

Discrete Structures

(adapted from the Final Report of the Computing Curricula 2001 project,
a joint undertaking of the IEEE-CS and the ACM)

Discrete structures is foundational material for computer science. By *foundational* we mean that relatively few computer scientists will be working primarily on discrete structures, but that many other areas of computer science require the ability to work with concepts from discrete structures. Discrete structures includes important material from such areas as set theory, logic, graph theory, and combinatorics. The material in discrete structures is pervasive in the areas of data structures and algorithms but appears elsewhere in computer science as well. For example, an ability to create and understand a formal proof is essential in formal specification, in verification, and in cryptography. Graph theory concepts are used in networks, operating systems, and compilers. Set theory concepts are used in software engineering and in databases. As the field of computer science matures, more and more sophisticated analysis techniques are being brought to bear on practical problems. To understand the computational techniques of the future, today's students will need a strong background in discrete structures.

Topics:

1. Functions (surjections, injections, inverses, composition)
2. Relations (reflexivity, symmetry, transitivity, equivalence relations)
3. Sets (Venn diagrams, complements, Cartesian products, power sets)
4. Pigeonhole principle
5. Cardinality and countability
6. Propositional logic
7. Logical connectives
8. Truth tables
9. Normal forms (conjunctive and disjunctive)
10. Validity
11. Predicate logic
12. Universal and existential quantification
13. Notions of implication, converse, inverse, contrapositive, negation, and contradiction
14. The structure of formal proofs
15. Direct proofs
16. Proof by counterexample
17. Proof by contraposition
18. Proof by contradiction
19. Mathematical induction
20. Recursive mathematical definitions
21. Counting arguments

Learning objectives:

1. Explain with examples the basic terminology of functions, relations, and sets.
2. Perform the operations associated with sets, functions, and relations.
3. Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.

4. Demonstrate basic counting principles, including uses of diagonalization and the pigeonhole principle.
5. Apply formal methods of symbolic propositional and predicate logic.
6. Describe how formal tools of symbolic logic are used to model algorithms and real-life situations.
7. Use formal logic proofs and logical reasoning to solve problems such as puzzles.
8. Outline the basic structure of and give examples of each fundamental proof technique
9. Relate the ideas of mathematical induction to recursion and recursively defined structures.