
CSE 3101 - Design & Analysis of Algorithms

Fall 2005

Course Outline

This document contains combined information from the CSE 3101 webpage. You may wish to visit the course webpage regularly for updated info <http://www.cs.yorku.ca/~papakons/>.

Instructor: Periklis A. Papakonstantinou, e-mail: papakons@cs.yorku.ca, office: CSEB 2022

Teaching assistant: TBA

Lectures: Tuesday and Thursday 5.30-7, Ross, S205.

Office hours: Tuesday and Thursday 4.30pm or by appointment.

Course web-page: <http://www.cs.yorku.ca/~papakons/>

Course news-group: <http://groups.google.com/group/CSE3101A> (Google groups)

1 Objectives & Prerequisites

This is an introductory course in the analysis and design of combinatorial algorithms. Emphasis is given on (i) familiarizing the students with fundamental algorithmic paradigms and (ii) rigorous analysis of combinatorial algorithms.

We (quickly) review the prerequisites needed from Discrete Mathematics (counting arguments, induction, recurrence relations and discrete probability); but the students are expected to take care (with the help of the instructor) for a more in-depth acquisition of the subjects taught in (prerequisite) courses at York University.

This is a modern introduction to combinatorial algorithms and it maintains some consistency with previous CSE 3101 courses. It is also consistent with the other session of CSE 3101 given this fall.

Primary textbook: *Algorithm Design*, by Jon Kleinberg and Eva Tardos

2 Grading

The final grade is calculated as follows:

- **Assignments (30%):** 4 or 5 assignments (each worth equally).
- **Midterm (25%):** closed book.
- **Final (45%):** with crib paper.

You have the chance to improve your final grade by adding an extra of at most 10%. There are some optional bonus (that count as a surplus to your final grade):

- On the third week of classes **September 22** students are taking an (optional) exam on general mathematical disciplines with emphasis on Discrete Mathematics (this is related to previous courses you attended during your studies). The test focuses on proofs. All of these topics are prerequisites for the course and thus students have to review them by themselves (possibly) by consulting with the instructor. This is a chance to improve your grade and to review things that will recur all the time during the course. The test is on:
 - Discrete mathematics: in general inductive proofs, Graph Theory, enumerating principles, discrete probability, Logic, sets, relations etc
 - Calculus of real functions: elementary calculus of real functions
 - Linear Algebra: elementary linear algebra

You will be provided with some sample problems (without solutions) to get prepared.

- 2-3 short quizzes each of 15 minutes long. All together will count as a total surplus of 5%.

3 How to study from the textbook

The style and the material presented in the textbook (TK) is very close to what we will be doing in class. Despite this, there are a few discrepancies that you should be aware of.

This course is a mathematically rigorous course on combinatorial algorithms. Some of the proofs presented in the textbook miss some details or follow a slightly different style. During the lectures the instructor will point out some of the details missing from the proofs. Also, a first course in algorithms involves several inductive proofs. The arguments in the textbook are also inductive per se, but they mostly use an equivalent form (the Well-Ordering principle). Note that induction and well-ordering are equivalent in the sense that one (semantically) implies the other. We find it a bit easier to replace arguments presented in the textbook with clearly stated inductive arguments. Note that this is a fairly easy task since the “hard” part of the inductive proof (namely the inductive step) is given in the book. The only thing that the students have to do is to do some practice by filling in the minor details needed to have an inductive argument. The only reason why we would like to do so is because we want to provide students (since this is a first course in algorithms) a unified way to argue rigorously without alternating between different proving styles.

The textbook is extremely well written, it contains lots of fully presented examples and it has a wide variety of exercises. Students are strongly encouraged to attempt to repeat the proofs of the arguments, perhaps to try to devise other than the mentioned properties and prove their validity. Every course has its own language and studying methods. This is a course that requires from students to algorithmically solve problems where “solve” means both to devise the algorithms and analyze them (prove them correct, analyze the running time etc). The language of this course is mathematics; the only language we have when **proving** that a “nice idea, solution etc” it really is “nice”. It worths noting that proving that an idea works is no less fun than coming up with the idea in the first place!

4 Collaboration & remarking requests

- The assignments should be done **individually** by each student. You are not only allowed but also encouraged to form study groups. Your assignment report must be prepared solely by you (avoid plagiarism).

- Every assignment comes with a cover page. Please mention every person you collaborated with or write "Collaborators: none". Fill-in every field and sign the cover page in the designated area.
- No late assignments unless there is an acceptable and documented reason. *In case of a late assignment you must inform the instructor before the deadline.*
- Remarking requests: Every remarking request should be addressed directly to the instructor of the course. You should attach an extra cover page mentioning the reason for remarking.

5 Assignments preparation

This is a suggestion of how to prepare your assignment reports (and partly how to answer your tests). Any other way of presenting your work is totally acceptable given that conforms to the nature and requirements of this course. Communicating technical mathematical material is a skill required in (or should developed during) this course. Here are some suggestions that hopefully will help you to improve the presentation of your work (and reduce the risk of losing credits - or help you to get partial marks):

- In most questions you will be asked to devise an algorithm (for a given problem) and analyze its (time) performance. In this course every algorithm should also be accompanied with a proof of correctness and proper time analysis.
- It helps your presentation if you begin every answer by stating (within 2-3 sentences) what is the problem you are solving and briefly state your results. Then you can elaborate by giving the (actual) answer.
- **Present your algorithm in English.** Alternatively (or additionally) you can give the description of your algorithm in **pseudo-code**. Please be careful when expressing your algorithm in English. This means that you shouldn't be vague or ambiguous about its structure. You are *strongly discouraged* in presenting algorithms using specific programming language primitives (e.g. avoid using ++, - or !=).
- There are specific problems where the algorithm might be obvious (or more obvious) but it appears that it gets more involved to prove it correct. Apart from convincing yourself that your algorithm works correctly; the ability to provide convincing (mathematically valid) arguments on the correctness of an algorithm has to do with the ability to grasp its very structure. A goal of this course is to make you develop this ability. Unless the question explicitly specifies that you should merely give an indication of the correctness of your algorithm you are required to carry the proof of correctness in detail.
- It's good to include (at least) one input example and show how your algorithm works on this input. Note that no example is sufficient, *not even as an indication*, to show the correctness of your algorithm.
- Stating the running time should always be accompanied with a counting argument for the time analysis. Any assumptions about the data-structures you use (as a black-box) should be stated explicitly.
- When arguing rigorously you should decide on how much detail you should give. It would be nice if you try to find a balance between detail and clarity. Moreover, the simpler (and shorter) the better your proof is. Note that messy, unnecessarily long and unnecessarily complicated answers involve the risk of losing marks.

6 Related readings

Apart from the KT (primary textbook) and the notes posted in the handouts we will also rely on two chapters from CLRS (see below): the chapter on asymptotic notation and the chapter on recurrence relations. Apart

from these **no other text or set of notes is required for this course**. Please note that using any other source apart from the recommended texts (see below) requires you to contact the instructor in advance.

You can use the references mentioned below. A short description for each can be found in the course webpage <http://www.cs.yorku.ca/~papakons/texts.html>.

- **Introduction to Algorithms (2nd edition)**, by *Cormen, Leiserson, Rivest and Stein, MIT Press*.
- **Algorithmics: The Spirit of Computing (2nd Edition)**, by *D. Harel, Pearson Education*.
- **Algorithms**, by *R. Sedgewick, Addison-Wesley, 1988*.
- **Introduction to Combinatorial Mathematics**, by *C.L. Liu, Mcgraw-Hill*.
- **Concrete Mathematics: A Foundation for Computer Science (2nd Edition)**, by *R.L. Graham, D.E. Knuth, O. Patashnik, Addison-Wesley*.
- **The Design and Analysis of Computer Algorithms**, by *A.V. Aho, J. E. Hopcroft, J.D. Ullman, Addison Wesley*.
- **The Art of Computer Programming (3 volumes)**, by *D.E. Knuth, Addison-Wesley*.
- **Data Structures and Algorithms (3 volumes)**, by *K. Melhorn, Springer-Verlag*.
- **Network Flows: Theory, Algorithms, and Applications**, by *R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Prentice Hall*.
- **Combinatorial Optimization : Algorithms and Complexity**, by *C.H. Papadimitriou and K. Steiglitz , Dover Publications*.
- **Introduction to the theory of computation**, by *M. Sipser, PWS Publishing*.
- **Computers and Intractability: A Guide to the Theory of NP-Completeness**, by *M.R. Garey and D.D. Johnson, W.H. Freeman & Co.*