Overview

The goal of this class is to help you learn how to think about algorithms. We’ll cover the ‘greatest hits’ of computer science: a selection of the most important fundamental concepts in the design and analysis of algorithms and data structures, including correctness, complexity analysis, and NP-completeness. In addition to implementing some of the classics, you will design some algorithms yourself. You should already be a fluent programmer in a C-like language, be able to generate standalone command-line programs under Linux, understand some basic data structures (linked lists, binary trees, and hash tables), and have some basic experience writing proofs.

Contact Info

Wheeler Ruml  Bence Cserna
email: ruml at cs.unh.edu  bence at cserna.net
office: Kingsbury N215D  Kingsbury W236
office hours: Tuesdays 12:30-1:30pm  ?
phone: 2-2683

Please come to my office hours! I’m always interested in talking with students. If you’d like to talk to me or Bence but can’t attend office hours, it’s easy to set up a time — just call, email, or catch us after class. If you send me email, please include CS 758 in the title and be aware that, due to the volume of email I receive, I cannot promise a prompt reply. If you have a question about course material, you will get the fastest results by posting a question on the course forum at piazza.com.

Required Texts


I think that these are both important books worth owning if you are interested in computer science. They should be available at the UNH Bookstore, the Durham Book Exchange, and on-line (I’m a fan of used.addall.com). If you buy used books, please be sure to select the correct editions. Note that exercises in CLRS appear at the end of each section (eg, exercise 7.2.1 is the first one at the end of section 2 in chapter 7) and problems appear at the end of each chapter (eg, problem 7-2).

I also recommend Kleinberg and Tardos, Algorithm Design (Addison Wesley, 2005). It is not as complete as CLRS but it has great examples. Bentley’s Programming Pearls (Second Edition, Addison Wesley, 2000) has case studies of real applications. Garey and Johnson’s Computers and Intractability (Freeman, 1979) is a readable introduction to NP-completeness.

Evaluation

Tentative weighting:
80% 14 weekly assignments. Some will have more programming, some will have more written work.
10% midterm exam
10% final exam

Each assignment will have extra challenges for those students taking the graduate version of the course.
General policies:

Clarity will be considered when grading your work—if it’s hard for us to understand, it will be hard for us to give you credit.

Assignment deadlines are not flexible and there will be no credit for late work. This allows us to discuss solutions promptly after the deadline. The schedule indicates when assignments are due so that you can plan your work in advance. When work is handed back, please look it over promptly. You may ask for something to be re-graded within 7 days of when it was handed back.

Any work you turn in must be yours. The purpose of the assignments is for you to think about the material yourself. Sometimes this will require a lot of hard thinking, even for a single problem. You may need to come back to a problem several times before inspiration strikes. Any collaboration must be cleared with me in advance and you must cite all sources you use in preparing your work, other than the printed textbook, my lectures, and the recitations. This material is very popular and there are almost certainly solutions available on the open Internet, including on the textbook website linked from the course homepage. Please don’t look at them. If you do, UNH policy provides for penalties starting with failing the course and including dismissal from the University. I reserve the right to hold unannounced quizzes or increase the weight of the written exams at any time.

If you are registered with the student disability office, please let me know right away so that I can provide proper accommodation.

Programming assignments:

All programming assignments will be in C and we will provide some scaffolding code and sample inputs to get you started. Your code must compile (with the `-ansi`, `-pedantic`, `-Wall`, and `-Werror` flags to gcc, unless you are using C99 in which case substitute `-std=c99` for `-ansi`) and run on `agate.cs.unh.edu`. You should learn how to use the `valgrind` tool to catch bugs.

Written exams:

No notes or books will be allowed. The final exam will be cumulative, but with emphasis on the second half of the course. Please be sure to attend the exams, as they are only given once.