Today’s announcements:

Code challenge #2, 12/10, 9p, Siebel 0224. (next week)
Please check your grade: chara.cs.illinois.edu
Final exam: 12/15, 7-10p, locations TBA
email c-heeren@illinois.edu asap w conflict in subject line

How do we get from here to there?
Need:

1. Common Vocabulary
2. Graph implementation
3. Traversal
Graphs – traversal

Objective: Visit every vertex and every edge, in the graph,

Purpose: We can search for interesting substructures in the graph,

Contrast graph traversal to BST traversal:

- Ordered
- Obvious start

•
Graphs: BFS example
Graphs: Traversal – BFS

Visits every vertex and classifies each edge as either “discovery” or “cross”

Algorithm BFS(G)
Input: graph G
Output: labeling of the edges of G as discovery edges and back edges

For all u in G.vertices()
setLabel(u, UNEXPLORED)

For all e in G.edges()
setLabel(e, UNEXPLORED)

For all v in G.vertices()
if getLabel(v) = UNEXPLORED
BFS(G,v)

Algorithm BFS(G,v)
Input: graph G and start vertex v
Output: labeling of the edges of G in the connected component of v as discovery edges and cross edges

queue q;
setLabel(v, VISITED)
q.enqueue(v);

While !(q.isEmpty)
q.dequeue(v)

For all w in G.adjacentVertices(v)
if getLabel(w) = UNEXPLORED
setLabel((v,w), DISCOVERY)
setLabel(w, VISITED)
q.enqueue(w)
else if getLabel((v,w)) = UNEXPLORED
setLabel((v,w), CROSS)
Graphs: BFS example

While loop

For loop

TOTAL RUNNING TIME:
Graphs: BFS observations

BFS
Graphs: Traversal - DFS

Ariadne, Theseus, and the Minotaur

http://www.cs.duke.edu/csed/jawaa2/examples/DFS.html
http://www.student.seas.gwu.edu/~idsv/idsv.html
http://www.youtube.com/watch?v=8qrZ1clEp-Y
Algorithm DFS(G)

Input: graph G

Output: labeling of the edges of G as discovery edges and back edges

For all u in G.vertices()
    setLabel(u, UNVISITED)

For all e in G.edges()
    setLabel(e, UNEXPLORED)

For all v in G.vertices()
    if getLabel(v) = UNVISITED
        DFS(G,v)

Algorithm DFS(G,v)

Input: graph G and start vertex v

Output: labeling of the edges of G in the connected component of v as discovery edges and back edges

setLabel(v, VISITED)

For all w in G.adjacentVertices(v)
    if getLabel(w) = UNVISITED
        setLabel((v,w), DISCOVERY)
        DFS(G,w)
    else if getLabel((v,w)) = UNEXPLORED
        setLabel(e, BACK)

DFS: “visits” each vertex classifies each edge as either “discovery” or “back”
Graphs: DFS example
Graphs: DFS Analysis

setting/getting labels
every vertex labeled twice
every edge is labeled twice

querying vertices
each vertex
total over algorithm
querying edges

TOTAL RUNNING TIME:
DFS: How can we count the number of connected components in the graph?

How can we determine if a graph has a cycle?

Algorithm DFS(G)

Input: graph G
Output: labeling of the edges of G as discovery edges and back edges

For all u in G.vertices()
setLabel(u, UNVISITED)
For all e in G.edges()
setLabel(e, UNEXPLORED)
For all v in G.vertices()
if getLabel(v) = UNVISITED
DFS(G,v)

Algorithm DFS(G,v)

Input: graph G and start vertex v
Output: labeling of the edges of G in the connected component of v as discovery edges and back edges

setLabel(v, VISITED)
For all w in G.adjacentVertices(v)
if getLabel(w) = UNVISITED
setLabel((v,w), DISCOVERY)
DFS(G,w)
else if getLabel((v,w)) = UNEXPLORED
setLabel(e, BACK)