Discrete Math: Learning Objectives

Logic, Sets, Functions, Proofs

L.1 I can make a truth table for a compound proposition, determine the truth value of a compound proposition, and determine when two propositions are logically equivalent.

L.2 I can determine the truth of a proposition with quantifiers, or the equivalence of two such propositions. I can negate a quantified statement. I can give examples of true and false propositions with quantifiers.

L.3 I can determine relationships between elements and sets that have been defined using set notation (e.g., Is $x \in A$? Is $A \subseteq B$? Write all elements in the powerset of A.).

L.4 I can determine if an integer, real, or rational valued function is one-to-one, onto, or bijective. I can give examples of functions satisfying combinations of these properties.

L.5 I can set up a framework of assumptions and conclusions for proofs using direct proof, proof by contrapositive, and proof by contradiction.

Relations

R.1 I can give examples of relations on a set that have combinations of the properties of reflexivity, symmetry, antisymmetry, and transitivity.

R.2 I can determine when a set with a relation is a partially ordered set, or a totally ordered set. I can determine when a relation is an equivalence relation, and I can determine the equivalence class for an element and determine whether two elements belong to the same equivalence class.

Algorithms, Complexity, Induction & Recursion

I.1 I can determine the output of an algorithm written in pseudocode. I can compare the time complexity of two algorithms to determine which would be better suited to large inputs using "Big Oh" notation.

I.2 I can identify the predicate being used in a proof by mathematical induction and use it to set up a framework of assumptions and conclusions for an induction proof.

I.3 I can determine the output of a recursive algorithm for small input values, and use pattern recognition to determine the output in general. I can list elements contained in a recursively defined set, or give a recursive definition of an explicitly defined set.

I.4 I can solve a linear homogeneous recurrence relation to find an explicit formula for the n-th term in a sequence.

Counting & Probability

C.1 I can perform the extended Euclidean algorithm to determine the greatest common divisor of two numbers, and to write the gcd as a linear combination of these numbers.

C.2 I can count permutations, combinations, and subsets in a variety of scenarios, confidently using the sum and/or product rules as appropriate.

C.3 I can count permutations with repetitions, or the number of ways to distribute a set of n items between k people, using equivalence and the inclusion/exclusion principle appropriately. I can use the pigeonhole principle.

C.4 Given an experiment with a discrete sample space, and a probability distribution on said sample space, I can compute the probability of an event occurring. I can also determine the probability distribution on a sample space for basic examples.

C.5 I can compute conditional probabilities, and probabilities for unions, complements, or independent events.

C.6 I can give examples of random variables satisfying specified properties. I can compute the range, distribution, and expectation of a random variable.

Graphs & Trees

G.1 I can create a graph given information or rules about vertices and edges. I can give examples of graphs having combinations of various properties and examples of graphs of special ("named") types.

G.2 I can represent a graph in different ways and change representations from one to another. I can determine whether or not two graphs are isomorphic, and justify my conclusion.

G.3 I can construct a planar embedding of a (planar) graph. I can apply Euler's identity (or its corollaries) to make conclusions about the planarity of a graph.

G.4 I can determine whether a graph has an Euler trail (or circuit), or a Hamiltonian path (or cycle), and I can clearly explain my reasoning.

G.5 I can list the nodes of a tree in the correct order when visited using preorder and postorder traversals. I can create a spanning tree for a graph using BFS or DFS algorithms.