CS 251: Discrete Mathematics

Instructor's Guide

10-Week Quarter, Two 2-Hour Sessions per Week

Course Materials with Category Theory Enrichment

Contents

1 Course Overview

1.1 Course Philosophy

This course introduces discrete mathematics with an emphasis on **mathematical maturity** and **proof fluency**. While covering standard topics (sets, functions, relations, counting, graphs), we incorporate "Going Deeper" sections that connect to category theory and type theory—giving students a glimpse of the elegant abstractions underlying discrete structures. The Category Theory Companion (aligned by week) provides optional readings and exercises; the weekly notes include short Category Lens inserts that reference it.

1.2 Prerequisites

Students should have completed a first course covering:

- Propositional and predicate logic
- Basic proof techniques (direct, contrapositive, contradiction)
- Mathematical induction (weak and strong)

Chapter 0 provides a refresher on these topics for the first class session.

1.3 Course Materials

- **Primary:** Course notes (Weeks 0–10 LaTeX documents)
- **Supplementary:** Category Theory Companion
  `materials/latex/category_theory_companion.pdf`
- **Supplementary:** Category Lens problem sets
  `materials/latex/category_lens_problem_sets.pdf`
- **Supplementary:** Agda developments for each week
- **Supplementary:** Optional Agda labs and quick-start guide
  `materials/agda/README.md`
- **Reference:** Epp, *Discrete Mathematics with Applications* (optional)

1.4 Assessment Suggestions

- Weekly problem sets (60%)—mix of computation and proof; optionally include 1–2 Category Lens problems
- Theorem proving projects (40%)—two projects, described in Section 16

1.5 Session Structure (2 hours)

Recommended breakdown for each class:

Time	**Activity**
0:00–0:10	Warm-up problem / Review questions
0:10–0:50	Lecture block 1 (new material)
0:50–1:10	In-class activity / Worked examples
1:10–1:50	Lecture block 2 (continued material)
1:50–2:00	Summary / Preview of next session

2 Week 0: Proof Refresher (Optional Pre-Week)

If your students need a refresher, cover this material in an optional session or assign as pre-reading before Week 1.

2.1 Topics

- Propositional logic review (connectives, truth tables, logical equivalence)
- Predicate logic review (quantifiers, negation of quantified statements)
- Proof techniques: direct proof, contrapositive, contradiction
- Mathematical induction: weak, strong, structural
- Common proof patterns and templates

In-Class Activity

Proof Workshop: Give students 4–5 statements and have them identify which proof technique is most appropriate, then complete one proof together.
Example statements:

1. If n^2 is even, then n is even. (Contrapositive)
2. $\sqrt{2}$ is irrational. (Contradiction)
3. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$. (Induction)
4. If $a \mid b$ and $b \mid c$, then $a \mid c$. (Direct)

Teaching Tip

Students often struggle with the *structure* of proofs. Emphasize that proofs are communication—they should guide the reader through the logical steps. Encourage students to write in complete sentences and explicitly state which technique they're using.

3 Week 1: Set Theory

Class A (Tuesday)

Topics:

- Set notation and membership ($\in$, $\notin$)
- Set-builder notation; common sets ($\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$)
- Subsets and set equality
- Power sets
- Set operations: union, intersection, complement, difference

Learning Goals:

- Translate between verbal descriptions and set-builder notation
- Prove set containment using element-chasing arguments
- Compute power sets for small sets

Warm-up (10 min): List all subsets of $\{1, 2, 3\}$. How many are there? Conjecture a formula for $|\mathcal{P}(A)|$ when $|A| = n$.

In-Class Activity

Venn Diagram Gallery Walk: Post 6 Venn diagrams around the room, each showing a shaded region. Students circulate and write the set expression for each shaded region. Discuss multiple correct answers (e.g., $A \cap B'$ vs $A \setminus B$).

Class B (Thursday)

Topics:

- Proving set identities (distributive laws, De Morgan's laws)
- Cartesian products
- Indexed families of sets; generalized union and intersection
- Introduction to cardinality (finite sets)

Learning Goals:

- Write rigorous proofs of set identities using element-chasing
- Understand Cartesian products and tuples
- Apply set identities to simplify expressions

Lecture Focus: Work through the proof of De Morgan's law in detail:

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

Emphasize the bidirectional nature of set equality proofs.

Category Lens pacing (Class B, 5–10 min):
Use the arrows/points/terminal object box; point students to the Category Theory Companion Week 1–2 and the Week 1 problem set.

Agda Connection

File: `Week01_SetTheory.agda`
Key concepts to highlight:

- Sets as predicates: `Pred A = A -> Set`
- Union as sum type: `(P ∪ Q) x = Sum (P x) (Q x)`
- Intersection as product type: `(P ∩ Q) x = Pair (P x) (Q x)`
- Subset proofs are *functions*: `P ⊆ Q = (x : A) -> P x -> Q x`

Going Deeper: Mention that sets form a *Boolean algebra*—this connects to circuit design and propositional logic. The Agda file proves the non-contradiction law constructively.

Teaching Tip

Students often write "Let $x \in A \cup B$" without considering both cases. Drill the pattern:

1. State what you want to prove
2. Handle both cases of the union separately
3. Conclude with what you've shown

4 Week 2: Functions

Class A (Tuesday)

Topics:

- Definition of function; domain, codomain, range
- Function equality
- Injective (one-to-one) functions
- Surjective (onto) functions
- Bijective functions

Learning Goals:

- Determine whether a function is injective/surjective from its definition
- Prove a function is injective or surjective
- Construct counterexamples when a function lacks these properties

Warm-up: For each function, classify as injective, surjective, both, or neither:

1. $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$
2. $g : \mathbb{Z} \to \mathbb{Z}$, $g(n) = n + 1$
3. $h : \mathbb{R} \to \mathbb{R}^+$, $h(x) = e^x$

In-Class Activity

Function Card Sort: Prepare cards with various functions (some with explicit formulas, some described verbally, some as diagrams). Students sort them into four categories: injective only, surjective only, bijective, neither. Discuss edge cases.

Class B (Thursday)

Topics:

- Function composition
- Identity function; composition laws (associativity, identity)
- Inverse functions
- Proving composition preserves injectivity/surjectivity

Learning Goals:

- Compute compositions of functions
- Prove that $g \circ f$ injective implies f injective
- Find inverse functions and verify they satisfy the inverse properties

Key Theorem: f has a two-sided inverse if and only if f is bijective.

Category Lens pacing (Class B, 5–10 min):
Use the sections/retractions/idempotents box; point students to the Category Theory Companion Week 1–2 and the Week 2 problem set.

Agda Connection

File: `Week02_Functions.agda`
Key concepts:

- Composition: `(g ∘ f) x = g (f x)`
- Injective: `(x y :  A) -> Eq (f x) (f y) -> Eq x y`
- Isomorphism record with `to`, `from`, and inverse proofs

Going Deeper: Functions form a *category*! Show students the three laws:

1. $\mathrm{id} \circ f = f$ (left identity)
2. $f \circ \mathrm{id} = f$ (right identity)
3. $(h \circ g) \circ f = h \circ (g \circ f)$ (associativity)

The Agda file proves these as `composeIdLeft`, `composeIdRight`, `composeAssoc`.

Teaching Tip

The notation $g \circ f$ ("g after f") confuses students because f appears second but is applied first. Use the mnemonic "read right to left" or write compositions as $f; g$ in class discussions if it helps.

5 Week 3: Relations and Modular Arithmetic

Class A (Tuesday)

Topics:

- Binary relations; relation as subset of $A \times B$
- Properties: reflexive, symmetric, transitive, antisymmetric
- Equivalence relations
- Equivalence classes and partitions
- Partial orders; Hasse diagrams

Learning Goals:

- Test whether a relation has specific properties
- Prove a relation is an equivalence relation
- Compute equivalence classes and understand the partition correspondence
- Draw Hasse diagrams and identify minimal/maximal/least/greatest elements

Warm-up: For each relation on $\mathbb{Z}$, identify which properties it has:

1. $a \sim b$ iff $a = b$
2. $a \sim b$ iff $a \leq b$
3. $a \sim b$ iff $a - b$ is even
4. $a \sim b$ iff $|a - b| \leq 1$

In-Class Activity

Partition Puzzle: Give students a set $S = \{1, 2, 3, 4, 5, 6\}$ and several partitions. For each partition, have them write out the corresponding equivalence relation explicitly as a set of ordered pairs. Then reverse: given an equivalence relation, find the partition.

Class B (Thursday)

Topics:

- Modular arithmetic: congruence modulo n
- Properties of congruence (equivalence relation)
- Modular arithmetic operations
- Division algorithm; div and mod
- Introduction to Fermat's Little Theorem (statement only)

Learning Goals:

- Perform modular arithmetic calculations
- Prove statements about divisibility and congruence
- Apply modular arithmetic to practical problems (checksums, cryptography preview)

Application: ISBN check digits, credit card validation (Luhn algorithm).

Category Lens pacing (Class B, 5–10 min):
Use the preorders-as-categories and Galois connection box; assign Companion Week 3 and the Week 3 problem set.

Agda Connection

File: `Week03_RelationsMod.agda`
Key concepts:

- Relation as function: `Rel A = A -> A -> Set`
- Equivalence record with reflexivity, symmetry, transitivity proofs
- `Prime` data type and `fermatLittle` postulate
- Chinese Remainder Theorem as `CRTSystem` record

Going Deeper: Equivalence relations correspond to *quotient types*. The kernel of any function is an equivalence relation—this is why modular arithmetic "works."

Teaching Tip

Students often confuse "$a \equiv b \pmod{n}$" with "$a = b \mod n$." Emphasize that the first is a *relation* (a statement that can be true or false), while the second is typically an *operation* that computes a remainder.

6 Week 4: Counting and Probability I

Class A (Tuesday)

Topics:

- Sample spaces and events; equally likely probability
- Counting principles: sum rule, product rule
- Permutations and arrangements with restrictions
- Addition rule and inclusion–exclusion (two or three sets)
- Pigeonhole principle

Learning Goals:

- Build sample spaces and compute basic probabilities
- Apply sum/product rules and inclusion–exclusion
- Use the pigeonhole principle in proofs

Warm-up: Two coins are flipped. List the sample space and compute the probability of exactly one head.

In-Class Activity

Counting Stations: Set up 4 stations with different counting problems. Groups rotate every 10 minutes, solving one problem at each station. Problems should vary in difficulty and technique required. Conclude with a gallery walk where each group presents their solution.

Class B (Thursday)

Topics:

- Complement counting and derangements
- Conditional probability and independence
- Bayes' theorem and total probability

Learning Goals:

- Use complement counting to simplify problems
- Compute conditional probabilities and test independence
- Apply Bayes' theorem in inverse-probability settings

Application: Medical testing (sensitivity, specificity, false positives).

Category Lens pacing (Class B, 5–10 min):

Use the algebra-of-types box and mention exponentials/currying; assign Companion Weeks 4–5 and the Week 4 problem set.

Agda Connection

File: `Week04_CountingProb1.agda`
Key concepts:

- `factorial` and `choose` functions defined recursively
- Pascal's identity verified: `pascal : (n k : Nat) -> Eq (choose (succ n) (succ k)) (choose n (succ k) + choose n k)`

Going Deeper: Combinatorial objects often have recursive structure. The recursive definition of $\binom{n}{k}$ corresponds to a decision: does the subset include element n or not?

Teaching Tip

For conditional probability problems, require students to define events clearly and write $P(A \mid B) = P(A \cap B)/P(B)$ before computing. Tree diagrams help avoid missed cases.

7 Week 5: Counting and Probability II

Class A (Tuesday)

Topics:

- Combinations and binomial coefficients
- Binomial theorem and Pascal's identity
- Stars and bars (combinations with repetition)
- Multinomial coefficients

Learning Goals:

- Decide when order matters and use combinations correctly
- Apply the binomial theorem and Pascal's identity
- Solve distribution problems with stars and bars

Warm-up: How many 5-card hands contain exactly 2 aces?

In-Class Activity

Stars and Bars Workshop: Give 3–4 distribution problems (cookies to kids, identical balls into bins, integer solutions). Have groups translate each into $x_1 + \cdots + x_k = n$ and compute $\binom{n+k-1}{k-1}$.
Have students:

1. Identify the "stars" and the "bars"
2. Write one small example explicitly
3. Compare answers across groups for consistency

Class B (Thursday)

Topics:

- Recurrence relations for sequences
- Solving simple linear recurrences
- Generating functions (optional enrichment)

Learning Goals:

- Write recurrences for counting problems
- Solve first- and second-order recurrences with constant coefficients
- Interpret generating functions as algebraic encodings of sequences

Application: Tiling and Fibonacci-style counting problems.

Category Lens pacing (Class B, 5–10 min):
Use the "subsets as maps to 2" lens; assign Companion Weeks 4–5 and the Week 5 problem set.

Agda Connection

File: `Week05_CountingProb2.agda`
Key concepts:

- `choose` and Pascal's identity
- `starsAndBars` and `multichoose`
- `multinomial` and basic combinatorial identities
- Optional enrichment: inclusion-exclusion and derangements

Going Deeper: Recurrences arise naturally from combinatorial decompositions (e.g., tilings and Fibonacci numbers). Use them to reinforce induction.

Teaching Tip

For stars and bars problems, always have students:

1. Identify what the "stars" represent
2. Identify what the "bars" separate
3. Write out a small example explicitly

The formula $\binom{n+k-1}{k-1}$ is easy to misremember; understanding beats memorization.

8 Week 6: Expected Value and Random Variables

Class A (Tuesday)

Topics:

- Random variables (discrete)
- Probability mass functions
- Expected value: definition and properties
- Linearity of expectation

Learning Goals:

- Define random variables for counting problems
- Compute expected values directly and using linearity
- Apply indicator random variables

Warm-up: You roll two dice. Let X be the sum. What is $E[X]$? What is $E[X^2]$? Is $E[X^2] = E[X]^2$?

In-Class Activity

Coupon Collector Exploration: A cereal company puts one of n different toys in each box. How many boxes do you expect to buy to collect all n toys?
Guide students through:

1. Define X_i = boxes needed to get the ith new toy (after having $i-1$)
2. Show X_i is geometric with success probability $(n-i+1)/n$
3. Apply linearity: $E[X] = \sum_{i=1}^{n} E[X_i] = n \cdot H_n$

Class B (Thursday)

Topics:

- Variance and standard deviation
- Common distributions: Bernoulli, binomial, geometric
- Applications of expected value in algorithms
- Introduction to graphs (transition to Week 7)

Learning Goals:

- Compute variance for simple distributions
- Recognize when to apply standard distributions
- Understand expected running time analysis

Application: Expected number of comparisons in randomized quicksort.

Category Lens pacing (Class B, 5–10 min): Use the parts/predicates and graph-homomorphism lens when introducing graphs; assign Companion Weeks 6–7 and the Week 6 problem set.

Agda Connection

File: `Week06_ExpectationGraphs.agda`
Key concepts:

- Probability distributions as functions to rationals
- Expected value computation
- Graph representations (adjacency list, matrix) introduced

Going Deeper: Expected value is a *linear functional* on the space of random variables. This linearity is what makes many complex calculations tractable.

Teaching Tip

The coupon collector problem is excellent for building intuition about linearity of expectation. Emphasize that we *don't need independence* to use linearity—this is what makes it so powerful.

9 Week 7: Graph Theory I

Class A (Tuesday)

Topics:

- Graph terminology: vertices, edges, degree, paths, cycles
- Graph representations: adjacency matrix, adjacency list
- Special graphs: complete, bipartite, cycle, path graphs
- Handshaking lemma

Learning Goals:

- Translate between different graph representations
- Identify properties of special graph families
- Apply the handshaking lemma to prove existence results

Warm-up: Draw K_4, $K_{2,3}$, C_5, and P_4. For each, determine the number of vertices, edges, and whether it's bipartite.

In-Class Activity

Graph Isomorphism Challenge: Present pairs of graphs drawn differently. Students determine whether each pair is isomorphic. For isomorphic pairs, find the bijection. For non-isomorphic pairs, identify an invariant that differs (vertex count, edge count, degree sequence, etc.).

Class B (Thursday)

Topics:

- Graph isomorphism; isomorphism invariants
- Walks, trails, paths; Eulerian trails and circuits
- Euler's theorem for Eulerian graphs
- Graph coloring introduction; chromatic number

Learning Goals:

- Prove graphs are non-isomorphic using invariants
- Determine whether a graph has an Eulerian trail/circuit
- Find proper colorings and bounds on chromatic number

Historical Note: The Seven Bridges of Königsberg problem (1736) launched graph theory.

Category Lens pacing (Class B, 5–10 min):
Use the free-category-on-a-graph and connected-components lens; assign Companion Weeks 6–7 and the Week 7 problem set.

Agda Connection

File: `Week07_GraphTheoryI.agda`
Key concepts:

- `MatrixGraph n` record with adjacency function
- `Walk`, `Trail`, `Path` data types
- `Coloring`, `ProperColoring`, `Colorable` definitions
- `chromaticNumber` and `fourColorTheorem` postulates

Going Deeper: Graph homomorphisms generalize colorings: a proper k-coloring is exactly a homomorphism to K_k. This connects graph theory to category theory.

Teaching Tip

When teaching Euler's theorem, have students physically trace paths on whiteboard graphs. The parity argument (even degree = can escape any vertex you enter) becomes intuitive with physical demonstration.

10 Week 8: Trees and Graph Algorithms

Class A (Tuesday)

Topics:

- Trees: definitions and characterizations
- Tree properties: $|E| = |V| - 1$, unique paths
- Rooted trees; binary trees
- Tree traversals: preorder, inorder, postorder

Learning Goals:

- Prove equivalent characterizations of trees
- Perform tree traversals
- Understand the structure of binary search trees

Warm-up: Prove: A connected graph with n vertices and $n - 1$ edges is a tree.

In-Class Activity

Traversal Race: Display a binary tree. Teams race to produce the preorder, inorder, and postorder traversals. Verify by reconstructing the tree from two traversals.
Follow-up question: Given preorder and postorder only, can you always reconstruct the tree? (No—consider a tree with only left children.)

Class B (Thursday)

Topics:

- Spanning trees; BFS/DFS spanning trees
- Minimum spanning trees: Kruskal's and Prim's algorithms
- Shortest paths: BFS, Dijkstra's, Bellman-Ford
- Matchings in bipartite graphs (Hall's theorem)
- Network flows (max-flow min-cut)
- Folds on trees (catamorphisms)

Learning Goals:

- Find MSTs using greedy algorithms
- Trace Dijkstra's algorithm
- Use Hall's theorem to certify matchings
- Compute a max flow and identify a minimum cut
- Express tree computations as folds

Category Lens pacing (Class B, 5–10 min):
Use the initial-algebra/fold lens at the end of the traversal section; assign Companion Week 8 and the Week 8 problem set.

Agda Connection

File: `Week08_TreesAlgorithms.agda`
Key concepts:

- `BinTree` and `Tree` (rose tree) data types
- `binFold :  (A -> B -> B -> B) -> B -> BinTree A -> B`
- Many operations as folds: `sizeAsFold`, `heightAsFold`, `mirror`, `mapTree`

Going Deeper: Tree folds are *catamorphisms*—the unique maps from an initial algebra. This is why so many tree operations factor through a single fold pattern. The fold essentially says: "Replace constructors with operations."

Teaching Tip

The fold abstraction is powerful but initially confusing. Start with concrete examples:

- Size: replace `leaf` with 0, `branch` with $\lambda x\, l\, r.\, 1 + l + r$
- Sum: replace `leaf` with 0, `branch` with $\lambda x\, l\, r.\, x + l + r$

Then show how both instantiate the same pattern.

11 Week 9: Formal Languages and Automata

Class A (Tuesday)

Topics:

- Alphabets, strings, languages
- Regular expressions
- Deterministic finite automata (DFA)
- DFA acceptance; designing DFAs

Learning Goals:

- Write regular expressions for given languages
- Design DFAs for simple languages
- Trace DFA execution on input strings

Warm-up: Write a regular expression for: binary strings with an even number of 1s.

In-Class Activity

DFA Design Workshop: Groups design DFAs for increasingly complex languages:

1. Strings ending in "01"
2. Strings with "010" as a substring
3. Strings where every "0" is immediately followed by "1"
4. Binary representations of multiples of 3

Groups present their solutions and peer-review for correctness.

Class B (Thursday)

Topics:

- Nondeterministic finite automata (NFA)
- Equivalence of DFA and NFA (subset construction)
- Equivalence of regular expressions and finite automata
- Pumping lemma (informal)
- Context-free grammars (CFG) and ambiguity
- Pushdown automata (PDA) and CFG–PDA equivalence

Learning Goals:

- Convert NFAs to DFAs
- Convert regular expressions to NFAs
- Use the pumping lemma to prove languages are not regular
- Write simple CFGs and derive example strings
- Describe a PDA for $a^n b^n$-type languages

Category Lens pacing (Class B, 5–10 min):
Use the coalgebra/automata lens and mention the free-monoid action; assign Companion Week 9 and the Week 9 problem set.

Agda Connection

File: `Week09_RegexAutomata.agda`
Key concepts:

- `Regex` data type with constructors for $\emptyset$, ε, literals, union, concatenation, star
- `DFA` and `NFA` record types
- `accepts` function for DFA execution

Going Deeper: Regular languages form a *Kleene algebra*—a structure with union, concatenation, and Kleene star satisfying specific axioms. DFAs can be viewed as coalgebras, dual to the algebraic view of syntax.

Teaching Tip

The subset construction can produce exponentially many states. Work through a small example completely, then ask: "What's the worst case?" Show that an NFA with n states can require a DFA with 2^n states (the "count the kth-from-last character" example).

12 Week 10: Complexity and Tic-Tac-Toe

Class A (Tuesday)

Topics:

- Review of Big-O notation
- Big-Ω and Big-Θ
- Analyzing recursive algorithms (recurrence relations)
- Master theorem
- Complexity classes overview (P, NP)

Learning Goals:

- Determine asymptotic complexity of algorithms
- Set up and solve simple recurrences
- Apply the master theorem
- Distinguish P vs. NP at a high level

Warm-up: Rank these functions by growth rate: n^2, 2^n, $n \log n$, $n!$, $\log n$, $n^{1.5}$, n

In-Class Activity

Algorithm Analysis Gallery: Post 6–8 code snippets (sorting algorithms, search algorithms, recursive functions). Students analyze each and determine Big-O complexity. Discuss which analyses require careful counting vs. simple pattern recognition.

Class B (Thursday)

Topics:

- Complexity classes: P, NP, NP-complete
- Polynomial-time reductions (conceptual)
- Optional: game trees and minimax
- Course review and synthesis

Learning Goals:

- Explain what it means for a problem to be NP-complete
- Recognize the role of reductions in complexity
- Synthesize course material: sets, functions, counting, graphs, trees

Final Activity: Optional: Tic-Tac-Toe is solved (always a draw with optimal play). Walk through the game tree analysis using minimax.

Category Lens pacing (Class B, 5–10 min):
Use the monoids-as-one-object-categories lens and the free/forgetful adjunction; assign Companion Week 10 and the Week 10 problem set.

Agda Connection

File: `Week10_Efficiency.agda`
Key concepts:

- Big-O as a relation: eventually bounded
- Optional: `GameTree` data type and `minimax`

Going Deeper: Complexity theory connects to logic via the idea that proofs correspond to programs; reductions formalize how difficulty transfers between problems.

Teaching Tip

The tic-tac-toe game tree is small enough to fully explore but large enough to motivate pruning. Use this as a bridge to discussing alpha-beta pruning for more complex games like chess.

13 Weekly Homework Suggestions

13.1 Homework Philosophy

Each problem set should include:

- **Computational problems** (40%): Practice with techniques
- **Proof problems** (40%): Develop proof-writing skills
- **Challenge problems** (20%): Deeper thinking, optional extra credit

13.2 Sample Problems by Week

Week	Sample Problems
1	Prove $(A \cap B) \cup (A \cap B^c) = A$. Compute $\lvert\mathcal{P}(\mathcal{P}(\{1, 2\}))\rvert$.
2	Show $f(x) = 2x + 1$ is bijective $\mathbb{R} \to \mathbb{R}$. Prove composition of bijections is bijective.
3	Draw a Hasse diagram for divisibility on $\{1, 2, 3, 6, 12\}$. Find all x with $3x \equiv 5 \pmod 7$.
4	Use inclusion-exclusion to count integers in $\{1, \ldots, 100\}$ divisible by 2 or 3 or 5. Compute a conditional probability from a medical test scenario.
5	Solve a stars and bars distribution problem. Solve $a_n = 4a_{n-1} - 4a_{n-2}$ with given initial conditions.
6	Find $E[X]$ where X = sum of two dice. Expected value of max of two dice.
7	Determine if two given graphs are isomorphic. Find $\chi(C_7)$ and $\chi(K_{3,3})$.
8	Prove every tree is bipartite. Use Hall's theorem to certify a matching.
9	Design DFA for $\{w \in \{0, 1\}^* : w \text{ has } 010 \text{ as substring}\}$. Give a CFG for $\{0^n 1^n\}$.
10	Solve $T(n) = 2T(n/2) + n$. Explain why HAMILTONIAN CYCLE is in **NP**.

14 Agda Integration Guide

14.1 Why Agda?

Agda serves several pedagogical purposes:

1. **Precision:** Forces students to be completely explicit about definitions and proofs
2. **Feedback:** Type errors catch logical mistakes immediately
3. **Exploration:** Interactive proving develops mathematical intuition
4. **Connection:** Demonstrates the Curry-Howard correspondence (propositions as types)

14.2 Integration Options

Option A: Demonstration Only

- Show Agda in lectures to illustrate key concepts
- No student coding required
- Use for: Sets as predicates, functions as morphisms, proofs as programs

Option B: Optional Enrichment

- Provide Agda files for interested students
- Offer extra credit for completing Agda exercises
- Include brief Agda explanations in "Going Deeper" sections
- Point students to the weekly lab files for starter exercises

Option C: Integrated Labs

- Weekly 1-hour Agda lab sessions
- Structured exercises with holes for students to fill
- Use `Week01_Lab.agda`–`Week10_Lab.agda` as lab handouts
- Graded on completion/effort

14.3 Optional Lab Scaffolding

- `materials/agda/README.md` includes setup, how-to-run, and the week-by-week map
- `materials/agda/Week01_Lab.agda`–`Week10_Lab.agda` contain starter exercises

Each lab file uses `--allow-unsolved-metas` so students can load the file before finishing proofs.

14.4 Agda Exercises by Difficulty

The table below highlights extra exercises in the main development files; the lab files cover week-by-week starters.

File	**Exercises**	**Difficulty**
Common	Prove `addAssoc`, `addZeroRight`	Easy
Week01	Prove subset transitivity, De Morgan's laws	Easy–Medium
Week02	Show `succ` is injective, functor laws for `Maybe`	Medium
Week03	Divisibility properties, kernel equivalence	Medium
Week05	Verify derangement values, Stirling recurrence	Medium
Week07	Euler formula verification (rearranged)	Medium
Week08	Express operations as folds, prove fold laws	Medium–Hard

15 "Going Deeper" Topic Guide

The "Going Deeper" sections introduce category theory and type theory concepts at an accessible level. Here's guidance on presenting these topics: Each week now has a matching section in the Category Theory Companion, which can be assigned as optional reading or used for extra problems.

15.1 Week 1: Arrows, Points, and Terminal Objects

Key Message: Elements are maps $1 \to A$; sets and functions form a category with terminal and initial objects.
Accessibility: Very accessible. Use arrows and small-set examples.
Connection: Universal properties begin here (products, terminal object).

15.2 Week 2: Isomorphisms, Sections, and Retractions

Key Message: Bijections are isomorphisms; left/right inverses are retractions/sections; idempotents record retracts.
Accessibility: Moderately accessible. Use concrete functions and small diagrams.
Connection: "Division of maps" and categorical cancellation.

15.3 Week 3: Preorders as Categories

Key Message: Preorders are thin categories; monotone maps are functors; Galois connections are adjunctions for orders.
Accessibility: Accessible with divisibility and subset examples.
Connection: Meets/joins as products/coproducts in a preorder.

15.4 Weeks 4–5: Products, Sums, and Exponentials

Key Message: Counting rules mirror categorical constructions: sums, products, and map objects B^A (currying).
Accessibility: Very accessible with type-counting and binomial examples.
Connection: Subsets as maps to 2; bijections as structure-preserving maps.

15.5 Week 6: Parts and Predicates

Key Message: Power sets describe "parts" of an object; preimage maps preserve structure.
Accessibility: Accessible via events and characteristic functions.
Connection: Probability as measures on parts; logic via $A \to 2$.

15.6 Week 7: Graphs Generate Categories

Key Message: The free category on a graph formalizes paths; adjacency matrices count morphisms; connected components form a functor.
Accessibility: Accessible with concrete graphs and paths.
Connection: Functorial invariants and structure-preserving maps.

15.7 Week 8: Initial Algebras and Folds

Key Message: Recursive data types are initial algebras; folds are unique maps out of them.
Accessibility: Most challenging. Focus on the pattern and examples.
Connection: Catamorphisms and recursion schemes.

15.8 Week 9: Coalgebras and Automata

Key Message: DFAs are coalgebras; automata are actions of the free monoid.
Accessibility: Moderate. Use small DFAs as examples.
Connection: Final coalgebras and language semantics.

15.9 Week 10: Monoids and Adjunctions

Key Message: Monoids are one-object categories; free/forgetful adjunction explains Σ^*.
Accessibility: Moderate.
Connection: Cost monoids and resource tracking.

16 Theorem Proving Projects

Students complete two theorem proving projects during the quarter. These projects develop deep understanding through extended engagement with proof, connecting paper-and-pencil mathematics to mechanized verification.

16.1 Project 1: Foundations (Due Week 5)

Overview: Students formalize and prove properties about sets, functions, and relations—either in Agda or as a written proof portfolio.

16.1.1 Option A: Agda Track

Complete the following proofs in Agda (fill in holes in provided files):

1. **Set Theory:** Prove 5 set identities including:
    - De Morgan's laws (both directions)
    - Distributivity of $\cap$ over $\cup$
    - Symmetric difference is associative
2. **Functions:** Prove 4 properties including:
    - Composition of injections is injective
    - If $g \circ f$ is surjective, then g is surjective
    - Functor laws for `Maybe` (identity and composition)
3. **Relations:** Prove 3 properties including:
    - Kernel of any function is an equivalence relation
    - Divisibility is transitive

16.1.2 Option B: Written Track

Submit a proof portfolio with:

1. 6 polished proofs of set/function/relation theorems
2. Each proof must include: theorem statement, proof strategy overview, complete proof, and reflection on difficulties encountered
3. At least 2 proofs must use different techniques (direct, contrapositive, contradiction, induction)

Teaching Tip

Grading Rubric (per proof):

- Correctness (40%): Logical validity, no gaps
- Clarity (30%): Well-organized, clear prose
- Style (20%): Appropriate level of detail, good notation
- Reflection (10%): Insightful discussion of proof strategy

16.2 Project 2: Structures (Due Week 10)

Overview: Students explore a deeper topic connecting discrete structures to broader mathematics, with emphasis on the "Going Deeper" themes.

16.2.1 Option A: Agda Track

Choose one of the following extended developments:

1. **Counting and Bijections:**
   - Prove Pascal's identity computationally
   - Prove the hockey-stick identity
   - Verify Stirling number recurrence
   - Implement and verify derangement counting
2. **Graph Theory:**
   - Implement graph representations
   - Prove handshaking lemma
   - Verify Euler's formula for specific planar graphs
   - Implement and verify 2-coloring for bipartite graphs
3. **Trees and Folds:**
   - Implement 5+ tree operations as folds
   - Prove fold fusion law
   - Prove mirror is an involution
   - Show that map preserves identity and composition (functor laws)
4. **Automata Theory:**
   - Implement regex matching via derivatives
   - Prove properties of regex operations
   - Implement DFA simulation
   - Verify DFA/NFA equivalence for small examples

16.2.2 Option B: Written Track

Write an expository paper (8–12 pages) on one of:

1. **Boolean Algebras and Sets:** Explain the Boolean algebra axioms, show sets satisfy them, connect to propositional logic and digital circuits.
2. **The Category of Sets:** Explain what a category is, show that sets and functions form one, discuss universal properties (products, coproducts).
3. **Equivalence Relations and Quotients:** Explain the correspondence between equivalence relations and partitions, develop modular arithmetic as a quotient, discuss quotient structures in algebra.

4. **Graph Coloring and Homomorphisms:** Explain graph homomorphisms, show colorings are homomorphisms to complete graphs, discuss the chromatic polynomial.

5. **Catamorphisms and Recursion Schemes:** Explain folds as structured recursion, show how many operations factor through folds, discuss the connection to initial algebras.

Paper Requirements:

- Clear exposition accessible to classmates
- At least 3 fully worked examples
- At least 2 complete proofs
- Annotated bibliography with 4+ sources
- Discussion of connections to course material

16.3 Project Timeline and Milestones

Week	Project 1	Project 2
2	Topic selection	—
3	Progress check (2 proofs done)	—
5	**Final submission**	Topic selection
7	—	Progress check
9	—	Draft for peer review
10	—	**Final submission**

16.4 Peer Review Process

For Project 2, incorporate peer review:

1. Week 9: Students exchange drafts with a partner
2. Partners provide written feedback on:
   - Clarity of exposition
   - Correctness of proofs
   - Suggestions for improvement
3. Final submission includes a "response to reviewers" section addressing feedback

In-Class Activity

Project Showcase (Finals Week): Optional presentations where students share their Project 2 work. Each presenter gets 10 minutes to explain the main ideas and one interesting proof. This builds communication skills and exposes students to topics beyond their own project.

17 Additional Resources

17.1 Recommended Reading

- **Primary Text:** Epp, *Discrete Mathematics with Applications*
- **Course Companion:** Category Theory Companion (aligned by week)
- **Proof Writing:** Hammack, *Book of Proof* (free online)
- **Graph Theory:** West, *Introduction to Graph Theory*
- **Category Theory (gentle):** Lawvere & Schanuel, *Conceptual Mathematics*
- **Type Theory:** Pierce, *Types and Programming Languages*

17.2 Online Resources

- **Agda:** `https://agda.readthedocs.io/`
- **Category Theory:** nLab (`https://ncatlab.org/`)
- **Proof Practice:** `https://www.proofwiki.org/`

17.3 Software

- **Agda:** Install via Haskell Stack or system package manager
- **Graph Visualization:** Graphviz, yEd
- **Automata Simulation:** JFLAP

This guide accompanies the CS 251 course materials.
Last updated: January 9, 2026