University of California at San Diego
Computer Science and Engineering
Course: CSE 200
Handout 1: Course Information
Instructor: Russell Impagliazzo Spring, 2008

CSE 200: COMPUTABILITY AND COMPLEXITY

Meets: M,W 3:30-4:50, CSE (EBU3b) 2154 **Instructor:**Russell Impagliazzo **Teaching Assistant** Kirill Levchenko

Office: 4248 CSE building (EBU3b)

E-mail: russell@cs.ucsd.edu

My Web Page: http://www-cse.ucsd.edu/users/russell

Course Web Page: http://www-cse.ucsd.edu/classes/sp08/cse200

Office Hours: TBA.

Texts: Sipser, Intro to the Theory of Computing, 2nd Edition. Available at UCSD bookstore.

Recommended: Garey and Johnson, NP-Completeness: A Guide to Intractibility

Recommended for those uncomfortable with proofs: Solow, How to Read and Do Proofs.

Purpose: This course will introduce you to computational complexity theory, which studies and classifies problems according to the resources required to solve them. In particular, we will try to understand what problems are "easy," what are "hard," and how different problems relate to each other. We will attempt to relate both the known results and the unsolved problems from complexity theory to various areas of computer science. In particular, we will show how computationally hard (intractible) problems are an obstacle to many areas of computing, but are actually useful for computer security. We will examine how various answers to the P vs NP related questions would radically change the nature of computer science.

Outline:

1. Models of computation: Turing Machines and variants, RAM model, Boolean circuits. Simulating one machine model with another. The Church-Turing thesis and versions for efficient computation. (Chapter 3, 7.1, 7.2, 9.3, 5 lectures).

2. Universal machines and diagonalization. Undecidability. Recursive enumerability. The Halting Problem. Time and Space Hierarchy Theorems. (Chapter 4, 5, 9.1, , 4 lectures, but intermingled with the above.)

3. NP-completeness and the P vs NP question. Consequences of $P = NP$ and the polynomial-time hierarchy. (5 lectures, Chapters 7.3,7.4,7.5)

4. Space Complexity, (2-3 lectures, Chapter 8)

5. Randomized computing (Chapter 10.2, 2 lectures)

6. Time permitting: Some of quantum computing, average-case complexity, the foundations of Cryptography (Chapter 10.6) and/or parallel complexity classes. (up to 3 lectures)

Prerequisites : We assume some undergraduate exposure to algorithms and their analysis, boolean logic, formal language and automata theory, and, most importantly, the ability to read, recognize and write a valid proof. If you have done well in CSE 105 (Theory of computation), CSE 101 (Design and analysis of algorithms) and CSE 20 (Discrete mathematics), or equivalent, you should have the necessary background for this course. If your preparation in these areas is weak, you may need to do some extra work to catch up. Please talk to me about this.

Assignments There will be four homework assignments, which may be done in groups of up to 3, and an individual take-home exam. In addition, there is an optional calibration homework, which will be graded but has no impact on your final grade.

Evaluation: Homework will account for 50 % of the grade, and the final will account for the remaining 50 % of the grade.

Grading Policies Students will be allowed to solve all homework assignments in groups of size up to 3. All group members should be acknowledged on the first page of the assignment. Group members are responsible for participating in the solution to all problems submitted, unless a note is attached to the assignment, specifying which problems a group member was not involved with. (Non-participating members will not get credit for the problem, but will also not be responsible for academic honesty for that problem.)

Students should not look for answers to homework problems in other texts, from people other than their study group and the professor, or on the web. However, students may use other texts as a general study tool, and may accidentally see solutions to homework problems. In this case, the student should write up the final solution without consulting the text or other source, and should give an acknowledgement of the text or other source on the first page of their solutions. Such a solution may be given partial or no credit if it too closely follows the text.

Be sure to follow the following ethical guidelines:

1. Do not discuss problems with people outside your group (except me and the TA, of course).
2. Do not share written solutions or partial solutions with other groups.
3. Prepare your final written solution without consulting any written material except class notes and the class text.
4. Acknowledge all supplementary texts or other sources that had solutions to homework problems.
5. Each member of the group bears responsibility for the honesty of all parts of an assignment. If you did not contribute to a particular part of an assignment, you must acknowledge this on the first page of the assignment. In that case, you will not recieve credit for that part, but you also will not have any responsibility for dishonesty regarding that part.

Standards for evaluation. **THIS IS IMPORTANT!!!** Grading of all problems (homework and exam) will be both on the basis of correctness and on logical consistency and completeness, i.e., "mathematical style". It is your obligation to provide a compelling argument that forces the reader to believe the result, not just notes from which a proof theoretically could be reconstructed. Describe your answer in a way you might present this material to a skeptical undergraduate honors class.

Some relaxation of this rule will apply to problems of a computational nature, where you are merely expected to present a solution and give some informal justification. Such problems will be designated by key phrases such as "Find a solution and justify your answer."

A few problems will involve performing experiments using your own and others' programs. The standards for these assignments involve performing a meaningful experiment, and clearly documenting the process and results. This involves describing the algorithm implemented, describing the architecture and environment that the programs were run in, acknowledging any code used written by others, and presenting a clear summary of timing and other data. Do not hand in code; we will not look at it. We may ask you to give a personal demo of your program if something is odd.

Lateness Policy Late homework will be accepted until I give out an answer key and no later. So you have to be no later than me.

Reading Schedule You are expected to do a **lot** of independent reading for this course. Most topics I will be presenting are also in the text, and it is good to try to follow along using the above schedule. I may not give as many details as the text does for some topics, and may use slightly different notation. Some (but not all) topics will also have lecture notes on the class website.