Algorithms (CS 6500), Spring 2013

Time and Place: 2:00-3:15 Tuesday and Thursday in ITT 322

Web-site: **http://www.cs.uni.edu/~fienup/cs6500s13/**

Class Email List: Send messages to Google group for the course at CS-6500-01-spring@uni.edu

Instructor: Mark Fienup (fienup@cs.uni.edu)
Office: ITTC 313
Phone: 273-5918 (Home 266-5379)
Office Hours: M 9-11:45, 1:10-3; T 9:30-10:45; W 8-9:45; 1:10-3; Th 9:30-10:45; F 9-11:45

Prerequisite: Undergraduate algorithms course like CS 3530 (810:153). You must have a working knowledge of some procedural type programming language (C, C++, Ada, Pascal, etc.) that supports recursion. Technically, you should have successfully completed an undergraduate algorithms course (like UNI's CS 3530), but I'll try to make this course fairly self-contained. This necessitates some overlap/review with an undergraduate course. However, the pace and depth of duplicated topics will be more demanding here.

Goals: After this course, you should be able to (1) design solutions to problems by using classical problem solving techniques, (2) analyze an algorithm to determine its (asymptotic complexity) expected run-time behavior as the problem size grows, and (3) select an appropriate problem solving technique for a specific problem. These problem solving techniques are divide-and-conquer, dynamic programming, greedy algorithms, backtracking, and branch-and-bound. Additionally, you should understand the concepts of computational complexity, intractability, and the theory of *NP*. Time permitting we may have an introduction to parallel algorithms.

Text: *Foundations of Algorithms*, Richard Neapolitan and Kumarss Naimipour, fourth edition, 2011.
ISBN: 978-0-7637-8250-4.

Assignments: Assignments will consist of weekly or bi-weekly assignments that will consist of both written and programming parts.

Pedagogic Approach: In the first part of the course we will focus on being able to apply the problem solving techniques and fully understanding recursion. Then, we will change gears and work on analyzing the asymptotic complexity of the algorithms and selecting the appropriate problem solving technique for a specific problem. After this, we switch from analyzing individual problems to analyzing the computational complexity for classes of problems, such as sorting and searching. The remainder of the semester will be spent studying intractability, the theory of *NP*, approximation algorithms for *NP*-hard problems, and (time permitting) parallel algorithms.

Grading policy: There will be three tests (including the final). I'll announce tests at least one week in advance to allow you time to prepare. Tentative weighting of course components is:

In-class Work:	5 %	
Assignments:	32 %	
In-class Test 1:	21 %	(about Feb 21)
In-class Test 2:	21 %	(about April 4)
Final:	21 %	(Wednesday, May 8 from 1-2:50 PM in ITT 322)

Grades will be assigned based on straight percentages off the top student score. If the top student's score is 92%, then the grading scale will be, i.e., 100-82 A, 81.9-72 B, 71.9-62 C, 61.9-52 D, and below 52 F. Plus and minus grades will be assigned for students near cutoff points.

Scholastic Conduct: You are responsible for being familiar with the University' Academic Ethics Policies (http://www.uni.edu/pres/policies/301.shtml). Copying from other students is expressly forbidden. Doing so on exams or assignments will be penalized every time it is discovered. The penalty can vary from zero credit for the copied items (first offense) up to a failing grade for the course. If an assignment makes you realize you don't understand the material, ask questions designed to improve your understanding, *not* ones designed to discover how

another student solved the assignment. The solutions to assignments should be **individual, original** work unless otherwise specified. Remember: discussing assignments is good. Copying code or test-question answers is cheating.

Any substantive contribution to your assignment solution by another person or taken from a publication (**or the web**) should be properly acknowledged in writing. Failure to do so is plagiarism and will necessitate disciplinary action. In addition to the activities we can all agree are cheating (plagiarism, bringing notes to a closed book exam, texting during an exam, etc.), assisting or collaborating on cheating is cheating. Cheating can result in failing the course and/or more severe disciplinary actions.

Special Notices:

- In compliance with the University of Northern Iowa policy and equal access laws, I am available to discuss appropriate academic accommodations that may be required for students with disabilities. Requests for academic accommodations are to be made during the first three weeks of the semester, except for unusual circumstances, so arrangements can be made. Students are encouraged to register with Student Disability Services, 103 Student Health Center, to verify their eligibility for appropriate accommodations.

- I encourage you to utilize the Academic Learning Center's assistance with writing, math, science, reading, and learning strategies. There is no charge for currently-enrolled UNI students. UNI's Academic Learning Center is located in 007/008 ITTC. Visit the website at http://www.uni.edu/unialc/ or phone 319-273-2361 for more information.

Algorithms Schedule Spring 2013

Lect #	Tuesday		Thursday	
1	1/15	Ch. 1: Textbook syntax and Motivation	1/17	Big-oh definition and usage
3	1/22	Ch 2: Divide-and-Conquer and App. B	1/24	Appendix B: Solving recurrence equations
5	1/29	Ch 2: Divide-and-Conquer: Multiplication of large integers	1/31	Ch 3: Dynamic Programming: coin-change and binomial coefficient problems
7	2/5	Dynamic Programming: binomial coefficient example; Memoization	2/7	Traveling Sales-person Problem
9	2/12	Ch 4: Greedy Algorithms: Prim's algorithm	2/14	Greedy Knapsack and Dynamic 0-1 Knapsack Algorithms
11	2/19	Review for Test 1	2/21	**Test 1**
13	2/26	Ch 5: Backtracking with Coin-change Problem	2/28	Backtracking with sum-of-subsets Problem
15	3/5	Backtracking 0-1 Knapsack Problem	3/7	Ch 6: Best-first search with Branach-and-Bound on the 0-1 Knapsack Problem and TSP
17	3/12	Ch 7: Computational complexity of sorting and (Ch 8) searching	3/14	Ch 9: Intro. To the theory of P and NP
	3/19	**Spring Break**	3/21	**Spring Break**
19	3/26	Reduction of CNF-SAT to clique problem	3/28	Overview of Cook's Theorem
21	4/2	Review for Test 2	4/4	**Test 2**
23	4/9	Def. of NP-Hard and Reduction Activity	4/11	TSP Approximation algorithm
25	4/16	Bin-Packing Approximation algorithm	4/18	Ch 10: Intro. to Parallel Architectures
27	4/23	PRAM model	4/25	Parallel Architectures vs. PRAM model
29	4/30	Intro. to Parallel Algorithms	5/2	Review for Final
Final: Wednesday, May 8 from 1:00 to 2:50 PM in ITT 322				