CSCI 317: Discrete Structures for Computer Science

Prof. David M. Keil, Framingham State University, Spring 2014

Syllabus

Invitation

What mathematical concepts do *you*, as a computer-science major or a computing professional, need to know how to apply?

This course provides an environment for you to explore the connection between mathematical abstractions and practical computing.

Are you curious about: How we know what we know? How to tell when problems can't be solved? What solvable problems are not worth solving? What is most likely to happen in the future? What is orderly about chaos? What ideas are the foundations for all of computer engineering, software engineering, web design, mobile apps, and robotics? If these questions seem worth asking, join our inquiry.

Course description (FSU catalog)

An intermediate to advanced course on discrete mathematical structures used in computer science. Students study abstract structures used to represent discrete objects and the relationships between these objects. Course topics include principles of logic, incompleteness, diagonal proof, and inductive proof of correctness of simple algorithms. Students will write recurrences to define computable functions and will explore discrete probability and randomness from a computational viewpoint.

Prerequisites: CSCI 271 Data Structures and either MATH 215 Finite Mathematics or MATH 292 Discrete Mathematics.

Meeting times

Tuesdays, 6:30-9:50pm, in Hemenway Hall 225.

To contact me:

Office hours (Hemenway Hall 318A):
Mon. 10:30-11:30 a.m.; Tue. 5:30-6:30 p.m.;
Wed. 3:30-4:30 p.m.
Telephone: (508) 626-4724
Email: dkeil@framingham.edu
URL: *www.framingham.edu/~ dkeil*

I like talking with students about what we're studying. Even if everything is clear enough, please check in with me at least twice in the semester.

Basic reading

- Susanna Epp, Discrete Mathematics with Applications (Brooks/Cole, 2011).
- Slides and handouts (see also course web site)
- Text material on Discrete Structures available at *www.saylor.org/courses/cs202*

Text material is essential for explanations. The Epp book has lots of explanations and examples. We will rely on it and its exercises under nearly every topic.

Our inquiry

Each of the seven topics of this course will invite you to find answers to questions that matter. The first question is: What matters?

Topic 1 explores reasoning and proofs. How can we be sure that we know what we know? What is reasoning? Can software be verified reliably, independent of testing?

Topic 2 looks at sets and relations. What are languages? How can we measure the running time of any algorithm, without knowing what hardware it runs on? What is a technique for defining any computable function?

In Topic 3, we consider graphs, transition systems and models of computation. We will ask: How do packets find their destination on the Internet? What are the simplest imaginary computing machines like?

Trees (Topic 4) are mathematical structures designed for speed. What enables the fastest computations to be fast? Why are trees found almost everywhere in applications like artificial intelligence and bioinformatics?

In Topic 5, we enter a strange world of impossibilities. Aren't all infinite sets the same size? Are some problems not solvable? Is there any claim that is certainly true, but can't be proven?

Topic 6 is about possibilities and probabilities. How many arrangements of symbols or numbers are possible? Can we precisely predict or explain events?

In Topic 7, we bring together some advanced ideas that seem to matter. Will you agree that they matter? What near-certain results emerge from chaotic random processes? What's the best explanation for what we observe? What is information and how is it measurable?

These questions and others will frame our investigation of discrete structures.

How the course will deliver what it offers

My goal is to create a *natural critical learning environment*. In Discrete Structures, this means *defining concepts and proving mathematical facts about computing.* We all learn at our own pace, and we are all together in this class learning.

For each of the seven topics, I'll speak about the topic for a few minutes, with slides and with examples to compile, run, and discuss; we'll make space for group work; and we'll have in-class and out-of-class written exercises. *Practice exercises* and *quiz questions* help me track what students learn.

For each topic, we'll have two to four two-hour sessions. I'll ask you to solve some topic practice problems and to let me see your solutions by the end of the session *before* the last one on the topic. I'll look at them and provide comments.

In the *last* session on the topic, we'll have a review, with problem solving by students; a multiple-choice quiz; and a set of problems in quiz form.

For every topic, there'll be two or three more chances to solve problems in make-up quizzes. What I track is a student's *best* work on problems that assess course objectives.

See the essay, "What we do in my classroom."

Semester project

Each student will summarize the learning in this course in a project that will include proofs from exercises, will discuss applicability of some mathematical results in computer science, and will include research outside the course materials.

Learning outcomes

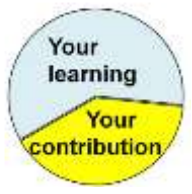
The course objectives are summarized and measured by several *learning outcomes* per topic; see next page. Some outcomes, mostly from Finite Math or Discrete Math, are *essential* for success. All students who pass the course will have shown success with these skills and capabilities.

I will work to help students reach the outcomes listed on the back page. *Outcomes* are a kind of very specific learning objective. Some of our outcomes are especially central; these I call "priority outcomes" and they are indicated by asterisks.

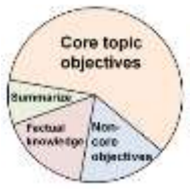
If you're tracking my opinions, it may help you to use that sheet as a score card, writing on it the numbers (from 1 to 4) that I write beside your quiz answers. Update these numbers as you answer questions on make-up versions of the quizzes; your highest score on an outcome is the one that matters.

Grading

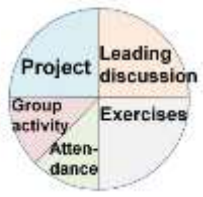
In evaluating the work of individual students, evidence of two kinds of accomplishment matter to me: *learning of course objectives*, and *contribution to the learning of others*. The graphs below show their relative importance to me and the relative importance of their components or methods of assessment.



Overall



Learning



Contribution

Rubric for assessment of course subtopic learning outcomes

I evaluate answers to quiz questions, as well as project work and some exercises, using the rubric below. 4 means, roughly, "excellent"; 3 means "good"; 2 means "OK"; 1 means "weak success." An answer without one of these scores is considered not yet successful; try again.

Code	Meaning	%
4	Solves problem thoroughly and accurately. Applies relevant concepts adeptly and insightfully. Fully supports claim of mastery of outcome.	100
3	A mostly successful solution with some omissions or errors. Generally accurate application of concepts. Gives strong support for claim of success with outcome.	87
2	Solution shows some grasp and application of relevant concepts, reflecting significant partial achievement of outcome.	73
1	A solution that shows some idea about relevant concepts, meeting minimum standards for outcome.	61

Learning objectives and expectations

The course is guided by the following objectives:

0a. Contribute to class activities throughout the semester
0b. Solve problems as part of a team
0c. Present results in the classroom
0d. Present written results
0e. Show knowledge of facts and concepts
0f. Summarize the semester's learning
0g. Relate mathematical concepts to applications

Topic objectives:

1. Explain and apply logical inference
2. Apply set-theoretic concepts, including the notions of relations, functions, and languages
3. Apply the basic notions of graph theory, including by means of proof
4. Prove properties of trees, describing their applications
5. Distinguish countable from uncountable sets, applying the diagonal proof method in number theory, logic, and computability
6. Apply the basic notions of combinatorics and discrete probability, including by proof
7. Explain connections between quantity of information and degree of randomness, comparing chaotic behavior to random behavior

For detailed outcomes, see page 5.

Math foundations of computer science

Computer science is a scientific discipline with empirical and theoretical aspects. This discipline is concerned with computation, the manipulation of symbols by machines. Computer programming and architecture are on the practical side. The theoretical aspect is a branch of mathematics. Its foundation is in discrete mathematics, the mathematics of logic, sets, and discrete quantities. This is a discrete math course.

Theoretical results in computer science are theorems about computations and about data structures relevant to computation. Theorems are mathematically proven assertions. For example, you will see that we can prove that an algorithm works.

The origin of computer science is in logic, because computers are machines that manipulate symbols using logic. Every component of a computer operates on truth values (bits).

This course builds on and applies concepts studied in discrete mathematics or finite mathematics courses. When you complete it, you will know some things for sure, because you will have proven them.

Discrete Structures helps prepare for work in verification and analysis of performance of algorithms and interactive processes, as well as in formal languages and models of computation, presented in presented in CSCI 347 (Analysis of Algorithms), CSCI 460 (Theory of Computing), and graduate courses. It provides a foundation for study in artificial intelligence, bioinformatics, and other applied areas.

Topics and examples in this course are chosen for their relevance to application in practical and theoretical computer science. For example, we focus on algorithms; on structures on which algorithms operate, such as graphs; and on the application of transition systems as models of computation.

Where the course material overlaps with CSCI 347 and CSCI 460, the emphasis in CSCI 317 is on the mathematics, especially theorems and proofs; whereas the emphasis in Algorithms is on design and time performance, and in Theory it is on the computing power of imaginary machines. In this course, the theorem and proofs are our focus, and in the other courses the focus is application.

The course is organized partly in a spiral model. That is, certain themes will be addressed repeatedly in the course, each time in a different context and at a more advanced level.

I organized the topics with the following reasoning. Logic (topic 1) is the foundation for all topics of this course, including proof methods. Sets and relations (topic 2) are the underpinning for recurrences and algorithm analysis (topic 2); for graphs and transition systems (topic 3); for trees (topic 4), a kind of graph; and for recursively defined functions (topic 5).
Tree concepts (topic 4) are used extensively in combinatorics and probability (topic 6). Probabilistic, uncertain processes (topic 6) are the objects of study in information theory, randomness, and chaos (topic 7).

Accommodations

"Students with disabilities who request accommodations are to provide Documentation Confirmation from the Office of Academic Support within the first two weeks of class. Academic Support is located in the Center for Academic Support and Advising (CASA). Please call (508) 626-4906 if you have questions or if you need to schedule an appointment." (See www.framingham.edu/CASA/ Accommodations/accomm.htm.)

Course Plan

Dates	**Topic or activity**	**Readings**
1/21	*Introduction to course and review of data structures and discrete-math / finite math*	Handout; Epp, Ch. 1-4; Sec. 6.4
1/28 – 2/4	1. Boolean algebras, logic, and induction	Epp, Sec. 6.1-6.4; handouts
2/11 – 2/19	2. Sets, relations, and recurrences	Epp, Ch. 5; Sec. 6.1; Sec. 8.1-8.5; 11.1-11.3; 12.1; handouts
2/25 – 3/4	3. Graphs and transition systems	Epp, Sec. 10.1-10.4; Sec. 12.2-12.3
3/4 – 3/11	4. Tree structures and their uses	Epp, Sec. 10.5-10.7; Sec. 11.4-11.5; handouts
3/11	*Review and make-up quizzes on topics 1-3*	
3/25 – 4/1	5. Countability and decidability	Epp, Sec. 6.4; Ch. 7; handouts
4/1 – 4/3	6. Combinatorics and discrete probability	Epp, Ch. 9; handouts
4/15	7. Information theory, randomness, and chaos	Handouts
4/22 – 4/29	*Summary and course review* *Make-up quizzes*	
Tues., 5/6	Final exam (student presentation) Optional objectives problems	

Ver. 1/28/14

CSCI 317 Discrete Structures subtopic outcomes

1. Boolean algebras, logic, and induction

____ 1.1a Describe the syntax of propositional logic**
____ 1.1b Apply the semantics of propositional logic**
____ 1.1c Apply logical inference**
____ 1.1d Explain Boolean algebras**
____ 1.2a Use a quantifier**
____ 1.2b Distinguish predicate from propositional logic**
____ 1.3a Write a direct proof**
____ 1.3b Write a proof by construction**
____ 1.3c Write a proof by contradiction**
____ 1.3d Describe the principle of mathematical induction**
____ 1.3e Use induction to prove a theorem about numbers**
____ 1.4a Explain concepts of algorithm correctness *
____ 1.4b Use induction to prove an algorithm correct*

2. Sets, relations, recurrences

____ 2.1a Explain or apply a concept in set theory**
____ 2.1b Prove a theorem in set theory**
____ 2.2a Describe a relation**
____ 2.2b Apply the notion of an equivalence relation*
____ 2.3a Describe a function**
____ 2.3b Define a class of functions
____ 2.4a Use a function to define a sequence**
____ 2.4b Define a language**
____ 2.5a Describe a recursively defined function**
____ 2.5b Write a recurrence to define a function
____ 2.6a Define O, Ω, and Θ notation
____ 2.6b Prove the correctness of a solution to a recurrence

3. Graphs and transition systems

____ 3.1a Construct a graph from a description**
____ 3.1b Describe a basic concept of graph theory**
____ 3.2 Apply the concept of graph isomorphism
____ 3.3 Describe a transition system*
____ 3.4 Use structural induction to prove an assertion about an expression or graph*

4. Trees

____ 4.1a Draw a tree with given specifications*
____ 4.1b Describe and prove a property of trees*
____ 4.2a Explain the running time of a tree-enabled algorithm
____ 4.2b Apply the Master Theorem to solve a recurrence
____ 4.3 Describe an AI or bioinformatics application of trees

5. Countability and decidability

____ 5.1a Prove that a set is countable*
____ 5.1b Prove that a set is uncountable*
____ 5.2 Describe the Incompleteness Theorem
____ 5.3 Explain how recursion captures computability*
____ 5.4 Prove that a problem is undecidable
____ 5.5 Define a non-well-founded set coinductively

6. Combinatorics and probability

____ 6.1 Solve a problem in permutations and combinations*
____ 6.2 Describe the relationship between combinatorics and intractable problems
____ 6.3a Describe a basic concept of probability theory*
____ 6.3b Prove a theorem in probability theory*
____ 6.4 Describe and apply Bayes' Theorem
____ 6.5 Describe a computational application of probability theory

7. Information theory, randomness, chaos

____ 7.1 Describe a way to quantify information*
____ 7.2 Define descriptional complexity and relate it to randomness
____ 7.3 Relate chaos, complex systems, and self-organization

* Priority objective
** Essential objective