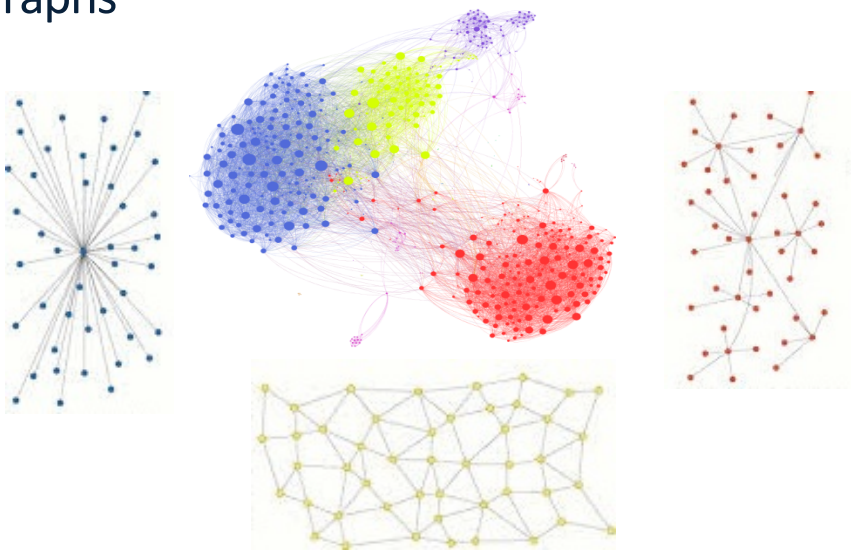
LECTURE 38: GRAPHS

Today

- Reading
  - Weiss Chapter 16
- Objectives
  - BFS, DFS
  - Dijkstra's Algorithm

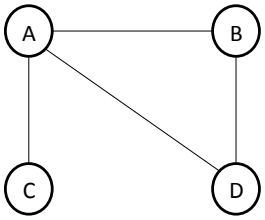
Graphs



Adjacency Matrix

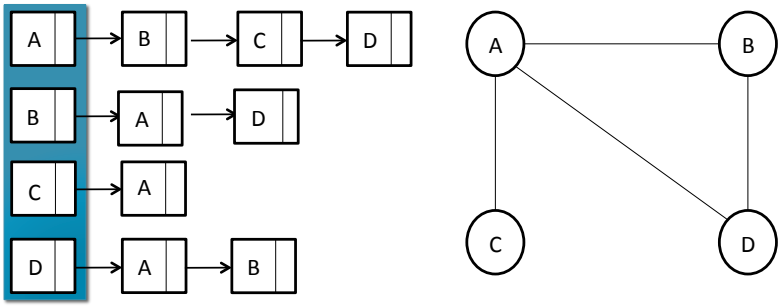
- Store a |V|-by-|V| boolean matrix
  - Entry (i,j) is 1 if there is an edge from vertex i to vertex j
  - Symmetric if undirected
  - Space? Time to lookup edge? Time to iterate over incident edges?

	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	0
D	1	1	0	0



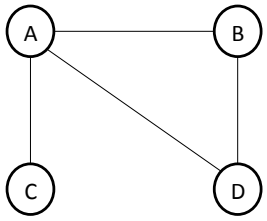
Adjacency List

- Store a list of linked lists
  - Use map from vertex labels to lists
  - Space? Time to lookup edge? Time to iterate over incident edges?



Breadth-first Search

- Equivalent to a level-order traversal of a tree
- Uses a queue data structure
- Basic algorithm:
  - Enqueue the start node
  - While the queue is not empty:
    - Dequeue a node
    - Check if node previously visited
    - If not, mark as visited and enqueue all children

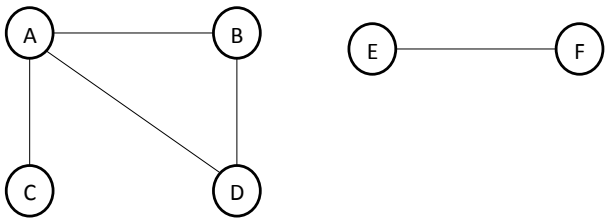


Breadth-first Search

- If graph has multiple connected components
  - Wrap BFS inside a for-loop that iterates through all nodes
- See *bfs_dfs_demo.cpp*
  - Uses a `typedef` (allows you to rename a type)
  - Better to use `map<string, vector<string>>` instead of `vector<pair<string, vector<string>>`

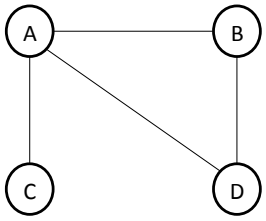
Depth-first search

- Equivalent to a pre-order traversal of a tree
- Use same algorithm as BFS but replace queue with stack/recursion



Detecting Cycles

- Can use DFS to see if we loop back
- A cycle exists if,
  - A node in adjacency list has already been visited but it is not the node that added us to the stack
  - i.e. ancestor (not parent) in search tree already visited
  - Works for undirected graphs



Single Source Shortest Path

- Starting at node s, find shortest path to all other nodes
- If edges have no weight then can use BFS
  - Shortest path is defined to be the path with fewest edges
- If edges have (non-negative) weights, use Dijkstra's Algorithm
  - Dijkstra's Algorithm is BFS with a priority queue
  - The priority is the distance from the start node to current node
  - Keep track of parent node (i.e. preceding node in the path)

Single Source Shortest Path