Second Semester

DISCRETE STRUCTURE

COURSE CODE: BCA 151
YEAR/SEMESTER: I/II
WORKLOAD: 6 HOURS A WEEK

CREDIT HOURS: 3
TEACHING HOURS: 48
LECTURE: 3 Hrs.
PRACTICAL: 3 Hrs.

Course description:

This course is designed to build a strong foundation in discrete mathematics—the kind of math that powers the world of computer science. It sharpens logical thinking, introduces techniques for writing solid mathematical proofs, and strengthens problem-solving skills needed in programming, algorithm design, and system development. Students will dive into key topics like logic, sets, functions, relations, combinatorics, graphs, trees, and Boolean algebra, learning not just the theory but also how these concepts apply in real tech scenarios like databases, circuits, and code structure. Through hands-on practice with tools like Python, Jupyter Notebooks, NetworkX, and Graphviz, students will bring these abstract ideas to life by building models, writing logic-based programs, and exploring how mathematics directly supports computing.

Course objectives:

Upon completion of this course , the students will be able to:

- develop the ability to think logically and construct valid mathematical arguments.
- apply set theory concepts such as set operations, Venn diagrams, Cartesian products, and power sets to solve problems in databases, programming, and decision-making structures.
- analyze relations and functions using matrix and graph representations and understand applications in modeling data and structures.
- use combinatorial techniques to solve real-life counting and arrangement problems.
- explore graph and tree structures, implement traversal algorithms (DFS, BFS, etc.), and apply these concepts in computer science domains such as networking, compiler design, and operating systems.

Course contents

1 Set Theory **6 hrs.**

1.1. Basic Concepts: Sets, elements, roster and set-builder notation, cardinality.

1.2. Set Relationships:

1.2.1. Subsets

1.2.2. Proper subsets

1.2.3. Universal set

1.2.4. Complement

1.2.5. Disjoint sets.

1.3. Set Operations

1.3.1. Union

1.3.2. Intersection

1.3.3. Difference

1.3.4. Complement

1.3.5. Symmetric difference.

1.4. Venn Diagrams: Visual representation of set relationships and operations.

1.5. Set Identities: Proof of identities using algebraic and Venn diagram methods.

1.6. Cartesian Products: Ordered pairs, cross product of two or more sets.

1.7. Power Sets: Definition and computation of power sets.

1.8. Applications: Use of sets in databases, computer programming, and decision structures.

2. Logic and Propositional Calculus **8 hrs.**

2.1. Propositions and Logical Operators: Definition of propositions, types (simple, compound), logical connectives: AND, OR, NOT, IMPLICATION, BICONDITIONAL.

2.2. Truth Tables: Constructing truth tables for expressions involving logical operators.

2.3. Tautologies, Contradictions, and Contingencies: Identifying always true/false/logical expressions.

2.4. Logical Equivalence and Implications: Laws of logic (De Morgan's, distributive, associative, etc.), verifying equivalences.

2.5. Predicate Logic and Quantifiers: Introduction to predicates, universal and existential quantifiers.

2.6. Rules of Inference: Modus ponens, modus tollens, hypothetical syllogism, and others.

2.7. Proof Methods: Direct, indirect, contradiction, contrapositive, and proof by cases.

3. Relations and Functions **8 hrs.**

3.1. Relations: Definition, Binary Relation, Representation, Domain, Range, Universal Relation, Void Relation, Union, Intersection, and Complement Operations on Relations

3.2. Properties of Binary Relations in a Set : Reflexive, Symmetric, Transitive, Anti-symmetric Relations

3.3. Relation Matrix and Graph of a Relation; Partition and Covering of a Set, Equivalence Relation, Equivalence Classes, Compatibility Relation, Maximum Compatibility Block, Composite Relation, Converse of a Relation,

3.4. Transitive Closure of a Relation R in Set X, examples from real-world scenarios.

3.5. Representation of Relations: Using matrices and directed graphs (digraphs).

3.6. Equivalence and Partial Order Relations: Properties and examples, Simple or Linear Ordering, Totally Ordered Set (Chain), Frequently Used Partially Ordered Relations, Representation of Partially Ordered Sets, Hesse Diagrams, Least & Greatest Members, Minimal & Maximal Members, Least Upper Bound (Supremum), Greatest Lower Bound (infimum), Well-ordered Partially Ordered Sets (Posets). Lattice as Posets, complete, distributive modular and complemented lattices Boolean and pseudo Boolean lattices. (Definitions and simple examples only)

3.7. Closures and Composition of Relations: Reflexive, symmetric, transitive closures.

3.8. Functions: Definition, domain, co-domain, range, examples.

3.9. Types of Functions: Injective (one-to-one), surjective (onto), bijective (one-to-one correspondence).

3.10. Inverse and Composition of Functions: Definitions and computations.

3.11. Applications: Use in programming, data mapping, and relational databases.

4. Mathematical Reasoning and Proof Techniques **6 hrs.**

4.1. Mathematical Reasoning: Basic structure of arguments, logical flow.

4.2. Mathematical Induction: Principle of induction, proof by induction, applications in series and recursive definitions.

4.3. Strong Induction: Differences from regular induction, applications.

4.4. Recursive Definitions: Defining sequences and structures recursively.

4.5. Structural Induction: Proofs involving recursively defined structures like trees and lists.

4.6. Applications: Problem-solving and validation of algorithms.

5. Combinatorics and Counting Principles **5 hrs**

5.1. Basic Counting Principles: Introduction to counting, rule of sum and rule of product with real-world examples (e.g., menu combinations, clothing combinations).

5.2. Permutations and Combinations: Concepts of ordered and unordered selections, factorial notation, formulae for permutations (nPr) and combinations (nCr), applications in password generation and team selection.

5.3. Pigeonhole Principle: Understanding the concept, simple and strong pigeonhole principle, applications such as birthday paradox, drawer problems, and error checking.

5.4. Inclusion-Exclusion Principle: Set-based approach to solving overlapping sets, solving problems involving counting elements in unions of sets (up to three sets), and its application in probability and combinatorics.

6. Graph Theory and Trees **12 hrs.**

6.1. Graphs: Introduction, definition, examples; Nodes, edges, adjacent nodes, directed and undirected edge, Directed graph, undirected graph, examples; Initiating and terminating nodes, Loop (sling), Distinct edges, Parallel edges, Multi-graph, simple graph, weighted graphs, examples, Isolated nodes, Null graph; Isomorphic graphs, examples; Degree, Indegree, out-degree, total degree of a node, examples

6.2. Subgraphs: definition, examples; Converse (reversal or directional dual) of a digraph, examples;

6.3. Path: Definition, Paths of a given graph, length of path, examples; Simple path (edge simple), elementary path (node simple), examples; Cycle (circuit), elementary cycle, examples;

6.4. Reachability: Definition, geodesic, distance, examples; Properties of reachability, the triangle inequality; Reachable set of a given node, examples, Node base, examples;

6.5. Connectedness: Definition, weakly connected, strongly connected, unilaterally connected, examples; Strong, weak, and unilateral components of a graph, examples, Applications to represent Resource allocation status of an operating system, and detection and correction of deadlocks;

6.6. Matrix representation of graph: Definition, Adjacency matrix, boolean (or bit) matrix, examples; Determine number of paths of length n through Adjacency matrix, examples; Path (Reachability) matrix of a graph, examples; Warshall's algorithm to produce Path matrix, Flowchart

6.7. Types of Graphs: Simple, multigraph, weighted, directed/undirected, complete, bipartite.

6.8. Graph Traversal: Breadth-First Search (BFS), Depth-First Search (DFS).

6.9. Trees: Trees: Definition, branch nodes, leaf (terminal) nodes, root, examples;

6.10. Different representations of a tree, examples; Binary tree, m-ary tree, Full (or complete) binary tree, examples;

6.11. Converting any m-ary tree to a binary tree, examples;

6.12. Representation of a binary tree: Linked-list; algorithms; Applications of List structures and graphs

6.13. Tree Traversals: Inorder, preorder, postorder traversal techniques.

6.14. Applications: Networking, pathfinding algorithms, compiler syntax trees, file systems.

7. Algebraic Structures **3 hrs.**

7.1. Binary Operations: Definition and examples of binary operations on sets.

7.2. Algebraic Systems: Semigroups, monoids, and groups - axioms and properties.

7.3. Group Theory Basics: Identity element, inverse, associativity, examples with integers and matrices.

7.4. Boolean Algebra: Basic postulates and theorems, duality, Boolean functions.

7.5. Logic Circuits: Simplification of logic circuits using Boolean expressions.

7.6. Applications: Automata theory, logic design, cryptography

Practical **(48 Hours)**

Instructor should encourage the use of open-source tools like Python, Jupyter Notebooks, NetworkX, and Graphviz for practical sessions.

Practical Report Contents: Theory, Source code, Output, Conclusion

1. Truth tables, logical operators, tautologies using Python or logic tools
2. Set operations implementation and Venn diagram visualization
3. Relation properties and matrix/digraph representation in code
4. Function definitions and mapping types with coding examples
5. Recursion-based programs (factorial, Fibonacci), validating inductive proofs
6. Permutations and combinations via iteration and recursion
7. Graph representation: adjacency list and matrix (using NetworkX or code)
8. Graph traversal: DFS and BFS implementations
9. Tree structures and traversals (inorder, preorder, postorder)
10. Boolean algebra simplification and expression evaluation via code
11. Mini project: Logical puzzle solver or graph simulation

Required readings:

Grimaldi, R. P. (3rd ed.). Discrete and Combinatorial Mathematics: An Applied Introduction. Pearson

Kolman, B., Busby, R. C., & Ross, S. (2000). *Discrete mathematical structures* (3rd ed.). Prentice Hall.

Liu, C. L. (1985). *Elements of discrete mathematics* (2nd ed.). McGraw-Hill.

Rosen, K. H. (2011). *Discrete mathematics and its applications* (8th ed.). McGraw-Hill.

Stein, C., & Drysdale, R. L. (2010). Discrete Mathematics for Computer Scientists. Pearson Education.

Required Journals/ Articles:

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.

Hopcroft, J. E., & Tarjan, R. E. (1973). Algorithm for finding the strongly connected components of a directed graph. Communications of the ACM, 16(6), 372–378. https://doi.org/10.1145/362248.362272

Knuth, D. E. (1968). The art of computer programming: Volume 1: Fundamental algorithms. Addison-Wesley.

Pippenger, N. (1978). Theories of computation. Journal of Computer and System Sciences, 16(1), 99–115. https://doi.org/10.1016/0022-0000(78)90030-6