1. The following figure shows the previsit and postvisit times for DFS on a directed graph (we did this problem in the last section). Now, find out the strongly connected components for the reverse of this graph.

2. Sticking with (a slight modification of) the same graph, let us now try to run Dijkstra’s algorithm on it. You may also, if you are in the mood for it, forget the edge-weights and run a BFS on it first.
3. You are given a directed graph in which each node \( u \in V \) has a price \( p_u \), which is a positive integer. Define the array \( \text{cost} \) as follows: for each \( u \in V \),

\[
\text{cost}[u] = \text{price of the cheapest node reachable from } u \text{ (including } u)\]

Give a linear time algorithm to find the entire \( \text{cost} \) array for directed acyclic graphs. Now extend your algorithm to general directed graphs.

4. Let us define tree, forward, back and cross edges for BFS in the same way as DFS. Let \((u, v)\) be an edge in the directed graph \( G \). What is the relationship between \( \text{dist}(u) \) and \( \text{dist}(v) \) if \((u, v)\) is a
   a) tree edge  b) cross edge  c) forward edge  d) back edge.

... and the really mean ones

1. Give a linear time algorithm to find an odd-length cycle in a directed graph. First solve this problem assuming the graph is strongly connected and then extend it to a general graph.

2. Let \( G \) be a connected undirected graph. We run DFS on \( G \) starting from a node \( r \) and get the DFS tree \( T \). Bored of DFS, we then run BFS on \( G \) (again starting from \( r \)) and to our extreme frustration, again get the same tree \( T \) also as the BFS tree. Prove that \( G = T \).

3. Your GSI tells you the following algorithm for finding shortest paths in a graph with negative weight edges: add a large enough constant to the weight of all the edges so that they all become positive, and then simply run Dijkstra’s algorithm on the new graph. Prove him/her wrong by giving an example of a graph where this algorithm will fail.

4. Prove that for the array \( \text{prev} \) computed by Dijkstra’s algorithm, the edges \( \{u, \text{prev}[u]\} \) (for all \( u \in V \)) form a tree.

5. Give an example of a small graph with negative weights where Dijkstra’s algorithm fails.