Graph Traversal

Prof. Ramin Zabih

http://cs100r.cs.cornell.edu

Cornell University
Computer Science

Administrivia

- Assignment 2 is due Friday
  - Working Roomba's are on the way (?)
- Quiz 3 on Tuesday 9/25
  - Coverage through today

Hamiltonian & Eulerian cycles

- Two questions that are useful for problems such as mailman delivery routes
- Is there a path that goes through each vertex exactly once?
  - Hamiltonian cycle
- Is there a path that uses each edge exactly once?
  - Eulerian cycle
- ♦♦Which is easier to compute?

Colorings and cliques

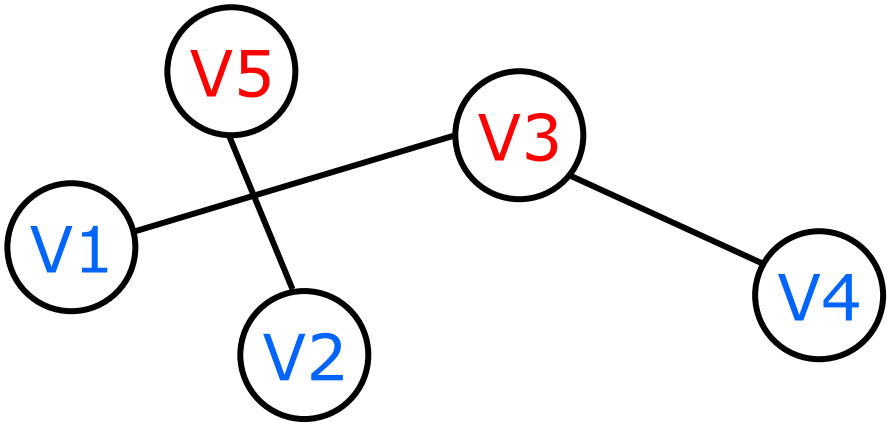
- A clique is a set of vertices which are all connected to each other
- If there is a clique with 4 vertices, what can you say about coloring the graph?
  - Need at least 4 colors
- With enough colors, can color any graph
  - Each vertex gets a different color

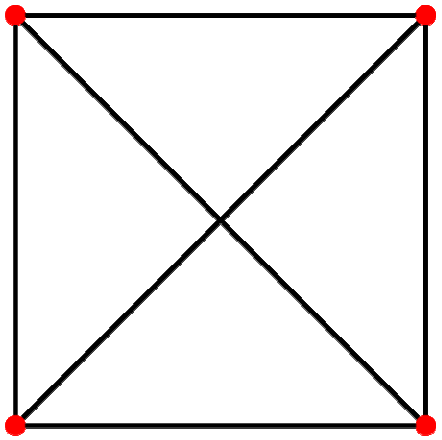
Independent set

- An independent set is a set of vertices where no pair is connected by an edge
  - In a coloring, each color is an independent set
- How can you schedule your classes via independent set?
  - Nodes are classes, edges connect classes that conflict with each other
  - "I can take M classes" iff "the graph has an independent set of size M"
    - This implies you can take <M classes also

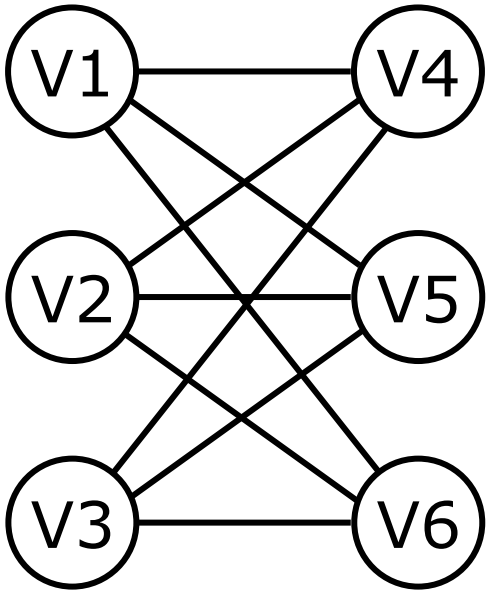
Planarity testing

- 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?



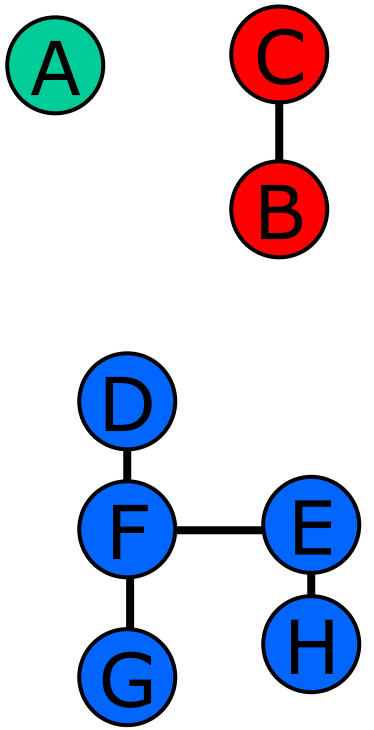
♦♦ *Can you prove it?*

Four-color theorem

- Any planar graph can be colored using no more than 4 colors
  - I.e., consider a map, where you want adjacent countries to be different colors
- Heavily computer-aided proof

Blob = connected component

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



Where 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	1	0	0	0	0	0
0	0	0	X	C	1	0	0	0	0
0	0	0	B	A	D	0	0	0	0
0	0	0	Y	E	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

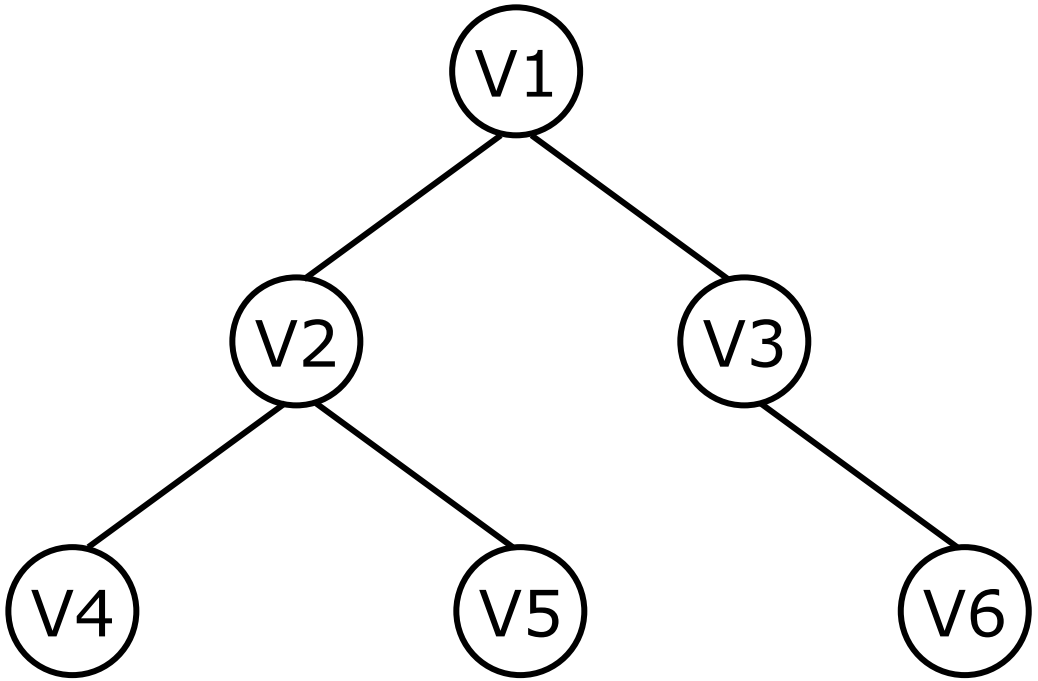
- We need to keep track of the vertices we need to "visit" (possibly label with our blob number)
  - Although, by the time we get there, they may already have been labeled
- Visit A, and label it
  - Need to visit its neighbors: B, C, D, E
  - Pick one to visit next, keep track of the rest
    - Visit B, and label it
      - Need to visit its neighbors X, Y
      - Pick one to visit next, keep track of the rest

Graph traversal

- Visit every vertex in our current component, giving it the label L
- While there is a vertex V that we have not yet given a label
  - Add V to our "to-do" collection C
  - While C is not empty
    - Take a vertex V' out of C
    - Give it label L, if it doesn't have this label yet
    - Add the **unlabeled** neighbors of V' to C
      - Why is this important?

Graphs and trees

- A graph is a tree if it has no cycles
  - We are only considering undirected graphs



- Start at V1
  - Put V1 in C
  - Take, label V1
  - Put V2, V3 in C
  - Take, label V2
  - Put V4, V5 in C
  - Now what??

Stacks and Queues

- The obvious ways to implement our collection C is 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…
- Other alternative is a queue
  - FIFO, First In First Out
  - Think of a line of (well-mannered) people
  - First come, first served

Two basic traversal orders

- Depth-first
  - Stack
- Breadth-first
  - Queue

Animations taken from:
http://www.rci.rutgers.edu/~cfs/472_html/AI_SEARCH/ExhaustiveSearch.html