Design and Analysis of Algorithms – CSE 101
Basic Information: Spring 2003

Instructor: Russell Impagliazzo

Class: Tuesday and Thursday, 2:00-3:20, Center 109;

Mandatory discussion section: Friday, 11:00-11:50, Solis 104

101 Professor Office Hours : WF, 2:00-4:00, in 4111 AP &M (may change)

email: russell@cs.ucsd.edu

webpage: www-cse.ucsd.edu/classes/sp03/cse101

TAs: Ari Frank, Jun Lu, and Barath Raghavan

TA Office Hours: TBA.

Prerequisites: CSE 21, CSE 12.

Text Books: Neapolitan and Naimipour, Foundations of Algorithms (2nd edition: "In C++"). (Covered chapters are 1-5, 7.6, 8.3, 9.3, 9.4 and all Appendices.)

Jeff Edmonds, Thinking About Algorithms Abstractly, Avaiable from class website. This is a work in progress. Some sections are missing, e.g, 5.2, 18. Please send comments to jeff@cs.yorku.ca. (In postscript. You'll need to print it yourself. Covered chapters are 1-3,4.3,8,10,11,15,16.)

Note: There are several topics where you can use either textbook. Use the one that makes most sense to you, or use both and "triangulate" on the ideas.

Using NN for Naimipour and Neopolitan, JE for Jeff Edmonds, the basic equivalences (in order of class schedule) are:

1. NN Chapter 1 and Appendix A, B = JE Chapter 1.
2. NN Chapter 5 = JE Chapter 15
3. NN Chapter 3= JE Chapter 10
4. NN Chapter 4= JE Chapter 16

This means the minimum you need to print from JE is : Chapters 2-4, 8 and 11.

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

Evaluation: Homework will acount 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 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, with the TAs or me).
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.

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).

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."

One problem on each 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 TAs 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.

Reading Schedule We will not be able to cover every example on each topic in the text in class. You are expected to read the other sub-sections independently. In particular, we will not cover the material in Chapter 1 explicitly, since it should have been covered in CSE 12 and 21. We will be using it extensively, however; so reading this chapter 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. NN= Neapolitan and Naimipour, Foundations of Algorithms. JE= Jeff Edmonds, Thinking about Algorithms

1. Background: Order notation, time analysis, recurrence relations: (NN, Chapter I and Appendices A, B.1-3 and/or JE Chapter 1) This material should be covered in CSE 12 and 21. Read it and try some exercises. If you have **any** problems, go back and read the chapter thoroughly. Basic data structures: lists, arrays, hash tables (NN: Section 8.4, JE: Chapters 2 and 5.1). These should have been covered in CSE 12. All of this material is important and none of it will be explicitly covered in class.
2. Basic iterative algorithms. Loop invariants and correctness proofs. Time analysis. (JE, Chapter 3). Examples: Euclid's GCD algorithm (JE, Chapter 4.3), Sorting (JE, Chapter 3.2), Depth-first search (JE, Chapter 8.4). (2 lectures).
3. Maximizing Efficiency in 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, heaps (NN, section 7.6), and B-Trees (NN,section 8.3). Heapsort (JE, 5.1 or NN 7.6), skylines, auction problem, min degree subgraph. (2-3 lectures).
4. Recursion vs. iteration, proving recursive algorithms correct, analyzing with recurrence relations (JE, Chapter 11). 1 lecture.
5. Divide-and-Conquer.
(NN, Chapter 2). Examples: Mergesort(NN, 2.2);Multiplication of large integers(NN, 2.6.2); all distances in balanced binary tree; closest pair of points; Quicksort(NN, 2.4); Depth-first search in trees; general comments on this technique are in sections NN, 2.3,2.7, 2.8. The analysis of some divide-and-conquer algorithms will require Theorem B.5 of appendix B, p. 492. 2-3 lectures.
6. Backtracking
(NN, Chapter 5 and/or JE Chapter 15) I'm doing this out of order because Dynamic Programming and Greedy Algorithms can be viewed as a modification of Backtracking.
Examples: n-queens problems (NN, 5.2) as a special case of Independent set (not explicitly mentioned in text, but see the clique problem defined in 9.4) ; graph coloring (NN, 5.5) ; Hamiltonian Circuits (NN, 5.6), and addition chains. Do NOT Read NN, 5.3, or NN, 5.7. 3 lectures.
7. Greedy Algorithms
(NN, Chapter 4 and/or JE Chapter 10). Examples: scheduling (NN, 4.3); Minimum spanning trees(NN, 4.1); Dijkstra's Algorithm (4.2); Independent set of a tree (not in text); others to be added. 2 lectures.
8. Dynamic Programming
, (NN, Chapter 3 and/or JE Chapter 16) Examples: Longest increasing sub-sequence (not in text); Shortest paths (NN, 3.2);Chained Matrix Multiplication (NN,3.4); Edit distance(not in text); scheduling. General comments on the method are in NN, 3.3. 3 lectures.
9. Reductions and NP-completeness
(NN, Chapter 9), 2 lectures. When can one type of problem "code" another; NP, a format for search problems; universal (NP-complete) search problems. Coping with intractibility.

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

1. April 10: Calibration homework (order, recurrences, simple algorithm analysis and correctness) due
2. April 22: homework 1 (efficient versions of algorithms) due
3. May 2: discussion section: practice mid-term (order, solving recurrences, algorithm analysis and correctness, data structures, divide-and-conquer)
4. May 6: homework 2 (divide-and conquer) due
5. May 16, discussion section: midterm (order, solving recurrences, algorithm analysis and correctness, data structures, divide-and-conquer)
6. May 22, homework 3 (back-tracking,greedy algorithms) due
7. June 5, homework 4 (dynamic programming) due
8. June 11, 3-6 PM, final exam