Graph Traversals

• If linear, then “efficient”
  • similar to tree, just count “visit” not recursive or iterative function returns, etc.
Why Traverse Undirected Graph?

• to find path connecting vertices, if it exists
• find shortest path between pair of vertices
• determine if graph connected
• if connected, find spanning tree
• find connected components
• find any cycles
Why Traverse Directed Graph?

• find directed path between two vertices, if it exists
• find all vertices reachable from a particular vertex
• determining if directed graph is acyclic
• determining if directed graph is strongly connected
Depth-First Search (DFS)

• Starting at a vertex, $v$
  • For each outgoing edge $e$, $(v, u)$ of $v$
    • If $u$ has not been visited, mark $u$ as visited (via $e$).
    • Recurse on $u$

• DFS gives a depth-first search tree
• Different starting points, different trees
• For this class – always use lexicographic order
• (vocabulary in textbook/webnotes)
DFS example

• (in webnotes and on board)
  • Visiting all nodes in alphabetical order
  • must unwind all the way back to start vertex to be certain its done!
DFS properties

• If $G$ is an undirected graph and we did a DFS starting at $s$
  • the DFS visited all the connected components of $s$
  • the discovery edges form a spanning tree of $s$’s connected components

• $O(n + e)$
  • (sketch of proof in textbook/webnotes)
  • depends on underlying implementation

• dfs.py
More DFS Properties

- Can use to find transitive closure in $O(n + e)$
- Textbook code can construct an actual path by backtracking after DFS tree is created
- Can test connectedness of undirected graphs
- Can test strongly connectedness of digraphs
  - Requires 2 DFSs
- Can find connected components of undirected graph
- Can find cycles by checking back edges
Breadth-First Search (BFS)

• Iterative, not recursive
• Pick a starting vertex, visit every node connected, make that a “level”.
• Then visit every node connected to that level, making a new level, etc.
BFS example

• (in webnotes/on board)
  • Visit nodes in alphabetic order!
BFS properties

• If G is either an directed or undirected graph which has a BFS starting at s, then:
  • The traversal visits all of s's reachable vertices.
  • The path in the BFS tree from s to any vertex in level i has i edges, and any other path has at least that many.
  • For an edge that is not in the BFS tree, the levels of the opposite vertices differ by at most one.

• O(n + e)
  • similar proof for DFS