CIS 502 Algorithms: Fall 2011

General

See the webpage `www.seas.upenn.edu/~cis502` for the most recent information about office hours etc. We will be using CourseWeb (blackboard) for grades, and handouts.

Grading

Homeworks 20%, Midterm (tentatively Oct 24th, 1:30-4:30pm) 35%, Final (December 14th, but likely to be longer than the alloted time for Registrar's schedule for finals) 45%. There will be no additional/extra-credit work to improve grades, etc.

Exam/Homework Policies

1. Midterm is open any handwritten material plus the handouts (solutions of homework) provided in class. The final exam is closed book, closed notes. You cannot use any electronic devices in any of the exams.

2. Extensions to homeworks and makeups for exams will only be given for (A) verifiable medical reasons (a doctor's note is necessary), (B) representing UPenn (conferences, meets which are sponsored/supported by Penn Faculty), or (C) out of town interviews (with a formal invitation letter **and** prior permission of instructor).

3. We will have weekly(ish) homeworks. You will be allowed to drop your worst homework score. Homework is due at the end of class on the due date, it can be submitted in class or by 3pm in Levine 502 with Cheryl Hickey. There will be a 25% reduction in grades if it is turned in late, by 5pm the next working day at Levine 502. Any submission later than this time will not count towards your grade. **Homeworks have to be typed** – using any software of your choice (figures can be drawn by hand). Keep an electronic copy of your homework.

4. You are allowed to discuss the homework problems (at a high level) in pairs but you have to write your own solution (**repeat**, that writing of your homework has to be your own work). Everyone has to write the names of the person they worked with. You cannot work with the same person for more than 1 homework. **You cannot use online resources or help from anyone not involved with the class/Penn to solve the problems.** You have to credit any reference book (not named in this handout) used for the homework (and explain how you came upon the book). Please see the academic integrity policy below.

Academic Integrity Policy

The following note will not concern most students. You are expected to write your work in your own words irrespective of any discussions/references. The dicussions (with other students in class) or references have to be cited. Homeworks of two or more persons/groups that become very similar **WILL** result in the involved persons being forwarded to office of student conduct. In addition, they may get zero in the entire 20% for the homeworks or fail the course depending on the severity (as determined by me).

Textbook and Syllabus

The textbook for the class is Algorithm Design by J. Kleinberg and E. Tardos, Addison Wesley.

1. Chapters 1 and 2: Review material. We will cover this cursorily, you are expected to know this.

2. Chapters 3–8: We will cover almost all of the material. The starred topics will be covered depending on the progress of the class.

3. Chapters 11 and 13: We will try to cover as much as we can.

4. Chapter 9, we will touch upon in connection to Chapter 8.

5. Special topics: Linear Programming, Computational Geometry (range searching, convex hulls, Voronoi diagrams) and Randomized algorithms (along with chapter 13).

Other Books: The following books may be handy if you feel you need more material beyond the class and the text. Do **not** feel compelled to get them. The course will stick to the Textbook.

1. Introduction to algorithms, 2nd ed., Cormen, Leiserson, Rivest, Stein. The first 14 chapters are a good review of what we expect you to know from undergrad classes.

2. Algorithms, by Dasgupta, Papadimitriou, and Vazirani.

3. The art of computer programming, Knuth. A timeless classic.

4. Introduction to Algorithms: A Creative Approach, Manber. I liked it as an undergrad; its an useful book to learn about the thinking process but will not have all the material.

5. Special topics: Randomized Algorithms, by Motwani and Raghavan; Probability and Computing, by Mitzenmacher and Upfal; Approximation Algorithms by Vazirani; Network Flows by Ahuja, Magnanti, and Orlin; Computational Geometry by beBerg, Van Kreveld, Overmars, and Schwarzkopf; and Linear Programming by Chvatal. We will be covering very small pieces of these books, 2-3 pages (if at all, no promises) and I will provide notes.

Interesting Articles about theory and CS: Selected Papers on Computer Science, Knuth. A bit dated, but a fantastic starting point for the algorithmic viewpoint.

How to prepare for the course

- **Give yourself time.** What you take away from the course is how to think about problems, so try to reserve time for thinking about problems (be it homeworks or exam preparations). You are strongly encouraged to try and solve the exercises at the end of the chapters. If you are thinking "how many problems should I solve" then the answer is "More".

- An algorithm without a proof is a Murakami novel. Brilliant picture as it is, unfortunately it will get minimal points. **Prove/show/argue/give an algorithm = PROVE**. If nothing is said, either ask or just to be on safe side, **Prove**.

- A correct answer (for this course) is typically **short and clean**. A significant part of any scientific effort is the ability of others to understand and reproduce them. Providing a long winded discussion of all material relevant to the posed question (e.g., history of modern science), is a waste of time.

What is a Proof?

A proof is an argument that covers **all** cases. Often the difficulty of a problem is in one particular case; not handling all cases is an incorrect answer. You can say "this case is similar to... if no other detail is necessary for that case. Be **extra** careful in using this.

Correctness and relevance is non-negotiable. A wrong proof or proof of an unconnected fact is not useful. Partial credit may be given but it is a subjective decision, you are encouraged to get the solutions right. Be extra careful with simple questions since it is impossible to decide if the answer was an omission or a typo – and there will usually be little partial credit for these problems.

Knowing **what** to prove about the proposed algorithm is a key part of the proof. Often we will be using the algorithms discussed in the class to solve newly posed problems – the proof in those cases would consist of (i) description of the mapping from the new problem to the problem you know how to solve (ii) proof that you can extract a correct solution of the new problem given a solution of the problem you know how to solve and (iii) accounting of the running time of various parts.