Syllabus for CECS 419-519, Theoretical Computer Science

Instructor: Dr. Todd Ebert

Spring 2024, Last Update January 26th, 2024

General Course Information

Academic Unit Department of Computer Engineering and Computer Science, California State University, Long Beach

Prerequisites CECS 329 for CECS 419 and CECS 528 for CECS 519

Catalog Description Advanced topics in computability theory and computational complexity theory. Topics include Turing Machines, coverage of the standard complexity classes P, NP, IP, and PSPACE, hierarchy, decidability, and recognizability theorems. Additional projects required for CECS 519.

Section Call Number 11570 for CECS 419, 11571 for CECS 519 (Section 1)

Instructor Dr. Todd Ebert (Todd.Ebert)

Instructor Office Hours Tuesday, Wednesday, Thursday: 9:30-10:30 am in ECS 548.

Course Meeting Times Friday 8:00-10:45 am (VEC 418)

Required Textbook M. Sipser, "Theory of Computation", Cengage Learning, 2012, 3rd Edition, 978-1133187790

Optional Textbook U. Schöning and R. Pruim "Gems of Theoretical Computer Science", Springer, 1998, 978-3540644255

Course Website `http://www.csulb.edu/~tebert/teaching/spring24/419-519/intro.html`

Course Motivation

The goal for each meeting is to cover one of the "Gems" of Computer Science Theory. These are results that, in addition to being important and foundational for computer science, may also seem surprising and counterintuitive, and are capable of being understood in one or two weeks of study. Each gem is supplemented with lecture notes and HW exercises that help bring clarity and deeper understanding of the results.

Course Topics and Schedule

Week 1. Review of Computability Theory, including undecidability and the Diagonalization method. The Diagonalization method for proving the undecidability of computing problems is one of the oldest and beautiful gems of computer science theory.

Week 2. Kleene's 2nd Recursion Theorem: a somewhat surprising result that demonstrates how a program is capable of accessing its own Gödel Number, which has several applications to both the undecidability and complexity of computing problems.

Week 3. Information Theory and the Noiseless Coding Theorem. Claude Shannon's theory of information has several applications in computer science, including machine learning. We'll cover the fundamental concept of Entropy of an information source and show that, when encoding an information source, the average codeword length must equal at least the amount of entropy present in the source.

Week 4. Kolmogorov Complexity and Its Applications. The Kolmogorov Complexity of an object is defined as the size of the smallest program that can generate the object. We'll demonstrate how to use K-Complexity theory to compute complexity lower bounds for various structures and apply the theory to define a probability distribution for which the average and worst-case running times are the same for *all* programs.

Week 5. Random sequences and their applications. Thinking of an infinite random sequence as a sequence of 0's and 1's obtained from tossing a fair coin an infinite number of times, we first show that these sequences exist in the same way that the number π exists. We then show that any such sequence is i.o. autoreducible, meaning that there is a program that can infinitely often "guess" a bit of the sequence and every guess is correct despite there being a 50% chance of guessing incorrectly each time! We'll prove this with the help of Hamming codes which can be used to correct errors when information is sent through a noisy channel.

Week 6. Complexity Theory Review: Part 1. This lecture is dedicated to reviewing the basics of computational complexity theory including definitions of classes `P` and `NP`, and the process for proving that a decision problem is in class `NP`.

Week 7. Exam 1 (March 8th)

Week 8. Complexity Theory Review: Part 2. We finish our review of complexity theory by reviewing the concepts of mapping reducibility, `NP`-completeness, and Cook's theorem. We then give examples of interdomain reductions and discuss how they have played a role in identifying a diverse collection of thousands of NP-complete problems.

Week 9. Turing machines and Space Complexity. We review the Turing machine model of computation and use it to give a proof of Cook's theorem. We also provide an introduction to some space complexity classes and prove the surprising theorem that $\texttt{NSPACE}(n) = \texttt{co-NSPACE}(n)$.

Week 10. Probabilistic Algorithms Part 1. We define the complexity class BPP to represent problems that can be solved in polynomial time by having access to an infinite random sequence of bits and the likelihood that the answer is incorrect can be made arbitrarily small. We review some basic number theory and prove the somewhat surprising result that the problem of primality testing is in BPP.

Week 11. Probabilistic Algorithms Part 2. A branching program is a kind of model of computation that has important uses in both complexity theory and in practical areas such as computer-aided design. We prove the surprising result that the problem of checking the equality of two read-once branching programs is in BPP.

Week 12. Interactive Proofs and Zero Knowledge Part 1. We introduce the concept of an interactive proof system which leads to the definition of the complexity class IP. IP systems have several applications to cryptography and computer security. We also review the complexity class PSPACE and prove certain problems to be PSPACE-complete.

Week 13. Interactive Proofs and Zero Knowledge Part 2. We prove Adi Shamir's surprising and remarkable result that $\texttt{IP} = \texttt{PSPACE}$

Week 14. : The PCP Characterization Theorem. We prove the celebrated and counterintuitive result that every decision problem in NP has probabilistically checkable proofs of constant query complexity and logarithmic randomness complexity.

Week 15. : TBD

Week 16. Final Exam (May 17th, 10:15 am to 12:15 pm)

Reading Assignments

A reading assignment will be provided on most weeks of the semester. Reading the textbook will offer a somewhat alternative and more comprehensive viewpoint of the subject matter. Please check the "Reading Assignments" link for the current and past assignments.

Class Meetings

Homework assignments are due at the beginning of class. The plan is to record each lecture and make it available online. Attendance is not mandatory. Class participation points will be awarded for students who ask exceptional questions or provide exceptional answers during class.

Exams

There will be both a midterm and final exam, both worth 25% of the final grade. Students have the entire class period to complete each exam.

Homework and Writing Assignments

Homework will be assigned each week. Solutions to homework problems are found at the end of each lecture. Additionally, a writing assignment will be assigned each week and each writing assignment is to be turned in for grading. Please adhere to the following guidelines when submitting your work.

1. All writing assignments are due at 8:00 am one week after they are assigned.
2. Write your name and SID on all submitted pages.
3. Order your solutions/pages so that they follow the order of the assigned problems.
4. You may write on both the front and back of each submitted page.
5. Please do *not* staple your pages. Loose pages are easier to grade.
6. In lieu of 5, please make sure your pages are together when submitting your work.
7. All solutions should be presented in a clear step-by-step manner and include rationale.
8. Plagiarism will not be tolerated. Plagiarism comes in to main forms: copying the work of another student, and copying a solution found elsewhere. Depending on the evidence, suspected plagiarisim may result in one warning, any subsequent occurrence will result in a failing course grade.

Research Paper

Each CECS 519 student must pair up with another 519 student and complete a 5-10 page research paper that presents a gem of computer science theory not covered in the class. The paper should be based on at least two sources. A list of suggested gems will be provided at a later date. The goal of the paper is to explain the result in a way that aims to smooth out all the bumps encountered along the road to understanding the result. In other words, the paper should add value (in terms of detail and understanding) to what was already presented by the sources. Any paper for which there is strong evidence of plagiarism will result in the student receiving a failing final class grade.

Each CECS 419 student will then be required to read and critique at least two of the CECS-519 research papers and solve each of the homework problems provided in both papers.

Final Grade Determination

The four main assessments (Midterm, Final, Writing Assignments, and Research Paper) will each be worth 25% of the final grade. Each assessment will be assigned anywhere from 0 to 4 grade points, with 0 representing an F, and 4 representing an A. The final course grade will be determined by rounding a student's grade-point average to the nearest whole number.

In addition, students may earn extra-credit points as follows.

1. Identify errors (excluding grammatical and spelling) in the lecture presentation (5 pts each, first come first serve, must raise hand during lecture).
2. Identify errors (excluding grammatical and spelling) in the (non-annotated) original lecture notes (5 pts each, first come first serve, 10 pts for an error found in an exercise solution).
3. Identify an error in the solution to an exam or HW problem: 5 points.
4. Class participation points (5 points per instance) for asking/answering exceptionally good questions during lecture.

At the end of the semester, the class will receive a scale showing the number of awarded grade points (added to final GPA) for different ranges of extra-credit points.

Cheating

In addition to plagiarizing writing-assignment solutions and the research paper, plagiarized exam solutions (and other forms of cheating during the exam) will also result in a failing final course grade. Both exams are closed-note exams with no access to electronic devices. No exceptions!

Tips for Studying

1. Outside of the class meetings, study for at least 60-90 minutes, six days per week. Learning computer-science theory is similar to learning how to program or play an instrument. It requires daily practice in order to attain mastery. Studying every day helps break down the resistance to learning new and more complex topics as the semester progresses.

2. Make a study schedule and stick with it. In the evening write down your goals for the next day so that, while sleeping, your brain can work on how to achieve them.

3. Focus on reading the notes and textbook and working homework problems. Your final grade will largely be based on your ability to solve new problems that are similar to ones from the homework.

4. Whenever a problem uses a term that was defined in lecture, force yourself to spend at least one minute recalling the definition before looking it up. This is a very good exercise for your brain and will help you accelerate towards mastery. Remembering formulas, definitions, and examples has a functional effect on the brain that leads to increased skill and creativity towards the subject.

5. Keep a journal with all your attempted problem solutions. Review them and improve them when preparing for quizzes and exams. Learning is more cyclic than linear, in that we benefit from returning to past work and improving it based on our increased level of understanding.

6. Make sure to get 6-8 hours of sleep each night. Much of your learning takes place while you sleep!

7. Stay connected with other students in the class and form study groups.

8. For the sake of additional practice, use your creativity to create problems that are similar to the HW problems.