COMSC 2043: Induction and Recursion
Teacher Solution Guide

Jeremy Evert
Southwestern Oklahoma State University

October 27, 2025

Contents

Chapter 1

Foundations of Induction and Recursion — Teacher's Guide

1.1 Overview for Instructors

This chapter anchors students in the duality of **induction and recursion**. Your goal is to help them see that these two concepts are not separate tools, but complementary ways of describing repetition and self-reference.

A strong introduction here sets the tone for the rest of the course: students must leave with intuition, not just definitions. Encourage them to see recursion as "definition by self-reference" and induction as "proof by dominoes."

1.2 Teaching Objectives

By the end of this chapter, students should be able to:

- Explain the relationship between recursion and induction.
- Write a simple recursive definition (e.g., factorial).
- Construct and justify a basic proof by mathematical induction.
- Recognize how recursive algorithms reflect inductive proofs.

1.3 Section 1.1 — The Big Picture

Student Goal: Understand that recursion defines a process and induction proves it correct.

Teaching Note: Open class with a visual metaphor — line up real dominoes or use a slide animation. Show one falling into the next. Then write on the board:

> *Induction: proving all dominoes fall.*
> *Recursion: building the dominoes themselves.*

Encourage students to explain how one supports the other. For programming-minded learners, connect the idea to a recursive call stack: each call relies on the truth of smaller subproblems.

1.4 Section 1.2 — Key Ideas from Rosen's Chapter 5

Basis Step: Verify that $P(0)$ or $P(1)$ is true. This builds confidence in the foundation.

Inductive Step: Assume $P(k)$ and prove $P(k+1)$. This forms the "engine" of reasoning.

Strong Induction: Stress that this is not stronger logic, but broader assumption.

Recursive Definition: Let the process mirror induction — base case + recursive rule.

Structural Induction: For trees and grammars, emphasize that the same logic applies to structure.

Instructor Tip: Rosen's section on structural induction (Chapter 5.3) pairs beautifully with programming examples. Ask students how a parse tree or HTML document could be proven valid using the same logic.

1.5 Section 1.3 — Why This Matters

Students often treat induction as abstract until it's made concrete. Use examples that connect mathematical induction to computer science:

- Recursive definitions in Python or Java mirror inductive reasoning.
- Correctness proofs for loops and algorithms rely on inductive invariants.
- Sorting, searching, and even AI search trees can be analyzed inductively.

Misconception Watch: Students may think induction proves "by example." Clarify: *One base case and one domino rule are enough for infinitely many cases.*

1.6 Section 1.4 — Example: The Factorial Function

Recursive Definition:

$$n! = \begin{cases} 1, & n = 0, \\ n \cdot (n-1)!, & n > 0. \end{cases}$$

Proof by Induction: Show that $n! \geq 2^{n-1}$ for $n \geq 1$.

Base case: $1! = 1 \geq 2^0 = 1$

Inductive step: Assume $k! \geq 2^{k-1}$. Then $(k+1)! = (k+1)k! \geq (k+1)2^{k-1} \geq 2^k$.

Thus the property holds for all $n \geq 1$.

Instructor Strategy: 1. Work through this on the board line by line. 2. Have students complete the inequality on their own for $(k+2)!$ to test comprehension. 3. Discuss what would break if the base case were omitted.

1.7 Section 1.5 — The Student Challenge

Challenge statement (student version):

> "Induction is not a leap of faith—it's a method of climbing an infinite ladder, one rung at a time."

Teacher Expansion: Encourage students to:

1. Write a recursive Python function for $n!$.
2. Prove its correctness using induction.
3. Identify the correspondence between code and proof:
   - Base case $\leftrightarrow$ if-statement for $n = 0$
   - Inductive step $\leftrightarrow$ recursive call to smaller n

Sample Code:

Listing 1.1: Recursive factorial function in Python

```
def factorial(n):
    """Return n! recursively."""
    if n == 0:
        return 1
    return n * factorial(n - 1)
```

Ask students to trace 'factorial(4)' and list every call on the board. Then show how the recursion tree mirrors the structure of the inductive proof.

Extension: Have advanced students formalize the induction as a theorem:

$$\forall n \geq 0, \ \text{factorial}(n) = n!$$

1.8 Section 1.6 — Checkpoint Questions with Model Answers

1. **What are the two main steps of a proof by induction?**
   Answer: The basis step (prove the first case) and the inductive step (assume $P(k)$, then prove $P(k+1)$).

2. **How is recursion related to induction?**
   Answer: Recursion defines a process in terms of smaller instances; induction proves that the process works for all instances.

3. **Give a real-world example of a recursive process.**
   Possible answers: Folding paper, Russian nesting dolls, family trees, the Tower of Hanoi, or fractal growth in nature.

4. **Can every recursive definition be proven correct using induction?**
   Answer: Yes—provided it terminates and is well-founded. Induction is the formal method used to prove the correctness of recursive definitions.

1.9 Teaching Discussion Prompts

- "Where do we see self-reference in the real world?"
- "If recursion builds, what does induction guarantee?"
- "What happens if a recursive function lacks a base case?"
- "How would you prove that your recursive function always terminates?"

1.10 Common Misconceptions

- **"Induction means guessing and checking."** $\rightarrow$ Clarify that it's logical deduction, not pattern recognition.
- **"Recursion runs forever."** $\rightarrow$ Only without a base case! Stress convergence and termination.
- **"Strong induction is different math."** $\rightarrow$ Emphasize it's the same principle with an expanded hypothesis.

1.11 Classroom Activity Ideas

1. Use Jupyter or Python to demo factorial recursion visually.
2. Have students form a "human call stack" — each student represents a recursive call.
3. Challenge groups to come up with non-math examples of recursion.
4. Close class by proving a fun pattern like $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ inductively.

1.12 Instructor Reflection

Induction is belief with a proof.

Reflect on how you framed induction not as a dry method but as a way of thinking. Students who “get it” here will see recursion everywhere—from fractals to AI to data structures.

Next Chapter: Recursive Algorithms — Turning Thought into Code

Chapter 2

ch02 recursive algorithms solution

This is a placeholder for `ch02`$_{r}ecursive_{a}lgorithms_{s}olution.tex$.

Chapter 3

ch03 fibonacci solution

This is a placeholder for ch03$_fibonacci_solution.tex$.

Chapter 4

ch04 measuring recursion solution

This is a placeholder for ch04$_{m}easuring_{r}ecursion_{s}olution.tex$.

Chapter 5

ch05 big o through fib solution

This is a placeholder for ch05$_{b}ig_{ot}hrough_{f}ib_{s}olution.tex.$

Chapter 6

ch06 memoization mind solution

This is a placeholder for ch06$_memoization_mind_solution.tex.$

Chapter 7

ch07 big o meets reality solution

This is a placeholder for ch07$_{b}ig_{om}eets_{r}eality_{s}olution.tex$.

Chapter 8

ch08 teaching tips and assessments

This is a placeholder for ch08$_{t}eaching_{t}ips_{a}nd_{a}ssessments.tex$.