INTRODUCTION

Discrete Structures

Discrete structures is the body of knowledge devoted to the study of discrete (as opposed to continuous) objects. Discrete objects form *countably* finite or infinite sets. For example, natural numbers, integers, number of CPU cycles spent executing an instruction, number of packets a message is decomposed into, number of routes between two nodes in a network, etc. *Life* also processes discrete information, stored as sequence of only four molecules represented as A, G, C, and T in DNA, the structure that stores all heredity information in cells.

On the other hand, continuous objects form *uncountably* infinite sets. For example, real numbers, power consumption of devices, time taken by a packet to travel from source to destination node in a network, etc.[1]

Motivation for the Course

All digital devices process discrete quantities and this inspires the significance of understanding discrete objects and their relation with each other. The joint undertaking of the Institute for Electrical and Electronic Engineers/Computer Society (IEEE/CS) and the Association for Computing Machinery (ACM) has identified a set of knowledge units "for which there is a broad consensus that the material is essential to an undergraduate degree in computer science." The joint forum describes Discrete Structures as "foundational material for computer science".

A Glimpse of Problems that can be solved using Discrete Mathematics

- Count of valid Internet addresses
- Number of ways to choose a valid password on a computer system
- Establishing if there is a link between two computers in a network
- Detecting spam emails
- Data encryption/decryption
- Locating shortest path between two cities using a transportation system or between two nodes on a computer network.

Pointers to other fields in CS

The course topics underlie much of Computer Science as highlighted below:

- Logic → Knowledge Representation, Reasoning, Natural Language Processing, Computer Architecture
- Proof Techniques → Algorithm Design, Code Verification
- Relations → Database Systems
- Functions → Hashing, Programming Languages
- Recurrence Relations → Recursive Algorithm Analysis
- Probability → Algorithm Design, Simulation, Big Data, Machine Learning

Course Objectives

- *Precise & Dependable Thought Process*
  To enable students to use and innovate sophisticated ideas for problem solving rather than just relying on the obvious “brute force” approach.
- *Writing Proofs*
  To enable students to establish formal proofs especially for proving correctness of hardware and programs, that in turn improves reliability of systems.

[1] *Calculus* deals with continuous objects.