Course Objectives

Algorithms are the soul of computing. Algorithmic thinking, unlike the very young electronic machinery it brings alive, is rooted in ancient mathematics. It can be roughly described as creating "recipes" (well defined sequences of computational steps) for getting "things" (computational problems specifying an input-output relation) "successfully" (correctly) "done" (in finite steps and time). Algorithm design and analysis provide the theoretical backbone of computer science and are a must in the daily work of the successful programmer. The goal of this course is to provide a solid background in the design and analysis of the major classes of algorithms. At the end of the course students will be able to develop their own versions for a given computational task and to compare and contrast their performance.

Course Overview

This course introduces basic methods for the design and analysis of efficient algorithms emphasizing methods useful in practice. Different algorithms for a given computational task are presented and their relative merits evaluated based on performance measures. The following important computational problems will be discussed: sorting, searching, elements of dynamic programming and greedy algorithms, advanced data structures, graph algorithms (shortest path, spanning trees, tree traversals), string matching, elements of computational geometry, NP completeness.

Prerequisites

MET CS 248 Discrete Mathematics, MET CS 301 Computer Science II.

Course Format and Grading Policy

New material will be presented in lecture format. Reviews, exercises and homework solutions will take place in discussion. Participation in the discussions, although not mandatory, is recommended and may result in extra credit. Homework is due one week after handed out. Late homework will not be accepted unless permission by the instructor was given prior to the due date. No predetermined scale will be used. The final grade will be assigned based on the following weighting:

Assignments: 20%  Midterm: 30%  Final: 30%  Presentation: 20%

Text

1. Lecture Notes.
Outline

- Introduction
  - What is an algorithm?
  - Notation for programs
  - Proof techniques
  - Basics review: Sets - Functions - Limits - Simple series

- Fundamentals
  - Instances and problems - Elementary operations.
  - Efficiency
  - Average and worst-case analysis
  - Examples

- Asymptotic notation
  - Introduction
  - A notation for "the order of"
  - The omega notation
  - The theta notation
  - The conditional asymptotic notation

- Analysis of algorithms
  - Analyzing control structures
  - Using a barometer
  - Amortized analysis
  - Solving recurrences

- Data structures
  - Arrays, stacks and queues
  - Records and pointers
  - Lists, graphs, trees and associative tables
  - Heaps
  - Disjoint set structures

- Greedy algorithms
  - Making change
  - General characteristics of Greedy algorithms
  - Graphs MST - Kruskal's and Prims's algorithms
  - Graphs: shortest paths
  - Knapsack problem
  - Scheduling
• Divide - and - Conquer
  • Multiplying large integers
  • Binary search
  • Sorting by: merging and quicksort
  • Finding the median
  • Matrix multiplication
  • Exponentiation
  • Quick look at cryptography

• Dynamic programming
  • Making change
  • Principles of optimality
  • The knapsack problem
  • Shortest paths - Floyd's algorithm
  • Chained matrix multiplication

• Introduction to probabilistic algorithms - Parallel algorithms
• Introduction to computational complexity