CS5800 – ALGORITHMS

MODULE 5. DATA STRUCTURES & GRAPHS

Lesson 1: Adjacency Matrices & Lists

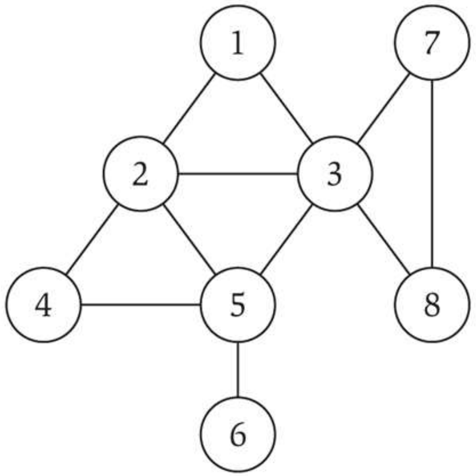
Ravi Sundaram

Topics

- Review of basic graph terminology
- Recap of graph representations
- Summary

Undirected & Directed Graphs

- Undirected graph. G = (V, E)
  - V = nodes.
  - E = edges between unordered pairs of nodes.
  - Captures pairwise relationship between objects.
  - Graph size parameters: n = |V|, m = |E|.



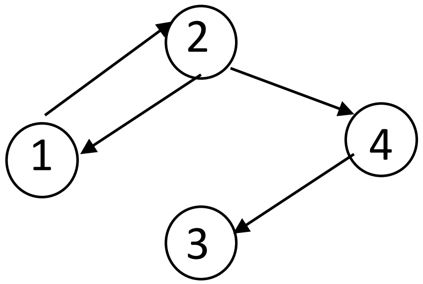
V = { 1, 2, 3, 4, 5, 6, 7, 8 }

E = { 1-2, 1-3, 2-3, 2-4, 2-5, 3-5, 3-7, 3-8, 4-5, 5-6, 7-8 }

n = 8

m = 11

- Directed graph.
  - E = arcs or directed edges, i.e., ordered pairs of nodes.



V = { 1, 2, 3, 4 }

E = { 1-2, 2-1, 2-4, 4-3

n = 4

m = 4

Graphs

- In an *undirected* graph, the edges are two-element subsets of V: {u, v}. Draw edges with lines.
- In a *directed* graph, the edges are ordered pairs of vertices (u, v). Draw edges with arrows.
- You can generally assume no self-loops.
- Usually, you don’t have multiple edges connecting the same two vertices — i.e., (u, v) is in the graph or not, but it isn’t there twice (E is a set). If a situation warrants multiedges then the graph is called a multigraph

Applications

- Computer networks
  - Shortest paths graph algorithm used for routing packets

- Pathfinding
  - In both video games and real-world applications, maps are represented as graphs

- The World Wide Web
  - Search engines want to find and penalize “link farms” that all link to each other … a property of the link graph

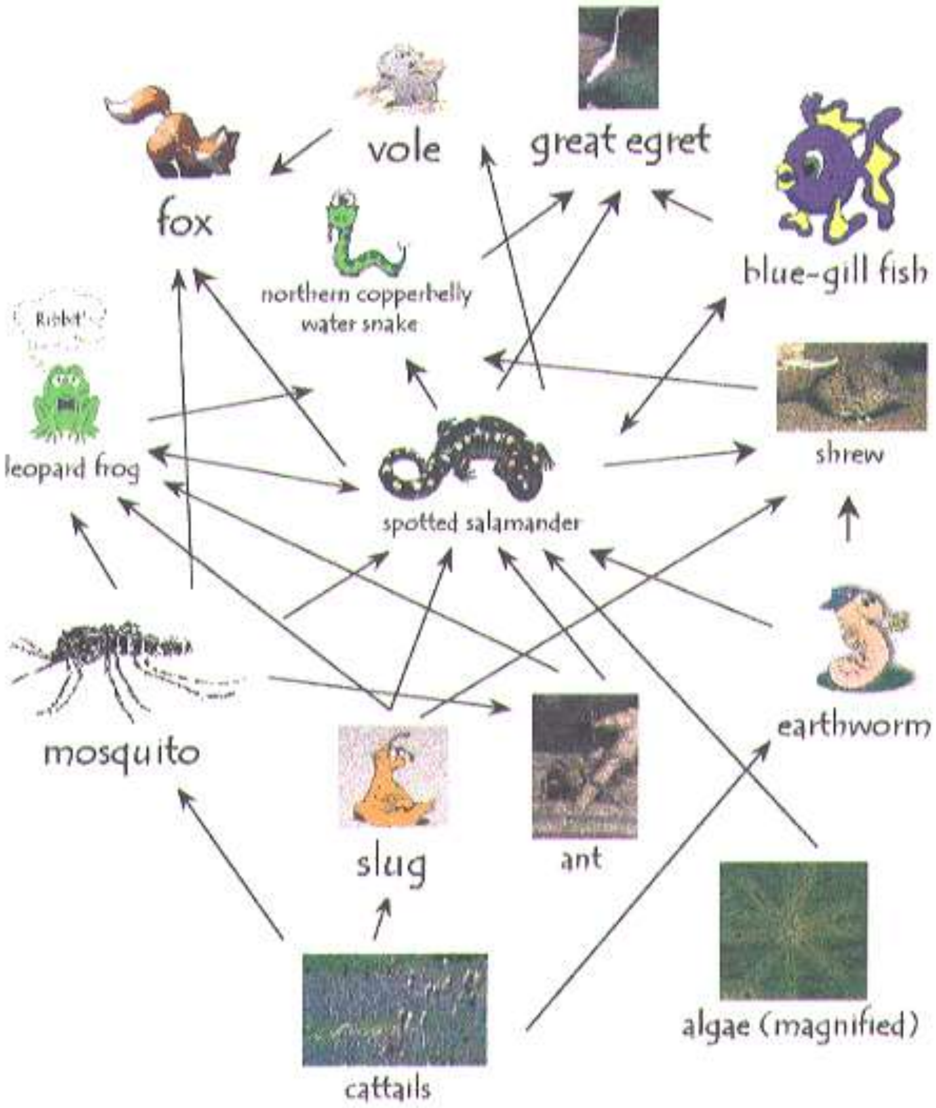
- Basically, any problem that involves reasoning about connections!

Graph Applications and models

Graph	*Nodes*	*Edges*
transportation	street intersections	highways
communication	computers	fiber optic cables
World Wide Web	web pages	hyperlinks
social	people	relationships
food web	species	predator-prey
software systems	functions	function calls
scheduling	tasks	precedence constraints
circuits	gates	wires

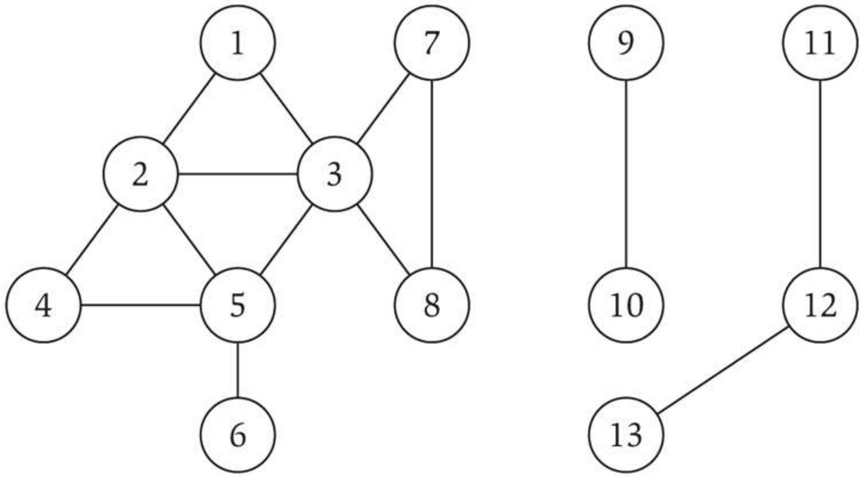
Ecological Food Web

- Food web graph.
  - Node = species.
  - Edge = from prey to predator.



cycle C = 1-2-4-5-3-1

Paths & Cycles



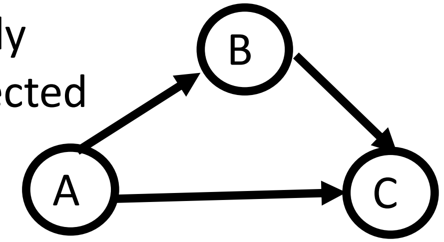
- A *path* in an undirected graph G = (V, E) is a sequence P of nodes $v_1, v_2, \ldots, v_{k-1}, v_k$ with the property that each consecutive pair v_i, v_{i+1} is joined by an edge in E.

- A path is *simple* if all nodes are distinct.

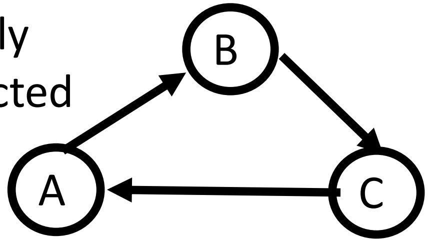
- A *cycle* is a path $v_1, v_2, \ldots, v_{k-1}, v_k$ in which $v_1 = v_k$, $k > 2$, and the first k-1 nodes are all distinct.

Connectivity & Distance

Weakly connected



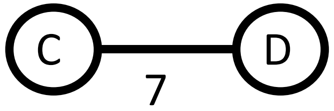
Strongly connected



- An undirected graph is *connected* if for every pair of nodes u and v, there is a path between u and v.
- In directed graphs, connectivity has two varieties.
  - A directed graph is *weakly connected* if it would be connected if it were undirected (ie, if the paths can “go the wrong way”)
  - A directed graph is *strongly connected* if there is a path from every node u to every other node v, and vice versa — each is reachable from the other.

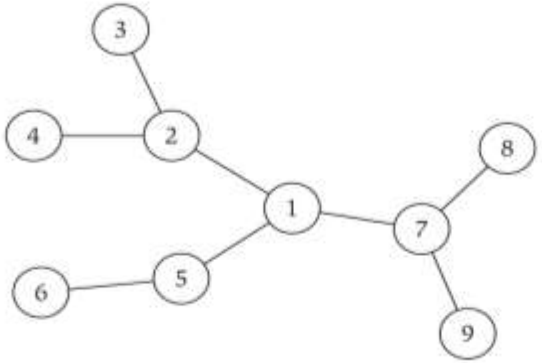
Unweighted

Weighted

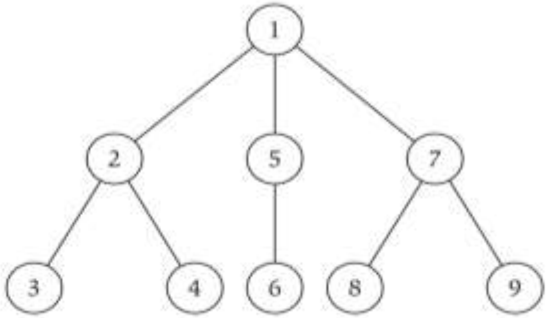


- By default, graphs are *unweighted*: the distance from a vertex to its *neighbor* (vertex reachable in one step) is assumed to be 1.
  - The *shortest path* between two nodes is the path between those nodes that uses the fewest edges
- *Weighted graphs* can represent distances and costs of paths between vertices (each edge gets a number that is its weight)
  - Shortest path (sometimes “cheapest path” to make a distinction) is path with smallest sum of weights.

Trees & Rooted Trees

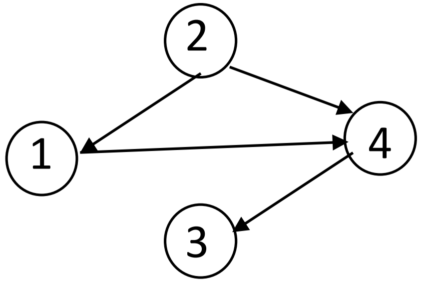


Same tree rooted at 1



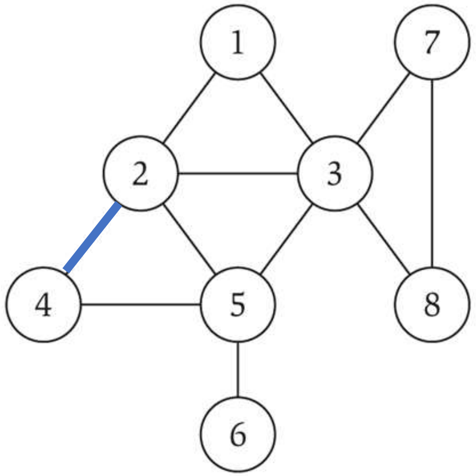
- A tree is an undirected connected graph that does not contain a cycle.
- Models hierarchical structure.
- Theorem. Let G be an undirected graph on n nodes. Any two of the following statements imply the third.
  - G is connected.
  - G does not contain a cycle.
  - G has n-1 edges.
- Rooted tree: choose a root node r and orient each edge towards r. Each edge goes from a child to a parent, directed paths from descendants to ancestors

- A DAG (directed acyclic graph) is the directed analog of a tree, a directed graph that does not contain a directed cycle.



Graph Representation: Adjacency Matrix

- Adjacency matrix. n-by-n matrix with A_{uv} = 1 if (u, v) is an edge.
  - Two representations of each edge.
  - Space proportional to n^2.
  - Checking if (u, v) is an edge takes $\Theta(1)$ time.
  - Identifying all edges takes $\Theta(n^2)$ time.

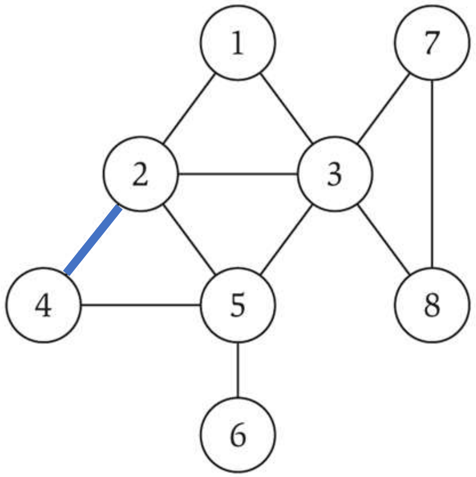


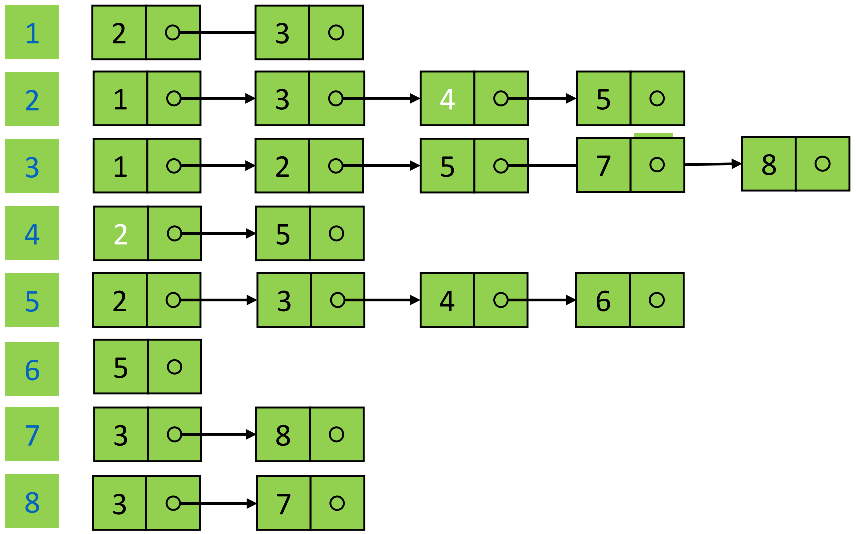
	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Graph Representation: Adjacency List

- Adjacency list. Node indexed array of lists.
  - Two representations of each edge.
  - Space proportional to m + n.
  - Checking if (u, v) is an edge takes O(deg(u)) time.
  - Identifying all edges takes Θ(m + n) time.

degree = number of neighbors of u





Summary

- Graphs are useful for representing a wide variety of situations
- Basic graph terminology: path, cycle, connected component, distance, DAG, trees…
- There are two main ways to represent graphical data - Adjacency Lists and Adjacency Matrices (which exemplify the pros and cons of linked lists vs. arrays)
- Main Takeaway: The organization of graphical data affects the running time of algorithms.