Department of Computer Science
Ashoka University

CS-1110 Discrete Mathematics - Spring 2025

COURSE OVERVIEW

Instructors : Prof. Partha Pratim Das
Teaching Fellow : Shubhajit Dey

Contents

1 Overview

Discrete mathematics is the study of mathematical structures that can be considered *discrete* (in a way analogous to discrete variables, having a bijection with the set of natural numbers) rather than *continuous* (analogous to continuous functions). A related but somewhat different interpretation of the phrase *discrete mathematics* denotes all those branches of mathematics that deal with discrete entities as opposed to continuous entities. For example, a typical undergraduate course on calculus will not be considered discrete in this sense, whereas probability theory is something that can come in both discrete and continuous flavors.

A sound understanding of *Discrete Mathematics* is critical for all areas of Computer Science. Hence, this course attempts to lay the foundation for a curriculum in CS. Many of the thematic areas are covered like logic, recursive design, combinatorics (counting methods), graph theory, number theory, information theory, etc. – are deep and vast subject areas on their own. Interested students are encouraged to follow up on any area/s they might find interesting. However, everyone interested in pursuing CS must grasp the fundamentals in all these well.

2 Objectives / Learning Outcomes

To lay the mathematical foundations for computational thinking for the core and elective courses including ***Data Structures and Algorithm***, ***Design and Analysis of Algorithms***, ***Programming Languages and Translation***, ***Theory of Computation***, ***Design Practices in CS***, ***Introduction to Machine Learning***, ***Data Science and Management***, *and* ***Information Security*** *to build on.*

This course:

1. introduces students to mathematical structures, tools, techniques, and frameworks for algorithmic and computational thinking.
2. illustrates ways for problem formulation, representation, and analysis using mathematical models.
3. elucidates how mathematical rigour and formalism ensure soundness and completeness of computation.
4. also aims to build basic programming skills to code the mathematical notions using programming languages.

The emphasis throughout the course is on teaching how various concepts and results of mathematics are used in correct, efficient, and robust computations. *On the one hand, it trains students to think and reason about computation in a programming language-agnostic way; on the other, it illustrates concrete computations using a specific programming language. In the process, it complements the course on* ***Introduction to Computer Science*** *from mathematical perspective and thinking.*

3 Pre-requisites

The course expects familiarity with class 12 level mathematics (sets, relations, functions, basic logic and truth tables, basic counting, the principle of mathematical induction and calculus).

4 Course Coverage and Modules

The course comprises **9 modules in 5 parts**. The first four parts deal with theory, while the fifth part is hands-on programming. The detailed outline of the topics of each module is presented in the next section.

Part 1: Models and Proofs

Module 1: *Propositional and Predicate Logic*
Module 2: *Proof Techniques*

Part 2: Mathematical Structures

Module 3: *Basic Structures: Sets, Functions, Relations, Sequences and Matrices*

Part 3: Computational Problem Solving

Module 4: *Problem Solving with Recursive Decomposition*
Module 5: *Graphs and Trees*

Part 4: Techniques for Analyses

Module 6: *Counting*
Module 7: *Number Theory*
Module 8: *Probability and Information Theory*

Part 5: Programming Hands-on
Module 9: *Coding Computational Concepts in C*

5 Detailed Outline of Topics in Modules

This course would introduce the notion of computation and the issues of correctness and efficiency in simple settings. We emphasize that algorithm and program design are not ad hoc and trial-and-error-based methods. The course would also introduce the notion of states, invariants, logic, and almost everything from the algorithm design and logic baskets of various CS courses with simple examples. It would also touch on the notions of abstract data types and data structures. This and the ***Data Structures and Algorithm*** *course would use the same style of treatment, with an emphasis on correctness using induction and invariants and on analysis of efficiency, on more complex examples.*

Part 1: Models and Proofs

The first part of the course deals with modeling the real world using mathematical logic and then solving various problems using rules of inference. It also introduces representations of truth statements such as theorems, lemmas, corollaries, etc., and various techniques to prove or refute them. These lay the foundation of thinking about the soundness and correctness of the computations we build.

Module 1: **Propositional and Predicate Logic**

- *Propositional Logic*: Propositions, Operators, Applications, Equivalences, and Satisfiability.
- *Predicate Logic*: Predicates, Quantification, Precedence, Bound & Free Variables, and Nested Quantifiers.
- *Arguments in Logic*: Rules of Inference (for Propositions), Resolution, and Rules of Inference with Quantification
- *Logic Programming in Prolog*: Modeling and Problem Solving with Prolog

Module 2: **Proof Techniques**:

- *What is a proof*: Types of truth statements, Fallacies
- *Proof Methods and Strategy*: Direct, Indirect – Contraposition & Contradiction, Induction and Others, Soundness & Completeness

Part 2: Mathematical Structures

The second part of the course deals with basic mathematical structures necessary to understand, model, and structure computational processes. Starting with sets, notions of functions, relations, and matrices are developed for formal reasoning in computations. The handling of infinity is elucidated with infinite sets and sequences including sets of integers, rational numbers, reals, and a Fibonacci sequence.

Module 3: **Basic Structures: Sets, Functions, Relations, Sequences and Matrices**

- *Sets*: Basic Notions, Operators & Representation of Sets, and Fuzzy & Rough Sets
- *Functions*: Basic Notions & Types, and Composition & Graphs
- *Finite and Infinite Sets*: Countability of Sets, and Theory of Infinity
- *Relations*: Equivalence Class, and Partial Order (Poset)
- *Matrices*: Basic notions & Operations of matrices
- *Sequences and Summations*: Arithmetic & Geometric Progressions, and Recurrence Relations & Summations

Part 3: Computational Problem Solving

The third part of the course addresses three major aspects of computational problem solving: problem decomposition, solution refinement, and problem representation. Recursive decomposition with refinement by solution-space exploration is introduced through a set of simple problems of graded complexity. These will lead to algorithmic strategies later in the ***Design and Analysis of Algorithms*** course. Graphs and trees are introduced as widely used models for problem representation in general and will lead to various data structures in the ***Data Structures and Algorithm*** course to follow.

Module 4: **Problem Solving with Recursive Decomposition**

- *Solution Space Exploration with Recursive Decomposition*: Demonstration with simple problems
- *Analysis of Algorithms*: Time & Space Complexity, and Counting Models
- *Growth of Functions*: Asymptotic Analysis, and Master Theorem
- *Recursion and Iteration*: Tail recursion, Fibonacci by recursion, and Fibonacci by memoization / iteration
- *Induction and Recursion*: Recursive Definitions, Structural Induction, Program Correctness

Module 5: **Graphs and Trees**

- *Graphs*: Graphs and Digraphs, Basic Notions, Representations, Graph Traversals, Special Graphs, Operations, and Properties
- *Trees*: Tree Traversal, and Spanning Tree
- *Graph Problems*: Problem solving with graphs

Part 4: Techniques for Analysis

The fourth part of the course addresses a number of useful techniques and mathematical notions that help in analyses of computations. Counting, recurrence relations, modular arithmetic, and discrete probability and information theory provide fundamental mathematical tools in computing.

Module 6: **Counting**

- *Counting Principles*: Counting Examples, Counting Rules, Tree Diagrams, Pigeonhole Principle, Permutations, Combinations, Binomial & Multinomial Coefficients, and Practice Problems
- *Recurrence Relations*: Linear Homogeneous Recurrence, Linear Non-Homogeneous Recurrence, Generating Functions of Sequences – Manipulations, Closed Forms, and Practice Problems

Module 7: **Number Theory**

- *Divisibility and Modular Arithmetic*: Division, Division Algorithm, Modular Arithmetic, and Arithmetic Modulo m

- *Integer Representations and Algorithms*: Algorithms for Integer Operations, and Modular Exponentiation
- *Primes and Greatest Common Divisors*: Primes, and Greatest Common Divisors & Least Common Multiples
- *Congruences*: Solving Congruences, and Applications of Congruences

Module 8: **Probability and Information Theory**

- *Discrete Probability*: Finite Probability, Complements & Unions, Probabilistic Reasoning, Probability Theory, Bayes' Theorem, and Expected Value & Variance
- *Information Theory*: What is Information Theory?, Information and Entropy, Joint & Conditional Entropy, Relative Entropy & Mutual Information, Chain Rules for Entropy, Relative Entropy, & Mutual Information, Source Coding and Huffman Code, and Channel Coding and Hamming Code

Part 5: Programming Hands-On

The fifth and final part of the course aims to complement the theory discussed in the first four parts by elucidating how the theory is used in coding in common programming languages such as C. This part is fully hands-on and is conducted in the laboratory. The problems listed in the following module are indicative. Some of these problems and / or similar problems will be practiced in the laboratory.

Module 9: **Coding Computational Concepts in C**

- *Simple Programming in C*: Programming with expressions, conditional and loop statements, functions (using only arrays, no other data structure or pointer) and simple I/O. For example, Min/Max, Swap, Sum, Mean and Variance, Sorting and Searching, Median, Factorial, Fibonacci, Perfect numbers, Kaprekar's constant, etc.
- *Recursive Decomposition and Refinement (Module 4)*: Computations of Max, Min and Max, Max and Second Max, Sorted Order, Fibonacci Series, etc. by recursive functions and refinements to recursive and non-recursive functions.
- *Coding for Mathematical Structures (Module 3 & 5)*: Matrix arithmetic, Binary tree traversal, Simple graph algorithms, etc.
- *Coding for Counting Problems (Module 6)*: Computing Binomial coefficients, Permutations, Partitions, etc.
- *Coding for Number Theory Problems (Module 7)*: Modular arithmetic, GCD, Prime, Bezout's identity, etc.
- *Coding for Information Theory Problems (Module 8)*: Computing Entropy, Huffman Codes, etc.

Note: *Students are expected to be familiar with the C programming language through self-study.*

6 Textbooks and Readings

No book discusses all topics. *So it is important to attend all lectures by the instructor. You are also expected to carefully study the solved problems that will be periodically posted.*

- Slides with links to online resources
- *Discrete Mathematics and its Applications*, 7^{th} Ed. by Kenneth H. Rosen (*Primary Textbook*)
- *Logic in Computer Science: Modelling and Reasoning about Systems* by Michael Huth and Mark Ryan, 2004 (*Logic*)
- *Logic for Computer Science* by Steve Reeves and Mike Clarke, 2003 (*Logic*)
- *Concrete Mathematics – A Foundation for Computer Science*, 2^{nd} Ed. by Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, 1994. (*Recurrence, Number Theory, Probability, Asymptotics*)
- *Discrete Mathematics for Computer Scientists and Mathematicians* by Joe L. Mott, Abraham Kandel, and Theodore P. Baker, 2008. (*Recurrence, Counting, Graph*)
- *Applied Combinatorics* by Fred Roberts and Barry Tesman, 2009. (*Advanced*)
- *Graph Theory with Applications to Engineering and Computer Science* by Narsingh Deo, 1979. (*Graph*)
- *Introduction to Graph Theory* by Douglas B. West, 2001. (*Graph*)
- *Elements of Information Theory*, 2^{nd} Ed. by Thomas M. Cover and Joy A. Thomas, 2006. (*Information Theory*)
- *The Book of Statistical Proofs* by Joram Soch, Wikimedia. `https://zenodo.org/records/10495684`
- *The C Programming Language*, 2^{nd} Ed. by Brian W. Kernighan and Dennis Ritchie, 2015. (*Primary Reference on C Language*)