Syllabus for CSCI 5454

Design and Analysis of Algorithms

Spring 2016

January 10, 2016

Lectures: Mon, Weds, Fri, at 10:00–10:50am in ECCS 1B12 (and via BBA)

Professor: Rafael Frongillo

Office:	ECCS 111A
Email:	raf@colorado.edu (but please ask clarification questions on Piazza)
Course URL:	https://www.cs.colorado.edu/~raf/teaching/5454-s16.html

Description: This graduate-level course will cover topics related to algorithm design and analysis. Topics include divide and conquer algorithms, greedy algorithms, graph algorithms, learning algorithms, algorithmic game theory, optimization, randomization, and general algorithm analysis. We will not cover any of these topics exhaustively. Rather, the focus will be on algorithmic thinking, performance guarantees and boundary cases, efficient solutions to practical problems and understanding how to analyze algorithms. Advanced topics will cover a selection of modern algorithms, many of which come from real-world applications.

Prerequisites: Undergraduate algorithms (CSCI 3104), data structures (CSCI 2270), discrete mathematics (CSCI 2824) and two semesters of calculus, or equivalents. This class assumes familiarity with asymptotic analysis (Big-O, etc.), recurrence relations and the correct implementation of basic algorithms. Students without the required background may struggle to keep up with the lectures and assignments.

Required Text: None. We will occasionally use Algorithms by Dasgupta, Papadimitriou, and Vazirani, which is available online for free (also check Moodle).

Overview:

- Problem sets (5 total) will be due every 2.5 weeks throughout the semester.
- The first 2/3 of the course will be lecture driven, with some time devoted to in-class problem solving. The remaining time will revolve around independent projects and presentations thereof (see below).
- The independent project (see below) is a major deliverable for the class. I expect students to commit considerable outside time to their completion (> 20 hours). As there are no formal exams, the project should be treated as a kind of written examination.

Tentative schedule:

- Week 1: Warm-up: Divide and conquer
- Week 2: Graphs and search
- Week 3: Dynamic programming
- Week 4: Greed and trees
- Week 5: Amortized data structures
- Week 6: Randomized algorithms
- Week 7: Online learning
- Week 8: Algorithmic game theory
- Week 9: Optimization
- Week 10: Parallelism and streaming
- Week 11: Spring break
- Week 12-16: Student lectures and project presentations

Assignment	Deadline
Problem set 1	Jan. 27 (Wednesday)
Project topic selection	Jan. 29 (Friday)
Projects assigned	Feb. 5 (Friday)
Problem set 2	Feb. 15 (Monday)
Problem set 3	Mar. 2 (Wednesday)
Problem set 4	Mar. 18 (Friday)
Problem set 5	Apr. 8 (Friday)
Project write-up	Apr. 29 (Friday)

Course work and grading:

The grading break-down will be: participation (0.1), project (0.3), problem sets (0.6). Participation is based on in-class discussions as well as contributions on Piazza. (For distance students, only the latter.)

Problem sets

- There will be 5 problem sets. The majority of the assignments will be proof-based (a rigorous deductive argument for a mathematical statement), though from time to time we will have small programming assignments.
- Your complete solution file must be submitted to Moodle as a single PDF file (apart from code; see below) by 11:50pm on the due date. *Late or improperly formatted solutions will receive no credit.* (I strongly recommend using LaTeX to produce your solutions.)
- Any reasonable imperative language (`C/C++`, `Java`, `Python`, etc.) may be used to complete the programming problems.

  Runable source code must be submitted to Moodle as a separate file. *Failure to submit your source code will result in no credit for the programming questions.*

  Unless specifically allowed, all parts of all algorithms and data structures must be implemented from scratch (that is, no libraries; if you use `Python` or another modern language, be sure you are not accidentally invoking non-trivial libraries; garbage collection features and static arrays are okay; "dictionary" data structures are not).
- Solutions to mathematical problems should assume a `RAM` computation model (unless otherwise specified).
- Your solutions must be detailed, clear, and *succinct*. Explain in words how you set up your analysis and explain in detail why your solution is correct, but only give details relevant to the solution. (Advice for doing this can be found at the end of this document.)
- Figures and graphs must be labeled correctly. *Figures with unlabeled axes or data series will receive no credit.*
- Collaboration is allowed on the problem sets, but you may not copy *in any way* from your collaborators and you must respect CU academic policies at all times. You may discuss the problems verbally, but you must **write up your solutions separately**.

If you discuss a problem with another student, **you must list and describe the extent of your collaboration** prominently at the top of your submission. Copying from any source in any way, including the Web but especially from another student (past or present), is strictly forbidden. If you are unsure about whether something is permitted, please ask before the assignment is due.

There will be a zero-tolerance policy to violations of this requirement. Violators will be removed from the class and given a failing grade.

- Some topics will only be covered through the problem sets.

Reading and video assignments: Most lectures will have an accompanying reading and/or video assignment. I expect you to read/view these outside of class and come prepared to discuss the material.

Independent project

For the course project, students may choose either an *implementation project* or a *research project*. The research project entails more work, and is intended for students wishing to get involved in "entry-level" research (though more advanced projects will be considered).

- Implementation project

  This type of project is relatively straightforward; you will be expected to:

  1. implement *from scratch* a non-trivial algorithm or data structure,
  2. give a detailed mathematical analysis of its correctness, space and time usage,
  3. explain the inputs resulting in worst-, average- and best-case performance,
  4. numerically characterize its worst- and average-case space and time usage,
  5. write up these results in a 10-page report, with figures and citations.

  In numerically characterizing the space and time performance, you must implement an appropriate randomized input generator, describe it in your writeup, and use it to demonstrate that your implementation achieves the claimed asymptotic bounds on both space and time across input sizes that vary over several orders of magnitude.

  Your writeup should explain clearly the type of problems the algorithm solves, the idea behind the algorithm, your analytic results, and it should both describe and comment

on the results of the numerical tests. You should close with a brief discussion of extensions, improvements and recent work in its general area.

Some students choosing implementation projects may be solicited for an optional in-class presentation to describe their topic and results.

- Research project

  For this type of project, a student will investigate a topic which is algorithmic in nature, but for which there is either an open question or unexplored facet, and work to fill in this gap through novel mathematical or experimental analysis (preferrably both). This could involve a well-defined open question, or alternatively, reading, critically evaluating, and trying to extend or modify some research papers (of algorithmic nature) beyond the scope of the class.

  The first step in the research project will be a short (1 page) "proposal", due at the topic selection time, suggesting the open question, or papers to read, and an outline of a plan for the project. *Not all proposals will be approved*; if not approved, the student will choose an implementation project topic instead. The final deliverable is a write-up of at least 10 pages which should contain a summary of the problem and model considered in the papers, an original critical evaluation of the problem, a summary of the previous results, your attempts to extend/modify the solution (or solve an open problem), an intuitive explanation and proof of why some of the results are true, and why it does or does not extend.

  In addition to the write-up, research projects will involve a brief 10-15 minute in-class presentation during the last 3 weeks of class.

A PDF of the writeup is due by 11:50pm on Friday, April 29th (last day of class). No late submissions will be accepted. Students will suggest project topics by January 29th from the list below. For both types of projects, the grade will be determined by the quality of the analysis and overall written presentation. Original code must be submitted for implementation projects and relevant research projects. For implementation projects, the presence of any code that is not original to the submitting student may result in a failing grade; for research projects, any reused code must be clearly attributed.

If you would like feedback about your project, please come to office hours.

Advice for writing up your solutions:

Your solutions for the problem sets should have the following properties. I will be looking for these when I grade them:

1. **Clarity**: All of your work and answers should be clear and well separated from other problems. If I can't quickly identify and understand your solution, I can't evaluate it. I will not spend much time looking at any particular solution, so the more clear you make your work, the more likely you are to get maximum credit.

2. **Completeness**: Full credit for all problems is based on both sufficient intermediate work (the lack of which often produces a "justify" comment) and the final answer. There are many ways of doing most problems, and I need to understand exactly how *you* chose to solve each problem. Here is a good rule of thumb for deciding how much detail is sufficient: if you were to present your solution to the class and everyone understood the steps, then you can assume it is sufficient.

3. **Succinctness**: The work and solutions that you submit should be long enough to convey exactly why the answer you get is correct, yet short enough to be easily digestible by someone with a basic knowledge of the material. If you find yourself doing more than half a page of dense algebra, generating more than a dozen numeric values or using more than a page or two per problem, you're probably not being succinct. Clearly indicate your final answer (circle, box, underline, etc.). Note: it's usually best to rewrite your solution to a problem before you hand it in. If you do this, you'll find you can usually make the solution much more succinct.

4. **Numerical experiments**: Some programming problems will require you to conduct numerical experiments. For instance, to show that an algorithm takes $O(n \log n)$ time, you will need to measure the number of atomic operation at multiple values of n, plot the measured values versus n, and then plot the asymptotic function showing that the function matches the data. Plotting the *average* number of operations for a given value of n will almost always improve your results. To get a good trend, I recommend using a dozen or so exponentially spaced values of n, e.g., $n = \{2^4, 2^5, \ldots, 2^{16}, \ldots\}$. When presenting your results, you must explain your experimental design.

5. **Source code**: Your source code for all programming problems must be included with your solution as a (single) separate file. It should be appropriately commented so that I can understand what you are doing and why, and it must be run-able – that is, if I try to compile and run it, it should work as advertised.

Suggestions: Suggestions for improvement of the course are welcome at any time. Any concern about the course should be brought first to my attention. Further recourse is available through the office of the Department Chair or the Graduate Program Advisor, both accessible on the 7th floor of the Engineering Center Office Tower.

Honor Code: As members of the CU academic community, we are all bound by the CU Honor Code. I take the Honor Code very seriously, and I expect that you will, too. Any significant violation will result in a failing grade for the course and will be reported. Here is the University's statement about the matter:

All students of the University of Colorado at Boulder are responsible for knowing and adhering to the academic integrity policy of this institution. Violations of this policy may include: cheating, plagiarism, aid of academic dishonesty, fabrication, lying, bribery, and threatening behavior. All incidents of academic misconduct shall be reported to the Honor Code Council (`honor@colorado.edu`; 303-735-2273). Students who are found to be in violation of the academic integrity policy will be subject to both academic sanctions from the faculty member and non-academic sanctions (including but not limited to university probation, suspension, or expulsion). Other information on the Honor Code can be found at `http://www.colorado.edu/policies/honor.html` and at
`http://www.colorado.edu/academics/honorcode/`

Special Accommodations: If you qualify for accommodations because of a disability, please submit to your professor a letter from Disability Services in a timely manner (for exam accommodations provide your letter at least one week prior to the exam) so that your needs can be addressed. Disability Services determines accommodations based on documented disabilities. Contact Disability Services at 303-492-8671 or by e-mail at `dsinfo@colorado.edu`.

If you have a temporary medical condition or injury, see Temporary Injuries under Quick Links at Disability Services website and discuss your needs with your professor.

Campus policy regarding religious observances requires that faculty make every effort to deal reasonably and fairly with all students who, because of religious obligations, have conflicts with scheduled exams, assignments or required attendance. In this class, I will make reasonable efforts to accommodate such needs if you notify me of their specific nature by the end of the 3rd week of class. See full details at
`http://www.colorado.edu/policies/fac_relig.html`

Classroom Behavior: Students and faculty each have responsibility for maintaining an appropriate learning environment. Those who fail to adhere to such behavioral standards may be subject to discipline. Professional courtesy and sensitivity are especially important with respect to individuals and topics dealing with differences of race, color, culture, religion, creed, politics, veterans status, sexual orientation, gender, gender identity and gender expression, age, disability, and nationalities. Class rosters are provided to the instructor with the student's legal name. I will gladly honor your request to address you by an alternate name or gender pronoun. Please advise me of this preference early in the semester so that I may make appropriate changes to my records. See policies at
`http://www.colorado.edu/policies/classbehavior.html` and at
`http://www.colorado.edu/studentaffairs/judicialaffairs/code.html#student_code`

Discrimination and Harrassment: The University of Colorado at Boulder Discrimination and Harassment Policy and Procedures, the University of Colorado Sexual Harassment Policy and Procedures, and the University of Colorado Conflict of Interest in Cases of Amorous Relationships policy apply to all students, staff, and faculty. Any student, staff, or faculty member who believes s/he has been the subject of sexual harassment or discrimination or harassment based upon race, color, national origin, sex, age, disability, creed, religion, sexual orientation, or veteran status should contact the Office of Discrimination and Harassment (ODH) at 303-492-2127 or the Office of Student Conduct (OSC) at 303-492-5550. Information about the ODH, the above referenced policies, and the campus resources available to assist individuals regarding discrimination or harassment can be obtained at
`http://www.colorado.edu/odh`