

General Information

[Alberto C Moreira](#), Adjunct Professor.

Wednesdays, 5:30 pm to 7:30pm, Sylvia Trottier 207.

Office Hours: Tuesdays 8:00 to 10pm, ED305.

Prerequisites: Calculus, Finite Math, Data Structures. In case of doubt about your math prerequisites, try the [Math Self-Eval Test](#).

Textbook: Cormen, Leiserson and Rivest, "Introduction to Algorithms".

Email to amoreira@ieee.org

Telephone: (603) 578-8570.

Course Outline

An Algorithm can be thought as a series of instructions that manipulate an input data set to yield the desired output. Algorithms can be expressed mathematically, or by a program written in a programming language. This language can be a real language, such as Scheme, C++ or Java, or it can be a pseudolanguage where its constructs are abstracted to the bare minimum required to express the algorithm.

As the title of Prof. Nicklaus Wirth's book states, "Programs = Data Structures + Algorithms". The formal study of Algorithms aims to develop in the student a skill to express programs formally, diminishing or altogether eliminating the need for intuitive ad-hoc techniques. A sound way to write computer programs is to first specify the data structures that the program will need, and then partition the problem into a set of algorithms that operate on those data structures.

This course intends to give an overview of fundamental algorithms, while teaching how to analyze them for important characteristics such as memory requirements and performance. The course will examine paradigms for designing algorithms: Recursion, Divide and Conquer, Greedy Algorithms, Dynamic Programming, Graph Algorithms, Mathematical Algorithms. Within each of these categories, representative algorithms will be studied and analyzed.

Some of the specific topics are: analyzing algorithm behavior for speed and memory consumption; asymptotic growth of functions; solving recurrence equations; divide and conquer; mergesort, heapsort, quicksort; dynamic programming and memoization; search techniques, backtracking, heuristics, branch-and-bound; greedy algorithms; Huffman codes; graph representation; depth first and breadth first searches; spanning trees, Kruskal and Prim's algorithms, blue and red rules; Dijkstra's, Bellman-Ford and Floyd-Warshall shortest path algorithms; solving systems of equations;

Rabin-Karp, Boyer-Moore and other String algorithms; Vertex Cover, Traveling Salesman and other approximation algorithms.

Prerequisites to this course are a background on Calculus and Discrete Mathematics, as well as working knowledge and programming skills on elementary Data Structures such as Stacks, Queues and Trees. Students are also expected to be able to program on a modern programming language, such as C, C++ or Java.

Prerequisites

This course has relatively few prerequisites, but a minimum level of mastery will be required on a continuous basis. We will basically need high school Mathematics, Freshman Calculus, elementary Discrete Mathematics, and Data Structures.

The course is math intensive, and demands the ability to follow and to elaborate proofs. A generally strong high school and college mathematics is assumed, but some areas are emphasized more than others. In high school math, you should be fluent in logarithms, exponentials, roots, basic trigonometry, polynomials, Newton's binomial, arithmetic and geometric progressions, combinatorial functions (combinations, permutations), factorials, complex numbers, determinants, and general algebraic manipulation. In calculus, you will be required to apply the concept of limit; to graph complex functions and understand their behavior; to be able to integrate moderately complex functions; and to understand sequences and series; to understand and manipulate Taylor series expansions; and to apply convergence criteria to series and sequences. Note that actual problem-solving and manipulation will be required, not just conceptual understanding and appreciation.

An introductory knowledge of linear algebra will help, but is not required as a prerequisite.

In this course you will be learning new mathematical material, such as floors and ceilings, computing sums, solving recurrence equations, manipulating special numbers such as Harmonic and Fibonacci Numbers, proving facts about asymptotic growth of functions. You will learn techniques to compute discrete sums, to operate on matrices and vectors, to operate on graphs, and to move between discrete and continuous representation of computational phenomena. This means you should feel comfortable with mathematical methods and manipulation, and that you should be able to understand an algorithm by looking at it exclusively from an abstract mathematical standpoint.

You will also be required to have a good command of elementary data structures. Trees, lists, stacks, queues, hash tables, recursion, will be used generously throughout the course as building blocks for more complex material. This also means that you must be able to write pseudocode in some block-structured language: most algorithms are expressed in block-structured pseudocode, and although this pseudocode is rarely longer than thirty or so lines, it is imperative that you can understand the code and be

able to analyze its behavior and speed by applying mathematical techniques to it.

For a sample of the mathematics required, you can try the [Math Self-Eval Test](#). This is a self-evaluation test, for your own perusal. One good way of using it is to take a couple of hours of free time, and do the exam with closed book. You should be able to handle about 75% of the questions without much problem. If your score is lower than 75%, you may have a steep learning curve in front of you while trying to follow the CS557A course. If your score is significantly lower than 75%, you should expect a lot of difficulty.

Textbook

The textbook for this course is

*Cormen Leiserson and Rivest: "[Introduction to Algorithms](#)"
MIT Press, ISBN 0-262-03141-8, McGraw-Hill, ISBN 0-07-013143-0
There's an [errata](#) available as a postscript file.*

There are other books that may be useful as complementary reading:

Brassard and Bratley, "[Fundamentals of Algorithms](#)", Prentice Hall, 1996. Strong in design techniques. If you're lucky, you'll find a French language version.

Graham, Knuth and Patashnik, "Concrete Mathematics", Addison Wesley. An excellent volume on mathematical techniques used in algorithmic analysis.

Kingston, "[Algorithms and Data Structures](#)", Addison Wesley, 1990. Strong undergrad text, valuable if you find the CLR textbook difficult to read.

Knuth, "The Art of Computer Programming", Addison Wesley. The classic reference on algorithms. A bit aged but still very relevant.

Kozen, "[The Design and Analysis of Algorithms](#)", Springer Verlag. A more advanced text with a terse but strong presentation.

Parberry, "[Problems on Algorithms](#)", Prentice-Hall, 1995. A wide collection of problems: makes for a great scratching pole!

Sedgewick, "[Algorithms in C++](#)", Addison Wesley. Good introduction at descriptive and conceptual level. Some C++ required.

Check out [Prof. Kirk Pruhs's web page](#) for lots of links to other algorithms courses throughout this country.

Resources

Here's a collection of links and additional material of interest to this course.

- [Math Self-Eval Test.](#)
- [Tutorial 1: On Program Timing.](#)
- [Tutorial 2: On Asymptotic Notation.](#)
- [Tutorial 3: On Matrix Algebra.](#)
- [Tutorial 4: On Solving Systems of Equations.](#)
- [Tutorial 5: On Backtracking, Branch and Bound, and Heuristics.](#)
- [Tutorial 6: On Fourier Transforms.](#)
- [Textbook Errata.](#)
- [Ghostscript: to read the Homework and other postscript files.](#)

Class Schedule

<u>Date</u>	<u>Subject</u>	<u>Homework</u>	<u>Textbook</u>
January 17	Review of Mathematics.		Ch. 2, 3
January 24	Analysis of Algorithms. Asymptotics.	Hw 1	Ch. 1, 2, Tut1, Tut2
February 5	Solving Recurrence Equations.	Hw 2	Ch. 4
February 7	Divide and Conquer: Mergesort, Quicksort.	Hw 3	Ch. 8, 10
February 14	Trees and Heaps.	Hw 4	Ch. 7, 13
February 21	String Algorithms.	Hw 5	Ch. 34
February 28	Winter Vacation.		
March 7	Matrices and Systems of Equations.	Hw 6	Ch. 31, Tut 3, 4
March 14	Dynamic Programming.	Hw 7	Ch. 16
March 21	Greedy Algorithms	Hw 8	Ch. 17
March 28	Graph Traversals and Topological Sort.	Hw 9	Ch. 23
April 4	Minimum Spanning Trees.	Hw 10	Ch 24
April 11	Shortest Path Algorithms.	Hw 11	Ch. 25, 26
April 18	Backtracking, Branch and Bound, Heuristics.	Hw 12	TUT 5
April 25	Approximation Algorithms.		Ch. 37
May 2	Final Exam.		

Grades and Evaluation.

The grading system is designed to evaluate to what extent you have acquired mastery of the course material. It enforces a balance between home work and in-class examination, and between individual work and teamwork.

I will be looking at four levels of mastery, roughly corresponding to grades A, B, C or F:

A: Master. You have mastery of the subject . You are able to reason about the material and draw your own conclusions. You understand the textbook and recommended material well. You can understand formal proofs and undertake simple proofs on your own. You can solve most problems from the textbooks, except for those tough "multiple star" researchy ones. You are able to write reasonably complex code that embodies your own personal approach. Not only you can tackle further study of the subject, but you are also able to independently study new material on your own. You can do introductory professional work on the field. You are able to teach the material to new students.

B. Journeyman. You know the subject. You understand the material. You can read the book, but you skip some of the tougher sections and mathematical proofs. You can solve many problems from your textbooks. You can write computer programs of moderate difficulty, using your knowledge of the field. You are able to tackle further study of the subject

C. Apprentice. You passed the course. You have managed to get the minimum grade or a few points above it. You have a basic understanding of the material and did all homework and exams to a minimum satisfactory level of competence. You can solve simple problems and answer simple questions. You can read the minimum required material in the textbook. You may not be as yet prepared to take a more advanced course on the subject.

F. Novice. You don't know enough to get a passing grade.

You will be graded on three types of assignments: Homeworks, Projects and Exams. The maximum grade for the course is 100. This is distributed in the following fashion:

12 Homeworks:	5 points each	60 points
Midterm Exam:	20 points	20 points
Final Exam:	20 points	20 points
Total:		100 points

The grade points for the four levels are as follows:

A. Master:	95 or above.
B. Journeyman:	85 or above.
C. Apprentice:	75 or above.
F. Novice:	74 or below.

The homeworks are intended to develop your problem-solving skills. Each homework problem embodies fundamental concepts that are required ingredients in the mastery of the material. All homework must be done individually. Although you are free to consult with your colleagues, it is important that you do your individual homework alone: the grading system is designed in such a way that you will have difficulty attaining Journeyman or Master level without having spent the time trying to solve the homework problems on your own.

The midterm and final exams will be done in class, with open book and open notes. The grading system is designed in such a way that you will not achieve a Master level without doing good exams.

Taking into account that some people have jobs, travel, and other problems, homework can be delivered up to one week after the due date provided a good excuse is forthcoming, and in advance. Otherwise, 20% penalty up to one week of delay, and then 20% additional per week.

Homework

This [postscript file](#) contains all homeworks. Due dates are shown in the schedule. Except when noted otherwise, homeworks are to be done individually. Answers will be posted on this web site as the semester evolves.

Homework 13: You can deliver homework 13 instead of homework 7:

[Homework 13 \(Teams of 2\): Write a program in your favorite language to solve the Traveling Salesman Problem using a Branch-and-Bound algorithm. Run it through a few different graphs to show it works. Test it with extreme values to improve your program's quality.](#)

Here's a list of all homework solutions that are available so far. Remember, you need Ghostsview to read these documents!

[Solutions to Homeworks 1 through 5.](#)

News, Tips, Stuff

This page has been created on December 10, 2000.

Visit William Stalling's cool [Computer Science Student Support Site!](#)

1/23/2001: The homework has been updated for editing bugs.

**1/27/2001: The January 31 Class has been moved to Monday February 5, 5:30 to 7:30.
Room: Ed 203.**

**People who can't attend that class, I'll be at the computer center on Saturday Feb 10
from 2pm onwards. If this doesn't fit, send me an email.**

**3/28/2001: Homework 13 is now posted in the homework section. You can now choose
between homework 7 and homework 13.**

3/28/2001: [Solutions for homeworks 1 through 5](#) are now available.