Graph algorithms

Prof. Noah Snavely
CS1114
http://www.cs.cornell.edu/courses/cs1114

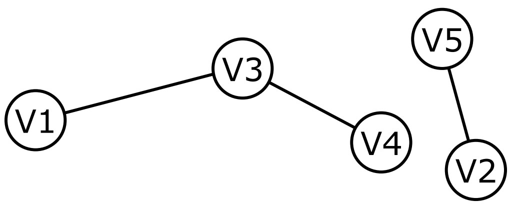
Cornell University
Computer Science

Administrivia

- Assignment 2 is out
  - Second part due this Friday by 5pm
  - Signup slots are up
- First prelim will be next week
  - Thursday, March 2, in class

What is a graph?

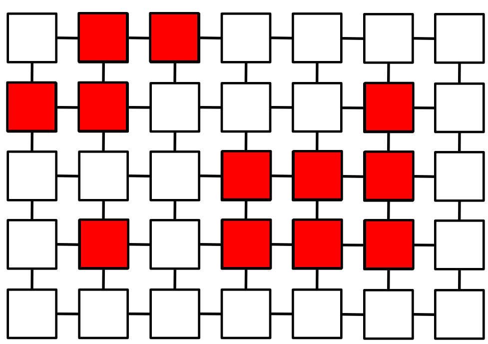
- Loosely speaking, a set of things that are paired up in some way
- Precisely, a set of vertices **V** and edges **E**
  - Vertices sometimes called nodes
  - An edge (or link) connects a pair of vertices



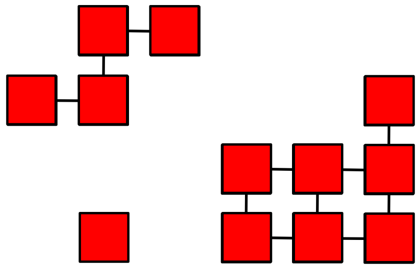
V = { V1, V2, V3, V4, V5 }

E = { (V1,V3), (V2,V5), (V3,V4) }

Images as graphs



Images as graphs

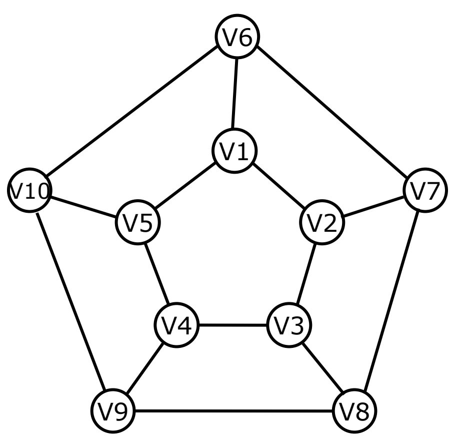


More graph problems

Hamiltonian & Eulerian cycles

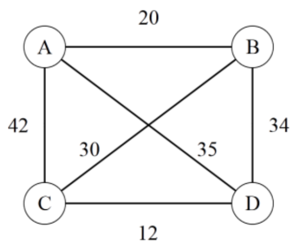
- Two questions that are useful for problems such as mailman delivery routes
- Hamiltonian cycle:
  – A cycle that visits each vertex exactly once (except the start and end)
- Eulerian cycle:
  – A cycle that uses each edge exactly once

Hamiltonian & Eulerian cycles



- ♦♦ Is it easier to tell if a graph has a Hamiltonian or Eulerian cycle?

Travelling Salesman Problem



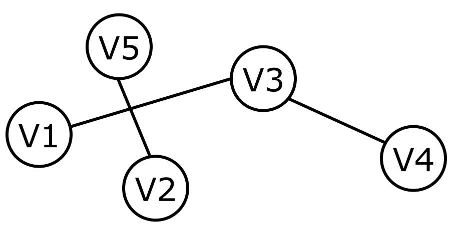
- For a **complete, weighted** graph
- Find the cycle that visits all vertices with the lowest total cost

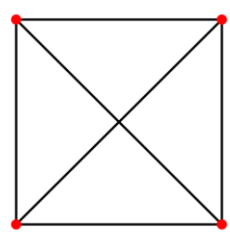
An exact solution for 15,112 German towns from TSPLIB was found in 2001 using the cutting-plane method proposed by George Dantzig, Ray Fulkerson, and Selmer M. Johnson in 1954, based on linear programming. The computations were performed on a network of 110 processors located at Rice University and Princeton University (see the Princeton external link). The total computation time was equivalent to 22.6 years on a single 500 MHz Alpha processor. In May 2004, the travelling salesman problem of visiting all 24,978 towns in Sweden was solved: a tour of length approximately 72,500 kilometers was found and it was proven that no shorter tour exists.[16]

In March 2005, the travelling salesman problem of visiting all 33,810 points in a circuit board was solved using *Concorde TSP Solver*: a tour of length 66,048,945 units was found and it was proven that no shorter tour exists. The computation took approximately 15.7 CPU-years (Cook et al. 2006). In April 2006 an instance with 85,900 points was solved using *Concorde TSP Solver*, taking over 136 CPU-years, see Applegate et al. (2006).

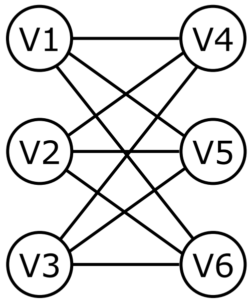
Planarity testing

- A graph is planar if you can draw it without the edges crossing
  - It’s OK to move the edges or vertices around, as long as edges connect the same vertices





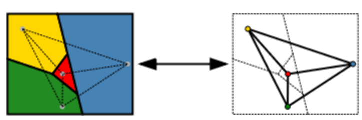
♦ Is this graph planar?



♦♦ *Can you prove it?*

Four-color theorem

- Any planar graph can be colored using no more than 4 colors



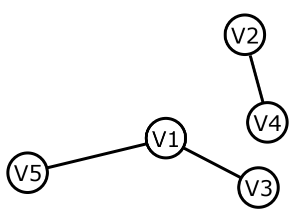
"Small world" phenomenon
(Six degrees of separation)

- How close together are nodes in a graph (e.g., what's the average number of hops connecting pairs of nodes?)

- Milgram's small world experiment:
  - Send postcard to random person A in Omaha; task is to get it to a random person B in Boston
  - If A knows B, send directly
  - Otherwise, A sends to someone A knows who is most likely to know B
  - People are separated by 5.5 links on average

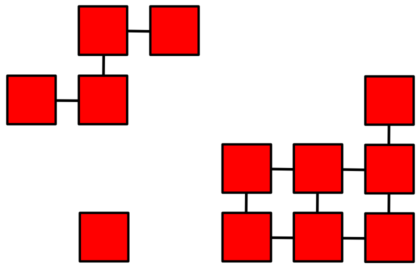
Connected components

- Even if all nodes are not connected, there will be subsets that are all connected
  - Connected components



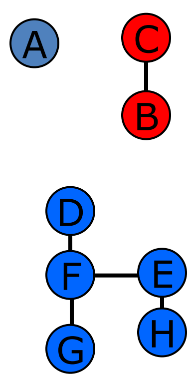
  - Component 1: { V1, V3, V5 }
  - Component 2: { V2, V4 }

Blobs are components!



Blobs are components!

A	0	0	0	0	0	0	0	B	0
0	0	0	0	0	0	0	0	C	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	D	0	0	0	0	0
0	0	0	E	F	G	0	0	0	0
0	0	0	H	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



Finding blobs

1. Pick a 1 to start with, where you don't know which blob it is in
   - When there aren't any, you're done
2. Give it a new blob color
3. Assign the same blob color to each pixel that is part of the same blob

Finding components

1. Pick a 1 to start with, where you don't know which component it is in
   - When there aren't any, you're done
2. Give it a new component color
3. Assign the same component color to each pixel that is part of the same component
   - Basic strategy: color any neighboring 1's, have them color their neighbors, and so on