CO2453: Discrete Structures
SGSITS, Indore

Discrete v/s Continuous objects

Discrete Objects:

Discrete objects are those which are separated from (not connected to/distinct from) each other.

Example:

- Integers (aka whole numbers),
- rational numbers (ones that can be expressed as the quotient of two integers),
- automobiles,
- houses,
- people etc. are all discrete objects
- steps taken by a computer program,
- distinct paths to travel from point A to point B on a map along a road network,
- ways to pick a winning set of numbers in a lottery.!

Discrete v/s Continuous objects

Continuous Objects:

Objects that are packed without any gaps and can not be separated from their immediate neighbors.

Example:

Real numbers which include irrational as well as rational numbers are not discrete. As you know between any two different real numbers there is another real number different from either of them.

Discrete Mathematics

Discrete mathematics is the branch of mathematics dealing with objects that can assume only distinct, separated values.

The term "discrete mathematics" is therefore used in contrast with "continuous mathematics," which is the branch of mathematics dealing with objects that can vary smoothly (and which includes, for example, calculus).

Discrete math concerns sets of objects that are countable where as continuous math concerns sets of objects that are "measurable".

More Examples

There are some intuitive ways of thinking about the discrete-vs.-continuous distinction:

1.

A piano makes music in a "discrete" way:

On a piano, there is no note between C and C#.

But a violin makes music in a "continuous" way:

There are (in theory at least) a continuum of notes between C and C#.

More Examples

2.

An analog clock (i.e., a clock with hands) is (at least in theory) continuous:

The set of times that it can show is not only dense, but it can even tell you when it is π o'clock!

But a digital clock is discrete:

In a typical digital clock, there is no time between 3:14 PM and 3:15 PM.

3.

A film, with "frames", is discrete.

The real events depicted on the film are continuous.

Why Discrete Mathematics in Computer Science Course

Discrete Mathematics is the foundation of many courses. A course in discrete mathematics provides the mathematical background needed for all subsequent courses in computer science.

Databases, Neural Network, Compilers, Computer Architecture, Network Programming, Computer Security, Game theory, Programming languages, Artificial Intelligence, Theory of Computation, Data Structures, Graphics, Algorithms, Operating System, Software Engineering…

Need of Discrete Mathematics

The Analysis of Programs

There are different algorithm to sort the numbers and each algorithm take different times to sort the same list of numbers. But is one better than the other? If so, in what way is it better, and by what measure?

Thus, to compare their relative efficiency, we would like to analyse these algorithms mathematically.

Need of Discrete Mathematics

Formal Specification of Requirements

When software or hardware producers start on a project to design a system, the first step is to identify and document the requirements of the client or user. Most projects use informal notation (such as English), which means that ambiguity is inevitable, and formal analysis impossible.

Thus, mathematics when used to document requirements is effective in describing things in a precise yet abstract fashion.

Need of Discrete Mathematics

Reasoning About Programs

Once we have a specification of the requirements and the program itself, we would like to know whether or not the program works as expected. One way to do this is to feed the program with a large number of inputs and check that the outputs are as expected. Even if it is usually not possible to test with every possible input, various techniques have been developed to intelligently choose values to test for or to quantify how much of the system's potential behaviour has been tested.

To address this, computer scientists have developed and used various logics and mathematical tools to give meaning to computer programs. *To check that a particular program satisfies a* certain requirement, it suffices to translate the program into logic and prove that the requirements always follow from this logic description.

Problems solved using discrete structures

- What is the shortest path between two cities using a transportation system?
- Find the shortest tour that visits each of a group of cities only once and then ends in the starting city?
- How can we represent English sentences so that a computer can reason with them?
- How can we prove that there are infinitely many prime numbers?
- How can a list of integers be sorted so that the integers are in increasing order?
- How many steps are required to do such a sorting?
- How can it be proved that a sorting algorithm always correctly sorts a list?

Problems solved using discrete structures

- How many ways are there to choose a valid password chosen given specific rules?
- How many valid Internet addresses are there?
- What is the probability of winning a particular lottery?
- Is there a link between two computers in a network?
- How can I identify spam email messages?
- How can I encrypt a message so that no unintended recipient can read it?
- How can we build a circuit that adds two integers?

And many more…

Outline of the Course

- Propositional Logic
- Predicates and Quantifiers
- Set Theory
- Relations and Functions
- Languages and Grammars
- Finite State Machines
- Counting Techniques
- Generating Functions
- Recurrence Relations
- Complexity of Algorithms
- Algebraic Systems
- Boolean Algebra
- Graphs and Trees
- Application of discrete structures in the field of Computer Science