Syllabus for CSCI 3104
Algorithms
Spring 2014

Lectures: Monday, Wednesday and Friday from 2:00–2:50pm in ECCR 265

Lecturer: Aaron Clauset

Office:	ECOT 743
Email:	aaron.clauset@colorado.edu
Course Web Page:	`http://santafe.edu/~aaronc/courses/3104/`
Office Hours:	Monday and Wednesday, 1:00–1:50pm, Tuesday 2:30–3:20, or by appointment

Description: This course will cover topics related to algorithm design and analysis. Topics will include asymptotic analysis, time and space constraints, dynamic programming, divide and conquer, greedy algorithms, graph algorithms, computability, and a selection of advanced topics. The focus throughout 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: Data structures (CSCI 2270), Discrete mathematics (CSCI 2824) and two semesters of Calculus. We will assume a thorough knowledge of basic algebra, logarithms, functions, inequalities, etc., all of which you saw in calculus. You should know basic combinatorics, set theory, math induction, and graph theory. You will also be expected to have facility with a programming language such as C, C++, Python or Java.

Required Text: *Introduction to Algorithms* (3rd ed.), by Cormen et al.

Overview:

- Lectures 3 times a week (MWF)
- Weekly recitation sections (MW), following the lecture
- Problem sets throughout the semester (10 total, roughly one per week)
- Several problems sets will involve programming problems
- One mid-term exam and one final exam
- This will be a challenging course. Plan accordingly

Schedule

Week 1	Fundamentals of algorithms
Week 2	Divide and conquer
Weeks 3–4	Greedy algorithms
Weeks 5–6	Dynamic programming
Week 7	Midterm exam
Week 8	Randomized data structures
Weeks 9–10	Graph algorithms
Week 11	Spring break
Week 12–13	Graph algorithms
Week 14	Stable matchings
Week 15	Problems in P and NP
Week 16	Internet algorithms
Week 17	Final exam

Assignments & Deadlines

assignment	assigned	deadline
Problem set 1	January 13 (Monday)	January 16 (Thursday)
Problem set 2	January 17 (Friday)	January 23 (Thursday)
Problem set 3	January 24 (Friday)	January 30 (Thursday)
Problem set 4	January 31 (Friday)	February 6 (Thursday)
Problem set 5*	February 7 (Friday)	February 20 (Thursday)
Midterm exam	February 26 (Wednesday)	
Problem set 6	February 28 (Friday)	March 6 (Thursday)
Problem set 7*	March 11 (Tuesday)	March 20 (Thursday)
Problem set 8*	March 31 (Monday)	April 10 (Thursday)
Problem set 9	April 11 (Friday)	April 17 (Thursday)
Problem set 10*	April 18 (Friday)	May 1 (Thursday)
Final exam	May 5 (Monday) 1:30–4:00pm	

Of the 10 problem sets, 6 will be due 1 week after they are assigned; the 4 with a * following their numbers above have longer deadlines.

Grading

Grades will be assigned as the weighted sum of scores in three areas: problem sets (0.65), midterm exam (0.15) and final exam (0.20).

Letter grades will not be assigned until after all work for the semester has been submitted and graded. In the meantime, only numerical grades will be tracked.

Problem sets

The course staff will have to evaluate more than 5000 pages of homework submissions this semester. To make this run smoothly, we honestly need your help to make sure the grading can be done as efficiently and accurately as possible.

- There will be 10 problem sets.
- Problem sets will always be due on Thursday, by 11:59pm, via email.
- Solutions to mathematical problems should assume a `RAM` computation model (unless otherwise specified).
- Some topics will only be covered through the problem sets.
- Any reasonable imperative language (`C/C++`, `Java`, `Python`, etc.) may be used to complete the programming problems.

  Run-able source code must be included at the *end* of your solutions 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).
  If you are unsure about what counts as "non-trivial," ask for clarification.

Getting help

- **Attend the lectures, recitation sections, and office hours first.** These are provided specifically to help you learn the material. Take advantage of them.

- **Do not abuse email**. Do not ask for help with specific parts of the problem sets via email. If every student in class sent us 1 email a day, and it took only 5 minutes to respond to each, the course staff would be spending over 9 hours a day just responding to email. Email should be reserved for high-priority issues only.

- If you absolutely must communicate with us via email, you may send your queries to `csci3104cub.Q@gmail.com`. Note that any work submitted to this address will be discarded (send that to `csci3104cub@gmail.com`; note the difference).

Working in teams

- Each student must submit individual solutions to Problem Set 1.

- Starting with Problem Set 2, team solutions may be submitted for groups of up to 3 students. We *strongly encourage*, but do not require, students to work in teams. Students are responsible for forming their own teams, and teams may be different for each problem set. Students are also responsible for ensuring that every member of their team contributes to the team solutions.

- For team solutions, each member of the team must themselves submit the solution file by the deadline.

- If you discuss some problem with another student (outside your team), you must list your collaboration on the page where you give the solution. If you discussed it with 20 other people, then all 20 names should appear in your solution. If someone in particular was particular helpful, say so. Be generous; if you're not sure whether someone should be included in your list of collaborators, include them. For discussions in class, in section, or in office hours, where collecting names is impractical, it's okay to write something like "discussions in class."

- **Copying from any source in any way is strictly forbidden**. This includes both the Web and other students (past or present). If you are unsure about whether something is permitted, please see me before the assignment is due.

  There will be a zero-tolerance policy to violations of this policy. Violators will be removed from the class, given a failing grade, and reported to the University Honor Council.

- **Write everything in your own words and cite all outside resources**. You are strongly encouraged to use outside resources, but you must write your solutions yourself. We are not interested in seeing Wikipedia's or anyone else's solution. The only sources you are not required to cite are the textbook and lecture notes, and the prerequisite material (which we assume you know by heart).

Formatting and submitting your work

- **All** submissions must be made electronically. Your complete solution file must be submitted as a **single** PDF via email to `csci3104cub@gmail.com` by 11:59pm the day they are due. Hard-copy submissions will not be accepted. Do not send questions or any other types of emails to this address — this address is only for submitting your solutions.

- **Late or non-PDF submissions will receive no credit.** There is no flexibility on this policy. However, to compensate for its unforgiving nature, the lowest 3 problem set grades will be automatically dropped at the end of the semester.

- **Submitted files should be named** as `Lastname-Firstname-MMDD-PSX.pdf`, where `X` is the number of the corresponding problem set and MMDD is the 2-digit month and 2-digit day of your birthday. (Birthday information is used to distinguish students with similar names.) For instance, `Clauset-Aaron-0701-PS1.pdf`.

- The solution to each numbered problem should start on a fresh page.

- Pages should be numbered in chronological order.

- The first page of your solution file should include the following information:

  - CSCI 3104 Spring 2014
  - the problem set number
  - your name
  - MM/DD of your birthday

  For team solutions, print the name and birthday of *every* member of the team on this page.

- Nicely typeset solutions will receive **a small amount of extra credit**, at the discretion of the course staff. If you are interested in this option, try using LaTeX to typeset your solutions (see course webpage for links to LaTeX installations) or a nicely formatted Mathematica notebook. There is a LaTeX template you can use on the course website.

- If you choose not to use typesetting software, an acceptable alternative is to write your solutions out on paper and then scan the document to PDF. If you do this, be sure no material is cutoff (sides, top, bottom).

- For problem sets with programming portions, you must include your source code *at the end* of the file. Failure to do so will result in no credit for the programming question(s).

- **Solutions must be detailed and clear.** The clearer your explanation, the more likely you are to receive full credit for a correct answer. Detailed advice for achieving this is included at the end of this document.

- **Figures and graphs must be labeled correctly.** Figures with unlabeled axes or data series will receive no credit.

- In the case of exceptional circumstances, e.g., documented illness or injury, assignments may be forgiven. Examples of circumstances that will not be considered exceptional include registering late, travel for job interviews or conferences or fun, forgetting the homework deadline, or simply not finishing on time. Final course grades will be computed as if forgiven assignments did not exist, which will increase the weight on other assignments.

What to write

- **Answer the right question**.

- **Justify your answer**. Unless the problem specifically says otherwise, every homework problem requires an explanation. Without one, even a perfectly correct solution is worth nothing. In particular, the sentence "It's obvious!" is not an explanation— "obvious" is often a synonym for "false"!

- **Answer the question completely**. When a problem says to give an an algorithm, you must do several things to receive full credit.

    - If necessary, formally restate the problem to make it clear exactly what problem you are solving.
    - Give a concise pseudocode description of your algorithm.
    - Explain what your pseudocode does.
    - Justify the correctness of your algorithm.
    - Describe the correct algorithm.

– Analyze your algorithm's running time. This may be as simple as "There are two nested loops from 1 to n, so the running time is $O(n^2)$." Or, it may require setting up and solving a summation or a recurrence, in which case you will need to prove your answer is correct.

– Describe the fastest algorithm you can think of, even if the problem does not include the words "fast" or "efficient." Fast algorithms are worth more points, while brute force is usually worth very little. Not every problem will tell you what to aim for (figuring that out is part of what you're learning). But, if your algorithm is incorrect, you won't receive any points, no matter how fast it is.

Some problems may deviate from these default requirements. For example, you may be asked to give an algorithm that uses a particular approach, even though another approach is more efficient. (Answer the right question!) Some problems may ask you to analyze the space used by your algorithm in addition to the running time.

Advice for writing up your solutions

The most important thing you can do is to make it easy for the grading staff to figure out what you mean within the short time they have to grade your solution. Solutions that are difficult to read or understand will cause you to lose points (and the graders will be less sympathetic to your other mistakes).

- **Write carefully**. You will only be graded on what you write, not what you mean. We cannot read your mind. If your solution is ambiguous, we will choose the interpretation that makes the solution wrong.

- **Write clearly**. If we cannot decipher your answer quickly, we cannot give you credit for it, even if it is correct. Write your answers clearly. Separate them from other problems. Box your results. Solve problems in the order they are listed. (If you have poor handwriting, try typesetting your solutions with LaTeX, and don't submit your first draft.)

- **Explain your solution**. Unless the problem specifies it, giving pseudocode alone, with no accompanying verbal explanation, is insufficient. A complete solution is a verbal explanation of the algorithm's generic behavior with supporting pseudocode (or something functionally equivalent). Pseudocode provides precision where English does not. English provides context and interpretability where pseudocode does not.

7

- **Give generic solutions, not examples**. Solutions that include phrases like "and so on", "etc.", "do this for all X", or ". . ." will receive no credit. Those phrases indicate precisely where you should have used iteration, recursion, or induction but did not. If your solution does not work on *every* valid input, then it is not a correct solution. A complete solution explains why there is no such input on which your algorithm fails.

- **Make it understandable**. Start by describing the key ideas or insights behind your solution before going into its details. The reader should understand your approach almost immediately without looking at the proofs or pseudocode. A clearly written solution that includes all the main ideas but omits some low-level details is worth more than a complete, correct, detailed, but opaque solution.

- **Be succinct**. Your solution should be long enough to convey exactly why your answer is correct, yet short enough to be easily understood.

- **Don't bullshit**. If you don't know the answer, just say so. Answering "I don't know" (and nothing else) is worth 25% (rounded down) partial credit. The "I don't know" rule applies to every problem and sub-problem (except extra credit problems) on every problem set and exam. A blank page is worth zero points. Readable, correct, but suboptimal solutions are always worth more than 25%.

- **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.

- **Source code**: Your source code for all programming problems must be included at the end of your solutions. It should be appropriately commented so that we can understand what you are doing and why, and it must be run-able – that is, if we try to compile and run it, it should work as advertised.

Suggestions: Suggestions for improvement 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 Harassment: 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`