Outline

Introduction

What is discrete math? How can it help us?

Propositional Logic

Applications: Digital logic, if-statements in computer programs

Sets and Boolean Algebra

Set building notations. Set operations. Reasoning about sets.

Application: Hamming codes

Predicate Logic

Application: preconditions, postconditions, assertions; temporal properties of systems.

Integers, Mathematical Reasoning, and Induction

Application: RSA encryption.

Counting and Recurence Relations

Application: Counting AVL trees.

Functions and Relations

Application: Cryptology. Relational databases. System models. Functional models of computer programs. Relational specifications of computer programs.

Graphs and Trees

Applications: Shortest distance/maximal reliability/etc. PCB testing. Maximum flow. Minimal spanning trees for power or data networks. Huffman encoding.

Models of Computation

Application: Design of discrete systems. Design of languages. Safety analysis of systems.

Unit 0: Introduction

Reading: Gosset pp. xxiii—xxiv and ch. 1.

What is discrete math?

- The real numbers are *continuous* in the senses that:
  - $*$ between any two real numbers there is a real number
- The integers do not share this property.

  In this sense the integers are lumpy, or “discrete”

Discrete math is the study of mathematical objects that are in some sense lumpy.

Some discrete mathematical concepts:

- Integers: Between two integers there is not another integer.
- Propositions: Either true or false, there are no 1/2 truths (in math)
- Sets: An item is either in a set or not in a set, never partly in and partly out.
- Relations: A pair of items are related or not.

- Networks (graphs): Between two terminals of a direct network connection there are no other terminals.

Relevence to ECE

Discrete math is relevent to computer and electrical engineering because we often deal with objects with discrete properties.

Consider a digital video clip:

- Pixels are discrete objects in space.
- Frames are discrete in time
- Colour frequency is discretized into 3 hues: red, green, and blue.
- For each hue, intensity comes in discrete levels, e.g. 256 per colour.

In this case the discrete values represent continuous time, space, frequency, and intensity variables.

Often discrete variables are used to represent discrete rather than continuous phenomena:

- The states of a computer program are discrete.
- The states of a digital hardware design are discrete.

- Discrete structures (sets, functions, relations, trees, graphs) are very useful for representing data in computers.
- Connecting a generator or load to a power grid is a discrete change.
- Two computers in a network either are or are not directly connected.

Unifying abstraction

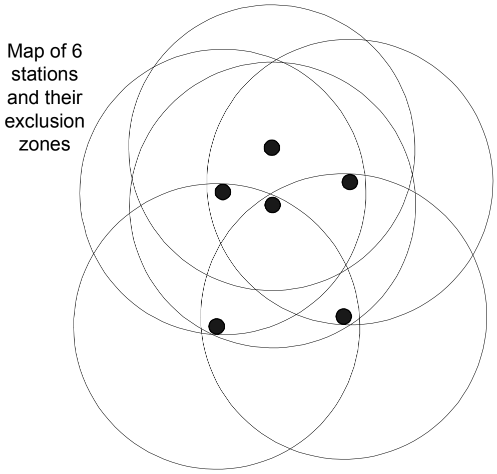
Mathematics provides abstractions that unify seemingly different situations.

- For example a resistor/capacitor/inductor network and a mass/spring system may have the same continuous mathematical model.
- Discrete math also provides a set of unifying abstractions.
- At some point about 30,000 years ago it was realized that one could use marks on a stick to count either bison killed, or people in a tribe, thus a unifying abstraction — the integers— was born.

An example

The TV channel problem:

- We have a bunch of TV stations geographically distributed
- Each base station must be assigned one of 11 VHF channels
- Channels that are within 250 km must be assigned different channels
- Can we assign channels? What is the minimum number of channels required.



The exam scheduling problem:

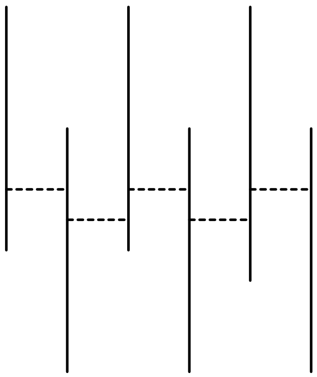
- A University has a bunch of courses to give
- Each course must be assigned one of, say 50, exam slots
- Courses that share one or more students must be assigned different slots.
- Can we assign slots to courses? What is the minimum number of slots required.

Example: 6 courses and some of their students. What is minimum # of slots required?

Engl 1984	Smith, Jones, Chan,...
Math 3141	Smith, Xu, Negm,...
Physics 2998	Smith, Lobel, Suess,...
Chem 6022	Jones, Xu, Lobel, ...
Engi 3422	Jones, Brown, ...
CompSci 0101	Negm, Brown, ...

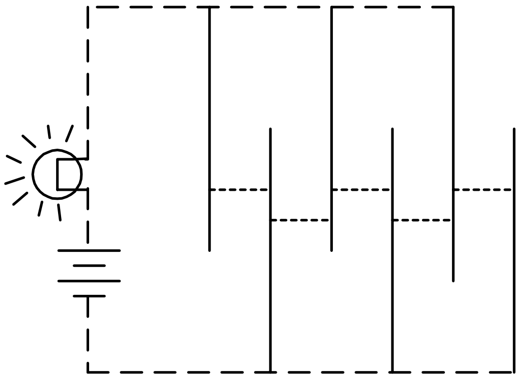
The PCB testing problem:

- A company manufactures printed circuit boards (PCBs)
- Each board has, say, 100 “traces” (a trace is a conducting path on the PCB)
- They wish to test for short circuits between pairs of traces.
- Of the 4950 potential short circuits, all but 2000 are judged physically impossible.
- Still 2000 tests would take too long.
- So we will assign traces to N groups such that no two traces in the same group can be shorted.
- Then for each pair of groups we temporarilly connect all members of each group and then test for continuity between the groups.
- The total number of tests is $N(N-1)/2$ — potentially much less than 2000.
- E.g. 10 groups of 10 traces means 45 tests suffice.
- What is the mimimum number N of groups?
- A simple example

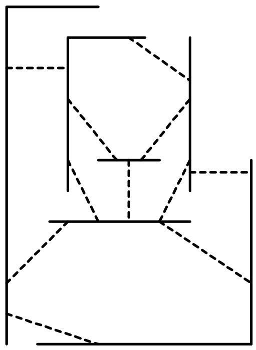


6 Traces

5 Physically possible shorts



Test circuitry (temporary)



6 Traces

11 Physically possible shorts

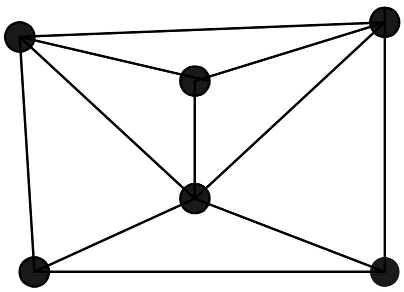
Unifying the examples

These problems all seem rather different on the surface.

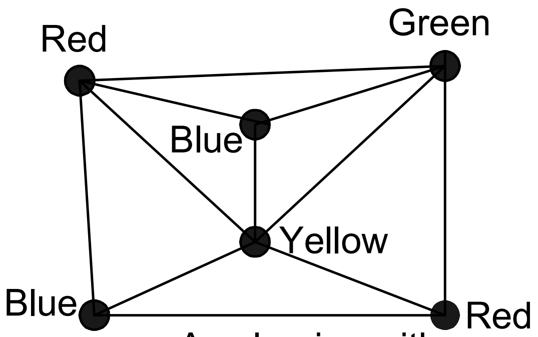
But mathematically they are closely related.

To see this we need a “unifying abstraction”.

- Consider a finite number of points on a page
- And a bunch of lines each connecting 2 points.
- Such a collection of points and lines is called a *“graph”*.
- We wish to assign each point a colour
  - ∗ such that no line connects two points of the same colour
- Problem: Given a graph, what is the minimum number of colours required?
- Example



A Graph



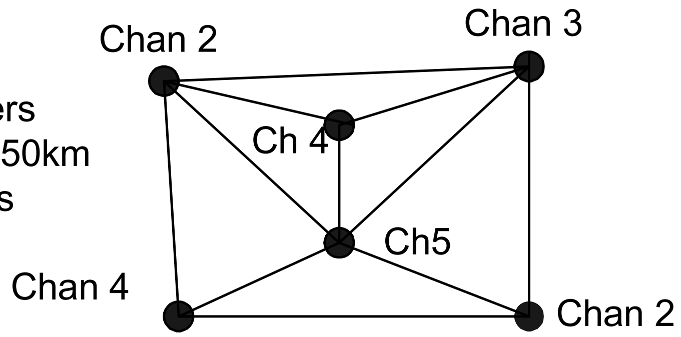
A colouring with 4 colours. This is minimal

This problem is called the “graph vertex colouring problem”.

We can solve our 3 problems if we can find minimal colourings for graphs.

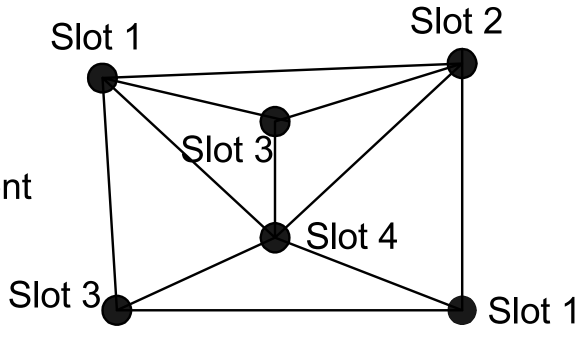
Assigning channels
to transmitters

Points represent transmitters
Line if transmitters within 250km
Colours represent channels



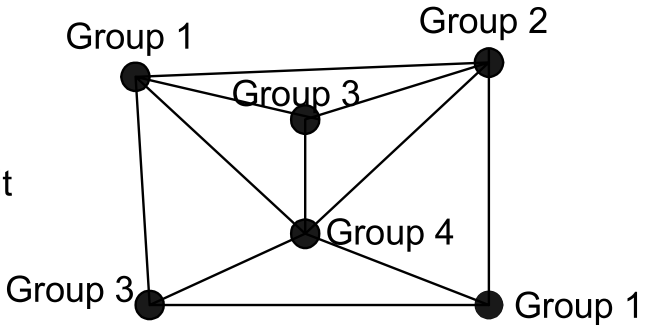
Assigning courses
to exam slots

Points represent courses
Line if courses share a student
Colours represent slots



Assigning traces to groups

Points represent traces
Line if potential short circuit
Colours represent groups



Graph colouring has many other applications: E.g. assignment of variables to registers in compiling computer programs

So graphs provide a *Unifying Abstraction*.

- Knowlege about graphs can be applied to problems in diverse fields
- including problems that haven't yet been encountered

And graphs have many applications beyond colouring

And there is much to discrete math beyond graphs