Instructor's Notes

Possible One-Semester Courses

Assumes roughly two sections per week

Math transition course

- 1.1, 1.2, 1.3, 1.4, 1.5
- 2.1, 2.2, 2.3, 2.4, 2.5
- 3.1, 3.2, 3.3
- 4.1, 4.2, 4.3, 4.5
- 5.1, 5.2, 5.3, 5.4
- 6.1, 6.2, 6.3
- 7.1, 7.2, 7.3, 7.4, 7.5

Computer Science core course

- 1.1, 1.2, 1.3, 1.4
- 2.1, 2.2, 2.3, 2.4
- 3.1, 3.3, 3.4, 3.5
- 4.1, 4.2, 4.3, 4.5
- 5.1, 5.2, 5.3, 5.6
- 6.1, 6.2, 6.3
- 7.1, 7.2, 7.3, 7.6, 7.7

Chapter 1

Section 1.1 is an overview of examples used in the rest of the book. The instructor should pick examples that will be relevant to the chapters he/she is planning to include in the course. Here is the correspondence:

- Example 1.1.1 concerns invariant properties. In Section 2.4, we use induction to prove that after *n* "shuffles" of any combination of these three types, the packet maintains specific properties that make this trick work.
- The Josephus problem (Example 1.1.2) builds awareness of inductive reasoning that will surface throughout the book. Properties about the Josephus problem are addressed with mathematical induction in the exercises of Section 2.3 and the text of Section 2.4.
- Practice Problem 3 raises awareness for organization in making a list, which will be important in Chapter 5. The idea is further developed when making row labels for a truth table in Section 1.2. Exercises that lead toward truth tables are 7, 8, 9 in Section 1.1.
- Example 1.1.3 is a classical graph theory problem that we will see again in Chapter 7.
- Practice Problem 5 concerns a two-player game of the type discussed in Section 7.3

Section 1.1 is typically used at the beginning of the course while students are still adjusting schedules, etc., so we try to develop some basic ideas without being too specific yet.

Section 1.2 serves three primary purposes: build an understanding of the difference between recursive and non-recursive models, lay the foundation for inductive reasoning as it applies to recursive models, and raise, for the purposes of review, the types of algebraic manipulations that the students will be required to do.

- Students discover closed and recursive descriptions of number sequences, and we try to help them see the difference in their thinking.
- Students learn that algebra is the main tool for discovering a recursive formula from a closed one.
- Students learn that finding a closed formula is often difficult. The only "method" we suggest is for them to compare a given recursive formula to another with similar recurrence whose closed formula they know. For example, when trying to find a closed form for $a_1 = 5$; $a_n = a_{n-1} + 3$, we would suggest they compare this to the sequence $b_n = 3n$, which has a similar recurrence relation.
- Students will have a chance to do inductive reasoning without the context of a proof. For example, if $a_n = a_{n-1} + 2n$ and $a_7 = 200$, what is a_6 and a_8?

The issue of proving a closed formula is correct, given a recursive description, is the main emphasis of the first examples of mathematical induction in Section 2.3.

Section 1.3 uses logic puzzles (of the liar/truthteller variety) to establish the "usefulness" of truth tables and logic notation. In this section, the logical connectives for **and**, **or,** and **not** are used while we get used to this new kind of abstraction and the truth table notation.

Section 1.4 focuses on "mathematical statements" of the form "for all *x*, if *x* has property *P*, then *x* has property *Q*." This raises issues of notation for implication, the logical meaning of an if-then statement, and, to a lesser extent, the notion of predicate. The logic and language of implication are the main focus in this section. Logic puzzles using if-then statements are given to round out the two-section discussion on the utility of truth tables for solving those puzzles.

It should be noted that even though we define predicate (in the interest of being honest about the difference between propositions and predicates, and so we can be precise about what we mean by "predicate" in the induction sections 2.3 and 2.4), this chapter does not have a detailed discussion of formal predicate logic. This would certainly be suitable for an additional section in this chapter. Please let us know if it is something you feel strongly ought to be here.

Section 1.5 is an overview of some formal propositional logic including the ideas of tautology and valid arguments. We feel that this is an interesting formalism of the way simple reasoning happens, but we do not think that this is an appropriate motivation for writing mathematical proof. Hence, there is little or referencing back to this section in the introductory material on mathematical proof in the next chapter. The instructor who *does not* wish to have mathematical proof follow formal propositional logic can skip this section entirely without losing the students.

Chapter 2

This chapter gives the foundation for mathematical writing. An important feature of this chapter is its **lack of dependence on** the discussion of truth tables and formal logic in **Chapter 1**.

Section 2.1 introduces the concept of mathematical proof from an intuitive point of view. There are several very important points in the development of proof in this (and the next) section.

- We do not use the truth tables from the previous section to justify the structure of mathematical proof. Our thesis is that the logic is natural but some of the concepts are abstract, so in our minds, basing proof on an abstraction of logic is counterproductive at this point.
- We prove only statements of the form, "For all x, if x has property P, then x has property Q," and we suppress the explicit universal quantifier.
- An early activity is to decide whether statements like "If n is odd, then $n^2 - 1$ is divisible by 4," or "If n is prime, then $2^n - 1$ is prime" are true or false. Students come up with their own idea that a false "if,then" statement is one for which there is an example that makes the hypothesis true and the conclusion false.
- With this foundation, contrapositive statements are naturally equivalent since they have the same requirements for a counterexample. We encourage students from early on to consider a statement and its contrapositive before deciding which to prove.
- Proofs are written with a specific audience in mind. Students imagine the Reader as a critical person who is searching for a counterexample to the given statement. (They understand the viewpoint of this person from their own activities in looking for counterexamples to "if,then" statements.) By writing for a particular audience, the language of mathematical proof is more natural to the student as the play the role of proof Author.
- It is significant that the first statement proven in the section is a non-trivial statement: "If n is a perfect square greater than 4, then $n - 1$ is not prime." Because this statement is not obviously true, students go through the process of looking for counterexamples to it before taking on the role of the Author to explain to the critical Reader why such a search is futile.
- Students are encouraged to read proofs the way the Reader would read them, by following the lines of the proof as if they are instructions. We ask students to trace proofs by using different numbers for the variables that occur.

Section 2.2 provides a setting in which the notion of proof can be practiced and developed in the context of divisibility of integers.

Section 2.3 introduces mathematical induction in the limited setting of finding closed formulas for (a) recursively defined sequences and (b) sums. We continue the emphasis on role playing by Author and Reader of the proof. This underscores a temporal aspect of induction (the Reader is considering a first statement, and then a second statement, and then a third statement, etc. in her search for a counterexample). A consequence of this point of view is that only the so-called "strong form" of mathematical induction is used. It is noted that often we only use the previous one or two statements in a proof, but we do not see a benefit of treating this case as if it is a different form of induction. Another feature of this section is our reasoning from the 1, 2, …, k-1 steps to the k step, as opposed to reasoning from k to $k + 1$. This is done for three reasons:

1. In problems involving closed formulas, there are fewer algebraic challenges with the former method. For example, we have found that it is easier for our students to **simplify** $(k - 1)^2 + 2(k - 1) + 1$ to k^2 than it is for them to **factor** $k^2 + 2k + 1$ to get $(k + 1)^2$.
2. In computer science courses, students will need to write recursive functions. In these functions, an input of size k is broken down into smaller pieces for subsequence function

calls. We want to foster the connection between recursion and induction in this course, so we once again prefer our approach.

3. Some students have difficulty with the scope of quantifiers, so if the write, "Let *k* be given and assume P(*k*) is true," it seems to them that they have just assumed what they are trying to prove, "For all *n*, P(*n*) is true." By providing a specific role for *m* (it is the index of the first statement not yet checked), we have not had this particular discussion with a student in the last two years of using this book.

Section 2.4 continues the discussion of mathematical induction. We first show that sums *are* recursively defined sequences, resolving the two types of problems in the previous section. We also apply induction to other statements like divisibility and inequalities. Finally we do induction proofs on games, most notably, following up on observations on the Josephus problem from Section 1.1 and on the magic trick that begins that section.

Section 2.5 introduces proof by contradiction. An important feature of the presentation is the emphasis on how not to use proof b contradiction. We find that students become so enamored with discovering a proof by approaching it "by contradiction" they often end up with direct proofs, or more commonly proofs of the contrapositive statement, without realizing it. In order to foster critical reading and thinking about proofs, we have examples and exercises where a proof by contradiction is given, and the student is asked to rewrite it without proof by contradiction. We also introduce the Pigeonhole Principle in this section. It is revisited in the Functions chapter in the discussion of the relationship between the sizes of two sets implied by properties of functions from one set to the other.

Section 2.6 is a real-world section on representing numbers in different bases. It allows us to discover and prove interesting things about numbers using the tools on divisibility and induction developed early in the chapter. *This section can be understood having only worked through Sections 2.1 through 2.4*. Applications include hexadecimal notation in computers and even a look at how binary numbers give information about the Josephus problem from Section 1.1.

Chapter 3

This chapter introduces sets and their operations, proofs about them, the generalization to Boolean algebra, and the application of these ideas to logic circuits. In **Section 3.1**, the basic operations of intersection, union, complement, Cartesian product, and power set are introduced. Definitions and relationships are reinforced through critical thinking exercises involving finding counterexamples to statements. Connections with counting (e.g.., product rule, inclusion-exclusion) are foreshadowed by investigations into the size of sets. Two types of proof are investigated in **Section 3.2**: element-wise proofs of the subset relation and proofs using "algebraic" properties of set operations. The latter type of proof is continued in the more general setting of Boolean algebras in **Section 3.3**. A major point of this development is to emphasize how the abstraction to Boolean algebra makes the work of proving set properties *easier*.

In **Section 3.4**, the Boolean algebra of logic circuits, complete with circuit diagrams is established. The Real World topic (**Section 3.5**) of Karnaugh maps gives an easy computational device for finding "minimal" circuits. This section is most likely only of interest to computer science students.

Chapter 4

This chapter discusses functions as mathematical objects and binary relations as the generalization of this concept. **Section 4.1** gives the definition of function and introduces the visualization tool of an "arrow diagram." It is noted that when one simply reverses the arrows of a function, one does not necessarily get the diagram of a function. This is the motivation for the discussion of general binary relations in **Section 4.2**. In **Section 4.3**, invertible functions are revisited and the methodology for proving standard properties (one-to-one and onto) of functions is discussed. Sections 4.4 and 4.5 are independent of one another and have very different focus, so the instructor might choose one or the other depending on the audience of the course.

Section 4.4 is a bit of departure from the formal to discuss some particular functions that are important in discrete math, like the discrete logarithm and the floor & ceiling functions. Many problems have to do with the length of certain large numbers, so there is a nice fit with Section 2.6 if it has been covered. **Section 4.5**, on the other hand, continues the discussion of binary relations in the direction of equivalence relations. In particular, we establish the equivalence of the "partition" definition and the "reflexive, symmetric, transitive" definition for equivalence relations, and along the way give tips about proving properties of relations. The "partition" version of the discussion comes up again in the context of counting in Section 5.3.

The Real World section (**Section 4.6**) in this chapter provides some explorations with iterating functions. Mathematically this only involves composition of functions and mathematical induction, so this can covered immediately after 4.1, if desired.

Chapter 5

This chapter discusses basic counting techniques. The emphasis is on thinking algorithmically about how to generate the objects being counted and then to apply the simple product or sum rule to this process. **Section 5.1** established the four types of structure we will be counting. These are "lists" in which (a) either order matters or not and (b) either repetition is allowed or not. The ordered lists fall quickly using the product rule in **Section 5.2**, and examples/exercises force students to apply these ideas along with the sum rule for breaking a problem into cases. The technique of "answering the complementary problem" is a side effect of this discussion. The notion of a partition (like with equivalence relations, although this discussion does not assume the student has covered Section 4.5) is used to establish the connection between combinations and permutations. Combinations and the Binomial Theorem are the highlights of **Section 5.3**, but the partition idea is the main focus, being further put into action to count permutations around a circular table and to count the number of arrangements of the letters in a word like PUZZLE. This idea (along with the algorithmic thinking and breaking into cases) makes these counting problems quite challenging.

Section 5.4 is primarily intended to finish up the search for formulas for all four types of list structure, the fourth one being unordered lists with repetition allowed (called *multisets* or *bags* in some books). The basic connection between the number of binary sequences of length n with k 1s and the number of k-element subsets of an n-element set is very important later on in the probability chapter, so this should be addressed even if you do not treat the full section. In a course which is emphasizing recursive processes, **Section 5.5** contains some nice applications of the idea to counting, and these ideas will arise again in the recursive games of Section 6.5.

Section 5.6 is a short overview of solving recurrence relations, a topic unearthed in Section 1.2, and addressed in Section 2.3, but never treated directly. In this section, we give a full treatment for finding closed formulas for sequences that have constant k^{th} differences for some k. (The solution involves the binomial coefficients, so this is the only reason for the placement of this section at the end of Chapter 5; otherwise this section could be covered right after induction in Chapter 2.) Next we show the general solution of a class of linear recurrence relations and give some applications of them. We also treat divide-and-conquer recurrence relations using proofs of inequalities, since this is the type of recurrence and the level of analysis expected in a computer science course on algorithms.

Chapter 6

This chapter discusses discrete probability and expected value. **Section 6.1** uses the simple definition of probability as a ratio in order tie probability in with the counting problems from Chapter 5. **Section 6.2** begins to depart from this direct connection by introducing rules for multiplying and adding probabilities. The concepts of independent events and disjoint events arise in this context. These rules and the fact that the number of binary sequences of length n with k 1s is C(n, k) are combined to establish the probability formula for Bernoulli trials in **Section 6.3**. Most of the examples in this section as well as in **Section 6.4** (on expected value) are taken from sports or games. **Section 6.5** is a nice recurrence of the idea of recurrence used to answer questions about probability / expected value for games that do not have a finite sample space (like a tennis game in which one player must "win by 2"). This section uses the same basic idea as the recursive counting section (5.5), but one is not technically

Chapter 7

This chapter discusses graphs and their applications. **Section 7.1** establishes some definitions in the historical context of the Euler bridge problem. The concept of isomorphism is treated informally but thoroughly since this is a key issue in understanding when two graphs (for example, two answers to the same question) are the same. **Section 7.2** extends the discussion of Eulerian graphs to include other traversal problems like Hamilton's dodecahedron puzzle and the traveling salesperson problem. Some naïve algorithms for this latter problem are given to establish familiarity with reading graph algorithms and to gain some understanding to the difficulty of the TSP in general.

Section 7.3 discusses the representation of graphs and establishes the important connection between directed graphs and adjacency matrices. **Section 7.4** revisits binary relations, summarizing the interconnections between directed graph, adjacency matrix, and binary relation. This discussion continues into another look at equivalence relations in **Section 7.5**. The focus on this section is on procedures for finding the reflexive, symmetric, and transitive closures of a given relation using operations on the adjacency matrix.

The chapter ends with two independent sections on trees. **Section 7.6**, focusing mainly on spanning trees in weighted graphs, treats trees as special graphs and includes many induction proofs about their structure. **Section 7.7** focuses on binary trees, which are presented as important structures in their own right. We give typical applications as well as the standard traversal algorithms, and we prove some global properties by induction on the height of the tree.

Chapter 8

This chapter includes additional "Real World sections" annotated with their prerequisite sections. These topics can be sued in place of the sections at the end of the chapters. Our intention in putting these at the end was not to de-emphasize them. Originally all of the real world sections were in this chapter, but we decided that as many as possible should be distributed throughout the text in order to emphasize that discrete math has important applications. The ones that remain in this chapter are simply those that seemed to us to be more specialized and hence of less general interest but certainly not of less importance.