Question 1 (Pre- and Post-Orders, 30 points). Run DFS on the graph below and compute the pre-order and post-order numbers of each vertex. Whenever, the algorithm needs to make a decision about what order to do things in, always use alphabetical order.

The pre- and post-order numbers are notated above.
**Question 2** (DAG-ification, 35 points). Let \( G \) be a finite, directed graph that consists of a single strongly connected component. Suppose that there is an edge \( e \) in \( G \) from a vertex \( v \) to a vertex \( w \) so that removing \( e \) from \( G \) leaves a DAG. Prove that \( e \) is the only edge in \( G \) leaving the vertex \( v \).

Since \( G \) is strongly connected, it cannot have any sinks (as you would be unable to get from that sink to any other vertex). However, since \( G - e \) is a finite DAG it must have a sink. Since \( v \) is the only vertex with fewer outgoing edges in \( G - e \) than in \( G \), it must be the sink. Therefore, \( e \) must be the only outgoing edge from \( v \).
Question 3 (Shapeshifter Maze, 35 points). Tristan is a shapeshifter. He has one form that can survive on land and another that can survive on water. Unfortunately, he is running out of energy to power his transformations and can only transform three more times before recharging. He has a map leading to a leyline that could be used to recharge.

The map is a graph $G$. Each vertex is either on land or under water and can only be survived in the correct form. Each edge takes exactly an hour for Tristan to traverse, and when he reaches a new vertex he can transform (if necessary) into a form suitable to survive there.

Suppose that you are given the graph $G$ along with markings as to which vertices are on land or in the water along with the locations of Tristan’s current location and the location of the leyline. Design an algorithm that computes the minimum number of hours that it will take Tristan to reach the leyline without using more than three transformations. For full credit your algorithm should run in linear time or better.

We create a bigger graph $\mathcal{G}$ whose vertices are pairs $(v, n)$ where $v$ is a vertex in $G$ and $n$ is an integer between 0 and 3. We think of this vertex as being Tristan in location $v$ having transformed exactly $n$ times so far. We have edges in $\mathcal{G}$ from $(v, n)$ to $(w, n)$ if there is an edge from $v$ to $w$ in $G$ and $v$ and $w$ are vertices of the same type (i.e. both land or both water) and from $(v, n)$ to $(w, n+1)$ if there is an edge $(v, w)$ in $G$ but $v$ and $w$ are of different types. We then run BFS on $\mathcal{G}$ starting at $(v_{\text{start}}, 0)$ and return the minimum distance to any of the vertices $(v_{\text{leyline}}, n)$.

It takes $O(|V|)$ time to create all of the vertices of $\mathcal{G}$ and then $O(|E|)$ to create the edges. BFS runs in linear time in the size of $\mathcal{G}$, and so the entire algorithm is linear.

To show correctness, we note that the paths Tristan can take starting from $v_{\text{start}}$ in $G$ while shapeshifting at most 3 times correspond exactly to the paths he can take in $\mathcal{G}$ starting from $(v_{\text{start}}, 0)$. This is because every such path in $G$ will have a corresponding path in $\mathcal{G}$ with the second coordinate of the ending vertex donating the number of times that Tristan will need to shapeshift (in particular, this number increases by 1 whenever he moves from a land square to a water square or visa versa).