Teaching Suggestions

In this part of the *Instructor's Resource Guide*, I suggest how the text can be used effectively to teach an introductory course in discrete mathematics. These views are based on my personal teaching experience as well as on the experiences of some of the many instructors who have used the text in previous editions.

In the following material I provide an overview of each chapter of the text. Along with a description of the contents of that chapter, I describe its importance in an introductory course. After this overview, I give detailed information about each section of the chapter. First, I state the section goals and identify the prerequisites for that section. Then, I give my suggestions on how to teach from the section. In particular, I identify troublesome concepts and suggest how to handle them. I point out particularly useful examples and important concepts. Finally, I describe exercises that I feel are noteworthy, especially those that tie together diverse concepts or introduce new ideas. I hope this information makes teaching from the text easier and more rewarding for you.

CHAPTER 1
The Foundations: Logic and Proofs

Overview: Chapter 1 begins with an introduction to propositional and predicate logic. It then continues with presentations of the rules of inference for both propositional and predicate logic. After developing logic and rules of inference, the chapter introduces an arsenal of proof methods. The chapter concludes with a discussion of proof strategy, as well as the process of formulating conjectures and then using proof methods and strategies to settle them. The material on logic and proof in this chapter provides the foundations needed throughout higher mathematics and computer science. Without a firm foundation in logic, students have a great deal of difficulty with this course and subsequent courses. If you are lucky enough to have students with strong backgrounds, you might be able to cover quickly, or even skip, some of the contents of this chapter. But be sure to cover what your students need from this chapter, or the rest of the course could be tough sledding.

The first five sections deal with logic; propositional logic is covered in Sections 1.1–1.3, and predicate logic is covered in Sections 1.4 and 1.5. Studying logic is the best way to start a course in discrete mathematics (unless, of course, your students already know this material) because students must be able to think logically and carry out precise reasoning. Section 1.6 introduces rules of inference, and Section 1.7 introduces basic proof methods. Section 1.8 introduces additional methods of proof and addresses key aspects of strategies for developing proofs. Take note that the proof methods discussed in Sections 1.7 and 1.8 are used throughout the text, particularly in the coverage of sets and functions in Chapter 2, and in the coverage of algorithms and number theory in Chapters 3 and 4. Chapter 5 introduces another key proof method, mathematical induction, together with its variants.

SECTION 1.1 Propositional Logic

Goals: To introduce the basic terminology of propositional logic, including logical connectives, to show how to construct truth tables, to illustrate the importance of logic with applications, and to motivate the study of logic through logic puzzles and system specifications.

Prerequisites: None.

Advice: The material on logical connectives is straightforward. Most difficulties with this material involve confusion between common English usage and precise mathematical definitions. In particular, students have trouble with inclusive versus exclusive *or*; make sure the distinction is clear (see Examples 6–9).

Stress the definition of a conditional statement, especially when the premise is false. That is, emphasize that $p \rightarrow q$ is false only when p is true and q is false. This material is used extensively in Section 1.6 when rules of inference are covered, in Section 1.7 when methods of proving conditional statements are discussed, and in Sections 5.1–5.3 when mathematical induction and its variants are covered. Go over the different ways conditional statements are expressed; these are listed after Definition 5. Define the converse, contrapositive, and inverse of a conditional statement (see Example 10); these terms are often confused with each other. Be sure to discuss the notion of equivalence of compound propositions. Explain that a conditional statement and its contrapositive are logically equivalent, whereas a conditional statement and its converse or inverse are not logically equivalent. Introduce biconditionals and how they are expressed. Mention that biconditionals are often implicitly stated. Briefly introduce truth tables; they are used extensively in Section 1.3. Also, quickly mention the precedence of logical operators.

Exercises: Exercises 49–51 introduce fuzzy logic, which is used in expert systems and artificial intelligence. Exercises 52–54 cover some logical paradoxes.

SECTION 1.2 Applications of Propositional Logic

Goals: To introduce some important applications of propositional logic, including many important applications in computer science. Also, to work with logic puzzles, which provide an entertaining way to learn and enjoy propositional logic.

Prerequisites: Section 1.1.

Advice: Cover the material on translating English sentences into logical statements; students often need help with this important task. The subsection on system specifications is of particular appeal to students in computer science and engineering; it shows that logic is of immediate practical importance. Computer science students are usually familiar with logical operators from their use in programming, so make the connection between the material in this section and logical operators used in programming languages. (In fact almost everyone has been forced to understand logical operations from doing Boolean searches on the web—see Example 6.) You may want to spend time covering the subsection on logical puzzles; many people find these puzzles fascinating—in particular, see Example 8, which introduces one of Raymond Smullyan's knights and knaves puzzles. A brief introduction to logic circuits is included here for instructors who want to make the connection between formal logic and logic circuits. (A thorough treatment of logic circuits is found in Chapter 12.)

Exercises: Exercises 7–12 are devoted to system specifications. Exercises 17–22 and 36–42 are logical puzzles that can challenge students. Exercises 23–27 are puzzles involving Smullyan's knights and knaves, and Exercises 28–35 are puzzles involving knights, knaves, and spies, also introduced by Smullyan.

SECTION 1.3 Propositional Equivalences

Goals: To show how propositional equivalences are established and to introduce the most important such equivalences.

Prerequisites: Section 1.1.

Advice: Introduce the notion of propositional equivalences by establishing De Morgan's laws—see Example 2. Table 6 presents basic propositional equivalences. We will see similar tables for set identities in Section 2.2 and for Boolean algebra in Section 12.1. Mention that these properties hold in a wide variety of settings that all fit into one abstract form. Many students have the tendency to just memorize the properties, so stress that it is more important to understand their meaning and why they are true. Explain the different ways that such propositional equivalences can be established, including by the use of truth tables, by showing that the propositions are true (or false) for precisely the same sets of values, and by using previously proved equivalences, including those in Tables 6–8. (Note: We discuss rules of inference formally in Section 1.6 and introduce proofs in Section 1.7. Some practice with straightforward proofs here will help motivate the in-depth coverage that will follow.) The concept of propositional satisfiability is introduced in this section. The section concludes with a discussion of how the n-Queens problem and Sudoku puzzles can be modeled as satisfiability problems.

Exercises: The exercise set introduces some new topics, including duality, disjunctive normal form, and functional completeness. Students can learn about duality by doing Exercises 38–43. Exercises 44–46 develop the concept of disjunctive normal form. Exercises 47–49 introduce the concept of functional completeness, and Exercises 50–58 introduce the operators *NAND* ($|$) and *NOR* ($\downarrow$) and show that the sets $\{|\}$ and $\{\downarrow\}$ are both functionally complete. Exercises 64–66 deal with satisfiability and Exercises 70–72 involve Sudoku.

SECTION 1.4 Predicates and Quantifiers

Goals: To introduce predicate logic, especially existential and universal quantification. Moreover, to explain how to translate between English sentences (or mathematical statements) and logical expressions.

Prerequisites: Sections 1.1 and 1.3.

Advice: This section is important because students often have trouble proving statements that involve quantification, including the inductive step in mathematical induction. Make sure they have a clear idea what the truth values of existential and universal quantifications mean. Tell students that a quantification is not well-defined unless the domain is specified and that changing the domain can change the truth value of the quantification. Mention that a statement of the form $\forall x\, P(x)$ can be shown to be false with a counterexample. Explain how to negate existential and universal quantifications (see Table 2). We will need this in Sections 1.7 and 1.8 when we discuss how to prove theorems that involve quantification (in particular, with existence proofs and counterexamples). Discuss the different ways to express universal and existential quantifications in English.

Devote special attention to the subsection on translating English sentences into logical statements; this is a particularly difficult task for many students. Be sure to stress that there is more than one way to translate a particular English sentence into a logical statement; see Examples 23 and 24. Example 25 illustrates how to use quantifiers in system specifications. Examples 26 and 27, taken from Lewis Carroll, illustrate the subtleties of translating English sentences into correct statements involving predicate and propositional logic. The subsection on logic programming shows that the material in this section is important in computer programming and AI.

Exercises: Exercises 23–28 provide a wide variety of examples of how quantifiers are used when English sentences are translated into logical statements. Exercises 32–36 deal with negations involving quantified statements. Exercises 40–44 cover the use of quantifiers to express system specifications. Exercises 48–51 introduce some useful logical equivalences called null quantifications. Exercises 54–56 deal with the uniqueness quantifier. Exercises 57–60 deal with Prolog. Exercises 61–64 are questions based on work by Lewis Carroll.

SECTION 1.5 Nested Quantifiers

Goals: This section explains how to work with nested quantifiers and makes clear that the order of quantification matters. This section helps students gain maturity working with complicated logical expressions involving multiple quantifiers.

Prerequisites: Sections 1.1, 1.3, and 1.4.

Advice: Describe how nested quantifiers work (see Table 1); you may find the analogy to nested loops useful. Use Examples 1–5 to illustrate the meaning of statements involving nested quantifiers. Cover Example 4 to illustrate that the order of quantification is important when several different quantifications occur in the same statement. Cover the process of translating mathematical statements into logical expressions involving nested quantifiers—see Examples 6–8. In particular, students who have studied the definition of limit should see Example 8. Then discuss the translation between complicated statements in English and logical expressions involving nested quantifiers—see Examples 9–13. Cover Examples 14–16 to illustrate how to negate logical expression involving quantifiers. In particular, Example 16 shows how to use quantifiers and predicates to express that a limit does not exist.

Exercises: Exercises 14–16 involve translating English sentences into logical expressions involving nested quantifiers. Exercises 17–18 involve translating system specifications into logical expressions involving nested quantifiers. Exercises 19–23 involve translating mathematical statements into expressions involving nested quantifiers, and Exercises 24 and 25 are about the reverse process. Negations of statements involving nested quantifiers are the subject of Exercises 36–38. Prenex normal form is introduced in Exercises 50–51.

SECTION 1.6 Rules of Inference

Goals: To introduce the notion of a valid argument and rules of inference for propositional logic. To explain how to use rules of inference to build correct arguments in propositional calculus. Moreover, to introduce rules of inference for predicate logic and how to use these rules of inference to build correct arguments in predicate logic. To show how rules of inference for propositional calculus and predicate calculus can be combined. Finally, to learn how to distinguish between correct and incorrect arguments.

Prerequisites: Sections 1.1, 1.3, 1.4, and 1.5.

Advice: Explain what it means for an argument form to be valid in propositional logic. Be sure to tell students that if the hypotheses in a valid argument form are not true, the conclusion of the argument may not be true. Introduce the basic rules of inference for propositional calculus—see Table 1. You may want to cover Examples 6 and 7, which show how to use rules of inference to construct valid arguments in propositional logic. Describe the rules of inference for quantified statements—see Table 2. Explain how rules of inference for propositional logic and predicate logic can be combined.

Mention begging the question; students will think this is not likely to occur, but you can show them examples from their own arguments.

Exercises: Exercises 23 and 24 ask students to find an error in an incorrect argument in predicate calculus. Exercise 26 asks for a justification of a rule of inference in predicate calculus called the rule of universal transitivity. Exercises 27–29 ask students to use rules of inference in predicate calculus to construct valid arguments.

SECTION 1.7 Introduction to Proofs

Goals: To introduce the notion of proof and basic methods of proof, including direct proof, proof by contraposition, and proof by contradiction. Furthermore, to learn how to distinguish between correct and incorrect arguments, and to understand and construct basic types of proofs.

Prerequisites: Sections 1.1, 1.3–1.6.

Advice: Begin with definitions of important terms, including *theorem*, *proof*, *corollary*, *lemma*, and *conjecture*. A key goal is for students to understand what constitutes a valid proof; they need to be able to understand existing proofs and create their own. Let them know that axioms and previously proven results can be used and that arguments must follow correct rules of inference for propositions and for predicates. (You may want to review the axioms for the real numbers in Appendix 1.) Students need to understand the difference between a formal proof and an informal one that could be expanded into a formal proof if necessary.

Spend substantial time showing how to prove conditional statements using direct proofs and proofs by contraposition; this will pay off when you discuss mathematical induction and whenever you prove theorems that are universal quantifications of conditional statements. Introduce some aspects of proof strategy that tell you when to try a direct proof and when to use an indirect proof (proof by contraposition or contradiction). Illustrate this by covering Examples 8 and 9. Be sure to spend adequate time discussing proof by contradiction. Example 10 illustrates a proof by contradiction and foreshadows the pigeonhole principle discussed in depth in Section 6.2. Explain what it means to show that statements are equivalent. Example 14 illustrates how the equivalence of three statements can be established.

Exercises: Assign some of Exercises 1–4, 6, 7, 9, 10, 17, 19, and 20 to give students practice with direct proofs, proof by contraposition, and proof by contradiction. In these exercises the method of proof is specified. Also assign some of Exercises 5, 8, 13, 14, 15, 16, and 18 to give students practice determining which method of proof to use. You may want to assign Exercises 11 and 12, which ask students to either prove or disprove a statement. Exercises 24–26 require proof by contradiction; these are really just examples of the pigeonhole principle. Exercises 28–30 ask students to show that two statements are equivalent. Exercises 32–35 and 43–44 ask for proofs that three or four statements are equivalent.

SECTION 1.8 Proof Methods and Strategy

Goals: To learn important methods of proofs including proof by cases and existence proofs, supplementing the basic methods introduced in Section 1.7. To introduce key strategies for proving theorems, to understand the roles of conjectures and counterexamples, and to learn about some important open problems.

Prerequisites: Sections 1.1, 1.3, and 1.4–1.7.

Advice: Introduce proof by exhaustion and proof by cases. You may want to cover Example 4, which shows how a proof by cases is used to prove that the absolute value of the product of two numbers is the product of their absolute values. Mention the notion of *without loss of generality* and how it can be used to simplify proofs that might need to consider separate cases. Cover some of the common errors that arise in incorrect proofs by cases.

Introduce existence proofs, and discuss the difference between constructive and nonconstructive existence proofs. Examples 11 and 12 provide good examples of nonconstructive existence proofs. Explain what is needed in a uniqueness proof, and use Example 13 to illustrate how a uniqueness proof proceeds.

The material presented here provides students with a window into what mathematics is really about. Explain some of the strategies used to find proofs of theorems. Explain that the proof methods studied in Section 1.7 and the first part of this section provide the tool kit, but the art of finding proofs is something altogether different. You can illustrate this by covering Examples 14 and 15. Also, be sure to cover Example 16, which illustrates how leveraging an existing proof can provide a good starting point for constructing a new proof.

Use the material on tilings to discuss the role of conjectures and how to use the proof methods developed in the text to settle them. This material requires no extra machinery,

so it is quite accessible. Formulating conjectures about tilings of checkerboards, and parts of checkerboards, is easy, but settling these conjectures ranges from straightforward to extremely tricky.

Devote some time to discussing the role of open problems. Students will find the story behind Fermat's Last Theorem compelling. Learning about easily understood conjectures that remain unsolved, such as the $3x + 1$ conjecture, also motivates many students.

Exercises: To give student practice with proof by cases, assign some of Exercises 2, 3, 5, and 6. Exercises 7 and 8 involve the notion of *without loss of generality*. Exercises 10–14 ask for existence proofs, together with an explanation why the proof is constructive or nonconstructive. Exercises 25 and 26 introduce the harmonic and quadratic means and give students practice with formulating and proving their own conjectures. Exercise 28 is an excellent example of working backwards. Exercise 35 provides an opportunity for students to adapt an existing proof. The famous three jug problem is the subject of Exercise 40, which asks students to prove or disprove that you can solve this problem. Exercises 43–52 asks students questions about tilings of checkerboards. Exercises 43–46 ask students to prove or disprove a statement about tilings; this provides practice analyzing whether a conjecture is true and which proof method to use to prove the conjecture or to show that it is false. Exercise 50 is a challenging exercise about tilings that has a particularly elegant solution.

CHAPTER 2
Basic Structures: Sets, Functions, Sequences, Sums, and Matrices

Overview: Chapter 2 presents an introduction to basic discrete structures, namely sets, functions, sequences, summations, and matrices. Some, but likely not all, of this material may be review for your students. You should quickly cover, or not cover at all, material that your students already know. However, be sure to cover topics that your students may not already know, such as set identities, countability, the floor and ceiling functions, and summation formulae.

SECTION 2.1 Sets

Goals: To introduce the basic terminology of set theory.

Prerequisites: Chapter 1.

Advice: Make sure students understand that when specifying the elements of sets the number of times an element is listed and the order in which the elements are listed do not matter. These facts are illustrated by Example 6. Students have trouble distinguishing between the sets $\emptyset$ and $\{\emptyset\}$, so explain that the empty set is the set with no elements and that it is a subset of every set. I like to start with the empty set and take the power set and then the power set again to force students to see the difference between the empty set, the set containing the empty set, and other confusing sets. You may want to present the proof of Theorem 1, which shows that every nonempty set has at least two subsets, the set itself and the empty set, especially because this is an excellent illustration of proof methods covered in Section 1.7.

Exercises: Russell's paradox is described in Exercise 50. This is a difficult exercise for students, but it is important since it shows that a consistent set of axioms is needed for set theory. Exercise 49 shows how to define ordered pairs in terms of sets.

SECTION 2.2 Set Operations

Goals: To show how set identities are established and to introduce the most important such identities.

Prerequisites: Chapter 1 and Section 2.1.

Advice: The relationship between set identities and logical equivalences becomes clear when set operations are expressed using set builder notation and logical operators. Show students several different ways to prove a set identity, namely by showing that each side is a subset of the other, by a membership table, by the use of logical equivalences, or by using set identities that have already been established. Explain that the set identities in Table 1 are analogous to the propositional equivalences in Section 1.3 and to Boolean identities that will be given in Chapter 12. We touch briefly on how to count elements in the union of two sets, foreshadowing the treatment of inclusion–exclusion in Chapter 8. The subsections on how computers represent sets and on multisets and their applications will be of particular interest to computer science students.

Exercises: The notion of the symmetric difference of two sets is introduced in the exercise set and studied in Exercises 38–49. Fuzzy sets, used in expert systems and artificial intelligence, are the subject of Exercises 73–75. You can ask your creative students to make the connection between fuzzy logic, introduced in Section 1.1, and fuzzy sets. Multisets are dealt with in Exercises 67–70. The successor of a set is defined in the preamble to Exercise 65. Jaccard similarity and distance are described in the preamble to Exercises 71 and 72.

SECTION 2.3 Functions

Goals: To introduce the concept of a function, the notion of one-to-one functions, onto functions, and the floor and ceiling functions.

Prerequisites: Chapter 1 and Sections 2.1 and 2.2.

Advice: We define functions as assignments and their graphs as the sets of ordered pairs determined by these assignments. As such, the graph of a function is a type of relation, a topic we cover in Chapter 9. Although functions are discussed in a general setting, most of the examples deal with functions from one discrete set to another, as is appropriate for a course in discrete mathematics. Sometimes students have trouble with the definitions of one-to-one and onto functions. Use Figure 5 to help make these concepts clear. Show students how to express the definitions of one-to-one and onto in terms of quantifiers. Make sure your students have a clear understanding of the floor and ceiling functions; there is often confusion about their values at negative real numbers. Examples 29 and 30 show how these functions are applied in basic problems in data communications. Table 1 displays useful properties of the floor and ceiling functions. Make sure students are familiar with these properties. Proving results about the floor and ceiling functions provides more practice with methods of proof. Example 33 illustrates computations with the factorial function; be sure to cover this if your students are not already familiar with factorials. Finally, the notion of a partial function, important in the study of Turing machines, is introduced.

Exercises: Exercises 48–59 give students the opportunity to work with the properties of the floor and ceiling functions, and Exercises 60–63 involve application of these functions to simple calculations in data communications. Exercise 74 asks students to show that the notions of one-to-one and onto are equivalent when the domain and codomain are finite sets of the same size. Exercise 81 establishes some important facts about the cardinality of finite sets and Exercise 82 establishes an important result about infinite sets.

SECTION 2.4 Sequences and Summations

Goals: To introduce terminology used for sequences and summations. To introduce recurrence relations and some methods for solving them. To work with summations and establish several important summation formulae.

Prerequisites: Chapter 1 and Section 2.3.

Advice: The first part of this section deals with sequences. Recurrence relations are introduced and the method of iteration for solving them is discussed. Example 11 illustrates how recurrence relations are used to solve a problem involving compound interest. The topic of integer sequences is covered, which requires more critical and creative thinking than the other material. Examples 12–16 involve conjecturing a formula or rule for generating the terms of a sequence when only the first few terms are known. Encourage students to try the *On-Line Encyclopedia of Integer Sequences*, mentioned in this section. Students should also understand that sequences and strings are just special types of functions.

The second part of the section introduces summation notation. Make sure students can work with the different forms of this notation and with shifting indices in summations. In particular, this will be helpful later when we prove summation formulae using mathematical induction.

Exercises: Exercises 9–10 ask students to conjecture the formula or rule for generating the terms of a sequence from the first few terms; these exercises are more challenging than Exercises 5–6, which ask students to list the terms of sequences defined in different ways. Exercises 7–8 are interesting since they point out that there are many different naturally arising sequences that have the same initial terms. Exercises 18–24 deal with solving problems using recurrence relations. Telescoping sums are defined in Exercise 35 and are used to find the sums of the first n positive integers and the squares of these integers in Exercises 37 and 38, respectively. Product notation is introduced in the exercise set. Assign Exercise 45 if you wish to cover this.

SECTION 2.5 Cardinality of Sets

Goals: To master the concept of the cardinality of sets. In particular, to understand the difference between countable sets and uncountable sets.

Prerequisites: Chapter 1 and Sections 2.1–2.3.

Advice: In general, the material in this section is more difficult than earlier material. Use the idea of Hilbert's Grand Hotel to explain the concept of countable sets. Students find this idea quite helpful when learning about countable sets. Showing that the set of positive rational numbers is countable is covered in Example 4. The proof that the set of real numbers is uncountable, using the Cantor diagonalization method, is elegant and quite subtle; it is given in Example 5. Motivate this by using a numerical example for the construction of a real number that was not listed. Mention the notion of computable and uncomputable functions and explain why uncomputable functions exists. Finally, better students may be fascinated by the continuum hypothesis, briefly described at the end of the section.

Exercises: Exercises 5–9 ask questions about finding rooms for newly arriving guests at Hilbert's Grand Hotel. Exercises 15 and 16 ask students to show that a set containing an uncountable set is also uncountable and a subset of a countable set is countable, respectively. Exercises 25–26 and 31–32 provide alternative methods of proving that the set of positive rational numbers is countable. Exercise 39 asks for a proof that there are functions that are not computable. Exercise 41 guides students through a proof of the Schröder-Bernstein theorem (Theorem 2 in the text).

SECTION 2.6 Matrices

Goals: To introduce basic properties of matrices and matrix arithmetic, including Boolean operations on zero–one matrices.

Prerequisites: Chapter 1 and Sections 2.3 and 2.4.

Advice: This section presents a brief review of the material on matrices needed in later sections of the text. Students should understand how matrix multiplication is defined and know that it is not commutative. The material on zero–one matrices and Boolean operations on them will be new to most students. This material is used only in Section 9.4, when the transitive closure of relations is discussed, and it may be omitted if you do not intend to cover that section.

Exercises: Make sure students know what a diagonal matrix is (see Exercise 14). You may want to assign Exercises 18–21, which deal with the notion of the inverse of a matrix.

CHAPTER 3
Algorithms

Overview: Section 3.1 introduces the concept of an algorithm. The purpose of this material is to ensure that students understand what an algorithm is and the different ways algorithms are expressed. The section illustrates the concept of an algorithm by covering searching and sorting algorithms. The notion of a greedy algorithm is also introduced. Section 3.2 introduces asymptotic notations used to describe the growth of functions, including big-O, big-Omega, and big-Theta notations. Section 3.3 describes how to express the complexity of an algorithm. This is important since later chapters discuss a variety of algorithms and their complexity. The notion of an algorithm paradigm is also discussed in Section 3.3. Appendices 2 and 3 are relevant to this chapter. In particular, Appendix 3 introduces pseudocode, which is used throughout the chapter, and Appendix 2 reviews exponential and logarithmic functions, which are important in the discussions of complexity.

SECTION 3.1 Algorithms

Goals: To introduce the concept and basic properties of an algorithm.

Prerequisites: Chapters 1 and 2.

Advice: The algorithm for finding the largest element in a finite sequence of integers provides a good example of an algorithm since it is simple and it solves a useful problem. Students should understand the steps used in actually solving a problem. First we find an algorithm, which is expressed initially in English and then in pseudocode. Next, we study the complexity of the algorithm. Then we construct a computer program to implement it. Finally we verify the correctness of the program. We concentrate on the mathematical portions of the study of algorithms in the text, namely, how to solve problems using algorithms (in this section), how to study their complexity (in Section 3.3), and how to prove them correct (in Sections 5.4 and 5.5).

Introduce the problems of searching and sorting and present the linear and binary searches and one or both of the bubble sort and the insertion sort. (We will study the complexity of these algorithms later on.) This may be a good time to introduce the notion of a greedy algorithm. The change-making algorithm presented here provides an easy introduction to this topic and the question of whether a particular greedy algorithm produces optimal solutions. Example 7 introduces the problem of the most possible talks that can be scheduled in a lecture

hall given their start and end times. This is a good example for examining different criteria to be used at each step.

The subsection on the famous halting problem, which computer science majors should see again in a Theory of Computation course, is optional. It is a beautiful example of proof by contradiction, but the argument is subtle (and gives students difficulties), partly because of its self-referential nature.

Exercises: A variant of the binary search algorithm is introduced in Exercise 26. This version of the algorithm stops if the middle term at any stage equals the desired integer. The ternary search algorithm is introduced in Exercise 27; this exercise gives students the opportunity to develop a search algorithm on their own, generalizing the binary search algorithm. The selection sort is introduced and studied in Exercises 43–44 and the binary insertion sort in Exercises 49–51. Exercises 60 and 62 ask for counterexamples that show that a particular criteria for each step of a greedy algorithm does not always lead to an optimal solution. The notion of a stable assignment is introduced in the preamble to Exercise 64, the deferred acceptance algorithm is introduced in the preamble of Exercise 65, and a vote algorithm is introduced in the preamble to Exercise 68.

SECTION 3.2 The Growth of Functions

Goals: To introduce big-O and related big-Omega and big-Theta notation, and to show how to estimate the size of functions using this notation.

Prerequisites: Chapters 1 and 2.

Advice: Students have trouble with big-O notation. Often they cannot decide how to choose the witnesses C and k in the definition. Show them how different pairs of constants can be used as witnesses. Give several different examples to illustrate the concept. Show how the definition of this notation involves the use of existential and universal quantifiers. Cover Examples 5 and 6, which give estimates for the sum of the n smallest positive integers and for $n!$, respectively. Go over the useful big-O estimates for logarithms, powers, and exponential functions; these provide useful guides for comparing the growth of common functions. Cover big-Omega and big-Theta notation and discuss the connections between them and big-O notation. Indicate the importance of big-O in estimating the complexity of algorithms. We will study this formally in Section 3.3.

Exercises: Exercises 21 and 22 ask that a list of functions be ordered so that each is big-O of the next function on the list. Another type of asymptotic notation is introduced in the exercise set—little-o notation, which depends on the notion of a limit. If your students have a satisfactory background working with limits you may want to assign some of Exercises 63–71, which deal with this concept. We will use the result in Exercise 74 when we use trees to study the complexity of sorting algorithms in Section 11.2.

SECTION 3.3 Complexity of Algorithms

Goals: To introduce computational complexity analysis.

Prerequisites: Chapters 1 and 2 and Sections 3.1–3.2.

Advice: This section deals with complexity of algorithms. This is an important mathematical part of computer science. We define different types of complexity but concentrate on time complexity. Explain the distinction between worst-case and average-case complexity. Tell students the merits, as well as the drawbacks, of using big-O estimates. Explain how the witnesses C and k in a big-O estimate have practical implications. Because average case complexity depends on notions of probability, a topic not formally studied until Chapter 7, tell students in an informal

way how average-case analysis depends on the distribution of input values. The complexity of matrix multiplication is studied and the problem of determining the best order for matrix-chain multiplication is introduced—we return to this problem in the exercises of Section 9.1. The notion of algorithmic paradigms is introduced in this section and brute-force algorithms are discussed. (The algorithmic paradigm of greed was introduced in Section 3.1; among the other algorithmic paradigms covered in the book are divide-and-conquer and dynamic programming in Chapter 8 and backtracking in Chapter 11.) I suggest giving students an informal introduction to tractable, intractable, solvable, unsolvable, NP, and NP-complete problems. A more formal treatment of these topics can found at the conclusion of the last section of the last chapter of the text.

Exercises: Exercises 1–4 ask students to give big-O estimates when various segments of algorithms, expressed as blocks of pseudocode, are carried out. Exercise 12 develops a big-Theta estimates for the number of steps used by an algorithm; the big-Omega part of this is harder than the big-O part. You may want to assign Exercise 14, which deals with Horner's method for evaluating polynomials. Have students compare the complexity of this algorithm with the conventional method described in Exercise 13.

CHAPTER 4
Number Theory and Cryptography

Overview: Section 4.1 introduces some basic notions of number theory, including divisibility of integers and congruences. Section 4.2 introduces base b representations of integers (including binary, octal, and hexadecimal) and presents algorithms for integer arithmetic. Primes are discussed in Section 4.3, including conjectures about primes. Section 4.3 also introduces greatest common divisors and the Euclidean algorithm. The fundamental theorem of arithmetic is also introduced in Section 4.3. In Section 4.4, we see how to solve linear congruences. We also see how to solve systems of linear congruences using the Chinese remainder theorem. Section 4.5 presents several important applications of congruences, namely pseudorandom number generation, hashing, and check digits. Finally, Section 4.6 provides an introduction to the basic ideas of cryptography. In this section, both classical and modern cipher systems are studied. Public-key cryptography and two important cryptographic protocols—key exchange and signed messages—are studied.

SECTION 4.1 Divisibility and Modular Arithmetic

Goals: To introduce some fundamental concepts from number theory, including the division algorithm, congruences, and the rules of modular arithmetic.

Prerequisites: Chapters 1 and 2.

Advice: Be sure you mention that what is called the division algorithm is not really an algorithm, because this is quite confusing. (We will present an algorithm that finds the quotient and remainder in Section 4.2.)

Explain the difference between congruence notation and the **mod** function. Cover the basic properties of congruences; we will need this material in Chapter 9 when we discuss congruence modulo m as an equivalence relation. Be sure to mention that working with congruences is similar to working with equalities, but that division of both sides of a congruence by the same integer may not produce a valid congruence. If you plan on covering recursive

algorithms in Section 5.4, be sure to cover Corollary 2, which is used to develop an efficient recursive algorithm for modular exponentiation.

You may want to cover the notion of arithmetic on $\mathbf{Z}_m$. This material will be useful to students who study abstract algebra in the future.

Exercises: Exercises 21–22 establish the relationship between the congruence notation and the **mod** function. Exercises 40–42 ask that certain results pertaining to congruences be established. These exercises give students some practice working with the notion of a congruence. Exercises 48–50 asks for proofs of properties of addition and multiplication in $\mathbf{Z}_m$.

SECTION 4.2 Integer Representations and Algorithms

Goals: To study representations of integers in different bases, including binary, octal, and hexadecimal representations, and to introduce algorithms involving integers based on these representations.

Prerequisites: Chapters 1–3 and Section 4.1.

Advice: If your students do not have practice using different bases for representing integers, spend some time on the discussion of such representations in this section. Show students how to convert from one base to another (see Algorithm 1). The algorithms for addition, subtraction, and multiplication of integers were the first procedures to be called algorithms. Students need to study this type of algorithm in order to understand how computers perform arithmetic. (Note: We will introduce a more efficient algorithm for multiplication in Section 8.3.) Performing modular exponentiation is important in cryptography; it is presented as Algorithm 5.

Exercises: The exercise set introduces other ways to represent integers, including those important in computer arithmetic. In particular, balanced ternary expansions are described in Exercise 30, one's complement representations are defined in the preamble to Exercise 40, two's complement representations are defined in the preamble to Exercise 46, binary coded decimal expansions are discussed in Exercise 53, and Cantor expansions are introduced in the preamble to Exercise 54. The simple conversions between binary, octal, and hexadecimal notations are the subject of Exercises 5–19. The complexity of modular exponentiation is the subject of Exercise 64.

SECTION 4.3 Primes and Greatest Common Divisors

Goals: To introduce some fundamental concepts from number theory, including primality, prime factorization, and greatest common divisors. To introduce some important conjectures about primes.

Prerequisites: Chapters 1–2 and Sections 3.1 and 4.1.

Advice: Students often do not see that when factoring an integer it is necessary to do trial divisions only by integers less than or equal to the square root of the integer being factored. This is emphasized in Example 3. Show how to use the sieve of Eratosthenes to find all primes less than a positive integer. Be sure to prove that there are infinitely many primes (Theorem 3); this is one of most elegant and famous proofs in mathematics. Briefly address the subject of primes in arithmetic progressions, addressing Dirichlet's theorem and the result of Green and Tao about arithmetic progressions of prime numbers. Be sure to mention the twin prime conjecture, the recent proof by of the bounded gap conjecture, and improvements to Zhang's bound. These demonstrate that new results are still being discovered and also illustrate how an open question can inspire work on similar or partial results. Discussing the search for new Mersenne primes (which can be monitored on the web) also illustrates that number theory is an active field. Briefly discuss conjectures about primes and some of the famous open questions about them, such as Goldbach's conjecture. New discoveries about prime numbers often find their way into the popular press and are the focus of many websites.

Introduce the greatest common divisor and least common multiple of two integers. Make it clear that using prime factorizations to find greatest common divisors is easy once these factorizations are known, but that factoring integers is extremely time consuming. Next, introduce the Euclidean algorithm. Besides being one of the oldest algorithms invented, it is an excellent illustration of the concept of an algorithm. We will study the complexity of the Euclidean algorithm in Section 5.3. (We defer the complexity analysis to that section because we will need an estimate for the size of Fibonacci numbers, which we will establish there.)

After showing that the greatest common divisor of two positive integers can be expressed as a linear combination of these integers, you can show that the prime factorization of an integer is unique (up to the order of the factors); the fact that every positive integer has a prime factorization is proved in Section 5.2.

Exercises: Exercise 11 asks for a proof that $\log_2 3$ is irrational; it is a simple, but challenging, exercise that follows from the fundamental theorem of arithmetic. The Euler ϕ-function is introduced in the preamble to Exercise 21. The extended Euclidean algorithm is covered in Exercises 41–45. Exercises 54 and 55 asks students to adapt the proof in the text that there are infinitely many primes to prove that there are infinitely many primes of the forms $3k + 2$ and $4k + 3$, respectively. Exercises 56 and 57 challenge students to present two different ways to show that the set of positive rational numbers is countable using material from this section.

SECTION 4.4 Solving Congruences

Goals: To learn how to solve linear congruences and simultaneous systems of linear congruences. To introduce Fermat's little theorem, pseudoprimes, primitive roots, and discrete logarithms.

Prerequisites: Chapters 1 and 2 and Sections 4.1 and 4.3.

Advice: Introduce the concept of a linear congruence. Explain what an inverse modulo m is and how to use inverses to solve linear congruences. Describe how to use the Euclidean algorithm to find modular inverses. Then introduce systems of linear congruences using Sun-Tsu's puzzle as motivation. Introduce the Chinese remainder theorem and explain the proof of the part of the theorem that asserts existence of a simultaneous solution. You may want to also show how to solve systems of linear equations by back substitution, besides using the construction in the proof of the theorem to find solutions. You may also want to illustrate how arithmetic with large integers can be carried out using the Chinese remainder theorem.

Present Fermat's little theorem and illustrate its use in computations. Introduce the notion of a pseudoprime and discuss the importance of pseudoprimes for finding large primes. You may want to discuss Carmichael numbers too. Introduce the notions of primitive roots. Discuss discrete logarithms, especially if you plan to cover cryptographic protocols in Section 4.6.

Exercises: Exercise 19 outlines a proof of Fermat's little theorem. Exercise 30 establishes the uniqueness part of the Chinese remainder theorem. Exercise 37 uses Fermat's little theorem to show that 341 is a pseudoprime to the base 2. Miller's test and the concept of a strong pseudoprime is introduced in the preamble to Exercise 44. Quadratic residues are introduced in the preamble to Exercise 58 and addressed in Exercises 58–64.

SECTION 4.5 Applications of Congruences

Goals: To introduce three important applications of congruences, which show the usefulness of number theory and also are important in their own right.

Prerequisites: Chapters 1 and 2 and Sections 4.1, 4.3, and 4.4.

Advice: Explain what a hashing function is and explain how to $h(k) = k$ **mod** m for hashing. Describe what a collision is and describe how to use a linear probing function to resolve collisions.