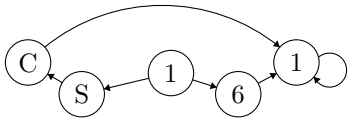
Exercises

Exercises should be completed **on your own**.

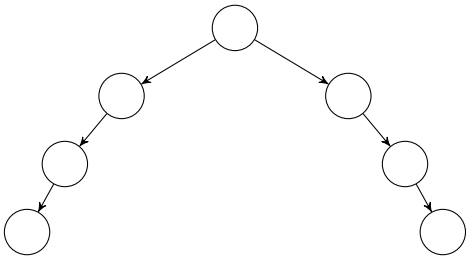
Drawing graphs: You might try `http://madebyevan.com/fsm/` which allows you to draw graphs with your mouse and convert it into LaTeX code:



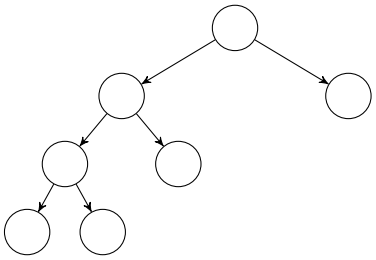
1. **(4 pt.)** For each of the unlabeled binary trees, state whether or not it can be the structure of a red-black tree. If so, color the vertices red or black. If not, state which of the red-black tree invariants (1 to 5) cannot be satisfied and provide an explanation.

   [We are expecting: A YES/NO, and either (1) a valid coloring or (2) a set of violated invariants and a brief explanation. For the colorings, feel free to redraw the trees by hand and upload the image.]

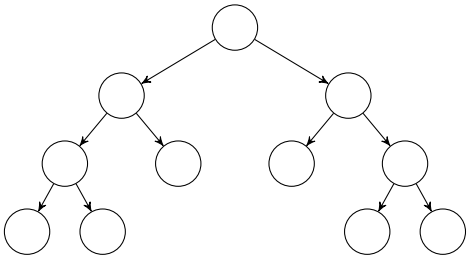
   (a) (1 point)

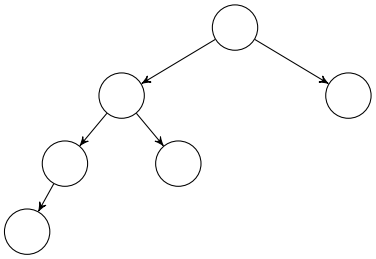
   (c) (1 point)

   (b) (1 point)

   (d) (1 point)

2. **(4 pt.)** Give one example of a directed graph on four vertices, A, B, C, and D, such that both depth-first search and breadth-first search discover the vertices in the same order when started at A. Give one example of a directed graph where DFS and BFS discover the vertices in a different order when started at A. "Discover" means the time that the algorithm first reaches the vertex, referred to as `start_time` during lecture. Assume that both DFS and BFS iterate over outgoing neighbors in alphabetical order.

   [We are expecting a drawing of your graphs and an ordered list of vertices discovered by DFS and BFS.]

Problems

You can collaborate with your classmates about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your solutions yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

---

1. **(Another topological sort algorithm) (7 pt.)** Consider a **complete directed acyclic graph** $G = (V, E)$ where $|V| = n$ and $|E| = \frac{n(n-1)}{2}$. Since the graph is complete, there exists an edge between each pair of the vertices $u, v \in V$ such that either $(u, v) \in E$ or $(v, u) \in E$ but not both (since the graph is acyclic).

   Suppose we define a function `contains(u,v)` which returns true if $(u, v) \in E$ and false if $(v, u) \in E$. Assume that we do not have access to the list of edges and can only access the graph through the `contains` function. (We do have access to the list of vertices though.)

   (a) (1 pt.) Give an asymptotically tight lower bound on the worst-case number of calls to `contains` required to find a topological sort of G.

   [We are expecting: An asymptotic tight lower bound like $\Omega(\ldots)$.]

   (b) (3 pt.) Prove that the bound is tight, i.e., that no algorithm can find a topological sort of G with fewer than that many calls to `contains` in the worst-case.

   [We expecting: A convincing argument.]

   (c) (3 pt.) Describe an algorithm that achieves this bound.

   [We are expecting: Either an English description or pseudocode of the algorithm, and an English justification of why it achives the bound.]

2. **(Gossip modeling) (9 pt.)** Suppose we have a community of n people. We can create a directed graph from this community as follows: the vertices are people, and there is a directed edge from person A to person B if A would forward a rumor to B. Assume that if there is an edge from A to B, then A will always forward any rumor they hear to B. Notice that this relationship isn't symmetric: A might gossip to B but not vice versa. Suppose there are m directed edges total, so $G = (V, E)$ is a graph with n vertices and m edges.

   Define a person P to be *influential* if for all other people A in the community, there is a directed path from P to A in G. Thus, if you tell a rumor to an influential person P, eventually the rumor will reach everybody. You have a rumor that you'd like to spread, but you don't have time to tell more than one person, so you'd like to find an influential person to tell the rumor to.

   In the following questions, assume that G is the directed graph representing the community, and that you have access to G as an array of adjacency lists: for each vertex v, in $O(1)$ time you can get a pointer to the head of the linked lists v`.outgoing_neighbors` and v`.incoming_neighbors`. Notice that G is not necessarily acyclic. In your answers, you may appeal to any statements we have seen in class, in the notes, or in CLRS.

   (a) (1 pt.) Show that all influential people in G are in the same strongly connected component, and that everyone in this strongly connected component is influential.

   [We are expecting: A short but formal proof.]

(b) (5 pt.) Suppose that an influential person exists. Give an algorithm that, given G, finds an influential person in $O(n+m)$-time.

[We are expecting: Either an English description or pseudocode of the algorithm, a formal proof of correctness, and an English justification of why it takes $O(n+m)$-time.]

(c) (3 pt.) Suppose that you don't know whether or not an influential person exists. Use your algorithm from part (b) to give an algorithm that, given G, either finds an influential person in time $O(n+m)$ if there is one, or else returns "no influential person."

[We are expecting: Either an English description or pseudocode of the algorithm.]