Goals for today

refresh on graph notation

examples of proofs about properties of graphs

understand BFS algorithm and runtime

understand what a topological sorting is

Undirected graphs

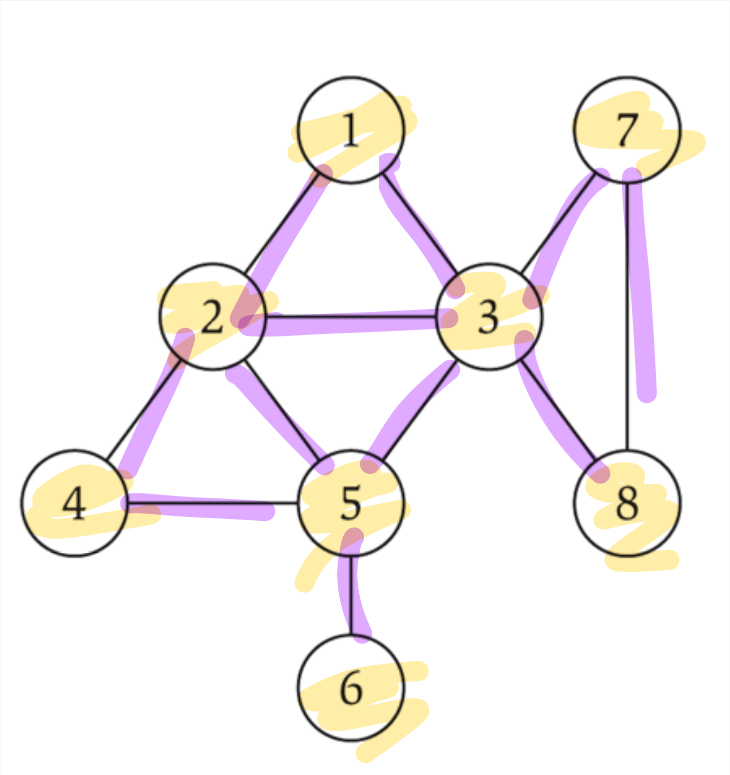
Notation. $G = (V, E)$

verts edges

$|V| = n$

$|E| = m$

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



ex: $n = \#$ verts $= 8$

$m = 11$

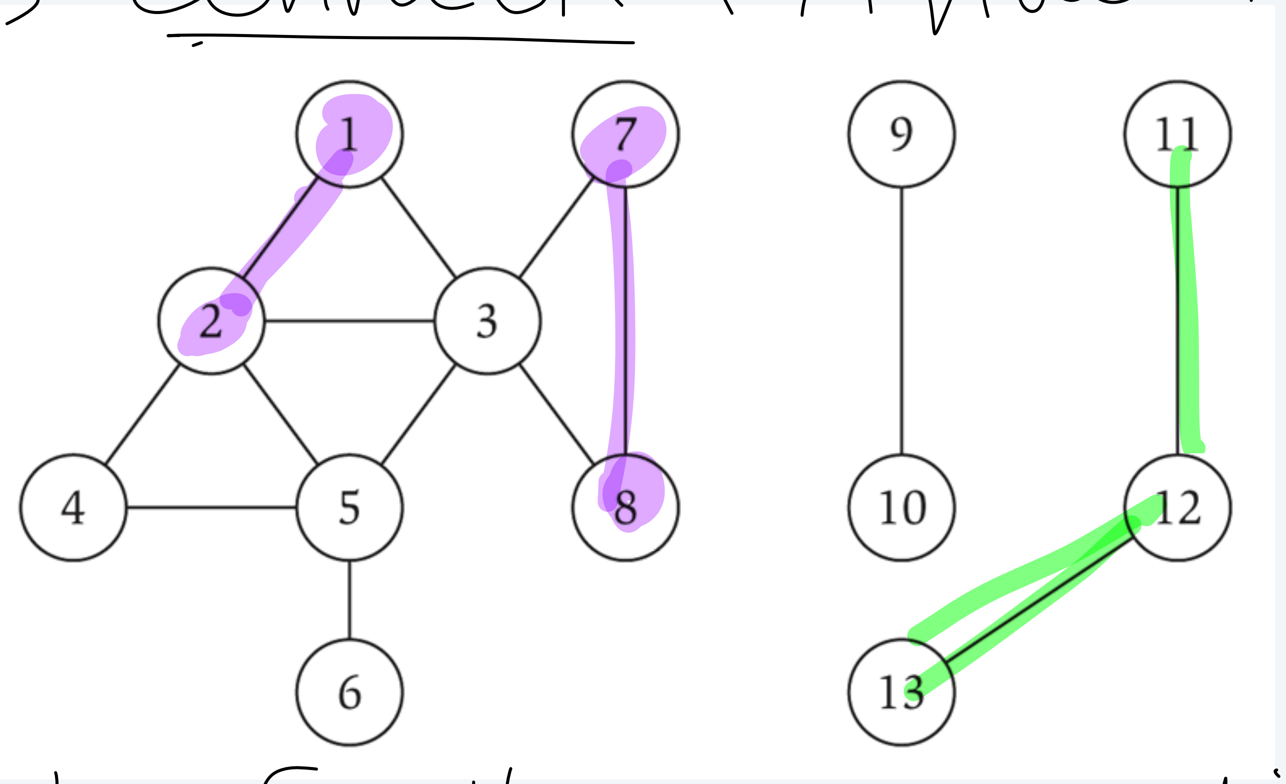
Paths and connectivity

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

proposed path: 1, 2, 7, 8 X

proposed path: 11, 12, 13, 12

a graph is connected if for every pair of nodes, there is a path btwn them.

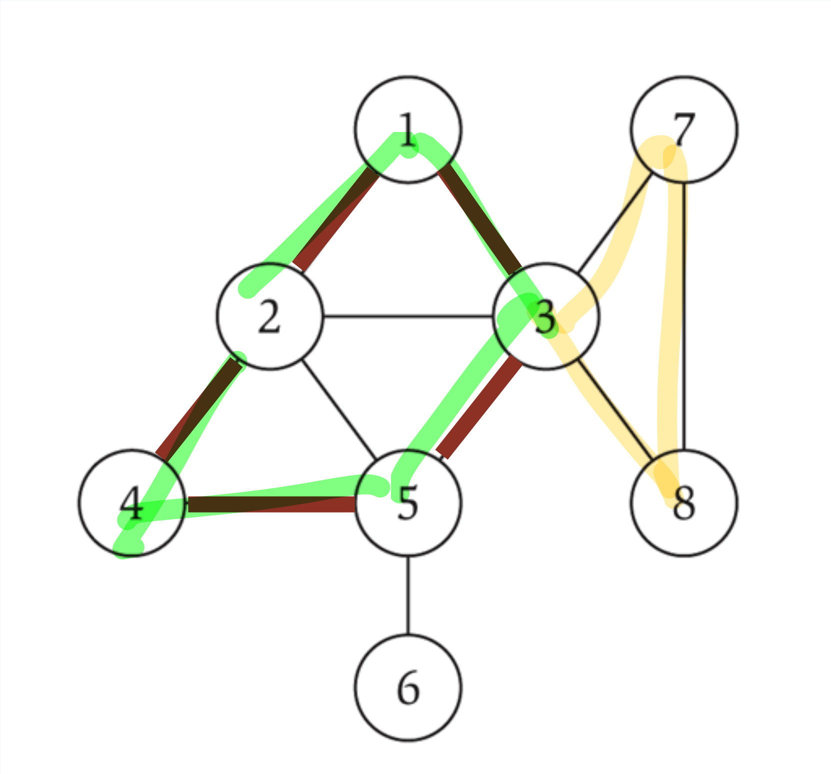


path is simple if all nodes are distinct.

Cycles

Def. A cycle is a path $v_1, v_2, \ldots, v_k$ in which $v_1 = v_k$ and $k \geq 2$.

Def. A cycle is simple if all nodes are distinct (except for v_1 and v_k).



cycle C = 1–2–4–5–3–1

is same as
2-4-5-3-1-2

simple

in tables: how many distinct cycles in ?

Cycles

Def. A cycle is a path $v_1, v_2, \ldots, v_k$ in which $v_1 = v_k$ and $k \geq 2$.

Def. A cycle is simple if all nodes are distinct (except for v_1 and v_k).

length 3

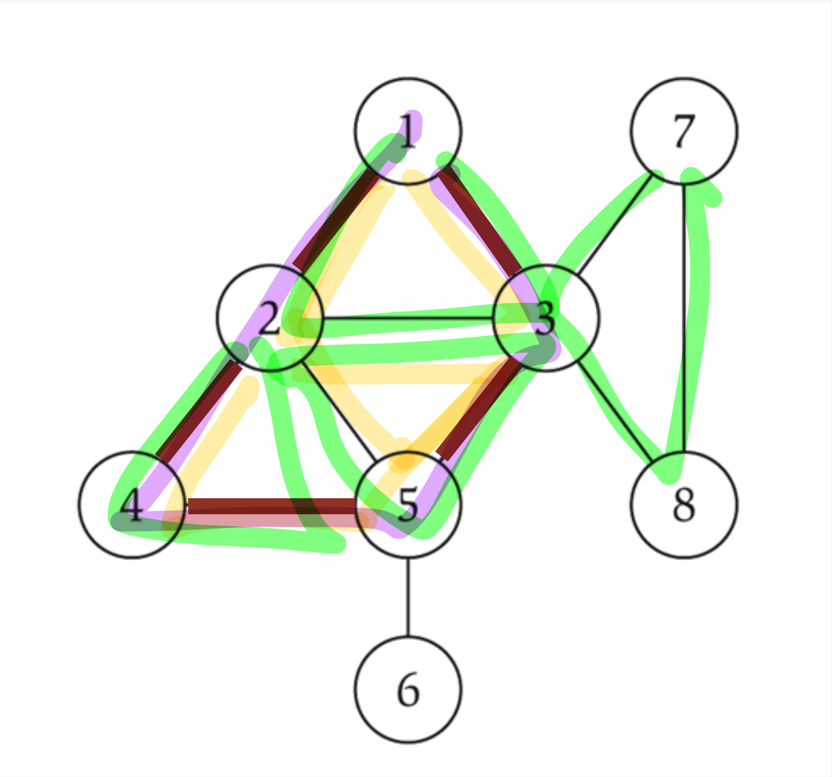
4

length 4

2

length 5

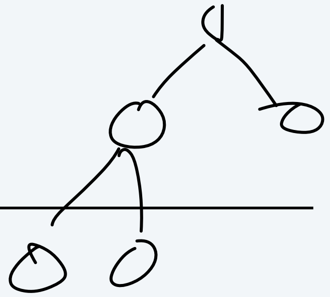
1



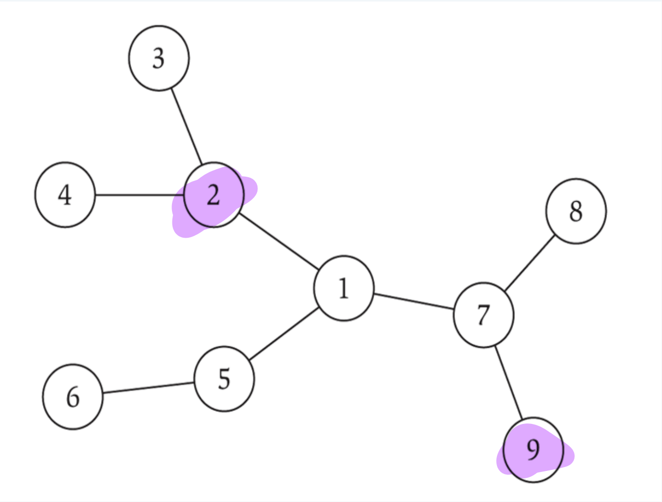
cycle C = 1–2–4–5–3–1

How many simple cycles are there in this graph?

Trees

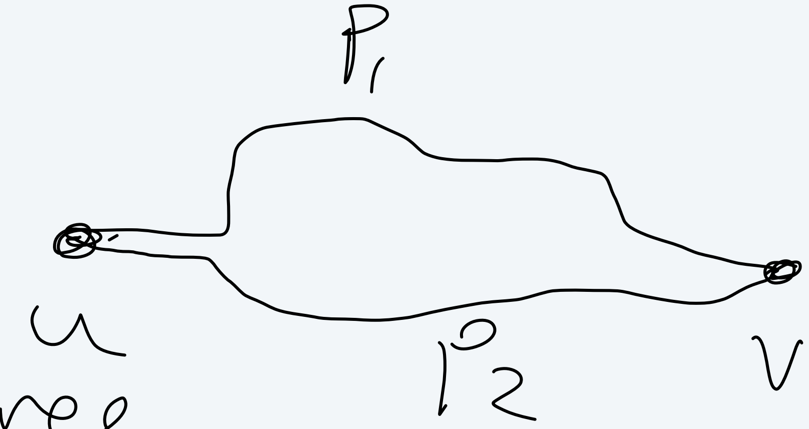


Def. An undirected graph is a tree if it is connected and does not contain a cycle.



claim: in a tree, for any pair of nodes u,v there is a unique path from u to v.

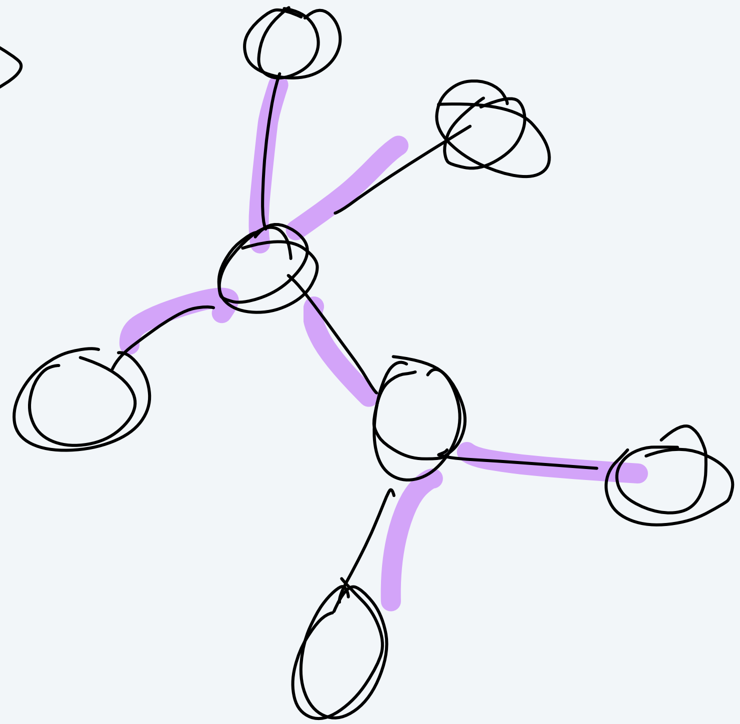
Proof: Let T be a tree. Let u, v be two nodes from T. For the sake of contradiction, suppose that there are two unique paths between u and v, P_1 and P_2. But following P_1 and then P_2 is a cycle, contradicting that T is a tree.



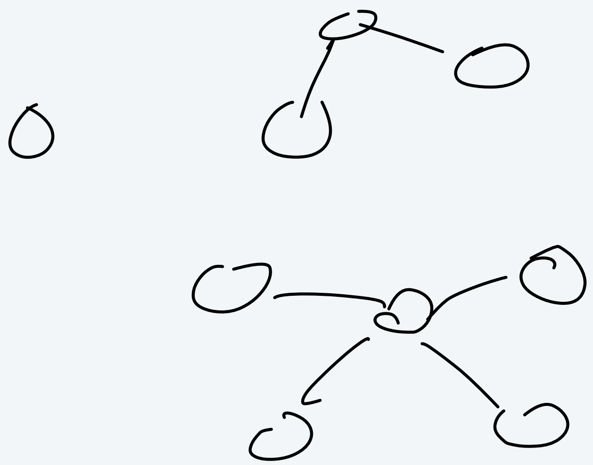
Suppose G is a tree with n nodes. How many edges does it have?

connected!

1. 0
2. [✓ $n - 1$]
3. $n + 1$
4. n^2
5. It depends



if disconnected,

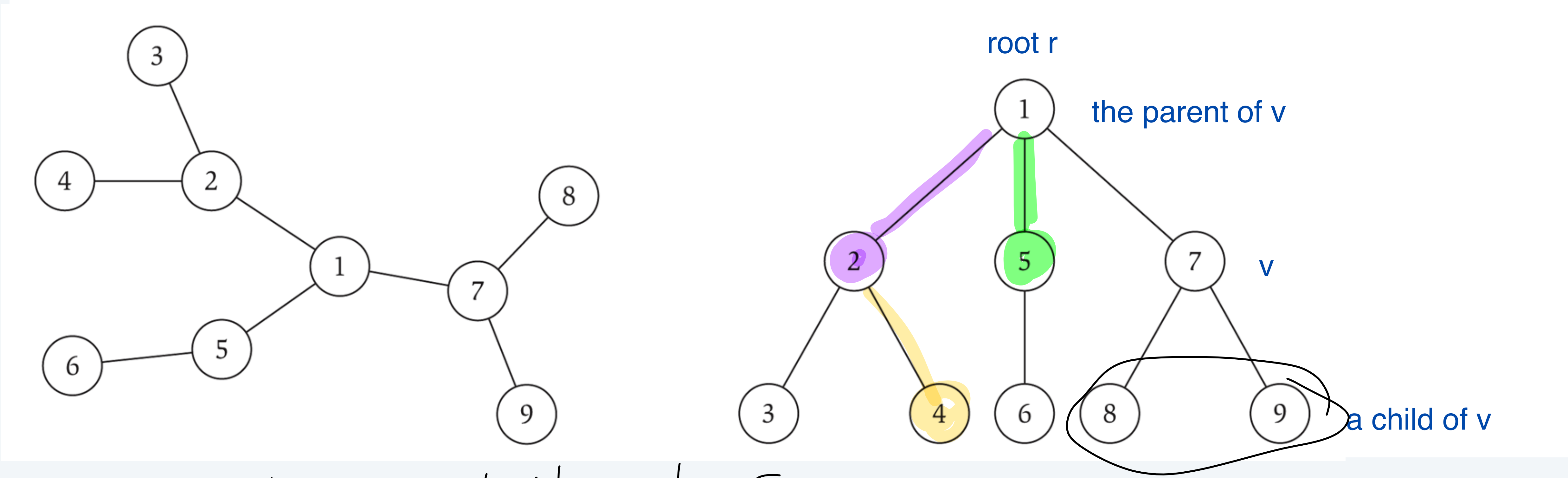


$n = 7 \quad m = 6$

$x < 5 + 1$

Rooted trees

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



Given T, root it at r.

Identify every edge w/ node below it.

Since there is only one path btwn every pair of nodes, no node can have more than one edge entering from above.

The root has no edge above. Every other node has exactly one edge above. n-1 edges.

Connectivity

s-t connectivity problem. Given two nodes s and t, is there a path between s and t ?

s-t shortest path problem. Given two nodes s and t, what is the length of a shortest path between s and t ?

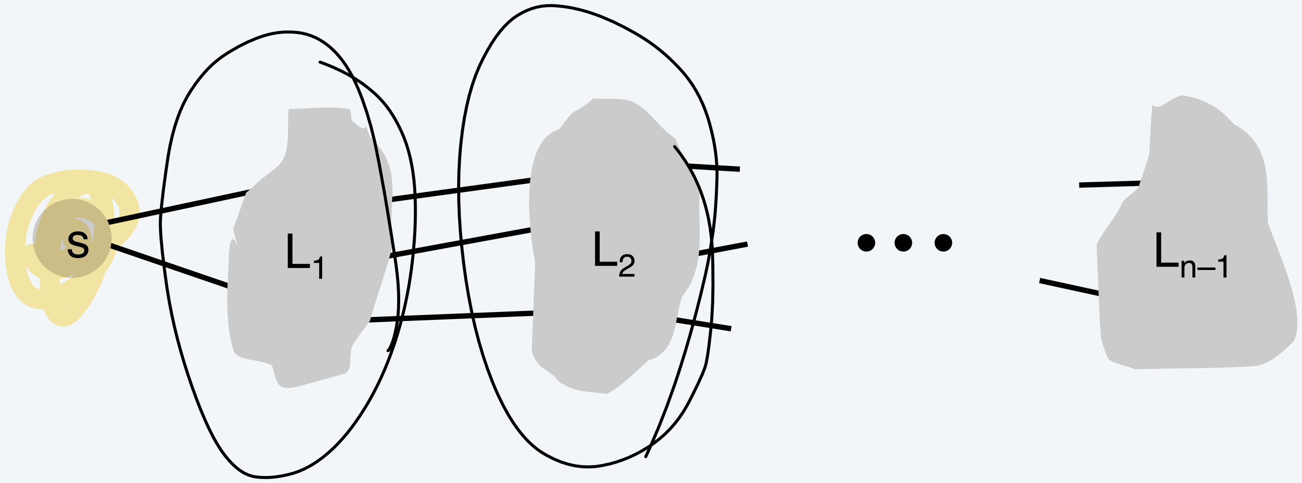
→ # edges
if no path, ∞

BFS

Given graph,
- how to answer connectivity ? how quickly?
- how to answer shortest path ? how quickly

Breadth-first search

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



Layers

$L_0 = \{s\}$

L_1 = all nodes connected to s

L_2 = all nodes connected to nodes in L_1
except s

⋮

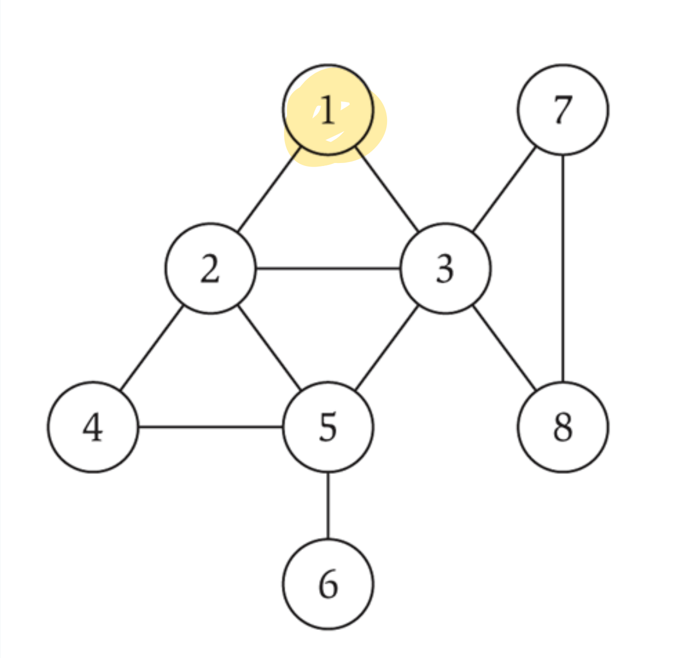
L_i = all nodes connected to nodes in L_{i-1} but not in any previous layer

Breadth-first search

Example

$L_0 = \{1\}$

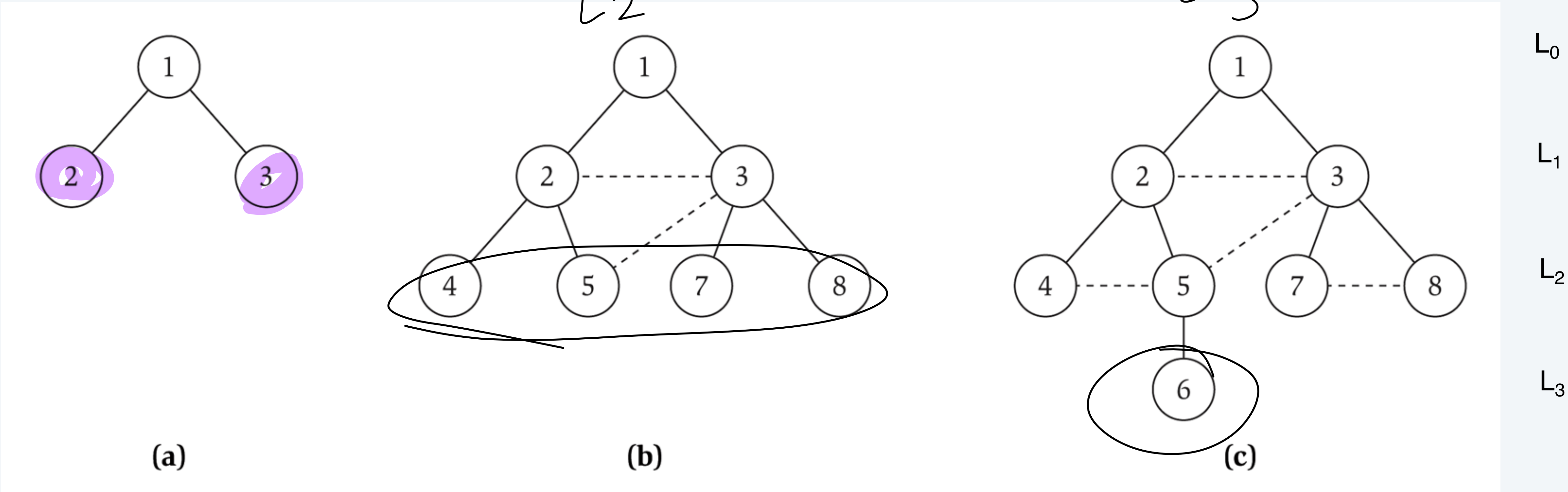
dist from 1 to 8



$L_1 = \{2, 3\}$

L_2

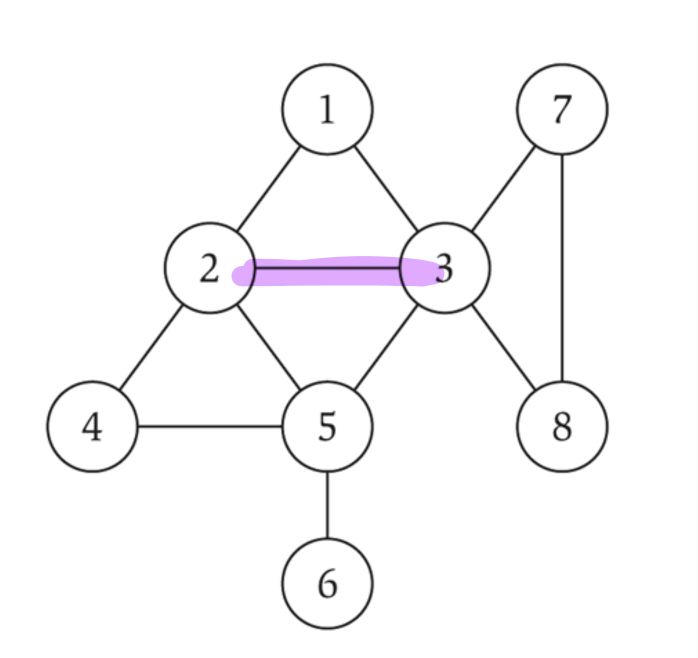
L_3



(a) (b) (c)

A helpful property of the BFS output

Property. Let T be a BFS tree of $G = (V, E)$, and let (x, y) be an edge of G. Then, the level of x and y differ by at most 1.



Proof: cases

case 1 BFS adds x first.

case 2 BFS adds y first

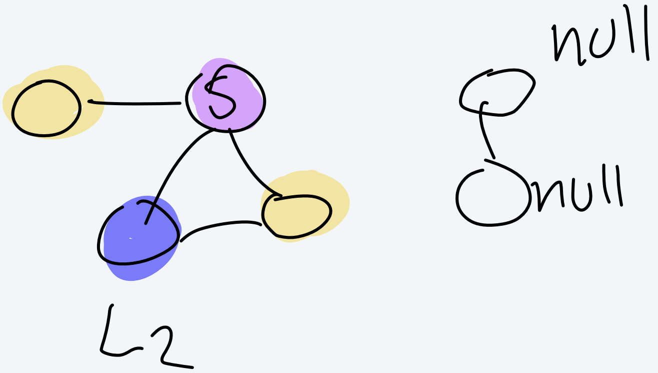
case 3 BFS adds x, y at same time

BFS pseudocode — connected

~~Say we're interested in some specific node v and our graph is connected.~~

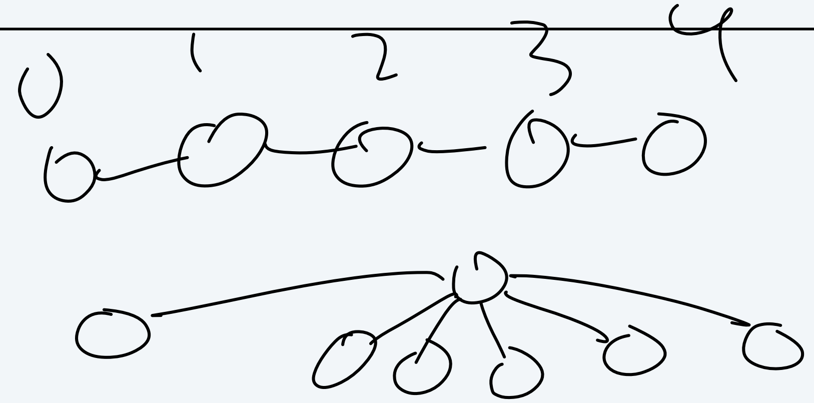
```
Set all nodes' layer to null, except s's layer to 0.
L = 0    (current layer)
While there is some node with a null layer:
    Mark all nodes that have a null layer
    and are adjacent to a node in L with
    L+1

    L = L+1
```



How can I adapt this code to answer connectivity?

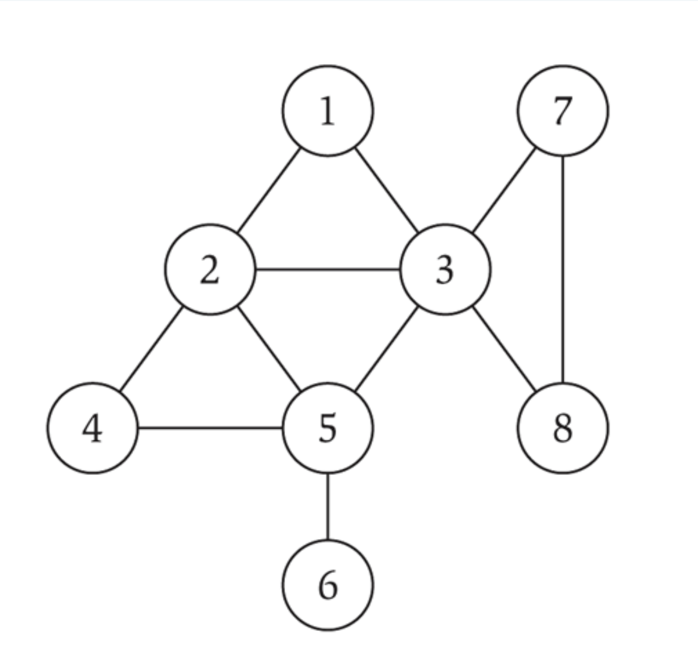
BFS runtime (connected graph)



BFS Pseudocode:

```
Set all nodes' layer to null, except set v's to 0
Set L = 0
While there is some node with a null layer
    Mark all nodes that have a null layer and are adjacent to a node in L as L + 1
    set L = L + 1
```

$\Theta(n^2)$



idea #2 $\Theta(n^2)$

- how many times does the loop run? O n
- how much time does the loop take? n

idea #2

- we only look at an edge once $\Theta(m+n)$
- make an argument that we only need constant ops per edge