**ArsDigita University**
**Month 5: Algorithms - Professor Shai Simonson**

**Syllabus**

General information about our text and readings:

Our text is Introduction to Algorithms by Cormen, Leiserson and Rivest. It is an encyclopedic work that serves well not only as an introduction to algorithms, but also as a reference for discrete mathematics, data structures and simple computational complexity. Chapters 2-6, and 11-12 are topics that you have seen before in month 1 (Scheme), month 2 (Discrete Math) or month 4 (Java – Software Engineering). I will not assign these as readings, but you are expected to be familiar with these topics and use the sections for reference when necessary. We will use recitations for reviewing these topics as necessary.

There are many sections in the text that we will not cover in class. In particular, I believe that the material on parallel algorithms (chapters 28-30) is best left for a graduate level course that concentrates exclusively on that topic. The material on advanced data structures like binomial heaps and Fibonacci heaps represents amortized improvements over simple binary heaps and are heavily dependent on concepts from the discrete math course. This is also true for the more advanced graph algorithms of max-flow/min-cut and matching problems. We will leave the study of these topics (Chapters 20-21, 27) for your own personal future projects. B-trees (Chapter 19) are a fundamental data structure used at the disk storage level behind relational databases, and is left as a topic in the database course. All these more advanced topics represent too deep of an excursion for a first course that needs to cover so much breadth.

Behind every algorithm are theorems and proofs, mostly proofs by induction. Behind every analysis of an algorithm is a sum, a recurrence equation, or a combinatorial argument. Throughout the course we will try to separate the discrete math details from the algorithms concentrating on the latter in class, and leaving the former for reading and reference. You may not remember all the mathematical details from the discrete math course, but as long as you remember the basics then the proofs and calculations will be accessible.

Of the many applications in which algorithms arise, we will study graph algorithms and geometric algorithms, leaving out mathematical algorithms and string matching unless time allows. You have seen a lot of the mathematical algorithms (Chapters 31, 33) in month 2 (Discrete Math), and the Fast Fourier Transform (Chapter 32), which you have not yet studied, is too advanced for a first course. The string matching algorithms (Chapter 34), although interesting and practical, are a special topic lacking the general scope of graph and geometric algorithms.

A detailed syllabus with readings follows below:

**Week 1:** Introduction and Overview.

Methods of algorithm design and analysis. The relationship of algorithms to data structures. Categorizing algorithms by methodology. Categorizing algorithms by application. Coping with intractability. Sorting and Searching. Hashing and basic data structures (recitation). Discrete Math Review (recitation).

**Reading:** CLR – Chapters 1, 7, 8, 9, 10, 13.0 – 13.3, 14, 36.0 – 36.1, 37.0, handouts.

**Lecture 1:** The big picture. Algorithm methodologies – divide and conquer, dynamic programming, greedy strategy. Algorithm analysis – worst case, average case, amortized. Algorithm applications – sorting and searching, graphs, mathematical – (number theory, algebra and linear algebra), string matching, geometrical. Coping with intractability - NP-Completeness, approximation and probabilistic algorithms. (Reading: 1, Garey and Johnson handout, 36.0 – 36.1, 37.0)

**Lecture 2:** Sorting – $O(n^2)$ time sorting algorithms – bubble sort, insertion sort. $O(n\log n)$ time sorting algorithms – mergesort, heapsort. (Reading: 7).

**Lecture 3:** Sorting – Quicksort, bucket sort, radix sort. (Reading: 8-9).

**Lecture 4:** Minimum, maximum, kth largest, median. Searching - binary search trees. (Reading: 10, 13.0 – 13.3).

**Lecture 5:** Data structures – stacks, queues, priority queues, heaps, graphs, red and black trees. (Reading: 14).

**Week 2:** Graph Algorithms. Geometric Algorithms.

Minimum spanning trees. Depth and breadth first search. Shortest path. Convex Hull. Optional advanced topics – Fibonacci heaps and Binomial heaps (recitation).

**Reading:** CLR – Chapters 18.0-18.2, 22, 23, 24, 25.0 – 25.4, 35.0 – 35.3.

**Lecture 1:** More data structures – Union-Find for disjoint sets. Applications to minimum spanning trees. (Reading: 18.0 – 18.2, 22.0 – 22.3, 24, Optional 22.4).
**Lecture 2:** Breadth and depth first search. Applications of depth first search to other graph algorithms – topological sorting, strongly connected components. (Reading: 23).

**Lecture 3:** Shortest path algorithms – all edges positive, acyclic graphs, no negative weight cycles. (Reading: 25.0 – 25.4)

**Lecture 4:** Computational geometry and $O(n^2)$ convex hull algorithm. (Reading: 35.0 – 35.3).

**Lecture 5:** An $O(n\log n)$ convex hull algorithm. (Class handouts).

**Week 3:** Algorithm methodologies – recursion, dynamic programming and greedy strategies

All pairs shortest path. Matrix-chain multiplication. Knapsack. Longest common subsequence (Recitation). Polygon triangulation. Activity scheduling. Optional advanced topics – The Shannon switching game (Bridg-It) versus Hex, and Matroid Theory (recitation).

**Reading:** CLR – Chapters 16, 17.0 – 17.3, 26.0 – 26.2.

**Lecture 1:** Divide and conquer (recursion) versus dynamic programming. Fibonacci numbers and binomial coefficients. All pairs shortest path. Matrix-chain multiplication. (Reading: 16.0 – 16.1, 26.0 – 26.2, Optional 26.3).

**Lecture 2:** More dynamic programming: Polygon triangulation. CYK parsing. Longest common subsequence (recitation). (Reading: 16.3 – 16.4)

**Lecture 3:** Pseudo-polynomial time dynamic programming algorithms for NP-Complete problems: The Knapsack problem, Bandwidth Minimization. (Reading: 17.2).

**Lecture 4:** Greedy strategy. Minimum spanning tree revisited. Activity selection. (Reading: 17.0 – 17.1).

**Lecture 5:** More greedy strategy - Huffman codes (Reading: 17.3).

**Week 4:** Coping with Intractability. NP-Completeness theory. Reductions. Approximation algorithms. String Matching Algorithms (recitation).

**Reading:** CLR – Chapters 36, 37.0 – 37.2.

**Lecture 1:** What is NP? What is NP-Complete? What are reductions? Satisfiability and variations. (Reading: 36.0-36.3)

**Lecture 2:** Reductions - Vertex cover, Simple Max Cut, Hamiltonian Circuit. (Reading: 36.4 – 36.5)

**Lecture 3:** More reductions **–** Traveling salesman problem, kernel, 3-dimensional matching, and other NP-Complete Problems. Fibonacci Nim (recitation). (Reading: Class handouts).

**Lecture 4:** Approximation algorithms – Traveling salesman problem revisited. Other problems as time allows. (Reading: 37.0 – 37.2).

**Lecture 5:** Review or other topics as time allows. DNA Computing. Mathematical algorithms. String matching algorithms (recitation).. (Optional Reading: 31.2, 33.5 – 33.9, 34).