

COURSE DESCRIPTION

Dept., Number	CS330	Course Title	Discrete Structures
Semester hours	3	Course Coordinator	Dr. Sanjiv Kapoor, Professor

Current Catalog Description

Introduction to the use of formal mathematical structures to represent problems and computational processes. Topics covered include Boolean algebra, first-order logic, recursive structures, graphs, and abstract language models. Prerequisite: CS 116 or CS 201. (3-0-3)

Textbook

Kenneth H. Rosen, Discrete Mathematics and Its Applications, McGraw-Hill, 6th Edition, 2007

References

None

Course Outcomes

Students should be able to:

- Illustrate by examples the basic terminology of functions, relations, and sets and demonstrate knowledge of their associated operations.
- Demonstrate in practical applications the use of basic counting principles of permutations, combinations, inclusion/exclusion principle and the pigeonhole methodology.
- Calculate probabilities of events and expectations of random variables for problems arising from games of chance.
- Establish and solve recurrence relations that arise in counting problems including the problem of determining the time complexity of recursively defined algorithms.
- Model logic statements arising in algorithm correctness and real-life situations and manipulate them using the formal methods of propositional and predicate logic.
- Outline basic proofs for theorems using the techniques of - direct proofs, proof by counterexample, proof by contraposition, proof by contradiction, mathematical induction.
- Relate the ideas of mathematical induction to recursion and recursively defined structures.
- Illustrate by example basic terminology of graph theory and model problems in computer science using graphs and trees.
- Deduce properties that establish particular graphs as Trees, Planar, Eulerian, and Hamiltonian.

- Illustrate the application of trees and graphs to data structures.
- Explain the basic concepts modeling computation including formal machines, languages, finite automata, Turing machines

Relationship between Course Outcomes and Program Outcomes

The following Program Outcomes are supported by the above Course Outcomes:

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices

Prerequisites by Topic

CS 116 or CS 201 - Experience with basic programming constructs and algorithms

Major Topics Covered in the Course

1. Sets, Functions and relations - sets, set operations, functions, summations, growth of functions, equivalence relations, countable and uncountable sets, examples of algorithm analysis	4.5 hours
2. Counting Methods – permutations, combinations, discrete probability, pigeonhole principle	6 hours
3. Advanced counting – inclusion-exclusion, recurrence relations, methods of solving recurrences, examples from computer sciences	6 hours
4. Introductory Logic – propositional logic, predicate logic, proof methodologies, examples of algorithm correctness	6 hours
5. Partially Ordered sets - trees, boolean algebra, example of minimizing circuits	4.5 hours
6. Introduction to Graphs - trees , connectivity, eulerian traversals, minimum spanning tree, planarity, Euler’s formula, matching	7.5 hours
7. Formal machines and languages-an introduction - automaton, grammars and turing machines	6 hours
8. Introduction to Algebraic Topics (OPTIONAL) – rings, groups, semi-groups.	1.5 hours
Exam #1, Exam #2	3 hours
Final Exam	-
	45 hours

Assessment Plan for the Course

End of every semester Course Objective Assessments by CS department. End of semester Course Evaluations by IIT. Reviewed every Spring semester by CS Undergraduate Studies Committee for possible updates in the following Fall. Once every 4-5 years a detailed review of all materials for the course is made by the CS Undergraduate Studies Committee.

How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

See the assessment discussion under Criterion 4

For a computer science program

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	2		Software design	0	
Data structures	1		Concepts of programming languages	0	