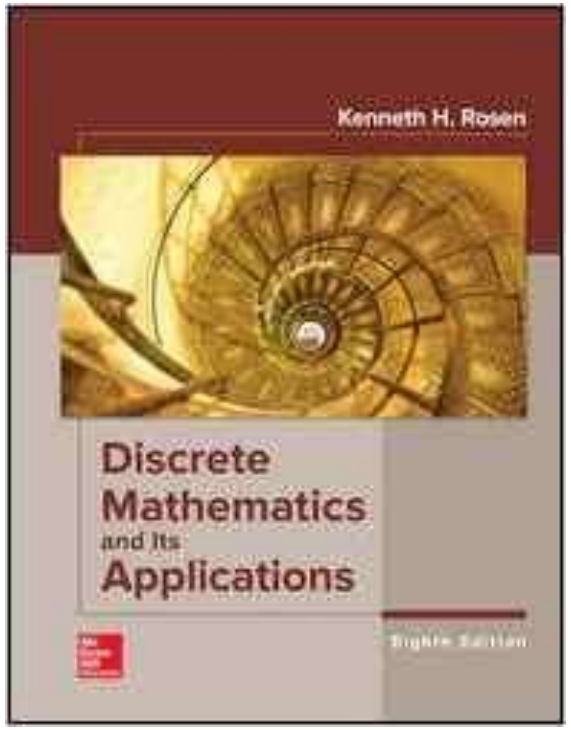
CSCE 222
Discrete Structures
Introduction

Dr. Tim McGuire

Grateful acknowledgement to Professor Bart Selman, Cornell University, and Prof. Johnnie Baker, Kent State, for some of the material upon which these notes are adapted.

Textbook



Discrete Mathematics and Its Applications
by Kenneth H. Rosen

Use lecture notes as study guide.

Syllabus Review

What is CSCE 222 about?

- Continuous vs. Discrete Math
- Why is it computer science?
- Mathematical techniques for Discrete Structures

So why do I need to learn all this nasty mathematics?

- Computation is something that you can't see and can't touch, and yet (thanks to the efforts of generations of hardware engineers) it obeys strict, well-defined rules with astonishing accuracy over long periods of time.
- Computations are too big for you to comprehend all at once. Imagine printing out an execution trace that showed every operation a typical $500 laptop computer executed in one (1) second. If you could read one operation per second, for eight hours every day, you would die of old age before you got halfway through. Now, imagine letting the computer run overnight.

So why do I need to learn all this nasty mathematics?

- So in order to understand computations, we need a language that allows us to reason about things we can't see and can't touch, that are too big or us to understand, but that nonetheless follow strict, simple, well-defined rules.
- We'd like our reasoning to be consistent: any two people using the language should obtain the same conclusions from the same information.
- That language is mathematics.

Discrete vs. Continuous Mathematics

Continuous Mathematics: It considers objects that vary continuously;

Example: analog wristwatch (separate hour, minute, and second hands).

From an analog watch perspective, between 1 :25 p.m. and 1 :26 p.m. there are infinitely many possible different times as the second hand moves around the watch face.

Real-number system --- core of continuous mathematics;

Continuous mathematics --- models and tools for analyzing real-world phenomena that change smoothly over time. (Differential equations etc.)

Discrete vs. Continuous Mathematics

Discrete Mathematics: It considers objects that vary in a discrete way.

Example: digital wristwatch.

On a digital watch, there are only finitely many possible different times between 1 :25 P.M. and 1:27 P.M. A digital watch does not show split seconds: - no time between 1 :25:03 and 1 :25:04. The watch moves from one time to the next.

Integers --- *core of discrete mathematics*

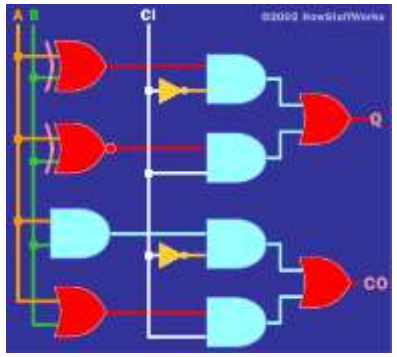
Discrete mathematics --- models and tools for analyzing real-world phenomena that change discretely over time and therefore ideal for studying

computer science – computers are digital! (numbers as finite bit strings; data structures, all discrete! Historical aside: earliest computers were analog.)

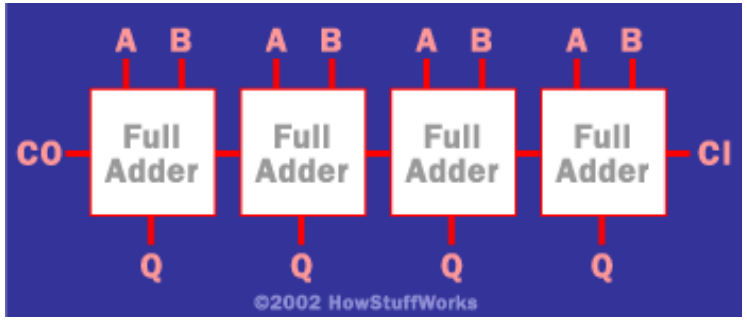
What is CSCE 222 about?

- Why is it computer science? (examples)

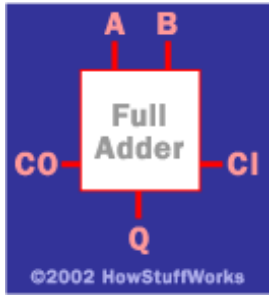
Logic:
Hardware and software specifications



One-bit Full Adder with Carry-In and Carry-Out



4-bit full adder



Formal: Input_wire_A
value in {0, 1}

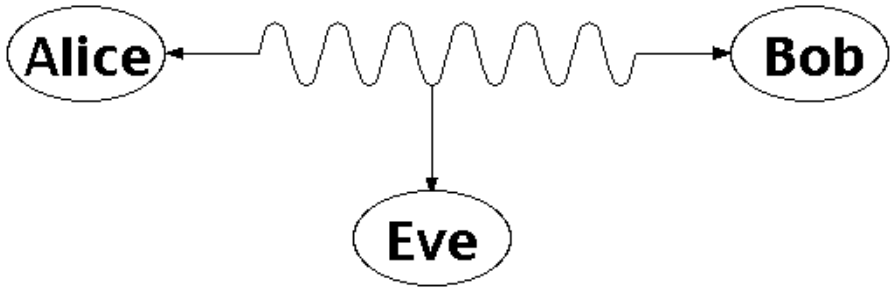
Example 1: Adder

Example 2: System Specification:

–The router can send packets to the edge system only if it supports the new address space.
– For the router to support the new address space it's necessary that the latest software release be installed.
–The router can send packets to the edge system if the latest software release is installed.
–The router does not support the new address space.

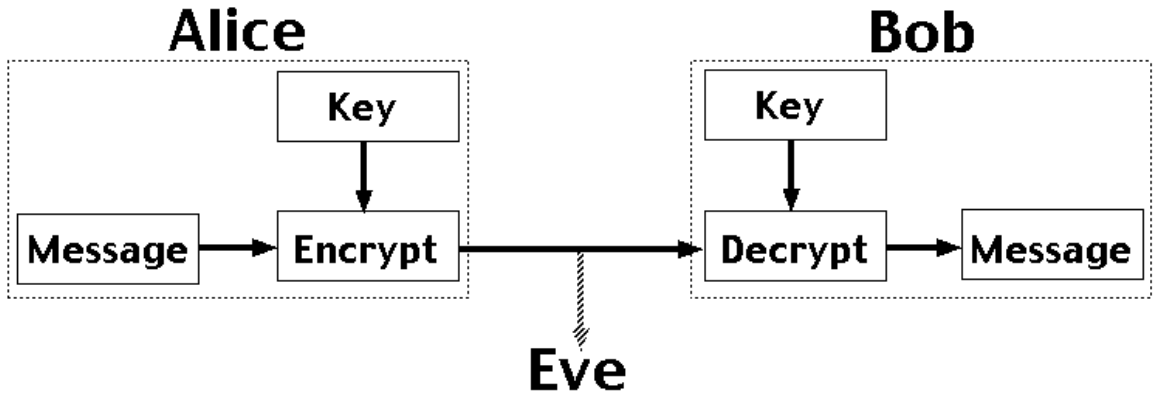
How to write these specifications in a rigorous / formal way? ***Use Logic.***

Number Theory:
RSA and Public-key Cryptography



Alice and Bob have never met but they would like to exchange a message. Eve would like to eavesdrop.

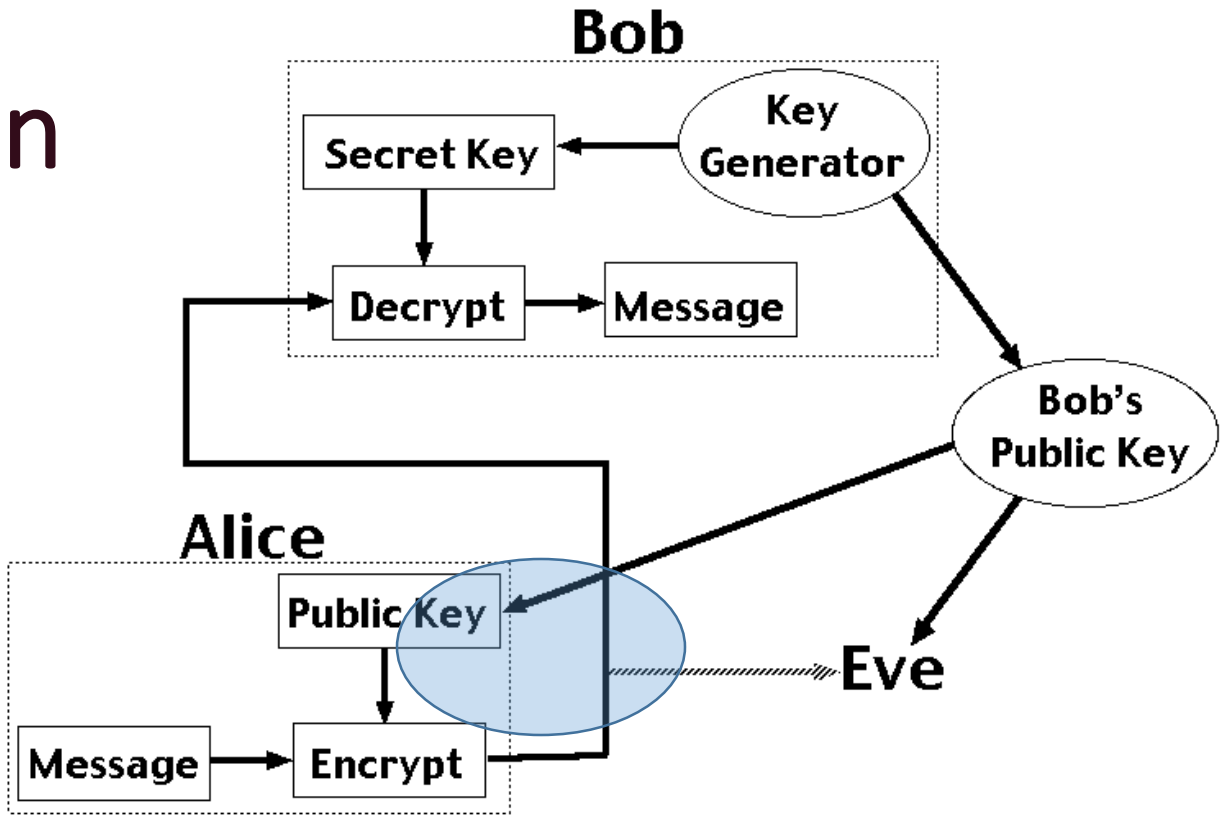
E.g. between you and the Bank of America.



They could come up with a good encryption algorithm and exchange the encryption key – but how to do it without Eve getting it? (If Eve gets it, all security is lost.)

CS folks found the solution:
public key encryption. Quite remarkable that this is feasible.

Number Theory: Public Key Encryption



RSA – Public Key Cryptosystem (why RSA?)

Uses modular arithmetic and large primes → Its security comes from the computational difficulty of factoring large numbers.

RSA Approach

Encode:

$C = M^e \pmod{n}$

M is the plaintext; C is ciphertext

n = pq with p and q large primes (e.g. 200 digits long!)

e is relative prime to (p-1)(q-1)

Decode:

$C^d = M \pmod{pq}$

d is inverse of e modulo (p-1)(q-1)

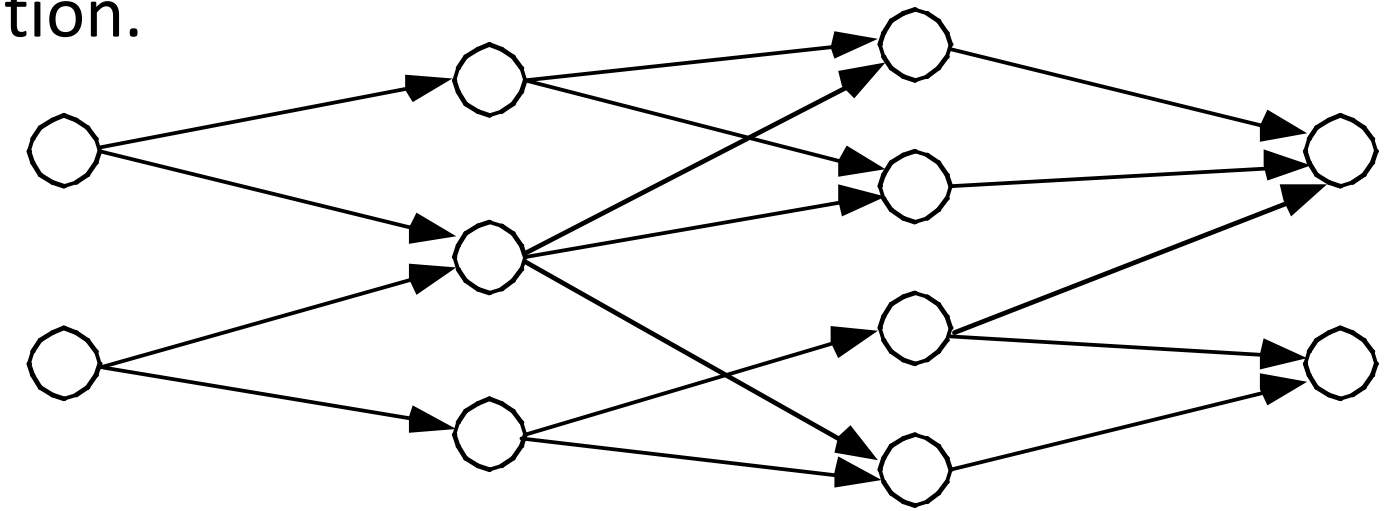
The process of encrypting and decrypting a message correctly results in the original message (and it's fast!)

Graph Theory

Graphs and Networks

- Many problems can be represented by a graphical network representation.

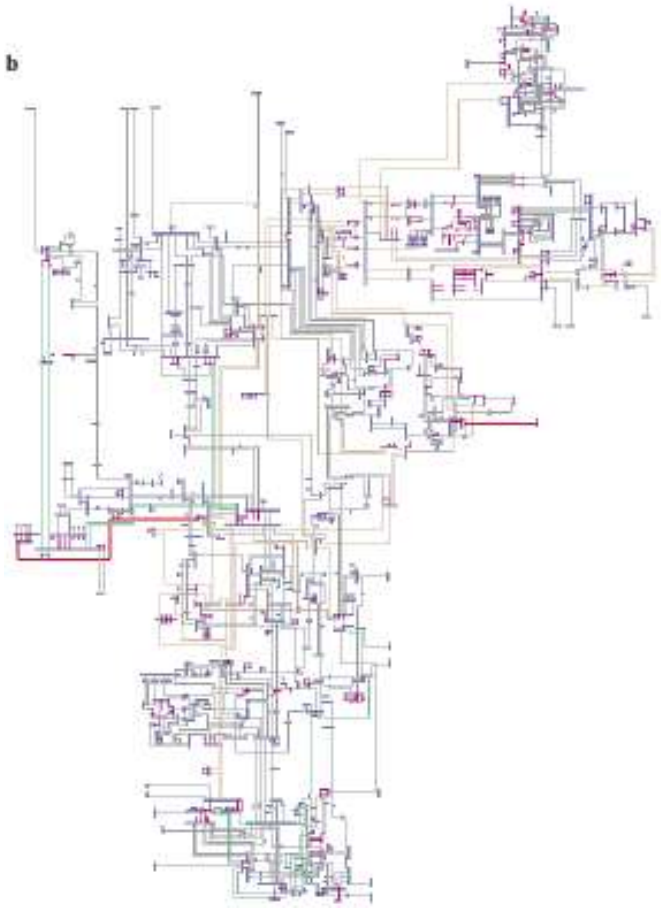
- Examples:
  - Distribution problems
  - Routing problems
  - Maximum flow problems
  - Designing computer / phone / road networks
  - Equipment replacement
  - And of course the Internet



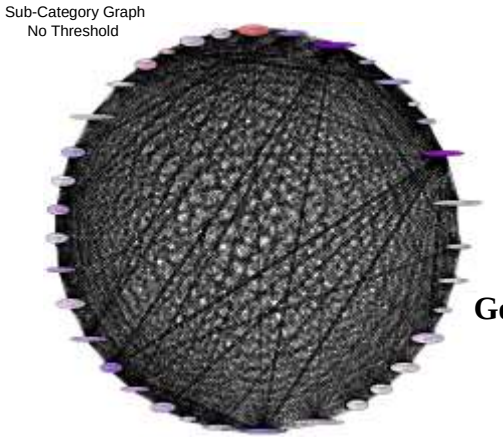
Aside: finding the right problem representation is one of the key issues.

New Science of Networks

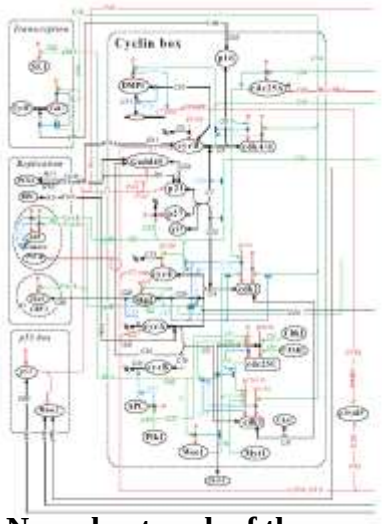
Networks are pervasive



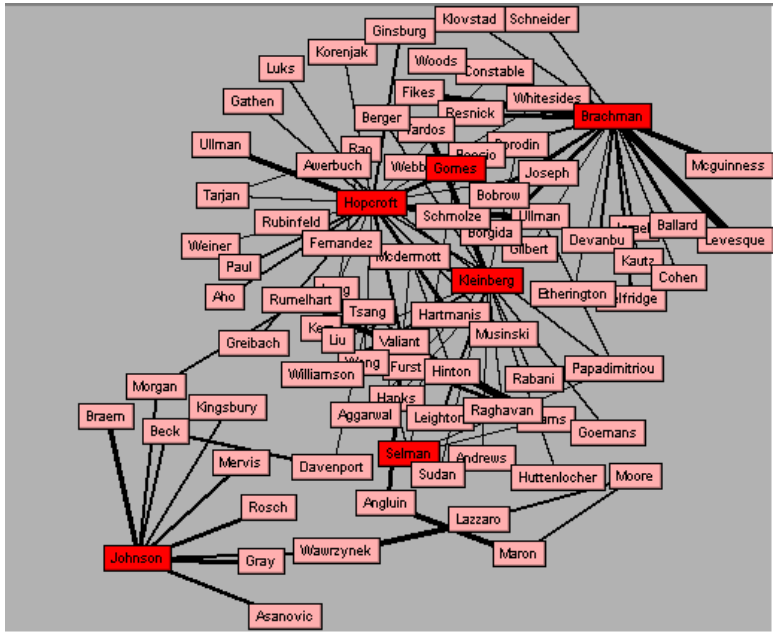
NYS Electric Power Grid
(Thorp,Strogatz,Watts)



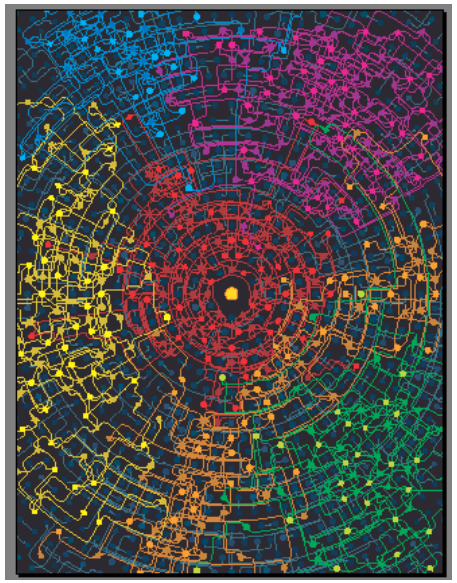
Utility Patent network
1972-1999
(3 Million patents)
Gomes,Hopcroft,Lesser,Selman



Neural network of the nematode worm C- elegans
(Strogatz, Watts)

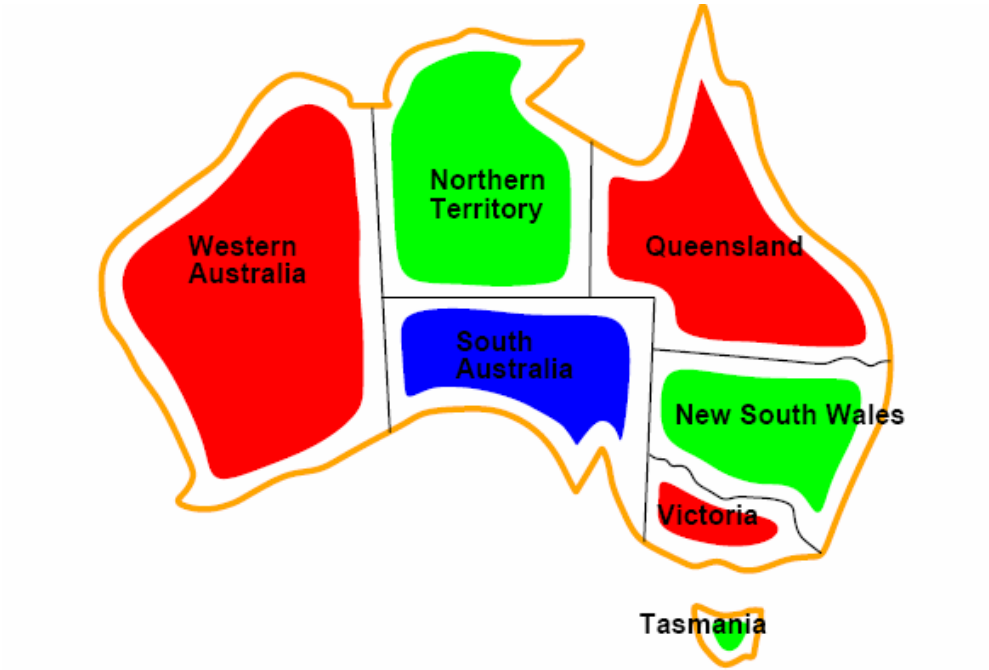


Network of computer scientists
ReferralWeb System
(Kautz and Selman)



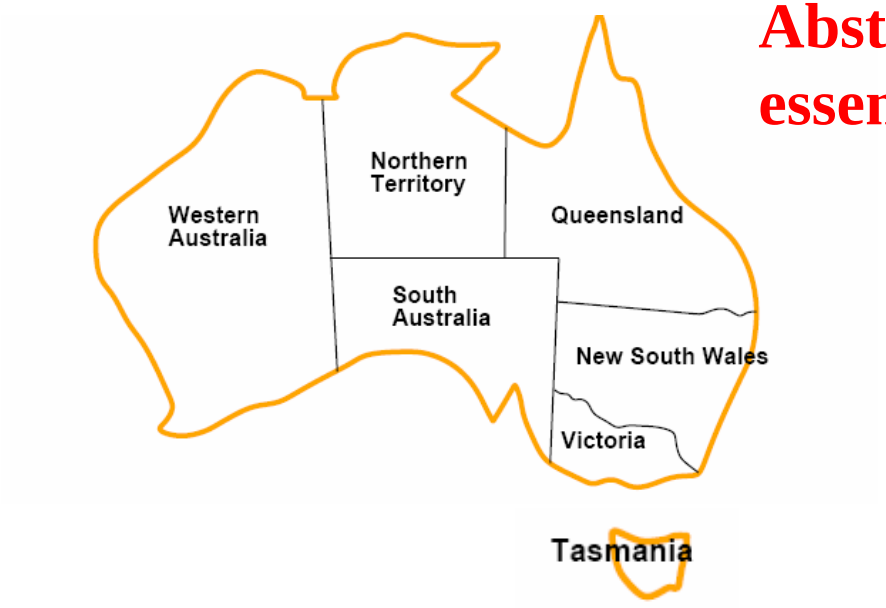
Cybercommunities
(Automatically discovered)
Kleinberg et al

Example: Coloring a Map

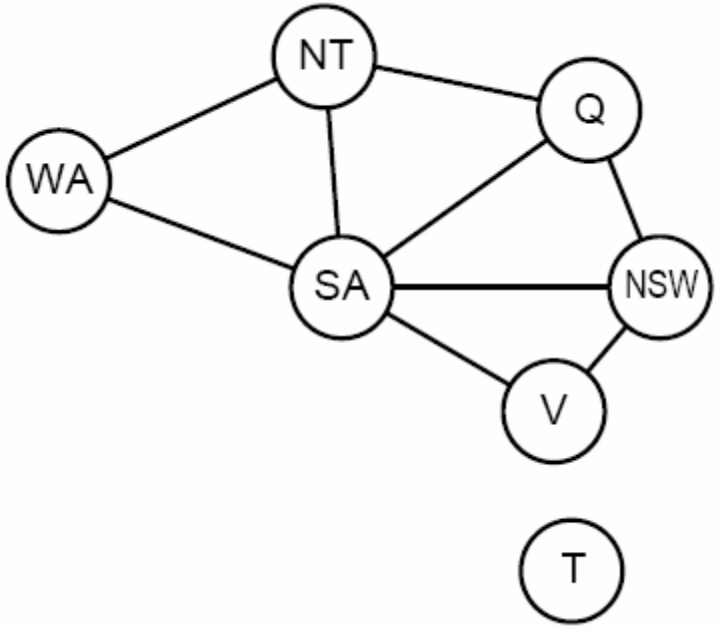


How to color this map so that no two adjacent regions have the same color?

Graph representation



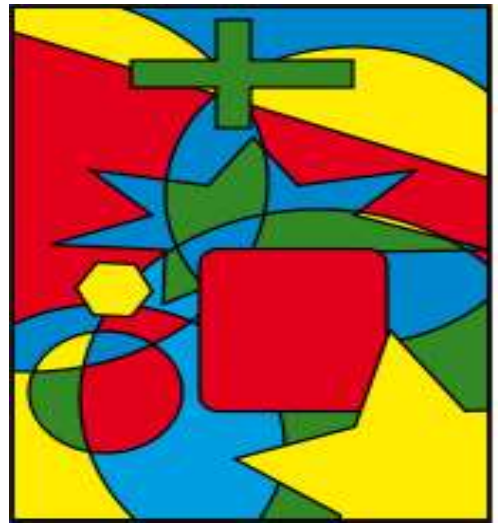
Abstract the essential info:



Coloring the nodes of the graph:
What's the minimum number of colors such that any two nodes connected by an edge have different colors?

Four Color Theorem

- The chromatic number of a graph is the least number of colors that are required to color a graph.
- The Four Color Theorem – ***the chromatic number of a planar graph is no greater than four. (quite surprising!)***



Four color map.

- Proof: Appel and Haken 1976; careful case analysis performed by computer; proof reduced the infinitude of possible maps to 1,936 reducible configurations (later reduced to 1,476) which had to be checked one by one by computer.
- The computer program ran for hundreds of hours. The first significant *computer-assisted* mathematical proof.
- *Write-up was hundreds of pages including code!*

How do we know the proof is actually correct?

(later CS folks to the rescue)

Examples of Applications of Graph Coloring

Scheduling of Final Exams

- How could the final exams at TAMU be scheduled so that no student has two exams at the same time? ***(Note not obvious this has anything to do with graphs or graph coloring!)***

Graph:

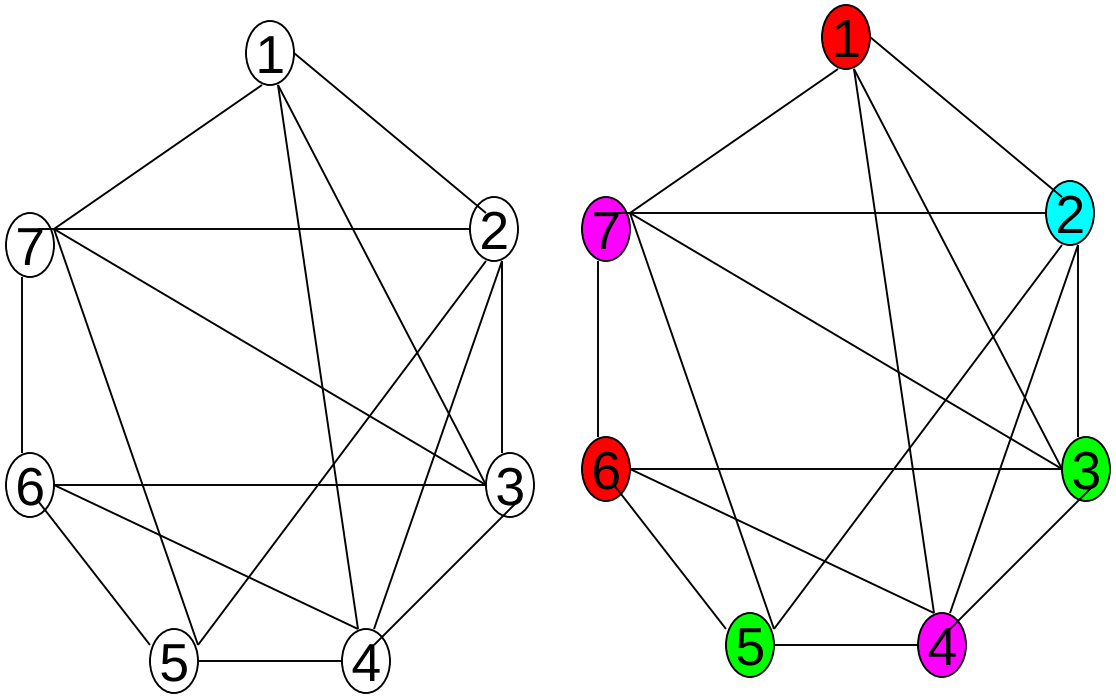
A vertex correspond to a course.

An edge between two vertices denotes that there is at least one common student in the courses they represent.

Each time slot for a final exam is represented by a different color.

A coloring of the graph corresponds to a valid schedule of the exams.

Scheduling of Final Exams



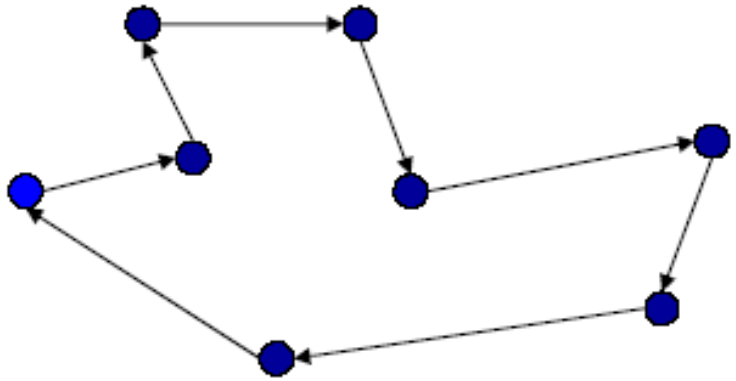
Time Period	Courses
I	1,6
II	2
III	3,5
IV	4,7

What are the constraints between courses?
Find a valid coloring

Why is mimimum number of colors useful?

Example 2: Traveling Salesman

Find a closed tour of minimum length visiting all the cities.



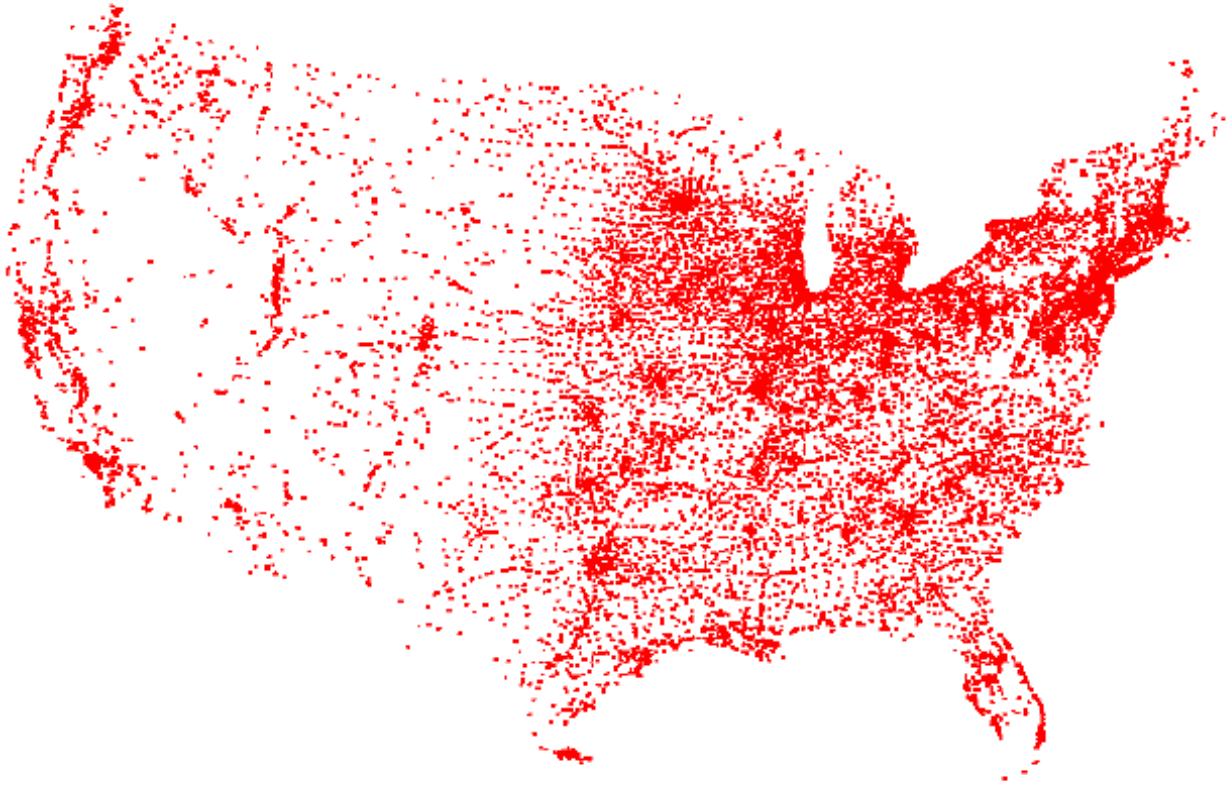
TSP → lots of applications:

Transportation related: scheduling deliveries

Many others: e.g., Scheduling of a machine to drill holes in a circuit board ;

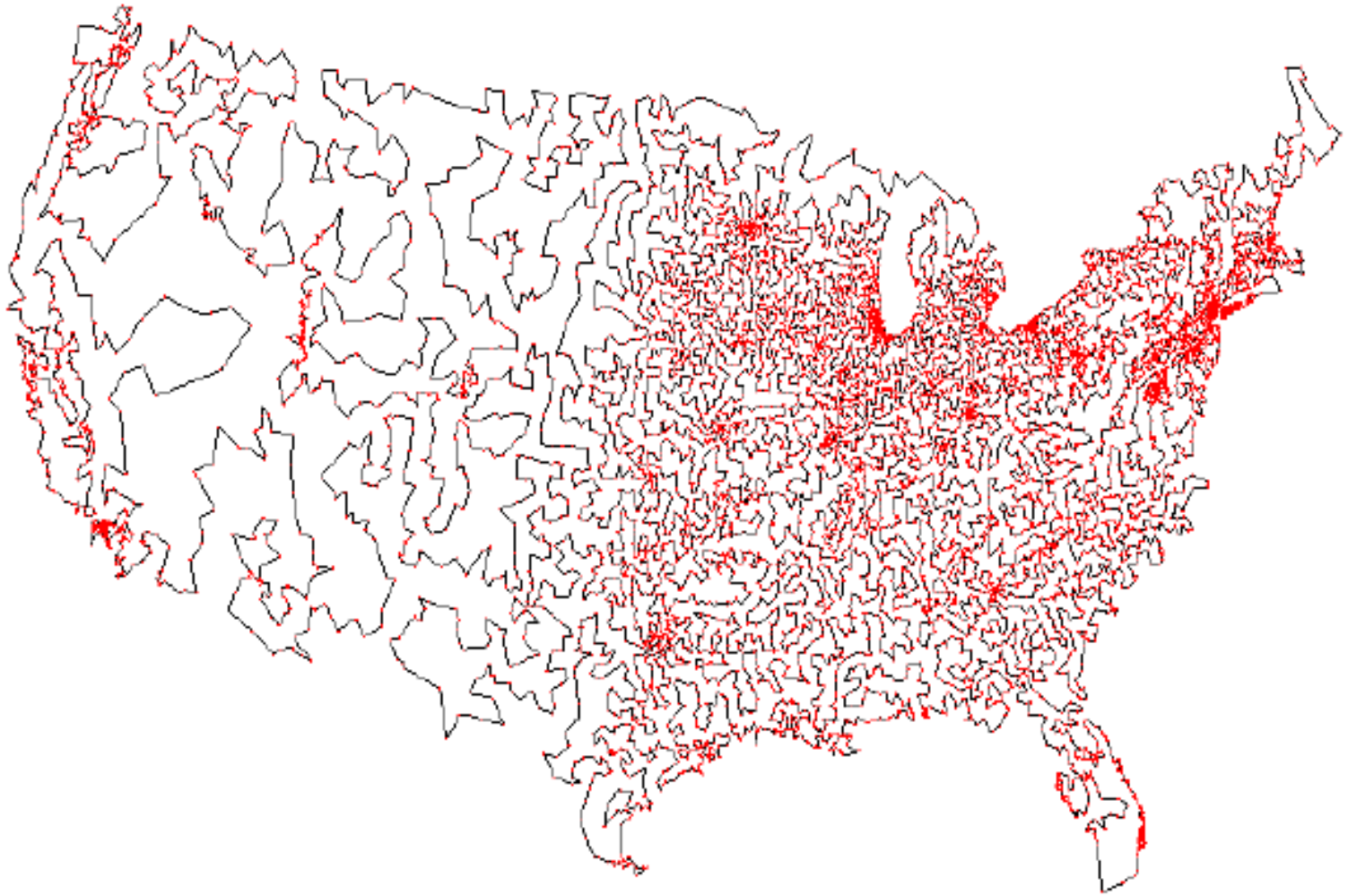
Genome sequencing; etc

13,509 cities in the US (with populations > 500)



13508!= 1.475977418846014819975134275320 8e+49936

13509 cities in the USA



The optimal tour!

(Applegate, Bixby, Chvatal and Cook, 1998)

Probability and Chance

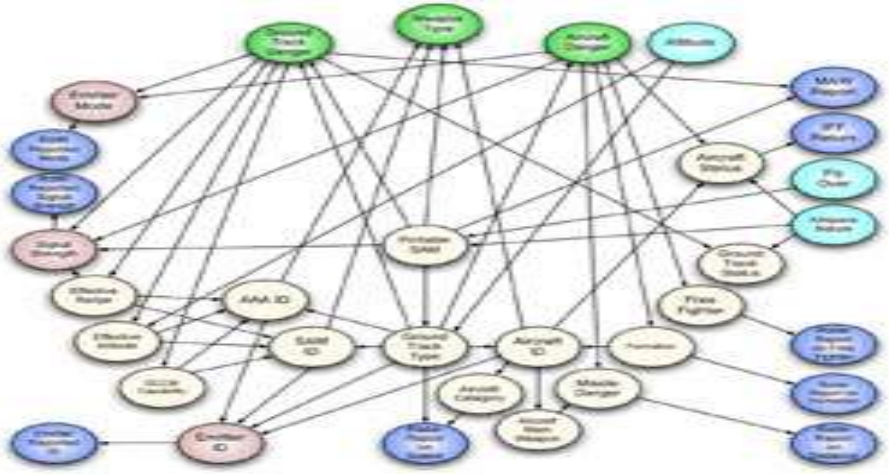
Importance of concepts from probability is rapidly increasing in CS:

- Randomized algorithms (e.g. primality testing; randomized search algorithms, such as simulated annealing, Google's PageRank, "just" a random walk on the web!) In computation, having a few random bits really helps!

- Machine Learning / Data Mining: Find statistical regularities in large amounts of data. (e.g. Naïve Bayes alg.)

- Natural language understanding: dealing with the ambiguity of language (words have multiple meanings, sentences have multiple parsings --- key: find the most likely (i.e., most probable) coherent interpretation of a sentence (the "holy grail" of NLU).

Probability: Bayesian Reasoning



Bayesian networks provide a means of expressing joint probability over many interrelated hypotheses and therefore reason about them.

Example of Query:
what is the most likely diagnosis for the infection given all the symptoms?

Bayesian networks have been successfully applied in diverse fields such as medical diagnosis, image recognition, language understanding, search algorithms, and many others.

$$P(\mathbf{x}|\omega_i) = \frac{P(\omega_i|\mathbf{x})P(\mathbf{x})}{P(\omega_i)}$$

Bayes Rule

Microsoft's David Hovel, Eric Horvitz, Carl Kadie and Andy Jacobs (from left) are putting Bayes' ideas to work in the company's Notification Platform.

"18th-century theory is new force in computing" CNET '07

Probability and Chance, cont.

Back to checking proofs...

Imagine a mathematical proof that is several thousands pages long. (e.g., the **classification of so-called finite simple groups**, also called *the enormous theorem, 5000+ pages)*.

How would you check it to make sure it's correct? Hmm…

Probability and Chance, cont.

Computer scientists have recently found a remarkable way to do this:

"holographic proofs"

- Ask the author of the proof to write it down in a special encoding (size increases to, say, 50,000 pages of 0 / 1 bits). You don't need to see the encoding! Instead, you ask the author to give you the values of **50** randomly picked bits of the proof. (i.e., "spot check the proof").
- *With almost absolute certainty, you can now determine whether the proof is correct of not!* (works also for 100 trillion page proofs, use eg 100 bits.) Aside: Do professors ever use "spot checking"?

Started with results from the early nineties (Arora et al. '92) with recent refinements (Dinur '06). Combines ideas from coding theory, probability, algebra, computation, and graph theory. It's an example of one of the latest advances in discrete mathematics. See Bernard Chazelle, *Nature* '07.

Course Themes, Goals, and Course Outline

Goals of CSCE 222

Introduce students to a range of mathematical tools from discrete mathematics that are key in computer science

Mathematical Sophistication

How to write statements rigorously

How to read and write theorems, lemmas, etc.

How to write rigorous proofs

Practice works!

Actually, *only* practice works!

Areas we will cover:

Logic and proofs
Set Theory
Induction and Recursion
Counting and combinatorics
Probability theory
Number Theory (if time permits)

Note: Learning to do proofs from watching the lecture is like trying to learn to play tennis from watching it on TV! So, do the exercises!

Aside: We're not after the shortest or most elegant proofs; verbose but rigorous is just fine! ☺

Topics CSCE 222

Logic and Methods of Proof

- Propositional Logic --- SAT as an encoding language!
- Predicates and Quantifiers
- Methods of Proofs
- Sets
- Sets and Set operations
- Functions

Counting

- Basics of counting
- Pigeonhole principle
- Permutations and Combinations
- Number Theory (if time permits)
- Modular arithmetic
- RSA cryptosystems

Topics CSCE 222

Probability

- Probability Axioms, events, random variable
- Independence, expectation, example distributions
- Birthday paradox
- Monte Carlo method

Graphs and Trees

(light coverage if time permits. Covered in Data Structures and Algorithms)

- Graph terminology
- Example of graph problems and algorithms:
  - graph coloring
  - TSP
  - shortest path

The END