CSE 450/598 Design and Analysis of Algorithms Fall 2007

Schedule line numbers: 75301 (CSE450); 86640 (CSE598)

Class time and place: MW 9:15–10:30, BYAC 190

Instructor: Goran Konjevod (`goran@asu.edu`) Office hours: MW 10:30–12:00 (BY450)

TA: Juraj Dzifcak (`juraj.dzifcak@asu.edu`) Office hours: TBA

Text: **Jon Kleinberg** and **Éva Tardos**, Algorithm Design, Addison-Wesley (ISBN 0-321-29535-8).

Web site: `http://thrackle.eas.asu.edu/cse450`

Course description. This is a “second” course in algorithms. It may be taken for undergraduate (CSE450) or graduate (CSE598) credit. The students are expected to understand the material typically covered in CSE310 (and its prerequisites such as MAT243). To give a specific example, you should know how quicksort and mergesort work (and be able to write and solve a reccurrence relation for mergesort), you should be able to use the “big-Oh” asymptotic notation, and you should have seen the algorithms of Dijkstra and Prim and have at least an intuitive understanding of how they work. For a review of most of the required background, read chapters 2 and 3 from the textbook. In the course, we will cover most of chapters 1, 4, 5, 6 and 7, and a selection of topics from later chapters.

Assignments and grading. There will be weekly homework assignments (14 of them) during the course, two midterms and a final exam. The breakdown of the grade will be: homework 20%, two midterms 25% each, final exam 30%. Almost every homework will contain an implementation question (see below for details). All tests and some homeworks will also contain extra-credit questions. In addition, a larger programming assignment will be given for extra credit.

The grades A+, A, A-, B+, B, B-, C+, C, D, have standard cutoffs 100, 95, 90, 85, 80, 75, 70, 60, 50. (For example, you would need at least an average of 85 to get a B+.) These numbers typically change and do not correspond exactly to the cutoffs used in the class—all I promise is that they will not be any higher. If you achieve above 90% on the final exam, your course grade will be one full level higher than what you would normally get. (Thus, if by averaging according to the numbers above you should get a C, but your final score is, say, 91, then you will in fact get a B in the course.) Each homework will be given on Tuesday, and will be due a week later, by the beginning of the class. **No late homeworks will be accepted**. No exceptions will be made to this rule. If you have concerns about arriving to class on time, you may hand in the homework early, either directly to the instructor, or through the Blackboard system. Graded homework will be returned a week later in class. You must solve the homework problems on your own. You may discuss the homework problems with your classmates, but you may not discuss solutions (there is a clear distinction between these two things). You must write up your own solutions independently. Exams will be closed-book, but you will be allowed to have a sheet of paper with your own notes.

450 vs 598. The 450 and 598 are two separate class sections. The schedule is one and the same, and so the two sections will share the lectures. The homeworks and tests will be mostly the same, but there will be additional questions for the graduate (598) section. The rosters will be kept separately, and the grading scales will be independent. (Note: even in absolute terms, the undergraduate students in this course have historically performed as well as the graduate students.)

Implementation questions. In order to help you learn and understand the algorithms we study in the course, most homework assignments will require you to implement an algorithm studied in class, or a variant of such an algorithm. These questions will be an integral part of homework assignments and will be graded as part of them as well. They should be submitted through the Blackboard system. Each implementation question will require a single source file, and will be graded by an automated script, so you should make sure you read the question and the program specification carefully. The programming language used for the implementation questions will be Python. Basics of Python will be covered by a handout given on the first day of class.

Honor policy and ethics. The highest standards of academic integrity are expected of all students. The failure of any student to meet these standards may result in sanctions as specified in the University Student Academic Integrity Policy (for example suspension or expulsion from the University). Violations of academic integrity include (but are not limited to) cheating, fabrication, tampering, plagiarism or facilitating such activities. It is highly unethical to bring to your instructor's attention the possible impact of your course grade on your future plans, including graduation, scholarships, jobs, etc. My job is to teach and to assess your work independently of any other consideration. I will have to withdraw you from the course if you compromise my ability to do this.

Students found to be involved in academic dishonesty will be removed from the class and a grade of E for the course will be submitted to the registrar. This is the least action taken. More serious actions may be taken if the situation indicates that such actions are appropriate, such as in the case of cheating during exams or on projects.

More references. Some other good texts and reference books on algorithms are the following. (Note: nothing on this list is required.) Links to additional material are available on the class web page.

Dasgupta, Papadimitriou and Vazirani, *Algorithms*, McGraw-Hill.

Cormen, Leiserson, Rivest and Stein, *Introduction to Algorithms*, MIT Press.

Goodrich and Tamassia, *Algorithm Design*, Wiley.

Knuth, *The Art of Computer Programming, Vols 1-3* (see also his website for material from the upcoming Vol. 4), Addison-Wesley.

Garey and Johnson, *Computers and Intractability*, Freeman.

Graham, Knuth and Patashnik, *Concrete Mathematics*, Addison-Wesley.

The first three of these are textbooks, each intended as a general introduction to algorithms. Each is slightly different in philosophy and presentation, and each has some nice features (DPV: concise, precise; CLRS: exhaustive; GT: good implementation examples in JAVA), but they are slightly more appropriate for a first course such as CSE310. The Knuth volumes provide comprehensive coverage of their topics, and are the ultimate reference set for algorithms, although they are still incomplete (Volume 4 has been in preparation since the late 70s and its material has only recently begun circulating in preprint "fascicles" available from the author's website). Garey and Johnson are a comprehensive introduction to the algorithms side of complexity theory. Finally, the book by Graham, Knuth and Patashnik is an excellent introduction to the more advanced mathematics involved in analysis of algorithms.

Python background. You may or may not already be familiar with the programming language Python. In any case, you should be able to easily pick up enough to make it possible to answer the implementation questions. Most algorithms we'll study in this course when implemented in Python will look almost exactly like the pseudocode we'll use in class. You can write Python source code using any test editor. In order to test your code, you'll want to have Python on your computer. If you use Linux or a Mac, you almost certainly already have Python; try entering `python` in a terminal window. On Windows, you may have to install Python by yourself. Look for pointers on how to do this on the website. (The official Python website is `http://www.python.org/`.) You may also want to try an IDE; there are several available.

As far as learning, the best introduction to Python is the official tutorial available in many formats at `http://www.python.org/doc/`. For this course, you should not need much beyond sections 1–5 and 7. (Note that the section on classes is out of date and only covers "old-style" classes. This should not be a problem in this course, because you will likely not need to build your own classes except perhaps for the extra-credit programming assignment given later in the semester.)

If you decide to look for more information, a great reference for Python is the book *Python Essential Reference (3rd Edition)* by Beazley.