PUBLIC SCHOOLS OF NORTH CAROLINA
State Board of Education
Department of Public Instruction

North Carolina Standard Course of Study
Discrete Mathematics for Computer Science *(Draft 3)*

Note on Numbering:
Discrete Math for Computer Science (DCS) Number and Quantity (N) Functions (F) Statistics and Probability (SP) Graph Theory (GT)

Discrete Mathematics for Computer Science Course Description:

The purpose of this course is to introduce discrete structures that are the backbone of computer science. Discrete mathematics is the study of mathematical structures that are countable or otherwise distinct and separable. The mathematics of modern computer science is built almost entirely on discrete mathematics, such as logic, combinatorics, proof, and graph theory. At most universities, an undergraduate-level course in discrete mathematics is required for students who plan to pursue careers as computer programmers, software engineers, data scientists, security analysts and financial analysts. Students will be prepared for college level algebra, statistics, and discrete mathematics courses.

Standards for Mathematical Practice	
1. Make sense of problems and persevere in solving them.	6. Attend to precision.
2. Reason abstractly and quantitatively.	7. Look for and make use of structure.
3. Construct viable arguments and critique the reasoning of others.	8. Look for and express regularity in repeated reasoning.
4. Model with mathematics.	9. Use strategies and procedures flexibly.
5. Use appropriate tools strategically.	10. Reflect on mistakes and misconceptions.

Number and Quantity

DCS.N.1 Apply operations with matrices and vectors.	
DCS.N.1.1	Implement procedures of addition, subtraction, multiplication, and scalar multiplication on matrices.
DCS.N.1.2	Implement procedures of addition, subtraction, and scalar multiplication on vectors.
DCS.N.1.3	Implement procedures to find the inverse of a matrix.
DCS.N.2 Understand matrices to solve problems.	
DCS.N.2.1	Organize data into matrices to solve problems.
DCS.N.2.2	Interpret solutions found using matrix operations including Leslie Models and Markov Chains, in context.
DCS.N.2.3	Represent a system of equations as a matrix equation.
DCS.N.2.4	Use inverse matrices to solve a system of equations with technology.
DCS.N.3 Understand set theory to solve problems.	
DCS.N.3.1	Recognize sets, subsets, and proper subsets.
DCS.N.3.2	Implement set operations to find unions, intersections, complements and set differences with multiple sets.
DCS.N.3.3	Represent properties and relationships among sets using Venn diagrams.
DCS.N.3.4	Interpret Venn diagrams to solve problems.
DCS.N. 4 Understand statements related to number theory and set theory.	
DCS.N.4.1	Use the Euclidean Algorithm to determine greatest common factor and least common multiple.
DCS.N.4.2	Use the Fundamental Theorem of Arithmetic to solve problems.
DCS.N.4.3	Conclude that sets are equal using the properties of set operations.

DCS.N.4.4	Explain theorems related to greatest common factor, least common multiple, even numbers, odd numbers, prime numbers, and composite numbers.

Functions

DCS.F.1 Apply recursively-defined relationships to solve problems.	
DCS.F.1.1	Implement procedures to find the *nth* term in an arithmetic or geometric sequence using spreadsheets.
DCS.F.1.2	Represent the sum of a sequence using sigma notation.
DCS.F.1.3	Implement procedures to find the sum of a finite sequence.
DCS.F.1.4	Implement procedures to find the sum of an infinite sequence and determine if the series converges or diverges.
DCS.F.1.5	Interpret the solutions to arithmetic and geometric sequences and series problems, in context.

Statistics and Probability

DCS.SP.1 Apply combinatorics concepts to solve problems.	
DCS.SP.1.1	Implement the Fundamental Counting Principle to solve problems.
DCS.SP.1.2	Implement procedures to calculate a permutation or combination.

Graph Theory

DCS.GT.1 Understand graph theory to model relationships and solve problems.	
DCS.GT.1.1	Represent real world situations using a vertex-edge graph, adjacency matrix, and vertex-edge table.
DCS.GT.1.2	Test graphs and digraphs for Euler paths, Euler circuits, Hamiltonian paths, or Hamiltonian circuits.
DCS.GT.1.3	Interpret a complete digraph to determine rank.
DCS.GT.2 Apply graph theory to solve problems.	
DCS.GT.2.1	Implement critical path analysis algorithms to determine the minimum project time.
DCS.GT.2.2	Implement the brute force method, the nearest-neighbor algorithm, and the cheapest-link algorithm to find solutions to a Traveling Salesperson Problem.
DCS.GT.2.3	Implement vertex-coloring techniques to solve problems.
DCS.GT.2.4	Implement Kruskal and Prim's algorithms to determine the weight of the minimum spanning tree of a connected graph.

Logic

DCS.L.1 Evaluate mathematical logic to model and solve problems.	
DCS.L.1.1	Construct truth tables that encode the truth and falsity of two or more statements.
DCS.L.1.2	Critique logic arguments (e.g., determine if a statement is valid or whether an argument is a tautology or contradiction).
DCS.L.1.3	Check 1s and 0s to determine whether a statement is true or false using Boolean logic circuits.
DCS.L.1.4	Judge whether two statements are logically equivalent using truth tables.