Lecture 35: Graphs II

CS 62

Fall 2017

Kim Bruce & Alexandra Papoutsaki

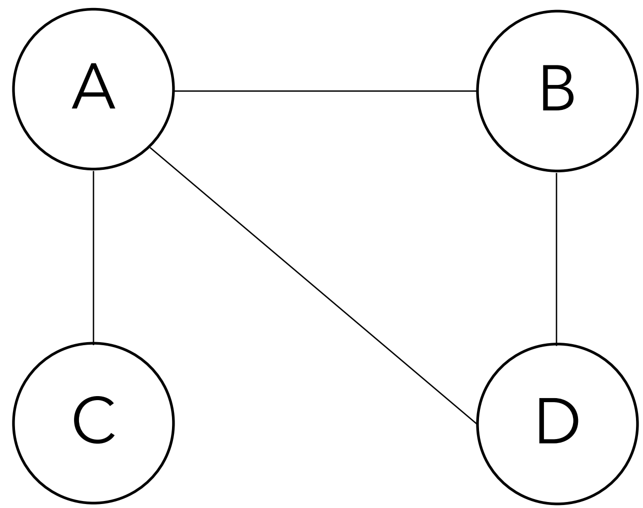
Number of Edges

- If $|V| = n$, then:
- minimum number: 0
  - A graph can have only nodes
- For simple directed graphs, maximum number: $n(n-1)$
- For simple undirected graphs, maximum number: $\frac{n(n-1)}{2}$

- Dense graphs → #edges close to maximum
- Sparse graphs → #edges close to n

Graph Representations

- Adjacency Matrix
- Adjacency List

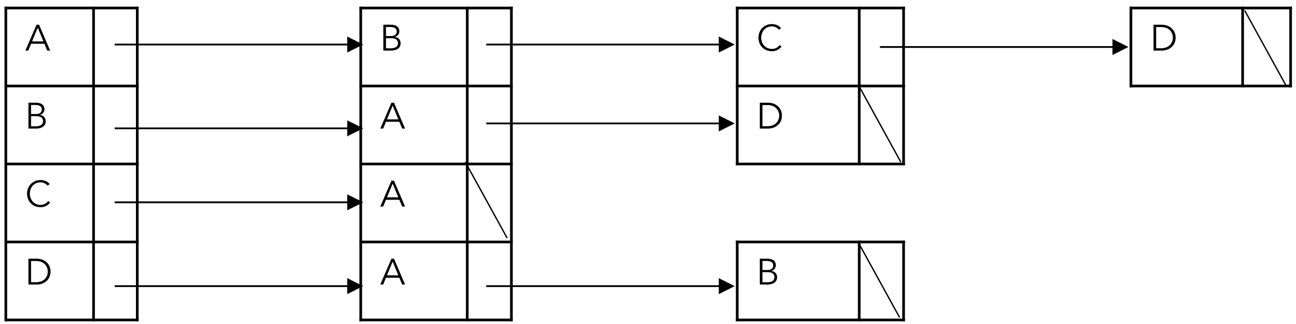


Adjacency Matrix

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

- Good for dense graphs.
- Constant time for lookup for edges.
- Symmetric if undirected.
- Can hold weights.

Adjacency Lists



- Good for sparse graphs, saves space.
- Linear time lookup for edges.

Spanning Trees

- Tree: connected undirected graph with no cycles
- Spanning tree of G: includes every vertex of G and is a subgraph of G (every edge belongs to G)
- Can have properties like minimum-cost
- Can be constructed by search algorithms

Depth-First Search

- Explore the graph without revisiting nodes
- Depth-first means go until you hit a dead end, then back up to branch out

- Algorithm:
  1. Mark current vertex
  2. Recursively explore all unmarked neighbors using a stack
  (optionally) record where you came from

Breadth-First Search

- Replace stack with queue
- Now we explore in order of distance from start

- Algorithm:
  1. Mark start vertex
  2. Add all unmarked neighbors to queue and mark them
  3. Repeat step 2 with next from queue until it's empty

DFS/BFS traversal

- Can be performed in $O(n + m)$, where $n = |V|, m = |E|$
- Can :
  - Test if G is connected
  - Compute a spanning tree of G, if G is connected
  - Find a path between two vertices, if it exits
  - Compute the connected components of G
    (needs to loop over all vertices and run DFS/BFS again)

Testing connectivity

- For an undirected graph:
  - Run DFS/BFS from any vertex without restarting and see if all vertic es are marked

- For strong connectivity on a directed graph:
  - 1. Run DFS/BFS without restarting from a specific vertex
    2. Run it again from that vertex after reversing all the edges
    It's strongly connected iff both runs mark all vertices