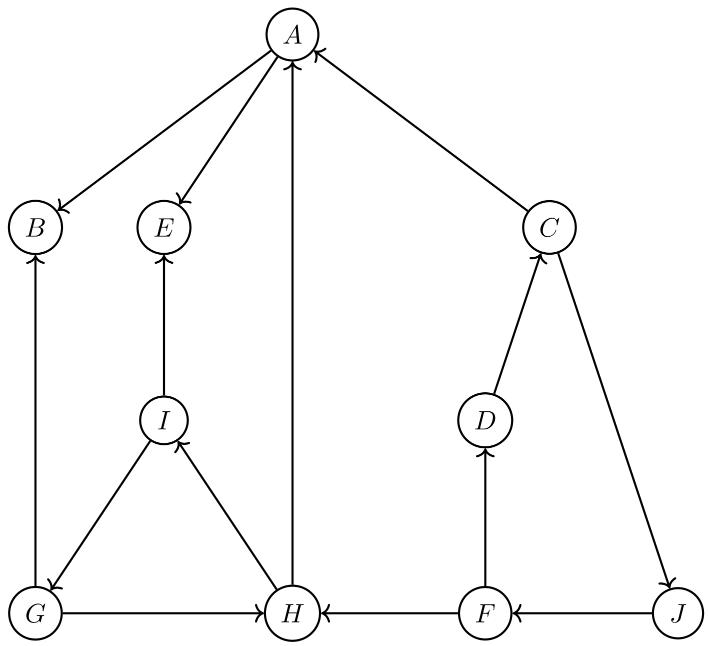
Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Graph Traversal



(a) Recall that given a DFS tree, we can classify edges into one of four types:

- Tree edges are edges in the DFS tree,
- Back edges are edges (u, v) not in the DFS tree where v is the ancestor of u in the DFS tree
- Forward edges are edges (u, v) not in the DFS tree where u is the ancestor of v in the DFS tree
- Cross edges are edges (u, v) not in the DFS tree where u is not the ancestor of v, nor is v the ancestor of u.

For the directed graph above, perform DFS starting from vertex A, breaking ties alphabetically. As you go, label each node with its pre- and post-number, and mark each edge as **T**ree, **B**ack, **F**orward or **C**ross.

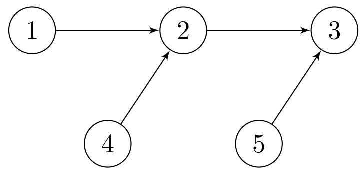
(b) A strongly connected component (SCC) is defined as a subset of vertices in which there exists a path from each vertex to each other vertex. What are the SCCs of the above graph?

(c) Collapse each SCC you found in part (b) into a meta-node, so that you end up with a graph of the SCC meta-nodes. Draw this graph below, and describe its structure.

2 Finding Clusters

We are given a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$, i.e. the vertices are integers in the range 1 to n. For every vertex i we would like to compute the value $m(i)$ defined as follows: $m(i)$ is the smallest j such from which you can reach vertex i. (As a convention, we assume that i is reachable from i.)

Example: Consider the following directed graph with 5 vertices:



The $m(i)$ values are:

- $m(1) = 1$: Only vertex 1 can reach vertex 1 (itself).
- $m(2) = 1$: Vertices 1, 2, and 4 can reach vertex 2. The smallest is 1.
- $m(3) = 1$: Vertices 1, 2, 3, 4, and 5 can all reach vertex 3. The smallest is 1.
- $m(4) = 4$: Only vertex 4 can reach vertex 4 (itself).
- $m(5) = 5$: Only vertex 5 can reach vertex 5 (itself).

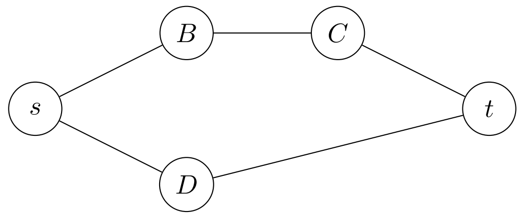
(a) Show that the values $m(1), \ldots, m(n)$ can be computed in $O(|V| + |E|)$ time.

(b) Suppose we instead define $m(i)$ to be the smallest j that can be reached from i, instead of the smallest j from which you can reach i. Can we use the same DFS approach from part (a) If not, what goes wrong, and how can we fix it?

3 Odd Shortest Path

Given an undirected graph $G = (V, E)$ and two vertices $s, t \in V$, find the shortest path from s to t that uses an **odd number of edges**, or report that no such path exists. Your algorithm should run in $O(|V| + |E|)$ time.

Example: Consider the following undirected graph:



The shortest path from s to t overall is $s \to D \to t$ with length 2 (even). However, this path has an **even** number of edges, so it doesn't count!

The shortest **odd-length** path is $s \to B \to C \to t$ with length 3.

4 Bottleneck Spanning Tree

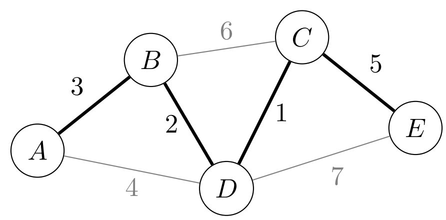
Recall that a spanning tree of a connected, undirected graph $G = (V, E)$ is a subgraph $T = (V, E_T)$ that:

- Contains all vertices of G
- Is a tree (connected and acyclic)
- Uses only edges from E

A **Minimum Spanning Tree (MST)** is a spanning tree that minimizes the *total* weight of all edges:

$$\text{MST minimizes } \sum_{e \in T} w(e)$$

Example: Consider the following weighted graph and its MST (bold edges):



A **Bottleneck Spanning Tree (BST)** is a spanning tree that minimizes the weight of the *heaviest* edge:

$$\text{BST minimizes } \max_{e \in T} w(e)$$

(a) Is every Bottleneck Spanning Tree also a Minimum Spanning Tree? Prove or give a counterexample.

(b) Is every Minimum Spanning Tree also a Bottleneck Spanning Tree? Prove or give a counterexample.