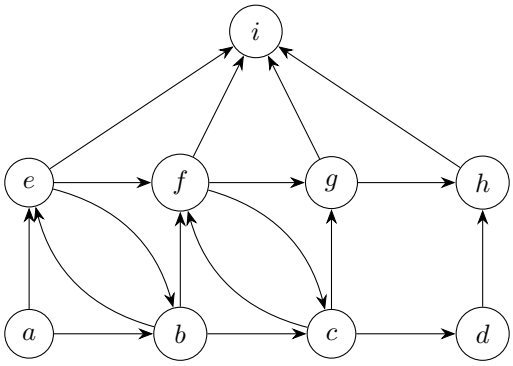
DSC 40B - Discussion 06

Problem 1.

In lecture, we saw that breadth-first search (BFS) can be used to find the shortest paths from a source node to all other nodes in an unweighted graph. Our implementation of BFS returned two dictionaries: `distance`, containing the shortest path distance from the source node to each node in the graph, and `predecessor`, containing the BFS predecessor of each node.

Consider a breadth-first search on the graph shown below, starting with node a.



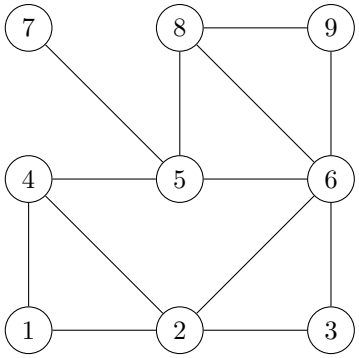
Write down the `distance` and `predecessor` dictionaries returned by BFS on this graph.

Note: you should adopt the convention that `.neighbors()` returns the neighbors of a node in **ascending alphabetical order**.

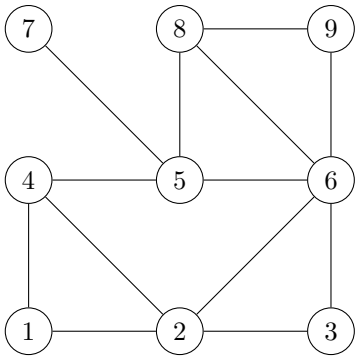
Problem 2.

For the following problems, recall that (u, v) is a *tree edge* if node v is discovered while visiting node u during a breadth-first or depth-first search. Assume the convention that a node's neighbors are produced in ascending order by label. You do not need to show your work for this problem.

a) Suppose a breadth-first search is performed on the graph below, starting at node 1. Mark every BFS tree edge with a bold arrow emanating from the predecessor.



b) Suppose a depth-first search is performed on the graph below, starting at node 1. Mark every DFS tree edge with a bold arrow emanating from the predecessor.



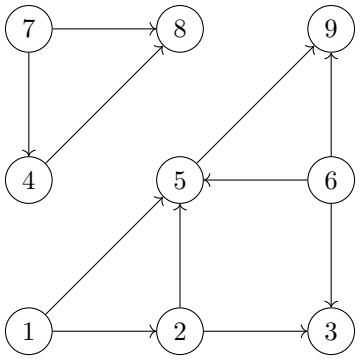
c) Fill in the table below so that it contains the start and finish times of each node after a DFS is performed on the above graph using node 1 as the source. Begin your start times with 1.

Node	Start	Finish
1		
2		
3		
4		
5		
6		
7		
8		
9		

Problem 3.

Topologically sort the vertices of the following graph. Note that there may be multiple, equally-correct topological sorts.

You do not need to show your work for this problem.


7
8
9
4
5
6
1
2
3