F, 09/13/19

Fall'19 CSCE 629

Analysis of Algorithms

Fang Song
Texas A&M U

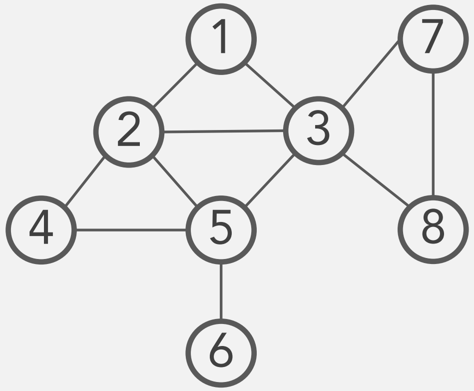
Lecture 6

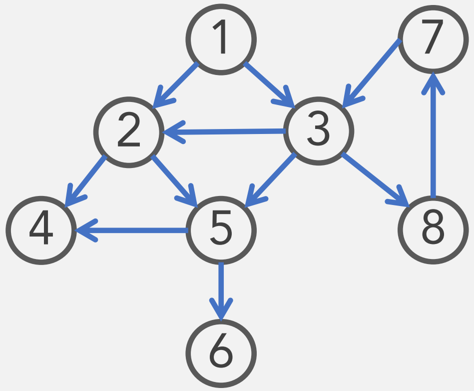
- Graph: terminology review
- Traversal
  - BFS
  - DFS
- Connected component

Credit: some based on slides by A. Smith & K. Wayne

Graph glossary

Graph $G = (V, E)$





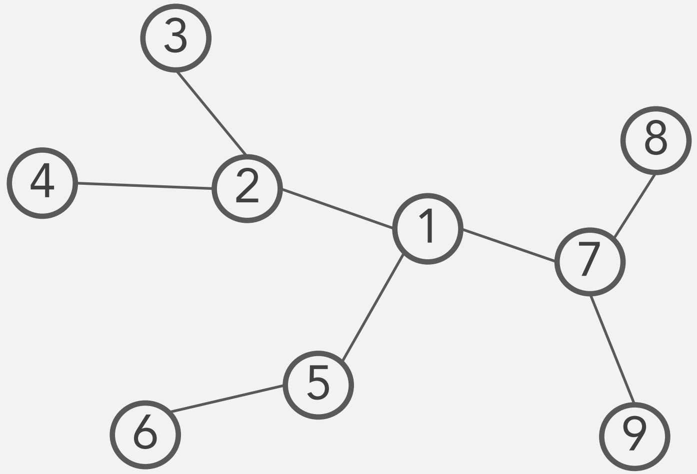
- Vertex/node, edge
- Undirected graph $e = (u, v)$
- Directed graph $e: u \to v$
- u adjacent to v, neighbors
- Degree $d(u)$
- Path, cycle
- u, v connected
- G connected: iff. u, v connected for any pair u and v

Warmup puzzles

- Suppose an undirected graph G is connected
  - True/False? G has at least $n - 1$ edges
- Suppose undirected G has exactly $n - 1$ edges (no self loops)
  - True/False? G is connected
  - What if in addition G has NO cycles?

Trees

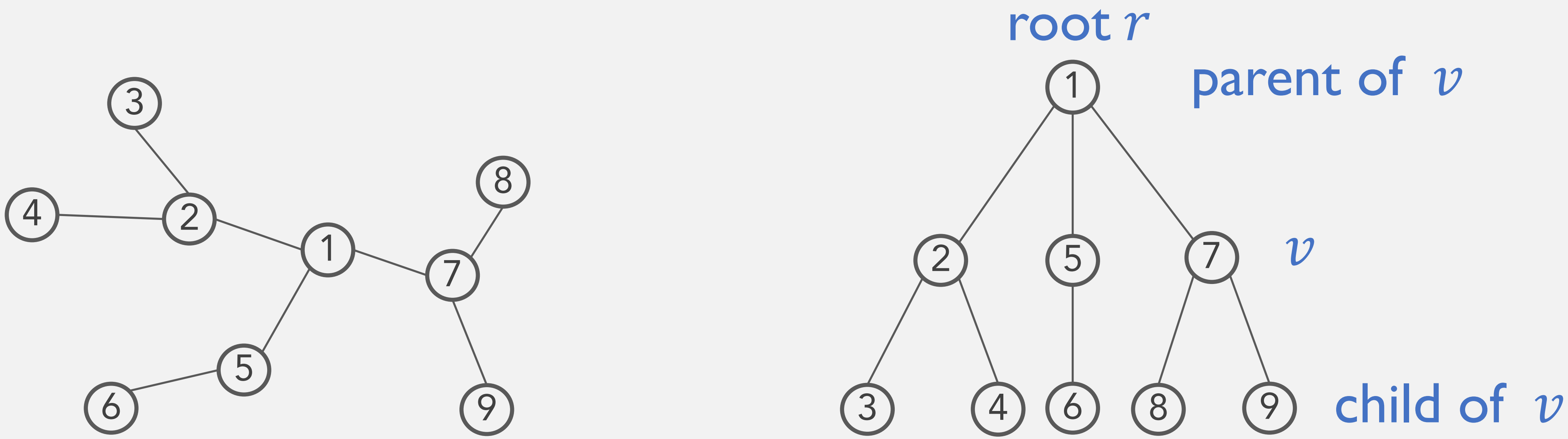
- Definition. An undirected graph is a tree if it is connected and does not contain a cycle.
- Theorem. Let G be an undirected graph on n nodes. Any two of the following statements imply the third.
  - G is connected.
  - G does not contain a cycle.
  - G has $n - 1$ edges.



Rooted trees

Given a tree T, choose a root node r
and orient each edge away from r.

Importance. Models hierarchical structure.



Exploring a graph

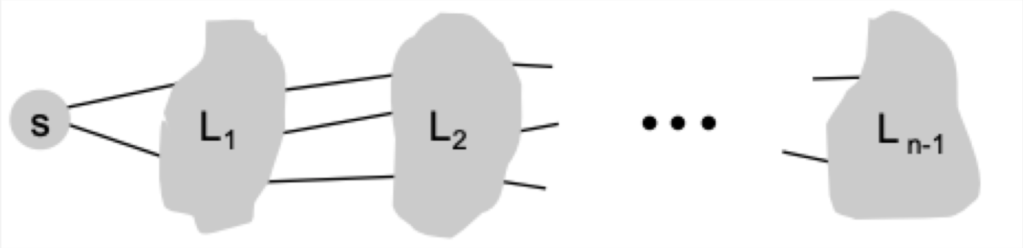
Connectivity problem:

Given vertices $s, t \in V$, is there a path from s to t?

- Breadth-first search (BFS)
  - Explore children in order of distance to start node
- Depth-first search (DFS)
  - Recursively explore vertex's children before exploring siblings

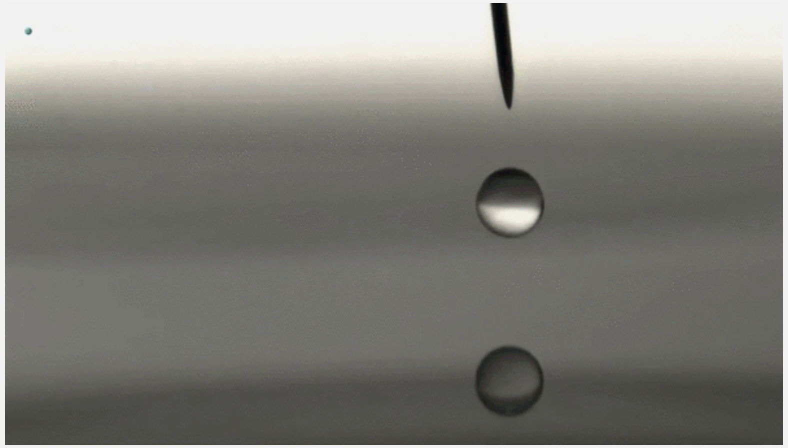
Breath-first search

Intuition. Explore outward from s in all possible directions, adding nodes one "layer" at a time.

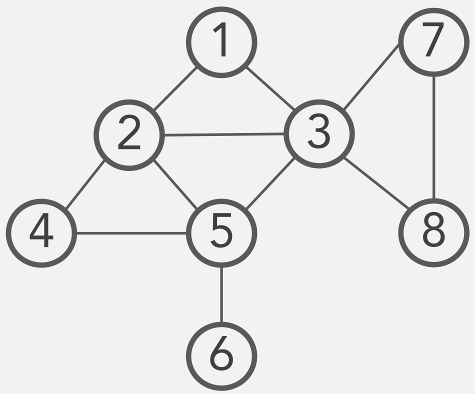


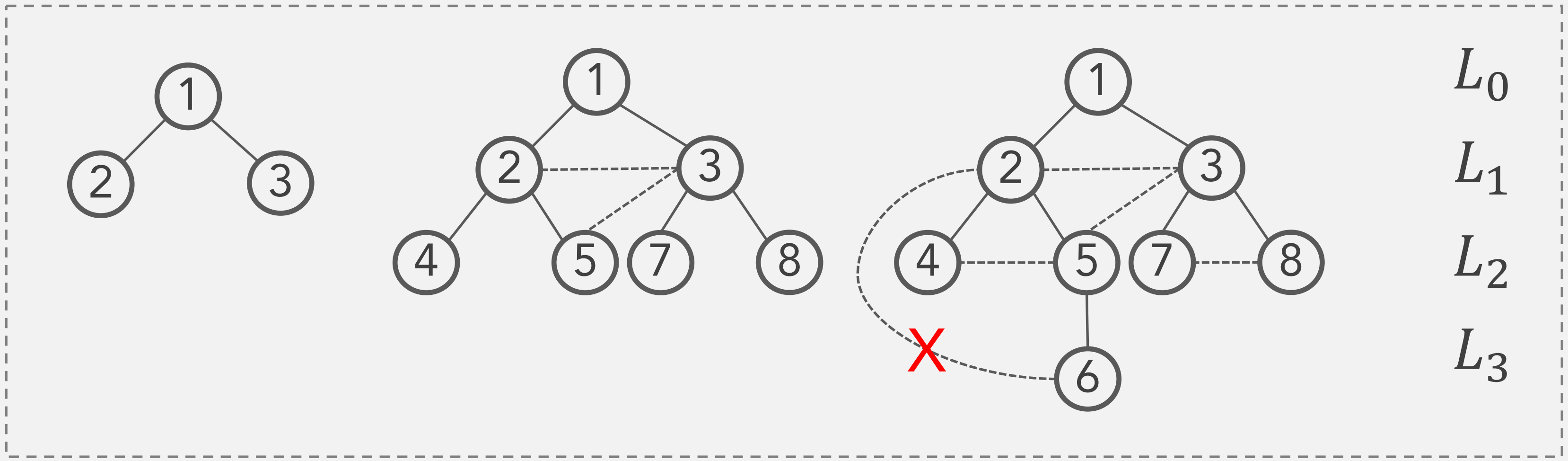
- $L_0 = \{s\}$
- $L_1 = \{\text{neighbors of } L_0\}$
- $L_2 = \{\text{neighbors of } L_1 \text{ not in } L_0 \ \& \ L_1\}$
- $\ldots$

Wave front of a ripple



Observations of BFS



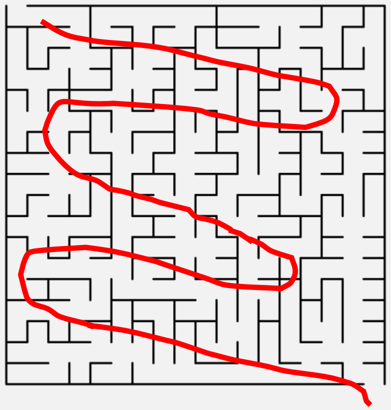


- Running time: linear $O(|V| + |E|)$ (more to come)
- For each i, L_i consists of all nodes at distance exactly i from s. There is a path from s to t iff. t appears in some layer.
- Let T be a BFS tree of $G = (V, E)$, and let (u, v) be an edge of G. Then, the levels of u and v differ by at most 1.

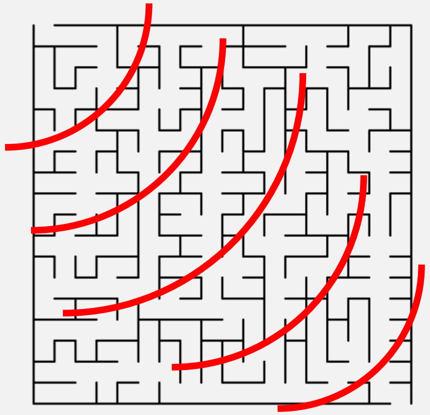
Depth-first search

Intuition. Children prior to siblings

DFS(s):
  // R will consist of nodes to which s has a path
  Mark u as "Explored" and add u to R
  for each edge (u, v) incident to u
    if v is not marked "Explored" then
      Recursively invoke **DFS**(v)



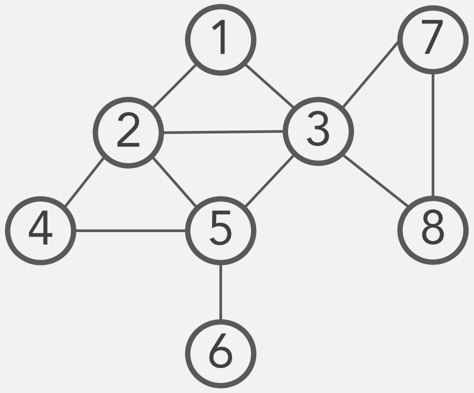
DFS
An "impatient"
maze runner

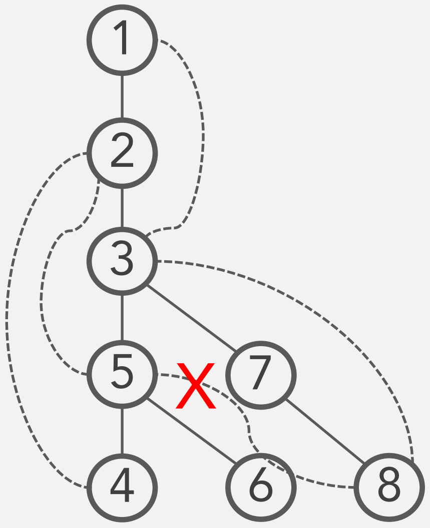


BFS
A "patient"
maze runner

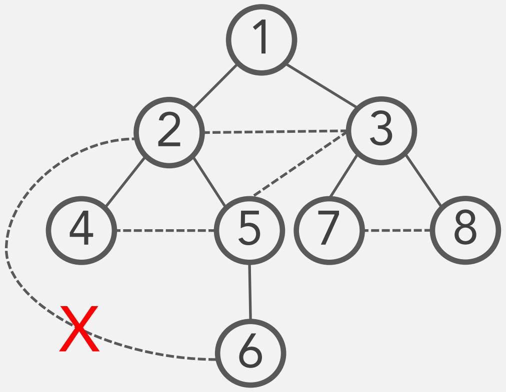
DFS in action

- Constructing DFS tree (on board)





Contrast with BFS tree



- Running time: linear $O(|V| + |E|)$ (more to come)
- Let T be a DFS tree of G, and let u & v be nodes in T. Let (u, v) be an edge of G that is not an edge of T. Then one of u or v is an ancestor of the other.

A lookahead

- Representation of graphs
  - Adjacency list vs. adjacency matrix
- BFS/DFS: some implementation details
- Connectivity in directed graphs
  - Topological sorting