given B is:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Theorem 5.7 (Multiplication rule).

$$P(A \cap B) = P(A \mid B)\, P(B)$$

More generally:

$$P(A \cap B \cap C) = P(A)\, P(B \mid A)\, P(C \mid A \cap B)$$

Definition 5.4 (Independence). Events A and B are **independent** if:

$$P(A \cap B) = P(A)P(B)$$

Equivalently (when $P(B) > 0$), $P(A \mid B) = P(A)$.

Example 5.2. Two cards are drawn from a standard deck without replacement. Let A = "first card is an ace" and B = "second card is an ace".

$$P(A) = \frac{4}{52} = \frac{1}{13}, \quad P(B \mid A) = \frac{3}{51} = \frac{1}{17}$$

So $P(A \cap B) = \frac{1}{13} \cdot \frac{1}{17} = \frac{1}{221}$. Since $P(B) = \frac{1}{13}$, we have $P(A \cap B) \neq P(A)P(B)$, so A and B are not independent.

Law of Total Probability and Bayes' Rule

Theorem 5.8 (Law of total probability)**.** *If $\{B_1, \dots, B_k\}$ is a partition of S with $P(B_i) > 0$, then:*

$$P(A) = \sum_{i=1}^{k} P(A \mid B_i) P(B_i)$$

Theorem 5.9 (Bayes' rule)**.** *Under the same assumptions:*

$$P(B_j \mid A) = \frac{P(A \mid B_j) P(B_j)}{\sum_{i=1}^{k} P(A \mid B_i) P(B_i)}$$

Example 5.3. A factory has two machines. M_1 produces 70% of items with a 2% defect rate; M_2 produces 30% with a 5% defect rate.

$$P(D) = 0.7 \cdot 0.02 + 0.3 \cdot 0.05 = 0.029$$

If an item is defective, then:

$$P(M_2 \mid D) = \frac{0.3 \cdot 0.05}{0.029} = \frac{15}{29} \approx 0.517$$

So a defective item is slightly more likely to come from M_2.

Derangements

Definition 5.5 (Derangement)**.** A **derangement** of $\{1, 2, \dots, n\}$ is a permutation σ with no fixed points: $\sigma(i) \neq i$ for all i. The number of derangements of n elements is denoted D_n (or $!n$).

Theorem 5.10 (Counting derangements)**.** *The number of derangements of n elements is:*

$$D_n = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!} = n! \left(1 - 1 + \frac{1}{2!} - \frac{1}{3!} + \cdots + \frac{(-1)^n}{n!}\right)$$

Equivalently:

$$D_n = n! \left(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + \frac{(-1)^n}{n!}\right)$$

Key Result

For large n: $D_n \approx \frac{n!}{e}$ (rounded to the nearest integer). The probability that a random permutation is a derangement approaches $\frac{1}{e} \approx 0.368$ as $n \to \infty$.

Example 5.4. Compute D_4 (derangements of 4 elements).

Solution. Using the formula:

$$D_4 = 4! \left(1 - 1 + \frac{1}{2} - \frac{1}{6} + \frac{1}{24}\right) = 24 \left(\frac{12 - 4 + 1}{24}\right) = 24 \cdot \frac{9}{24} = 9$$

We can verify by listing: the derangements of $\{1, 2, 3, 4\}$ are:

$$2143,\ 2341,\ 2413,\ 3142,\ 3412,\ 3421,\ 4123,\ 4312,\ 4321$$

Indeed, there are 9.

Example 5.5 (The Hat Check Problem)**.** A hat check attendant returns n hats to n people at random. What is the probability that nobody gets their own hat?

Solution. There are $n!$ equally likely ways to distribute hats. The favorable outcomes are derangements: D_n.

$$P(\text{no one gets own hat}) = \frac{D_n}{n!} = \sum_{k=0}^{n} \frac{(-1)^k}{k!} \approx \frac{1}{e} \approx 0.368$$

Proof Strategy

[Deriving the derangement formula via inclusion-exclusion] Let A_i = permutations where person i gets their own hat (a fixed point at position i).
We want $D_n = n! - |A_1 \cup A_2 \cup \cdots \cup A_n|$.

- $|A_i| = (n-1)!$ (fix position i, permute the rest)
- $|A_i \cap A_j| = (n-2)!$ (fix positions i and j)
- In general: $|A_{i_1} \cap \cdots \cap A_{i_k}| = (n-k)!$
- There are $\binom{n}{k}$ ways to choose k positions

By inclusion-exclusion:

$$\begin{aligned} |A_1 \cup \cdots \cup A_n| &= \binom{n}{1}(n-1)! - \binom{n}{2}(n-2)! + \binom{n}{3}(n-3)! - \cdots \\ &= \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k}(n-k)! = \sum_{k=1}^{n} (-1)^{k-1} \frac{n!}{k!} \end{aligned}$$

So $D_n = n! - \sum_{k=1}^{n} (-1)^{k-1} \frac{n!}{k!} = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}$.

Definition 5.6 (Permutation)**.** A **permutation** of a set is an arrangement of its elements in a sequence. The number of permutations of n distinct objects is:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$$

By convention, $0! = 1$.

Definition 5.7 (r-permutation)**.** An r**-permutation** of n objects is an ordered arrangement of r objects chosen from n distinct objects. The count is:

$$P(n,r) = \frac{n!}{(n-r)!} = n \times (n-1) \times \cdots \times (n-r+1)$$

Theorem 5.11 (Pigeonhole principle (basic))**.** *If $n+1$ objects are placed into n boxes, then at least one box contains at least 2 objects.*

Theorem 5.12 (Pigeonhole principle (generalized))**.** *If n objects are placed into k boxes, then at least one box contains at least $\lceil n/k \rceil$ objects.*

Key Result

The pigeonhole principle guarantees existence but doesn't tell you *which* box has multiple objects. It's powerful for proving that certain configurations must exist.

Definition 5.8 (Distinguishable vs. indistinguishable)**.** Objects are **distinguishable** if we can tell them apart; **indistinguishable** if we cannot. Similarly for boxes/positions.

- Distinguishable objects into distinguishable boxes: use multiplication rule
- Indistinguishable objects into distinguishable boxes: use combinations (Week 5)

Counting techniques

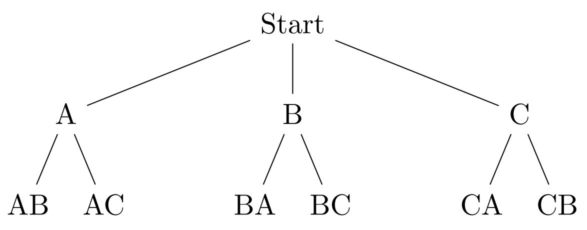
Proof Strategy

Direct counting: Count the desired outcomes directly using multiplication/addition rules.
Complement counting: Sometimes it's easier to count what you *don't* want:

$$|\text{desired}| = |\text{total}| - |\text{undesired}|$$

Definition 5.9 (Possibility tree)**.** A **possibility tree** (or decision tree) is a diagram that systematically lists all possible outcomes of a multi-step process. Each branch point represents a choice, and each path from root to leaf represents one complete outcome. The total count equals the number of leaves.

Example 5.6 (Possibility tree)**.** How many 2-letter strings over $\{A, B, C\}$ have no repeated letters?
Solution via possibility tree:



First letter: 3 choices. Second letter: 2 choices (can't repeat). Total: $3 \times 2 = 6$ strings.

Key Result

Possibility trees are most useful when:

- The number of outcomes is small enough to draw
- Choices at each step depend on previous choices
- You need to verify your counting is correct

For larger problems, use the multiplication/addition rules directly.

Definition 5.10 (With vs. without replacement)**.** • **With replacement:** After selecting an object, it goes back into the pool. Selections are independent.

- **Without replacement:** Once selected, an object is removed from the pool. Later selections have fewer choices.

Proposition 5.1 (Counting sequences)**.** *From a set of n distinct elements:*

- *Sequences of length k **with replacement**:* n^k
- *Sequences of length k **without replacement**:* $P(n,k) = \frac{n!}{(n-k)!}$

Worked examples

Example 5.7. A license plate consists of 3 letters followed by 3 digits. How many license plates are possible?

Solution. Using the multiplication rule:

- 3 letters: $26 \times 26 \times 26 = 26^3$ choices (with replacement)
- 3 digits: $10 \times 10 \times 10 = 10^3$ choices

Total: $26^3 \times 10^3 = 17,576 \times 1000 = 17,576,000$.

Example 5.8. How many 3-letter strings over $\{A, B, C, D\}$ have no repeated letters?

Solution. This is a 3-permutation of 4 objects:

$$P(4,3) = 4 \times 3 \times 2 = 24$$

Alternatively: First letter: 4 choices. Second letter: 3 choices (can't repeat first). Third letter: 2 choices.

Example 5.9. A fair die is rolled twice. What is the probability that the sum is 7?

Solution.

- Sample space: All pairs (a,b) with $a, b \in \{1,2,3,4,5,6\}$. Size: $6 \times 6 = 36$.
- Event E: pairs summing to 7. These are $(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)$. Size: 6.

$$P(E) = \frac{6}{36} = \frac{1}{6}$$

Example 5.10. How many 5-bit binary strings contain at least one 1?

Solution (complement counting).

- Total 5-bit strings: $2^5 = 32$
- Strings with no 1s: just 00000, so 1

Strings with at least one 1: $32 - 1 = 31$.

Example 5.11. How many integers in $\{1, 2, \ldots, 100\}$ are divisible by 2 or 3?

Solution (inclusion-exclusion). Let $A = \{n : 2 \mid n\}$ and $B = \{n : 3 \mid n\}$.

- $|A| = \lfloor 100/2 \rfloor = 50$
- $|B| = \lfloor 100/3 \rfloor = 33$
- $|A \cap B| = |\{n : 6 \mid n\}| = \lfloor 100/6 \rfloor = 16$

$$|A \cup B| = 50 + 33 - 16 = 67$$

Example 5.12. Prove: Among any 13 people, at least two share a birth month.

Solution. There are 12 months (boxes) and 13 people (objects). By the pigeonhole principle, at least one month contains at least 2 people.

Example 5.13. Prove: In any set of 6 integers, there exist two with the same remainder when divided by 5.

Solution. Remainders modulo 5 are in $\{0, 1, 2, 3, 4\}$ (5 boxes). With 6 integers (objects), by pigeonhole, at least two have the same remainder.

Example 5.14. How many ways can 8 people sit in a row if two specific people (Alice and Bob) must sit together?

Solution. Treat Alice and Bob as a single "super-person." Then we arrange 7 objects in a row: 7! ways. But Alice and Bob can be in either order within their block: 2 ways.

Total: $7! \times 2 = 5040 \times 2 = 10080$.

Example 5.15. How many ways can 8 people sit in a row if Alice and Bob must NOT sit together?

Solution (complement counting).

- Total arrangements: $8! = 40320$
- Arrangements where they sit together: 10080 (from previous example)

Answer: $40320 - 10080 = 30240$.

Example 5.16. A committee of 5 is to be chosen from 8 candidates. In how many ways can this be done if the order of selection matters?

Solution. This is a 5-permutation of 8:

$$P(8, 5) = 8 \times 7 \times 6 \times 5 \times 4 = 6720$$

Example 5.17. Prove: Among any 5 points placed inside a unit square, at least two are within distance $\frac{\sqrt{2}}{2}$ of each other.

Solution. Divide the unit square into 4 smaller squares of side $\frac{1}{2}$. By pigeonhole, at least two of the 5 points lie in the same small square. The maximum distance between two points in a square of side $\frac{1}{2}$ is the diagonal length: $\frac{\sqrt{2}}{2}$.

Common Mistake

Overcounting. When counting arrangements, make sure you're not counting the same configuration multiple times. Ask yourself:

- Does order matter? (permutation vs. combination)
- Are objects distinguishable?
- Are positions/boxes distinguishable?

Common Mistake

Misapplying the multiplication rule. The multiplication rule requires that the number of choices at each step is *independent* of previous choices (or carefully accounted for). If earlier choices affect later options, you must account for this.

Going Deeper: The Algebra of Types

The counting rules we've learned—multiplication and addition—have a surprising connection to types in programming. This connection reveals why these rules are so fundamental. For more detail, see the Category Theory Companion, Weeks 4–5.

Types Have Sizes

In programming, a *type* is a set of values. We can count how many values a type has:

Type	**Description**	**Size**
`Void`	empty type (no values)	0
`Unit` or `()`	type with one value	1
`Bool`	`True` or `False`	2
`Char`	ASCII characters	128 (or 256)

Products and Sums of Types

Now here's the magic. If we combine types:

- **Product type** (A, B) (pairs): $|A \times B| = |A| \times |B|$
- **Sum type** `Either A B`: $|A + B| = |A| + |B|$

This is exactly the multiplication and addition rules for counting!
Example. The type `(Bool, Bool)` has $2 \times 2 = 4$ values:

```
(True, True), (True, False), (False, True), (False, False)
```

Example. The type `Either Bool ()` has $2 + 1 = 3$ values:

```
Left True, Left False, Right ()
```

Function Types as Exponentials (Map Objects)

Function types correspond to **exponentials** in category theory. In **Set**, the object B^A is the set of all functions $A \to B$. It comes with an *evaluation* map:

$$\mathrm{ev} : B^A \times A \to B, \quad \mathrm{ev}(f, a) = f(a)$$

The universal property says: any $g : X \times A \to B$ uniquely corresponds to a map $\tilde{g} : X \to B^A$ ("currying").

Why the Names "Product" and "Sum"?

This isn't a coincidence! The names come from the fact that type sizes multiply and add. The categorical perspective (from Week 2) explains why:

- Products satisfy the universal property of products
- Sums (coproducts) satisfy the dual universal property

Algebraic Laws

These type operations satisfy the same laws as arithmetic:

- $A \times 1 \cong A$ (pairing with unit adds no information)
- $A + 0 \cong A$ (sum with empty type is just A)
- $A \times (B + C) \cong (A \times B) + (A \times C)$ (distributivity)
- $A \times B \cong B \times A$ (commutativity)

Exercises: Types and Counting

1. How many values does the type `(Bool, Bool, Bool)` have? List them.
2. How many values does `Either Bool Bool` have? How is this different from `(Bool, Bool)`?
3. If type A has 3 values and type B has 4 values, how many values does $A \times B$ have? How many does $A + B$ have?
4. The type `Maybe A` is defined as `Either () A` (either "nothing" or a value of type A). If $|A| = n$, what is $|$`Maybe A`$|$?
5. Verify the distributive law by counting: if $|A| = 2$, $|B| = 3$, $|C| = 1$, check that $|A \times (B + C)| = |A \times B| + |A \times C|$.
6. **Challenge:** Function types $A \to B$ have $|B|^{|A|}$ values (every function is a choice of output for each input). Verify this for $A =$ `Bool` and $B = \{1, 2, 3\}$ by listing all functions.

Practice

1. A fair die is rolled twice. What is the probability the sum is 7?
2. How many 5-bit binary strings contain at least one 1?
3. Use inclusion–exclusion to count integers in $\{1, \dots, 100\}$ divisible by 2 or 3.
4. Use the pigeonhole principle to show that among 13 people, two share a birth month.
5. How many 4-digit PINs (using digits 0–9) have no repeated digits?

6. A standard deck has 52 cards. How many 5-card hands are possible? (Order doesn't matter—use $\binom{52}{5}$ from Week 5, or just set up the problem.)
7. How many ways can 6 different books be arranged on a shelf if two specific books must be at the ends?
8. How many bit strings of length 8 start with 1 or end with 00?
9. Prove: Among any 10 integers, there exist two whose difference is divisible by 9.
10. A restaurant offers 3 appetizers, 5 main courses, and 2 desserts. How many different 3-course meals are possible?
11. How many permutations of ABCDEF contain ABC as a consecutive substring?
12. Prove: If 5 points are selected from the integer lattice points in $\{0, 1, 2\}^2$, then two of them have midpoint that is also a lattice point.
13. Use inclusion-exclusion to count integers in $\{1, \ldots, 1000\}$ divisible by 2, 3, or 5.
14. Compute D_5 (the number of derangements of 5 elements).
15. Five friends exchange gifts so that no one receives their own gift. How many ways can this be done?
16. How many permutations of $\{1, 2, 3, 4, 5, 6\}$ have at least one fixed point? (Hint: Use complement counting with derangements.)
17. Use inclusion-exclusion to count the number of surjective functions from a 4-element set to a 3-element set. (Hint: Let A_i be functions missing element i in the image.)
18. Two fair dice are rolled. Let A be the event "the sum is even" and B be the event "the sum is at least 10." Compute $P(A \mid B)$ and determine whether A and B are independent.
19. A box contains 3 fair coins and 1 double-headed coin. A coin is chosen uniformly at random and flipped once. If it lands heads, what is the probability the chosen coin was double-headed?
20. A test for a disease has sensitivity 0.95 and false positive rate 0.02. If 1% of the population has the disease, compute the probability that a person who tests positive actually has the disease.
21. Suppose $P(A) = 0.4$, $P(B) = 0.5$, and $P(A \cap B) = 0.2$. Compute $P(A \mid B)$ and $P(A \cup B)$, and determine whether A and B are independent.

6 Week 5: Counting and Probability II

Reading

Epp §9.5–9.7; 5.6–5.7. Supplemental: generating functions.
Category theory companion: Weeks 4–5 (`category_theory_companion.pdf`).

Learning objectives

- Compute combinations and binomial coefficients.
- Count with repetition using stars and bars.
- Use Pascal's identity and the binomial theorem.
- Apply counting techniques to probability problems.

Key definitions and facts

Definition 6.1 (Binomial coefficient)**.** $\binom{n}{k} = \dfrac{n!}{k!(n-k)!}$ counts k-element subsets of an n-element set. Equivalently, it is the number of ways to choose k items from n items without regard to order.

Theorem 6.1 (Pascal's identity)**.** *For $1 \le k \le n$:*

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Combinatorial proof: *Consider element n. Either it's in the subset (choose $k-1$ more from $n-1$) or it's not (choose k from $n-1$).*

Theorem 6.2 (Binomial theorem)**.** $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$.

Theorem 6.3 (Stars and bars)**.** *The number of ways to place n identical objects into k distinct bins is $\binom{n+k-1}{k-1}$. Equivalently, this counts non-negative integer solutions to $x_1 + x_2 + \cdots + x_k = n$.*

Theorem 6.4 (Useful identities)**.**
- $\binom{n}{k} = \binom{n}{n-k}$ *(symmetry)*
- $\sum_{k=0}^{n} \binom{n}{k} = 2^n$ *(total subsets)*
- $\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$ *(alternating sum)*
- $\binom{n}{0} + \binom{n}{2} + \cdots = \binom{n}{1} + \binom{n}{3} + \cdots = 2^{n-1}$

Definition 6.2 (Multinomial coefficient)**.** The number of ways to partition n objects into groups of sizes $k_1, k_2, \ldots, k_r$ (where $k_1 + \cdots + k_r = n$) is:

$$\binom{n}{k_1, k_2, \ldots, k_r} = \frac{n!}{k_1! k_2! \cdots k_r!}$$

Recurrence relations for sequences

Definition 6.3 (Recurrence relation)**.** A **recurrence relation** defines a sequence $\{a_n\}$ by expressing a_n in terms of earlier terms (and initial conditions). Example: $a_0 = 1, a_1 = 1$, and $a_n = a_{n-1} + a_{n-2}$ for $n \geq 2$ (Fibonacci numbers).

Theorem 6.5 (Second-order linear homogeneous recurrences)**.** *If $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ for $n \geq 2$, then the* ***characteristic equation*** *is*

$$r^2 = c_1 r + c_2$$

If the roots are distinct r_1, r_2, then $a_n = \alpha r_1^n + \beta r_2^n$. If there is a repeated root r, then $a_n = (\alpha + \beta n) r^n$. Constants are determined from the initial conditions.

Example 6.1. Solve $a_n = 2a_{n-1} + 1$ with $a_0 = 0$.

Solution. Unroll the recurrence:

$$a_n = 2a_{n-1} + 1 = 2(2a_{n-2} + 1) + 1 = 2^2 a_{n-2} + 2 + 1$$

Continuing gives $a_n = 2^n a_0 + (2^{n-1} + \cdots + 2 + 1) = 2^n - 1$.

Going Deeper: Generating Functions

The **ordinary generating function** of a sequence $\{a_n\}$ is:

$$A(x) = \sum_{n=0}^{\infty} a_n x^n$$

Example 1 (constant sequence). If $a_n = 1$ for all n, then:

$$A(x) = 1 + x + x^2 + \cdots = \frac{1}{1-x}$$

Example 2 (Fibonacci). Let $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$. If $F(x) = \sum_{n \geq 0} F_n x^n$, then the recurrence gives:

$$F(x) - x = xF(x) + x^2 F(x) \quad \Rightarrow \quad F(x) = \frac{x}{1 - x - x^2}$$

Generating functions provide a systematic way to solve recurrences and count combinatorial objects.

Category Lens: Subsets as Maps to 2

In **Set**, every subset $S \subseteq A$ corresponds to a **characteristic function**:

$$\chi_S : A \to 2, \quad \chi_S(a) = \begin{cases} 1 & \text{if } a \in S \\ 0 & \text{if } a \notin S \end{cases}$$

So the set of all subsets of A is the exponential object 2^A.

- $|2^A| = 2^{|A|}$ explains the total number of subsets.
- $\binom{n}{k}$ counts functions $A \to 2$ with exactly k elements mapped to 1.

This reframes binomial coefficients as counting *maps* rather than subsets.

Worked examples

Example 6.2. How many solutions are there to $x_1 + x_2 + x_3 = 7$ with $x_i \geq 0$?

Solution. Using stars and bars: we have 7 stars and need 2 bars to separate into 3 groups. Total positions: $7 + 2 = 9$. Choose 2 positions for bars: $\binom{9}{2} = \frac{9 \cdot 8}{2} = 36$.

Example 6.3. Prove $\sum_{k=0}^{n} \binom{n}{k} = 2^n$.

Proof 1 (Binomial theorem): Set $x = y = 1$ in $(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$.

Proof 2 (Combinatorial): LHS counts all subsets of an n-element set (choose 0, or 1, or 2, ..., or n elements). RHS: each element is either in or out, giving 2^n choices. □

Example 6.4. How many ways can the letters of MISSISSIPPI be arranged?

Solution. Total 11 letters: M(1), I(4), S(4), P(2). Using the multinomial coefficient: $\binom{11}{1,4,4,2} = \frac{11!}{1! \cdot 4! \cdot 4! \cdot 2!} = \frac{39916800}{1 \cdot 24 \cdot 24 \cdot 2} = 34650$.

Example 6.5. Prove Pascal's identity: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

Algebraic proof:

$$\begin{aligned}
\binom{n-1}{k-1} + \binom{n-1}{k} &= \frac{(n-1)!}{(k-1)!(n-k)!} + \frac{(n-1)!}{k!(n-k-1)!} \\
&= \frac{(n-1)! \cdot k + (n-1)! \cdot (n-k)}{k!(n-k)!} \\
&= \frac{(n-1)! \cdot n}{k!(n-k)!} = \frac{n!}{k!(n-k)!} = \binom{n}{k}
\end{aligned}$$

Example 6.6. Use the binomial theorem to expand $(2x - 1)^4$.

Solution. By the binomial theorem with $a = 2x$ and $b = -1$:

$$\begin{aligned}
(2x-1)^4 &= \sum_{k=0}^{4} \binom{4}{k} (2x)^{4-k} (-1)^k \\
&= \binom{4}{0}(2x)^4 - \binom{4}{1}(2x)^3 + \binom{4}{2}(2x)^2 - \binom{4}{3}(2x) + \binom{4}{4} \\
&= 1 \cdot 16x^4 - 4 \cdot 8x^3 + 6 \cdot 4x^2 - 4 \cdot 2x + 1 \\
&= 16x^4 - 32x^3 + 24x^2 - 8x + 1
\end{aligned}$$

Example 6.7. How many positive integer solutions are there to $x_1 + x_2 + x_3 = 10$?

Solution. Since we want *positive* integers ($x_i \geq 1$), substitute $y_i = x_i - 1$ so $y_i \geq 0$. Then:

$$(y_1 + 1) + (y_2 + 1) + (y_3 + 1) = 10 \implies y_1 + y_2 + y_3 = 7$$

Now we count non-negative integer solutions using stars and bars. We have 7 stars and need 2 bars to separate into 3 groups:

$$\binom{7+3-1}{3-1} = \binom{9}{2} = \frac{9 \cdot 8}{2} = 36$$

Example 6.8. A committee of 5 is chosen from 6 men and 4 women. How many committees have at least 2 women?

Solution. "At least 2 women" means 2, 3, or 4 women. Count each case:

- 2 women, 3 men: $\binom{4}{2}\binom{6}{3} = 6 \cdot 20 = 120$

- 3 women, 2 men: $\binom{4}{3}\binom{6}{2} = 4 \cdot 15 = 60$
- 4 women, 1 man: $\binom{4}{4}\binom{6}{1} = 1 \cdot 6 = 6$

Total: $120 + 60 + 6 = 186$.

Alternative (complement): Total committees: $\binom{10}{5} = 252$. Committees with fewer than 2 women:

- 0 women: $\binom{4}{0}\binom{6}{5} = 6$
- 1 woman: $\binom{4}{1}\binom{6}{4} = 4 \cdot 15 = 60$

Answer: $252 - 6 - 60 = 186$. ✓

Example 6.9. In how many ways can 10 identical apples be distributed among 4 children?

Solution. This is distributing $n = 10$ identical objects into $k = 4$ distinct bins. By stars and bars:

$$\binom{10+4-1}{4-1} = \binom{13}{3} = \frac{13 \cdot 12 \cdot 11}{3 \cdot 2 \cdot 1} = \frac{1716}{6} = 286$$

Example 6.10. Prove combinatorially: $\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n} = 2^n$.

Proof. The LHS counts subsets of an n-element set by size: there are $\binom{n}{k}$ subsets of size k.

The RHS counts subsets directly: each of the n elements is either in or out of the subset, giving 2^n choices.

Both count the same thing (total number of subsets), so they're equal. □

Practice

1. Compute $\binom{12}{5}$.
2. How many 8-card poker hands contain exactly 3 hearts?
3. Prove Pascal's identity: $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.
4. Use the binomial theorem to expand $(2x-1)^5$.
5. How many positive integer solutions are there to $x_1 + x_2 + x_3 + x_4 = 15$?
6. Prove: $\binom{n}{0}^2 + \binom{n}{1}^2 + \cdots + \binom{n}{n}^2 = \binom{2n}{n}$.
7. A committee of 5 is to be chosen from 6 men and 4 women. How many committees have at least 2 women?
8. Solve the recurrence $a_n = 3a_{n-1} - 2$ with $a_0 = 1$.
9. Solve the recurrence $a_n = 4a_{n-1} - 4a_{n-2}$ with $a_0 = 1$, $a_1 = 2$.
10. Let t_n be the number of ways to tile a $2 \times n$ board with dominoes. Write a recurrence for t_n with initial conditions and compute t_5.
11. Find the ordinary generating function for the sequence $a_n = n$ for $n \geq 0$.

7 Week 6: Expected Value and Introduction to Graphs

Reading

Epp §9.8; 10.1; 10.2.
Category theory companion: Weeks 6–7 (`category_theory_companion.pdf`).

Learning objectives

- Define and compute expected value for discrete random variables.
- Apply linearity of expectation to simplify calculations.
- Use indicator random variables for counting.
- Define graphs, vertices, edges, and basic terminology.
- Apply the handshake theorem to relate degrees and edges.
- Distinguish simple graphs, multigraphs, and digraphs.

Part I: Expected Value

Definition 7.1 (Probability axioms (Kolmogorov))**.** A **probability measure** on a sample space S is a function P assigning to each event $A \subseteq S$ a number $P(A)$ satisfying:

1. **Non-negativity:** $P(A) \geq 0$ for all events A.
2. **Normalization:** $P(S) = 1$.
3. **Additivity:** If $A_1, A_2, \ldots$ are pairwise disjoint events, then

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$$

Proposition 7.1 (Consequences of the axioms)**.** *From the three axioms, we can derive:*

- $P(\emptyset) = 0$
- $P(A^c) = 1 - P(A)$
- *If* $A \subseteq B$*, then* $P(A) \leq P(B)$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $0 \leq P(A) \leq 1$ *for all events* A

Definition 7.2 (Random variable)**.** A **random variable** X on a sample space S is a function $X : S \to \mathbb{R}$ that assigns a real number to each outcome. For discrete random variables, the possible values form a finite or countably infinite set.

Definition 7.3 (Expected value)**.** The **expected value** (or **expectation** or **mean**) of a discrete random variable X with possible values $x_1, x_2, \ldots$ and probabilities $p_i = P(X = x_i)$ is:

$$E[X] = \sum_i x_i \cdot P(X = x_i) = \sum_i x_i \cdot p_i$$

provided the sum converges absolutely.

Theorem 7.1 (Linearity of expectation)**.** *For any random variables X and Y (even if dependent) and constants $a, b \in \mathbb{R}$:*

$$E[aX + bY] = aE[X] + bE[Y]$$

More generally, for any $X_1, \ldots, X_n$:

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]$$

Key Result

Linearity of expectation is extremely powerful because it works *regardless of whether the random variables are independent.* This makes many expected value calculations surprisingly simple.

Definition 7.4 (Indicator random variable)**.** An **indicator random variable** I_A for event A is:

$$I_A = \begin{cases} 1 & \text{if } A \text{ occurs} \\ 0 & \text{otherwise} \end{cases}$$

Note that $E[I_A] = P(A)$.

Theorem 7.2 (Counting with indicators)**.** *If X counts the number of events $A_1, \ldots, A_n$ that occur, then:*

$$X = I_{A_1} + I_{A_2} + \cdots + I_{A_n}$$

and by linearity:

$$E[X] = P(A_1) + P(A_2) + \cdots + P(A_n)$$

Definition 7.5 (Common distributions)**.**

- **Bernoulli(p):** $X = 1$ with probability p, $X = 0$ with probability $1 - p$. $E[X] = p$.
- **Binomial(n, p):** Number of successes in n independent trials, each with success probability p. $E[X] = np$.
- **Geometric(p):** Number of trials until first success. $E[X] = 1/p$.
- **Uniform on $\{1, \ldots, n\}$:** Each value equally likely. $E[X] = (n + 1)/2$.

Definition 7.6 (Variance and standard deviation)**.** The **variance** of X is:

$$\mathrm{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2$$

The **standard deviation** is $\sigma = \sqrt{\mathrm{Var}(X)}$.

Part II: Introduction to Graphs

Definition 7.7 (Graph)**.** A **graph** $G = (V, E)$ consists of:

- V: a finite nonempty set of **vertices** (or nodes)
- E: a set of **edges**, each connecting two vertices

Definition 7.8 (Types of graphs)**.**
- **Simple graph:** No loops (edges from a vertex to itself) and no multiple edges between the same pair of vertices.
- **Multigraph:** Allows multiple edges between the same pair of vertices.
- **Pseudograph:** Allows loops and multiple edges.
- **Directed graph (digraph):** Edges have direction, going from one vertex to another.

Definition 7.9 (Basic terminology)**.** Let $G = (V, E)$ be a graph.

- Two vertices are **adjacent** if an edge connects them.
- An edge is **incident** to its endpoints.
- The **degree** $\deg(v)$ of vertex v is the number of edges incident to v (loops count twice).
- A vertex with degree 0 is **isolated**.
- A vertex with degree 1 is a **leaf** (or pendant vertex).
- The **neighborhood** $N(v)$ is the set of vertices adjacent to v.

Theorem 7.3 (Handshake theorem)**.** *In any graph $G = (V, E)$:*

$$\sum_{v \in V} \deg(v) = 2|E|$$

Proof idea: *Each edge contributes exactly 2 to the sum of degrees (1 to each endpoint).* □

Corollary 7.1. *In any graph, the number of vertices with odd degree is even.*

Definition 7.10 (Special graphs)**.**
- **Complete graph K_n:** Simple graph on n vertices with all possible edges. Has $\binom{n}{2} = \frac{n(n-1)}{2}$ edges.
- **Cycle C_n:** Graph on n vertices forming a single cycle. Has n edges.
- **Path P_n:** Graph on n vertices forming a single path. Has $n - 1$ edges.
- **Complete bipartite graph $K_{m,n}$:** Vertices partitioned into sets of sizes m and n; every vertex in one set is adjacent to every vertex in the other. Has mn edges.
- **n-cube Q_n:** Vertices are n-bit strings; edges connect strings differing in exactly one bit. Has 2^n vertices and $n \cdot 2^{n-1}$ edges.

Definition 7.11 (Degree sequence)**.** The **degree sequence** of a graph is the list of vertex degrees in non-increasing order. For example, K_4 has degree sequence $(3, 3, 3, 3)$.

Theorem 7.4 (Degree sequence realizability)**.** *A sequence of non-negative integers* $(d_1, d_2, \ldots, d_n)$ *with* $d_1 \geq d_2 \geq \cdots \geq d_n$ *is the degree sequence of a simple graph if and only if:*

1. *The sum* $\sum d_i$ *is even.*
2. *The sequence satisfies the Erdős–Gallai conditions (or can be checked using the Havel–Hakimi algorithm).*

Definition 7.12 (Subgraph)**.** $H = (V', E')$ is a **subgraph** of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. H is an **induced subgraph** if E' contains all edges of G whose endpoints are both in V'.

Definition 7.13 (Graph complement)**.** The **complement** $\overline{G}$ of a simple graph $G = (V, E)$ has the same vertices as G, and two vertices are adjacent in $\overline{G}$ iff they are not adjacent in G.

Category Lens: Parts and Graphs

This week connects probability and graphs to two categorical ideas: *parts* and *structure-preserving maps*. For more detail, see the Category Theory Companion, Weeks 6–7.

Parts and Predicates

Events are subsets of a sample space. In category language, subsets of A are *parts* of A and correspond to characteristic functions $A \to 2$. If $f : A \to B$, the preimage map $f^{-1} : \mathcal{P}(B) \to \mathcal{P}(A)$ is *contravariant* and preserves unions and intersections. This is a categorical way to explain why pulling back events behaves nicely under functions.

Graphs as a Category

There is a category **Graph** whose objects are graphs and whose morphisms are **graph homomorphisms** (vertex maps that preserve adjacency). There is a forgetful functor U : **Graph** $\to$ **Set** sending a graph to its vertex set. This lets us reuse set-based reasoning while respecting graph structure.

Worked examples

Example 7.1. A fair die is rolled. Let X be the outcome. Compute $E[X]$.

Solution. Each outcome $1, 2, 3, 4, 5, 6$ has probability $\frac{1}{6}$.

$$E[X] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} = \frac{21}{6} = 3.5$$

Example 7.2. A coin is flipped 10 times. What is the expected number of heads?

Solution. Let $X_i = 1$ if flip i is heads, 0 otherwise. Then $X = X_1 + \cdots + X_{10}$ counts heads.

By linearity: $E[X] = E[X_1] + \cdots + E[X_{10}] = 10 \cdot \frac{1}{2} = 5$.

Example 7.3. In a random permutation of n elements, what is the expected number of fixed points (elements in their original position)?

Solution. Let $X_i = 1$ if element i is in position i. Then $X = \sum_{i=1}^{n} X_i$ counts fixed points.

$P(\text{element } i \text{ is in position } i) = \frac{1}{n}$ (any of the $n!$ permutations, the element has $\frac{(n-1)!}{n!} = \frac{1}{n}$ chance of being fixed).

By linearity: $E[X] = n \cdot \frac{1}{n} = 1$.

Remarkably, the expected number of fixed points is exactly 1, regardless of n!

Example 7.4. What is the expected number of times we must roll a die to get a 6?

Solution. This is a geometric random variable with success probability $p = \frac{1}{6}$.
$E[X] = \frac{1}{p} = 6$.

Example 7.5. Verify the handshake theorem for K_4.

Solution. K_4 has 4 vertices, each with degree 3 (connected to all others).

- Sum of degrees: $3 + 3 + 3 + 3 = 12$
- Number of edges: $\binom{4}{2} = 6$
- Check: $2 \times 6 = 12$ ✓

Example 7.6. Is there a simple graph with degree sequence $(3, 3, 2, 2, 2)$?

Solution. Sum of degrees: $3 + 3 + 2 + 2 + 2 = 12$, which is even. ✓

Using Havel–Hakimi: Sort: $(3, 3, 2, 2, 2)$. Remove 3 and subtract 1 from next 3 degrees: $(2, 1, 1, 2)$. Sort: $(2, 2, 1, 1)$. Remove 2: $(1, 0, 1)$. Sort: $(1, 1, 0)$. Remove 1: $(0, 0)$. This is realizable (empty graph).

Yes, such a graph exists.

Example 7.7. How many edges does the n-cube Q_n have?

Solution. Q_n has 2^n vertices, each an n-bit string. Each vertex has degree n (can flip any of n bits).

Sum of degrees: $n \cdot 2^n$.

By handshake theorem: $|E| = \frac{n \cdot 2^n}{2} = n \cdot 2^{n-1}$.

Example 7.8. Prove: The sum of degrees in a tree on n vertices is $2(n - 1)$.

Solution. A tree on n vertices has exactly $n-1$ edges (this is a standard fact—see Week 8). By the handshake theorem:

$$\sum_{v \in V} \deg(v) = 2|E| = 2(n - 1)$$

Example 7.9. Show that every simple graph on $n \geq 2$ vertices has at least two vertices of the same degree.

Solution. Degrees in a simple graph range from 0 to $n - 1$. That's n possible values. But if some vertex has degree 0 (isolated), no vertex can have degree $n-1$ (connected to all). So at most $n - 1$ distinct degrees are possible among n vertices. By pigeonhole, two must share a degree.

Common Mistake

Forgetting linearity works for dependent variables. The formula $E[X+Y] = E[X] + E[Y]$ does NOT require X and Y to be independent. Many students add independence as an assumption when it's unnecessary.

Common Mistake

Confusing $E[X \cdot Y]$ with $E[X] \cdot E[Y]$. These are equal only when X and Y are independent. In general, $E[XY] = E[X]E[Y] + \text{Cov}(X, Y)$.

Practice

1. A coin is flipped 10 times. What is the expected number of heads?
2. Show that the sum of degrees in a tree on n vertices is $2(n-1)$.
3. Find $E[X]$ for a geometric random variable with success probability p.
4. Decide whether a graph with degree sequence $(3, 3, 2, 2, 2)$ is possible.
5. In a random permutation of $\{1, 2, \ldots, n\}$, what is the expected number of elements greater than all previous elements?
6. How many edges does $K_{4,5}$ have? What are the degrees of the vertices?
7. Prove that the complement of K_n is an empty graph (no edges).
8. A bag contains 5 red and 3 blue marbles. Two are drawn without replacement. What is the expected number of red marbles drawn?
9. Show that the number of edges in a simple graph on n vertices is at most $\binom{n}{2}$.
10. Using the handshake theorem, prove: If G is a graph where every vertex has degree at least k, then $|E| \geq \frac{k|V|}{2}$.
11. Prove that every graph has an even number of vertices with odd degree.
12. In a room of 100 people, everyone shakes hands with exactly 3 other people. Is this possible?

8 Week 7: Graph Theory I — Paths and Connectivity

Reading

Epp §10.1–10.3.
Category theory companion: Weeks 6–7 (`category_theory_companion.pdf`).

Learning objectives

- Distinguish walks, trails, paths, and circuits.
- Apply Euler's criteria for trails and circuits.
- Determine graph connectivity and connected components.
- Represent graphs with adjacency matrices and adjacency lists.
- Determine whether two graphs are isomorphic.

Key definitions and facts

Definition 8.1 (Walk)**.** A **walk** in a graph G from vertex v_0 to vertex v_n is a sequence:

$$v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$$

where each e_i is an edge connecting v_{i-1} and v_i. The **length** of the walk is n (the number of edges).

Definition 8.2 (Types of walks)**.**
- A **trail** is a walk with no repeated edges.
- A **path** is a walk with no repeated vertices (hence no repeated edges).
- A **closed walk** is a walk where $v_0 = v_n$.
- A **circuit** (or closed trail) is a closed walk with no repeated edges.
- A **cycle** (or simple circuit) is a circuit with no repeated vertices except $v_0 = v_n$.

Proposition 8.1 (Path existence)**.** *If there is a walk from u to v in a graph, then there is a path from u to v.*

Definition 8.3 (Connectivity)**.**
- A graph is **connected** if there is a path between every pair of vertices.
- A **connected component** of a graph is a maximal connected subgraph.
- A **cut vertex** (or articulation point) is a vertex whose removal disconnects the graph.
- A **bridge** is an edge whose removal disconnects the graph.

Definition 8.4 (Euler trail and circuit)**.** An **Euler trail** is a trail that uses every edge of the graph exactly once. An **Euler circuit** is a circuit that uses every edge exactly once (starts and ends at the same vertex).

Theorem 8.1 (Euler's theorem)**.** *Let G be a connected graph.*

1. *G has an **Euler circuit** if and only if every vertex has even degree.*
2. *G has an **Euler trail** (but no Euler circuit) if and only if exactly two vertices have odd degree. In this case, the trail must start and end at the odd-degree vertices.*

Key Result

To check if a connected graph has an Euler circuit or trail:

1. Count vertices of odd degree.
2. 0 odd-degree vertices $\Rightarrow$ Euler circuit exists.
3. 2 odd-degree vertices $\Rightarrow$ Euler trail exists (but no circuit).
4. > 2 odd-degree vertices $\Rightarrow$ no Euler trail.

Definition 8.5 (Hamiltonian path and cycle)**.** A **Hamiltonian path** visits every vertex exactly once. A **Hamiltonian cycle** is a cycle that visits every vertex exactly once (except returning to start).

Remark. Unlike Euler paths/circuits, there is no simple characterization for when Hamiltonian paths/cycles exist. Determining existence is NP-complete.

Graph representations

Definition 8.6 (Adjacency matrix)**.** The **adjacency matrix** A of a graph G with n vertices is an $n \times n$ matrix where:

$$A_{ij} = \text{number of edges between vertex } i \text{ and vertex } j$$

For a simple graph, $A_{ij} \in \{0, 1\}$. The matrix is symmetric for undirected graphs.

Proposition 8.2 (Properties of adjacency matrices)**.** *For the adjacency matrix A of a simple graph:*

- *The sum of row i (or column i) equals* $\deg(v_i)$*.*
- *The sum of all entries equals* $2|E|$*.*
- *The diagonal is all zeros (no loops).*
- $(A^k)_{ij}$ *counts the number of walks of length k from v_i to v_j.*

Definition 8.7 (Adjacency list)**.** An **adjacency list** representation stores, for each vertex, a list of its neighbors. This is more space-efficient for sparse graphs.

Graph isomorphism

Definition 8.8 (Graph isomorphism)**.** Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic**, written $G_1 \cong G_2$, if there exists a bijection $f : V_1 \to V_2$ such that:

$$\{u, v\} \in E_1 \iff \{f(u), f(v)\} \in E_2$$

The function f is called an **isomorphism**.

Theorem 8.2 (Isomorphism invariants)**.** *If $G_1 \cong G_2$, then:*

1. $|V_1| = |V_2|$
2. $|E_1| = |E_2|$

3. They have the same degree sequence

4. They have the same number of cycles of each length

5. They have the same number of connected components

6. Corresponding subgraphs are isomorphic

These are necessary *but not* sufficient *conditions for isomorphism.*

Proof Strategy

To show two graphs are NOT isomorphic, find an invariant they don't share. To show they ARE isomorphic, construct an explicit bijection and verify edge preservation.

Definition 8.9 (Automorphism)**.** An **automorphism** of a graph G is an isomorphism from G to itself. The set of all automorphisms forms a group under composition.

Distance and diameter

Definition 8.10 (Distance)**.** The **distance** $d(u, v)$ between vertices u and v is the length of the shortest path between them. If no path exists, $d(u, v) = \infty$.

Definition 8.11 (Eccentricity, radius, diameter)**.**

- The **eccentricity** of a vertex v is the maximum distance from v to any other vertex: $\max_u d(v, u)$.
- The **diameter** of a connected graph is the maximum eccentricity.
- The **radius** is the minimum eccentricity.
- A **center** is a vertex with minimum eccentricity.

Graph coloring

Definition 8.12 (Vertex coloring)**.** A **(proper) vertex coloring** of a graph G is an assignment of colors to vertices such that no two adjacent vertices share the same color. A **k-coloring** uses at most k colors.

Definition 8.13 (Chromatic number)**.** The **chromatic number** $\chi(G)$ is the minimum number of colors needed to properly color G.

Theorem 8.3 (Chromatic number bounds)**.** *For any graph G:*

1. $\chi(G) \geq \omega(G)$, where $\omega(G)$ is the size of the largest clique (complete subgraph).

2. $\chi(G) \leq \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree.

3. If G is connected and not a complete graph or odd cycle, then $\chi(G) \leq \Delta(G)$ (Brooks' theorem).

Theorem 8.4 (Chromatic numbers of special graphs)**.**

- *$\chi(K_n) = n$ (complete graph needs n colors)*
- *$\chi(C_n) = 2$ if n is even; $\chi(C_n) = 3$ if n is odd*
- *$\chi(K_{m,n}) = 2$ (bipartite graphs are 2-colorable)*

- *A tree with at least one edge has $\chi(T) = 2$*

Definition 8.14 (Bipartite graph)**.** A graph is **bipartite** if its vertices can be partitioned into two sets such that every edge connects a vertex in one set to a vertex in the other. Equivalently, G is bipartite iff $\chi(G) \leq 2$.

Theorem 8.5 (Bipartite characterization)**.** *A graph is bipartite if and only if it contains no odd-length cycle.*

Example 8.1. Is the Petersen graph 3-colorable?

Solution. The Petersen graph contains triangles (3-cycles), so $\chi \geq 3$. In fact, $\chi(\text{Petersen}) = 3$. You can verify by constructing a 3-coloring: color the outer 5-cycle with alternating colors (using 3 since it's odd), then color the inner 5-cycle consistently.

Planar graphs

Definition 8.15 (Planar graph)**.** A graph is **planar** if it can be drawn in the plane with no edges crossing (except at vertices). Such a drawing is called a **planar embedding**.

Definition 8.16 (Faces)**.** In a planar embedding, the plane is divided into **faces** (regions), including one unbounded **outer face**. The boundary of each face consists of edges and vertices.

Theorem 8.6 (Euler's formula for planar graphs)**.** *For a connected planar graph with V vertices, E edges, and F faces:*

$$V - E + F = 2$$

Example 8.2. Verify Euler's formula for the tetrahedron graph K_4.

Solution. K_4 has $V = 4$ vertices and $E = \binom{4}{2} = 6$ edges. Drawing it as a triangle with a point in the center gives $F = 4$ faces (3 inner triangles + 1 outer face).

Check: $4 - 6 + 4 = 2$. ✓

Theorem 8.7 (Edge bound for planar graphs)**.** *For a connected planar graph with $V \geq 3$ vertices:*

$$E \leq 3V - 6$$

If the graph has no triangles (is triangle-free), then $E \leq 2V - 4$.

Corollary 8.1. *K_5 and $K_{3,3}$ are not planar.*

Proof. For K_5: $V = 5$, $E = 10$. But $3V - 6 = 9 < 10$. Violates the bound.

For $K_{3,3}$: $V = 6$, $E = 9$. Since $K_{3,3}$ is bipartite, it has no triangles, so we need $E \leq 2V - 4 = 8 < 9$. Violates the bound. ☐

Theorem 8.8 (Kuratowski's theorem)**.** *A graph is planar if and only if it contains no subgraph that is a subdivision of K_5 or $K_{3,3}$.*

*(A **subdivision** is obtained by inserting vertices of degree 2 into edges.)*

Theorem 8.9 (Four Color Theorem)**.** *Every planar graph can be colored with at most 4 colors: $\chi(G) \leq 4$ for planar G.*

Remark. The Four Color Theorem was proved in 1976 using computer assistance to check thousands of cases. Simpler proofs exist but none are "hand-checkable."

Example 8.3. Show that the cube graph Q_3 is planar.

Solution. Q_3 has $V = 8$ vertices and $E = 12$ edges. Check: $3V - 6 = 18 \geq 12$. ✓(This doesn't prove planarity, but it's consistent.)

To prove planarity, we draw Q_3 without crossings: draw the outer square as the front face, the inner square as the back face, and connect corresponding vertices.

Proof Strategy

To show a graph is NOT planar:

1. Show it violates $E \leq 3V - 6$, or
2. Find a K_5 or $K_{3,3}$ subdivision.

To show a graph IS planar:

1. Draw it without crossings, or
2. Prove V and E satisfy the bounds (necessary but not sufficient).

Worked examples

Example 8.4. Does a connected graph with degrees $(2, 2, 2, 4, 4)$ have an Euler circuit?
Solution. All degrees are even (2, 2, 2, 4, 4), so yes, an Euler circuit exists by Euler's theorem.

Example 8.5. Does the graph K_4 (complete graph on 4 vertices) have an Euler circuit?
Solution. In K_4, each vertex has degree 3 (odd). All 4 vertices have odd degree. Since we need 0 or 2 vertices of odd degree for an Euler trail/circuit, K_4 has neither.

Example 8.6. Does the graph K_5 have an Euler circuit?
Solution. In K_5, each vertex has degree 4 (even). All vertices have even degree, so K_5 has an Euler circuit.

Example 8.7. Find the adjacency matrix for the cycle C_4 on vertices $\{1, 2, 3, 4\}$.
Solution. The edges are $\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}$.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Example 8.8. Show that the sum of entries in an adjacency matrix of a simple graph equals $2|E|$.
Solution. Each edge $\{u, v\}$ contributes 1 to entry (u, v) and 1 to entry (v, u), for a total of 2 per edge. Thus the sum equals $2|E|$.

Example 8.9. Determine whether these two graphs are isomorphic:
G_1: vertices $\{a, b, c, d\}$, edges $\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}$
G_2: vertices $\{1, 2, 3, 4\}$, edges $\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}$
Solution. Both are cycles of length 4.

- Same number of vertices: 4 ✓
- Same number of edges: 4 ✓
- Same degree sequence: $(2, 2, 2, 2)$ ✓

The bijection $f : a \mapsto 1, b \mapsto 2, c \mapsto 3, d \mapsto 4$ preserves edges: $\{a, b\} \mapsto \{1, 2\}$, $\{b, c\} \mapsto \{2, 3\}$, $\{c, d\} \mapsto \{3, 4\}$, $\{d, a\} \mapsto \{4, 1\}$. All edges match, so $G_1 \cong G_2$.

Example 8.10. Prove that C_5 and K_5 are not isomorphic.

Solution. C_5 has 5 edges (a cycle). K_5 has $\binom{5}{2} = 10$ edges. Since they have different numbers of edges, they are not isomorphic.

Example 8.11. Are two graphs with the same degree sequence necessarily isomorphic?

Solution. No! Consider:

- G_1: a 6-cycle C_6. Degree sequence: $(2, 2, 2, 2, 2, 2)$.
- G_2: two disjoint triangles $K_3 \sqcup K_3$. Degree sequence: $(2, 2, 2, 2, 2, 2)$.

Same degree sequence, but G_1 is connected and G_2 is not. Not isomorphic.

Example 8.12. Find the diameter of the complete graph K_n.

Solution. Every pair of vertices is connected by an edge, so $d(u, v) = 1$ for all $u \neq v$. The diameter is 1.

Example 8.13. Find an Euler trail in a graph with vertices $\{A, B, C, D\}$ and edges

$$\{A, B\}, \{A, C\}, \{B, C\}, \{B, D\}, \{C, D\}.$$

Solution. First, check degrees: $\deg(A) = 2$, $\deg(B) = 3$, $\deg(C) = 3$, $\deg(D) = 2$.

Odd-degree vertices: B and C (exactly 2). So an Euler trail exists, starting and ending at B and C.

One Euler trail starting at B: $B \to A \to C \to B \to D \to C$.

Verify: Uses edges $\{B, A\}, \{A, C\}, \{C, B\}, \{B, D\}, \{D, C\}$ — all 5 edges, each exactly once. ✓

Example 8.14. Compute A^2 for the path graph P_3 on vertices $\{1, 2, 3\}$ with edges $\{1, 2\}$ and $\{2, 3\}$. Interpret the result.

Solution.

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$A^2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Interpretation: $(A^2)_{ij}$ is the number of walks of length 2 from i to j.

- $(A^2)_{11} = 1$: one walk $1 \to 2 \to 1$.
- $(A^2)_{13} = 1$: one walk $1 \to 2 \to 3$.
- $(A^2)_{22} = 2$: two walks $2 \to 1 \to 2$ and $2 \to 3 \to 2$.

Common Mistake

Confusing Euler and Hamiltonian.

- Euler: visits every *edge* exactly once.
- Hamiltonian: visits every *vertex* exactly once.

Euler has a simple characterization (degree conditions). Hamiltonian does not.

Common Mistake

Thinking matching invariants proves isomorphism. Equal vertex count, edge count, and degree sequence are *necessary* but not *sufficient* for isomorphism. You must construct a bijection or find a distinguishing property.

Going Deeper: Graphs Generate Categories

The categorical thread continues: graphs give rise to categories, and this perspective illuminates why adjacency matrices count paths. For more detail, see the Category Theory Companion, Weeks 6–7.

The Free Category on a Graph

Given a directed graph G, we can build a category $\mathbf{Path}(G)$:

- **Objects:** Vertices of G
- **Morphisms from u to v:** Directed paths from u to v
- **Composition:** Concatenation of paths
- **Identity at v:** The empty path (length 0) at v

This is called the *free category* on G—it's the category with "just enough structure" to capture the graph.
Example. For the graph $1 \to 2 \to 3$:

- Morphisms $1 \to 3$: just the path $1 \to 2 \to 3$ (one morphism)
- Morphisms $2 \to 2$: just the empty path id_2 (one morphism)
- Morphisms $3 \to 1$: none (no path backwards)

Example. For a cycle $1 \to 2 \to 3 \to 1$:

- Morphisms $1 \to 1$: empty path, $1 \to 2 \to 3 \to 1$, twice around, thrice around, ...
- This is infinite! The cycle generates infinitely many paths.

Adjacency Matrices Count Morphisms

Here's the key insight: $(A^k)_{ij}$ counts the number of **morphisms of length** k from i to j in $\mathbf{Path}(G)$.
Why? Let's see for $k = 2$:

$$(A^2)_{ij} = \sum_m A_{im} \cdot A_{mj}$$

Each term $A_{im} \cdot A_{mj}$ counts paths that go $i \to m \to j$ (one for each intermediate vertex m with edges from i and to j).
This is exactly the **composition** of morphisms in $\mathbf{Path}(G)$!

Composition as Matrix Multiplication

The correspondence is:

Category	**Matrix**
Composition of paths	Matrix multiplication
Length-k paths	A^k
Identity (length-0 path)	I (identity matrix)

Connected Components as a Functor

There is a functor $\pi_0 : \mathbf{Graph} \to \mathbf{Set}$ that sends a graph to its set of connected components. A graph homomorphism $f : G \to H$ induces a function $\pi_0(f)$ by sending each component of G to the component of H containing its image.
This functor captures the idea that "connectivity" is preserved by structure-preserving maps.

Quotient Categories: Imposing Relations

What if we want to declare two paths equal? For example, in a commutative square:

$$\begin{tikzcd}
1 \arrow[r, "a"] \arrow[d, "c"'] & 2 \arrow[d, "b"] \\
3 \arrow[r, "d"'] & 4
\end{tikzcd}$$

In $\mathbf{Path}(G)$, the paths $b \circ a$ and $d \circ c$ are different morphisms. But if we impose the relation $b \circ a = d \circ c$, we get a *quotient category* where these paths are identified.
This is how commutative diagrams work: they specify which paths should be considered equal.

Exercises: Graphs and Paths

1. For the graph $1 \to 2 \to 3$, list all morphisms in $\mathbf{Path}(G)$ from each vertex to each vertex.
2. For the graph with edges $1 \to 2$, $2 \to 3$, $3 \to 1$, how many morphisms of length 3 are there from 1 to 1? Verify using A^3.
3. For a graph with edges $a \to b$, $b \to c$, $a \to c$, are there two different morphisms from a to c? In the *quotient* category where we impose $c \circ b \circ a^{-1} = \text{id}$... wait, we can't do that without inverses. Just count: how many distinct paths from a to c?
4. Write the adjacency matrix for the 3-cycle. Compute A^2 and verify that $(A^2)_{11}$ equals the number of length-2 paths from 1 to 1.
5. A graph has edges $1 \to 2$, $2 \to 1$ (a 2-cycle). How many morphisms of length 4 are there from 1 to 1? Compute using A^4.
6. **Challenge:** If G is a graph with no directed cycles, prove that $\mathbf{Path}(G)$ has finitely many morphisms between any two vertices. (Hint: What bounds the length of paths?)

7. Draw a directed graph whose path category has exactly 3 morphisms from vertex a to vertex b.
8. In the path category, explain why composition is associative and why the empty path is the identity.

Practice

1. Give the adjacency matrix for the 4-cycle C_4.
2. Determine whether the two graphs below are isomorphic (construct your own example).
3. Find an Euler trail in a graph with exactly two odd-degree vertices.
4. Show that the sum of entries in an adjacency matrix equals $2|E|$.
5. Prove: If G is a simple graph and $\overline{G}$ is its complement, then $G \cong \overline{G}$ implies $|V| \equiv 0$ or $1 \pmod 4$.
6. Compute A^2 for K_3 and interpret the entries.
7. Prove that every connected graph on n vertices has at least $n-1$ edges.
8. Find the diameter of the n-cube Q_n.
9. Does $K_{3,3}$ (complete bipartite graph) have an Euler circuit? An Euler trail?
10. Prove that a graph is bipartite if and only if it contains no odd-length cycles.
11. How many automorphisms does the cycle C_n have?
12. Prove: If G is connected and has exactly 2 vertices of odd degree, any Euler trail must start and end at those vertices.
13. Find $\chi(C_7)$ and $\chi(C_8)$.
14. Find the chromatic number of the wheel graph W_5 (a 5-cycle with a central vertex connected to all).
15. Use Euler's formula to find the number of faces in a connected planar graph with 10 vertices and 15 edges.
16. Prove that every planar graph has a vertex of degree at most 5.
17. Is the Petersen graph planar? Prove your answer.
18. A planar graph has 12 faces, and each face is bounded by exactly 3 edges. How many edges and vertices does it have?
19. Give a 3-coloring of the graph K_4 minus one edge.
20. Prove: If G is planar with no cycles of length ≤ 4, then $E \leq \frac{5}{3}(V-2)$.

9 Week 8: Trees and Graph Algorithms

Reading

Epp §10.4–10.6. Supplemental: matchings and flows.
Category theory companion: Week 8 (`category_theory_companion.pdf`).

Learning objectives

- Identify trees, forests, and rooted trees.
- Use characterizations of trees (connected + acyclic, $|E| = |V| - 1$, etc.).
- Understand m-ary trees and binary trees.
- Find spanning trees using BFS and DFS.
- Apply shortest-path algorithms (Dijkstra's, Bellman-Ford).
- Find minimum spanning trees (Prim's, Kruskal's).

Key definitions and facts

Definition 9.1 (Tree)**.** A **tree** is a connected graph with no cycles. A **forest** is a graph with no cycles (each connected component is a tree).

Theorem 9.1 (Characterizations of trees)**.** *For a graph G on n vertices, the following are equivalent:*

1. *G is a tree (connected and acyclic).*
2. *G is connected and has exactly $n - 1$ edges.*
3. *G is acyclic and has exactly $n - 1$ edges.*
4. *There is exactly one path between any two vertices.*
5. *G is connected, but removing any edge disconnects it.*
6. *G is acyclic, but adding any edge creates exactly one cycle.*

Definition 9.2 (Rooted tree)**.** A **rooted tree** is a tree with a designated vertex called the **root**. This induces a parent-child relationship: every non-root vertex has a unique parent (the neighbor closer to the root) and zero or more children.

- The **depth** of a vertex is its distance from the root.
- The **height** of the tree is the maximum depth.
- A **leaf** is a vertex with no children.
- An **internal vertex** has at least one child.

Definition 9.3 (m-ary tree)**.** An **m-ary tree** is a rooted tree where every internal vertex has at most m children.

- A **binary tree** is a 2-ary tree.
- A **full m-ary tree** has every internal vertex with exactly m children.
- A **complete m-ary tree** is a full m-ary tree where all leaves are at the same depth.

Theorem 9.2 (Properties of full m-ary trees)**.** *For a full m-ary tree with i internal vertices:*

1. *Total vertices:* $n = mi + 1$
2. *Leaves:* $\ell = (m-1)i + 1 = \frac{(m-1)n+1}{m}$
3. *Internal vertices:* $i = \frac{n-1}{m} = \frac{\ell-1}{m-1}$

Theorem 9.3 (Height bounds for binary trees)**.** *A binary tree with ℓ leaves has height h satisfying:*

$$\lceil \log_2 \ell \rceil \leq h \leq \ell - 1$$

The minimum height is achieved by a complete binary tree; the maximum by a "linear" tree.

Definition 9.4 (Spanning tree)**.** A **spanning tree** of a connected graph G is a subgraph that is a tree containing all vertices of G.

Theorem 9.4 (Existence of spanning trees)**.** *Every connected graph has a spanning tree. (Proof: Remove edges from cycles until no cycles remain.)*

Tree traversals

Definition 9.5 (Binary tree traversals)**.** For a binary tree with root r, left subtree L, and right subtree R:

- **Preorder:** Visit r, traverse L, traverse R
- **Inorder:** Traverse L, visit r, traverse R
- **Postorder:** Traverse L, traverse R, visit r

Definition 9.6 (BFS and DFS)**.**

- **Breadth-First Search (BFS):** Explore vertices layer by layer (by distance from start). Uses a queue.
- **Depth-First Search (DFS):** Explore as deep as possible before backtracking. Uses a stack (or recursion).

Both produce spanning trees of connected graphs.

Shortest path algorithms

Definition 9.7 (Weighted graph)**.** A **weighted graph** assigns a weight $w(e)$ to each edge e. The weight of a path is the sum of its edge weights.

Definition 9.8 (Shortest path problem)**.** Given a weighted graph and vertices s and t, find a path from s to t with minimum total weight.

Theorem 9.5 (Dijkstra's algorithm)**.** *For a graph with non-negative edge weights, Dijkstra's algorithm finds shortest paths from a source vertex to all other vertices.*

Idea: *Maintain a set S of vertices with known shortest distances. Repeatedly add the unvisited vertex with smallest tentative distance, updating neighbors.*

Time complexity: $O((|V| + |E|) \log |V|)$ *with a priority queue.*

Theorem 9.6 (Bellman-Ford algorithm)**.** *For graphs that may have negative edge weights (but no negative cycles), Bellman-Ford finds shortest paths from a source.*
   Idea: *Relax all edges $|V| - 1$ times.*
   Time complexity: $O(|V| \cdot |E|)$.

Theorem 9.7 (Floyd-Warshall algorithm)**.** *Finds shortest paths between all pairs of vertices.*
   Idea: *Dynamic programming on intermediate vertices.*
   Time complexity: $O(|V|^3)$.

Minimum spanning trees

Definition 9.9 (Minimum spanning tree (MST))**.** For a connected weighted graph, a **minimum spanning tree** is a spanning tree with minimum total edge weight.

Theorem 9.8 (Cut property)**.** *For any cut (partition of vertices into two sets), the minimum-weight edge crossing the cut is in some MST.*

Theorem 9.9 (Prim's algorithm)**.** *Starting from any vertex, repeatedly add the minimum-weight edge connecting the tree to a new vertex.*
   Time complexity: $O((|V| + |E|) \log |V|)$ *with a priority queue.*

Theorem 9.10 (Kruskal's algorithm)**.** *Sort edges by weight. Add edges in order, skipping those that would create a cycle.*
   Time complexity: $O(|E| \log |E|)$ *(dominated by sorting).*

Matchings in bipartite graphs

Definition 9.10 (Matching)**.** A **matching** is a set of edges with no shared endpoints. A **maximum matching** has the largest possible size. A **perfect matching** covers every vertex.

Theorem 9.11 (Hall's marriage theorem)**.** *Let $G = (X \cup Y, E)$ be a bipartite graph. There is a matching that covers all vertices of X iff for every subset $S \subseteq X$:*

$$|N(S)| \geq |S|$$

where $N(S)$ is the set of neighbors of S in Y.

Example 9.1. Let $X = \{1, 2, 3\}$, $Y = \{a, b, c\}$, and edges $\{1a, 1b, 2b, 3b, 3c\}$. For any $S \subseteq X$:

$$|N(\{1\})| = 2,\ |N(\{2\})| = 1,\ |N(\{3\})| = 2,\ |N(\{2,3\})| = 2,\ |N(\{1,2,3\})| = 3$$

Hall's condition holds, so a matching covering X exists (e.g., $\{1a, 2b, 3c\}$).

Remark. Maximum matchings can be found by **augmenting paths**: repeatedly flip edges along a path that alternates between unmatched and matched edges to increase the matching size.

Network flows

Definition 9.11 (Flow network)**.** A **flow network** is a directed graph with a source s, sink t, and nonnegative capacities $c(u, v)$ on edges. A flow assigns values $f(u, v)$ satisfying:

- **Capacity constraint:** $0 \leq f(u, v) \leq c(u, v)$

- **Flow conservation:** For all $v \neq s, t$, $\sum_u f(u,v) = \sum_w f(v,w)$

The **value** of the flow is the total outflow from s.

Definition 9.12 (Cut)**.** An s-t **cut** is a partition (S,T) with $s \in S$ and $t \in T$. Its **capacity** is $\sum_{u \in S, v \in T} c(u,v)$.

Theorem 9.12 (Max-flow min-cut)**.** *The maximum value of an s-t flow equals the minimum capacity of any s-t cut.*

Remark. The Ford–Fulkerson method repeatedly finds an augmenting path in the residual graph until no such path remains.

Worked examples

Example 9.2. Prove that a tree on n vertices has exactly $n-1$ edges.

Proof (by induction).

- **Base case:** $n = 1$. A single vertex has 0 edges. $0 = 1 - 1$. ✓
- **Inductive step:** Assume true for trees with k vertices. Consider a tree T with $k+1$ vertices.

  A tree has at least one leaf (vertex of degree 1). Remove a leaf v and its incident edge. The result is a tree T' with k vertices.

  By the inductive hypothesis, T' has $k-1$ edges. Adding back the one edge to v, we get k edges for T.

  $k = (k+1) - 1$. ✓

Example 9.3. A full binary tree has 15 vertices. How many are leaves?

Solution. For a full binary tree ($m = 2$): $\ell = \frac{(m-1)n+1}{m} = \frac{(2-1)\cdot 15+1}{2} = \frac{16}{2} = 8$ leaves.

Alternatively: If there are i internal vertices and ℓ leaves, then $n = i + \ell$ and for full binary trees, $\ell = i + 1$. So $15 = i + (i+1) = 2i + 1$, giving $i = 7$ and $\ell = 8$.

Example 9.4. How many leaves can a full m-ary tree of height h have at most?

Solution. A complete m-ary tree of height h has leaves only at depth h. At depth d, there are at most m^d vertices. So the maximum number of leaves is m^h.

Example 9.5. Find a spanning tree of K_4.

Solution. K_4 has 4 vertices and 6 edges. A spanning tree needs 3 edges. Remove any 3 edges that don't disconnect the graph.

For example, with vertices $\{1,2,3,4\}$, keep edges $\{1,2\}, \{2,3\}, \{3,4\}$. This is the path $1-2-3-4$, which is a spanning tree.

Example 9.6. Explain why removing any edge from a tree disconnects it.

Solution. In a tree, there is exactly one path between any two vertices. An edge $\{u,v\}$ is on the unique path from u to v. Removing it eliminates this path, and since there was only one path, u and v become disconnected.

Example 9.7. Run Dijkstra's algorithm on a simple weighted graph.

Consider vertices $\{A,B,C,D\}$ with weighted edges: $A-B$ (1), $A-C$ (4), $B-C$ (2), $B-D$ (5), $C-D$ (1). Find shortest paths from A.

Solution.

1. Initialize: $d[A] = 0$, $d[B] = d[C] = d[D] = \infty$.
2. Process A: Update $d[B] = 1$, $d[C] = 4$.
3. Process B (smallest tentative): Update $d[C] = \min(4, 1 + 2) = 3$, $d[D] = \min(\infty, 1 + 5) = 6$.
4. Process C: Update $d[D] = \min(6, 3 + 1) = 4$.
5. Process D: No updates.

Shortest distances: $d[A] = 0$, $d[B] = 1$, $d[C] = 3$, $d[D] = 4$.

Example 9.8. Use Kruskal's algorithm to find an MST.

Consider vertices $\{A, B, C, D\}$ with edges: $A - B$ (3), $A - C$ (1), $A - D$ (4), $B - C$ (2), $B - D$ (5), $C - D$ (6).

Solution.

1. Sort edges: $A - C$ (1), $B - C$ (2), $A - B$ (3), $A - D$ (4), $B - D$ (5), $C - D$ (6).
2. Add $A - C$ (1): No cycle. Tree: $\{A - C\}$.
3. Add $B - C$ (2): No cycle. Tree: $\{A - C, B - C\}$.
4. Add $A - B$ (3): Would create cycle $A - B - C - A$. Skip.
5. Add $A - D$ (4): No cycle. Tree: $\{A - C, B - C, A - D\}$.

MST edges: $\{A - C, B - C, A - D\}$ with total weight $1 + 2 + 4 = 7$.

Example 9.9. Give the preorder, inorder, and postorder traversals of a binary tree.

Consider a binary tree:

```
    A
   / \
  B   C
 / \
D   E
```

Solution.

- **Preorder** (root, left, right): A, B, D, E, C
- **Inorder** (left, root, right): D, B, E, A, C
- **Postorder** (left, right, root): D, E, B, C, A

Example 9.10. Prove that a forest with n vertices and k connected components has $n - k$ edges.

Solution. Each connected component is a tree. If component i has n_i vertices, it has $n_i - 1$ edges.

Total edges: $\sum_{i=1}^{k}(n_i - 1) = \sum_{i=1}^{k} n_i - k = n - k$.

Going Deeper: Trees as Initial Algebras and the Universality of Fold

There's a deep reason why recursive functions on trees always terminate, and why the "fold" pattern is so universal. Trees are *initial algebras*, and fold is the unique map from an initial object. For more detail, see the Category Theory Companion, Week 8.

The recursive structure of trees. Notice that a binary tree is either:

- A leaf (containing data), or
- A node with a left subtree and a right subtree

Writing this as an equation: $\text{Tree}(A) = A + \text{Tree}(A) \times \text{Tree}(A)$.

Here $+$ means "or" (disjoint union) and $\times$ means "and" (Cartesian product). This recursive equation *defines* the type of trees.

Algebras for tree-building. An "algebra" for this structure over a set B consists of:

- A function $\mathit{leaf} : A \to B$ (what to do with leaves)
- A function $\mathit{node} : B \times B \to B$ (how to combine subtree results)

The fold (catamorphism). Given any algebra $(\mathit{leaf}, \mathit{node})$ over B, there is a *unique* function $\mathit{fold} : \text{Tree}(A) \to B$ satisfying:

$$\mathit{fold}(\text{Leaf } a) = \mathit{leaf}(a) \qquad \mathit{fold}(\text{Node } \ell\, r) = \mathit{node}(\mathit{fold}(\ell), \mathit{fold}(r))$$

Examples of fold.

- **Sum leaves:** $\mathit{leaf}(a) = a$, $\mathit{node}(x, y) = x + y$
- **Count leaves:** $\mathit{leaf}(a) = 1$, $\mathit{node}(x, y) = x + y$
- **Tree height:** $\mathit{leaf}(a) = 0$, $\mathit{node}(x, y) = 1 + \max(x, y)$
- **Preorder list:** $\mathit{leaf}(a) = [a]$, $\mathit{node}(x, y) = x \mathbin{+\!\!+} y$

All tree traversals from this section (preorder, inorder, postorder) are folds with appropriate algebra choices!

Why recursion terminates. The tree type is the *initial* algebra—the "smallest" solution to the recursive equation. The uniqueness of fold means there's exactly one way to recursively compute any result. If recursion didn't terminate, no function would exist (violating existence). If there were multiple ways to compute, uniqueness would fail.

The pattern generalizes. Lists are also an initial algebra: $\text{List}(A) = 1 + A \times \text{List}(A)$. The fold for lists is:

$$\mathit{foldr}(f, z, []) = z \qquad \mathit{foldr}(f, z, x : xs) = f(x, \mathit{foldr}(f, z, xs))$$

Natural numbers are too: $\mathbb{N} = 1 + \mathbb{N}$ with "algebra" (z, s) giving primitive recursion.

Why this matters: Understanding data structures as initial algebras explains why structural recursion is well-founded, enables powerful optimizations ("fold fusion"), and connects programming to deep mathematics.

Exercises: Folds and Algebras

1. The "sum" function on lists adds up all elements. What is the base case (what does the empty list [] map to)? What is the combining function?
2. The "length" function counts elements. What is the base case? What is the combining function?
3. The "product" function multiplies all elements. Define it as a fold.
4. Define `map f` as a fold. What is the base case? What is the combining function? (Hint: the result type is also a list.)
5. For natural numbers with Zero and Succ: define addition $n+m$ by fixing m and folding over n. What algebra (z, s) do you use?
6. Define multiplication $n \times m$ as a fold over n, with m fixed.
7. For binary trees with data at leaves, define "count leaves" as a fold. Specify *leaf* and *node*.
8. Define "tree height" as a fold on binary trees.
9. **Uniqueness:** Suppose sum_1 and sum_2 both satisfy:

$$\begin{aligned} sum_1([]) &= 0 & \qquad sum_1(x : xs) &= x + sum_1(xs) \\ sum_2([]) &= 0 & \qquad sum_2(x : xs) &= x + sum_2(xs) \end{aligned}$$

   Prove that $sum_1 = sum_2$ by showing they agree on all lists.
10. Express preorder traversal of a binary tree as a fold. What about inorder? (Hint: for inorder, think about where to put the current node's data relative to left and right results.)
11. Why can't we fold over an *infinite* list to get a finite answer (in general)? What would go wrong with the uniqueness argument?
12. **Challenge:** The "fold fusion" law says: if $h \circ g = g' \circ F(h)$ in a suitable sense, then $h \circ fold_g = fold_{g'}$. For lists, this gives: if $h(z) = z'$ and $h(f(x, y)) = f'(x, h(y))$ for all x, y, then $h \circ foldr(f, z) = foldr(f', z')$. Verify this for $h(n) = 2n$, $f(x, y) = x + y$, $z = 0$.

Common Mistake

Applying Dijkstra's algorithm with negative weights. Dijkstra assumes non-negative weights. With negative edges, use Bellman-Ford instead.

Common Mistake

Confusing "connected and acyclic" with "exactly $n-1$ edges." Both characterize trees, but having $n-1$ edges alone does NOT guarantee a tree—the graph must also be connected (for trees) or acyclic (for forests).

Practice

1. How many leaves can a full m-ary tree of height h have?
2. Find a spanning tree of the complete graph K_5.
3. Explain why removing any edge from a tree disconnects it.
4. Run Dijkstra's algorithm on a weighted graph with 5 vertices of your choice.
5. A full binary tree has 31 vertices. How many are leaves? How many are internal?
6. Prove: Every tree with at least 2 vertices has at least 2 leaves.
7. Use Prim's algorithm to find an MST of a weighted graph (create your own example).
8. Prove that a tree is bipartite.
9. Give the BFS and DFS spanning trees of a 4-cycle C_4 starting from vertex 1.
10. Prove: If a graph has n vertices and fewer than $n-1$ edges, it is not connected.
11. How many spanning trees does the cycle C_n have?
12. Prove: In any tree, the sum of all vertex degrees equals $2(n-1)$.
13. Let $X = \{1, 2, 3\}$ and $Y = \{a, b, c\}$ with edges $\{1a, 1b, 2b, 2c, 3c\}$. Use Hall's theorem to decide whether there is a matching that covers X, and find one if it exists.
14. Find a maximum matching in the bipartite graph with $X = \{u, v, w\}$, $Y = \{x, y, z\}$, and edges $\{ux, uy, vx, vy, wz\}$.
15. Compute the maximum flow value in a network with edges $s \to a$ (3), $s \to b$ (2), $a \to b$ (1), $a \to t$ (2), $b \to t$ (3). Give one maximum flow and a minimum cut.

10 Week 9: Regular Expressions and Finite Automata

Reading

Epp §12.1–12.3. Supplemental: CFGs and PDAs.
Category theory companion: Week 9 (`category_theory_companion.pdf`).

Learning objectives

- Define languages, alphabets, and strings.
- Construct regular expressions to describe languages.
- Build deterministic finite automata (DFAs) and trace their execution.
- Convert between regular expressions and DFAs.
- Minimize DFAs by merging equivalent states.
- Understand the pumping lemma for proving non-regularity.

Key definitions and facts

Definition 10.1 (Alphabet, string, language).

- An **alphabet** Σ is a finite, nonempty set of symbols.
- A **string** (or word) over Σ is a finite sequence of symbols from Σ.
- The **empty string** ε has length 0.
- The set of all strings over Σ is denoted Σ^*.
- A **language** over Σ is a subset $L \subseteq \Sigma^*$.

Definition 10.2 (String operations).

- **Length:** $|w|$ is the number of symbols in w.
- **Concatenation:** $w_1 w_2$ appends w_2 to w_1. Note: $w\varepsilon = \varepsilon w = w$.
- **Exponentiation:** $w^n = \underbrace{ww\cdots w}_{n\text{ times}}$; $w^0 = \varepsilon$.
- **Reversal:** w^R is w written backwards.

Definition 10.3 (Language operations). For languages $L, L_1, L_2 \subseteq \Sigma^*$:

- **Union:** $L_1 \cup L_2 = \{w : w \in L_1 \text{ or } w \in L_2\}$
- **Concatenation:** $L_1 L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$
- **Kleene star:** $L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup \cdots = \bigcup_{n\geq 0} L^n$
- **Kleene plus:** $L^+ = L \cup L^2 \cup L^3 \cup \cdots = LL^*$

Regular expressions

Definition 10.4 (Regular expression)**.** A **regular expression** (regex) over alphabet Σ is defined recursively:

1. $\emptyset$ is a regex denoting the empty language $\{\}$.
2. ε is a regex denoting the language $\{\varepsilon\}$.
3. For each $a \in \Sigma$, a is a regex denoting $\{a\}$.
4. If r_1 and r_2 are regexes, then:
    - $(r_1 \mid r_2)$ denotes $L(r_1) \cup L(r_2)$ (union/alternation)
    - $(r_1 r_2)$ denotes $L(r_1)L(r_2)$ (concatenation)
    - $(r_1)^*$ denotes $L(r_1)^*$ (Kleene star)

Definition 10.5 (Precedence)**.** Operator precedence (highest to lowest): Kleene star $*$, concatenation, union $|$.

So $ab^* \mid c$ means $(a(b^*)) \mid c$, not $a(b^* \mid c)$ or $(ab)^* \mid c$.

Example 10.1 (Common regex patterns)**.** Over $\Sigma = \{0, 1\}$:

- All strings: $(0 \mid 1)^*$
- Strings starting with 1: $1(0 \mid 1)^*$
- Strings ending with 01: $(0 \mid 1)^*01$
- Strings with exactly one 1: 0^*10^*
- Strings with at least one 0: $(0 \mid 1)^*0(0 \mid 1)^*$
- Even-length strings: $((0 \mid 1)(0 \mid 1))^*$

Deterministic finite automata

Definition 10.6 (DFA)**.** A **deterministic finite automaton** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite set of **states**
- Σ is the input **alphabet**
- $\delta : Q \times \Sigma \to Q$ is the **transition function**
- $q_0 \in Q$ is the **start state**
- $F \subseteq Q$ is the set of **accept (final) states**

Definition 10.7 (DFA execution)**.** A DFA **accepts** a string $w = a_1 a_2 \cdots a_n$ if there exists a sequence of states $r_0, r_1, \ldots, r_n$ such that:

1. $r_0 = q_0$ (start in the start state)
2. $r_{i+1} = \delta(r_i, a_{i+1})$ for each i (follow transitions)

3. $r_n \in F$ (end in an accept state)

The language of a DFA M, denoted $L(M)$, is the set of all strings it accepts.

Definition 10.8 (State diagram)**.** A DFA can be represented as a directed graph:

- Vertices are states
- An edge from q to q' labeled a indicates $\delta(q, a) = q'$
- The start state has an incoming arrow from nowhere
- Accept states are drawn with a double circle

Nondeterministic finite automata

Definition 10.9 (NFA)**.** A **nondeterministic finite automaton** (NFA) is like a DFA, but:

- The transition function is $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$
- From a state, there can be 0, 1, or many transitions on the same symbol
- ε-transitions allow changing state without consuming input

An NFA accepts if *some* path leads to an accept state.

Theorem 10.1 (NFA-DFA equivalence)**.** *For every NFA, there exists a DFA that accepts the same language. The subset construction converts an NFA with n states to a DFA with at most 2^n states.*

Regular languages

Definition 10.10 (Regular language)**.** A language L is **regular** if it is recognized by some DFA (equivalently, by some NFA, or described by some regex).

Theorem 10.2 (Kleene's theorem)**.** *The following are equivalent for a language L:*

1. *L is described by a regular expression.*
2. *L is recognized by a DFA.*
3. *L is recognized by an NFA.*

Theorem 10.3 (Closure properties)**.** *Regular languages are closed under:*

- *Union, concatenation, Kleene star (by definition)*
- *Complement: If L is regular, so is $\Sigma^* \setminus L$ (swap accept/non-accept states in DFA)*
- *Intersection: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ (De Morgan)*
- *Reversal: If L is regular, so is $L^R = \{w^R : w \in L\}$*

DFA minimization

Definition 10.11 (Equivalent states)**.** Two states p and q in a DFA are **equivalent** if for all strings $w \in \Sigma^*$:

$$\hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F$$

where $\hat{\delta}$ is the extended transition function.

Theorem 10.4 (Minimization)**.** *Every regular language has a unique minimum-state DFA (up to isomorphism).*
It is obtained by merging equivalent states.

Definition 10.12 (Table-filling algorithm)**.** To find equivalent states:

1. Mark all pairs (p, q) where exactly one is in F as distinguishable.
2. Repeat: Mark (p, q) as distinguishable if for some $a \in \Sigma$, $(\delta(p, a), \delta(q, a))$ is distinguishable.
3. Unmarked pairs are equivalent; merge them.

Non-regular languages

Theorem 10.5 (Pumping lemma for regular languages)**.** *If L is regular, then there exists a "pumping length" p such that any string $w \in L$ with $|w| \geq p$ can be written as $w = xyz$ where:*

1. $|y| > 0$ *(the pump is non-empty)*
2. $|xy| \leq p$ *(the pump is near the start)*
3. *For all $i \geq 0$, $xy^iz \in L$ (pumping preserves membership)*

Proof Strategy

To prove a language L is *not* regular using the pumping lemma:

1. Assume L is regular (for contradiction).
2. Let p be the pumping length.
3. Choose a string $w \in L$ with $|w| \geq p$ (often depending on p).
4. Show that no matter how w is split as xyz (satisfying conditions 1 and 2), there exists i such that $xy^iz \notin L$.
5. Contradiction: L is not regular.

Context-free grammars

Definition 10.13 (Context-free grammar)**.** A CFG is a 4-tuple (V, Σ, R, S) where:

- V is a set of variables (nonterminals)
- Σ is a set of terminals (alphabet)
- R is a set of production rules of the form $A \to \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$

- $S \in V$ is the start symbol

The language $L(G)$ is the set of strings in Σ^* derivable from S.

Definition 10.14 (Derivation and parse tree)**.** We write $S \Rightarrow^* w$ if w can be derived from S by repeatedly applying production rules. A **parse tree** records the rule applications that produce w.

Example 10.2. The grammar with rules $S \to aSb \mid \varepsilon$ generates $L = \{a^n b^n : n \geq 0\}$. For example, $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$.

Remark. A grammar is **ambiguous** if some string has two different parse trees. Ambiguity complicates parsing and language reasoning.

Pushdown automata

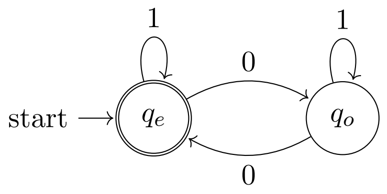
Definition 10.15 (Pushdown automaton (PDA))**.** A **pushdown automaton** is like an NFA with a stack. Formally it is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Γ is the stack alphabet and δ reads an input symbol (or ε) and the top of the stack, then updates the state and stack.

Theorem 10.6 (CFG-PDA equivalence)**.** *A language is context-free iff it is recognized by a pushdown automaton.*

Worked examples

Example 10.3. Design a DFA over $\{0, 1\}$ that accepts strings with an even number of 0s.
Solution. Two states: q_e (even 0s so far) and q_o (odd 0s so far).

- Start state: q_e (zero 0s is even)
- Accept state: $\{q_e\}$
- Transitions: On 0, toggle between q_e and q_o. On 1, stay in current state.



Example 10.4. Write a regex for all binary strings that end with 01.
Solution. (0 | 1)*01
Any sequence of 0s and 1s, followed by 01.

Example 10.5. Construct a DFA for strings over $\{a, b\}$ containing no substring bb.
Solution. Three states tracking what we've seen at the end:

- q_0: Start, or last symbol was a (no recent b)
- q_1: Last symbol was b
- q_{dead}: Saw bb, reject
- From q_0: On a, stay in q_0. On b, go to q_1.

- From q_1: On a, go to q_0. On b, go to q_{dead}.
- From q_{dead}: Stay in q_{dead} on any input.
- Accept states: $\{q_0, q_1\}$

Example 10.6. Prove that the intersection of two regular languages is regular.

Solution. Let L_1 be recognized by DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and let L_2 be recognized by $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct the product DFA $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$ where:

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

This DFA accepts w iff both M_1 and M_2 accept w, so $L(M) = L_1 \cap L_2$.

Example 10.7. Minimize the following DFA over $\{0, 1\}$.

States: $\{A, B, C, D\}$. Start: A. Accept: $\{C\}$. Transitions: $\delta(A, 0) = B$, $\delta(A, 1) = C$, $\delta(B, 0) = B$, $\delta(B, 1) = C$, $\delta(C, 0) = D$, $\delta(C, 1) = C$, $\delta(D, 0) = D$, $\delta(D, 1) = C$.

Solution. Using the table-filling algorithm:

1. Mark $(A, C), (B, C), (D, C)$ (one accept, one non-accept).
2. Check remaining pairs:
   - (A, B): $\delta(A, 0) = B$, $\delta(B, 0) = B$ — same. $\delta(A, 1) = C$, $\delta(B, 1) = C$ — same. Not distinguishable yet.
   - (A, D): $\delta(A, 0) = B$, $\delta(D, 0) = D$. Check (B, D) first.
   - (B, D): $\delta(B, 0) = B$, $\delta(D, 0) = D$ — need to check (B, D). $\delta(B, 1) = C$, $\delta(D, 1) = C$ — same. Not distinguishable.
3. States A, B, D are equivalent. Merge them into one state.

Minimal DFA has 2 states: $\{A, B, D\}$ and $\{C\}$.

Example 10.8. Prove that $L = \{0^n 1^n : n \geq 0\}$ is not regular using the pumping lemma.

Proof. Assume L is regular. Let p be the pumping length.

Choose $w = 0^p 1^p \in L$. Then $|w| = 2p \geq p$.

By the pumping lemma, $w = xyz$ where $|y| > 0$, $|xy| \leq p$, and $xy^i z \in L$ for all i.

Since $|xy| \leq p$ and w starts with p zeros, xy consists only of 0s. So $y = 0^k$ for some $k > 0$.

Consider $i = 2$: $xy^2 z = 0^{p+k} 1^p$. Since $k > 0$, this has more 0s than 1s, so $xy^2 z \notin L$.

Contradiction. Therefore L is not regular. □

Example 10.9. Write a regular expression for all binary strings with at least two 0s.

Solution. We need at least two 0s, with any number of 0s and 1s before, between, and after them:

$$(0 \mid 1)^* 0 (0 \mid 1)^* 0 (0 \mid 1)^*$$

Equivalently, using 1^* instead of $(0 \mid 1)^*$ where appropriate: 1*01*0$(0 \mid 1)^*$ (but the first form is more symmetric).

Example 10.10. Design a DFA over $\{a, b\}$ that accepts strings where the number of as is divisible by 3.

Solution. Track (number of as) $\bmod 3$. Three states: q_0 (seen $0 \bmod 3$), q_1 (seen $1 \bmod 3$), q_2 (seen $2 \bmod 3$).

- Start state: q_0 (zero as)
- Accept state: $\{q_0\}$ (divisible by 3)
- Transitions on a: $q_0 \to q_1 \to q_2 \to q_0$ (cycle)
- Transitions on b: self-loops (don't change count)

Formally: $\delta(q_i, a) = q_{(i+1) \bmod 3}$ and $\delta(q_i, b) = q_i$.

Example 10.11. Prove that $L = \{ww : w \in \{0,1\}^*\}$ is not regular.

Proof. Assume L is regular with pumping length p.

Choose $s = 0^p10^p1 \in L$ (where $w = 0^p1$). We have $|s| = 2p + 2 \geq p$.

By the pumping lemma, $s = xyz$ with $|y| > 0$, $|xy| \leq p$.

Since $|xy| \leq p$ and s starts with p zeros, we have $y = 0^k$ for some $k \geq 1$.

Consider $xy^0z = 0^{p-k}10^p1$. The first half has $p-k$ zeros before its first 1, while the second half has p zeros before its first 1. Since $k \geq 1$, these halves are different, so $xy^0z \notin L$.

Contradiction. Therefore L is not regular. □

Example 10.12. Design a DFA for binary strings representing numbers divisible by 3 (reading left to right, most significant bit first).

Solution. Track the value $\bmod 3$ as we read. If current value is v and we read bit b, new value is $2v + b \pmod 3$.

States: q_0, q_1, q_2 representing value $\bmod 3$.

- Start: q_0 (value 0)
- Accept: $\{q_0\}$
- From q_0: on 0, $2 \cdot 0 + 0 = 0 \to q_0$; on 1, $2 \cdot 0 + 1 = 1 \to q_1$
- From q_1: on 0, $2 \cdot 1 + 0 = 2 \to q_2$; on 1, $2 \cdot 1 + 1 = 3 \equiv 0 \to q_0$
- From q_2: on 0, $2 \cdot 2 + 0 = 4 \equiv 1 \to q_1$; on 1, $2 \cdot 2 + 1 = 5 \equiv 2 \to q_2$

Test: $110_2 = 6$. Path: $q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_0$. Accept. ✓

Common Mistake

Confusing $\emptyset$ and ε. $\emptyset$ is the regex for the empty language (no strings). ε is the regex for the language containing only the empty string $\{\varepsilon\}$.

Common Mistake

Forgetting that the pumping lemma is only for proving non-regularity. You cannot use it to prove a language IS regular. Satisfying the pumping lemma is necessary but not sufficient for regularity.

Going Deeper: Duality and Observation

Week 8 introduced *initial algebras* for understanding recursive data structures like trees. This week, we glimpse the dual concept: *coalgebras* for understanding systems with observable behavior, like automata. For more detail, see the Category Theory Companion, Week 9.

Algebras vs Coalgebras

Recall from Week 8: an algebra takes us *from* structure *to* a set:

- List algebra: $([], \text{cons})$ tells how to *build* lists
- Tree algebra: $(\text{leaf}, \text{node})$ tells how to *build* trees
- The arrow points "inward": $F(A) \to A$

A *coalgebra* reverses the direction—it tells us how to *observe* or *decompose*:

- The arrow points "outward": $A \to F(A)$
- Given a state, we observe something about it

DFAs as Coalgebras

A DFA has a transition function and acceptance condition. For each state q, we can observe:

1. Is q accepting? (Output: yes/no)
2. Where do we go on each input symbol? (Transitions)

This gives a function: state $\to$ (output $\times$ next-states).
Formally, a DFA over alphabet Σ is a coalgebra for the pattern:

$$Q \to \{0, 1\} \times Q^{\Sigma}$$

where Q^{Σ} means "a function from Σ to Q."

Automata as Actions of the Free Monoid

The set of all strings Σ^* is the **free monoid** on Σ. A DFA induces a transition action:

$$\delta^* : Q \times \Sigma^* \to Q$$

that applies a whole string at once. Categorically, this is a functor from the one-object category Σ^* to **Set**, sending the single object to Q and each word to its transition function. This makes automata into dynamical systems driven by strings.

Streams: Another Coalgebra

An infinite stream of values from A can be observed:

1. What's the first element? (The head)
2. What's the rest of the stream? (The tail)

This gives: $\text{Stream}(A) \to A \times \text{Stream}(A)$.
Contrast with lists: for lists, we have $1 + A \times \text{List}(A) \to \text{List}(A)$ (building up from [] and cons).
For streams, we have $\text{Stream}(A) \to A \times \text{Stream}(A)$ (tearing down into head and tail).

The Duality Pattern

	Algebra	**Coalgebra**
Structure map	$F(A) \to A$	$A \to F(A)$
Intuition	Constructors	Observers
Universal object	Initial (smallest)	Final (largest)
Universal map	Fold (catamorphism)	Unfold (anamorphism)
Data	Finite	Potentially infinite
Example	Lists, trees	Streams, automata

Bisimulation: When Are Two States "The Same"?

Two DFA states are *bisimilar* if no sequence of observations can distinguish them. This means:

- They give the same accept/reject output
- For each input symbol, they transition to bisimilar states

This is exactly what DFA minimization does: merge bisimilar states!

Exercises: Duality and Observation

1. For an infinite stream, what is $\text{head}(\text{tail}(\text{tail}([0, 1, 2, 3, \ldots])))$?
2. The Fibonacci stream is $[0, 1, 1, 2, 3, 5, 8, \ldots]$. If we define it by $\text{fib} = [0, 1] \mathbin{+\!\!+} \text{zipWith}(+, \text{fib}, \text{tail}(\text{fib}))$, what are the first 5 elements? (Trace through.)
3. For the DFA from the "even number of 0s" example, write out the coalgebra structure: for each state, give the pair $(\text{accept?}, \text{transitions})$.
4. Fill in this duality table:

Concept	**Algebra**	**Coalgebra**
Structure map	$F(A) \to A$	?
Constructors	Build up	?
Universal object	Initial	?
Universal map	Fold	?
Data type	Finite	?

5. Why can we have infinite streams but (in a total language) not "infinite lists"? (Hint: What goes wrong if we try to fold over an infinite structure?)
6. In the DFA minimization example, states A, B, D were merged because they were bisimilar. Verify that A and B are bisimilar by checking: (a) same accept status, (b) transitions lead to bisimilar states.
7. **Challenge:** The "Brzozowski derivative" of a language L with respect to symbol a is $\partial_a(L) = \{w : aw \in L\}$. Show that this is exactly the "observe a transition" operation in the coalgebra view of languages.

Practice

1. Write a regex for all binary strings that end with 01.
2. Construct a DFA for strings over $\{a, b\}$ that contain no substring bb.
3. Minimize a DFA with 4 states of your choosing.
4. Prove that the intersection of two regular languages is regular.
5. Write a regex for binary strings with at least two 0s.
6. Design a DFA that accepts strings over $\{a, b\}$ where the number of as is divisible by 3.
7. Prove that $L = \{w \in \{0, 1\}^* : w = w^R\}$ (palindromes) is not regular.
8. Convert the regex $(a \mid b)^*aba$ to an NFA.
9. Show that if L is regular, then $L^R = \{w^R : w \in L\}$ is regular.
10. Design a DFA for binary strings representing numbers divisible by 3.
11. Prove that $L = \{a^{n^2} : n \geq 0\}$ is not regular.
12. Given DFAs for L_1 and L_2, construct a DFA for $L_1 \setminus L_2$.
13. Give a CFG for $L = \{0^n 1^n : n \geq 0\}$ and derive the string 000111.
14. Show that the grammar with rules $S \to SS \mid a$ is ambiguous by finding two distinct parse trees for aa.
15. Describe (in words) a PDA that recognizes $L = \{a^n b^n : n \geq 0\}$, and trace its stack contents on input $aabb$.

11 Week 10: Analysis of Algorithm Efficiency

Reading

Epp §11.1–11.5.
Category theory companion: Week 10 (`category_theory_companion.pdf`).

Learning objectives

- Compare growth rates of functions using limits and dominance.
- Apply big-O, big-Ω, and big-Θ notation correctly.
- Analyze the time complexity of loops and nested loops.
- Solve recurrences using expansion, substitution, and the master theorem.
- Classify algorithms by their complexity class.

Key definitions and facts

Definition 11.1 (Asymptotic notation)**.** Let $f, g : \mathbb{N} \to \mathbb{R}^+$ be functions.
Big-O (upper bound): $f(n) = O(g(n))$ if there exist constants $c > 0$ and n_0 such that:

$$f(n) \leq c \cdot g(n) \quad \text{for all } n \geq n_0$$

Big-Omega (lower bound): $f(n) = \Omega(g(n))$ if there exist constants $c > 0$ and n_0 such that:

$$f(n) \geq c \cdot g(n) \quad \text{for all } n \geq n_0$$

Big-Theta (tight bound): $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Theorem 11.1 (Limit test for asymptotic notation)**.** *If* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = L$*, then:*

- $L = 0 \Rightarrow f(n) = O(g(n))$ *but* $f(n) \neq \Theta(g(n))$
- $0 < L < \infty \Rightarrow f(n) = \Theta(g(n))$
- $L = \infty \Rightarrow f(n) = \Omega(g(n))$ *but* $f(n) \neq O(g(n))$

Definition 11.2 (Little-o and little-omega)**. Little-o:** $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. This means f grows strictly slower than g.
Little-omega: $f(n) = \omega(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$. This means f grows strictly faster than g.

Theorem 11.2 (Properties of asymptotic notation)**.**
1. ***Transitivity:*** *If* $f = O(g)$ *and* $g = O(h)$*, then* $f = O(h)$.
2. ***Reflexivity:*** $f = O(f)$, $f = \Omega(f)$, $f = \Theta(f)$.
3. ***Symmetry:*** $f = \Theta(g)$ *iff* $g = \Theta(f)$.
4. ***Transpose symmetry:*** $f = O(g)$ *iff* $g = \Omega(f)$.
5. ***Sum rule:*** $O(f) + O(g) = O(\max(f, g))$.
6. ***Product rule:*** $O(f) \cdot O(g) = O(f \cdot g)$.
7. ***Constant factors:*** $O(cf) = O(f)$ *for any constant* $c > 0$.

Common complexity classes

Definition 11.3 (Growth rate hierarchy)**.** Listed from slowest to fastest growth:

$$O(1) \subset O(\log n) \subset O(\sqrt{n}) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset O(2^n) \subset O(n!)$$

Notation	**Name**	**Example**
$O(1)$	Constant	Array access
$O(\log n)$	Logarithmic	Binary search
$O(n)$	Linear	Linear search
$O(n \log n)$	Linearithmic	Merge sort
$O(n^2)$	Quadratic	Bubble sort
$O(n^3)$	Cubic	Matrix multiplication (naive)
$O(2^n)$	Exponential	Subset enumeration
$O(n!)$	Factorial	Permutation enumeration

Analyzing code

Theorem 11.3 (Loop analysis)**.**

- *A loop that runs n times with $O(1)$ body: $O(n)$*
- *Two nested loops, each running n times: $O(n^2)$*
- *Three nested loops, each running n times: $O(n^3)$*
- *A loop that halves the problem size each iteration: $O(\log n)$*

Proof Strategy

To analyze a loop:

1. Count how many times the loop body executes.
2. Multiply by the cost of one iteration.
3. For nested loops, multiply the counts of each level.

Theorem 11.4 (Summation formulas)**.**

$$\sum_{i=1}^{n} 1 = n$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$$

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1}-1}{r-1} = \Theta(r^n) \textit{ for } r > 1$$

$$\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n) \textit{ (harmonic series)}$$

Recurrence relations

Definition 11.4 (Recurrence relation)**.** A **recurrence relation** expresses $T(n)$ in terms of T applied to smaller inputs. Common form for divide-and-conquer:

$$T(n) = aT(n/b) + f(n)$$

where $a \geq 1$ is the number of subproblems, n/b is the subproblem size, and $f(n)$ is the work outside the recursive calls.

Theorem 11.5 (Master theorem)**.** *For recurrence $T(n) = aT(n/b) + f(n)$ where $a \geq 1$, $b > 1$:*
Let $c = \log_b a$. Compare $f(n)$ with n^c:
Case 1: *If $f(n) = O(n^{c-\epsilon})$ for some $\epsilon > 0$, then $T(n) = \Theta(n^c)$.*
Case 2: *If $f(n) = \Theta(n^c \log^k n)$ for some $k \geq 0$, then $T(n) = \Theta(n^c \log^{k+1} n)$.*
Case 3: *If $f(n) = \Omega(n^{c+\epsilon})$ for some $\epsilon > 0$ and $af(n/b) \leq kf(n)$ for some $k < 1$, then $T(n) = \Theta(f(n))$.*

Theorem 11.6 (Common recurrences)**.**

Recurrence	***Solution***	***Example***
$T(n) = T(n/2) + O(1)$	$O(\log n)$	*Binary search*
$T(n) = T(n-1) + O(1)$	$O(n)$	*Linear recursion*
$T(n) = T(n-1) + O(n)$	$O(n^2)$	*Selection sort*
$T(n) = 2T(n/2) + O(1)$	$O(n)$	*Tree traversal*
$T(n) = 2T(n/2) + O(n)$	$O(n \log n)$	*Merge sort*
$T(n) = 2T(n-1) + O(1)$	$O(2^n)$	*Fibonacci (naive)*

Solving recurrences

Proof Strategy

Method 1: Expansion (iteration)

1. Expand the recurrence several times.
2. Identify the pattern.
3. Sum the terms.

Method 2: Substitution (guess and verify)

1. Guess the form of the solution.
2. Use induction to verify.
3. Adjust constants as needed.

Method 3: Master theorem

1. Identify a, b, and $f(n)$.
2. Compute $c = \log_b a$.
3. Determine which case applies.

Best, worst, and average case

Definition 11.5 (Case analysis).

- **Worst case:** Maximum time over all inputs of size n.
- **Best case:** Minimum time over all inputs of size n.
- **Average case:** Expected time over a probability distribution on inputs.

Usually, we report worst-case complexity.

Complexity classes and reductions

Definition 11.6 (Decision problem). A **decision problem** has a yes/no answer for each input. Complexity classes are typically defined for decision problems.

Definition 11.7 (Classes P and NP).

- **P**: problems solvable in polynomial time by a deterministic algorithm.
- **NP**: problems whose solutions can be *verified* in polynomial time (equivalently, solvable by a nondeterministic polynomial-time algorithm).

Definition 11.8 (Reductions, NP-hard, NP-complete). A polynomial-time reduction from A to B (written $A \leq_p B$) transforms instances of A into instances of B preserving yes/no answers.

- B is **NP-hard** if every problem in NP reduces to B.
- B is **NP-complete** if $B \in \mathbf{NP}$ and B is NP-hard.

Proof Strategy

To prove a problem B is NP-complete:

1. Show $B \in \mathbf{NP}$ (solutions are verifiable in poly time).
2. Reduce a known NP-complete problem A to B in polynomial time.

Example 11.1 (Common NP-complete problems). SAT, 3-SAT, CLIQUE, VERTEX COVER, HAMILTONIAN CYCLE, and SUBSET SUM are all NP-complete.

Worked examples

Example 11.2. Show that $3n^2 + 5n + 7 = \Theta(n^2)$.

Solution.

Upper bound: For $n \geq 1$: $3n^2 + 5n + 7 \leq 3n^2 + 5n^2 + 7n^2 = 15n^2$. So $3n^2 + 5n + 7 = O(n^2)$ with $c = 15$, $n_0 = 1$.

Lower bound: For $n \geq 1$: $3n^2 + 5n + 7 \geq 3n^2$. So $3n^2 + 5n + 7 = \Omega(n^2)$ with $c = 3$, $n_0 = 1$.

Therefore $3n^2 + 5n + 7 = \Theta(n^2)$.

Example 11.3. Order the functions $n \log n$, $n^{1.5}$, 2^n, n^3 by growth rate.

Solution. Compare using limits:

- $\lim_{n\to\infty} \frac{n \log n}{n^{1.5}} = \lim_{n\to\infty} \frac{\log n}{\sqrt{n}} = 0$ (L'Hôpital's). So $n \log n = o(n^{1.5})$.
- $\lim_{n\to\infty} \frac{n^{1.5}}{n^3} = \lim_{n\to\infty} \frac{1}{n^{1.5}} = 0$. So $n^{1.5} = o(n^3)$.

- $\lim_{n\to\infty} \frac{n^3}{2^n} = 0$ (exponential dominates polynomial). So $n^3 = o(2^n)$.

Order (slowest to fastest): $n \log n \prec n^{1.5} \prec n^3 \prec 2^n$.

Example 11.4. Analyze the runtime of nested loops:

```
for i = 1 to n:
    for j = 1 to n:
        // O(1) operation
```

Solution. The inner loop runs n times for each of n iterations of the outer loop. Total iterations: $n \times n = n^2$. Each iteration is $O(1)$. Total: $O(n^2)$.

Example 11.5. Analyze the runtime:

```
for i = 1 to n:
    for j = 1 to i:
        // O(1) operation
```

Solution. The inner loop runs i times. Total iterations:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$$

Example 11.6. Solve the recurrence $T(n) = 2T(n/2) + n$ with $T(1) = 1$.
Solution (Master theorem). Here $a = 2$, $b = 2$, $f(n) = n$.
$c = \log_b a = \log_2 2 = 1$, so $n^c = n$.
Compare: $f(n) = n = \Theta(n^1) = \Theta(n^c \log^0 n)$.
This is Case 2 with $k = 0$: $T(n) = \Theta(n^c \log^{k+1} n) = \Theta(n \log n)$.

Example 11.7. Solve $T(n) = 2T(n/2) + n$ by expansion.
Solution.

$$\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2[2T(n/4) + n/2] + n = 4T(n/4) + 2n \\
&= 4[2T(n/8) + n/4] + 2n = 8T(n/8) + 3n \\
&\vdots \\
&= 2^k T(n/2^k) + kn
\end{aligned}$$

When $n/2^k = 1$, we have $k = \log_2 n$ and $T(1) = 1$:

$$T(n) = 2^{\log n} \cdot 1 + n \log n = n + n \log n = \Theta(n \log n)$$

Example 11.8. Analyze the runtime of binary search.
Solution. At each step, the search space is halved. If $T(n)$ is the time for a search in an array of size n:

$$T(n) = T(n/2) + O(1), \quad T(1) = O(1)$$

By Master theorem: $a = 1$, $b = 2$, $f(n) = O(1)$. $c = \log_2 1 = 0$, so $n^c = 1$. $f(n) = O(1) = \Theta(n^0)$. Case 2 with $k = 0$: $T(n) = \Theta(\log n)$.

Example 11.9. Show that $\log(n!) = \Theta(n \log n)$.

Solution.

Upper bound: $n! = 1 \cdot 2 \cdots n \leq n^n$, so $\log(n!) \leq n \log n$.

Lower bound: $n! = 1 \cdot 2 \cdots n \geq (n/2)^{n/2}$ (considering only the largest $n/2$ terms), so:

$$\log(n!) \geq \frac{n}{2} \log \frac{n}{2} = \frac{n}{2}(\log n - 1) = \Omega(n \log n)$$

Therefore $\log(n!) = \Theta(n \log n)$.

Example 11.10. Prove that $\log n = O(n^\epsilon)$ for any $\epsilon > 0$.

Proof. Use the limit test:

$$\lim_{n\to\infty} \frac{\log n}{n^\epsilon}$$

This is an ∞/∞ form, so apply L'Hôpital's rule:

$$\lim_{n\to\infty} \frac{1/n}{\epsilon n^{\epsilon-1}} = \lim_{n\to\infty} \frac{1}{\epsilon n^\epsilon} = 0$$

Since the limit is 0, we have $\log n = O(n^\epsilon)$. In fact, $\log n = o(n^\epsilon)$, meaning log grows strictly slower than any positive power of n.

Example 11.11. Show that $2^{n+1} = \Theta(2^n)$ but $2^{2n} \neq O(2^n)$.

Solution.

Part 1: $2^{n+1} = 2 \cdot 2^n$. So $2^{n+1} \leq 2 \cdot 2^n$ (with $c = 2$) and $2^{n+1} \geq 2 \cdot 2^n$ (with $c = 2$). Thus $2^{n+1} = \Theta(2^n)$.

Part 2: $2^{2n} = (2^2)^n = 4^n$. Check if $4^n = O(2^n)$:

$$\lim_{n\to\infty} \frac{4^n}{2^n} = \lim_{n\to\infty} 2^n = \infty$$

Since the limit is ∞, 4^n grows faster than 2^n, so $2^{2n} \neq O(2^n)$.

Intuition: Constant factors in the exponent matter! 2^{cn} for $c > 1$ is exponentially larger than 2^n.

Example 11.12. Solve $T(n) = 3T(n/2) + n$ using the Master theorem.

Solution. Identify: $a = 3$, $b = 2$, $f(n) = n$.

Compute $c = \log_b a = \log_2 3 \approx 1.585$.

Compare $f(n) = n = n^1$ with $n^c = n^{1.585}$.

Since $1 < 1.585$, we have $f(n) = O(n^{c-\epsilon})$ for $\epsilon = 0.585$.

This is **Case 1**: $T(n) = \Theta(n^c) = \Theta(n^{\log_2 3})$.

Example 11.13. Analyze the time complexity of this code:

```
for i = 1 to n:
    for j = i to n:
        // O(1) work
```

Solution. The inner loop runs $n - i + 1$ times for each i. Total iterations:

$$\sum_{i=1}^{n}(n - i + 1) = \sum_{k=1}^{n} k = \frac{n(n+1)}{2} = \Theta(n^2)$$

(Substituted $k = n - i + 1$.)

Example 11.14. Solve the recurrence $T(n) = T(n-1) + n$ by expansion.

Solution. Expand:

$$\begin{aligned}
T(n) &= T(n-1) + n \\
&= T(n-2) + (n-1) + n \\
&= T(n-3) + (n-2) + (n-1) + n \\
&\vdots \\
&= T(1) + 2 + 3 + \cdots + n \\
&= T(1) + \sum_{k=2}^{n} k = T(1) + \frac{n(n+1)}{2} - 1
\end{aligned}$$

If $T(1) = 1$: $T(n) = \frac{n(n+1)}{2} = \Theta(n^2)$.

Common Mistake

Treating big-O as equality. $f = O(g)$ means f is bounded above by g, not equal to it. Better to read as "f is $O(g)$" rather than "f equals $O(g)$."

Common Mistake

Confusing worst-case and big-O. Big-O describes an upper bound on a function. Worst-case describes the maximum over inputs. They're related but distinct concepts.

Common Mistake

Ignoring the regularity condition in Master theorem Case 3. Case 3 requires $af(n/b) \leq kf(n)$ for some $k < 1$. This is usually satisfied but should be checked.

Going Deeper: Monoids—Categories with One Object

Throughout this course, we've seen how categories unify different areas of mathematics. We end with a beautiful observation: *monoids are categories with exactly one object.* For more detail, see the Category Theory Companion, Week 10.

Monoids: The Definition

A **monoid** $(M, \cdot, e)$ is a set M equipped with:

- A binary operation $\cdot : M \times M \to M$
- An identity element $e \in M$

satisfying:

- **Associativity:** $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in M$
- **Identity:** $e \cdot a = a = a \cdot e$ for all $a \in M$

Compare this to the axioms for a category:

Monoid	**Category**
Elements of M	Morphisms
Multiplication $\cdot$	Composition $\circ$
Identity e	Identity morphism id
Associativity of $\cdot$	Associativity of $\circ$

If a category has only one object $\star$, then *all* morphisms go from $\star$ to $\star$—they can always be composed! The morphisms form a monoid under composition.

$$\begin{tikzcd}
\star \arrow[loop above, "e"] \arrow[loop left, "a"] \arrow[loop right, "b"] \arrow[loop below, "ab"]
\end{tikzcd}$$

Examples of Monoids

1. **Natural numbers under addition:** $(\mathbb{N}, +, 0)$
   - Elements: $0, 1, 2, 3, \ldots$
   - Operation: addition
   - Identity: 0 (since $0 + n = n = n + 0$)
2. **Natural numbers under multiplication:** $(\mathbb{N}, \times, 1)$
   - Same set, different monoid structure!
   - Identity: 1 (since $1 \times n = n = n \times 1$)
3. **Strings under concatenation:** $(\Sigma^*, \cdot, \varepsilon)$
   - Elements: all strings over alphabet Σ
   - Operation: concatenation (e.g., "*ab*" $\cdot$ "*cd*" $=$ "*abcd*")
   - Identity: empty string ε
4. **Functions under composition:** $(A \to A, \circ, \mathrm{id}_A)$
   - Elements: functions from A to itself
   - Operation: function composition
   - Identity: the identity function
5. **Complexity classes:** The big-O notation satisfies monoid-like properties:
   - $O(f) \cdot O(g) = O(f \cdot g)$ (product rule)
   - $O(1)$ acts as a multiplicative identity

The Free Monoid

Given any set Σ (an "alphabet"), we can form the **free monoid** Σ^*: the set of all finite strings over Σ, with concatenation as the operation.
Why "free"? Because Σ^* is the *most general* monoid generated by Σ—it satisfies no equations other than the monoid axioms. This is a **universal property**:

> For any monoid M and any function $f : \Sigma \to M$, there is a *unique* monoid homomorphism $\bar{f} : \Sigma^* \to M$ extending f.

$$\begin{tikzcd}
\Sigma \arrow[r, "f"] \arrow[d, hook, "\eta"'] & M \\
\Sigma^* \arrow[ur, dashed, "\exists! \bar{f}"']
\end{tikzcd}$$

The map $\eta : \Sigma \to \Sigma^*$ sends each letter to the one-letter string.
Example. Let $\Sigma = \{a, b\}$ and let $M = (\mathbb{N}, +, 0)$. Define $f(a) = 3$, $f(b) = 5$. The unique extension $\bar{f} : \Sigma^* \to \mathbb{N}$ maps each string to the sum of its letter values:

$$\bar{f}(aabba) = f(a) + f(a) + f(b) + f(b) + f(a) = 3 + 3 + 5 + 5 + 3 = 19$$

This is why Σ^* appears in automata theory: a DFA computes exactly a monoid homomorphism from the free monoid to a finite monoid!

Monoid Homomorphisms

A **monoid homomorphism** $\phi : (M, \cdot_M, e_M) \to (N, \cdot_N, e_N)$ is a function $\phi : M \to N$ satisfying:

- $\phi(a \cdot_M b) = \phi(a) \cdot_N \phi(b)$ (preserves multiplication)
- $\phi(e_M) = e_N$ (preserves identity)

In categorical terms: a homomorphism between one-object categories is exactly a **functor**!
Example. The length function $\text{len} : \Sigma^* \to \mathbb{N}$ is a monoid homomorphism from $(\Sigma^*, \cdot, \varepsilon)$ to $(\mathbb{N}, +, 0)$:

- $\text{len}(s \cdot t) = \text{len}(s) + \text{len}(t)$
- $\text{len}(\varepsilon) = 0$

Example. The determinant function $\det : \text{GL}_n(\mathbb{R}) \to \mathbb{R}^*$ is a monoid homomorphism:

- $\det(AB) = \det(A) \cdot \det(B)$
- $\det(I) = 1$

Connection to Complexity Analysis

Consider the monoid $(\mathbb{N}, +, 0)$ of running times. When we analyze:

$$T(n) = T(n/2) + O(1)$$

we're saying: the total time is the *sum* (monoid operation) of the recursive time plus the local work. The structure of recurrence relations reflects monoid structure!
For divide-and-conquer with a subproblems:

$$T(n) = \underbrace{T(n/b) + T(n/b) + \cdots + T(n/b)}_{a \text{ terms}} + f(n)$$

The a-fold sum uses the monoid structure repeatedly.

Exercises: Monoids and Structure

1. Verify that $(\mathbb{Z}, +, 0)$ is a monoid. Is $(\mathbb{Z}, -, 0)$ a monoid? Why or why not?
2. The set $\{0, 1\}$ with operation "OR" ($\vee$) and identity 0 forms a monoid. Write out its multiplication table.
3. Show that the set of $n \times n$ matrices over $\mathbb{R}$ with matrix multiplication and the identity matrix I forms a monoid. (Note: this is *not* a group—why?)
4. Let $\Sigma = \{0, 1\}$. The *run-length encoding* maps a string to the sequence of run lengths. For example: $00111001 \mapsto [2, 3, 2, 1]$. Is run-length encoding a monoid homomorphism? Prove or give a counterexample.
5. **(Important)** Prove that if $\phi : M \to N$ is a monoid homomorphism, then the image $\phi(M) = \{\phi(m) : m \in M\}$ is a submonoid of N.
6. Let $\text{End}(A)$ be the monoid of functions $A \to A$ under composition. If $|A| = n$, what is $|\text{End}(A)|$? What is the identity element?
7. Consider the monoid $(\{0, 1, 2, \ldots, n-1\}, +_n, 0)$ where $+_n$ is addition mod n. Define a map $\phi : \mathbb{Z} \to \mathbb{Z}_n$ by $\phi(k) = k \mod n$. Prove this is a monoid homomorphism.
8. **(Challenge)** The powerset $\mathcal{P}(A)$ forms a monoid under union with $\emptyset$ as identity. It also forms a monoid under intersection with A as identity. Are these monoids isomorphic? (Two monoids are isomorphic if there's a bijective homomorphism between them.)
9. **(Big-O as Monoid)** Define a relation on functions where $f \sim g$ iff $f = \Theta(g)$. Show that the equivalence classes under $\sim$ form a monoid with multiplication $[f] \cdot [g] = [f \cdot g]$. What is the identity?
10. **(Connection to Automata)** Recall from Week 9 that a DFA defines a transition function $\delta : Q \times \Sigma \to Q$. Extend this to $\hat{\delta} : Q \times \Sigma^* \to Q$ by:
    - $\hat{\delta}(q, \varepsilon) = q$
    - $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$

    Fix a state q_0. Show that the map $w \mapsto \hat{\delta}(q_0, -)$ gives a monoid homomorphism from Σ^* to the monoid of functions $Q \to Q$.

The Categorical Perspective: Full Circle

We began this course with sets and functions, introduced diagrams to visualize composition, and discovered that many structures (preorders, graphs, automata) can be viewed categorically. Now we see that even the simplest algebraic structure—a monoid—is secretly a category.

This suggests a powerful principle: *category theory doesn't replace other mathematics; it reveals the common structure underlying diverse areas.* The monoid of complexity classes, the monoid of string concatenation in automata theory, and the monoid of functions under composition are all instances of the same abstract pattern.

As you continue in computer science and mathematics, you'll encounter this pattern repeatedly:

- Types form categories (where morphisms are functions)
- Proofs form categories (where morphisms witness logical entailment)
- Programs form categories (where morphisms are computations)

The tools you've developed—diagrams, universal properties, functors—are the beginning of a rich vocabulary for understanding computation, logic, and mathematics as aspects of a unified whole.

Practice

1. Order the functions $n \log n$, $n^{1.5}$, 2^n, n^3 by growth rate.
2. Show that $3n^2 + 5n + 7$ is $\Theta(n^2)$.
3. Solve the recurrence $T(n) = 2T(n/2) + n$ with $T(1) = 1$.
4. Analyze the runtime of binary search.
5. Prove: $\log n = O(n^\epsilon)$ for any $\epsilon > 0$.
6. Analyze the complexity of:

```
for i = 1 to n:
    for j = i to n:
        for k = 1 to j:
            // O(1)
```

7. Solve $T(n) = 3T(n/2) + n$ using the Master theorem.
8. Prove: $(n+1)! = O((n!)^2)$ but $(n+1)! \neq \Theta(n!)$.
9. Analyze the recurrence $T(n) = T(n-1) + n$ with $T(1) = 1$.
10. Show that $2^{n+1} = \Theta(2^n)$ but $2^{2n} \neq \Theta(2^n)$.
11. Prove: $f(n) = o(g(n))$ implies $f(n) = O(g(n))$.
12. A recursive algorithm satisfies $T(n) = T(n/3) + T(2n/3) + n$. Prove $T(n) = O(n \log n)$.

13. If $A \leq_p B$ and $B \in \mathbf{P}$, what can you conclude about A? Justify.
14. Explain why HAMILTONIAN CYCLE is in **NP**.
15. Outline a polynomial-time reduction from CLIQUE to VERTEX COVER (state how k changes and what graph transformation you use).