Introduces discrete mathematics concepts in relation to computer science. Applies the use of Boolean algebra, analysis of algorithms such as logic, sets and functions, recursive algorithms, and recurrence relations, combinatorics, graphs, and trees. Assignments in this course require a basic understanding of programming concepts, problem solving, basic college algebra and trigonometry skills. Lecture 3 hours per week. 3 Credits.

CSC 208 is designed to provide students with components of discrete mathematics in relation to computer science used in the analysis of algorithms, including logic, sets and functions,

R 5

- Explain with examples the basic terminology of functions, relations, and sets.
- Perform the operations associated with sets, functions, and relations.
- Compare practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.
- Use the terms cardinality, finite, countably infinite, and uncountably infinite to identify characteristics is associated with a given set.
- Demonstrate the algebra of sets, functions, sequences, and summations.

- Outline the basic structure of each proof technique, including direct proof, proof by contradiction, and induction.
- Apply each of the proof techniques (direct proof, proof by contradiction, and proof by induction) correctly in the construction of a sound argument.
- Deduce the best type of proof for a given problem.
- Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures.
- Explain the relationship between weak and strong induction and give examples of the appropriate use of each.
- Construct induction proofs involving summations, inequalities, and divisibility arguments.

- Apply counting arguments, including sum and product rules, inclusion-exclusion principle and arithmetic/geometric progressions.
- Apply the pigeonhole principle in the context of a formal proof.
- Calculate permutations and combinations of a set, and interpret the meaning in the context of the particular application.
- Compare real-world applications appropriate to counting formalisms.
- Solve a variety of basic recurrence relations.
- Analyze a problem to determine underlying recurrence relations.
- Perform computations involving algebraic and modular arithmetic.
- Determine if a recursive solution is more efficient than an iterative solution.

- Use a truth table to prove the logical equivalence of statements.
- Convert logical statements from informal language to propositional and predicate logic expressions.
- Apply formal logic proofs and/or informal, but rig2 Tc 0.007Tw (m)-an (an)10.5(g)-11.2 (ua)10.6(ge8Tw T[

- Solve a variety of real-world problems in computer science using appropriate forms of graphs and trees, such as representing a network topology or the organization of a hierarchical file system.
- Implement graph algorithms.
- Implement and use balanced trees and B-trees.
- Demonstrate how concepts from graphs and trees appear in data structures, algorithms, proof techniques (structural induction), and counting.
- Describe binary search trees and AVL trees.
- Explain complexity in the ideal and in the worst-case scenario for both implementations.

- Calculate probabilities of events and expectations of random variables for elementary problems.
- Differentiate between dependent and independent events.
- Explain the significance of binomial distribution in probabilities.
- Apply Bayes Theorem to determine conditional probabilities in a problem.
- Apply the tools of probability to solve problems.

- Explain recurrence relations in respect sequence or multidimensional array of values in computing.
- Explain what types of problems are solved using recurrence methods.
- Explain how recurrence ties to complexity analysis.
- Apply recurrence relations in a given scenario.

- Convert a verbal specification into a Boolean expression
- Explain basic properties of Boolean algebra: duality, complements, standard forms.
- Apply Boolean algebra to prove identities and simplify expressions.
- Translate verbal specifications into Boolean expressions and state machines.
- Use Karnaugh maps to find minimal sum-of-products and products-of-sums expressions.

- Explain the operation of discrete logic gates.
- Describe the relationship between Boolean algebra and electronic circuits.
- Analyze a combinational network using Boolean expressions.
- Design simple combinational networks that use NAND, NOR, and XOR gates.
- Design with MSI components such as encoders, decoders, multiplexers, adders, arithmetic-logic units, ROMs, and simple programmable logic arrays
- Calculate delays in ripple carry adders and simple combinational arrays

- Sets, Relations, and Functions
- Proof Techniques
- Basics of Counting
- Basic Logic
- Graph & Trees

- Discrete Probability
- Recurrence Relations
- Boolean Algebra & Expressions
- Combinatorial Circuits

August 1, 2023