CSci 4041H - Sample Course Syllabus

Overall Description

4 credits, 3 large class + 1 recitation hour per week. Recitation can be used for problem solving and review, or more in-depth discussion of topics and examples given in the large class. Since many students struggle with topics like analyzing algorithms, identifying what algorithms/data structures are appropriate to use in given situations, etc., the practice given in recitation will be valuable.

Text
Introduction to Algorithms, 2nd ed., Cormen, Leiserson, Rivest, and Stein. McGraw-Hill, 2001.

Learning goals: 4041 is a required course for computer science and computer engineering majors. It is a useful course for other students who desire to learn some core material in computing. We expect computer science and computer engineering students to take 4041 in their junior year. 4041 builds upon material from 2011 (Discrete Structures) and 1902 (CS II), and is a prerequisite for many CSci elective courses. Upon successfully completing this course students should know many fundamental algorithms and data structures commonly used in computer science. Moreover, they should be able to use these algorithms and data structures both in programming, and in problem analysis. Also, they should be familiar with other related topics such as NP-completeness.

Honors Class: This is the honors version of CSci 4041. It will therefore cover the usual 4041 material such as sorting and searching, advanced data structures, dynamic programming, greedy algorithms, and graph algorithms. However, as the honors version it will also allow looking at some of those topics in more detail, and including some additional material. Examples of possible additional topics include, but are not limited to: NP-Completeness, proving hardness, approximation algorithms for dealing with hardness.

Content

WEEKS 1-2: INTRODUCTION AND REVIEW: Elements of algorithm analysis. Review of mathematical background: proof by induction, big-O notation. Review of basic data structures: stacks, queues, linked lists.
WEEKS 3-4: SORTING: Review of insertion sort. Analysis of mergesort and Quicksort. Lower bound results on sorting. Radix Sort and Bucket Sort.
WEEKS 5-6: BINARY TREES: Review of general properties, traversals, and binary search trees. Applications such as prefix codes and Huffman's algorithm. Heaps. Priority queues. Heapsort. Balanced Binary Search trees. Red-Black Trees. Optional: AVL trees. Splay trees.
WEEK 7: HASHING: Hash Tables and Hashing.
WEEK 8: DISJOINT SETS: Equivalence problems and disjoint sets.

WEEKS 9-10: ALGORITHM DESIGN: Greedy algorithms and Dynamic Programming
WEEKS 11-13: GRAPHS: Review of definitions, traversal, topological sorting, single source shortest path, minimum cost spanning trees. Maximum flow problems. Connected components. Strong components. Biconnectivity.
WEEKS 14: COMPLEXITY: Introduction to NP-Completeness.

Grading is absolute (i.e. not on a curve).
A minimum of 60% is necessary for an S or C- grade.

Grading will be as follows: 94.0% or above yields an A, 90.0% an A-, 85% = B+, 80% = B, 75% = B-, 70% = C+, 65% = C, 60% = C-, 55% = D+, 50% = D, and less than 50% yields an F.

Policies

Makeup Exam Policy: There will be no provision for make-up exams, except in extraordinary and documented circumstances.

Academic Integrity Policy: Every student is expected to turn in his or her own work for all homeworks, assignments and exams. While students can generally discuss how to solve homework/assignment problems, they have to submit their own solutions/code for the problems. The University Student Conduct Code defines scholastic dishonesty as: submission of false records of academic achievement; cheating on assignments or examinations; plagiarizing; altering, forging, or misusing a University academic record; taking, acquiring, or using test materials without faculty permission; acting alone or in cooperation with another to falsify records or to obtain dishonestly grades, honors, awards, or professional endorsement. In this course, a student responsible for scholastic dishonesty will be assigned a penalty of an "F" or "N" for the course. If you have any questions regarding the expectations for a specific assignment or exam, please ask us. See the University of Minnesota Conduct Code for details.

Special Circumstances: Students with special needs or circumstances should contact me as soon as possible to make any necessary arrangements. As outlined above, extensions are only granted for unforeseen and extraordinary circumstances. Other accommodations may be arranged in cooperation with disability services.

Class outcomes

1. For each of the data structures (e.g., graphs), or techniques (e.g., big-O analysis, greedy algorithms) discussed in class, the student should be able to:

a, define the basic terminology and use it correctly.

b. give an explanation of why it is important.

c. provide and discuss specific CSci examples of its use.

d. be able to identify its important characteristics, as well as any variants or special cases.

e. perform the basic operations associated with it.

f. use it, when applicable, to analyze and solve problems.

2. For each of the algorithms covered, the student should be able to:

a. explain the algorithm's purpose, key steps, major characteristics, and what types of problems it is applicable to; and given input, be able to trace through the workings of the algorithm.

b. be able to analyze the time and space requirements of the algorithm (including variants or special cases of the algorithm).

c. be able to analyze correctness of a purported algorithm.

d. be able to explain general features of particular types of algorithms (e.g., greedy algorithms).

3. Students should also have the following additional analysis, problem solving, and presentation skills. Given a problem, students should be able to:

a. identify which structures, algorithms, and/or techniques could be useful in analyzing or solving the problem, and why; and be able to modify or specialize structures, algorithms, or techniques to make them applicable to problems that are not amenable to straightforward use of the structure, algorithm or technique.

b. present a clear, concise, logically accurate, and rigorous solution.

c. tell whether a purported solution or analysis is accurate.

d. if appropriate, be able to design and implement a coded solution.