More Graph Algorithms

Announcements

- Final HW up - due Monday

- No class tomorrow (also no office hours)

  Office hours Friday 9-10am

- Set review session: Friday at 2pm
Algorithms on Graphs

Basic Question: Given 2 vertices, are they connected?

How to solve?

Search strategy
Recursive DFS (u):

If u is unmarked:
  mark u
  for each edge \( (u, v) \in E \)
  Recursive DFS(v)

To check if s and t are connected,
call DFS(s).

At end, if t is marked, return true
DFS (1)

DFS "tree"
Another version of DFS

Iterative DFS$(u)$:

1. create empty stack $S$
2. $S$.push($u$)

while $S$ is not empty:

1. $v \leftarrow S$.pop
2. if $v$ is not marked
   1. mark $(v)$
   2. for each edge $vw$
      1. $S$.push($w$)
Iterative DFS (1):

Stack:
1 7 8

Diagram of a graph with nodes and edges.
BFS: Breadth-first search

Instead of a stack, could push all the neighbors on a queue!
So from S all of S's neighbors will connect to it.
Iterative BFS \((u)\)

\[\text{Q. push}(u)\]

while Q is not empty:
\[v \leftarrow \text{Q. pop}\]
if \(v\) is not marked
\[\text{mark}(v)\]
for each edge \(vw\)
\[S. push(w)\]
BFS (1):

Queue: 1 2 3 4 5 6 7 8 1 3 8

BFS "tree":

1
  / \  \
2   3
  \   \
   \    \
    \   \
     \  \
      4 5
       \  \
        \  \
         6

BFS versus DFS

- Both can tell if 2 vertices are connected.
- Both can be used to detect cycles.
  How?
  Any "extra" edge forms a cycle.
- Difference is structure of trees.
Runtimes:

\[ m = |E| \quad n = |V| \]

- each push/pop is \( O(1) \)
- must visit each node \( \Rightarrow O(n) \)

Each node \( v \) is pushed \( d(v) \) times (and popped)

\[ \sum d(v) = 2m \] (degree sum formula)

\[ \Rightarrow O(m) \]

\[ \Rightarrow O(m+n) \quad [\text{in general, } O(m)] \]
Other graph algorithms

- BFS returns a “short” S-t path, in some sense.

But won’t work if graph has weights on the edges.

Why?

Which s-t path will be in BFS tree?
Shortest path trees

Given a weighted graph, find shortest path from s to t.

Uses?

- Road networks
- All other vertices

Find all shortest paths from s to all other vertices
Algorithms to solve this actually solve a more general problem: find the shortest path from \( s \) to every other vertex. Called the shortest path tree rooted at \( s \).

Can be computed in polynomial time. Key: use a heap.
Another question:
Given $G$, find a tree containing every vertex with minimum total weight.

Uses? - Network planning
This is called the **minimum spanning tree** of 6.

Note: Not the same as shortest path tree!
Review Session:

Thursday
9 am
10 am
11 am

Friday
9 am
10 am
11 am
2 pm
3 pm
4 pm