Graphs: Connectivity and Searching

CS 62 - Spring 2016
Michael Bannister

This Week

Weekly Assignment

- 20 Questions Animal Game
- Binary trees
- Arbitrary length strings is **extra credit** and easy!

Weekly Lab

- Short lecture on makefiles
- Implement some graph algorithms

Reading: Bailey Chapter 16

Depth First Search

- Similar to recursive tree traversals
  - Need to mark visited nodes to avoid cycles
- Outline:
  - Mark start vertex
  - Recursively search all unmarked neighbors
  - Record “parent vertex”
- Implementation details:
  - How do we mark vertices?
  - How do we record parents?

Breadth-First Search (BFS)

- Similar to level order traversal of a tree
  - Search all nodes one hop from the source, then two hops, then three hops, etc…
- Uses a queue instead of the recursion call stack

BFS Outline

- Add start vertex to queue and mark the start
- while the queue is not empty:
  - remove the vertex at the start of the queue
  - for each of its unmarked neighbors:
    - add them to the queue and mark them

BFS C Code

(See Example Code)

Restarting

- BFS and DFS will only explore the connected component in which they are started!
- To explore the entire graph you need to loop through all of the vertices in the graph and run DFS/BF on all unmarked vertices
- Total runtime is: O(n+m)

Connectivity

Question: How do we test if a graph is connected?

Answer: Run DFS or BFS, without restarting, from an arbitrary vertex and see if all vertices are marked!

Runtime: O(n+m)

Strong Connectivity

Question: How do we test if a graph is strongly connected?

Answer: (1) Run a search, without restarting from an arbitrary vertex *s* and see if all vertices get marked. (2) Reverse all of the edges in the graph. (3) Run a second search, without restarting, from *s* and see if all vertices are marked. The graph is strongly connected iff both searches mark all vertices

Runtime: O(n+m)