CS1800 Discrete Structures Syllabus

August 21, 2014

1 Course Goals

This course introduces the mathematical structures and methods that form the foundation of computer science. The material will be based on theory and motivated by applications from computer science. Students will learn:

- **specific skills**, e.g., binary and modular arithmetic, set notation, and methods of counting, evaluating sums, solving recurrences, ...
- **general knowledge**, e.g., basics of probability, proof by induction, growth of functions, and analysis techniques
- **general problem solving techniques** with many applications to real problems.

2 Topics

The course material is divided into five modules. Each module starts with a motivating application then goes into techniques related to that application and the theory behind those techniques. Each module ends with one or more fairly deep applications based on the material.

2.1 Computers and Computing: Numbers, Circuits, and Logic

In chapter 1, we discuss how we can represent things on a computer using only 0s and 1s? By the end of this chapter, you should be able to convert easily between base 10 (decimal) and base 2 (binary) integers as well as their base 16 (hexadecimal) equivalents. You should be able to estimate the decimal value of binary integers.

In chapter 2, we see how transistors and switches can be used to actually represent 0s and 1s on a machine. This leads naturally into a study of gates (e.g. AND, OR, NOT gates) and using these gates to build circuits. By the end of this chapter, you should be able to write down the truth table for a simple circuit and design a circuit based on a truth table. You should understand the construction of a Ripple-Carry-Adder.

In chapter 3, we go more deeply into the mathematics (*Boolean Algebra* or *Symbolic Logic*) underlying logic gates and circuits. You will see the axioms and methods used to prove logical statements. You also will use truth tables to show logical equivalence of statements and you will be able to construct circuits from logical expressions.

Chapter 4 presents the building of a simple CPU (Central Processing Unit) based on the material of the previous three chapters.

2.2 Cryptography: Integers and Modular Arithmetic

We start chapter 5 with the *Caesar Cipher* and other simple shift ciphers. We use modular arithmetic to encipher messages and study the algebra of modular arithmetic. You will learn to encipher messages and decipher them by find additive and multiplicative inverses You will be able to do modular arithmetic perform efficient computation of high power of number $\bmod n$.

In chapter 6, we see the Division Algorithm and study primes and prime factorization. We introduce the greatest common divisor (GCD) and the Euclidean Algorithm for efficiently computing the GCD of two positive integers. You will also use the Extended Euclidean Algorithm to efficiently compute multiplicative inverses $\bmod n$ when they exist.

Finally, you will learn about public-key cryptography and you will apply all of this material to implement simple instances of the RSA (Rivest-Shamir-Adleman) public-key cryptosystem.

2.3 Combinatorics: Sets, Counting, and Probability

The opening application for this module is evaluating password criteria by counting how many passwords satisfy the conditions. Before we can count how many items there are in a set, we need some of the mathematics of sets. Chapter 7 introduces some of the formalities of sets, set-builder notation, Venn diagrams, set operations, power sets, and Cartesian products. It ends with the computer representation of sets.

Chapter 8 is all about counting. That sounds simple but, for most students, it is the hardest part of the course. You will use the sum and product rules, the inclusion-exclusion principle, the pigeon-hole principle, permutations and combinations, the Binomial theorem, and balls-in-bins, alone and together to solve complex problems. Counting methods often seem simple but it really takes stepping back and analyzing a problem to use them properly.

Chapter 9 introduces simple probability. It relies heavily on the counting methods of the previous chapter. It goes on to conditional probability and Bayes theorem that you will apply to hypothesis testing. It ends with an introduction to Markov chains that we will discuss applying to page rank in search engines.

2.4 Algorithmic Analysis: Searching and Sorting

Chapter 10 introduces a few algorithms for searching lists to see if a particular item is present and for sorting lists according to some key like an id number or last name. You will perform these algorithms by hand on small lists to see how they work.

Our goal is to compare the algorithms for potential speed. We measure this by counting the number of comparisons that are made during the run of the algorithms, especially in the worst cases. The next two chapters cover some of the mathematical tools that are necessary to do this analysis.

Chapter 11 is on sequences and series. By the end of this chapter you should be able to generate the next term of linear, geometric, and quadratic sequences give the first few entries and give a formula for a general term. You will also find the sums of finite stretches of linear and geometric sequences. You should become confident at using sigma notation. This is the math we need to analyze some of the search and sort algorithms we looked at, e.g. insertion and selection sort but we need additional tools for binary search and merge sort.

Recurrence equations are just what we need to model and analyze merge sort and other algorithms where we break the problem into smaller pieces. You will set up recurrences to model problems and solve the recurrences to complete the analysis. The methods we present of solving these equations are based on mathematical induction so we will cover that here.

This module ends with a discussion of growth of functions. This doesn't sound applied but it is very important to understanding what all our analysis of algorithms means on real computers. We will see that a good algorithm is probably more important than a faster machine. We will also see that some algorithm won't do us any good in the real world because they will take too long to run.

2.5 Networks: Graphs and Trees

Networks are ubiquitous these days, the internet, phone networks, the LAN where you work, the subway system in your city are all networks. We will use mathematical graphs to model networks and work with the data structures that are used to represent them on computers. We will look some graph theory problems that are important in computing, e.g finding the shortest or cheapest path between two nodes and finding a minimal sub network that gets to all the nodes. When you finish this module, you should be able to give the adjacency list of a graph, use traversal algorithms to visit all the nodes of a graph, and follow algorithms to find shortest paths and minimal spanning trees.