CSCI 445 SYLLABUS

COURSE NAME, NUMBER	CSCI 445: Analysis of Algorithms
SEMESTER, YEAR	Fall 2010
INSTRUCTOR	T. VanDrunen
OFFICE / TELEPHONE / EMAIL	Armerding 112 752-5692 Thomas.VanDrunen@wheaton.edu
OFFICE HOURS	MWF 9:15-10:20 am; Tu 2:15-3:45 pm; Th 9:00-11:30 am
COURSE WEBSITE	`http://cslab.wheaton.edu/~tvandrun/cs445`

RESOURCES Cormen et al. *Introduction to Algorithms*, third edition, McGraw Hill / MIT Press, 2001. Lewis and Papadimitriou. *Elements of the Theory of Computation*, second edition, Prentice Hall, 1998

COURSE DESCRIPTION

An introduction to algorithmic efficiency and to techniques for the design and analysis of efficient algorithms. General topics include a review of asymptotics, algorithm design techniques (such as divide-and-conquer, dynamic programming, and greedy algorithms), graph algorithms, languages and automata, and NP-completeness.

GOALS AND OBJECTIVES

1. Students will be able to analyze the time complexity of algorithms
   - Using worst- and average-case analysis.
   - Using recurrences and substitution.
   - Using the master method.
   - Using amortized analysis.
   - Using rigorous experimental methods
2. Students will be able to understand, adapt, and devise advanced algorithms
   - For classical problems such as sorting and searching.
   - For specialized sets of data.
   - Using advanced and specialized data structures.
   - Using dynamic programming and greedy strategies.
3. Students will be able to identify and categorize intractable problems
   - In articulating the nature of NP-completeness.
   - In proving problems to be NP-hard and NP-complete.
   - In finding approximate solutions to NP-complete problems.

ASSESSMENT PROCEDURES

1. Problem sets will exercise students' abilities to compose algorithms, analyze their complexity, and prove facts about them.
2. An implementation project will demonstrate the students' abilities to apply the principles learned in the material and compare theoretical and experimental methods.
3. A presentation will demonstrate the students' abilities to explore the depth of a topic and articulate the important points.
4. The midterm and final exam will evaluate students' mastery of the comprehensive material.

Grading:

	weight
Participation	5
Problem sets	40
Implementation/experimentation project	10
Presentation	15
Midterm	15
Final	15

SPECIAL EXPECTATIONS

Academic Integrity

Homework: Collaboration among students in the class is permitted on most assignments. Many problems you will be assigned are discussed or solved in the computer science literature or on the Internet. Use of these resources for graded assignments is discouraged, and if they are ever used, full citation must be given, just as in a research paper. Use of any resources that specifically serve as solutions to exercises in Cormen et al is not permitted.

Projects: Work on projects must be done independently. Students should not receive help (except for minor debugging advice) from anyone, inside or outside the course, except the instructor. Any ideas (such as algorithms, the design of data structures, or analysis) received from outside resources must be cited.

Examinations: Any take-home portion of the midterm or final must be completed independently, without any aide from any person inside or outside the class. Unless stated otherwise, assume take-home portions of the examinations are closed-book, closed-notes and that the use of other resources is not permitted.

Attendance

Students are expected to attend all class periods. It is courtesy to inform the instructor when a class must be missed.

Late Assignments

Graded homework will not be accepted late. If an assignment is not complete by the deadline, turn in what you have for partial credit.

Examinations

This class has a midterm and a final. I have not yet decided whether the midterm will be in-class, take-home, or a combination. The final exam is Thursday, Dec 16, at 10:30 AM. I do not allow students to take finals early (which is also the college's policy), so make appropriate travel arrangements.

Special needs

Whenever possible, classroom activities and testing procedures will be adjusted to respond to requests for accommodation by students with disabilities who have documented their situation with the Registrar and who have arranged to have the documentation forwarded to the course instructor. Computer Science students who need special adjustments made to computer hardware or software in order to facilitate their participation must also document their needs with the Registrar in advance before any accommodation will be attempted.

Office hours.

As always, I try to keep a balance: Stop by anytime, but prefer my scheduled office hours. This semester I am trying to reserve Tuesday mornings for uninterrupted work. If possible, please find another time to stop by with questions (but if the matter is urgent or no other time is possible, then you may still stop by then). Also, any time my door is closed, it means I'm doing something uninterruptable, such as making an important phone call. Rather than knocking, please come back in a few minutes or send me an email.

Dress and deportment.

Please dress in a way that shows you take class seriously. Do not wear sweat pants or other slumber-party wear to class. (If you need to wear althletic clothes because of activities before or after class, that's ok, but try to make yourself as profession-looking as possible.) If you must eat during class (for schedule or health reasons), please let the instructor know ahead of time; we will talk about how to minimize the distraction.

Electronic devices.

Do not use laptops during class (unless you are using it for a presentation or similar purpose). If you feel you will take better notes on your laptop, then talk to me. I will give you a stern warning against doing anything else besides note-taking on your laptop during class. Please make sure other electronic devices are silenced and put away. ***Text in class and DIE.***

Textbook. Students are expected to purchase Cormen et al. To reduce the cost to students, the computer science department has purchased copies of Lewis and Papadimitriou, which students may borrow.

Class format. This course will not follow a typical lecture format. To prepare for a class period, students will read a portion of the textbook and work on a few exercises based on the reading. Class will be spent clarifying and discussing points in the reading and working out problems and exercises.

Problem sets. The bulk of the work in this class will be pencil-and-paper problem sets. You are permitted to work together to a reasonable extent. If you work with another person or two, be your own judge that you are making a grade-worthy contribution and receiving the full benefit of the exercise.

Implementation project. Each student is required to implement a significant algorithm, data structure, and/or application thereof. This project has two purposes. First, it will help you consider the practical, engineering aspect of the algorithms. Second, it will give you the opportunity to compare experimental evaluation of the algorithm to the theoretical evaluation.

To begin your project, you should find an example or problem from the text and propose a project to me based on it. You and I will meet to work out the details and set a time table for completion. A completed project will consist in the code, a set of test cases demonstrating correctness, experimental data, and a report comparing the experimental results with theoretical and describing any other lessons learned about software development.

This project may be done anytime during the semester. However, when we meet, we will decide on an appropriate deadline for your particular project.

Presentation. Each student is required to present on one topic from the textbook. The presentation will be for an entire class period. The presentation should include

- a recap of the main points of the chapter (the other students will have at least skimmed the chapter beforehand)
- a presentation of an implementation of an algorithm from the chapter. Either the implementation should have a visual element or the presenter should present experimental results from the implementation.
- a leading the class through some of the exercises or problems from the chapter.

Presentation topics should be chosen from

- Max flow (CLRS 26)
- Multithreading (CLRS 27)
- Matrix operations (CLRS 28)
- Linear programming (CLRS 29)
- Fast Fourier transform (CLRS 30)
- String matching (CLRS 32)
- Computational geometry (CLRS 33)
- Sorting networks (CLRS2e 27)

Presentations will be given Oct 8–Oct 22. Students should choose a topic by Sept 10.

Topics. The primary textbook for this course ("CLRS") is the standard book for advanced undergraduate and introductory graduate study in the field of algorithms, and we will follow it closely. We will cover most of Parts I, II, IV, and VI. We will not cover Parts III and V (except Ch 21, which we will cover) as that material is adequately addressed in CSCI 345. Broadly, the topics of this course are divided into five modules:

I. Elementary analysis techniques (CLRS 2–4). 2–3 weeks.

> *These techniques are called "elementary" in comparison to some of the advanced techniques we use later, but some are quite sophisticated in their own right. This portion will review algorithm analysis and proof of correctness, perform a careful study of complexity classes, and examine techniques for analyzing recursive algorithms.*

II. Sorting (CLRS 7 & 8). 1 week.

> *Sorting is the classic problem in the field of algorithms. The quick sort algorithm is one of computer science's gems, so we will give it a close examination. It serves as a good case study for techniques already reviewed and learned. We will also consider theoretical limits to sorting algorithms.*

III. Advanced algorithm design and analysis techniques (CLRS 15–17). 2–3 weeks.

> *Dynamic programming, greedy algorithms, and amortized analysis.*

IV. Disjoint sets (CLRS 21). 1 week.

A special topic I've never studied before but have always wanted to.

V. Miscellaneous topics (CLRS 26–33). 2 weeks.

See description of presentations above.

VI. Graph algorithms (CLRS 22–25). 2 weeks.

Basic graph algorithms (BFS, DFS, etc) will be reviewed as necessary. We will study minimum spanning trees, single-source shortest paths and all-pairs shortest paths algorithms.

V. Theory and NP-Completeness. 4–5 weeks.

Most of this will come from our secondary textbook ("LP"). Automata and languages (LP 2 and 3) will be introduced and reviewed as necessary. Our focus will be on Turing machines (LP 4), computational complexity (LP 6), and NP-completeness (LP 7). We will also look at an algorithmic approach to understanding and dealing with NP-completeness (CLRS 34 and 35) for comparison. To ensure sufficient time before the end of the semester, this module will begin on Nov 8.

See the course website for the current plan for specific topics and their order. This schedule is subject to change.

Examinations. The midterm will be sometime near the middle of the semester. The final exam is on Thursday, Dec 16, 10:30 am. The final will be mostly non-cumulative.