HW5 will be posted after class.

CS4800: Algorithms & Data
Jonathan Ullman

Midterm will be back on tuesday.

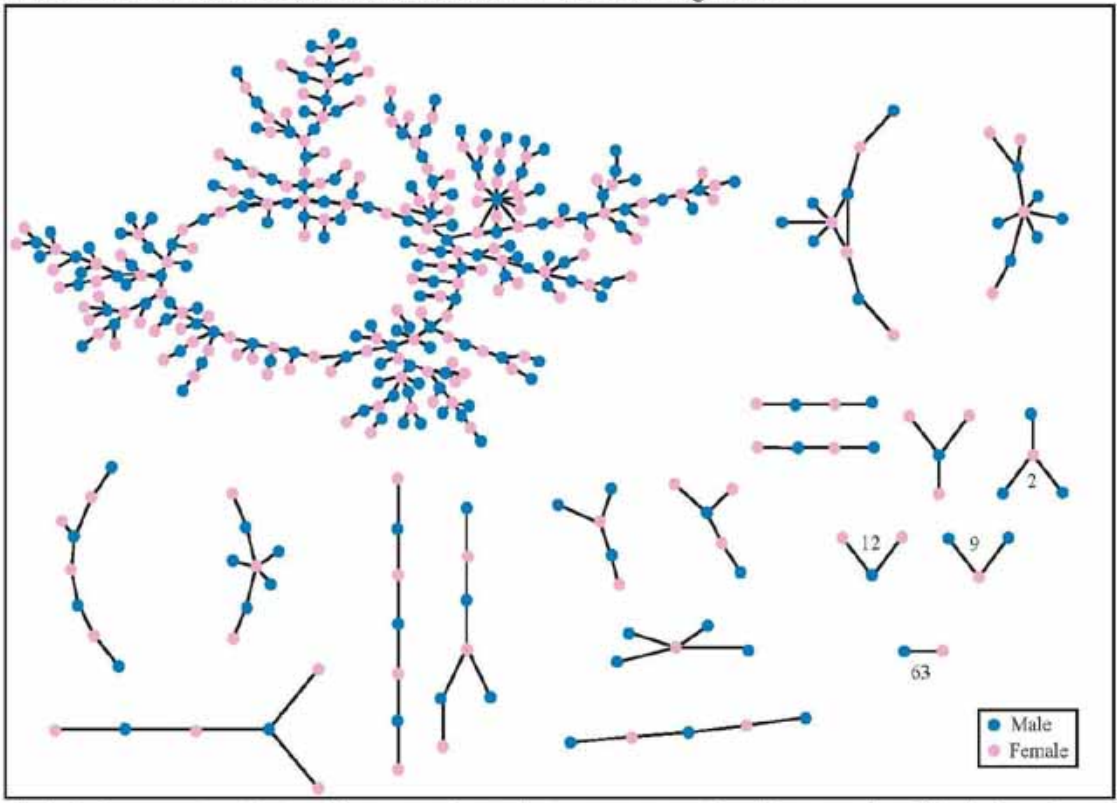
Lecture 11:

- Graphs
- Graph Traversals: BFS

Feb 16, 2018

What's Next

The Structure of Romantic and Sexual Relations at "Jefferson High School"



Each circle represents a student and lines connecting students represent romantic relations occuring within the 6 months preceding the interview. Numbers under the figure count the number of times that pattern was observed (i.e. we found 63 pairs unconnected to anyone else).

What's Next

- Graph Algorithms:
  - Graphs: Key Definitions, Properties, Representations
  - Exploring Graphs: Breadth/Depth First Search
    - Applications: Connectivity, Bipartiteness, Topological Sorting
  - Shortest Paths:
    - Dijkstra
  - Minimum Spanning Trees:
    - Borůvka, Prim, Kruskal
  - Network Flow:
    - Algorithms
    - Unlimited Applications ← Reductions

Heaps

Graphs

Graphs: Key Definitions

$|V| = n$ # nodes

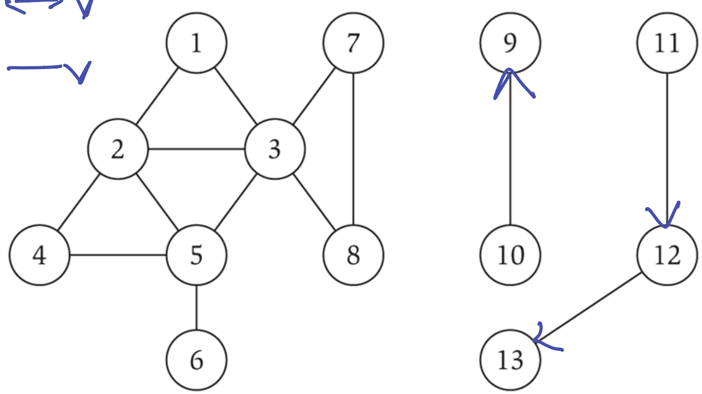
$|E| = m$ # edges

- Definition: A directed graph $G = (V, E)$
  - V is the set of nodes/vertices
  - $E \subseteq V \times V$ is the set of edges $u \to v$
  - An edge is an ordered $e = (u, v)$ "from u to v" directed
- Definition: An undirected graph $G = (V, E)$
  - Edges are unordered $e = (u, v)$ "between u and v" undirected

$u \leftrightarrow v$

$u \to v$

- Simple Graph:
  - No duplicate edges
  - No self-loops $e = (u, u)$



Ask the Audience

- How many edges can there be in a simple directed/undirected graph?

n^2

$n^2 - 1$

$\boxed{n^2 - n}$ ✓

$\underbrace{u}_{n \text{ choices}} \rightarrow \underbrace{v}_{n-1 \text{ choices}}$, no edge $u \rightarrow u$

$\#$ of edges $\leq n(n-1)$

$\underbrace{u}_{n \text{ choices}} \leftrightarrow \underbrace{v}_{n-1 \text{ choices}}$

$\#$ of edges $\leq \frac{n(n-1)}{2}$

$$\boxed{0 \leq m = O(n^2)}$$

Graphs Are Everywhere

- Transportation networks
- Communication networks
- WWW
- Biological networks
- Citation networks
- Social networks
- …

nodes = places
edges = roads

nodes = species
edges = evolutionary/ancestors

nodes = people
edges = friendships

Paths/Connectivity

for both directed and undirected

- A path is a sequence of consecutive edges in E
  - $(u, w_1), (w_1, w_2), (w_2, w_3), \dots, (w_{k-1}, v)$
  - The length of the path is the # of edges

    $u - w_1 - w_2 \cdots - w_{k-1} - v$

    one edge = path of length 1
- An undirected graph is connected if for every two vertices $u, v \in V$, there is a path from u to v

Paths/Connectivity

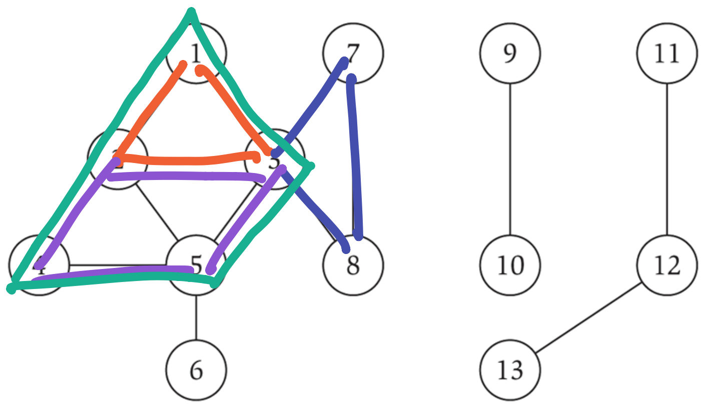
- A path is a sequence of consecutive edges in E
  - $(u, w_1), (w_1, w_2), (w_2, w_3), \dots, (w_{k-1}, v)$
  - The length of the path is the # of edges

- A directed graph is strongly connected if for every two vertices $u, v \in V$, there are paths from u to v and from v to u

Cycles

- A cycle is a path $v_1 - v_2 - \cdots - v_k - v_1$ where $k \geq 2$ and $v_1, \ldots, v_k$ are distinct

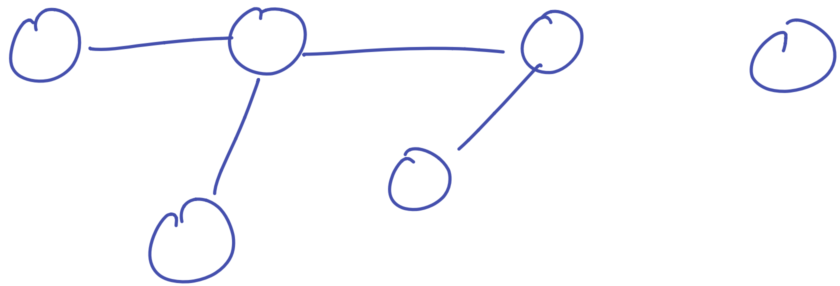
3 — 7 — 8 — 3



Ask the Audience

w/ n nodes

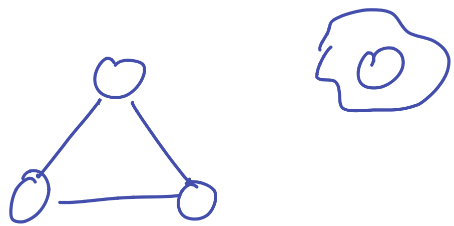
- Suppose an undirected graph G is connected
  - True/False? G has $\geq n-1$ edges



$$\Omega(n) = m = O(n^2)$$

Ask the Audience

- Suppose an undirected graph G has $= n - 1$ edges
  - True/False? G is connected



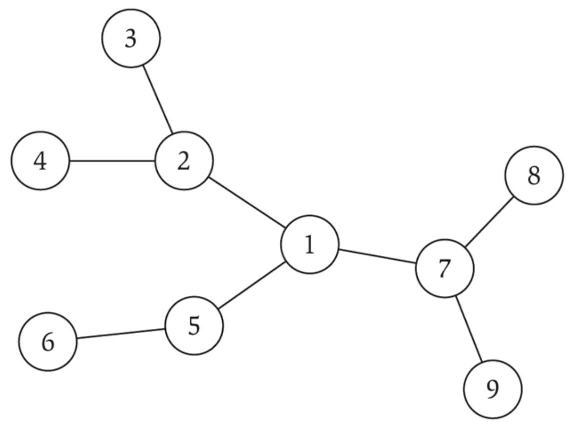
$n = 4$

$m = 3 = n - 1$

Trees

- An undirected graph G is a tree if:
  - G is connected
  - G contains no cycles

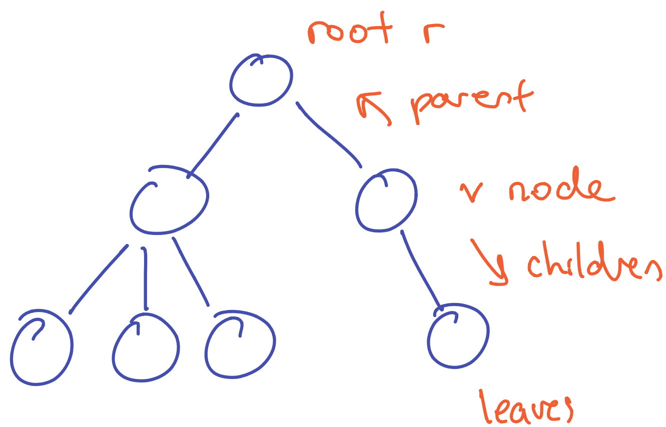
- Theorem: any two of the following implies the third
  - G is connected
  - G contains no cycles
  - G has $= n - 1$ edges



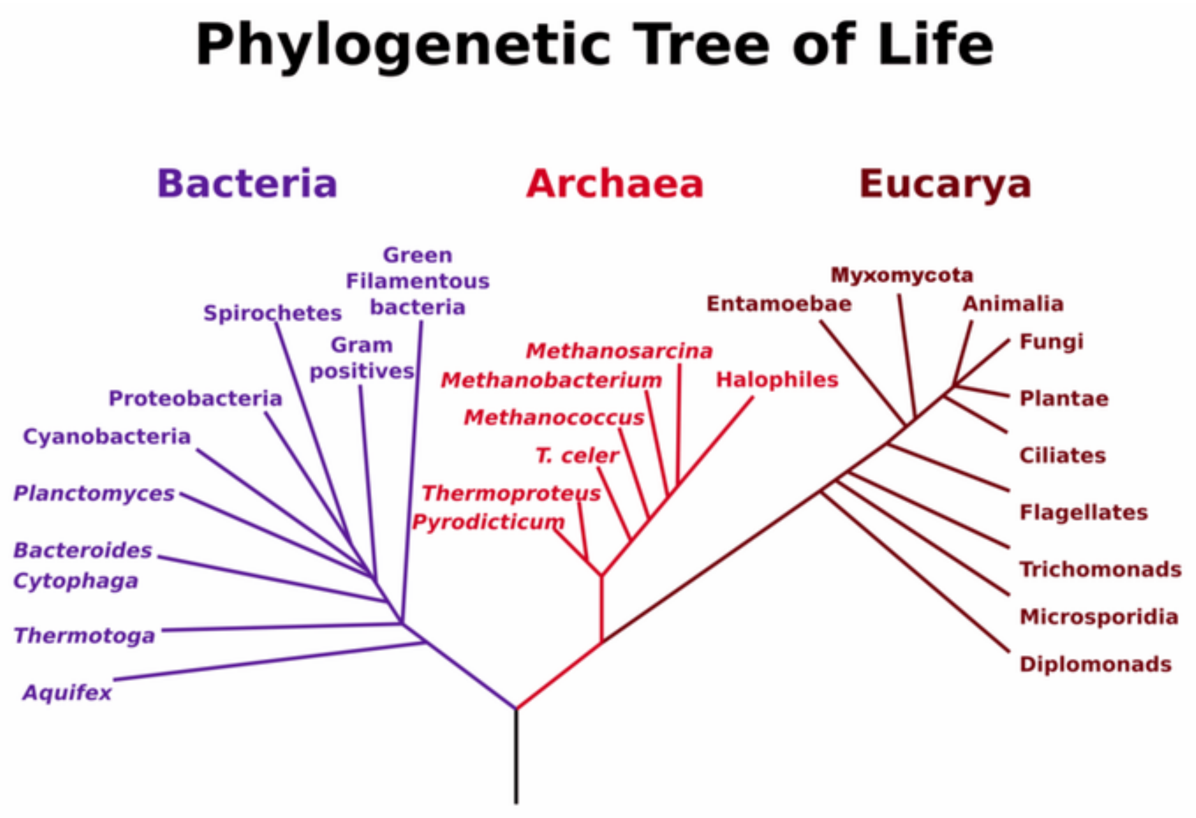
Trees

- Rooted tree: choose a root node r and orient edges away from r
  - Models hierarchical structure

- Rooted tree: choose a root node r and orient edges away from r
  - Models hierarchical structure

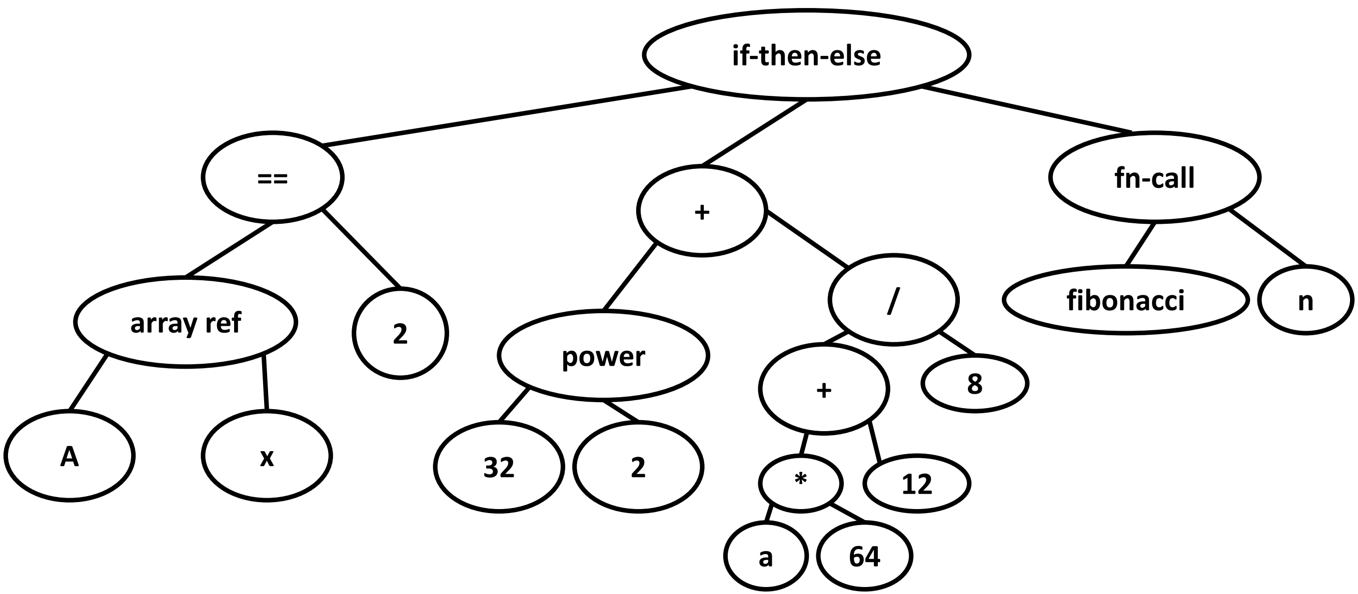


Phylogeny Trees



Parse Trees

```
if (A[x]==2) then
   (32² + (a*64 +12)/8)
else
   fibonacci(n)
```



Exploring a Graph

Exploring a Graph

- Problem: Is there a path from s to t?
- Idea: Explore all nodes reachable from s.

- Two different search techniques:
  - Breadth-First Search: explore all nearby nodes before moving on to further away nodes
  - Depth-First Search: follow a path until you get stuck

Exploring a Graph

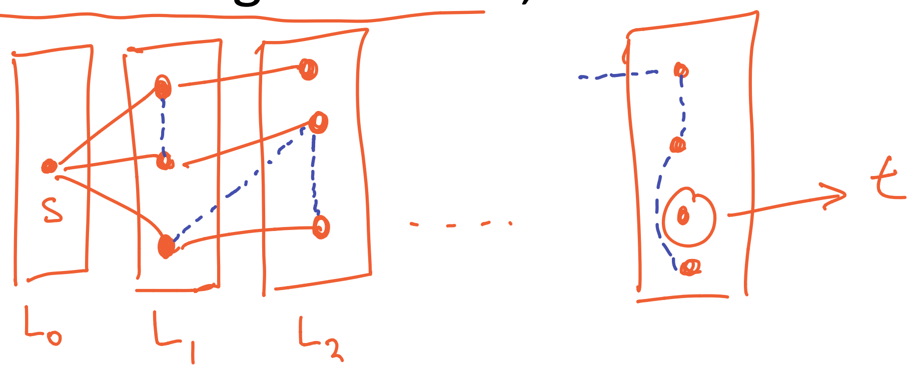
- BFS/DFS are a General Template for Graph Algs
  - Extensions of Breadth-First Search:
    - 2-Coloring (Bipartiteness)
    - Shortest Paths
    - Minimum Spanning Tree (Prim's Algorithm)
  - Extensions of Depth-First Search:
    - Finding Cycles
    - Topological Sorting
    - Strongly Connected Components

Breadth-First Search (BFS)

v is a neighbor of u if $(u,v) \in E$

0

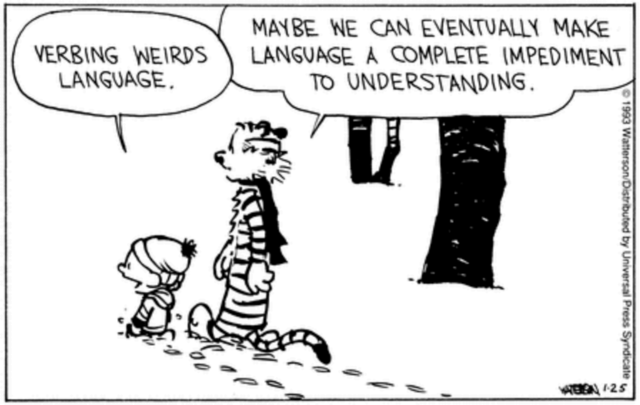
- Informal Description: start at s, find all $\boxed{\text{neighbors}}$ of s, find all neighbors of neighbors of s, …

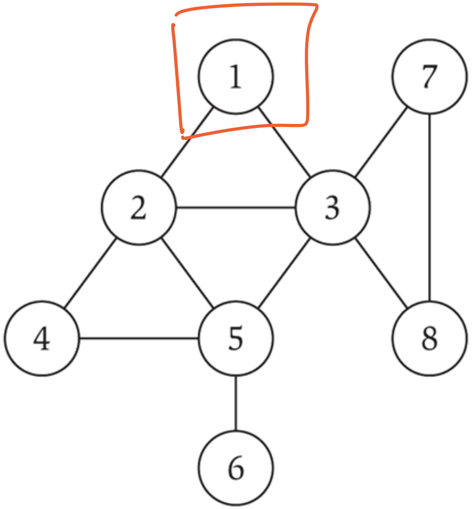


- BFS Algorithm:
  - $L_0 = \{s\}$
  - $L_1 =$ all neighbors of L_0
  - $L_2 =$ all neighbors of L_1 that are not in L_0, L_1
  - …
  - $L_d =$ all neighbors of L_{d-1} that are not in $L_0, \ldots, L_{d-1}$
  - Stop when L_{d+1} is empty.

Ask the Audience

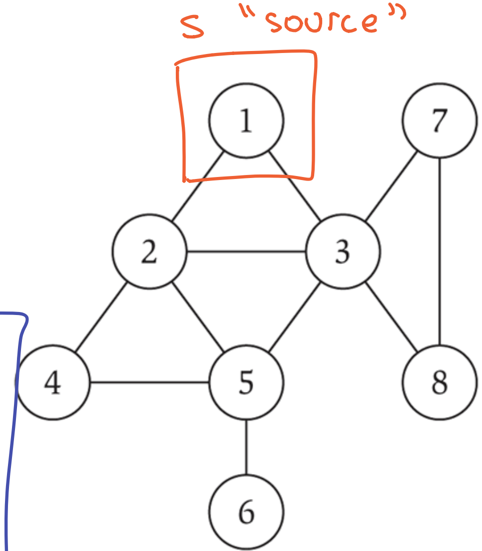
- BFS this graph from $s = 1$

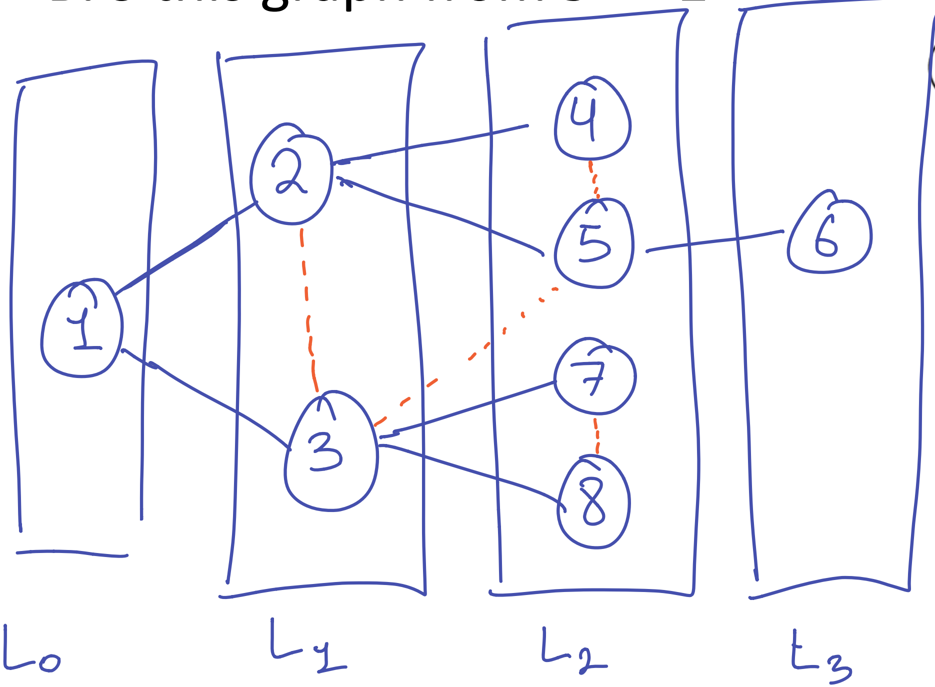




Ask the Audience

- BFS this graph from $s = 1$



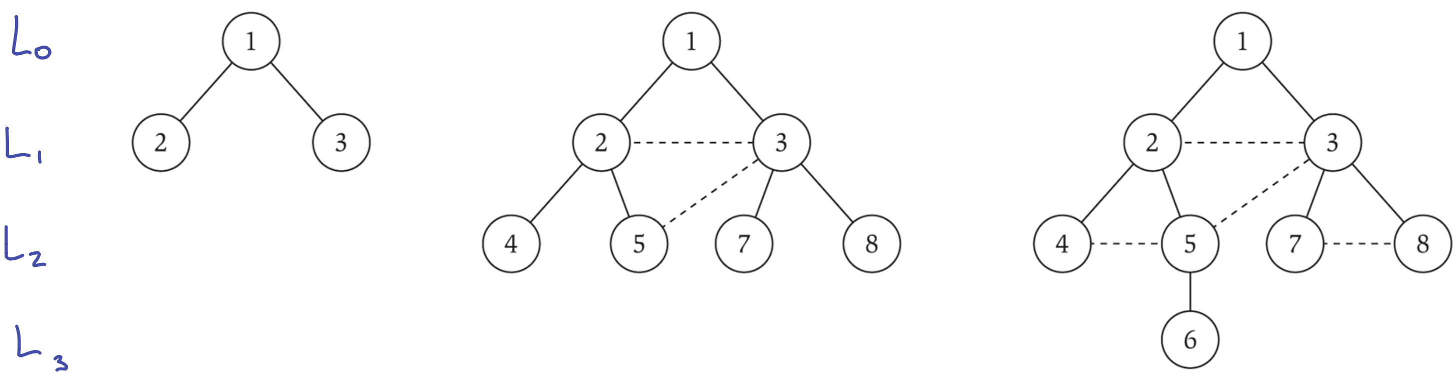


blue edges are the "BFS tree"

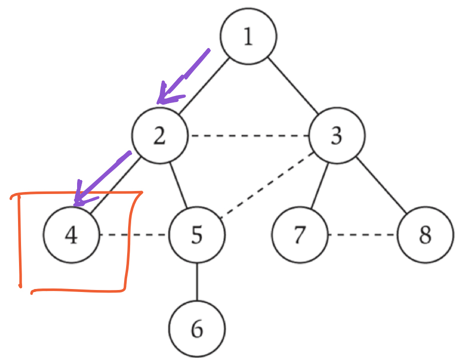
- all BFS trees have the same layers

Breadth-First Search (BFS)

- Definition: the distance between s, t is the number of edges on the shortest path from s to t
- Theorem: BFS reveals the distance from s to all other nodes!
  - Nodes in layer L_i have distance exactly i from s
  - Nodes not in any layer are not reachable from s



Theorem: All nodes in layer i have distance exactly i from s.



- Everything in layer 0 has dist 0
- Everything in layer 1 has dist 1

Every node in layer 2 has distance ≤ 2.

Every node in layer 2 has distance ≥ 2

$\implies$ distance $= 2$.

Implementing Graph Search

GenericSearch(s):
$R = \{s\}$
While there is an edge (u, v) where $u \in R, v \notin R$
Add v to R

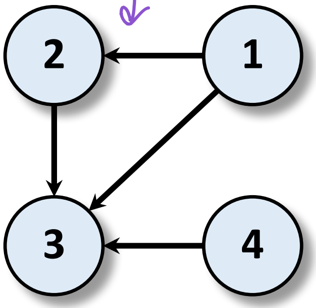
- To implement we need to decide:
  - How to represent the graph as input?
  - How to track the vertices that are already explored?
  - How to choose the next edge to explore?

Adjacency-Matrix Representation

- The adjacency matrix of a graph $G = (V, E)$ with n nodes is the matrix $A[1{:}n, 1{:}n]$ where

$$A[i,j] = \begin{cases} 1 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

A	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0



Cost

Space: $\Theta(V^2)$ $\Theta(n^2)$

Lookup: $\Theta(1)$ time

List Neighbors: $\Theta(V)$ time

Adjacency-List Representation

- The adjacency list of a vertex $v \in V$ is the list $A[v]$ of all the neighbors of v

one linked list per person

$$\boxed{A[1] = \{2,3\}}$$

$$A[2] = \{3\}$$

$$A[3] = \{\,\}$$

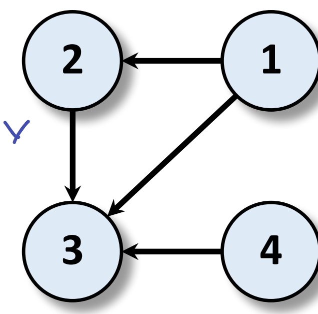
$$A[4] = \{3\}$$

Costs:

Total size of all linked lists $\Theta(m+n)$

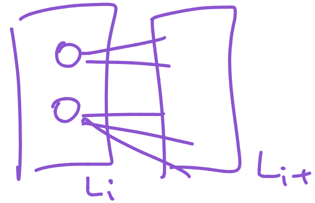
List neighbors of v $O(\underbrace{\deg(v)}_{\text{\# of neighbors of } v})$

Lookup $O(\deg(v))$



Total Running Time $= O(n+m)$

BFS Implementation



Initialize found$[v] \leftarrow$ false $\forall\, v \in V$

BFS(s):

  Initialize found$[s] \leftarrow$ true
  Initialize layer$[s] \leftarrow 0$, layer$[v] \leftarrow \infty\ \forall\, v \neq s$
  Initialize $i \leftarrow 0$, $L_0 \leftarrow \{s\}$, $T \leftarrow \emptyset$

$O(n)$

  While (L_i is not empty)
    Initialize a new layer $L_{i+1} \leftarrow \emptyset$
    For ($u \in L_i$):

$\sum_{u \in V} O(\deg(u)) = O(m)$

Each appears twice

$O(\deg(u))$

      For ($(u, v) \in E$):
        If (found$[v] =$ false) then
          Set found$[v] \leftarrow$ true, layer$[v] \leftarrow i + 1$
          Add (u, v) to T, add v to L_{i+1}

$O(1)$

    Increment the layer $i \leftarrow i + 1$