Design and Analysis of Algorithms – CSE 101
**Basic Information: Spring, 2011**

**Instructor:** Russell Impagliazzo

**Class:** TT, 11:00-12:20, HSS 1330,

**Mandatory discussion section:** Wed. 1-1:50, Center 105

**101 Professor Office Hours :** Wed., Friday, 2:30-4, start in CSE 4248 (may move to bigger room).

**email:** russell@cs.ucsd.edu

**webpage:** www-cse.ucsd.edu/classes/sp09/cse101

**TA:** Qian Peng

**TA office Hours:** Thu 5-7PM at EBU3B room B250a

**Prerequisites:** CSE 21, CSE 100

**Text Books:** Johnsonbaugh and Schaefer, Algorithms. OR

Edmonds, How to Think About Algorithms.

You need at least one of these two textbooks. Preferably, each study group will have both available. I mark the reading JS for Johnsonbaugh and Schaefer or JE for Jeff Edmonds.

**Assignments** There will be a calibration homework (not for credit), four homework assignments, a mid-term exam, and a final exam.

**Evaluation:** Homework will account for 30 % of the grade, the mid-term, 30%, and the final will account for the remaining 40 % of the grade. The calibration homework does not count for credit. The best 3 out of 4 homework assignments will be counted, so each homework is worth 10 % of grade. There will be a practice mid-term; the mid-term grade will be the better of the practice and real mid-term grades. Sorry, no practice final.

**Ethics and Academic Dishonesty** In the past, there has been epidemic cheating in this class. For example, dishonesty caused 25 % of the class to fail in 1997. For this reason, some rather intrusive rules have been instituted.

Students will be allowed to solve and write up all homework assignments in groups of size up to 4. All names should appear on the assignment.

Members of a group are responsible for all parts of any assignment with their names on it. Problems should be solved by the group, not divided up between group members. Each member of a group should participate in discussions about each problem. The front page should be signed by each member of a group; this is interpretted as the statement: "I participated in discussion for each problem, and have read and understood the answers here, which are summaries of our discussion." If this statement is true, just sign your name. If you wish to modify this statement, write and sign the modified statement instead. If the statement is not true of some of the problems, add "except for problems ...". You will not receive credit for these problems, but you also will not bear responsibility for them.

Students should not look for answers to homework problems in other (i.e., other than the course texts and class notes) texts or other sources (e.g. Internet discussion groups or newsgroups). 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 this text or source, and should give an acknowledgement of the text or source on the first page of their solutions. Such a solution may be given partial or no credit if it too closely follows the source. **Not giving an acknowledement is academic dishonesty, and will be treated as such. This rule applies to any material found on the internet, and to conversations with or written material from other people, whether or not they are students in the class. However, it does not apply to material handed out in class or on the class web-page for this year, or to conversations with the instructor or teaching assistants.**

Be sure to follow the following guidelines:

1. Do not discuss problems with people outside your group (except during office hours, or with the TAs or myself).

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 sources that had solutions to homework problems.
5. All problems should be discussed by the entire group.

**Standards for assignments** Most assignments and exam problems will be mathematical or theoretical in nature, and will require you to prove your answer correct. Grading of all such 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 an argument could be constructed. In particular, correct formulas or pseudo-code are not a complete solution by themselves; their significance and the logic of their application need to be explained.

A typical assignment is to design an efficient algorithm for a given problem. When giving an algorithm, the following two things should **always** be included, unless the problem explicitly says not to: a correctness argument, showing why the algorithm solves the problem; and a time analysis, giving the order of the worst-case runtime (in $O$-notation).

One problem on each homework assignment will involve implementing an algorithm, and reporting time usage data on a variety of inputs (which will be either completely specified or specified as a distribution on random instances). This implementation may be done in any language, and be run on any machine. Your solution should only include a brief description of your program; in particular, we will not read actual code, so you needn't hand it in. You should hand in only a description of your program, specifying the basic algorithm used, any modifications that you made to this algorithm, the language used, the performance characteristics of the machine used, and timing information for the various inputs you ran the program on. Discuss whether the timing results seemed consistent with the asymptotic analysis; if not, what in your opinion is the reason? The TA or I may ask to see a demonstration of your program on other instances.

**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. I will also not accept homework after the first 10 minutes of the class it is due. Working on the homework is no excuse for missing class or not paying attention.

Reading Schedule We will not be able to cover every example on each topic in the text in class. (JS=Johnsonbaugh and Shaeffer, required text; JE= Jeff Edmonds' How to Think About Algorithms.) You are expected to read the other sub-sections independently. In particular, we will only quickly review the material in JE Chapter 1 = JS Ch.1, 2.1-4 and the basic data structures (JE Chapter 2&3 = JS Ch. 2.5, 2.6, 3). These should be familiar from CSE 100 and 21, but will be used heavily in this class. Reading this material in advance is a good plan.

To help you plan your reading, here is a tentative schedule of topics to be covered in class, and the corresponding sections of the text to be read. I reserve the option to change the schedule at a later point. You should find most of the material in EITHER of: JS= Johnsonbaugh and Schaefer, Algorithms OR JE= Jeff Edmonds, How To Think about Algorithms. (You don't need to read both textbooks, but sometimes it may be helpful to see things explained in different ways.)

1. Background: This is material we are only quickly reviewing. You should be familiar with this material from previous classes. If you are having trouble with this material, you will need to work much harder throughout the course to keep up. Order notation, time analysis, recurrence relations: ( JE Chapter 1, JS Chapter 2.3,2.4 ) Basic data structures: lists, arrays, graphs, trees stacks, heaps(JS: Chapters 2.5,2.6,3 JE: Chapters 2 and 5.1). This material should be covered in CSE 100 and 21. Read it and try some exercises. If you have **any** problems, go back and read the chapter thoroughly.
2. Basic iterative algorithms. Loop invariants and correctness proofs. Time analysis. (JE, Chapter 3; JS pp37-38, section 2.3). Examples: Largest sum consecutive subarray; depth-first and breadth-first search (JS Ch4.2, 4.3, JE 8.2, 8.4), topological sort (JS: Chapter 4.4, JE 8.5). (2 lectures).

3. Maximizing Efficiency in Iterative Algorithms. Using restructuring, pre-processing and data structures to get the most efficient versions of algorithms. Graph and integer representations. Using data structures such as lists, arrays, heaps and balanced search trees. Sorting (JS, Chapter 6), skylines, auction problem, maximum min-degree subgraph. (2 lectures).
4. Greedy Algorithms
(JE, Chapter 10; JS, Chapter 7) When do greedy algorithms work? Proof techniques for optimality of greedy algorithms. Examples: scheduling (JE, 10.2.1); Minimum spanning trees(JS, 7.2,7.3; JE, 10.2.3); Dijkstra's Algorithm ( JS 7.4; JE 8.3); Independent set of a tree ; others to be added. 3 lectures.
5. Recursive algorithms and their analysis. Correctness proofs by strong induction. Recurrence relations. (JE, Chapter 5,6, JS 2.4). Euclid's GCD algorithm (JE, Chapter 4.3), (1 lectures.)
6. Arithmetic and numerical algorithms. Euclid's GCD algorithm revisited. Amortized analysis of GCD. (1 lecture)
7. Divide-and-Conquer.
(JS, Chapter 5.) Examples: Mergesort(JS, 5.2 );Multiplication of large integers; all distances in balanced binary tree; closest pair of points (JS, 5.3); Quicksort(JS, 6.2); The analysis of some divide-and-conquer algorithms will require Lemma 2.4.15 of JS, which is also in Chapter 1.6 of JE. (3 lectures.)
8. Backtracking
(JS, Chapter 4.5, JE Chapter 11) I spend more time on this because Dynamic Programming can be viewed as a modification of Backtracking.
Examples: Independent set; n queens ((JS 4.5, JE 11.2.3), graph coloring, Hamiltonian Cycle (JS, 4.5), and addition chains. 2 lectures.
9. Dynamic Programming
, (JS, Chapter 8; JE Chapter 12) Examples: Fibonnacci numbers (JS, 8.1); Longest increasing sub-sequence ; Shortest paths (JE 12.2.7, JS 8.5); Matrix Multiplication (JE, 12. 2.5, JS 8.3); Edit distance=Longest Common subsequence(JE, 12.2.2, JS, 8.4); scheduling. 3 lectures.
10. Reductions and NP-completeness

(JS, Ch 10, 11; JE, Ch. 13) When can one type of problem "code" another; NP, a format for search problems; universal (NP-complete) search problems. Coping with intractibility. 2 lectures

**Assignment and Exam schedule** To help you plan, here is the tentative assignment and exam schedule:

1. April 5: Calibration homework (order, recurrences, simple algorithm analysis and correctness) due
2. April 19: Homework 1 (efficient versions of algorithms) due
3. April 27: Discussion section: practice mid-term (order, solving recurrences, algorithm analysis and correctness, data structures, greedy algorithms)
4. May 3 : homework 2 (greedy algorithms) due
5. May 11: discussion section: midterm (order, solving recurrences, algorithm analysis and correctness, data structures, greedy algorithms)
6. May 17 , homework 3 (divide-and-conquer and back-tracking) due.
7. June 2 , homework 4 (dynamic programming) due
8. June 7, Final exam.