Connected components and graph traversal

Prof. Noah Snavely

CS1114

http://cs1114.cs.cornell.edu

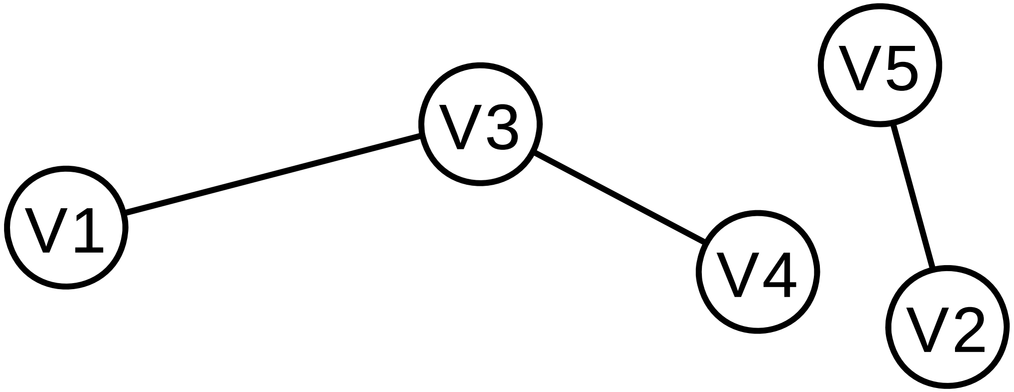
Cornell University
Computer Science

Administrivia

- Assignment 2 is out
  - First part due tomorrow by 5pm
  - Second part due next Friday by 5pm
- First prelim will be in two weeks
  - Thursday, February 26, in class

What is a graph?

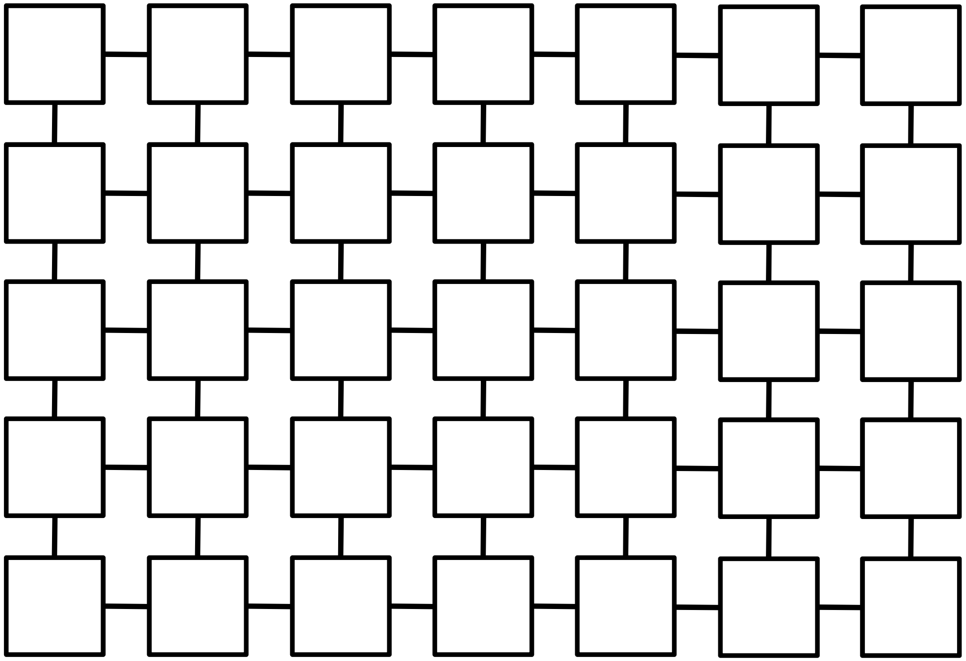
- Loosely speaking, a set of things that are paired up in some way
- Precisely, a set of vertices **V** and edges **E**
  - Vertices sometimes called nodes
  - An edge (or link) connects a pair of vertices



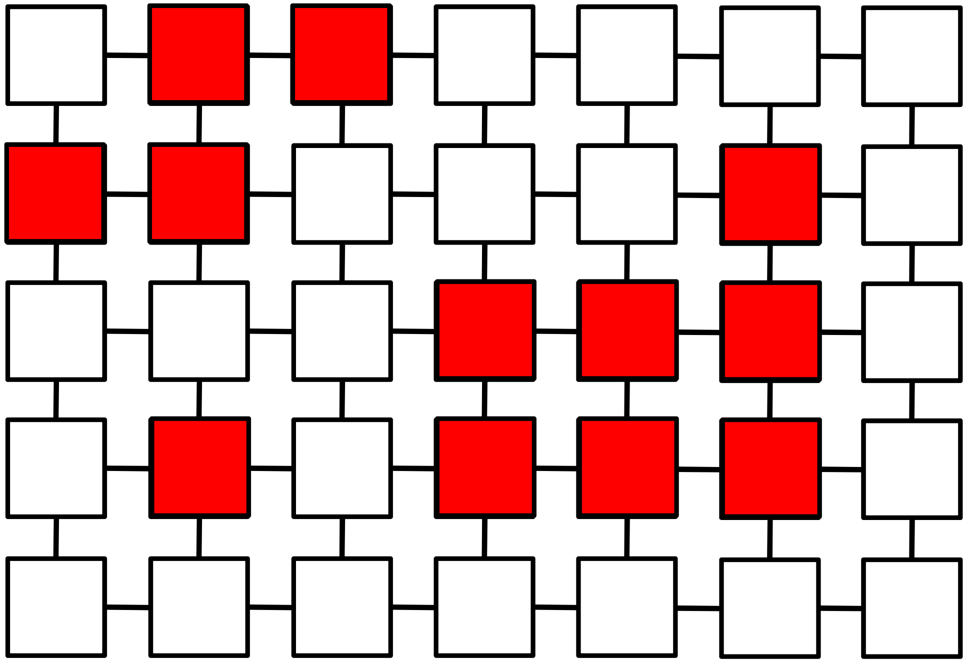
V = { V1, V2, V3, V4, V5 }

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

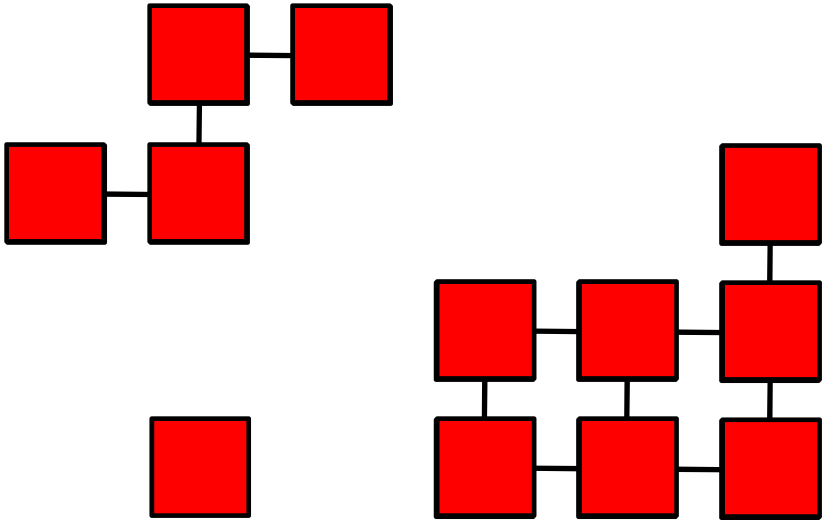
Images as graphs



Images as graphs



Images as graphs

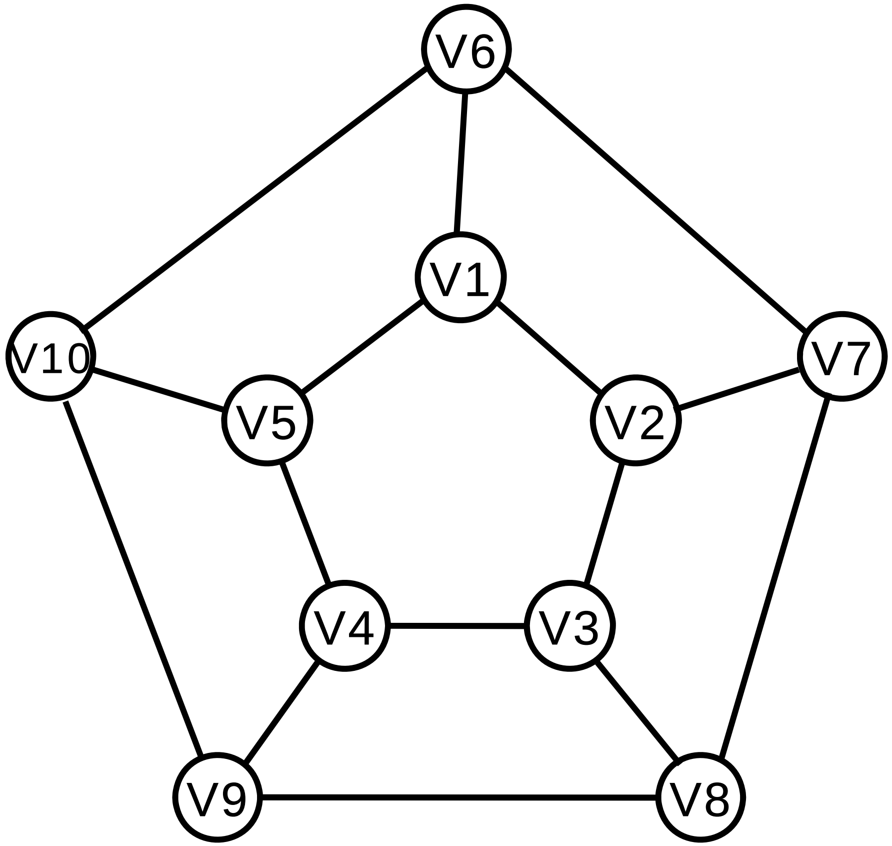


More graph problems

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

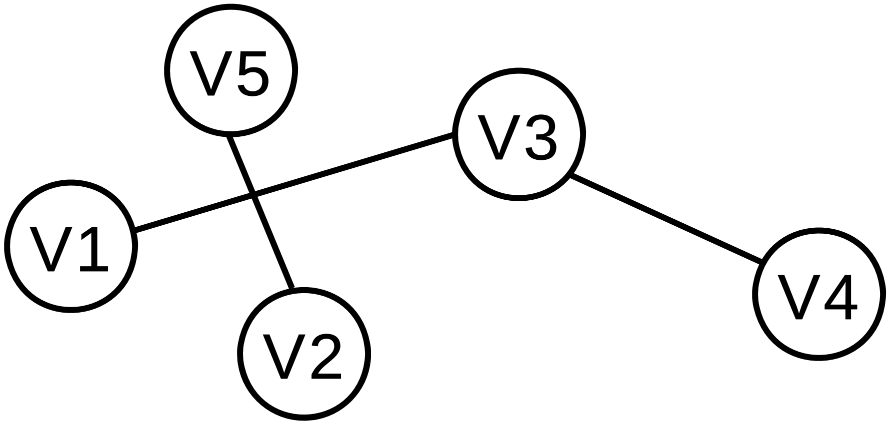
Hamiltonian & Eulerian cycles

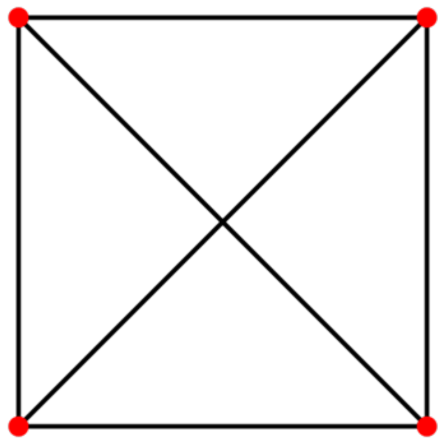


- ♦♦ Is it easier to tell if a graph has a Hamiltonian or Eulerian cycle?

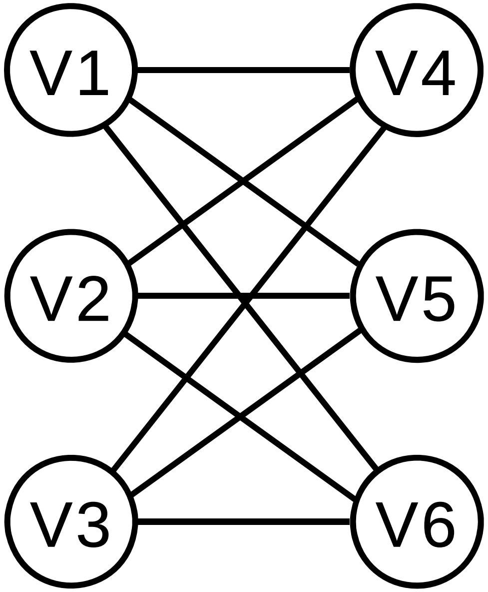
Planarity testing

- 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





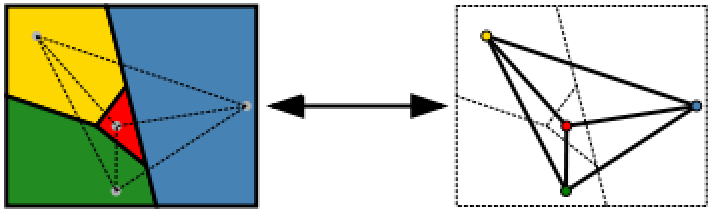
♦ Is this graph planar?

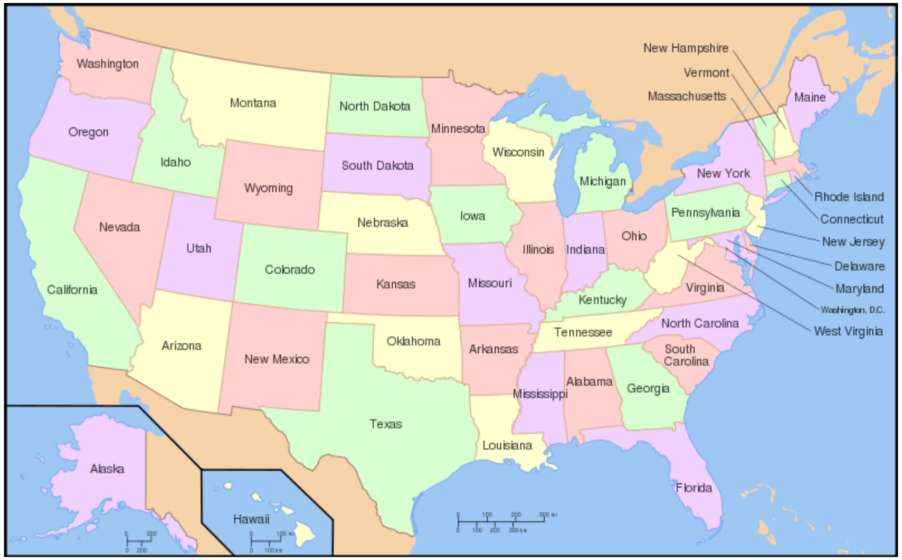


♦♦ *Can you prove it?*

Four-color theorem

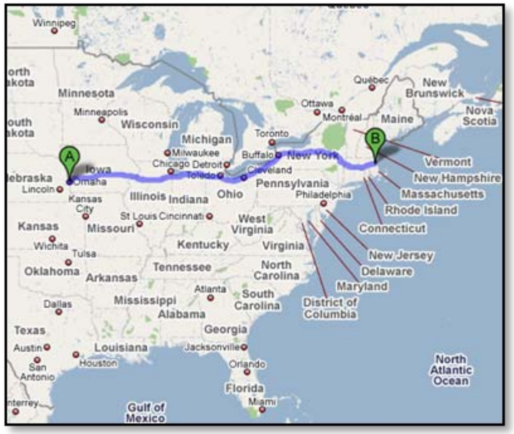
- Any planar graph can be colored using no more than 4 colors





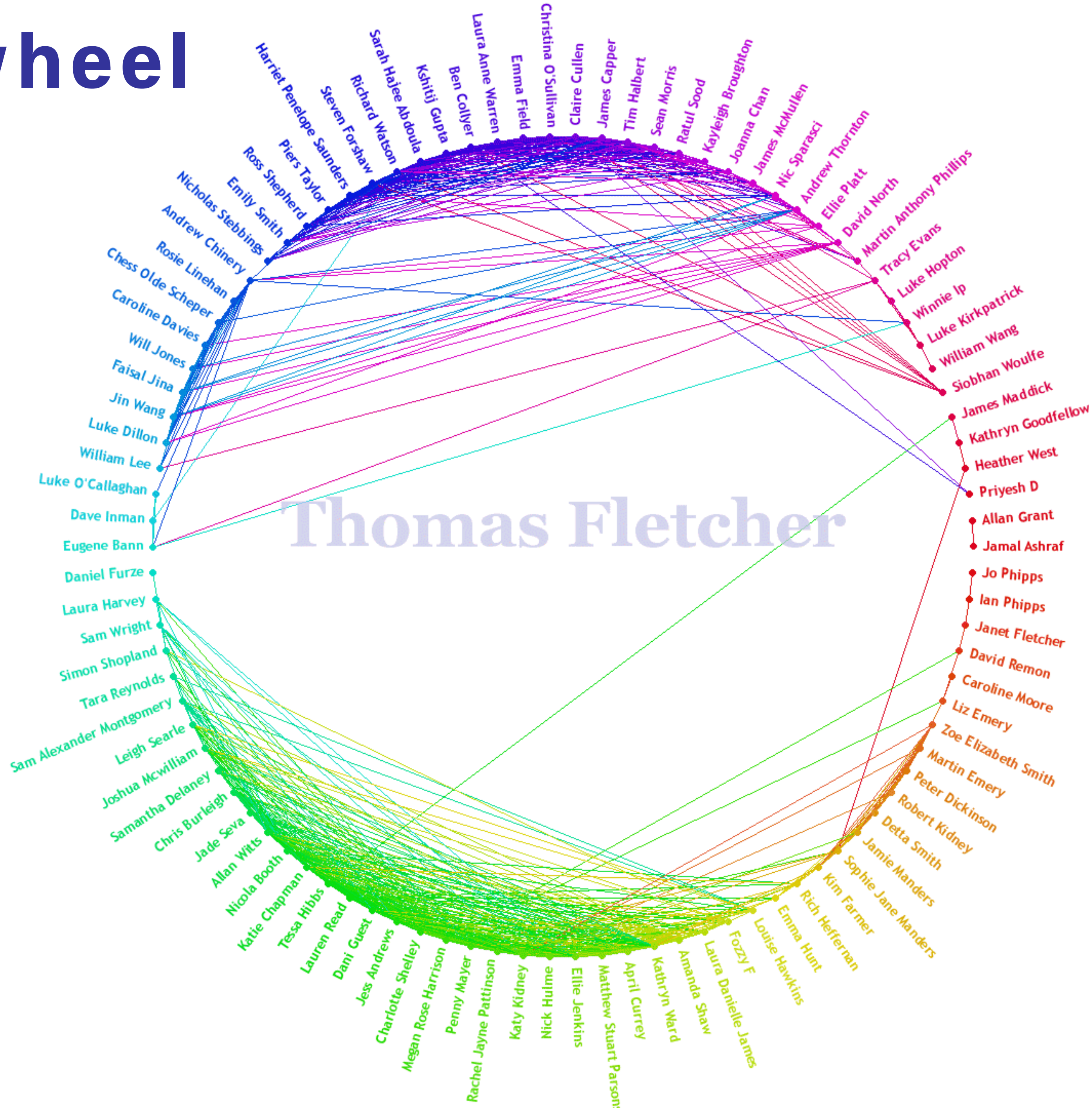
“Small world” phenomenon
(Six degrees of separation)

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

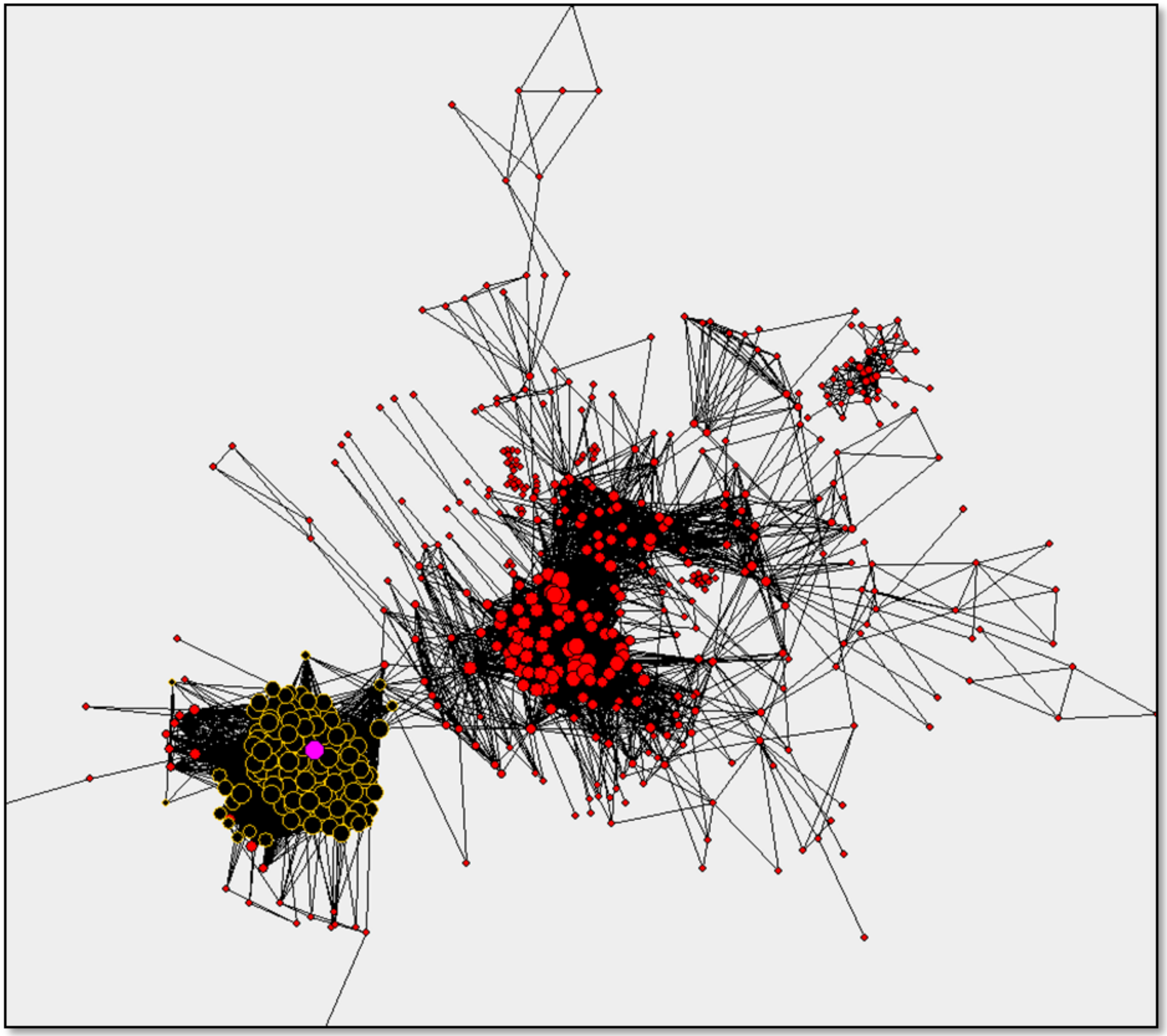


- Milgram’s small world experiment:
  - Send postcard to random person A in Omaha; task is to get it to a random person B in Boston
  - If A knows B, send directly
  - Otherwise, A sends to someone A knows who is most likely to know B
  - People are separated by 5.5 links on average

Friend wheel



Another graph

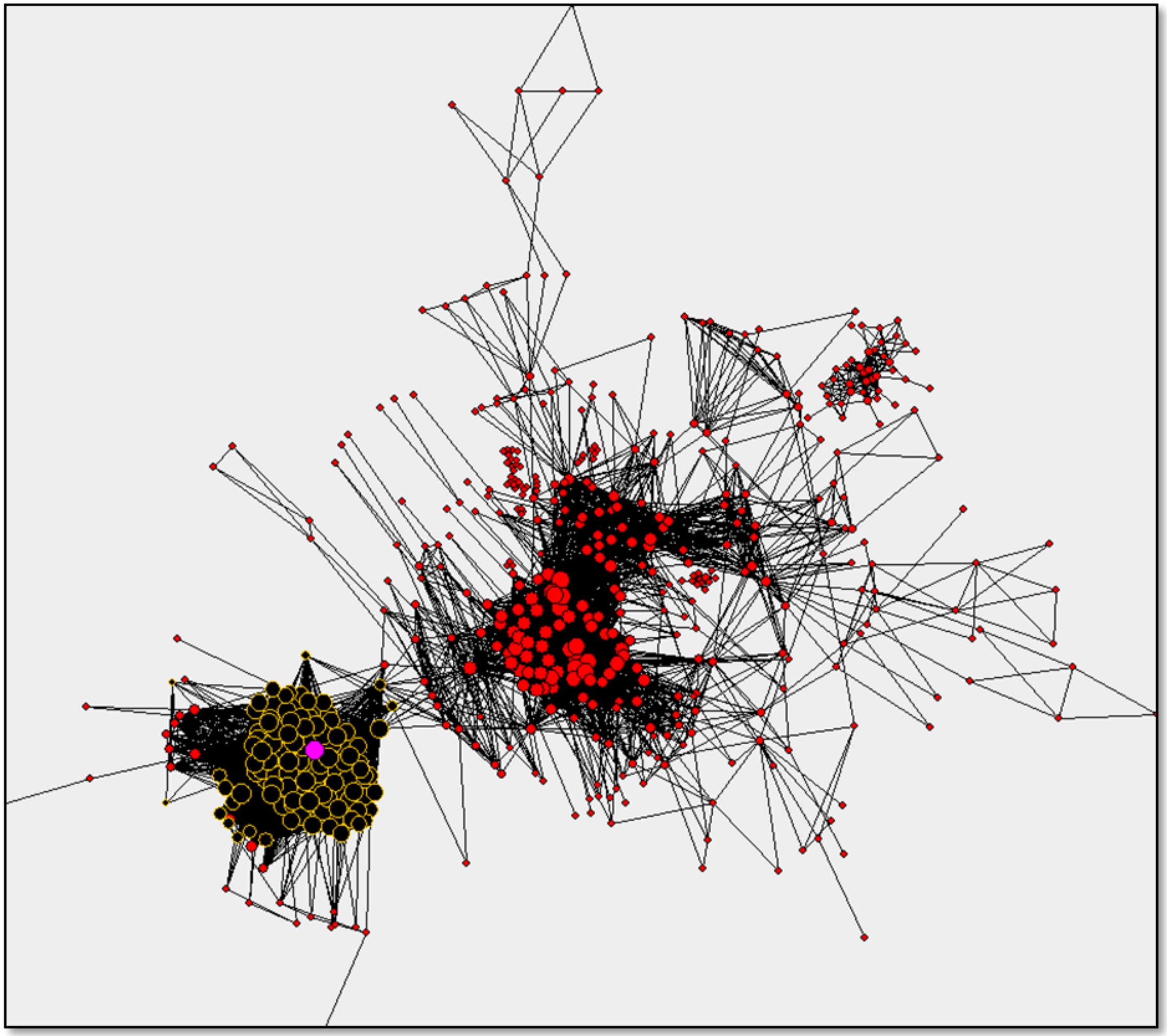


Graph of Flickr images

Flickr images of the Pantheon, Rome (built 126 AD)

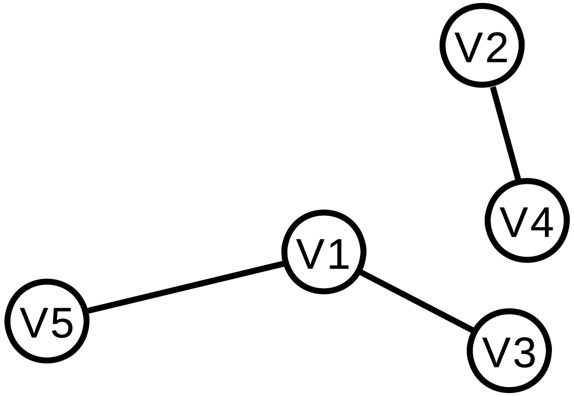
Images are matched using visual features

Image graph of the Pantheon



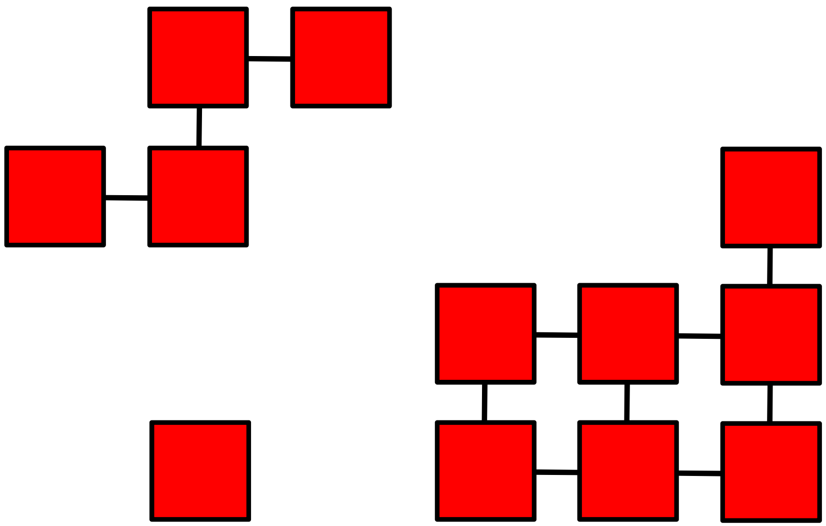
Connected components

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



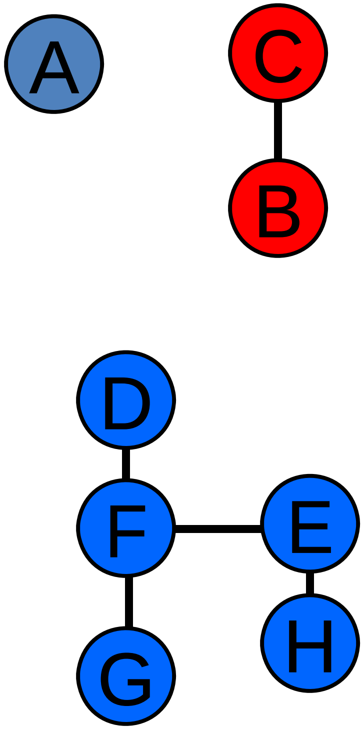
  - Component 1: { V1, V3, V5 }
  - Component 2: { V2, V4 }

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

Coloring a component

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	1	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Coloring a component

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	1	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Coloring a component

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	1	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Coloring a component

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	1	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Strategy

- For each vertex we visit, we color its neighbors and remember that we need to visit them at some point
  - Need to keep track of the vertices we still need to visit in a todo list
  - After we visit a vertex, we'll pick one of the vertices in the todo list to visit next

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	**A**	0	0	0	0	0
0	0	0	B	C	D	0	0	0	0
0	0	0	E	F	G	0	0	0	0
0	0	0	H	I	J	0	0	0	0
0	0	0	K	L	M	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Current node: A
Todo List: [ ]

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	**A**	0	0	0	0	0
0	0	0	B	C	D	0	0	0	0
0	0	0	E	F	G	0	0	0	0
0	0	0	H	I	J	0	0	0	0
0	0	0	K	L	M	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Current node: A
Todo List: [C] ← *Done with A, choose next from Todo List*

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	A	0	0	0	0	0
0	0	0	B	**C**	D	0	0	0	0
0	0	0	E	F	G	0	0	0	0
0	0	0	H	I	J	0	0	0	0
0	0	0	K	L	M	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Current node: C
Todo List: [ ]

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	A	0	0	0	0	0
0	0	0	B	**C**	D	0	0	0	0
0	0	0	E	F	G	0	0	0	0
0	0	0	H	I	J	0	0	0	0
0	0	0	K	L	M	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Current node: C
Todo List: [B, F, D]

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	A	0	0	0	0	0
0	0	0	**B**	C	D	0	0	0	0
0	0	0	E	F	G	0	0	0	0
0	0	0	H	I	J	0	0	0	0
0	0	0	K	L	M	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Current node: B
Todo List: [F, D]

1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	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	A	0	0	0	0	0
0	0	0	**B**	C	D	0	0	0	0
0	0	0	E	F	G	0	0	0	0
0	0	0	H	I	J	0	0	0	0
0	0	0	K	L	M	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Current node: B
Todo List: [F, D, E]

Graph traversal

- Select an uncolored vertex, add it to the todo list
- While the todo list is not empty
  - Remove a vertex V from the todo list
  - Add the **uncolored** neighbors of V to the todo list and color them

Stacks and Queues

- One way to implement the todo list is as a *stack*
  - LIFO: Last In First Out
  - Think of a pile of trays in a cafeteria
    - Trays at the bottom can stay there a while…

- The alternative is a *queue*
  - FIFO: First In First Out
  - Think of a line of (well-mannered) people
  - First come, first served

Next time

- More on stacks and queues