CSCI 445

Analysis of Algorithms

Fall 2012 MFW 2:00-3:05 pm SCI 131

http://cs.wheaton.edu/~tvandrun/cs445

Thomas VanDrunen ☎630-752-5692 630-639-2255 ✉Thomas.VanDrunen@wheaton.edu
Office: SCI 163 Office hours: MWF 9:15-10:15; Tu 2:15-3:45; Th 9:00-11:00.

Contents

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

TEXTBOOKS.

Cormen et al. *Introduction to Algorithms*, third edition, McGraw Hill / MIT Press, 2009. (See the course website for a link to solutions to some of the problems as found on the book's official website and a link to lectures by one of the book's authors from a class similar to ours.)

Lewis and Papadimitriou. *Elements of the Theory of Computation*, second edition, Prentice Hall, 1998. (The department owns copies for students to borrow.)

OBJECTIVES. The outcomes of this course fall under four headings: skills in the *analysis* of algorithms, *literacy* in well-known algorithms and data structures, skills in *devising and implementing* algorithms, and understanding of problems and results in *computational complexity*. In detail, at the end of this course students should be able to

- analyze the time complexity of algorithms
  - using worst-case and expected-case analysis
  - using recurrences and substitution
  - using the master method
  - using amortized analysis
  - using rigorous experimental methods
- identify, articulate, and adapt algorithms and data structures
  - from classical problems such as sorting and searching
  - from a variety of topics chosen by the instructor after considering students' interests
- devise and implement advanced algorithms

  The other three goals are linked to the three units of the course, but this goal—improving students' algorithm-writing ability—will be pursued and assessed throughout the semester.
- identify and categorize intractable problems
  - in articulating and using the tools of formal languages and automata
  - in articulating the nature of computability
  - in articulating the nature of $\mathcal{NP}$-completeness

- in proving problems to be $\mathcal{NP}$-hard and $\mathcal{NP}$-complete
- in finding approximate solutions to $\mathcal{NP}$-complete problems

OUTLINE. This course is organized under the following units:

Core material. We study and practice the basic and advanced tools of understanding, composing, and analyzing algorithms.

- We review topics of algorithm analysis and correctness proof that students have seen in previous courses and make students' skills in these areas stronger and more rigorous.
- We learn analysis concepts and strategies such as asymptotics and complexity classes (big-theta, etc), recurrences and substitution, the master method, and amortized analysis.
- We learn algorithmic strategies such as divde and conquer, dynamic programming, and greedy algorithms
- We study sorting algorithms at a level of rigor beyond those in earlier courses; some ideas of computational complexity are anticipated as we consider the lower bound for complexity of comparison-based sorting.

Topical material. The course will include a variety of specific algorithmic topics, some to be studied in-class, some to be worked on by students as large out-of-class assignments. See below for details. Topics are drawn from

- Hash tables
- Red-black trees
- B-trees
- Fibonacci heaps
- Disjoint sets
- Introductory graph theory and algorithms[1]
- Minimum spanning trees
- Single-soure shortest paths
- Maximum flow
- Linear programming
- String matching
- Computational geometry

Computational complexity. This course plays a second duty: it provides advanced coverage of the theory of computation. Specifically

- A review of formal languages and automata
- Turing machines
- Undecidability
- Computational complexity; $\mathcal{P}$ vs $\mathcal{NP}$
- $\mathcal{NP}$-completeness, including reductions, and approximation algorithms.

For a schedule, see the course website.

[1] This topic almost certainly must be included because of dependencies among topics.

Course procedures

How we do this course. Of the courses I teach, this is the most textbook-dependent (besides a course for which I wrote the textbook) and the least lecture-oriented. Class time is used more for collaborative practice than for presentation of new material.

For most class periods, students will have a textbook reading and practice problems; in class we will clarify points in the readings, review solutions to practice problems, and work together on new problems. Students will have more serious problems sets (for turn-in and grading) about every week-and-a-half. Students will have two pair projects (see below) during the course of the semester. There will be three mostly-noncumulative tests (two held during class time, the last during the exam block).

Problem sets. The bulk of the work in this course 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. Some problems sets may include the implementation of an algorithm.

Students who expect to do further academic work in computer science, math, or physics are encourgaged to typeset their solutions to problem sets using LaTeX. The instructor will happily provide personal assistance and other resources to students wanting help learning LaTeX.

Topics and pair projects. Of the topics listed under "Topical material" in the course outline, each student will study seven: five will be chosen for the entire class, each given two days of class time; each student will study two others as pair projects.

In the first two weeks of the semester, each student should email the instructor his or her preferences among the twelve listed available topics. The instructor will then choose five topics to include in the topical unit. (Note that the survey of student interest is for the instructor's information only; the instructor retains sole discretion over the final choice of topics.)

After the in-class topics are chosen, students will choose two other topics to study as projects. The projects will consist in a portion of the relevant chapter in CLRS, a set of problems, and (probably) an implementation, decided by the instructor in consultation with the students. Students will work in pairs; each student should have a different partner for each project. The projects are officially due on the last day of class (Friday, Dec 14), but students are strongly encouraged to set earler due dates for themselves and to spread the work judiciously throughout the semester.

Course notebook. Please obtain a notebook to use for daily work and in-class problems. Please keep it presentable so that the instructor can inspect it for evaluating daily work and so that your classmates can see your work in it for collaboration.

Grading. There will be three mostly-noncumulative tests (scheduled for classtime Oct 5, classtime Nov 2, and the final exam block, Tuesday, Dec 18, 1:30 pm).

instrument	*weight*
Test 1	12
Test 2	12
Test 3	12
Team topic (project) 1	10
Team topic (project) 2	10
Problem sets	40
Participation/daily work	4

Note that test 3, held during the final exam block, is not worth more than the other two tests. The heavy weight given to problem sets reflects how they demand most of the work in this course. At an estimated 8 problem sets, students can expect that each assignment will be around 5% of the semester score. Note that each "team topic," then, is worth about as much as two problem sets.

Policies etc

Academic Integrity. 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 work (that is, problem sets and team projects, as opposed to daily work) 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.* (One exception: Solutions posted on the website that accompanies the textbook may be used to check answers for daily work; I will not assign any problems in the problem sets that have solutions posted there.)

Assignments. Graded homework (problem sets and team projects) will not be accepted late. If an assignment is not complete by the deadline, turn in what you have at that time for partial credit.

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

Examinations. Although the final exam block (Tuesday, Dec 18, 1:30 PM) is used for an assessment instrument similar to other tests, final-exam rules apply: 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.

Gender-inclusive language. For academic discourse, spoken and written, the faculty expects students to use gender inclusive language for human beings. This is not actually relevant for this course, but it's an official college policy that the syllabus contain some sort of notice like this.

Office hours. I try to keep a balance: Stop by anytime, but prefer my scheduled office hours. Please note that my teaching schedule this semester prevents me from holding office hours on MWF afternoons. Also, any time my door is closed, it means I'm doing something uninterruptable, such as making an important phone call. Do not bother knocking; instead, 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—more like a job than a slumber party. (If you need to wear athletic clothes because of activities before or after class, that's ok, but try to make yourself as professional-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. Please talk to me before using a laptop or other electronic device for note-taking. You will need to convince that it truly aides your comprehension. No student has convinced me yet, but if you're the first one then I will give you a stern warning against doing anything else besides note-taking. Trying out programming concepts on your own during class time (for example) is not productive because it takes you away from class discussion. You cannot multi-task as well as you think you can. Moreover, please make sure other electronic devices are silenced and put away. ***Text in class and DIE.***