Course Fact Sheet for CSCI 3104: Introduction to Algorithms

Time	M W F 2:00 – 2:50	ECCR 150	
	Recitations (Monday):		
	R11: 3:00-3:50 ECCR 118		
	R12: 4:00-4:50 ECCR 118		
Instructor	Hal Gabow	Office ECOT 624	Mailbox ECOT 717
	Office phone 492-6862	Email hal@cs.colorado.edu	
Office Hours	M 3:00–3:45, 5:15–6:00; W 3:00–3:45; F 4:00 – 4:45 These times will probably change – see my home page for the authoritative list. After class is always good for questions, or by appointment.		
TA	Shuxin Nie	Shuxin.Nie@colorado.edu	
	Office Hour TBA		
Prerequisites	Calculus 2 & CSCI 2270 (Data Structures) (grade at least C− in both). No exceptions can be made.		
Requirements	Programming assignments and written homework (worth 20% of the final grade). 3 in-class exams (15% each). Final Exam (Sat. May 5, 7:30am–10:30am, 35%). If you cannot take the final at this time, you must make arrangements with me in the first month of the semester. Final grades are assigned on a curve; in Spring '06 there were 18 As, 14 Bs, 12 C or below.		

Homework

Assignments are usually 1 week long and usually due Friday; sometimes they'll be $1\frac{1}{2}$ or 2 weeks. Assignments are always due *in class.* Assignments *will not be accepted after class* (so don't slip them underneath my office door). A program with no output can receive only 40 points maximum.

Extra credit: In each class you can get 1 "VEC" (verbal extra credit) for verbal participation. I'll sometimes give extra homework problems that are more challenging, and meant for students who want to explore the subject deeper. You can get 5–10 "HEC's" (homework extra credit points) for this. Final grades are based on the main assignments and are only marginally influenced by ECs.

Collaboration Policy

Talking about the programs or problem sets is fine. But the final code & the final writeup must be entirely your own. Verbal advice on debugging is fine. But not copying code. If there's any doubt ask me. You will lose points if the work you hand in is consistently like someone else's. You will get a 0 for copying even part of an assignment from any source; a second violation will result in failing the course. Be warned, I usually talk to students about suspected copying many weeks after the fact.

All students are required to know and obey the University of Colorado Student Honor Code, posted at `http://www.colorado.edu/academics/honorcode`. The class website has this link posted under Campus Rules, which also has links to policies on religious observances and student disability services. Each assignment must include a statement that the honor code was obeyed – see directions on the class website. It's not unusual for me to lower grades because of honesty issues. I once sent a student to jail for cheating– unbelievable but true!

Late homeworks: Late homeworks *will not be accepted* (see above).

Exceptions for late homeworks due to travel etc. will be granted only if arrangements are approved by me at least 2 classes in advance (unless clearly impossible). Medical excuses require a physician's note.

Ignorance of class rules is not an excuse.

Class Website

The class website is `http://www.cs.colorado.edu/~hal/CS3104/home.html`. Assignments, solution sheets, current HW point totals and other course materials will be posted there. Also clarifications or changes to homework. The website is the main form of communication aside from class. There will be a class mailing list, but it will be used only for last minute announcements and pointers to new information posted on the class website.

Email programming questions to the TA, and questions on class material to me. I check my email until 9:30 PM most nights. I try to answer all email questions. But sometimes students misuse email and ask me to do their work for them. Don't do this.

Computer Accounts

Activate your account in the CSEL by following the instructions on the class website.

C++

Although the intellectual content of the course is at a higher level than any programming language, programming assignments will be in C++ since this is the main language of previous courses. If you do not know C++ you must learn the basics on your own. This shouldn't be hard since the programming assignments will use vanilla C++. This course is less about programming than its prerequisite courses – it's more about theory.

Text *Introduction to Algorithms*, 2nd Edition
T.H. Cormen, C.E. Leiserson, R.L. Rivest & C. Stein; McGraw-Hill, NY, 2001.

References On reserve in Lester Math-Physics Library, or available from me.

Discrete Math:
Discrete Mathematics and its Applications
by K.H. Rosen, McGraw Hill, 1995, 3rd Edition.

Discrete Mathematics for Computer Science
by K. Bogart, C. Stein and R.L. Drysdale, Key College Publishing, 2006

Algorithms:
Algorithms in C++
by R. Sedgewick, Addison-Wesley, Reading, MA, 3rd edition: Parts 1-4, 1998;
Part 5: Graph Algorithms, 2002.

Data Structures and Algorithm Analysis in C++
by M.A. Weiss, Addison-Wesley, Reading, MA, 2nd edition, 1999

Algorithm Design: Foundations, Analysis, and Internet Examples
by M.T. Goodrich and R. Tamassia, John Wiley & Sons, NY, 2002

There are many texts on algorithms, written at various levels. See me for advice if you want to see other material.

Course Content

Our primary goal is to learn how to design and analyze efficient algorithms.
We achieve this by studying
(*i*) specific algorithms in combinatorial computing
(*ii*) the general techniques these algorithms illustrate
"Combinatorial computing" means algorithms that are symbolic & discrete, as opposed to numeric & continuous.

The course has an additional goal:
(*iii*) provide a background in discrete mathematics.
Discrete math deals with integer quantities and objects. It provides the foundations of theoretical computer science. We cover ideas in discrete math as we do the algorithms.

A course outline is given by the Handout List (Handout #i). Our study is organized by algorithm design technique. We study the most important techniques. Because of the emphasis on techniques we will not proceed sequentially through CLRS. Here's an overview of each unit and its purpose.

Unit 1, *Data Structures for Searching*, resumes the study of data structure ideas from CSCI 2270. We study red-black trees, probably the best overall binary search tree with guaranteed logarithmic worst-case behavior. We also study *bst* operations for queries on more than one key.

Unit 2, *Computational Geometry*, is an area that's useful for graphics and scientific computation. We cover several applications of sweep lines, the technique that solves many of the basic problems in computational geometry.

Unit 3, *Cryptography*, touches on the basic ideas of digital crytography, a cornerstone of computing in the Internet era. It also introduces randomized algorithms. The unit also fleshes out the details of our asymptotic model of computation.

Unit 4 covers the one of the most powerful general techniques for efficient algorithm design, *Dynamic Programming*. This approach is particularly suited for real-world problems with messy details. Our example from computerized typesetting is used in a number of real-world systems; the lcs algorithm that we study is the basis of `diff`.

Unit 5 covers a more specialized technique, the *Greedy Method*. When it applies it gives the simplest & fastest algorithms. We cover classic applications to data compression and merging.

Unit 6 deals with a mathematical model that pervades computer science and other disciplines, *Graphs*. We study 3 fundamental algorithms: shortest paths (used e.g. in Internet mapping services), minimum spanning trees, and depth-first search, the most powerful method for solving graph problems. We cover the basic connectivity properties of graphs and their algorithms. We combine these techniques to give an approximation algorithm for the famous Travelling Salesman Problem (which arises frequently in industrial settings).

Unit 7 is a brief introduction to *Divide-and-Conquer*, a technique on par with dynamic programming in terms of applicability and power. Our example, cache efficient algorithms, is both a simple illustration of divide-and-conquer and an issue at today's research frontier.

The *Supplemental Material* in Unit 8 serves a variety of purposes: review (e.g., loop invariants, breadth-first search), detailed proofs, more advanced material, related examples and algorithms. We cover this material at various levels of depth and as time permits.

How CSCI 3104 Relates to Other CS Courses

The data structures unit builds on background from CSCI 2270. It is a transitional unit as we become more rigorous and mathematical in the study of algorithms. The use of mathematical proofs and abstractions throughout this course is good preparation for CSCI 3434, which is more abstract. Specific algorithms we cover, and our techniques, should be helpful in CSCI 3202 (searching, dags), 3287 (searching), 4229 (geometry) and 4308. Some undergraduates have continued in the graduate algorithms courses. Others have done UROP projects on algorithm implementations – see me about opportunities.

Tips on using these notes

These notes are designed to be the starting point of class presentations. The notes are terse, as class notes should be. The notes are not a text. In class we will explain the notes and supplement them. The notes are designed to save you writing – *they do not make class attendance optional!*

In class you should write comments at appropriate places in the notes. Use backs of pages for long remarks or if you write big. If I switch to the blackboard you may want to use extra free sheets of paper.

A multi-colored pen can be useful (to separate different comments; to draw complicated diagrams; once you start you'll develop your own system). These pens are required equipment for many computer scientists!

If you want to review the comments I wrote in class, I can give you a copy of the overheads. I can also provide extra copies of these notes.

Tips on using CLRS

Both the notes & CLRS will be useful, they complement each other. The material in CLRS corresponding to a handout is given in the upper left corner of the handout's page 1, & also in Handout #i. The notes often explain things differently from CLRS. They are much more succinct, and we explain the notes in class. There can be more detail in CLRS, and more examples. I'll often show pictures from CLRS, so you may want to bring it to class. Parts of CLRS may be too mathematical and can be skipped. The same goes for certain handout sections. I'll indicate both of these as we proceed. But some of you will find this extra material interesting.

Extra credit for finding a mistake in these notes!