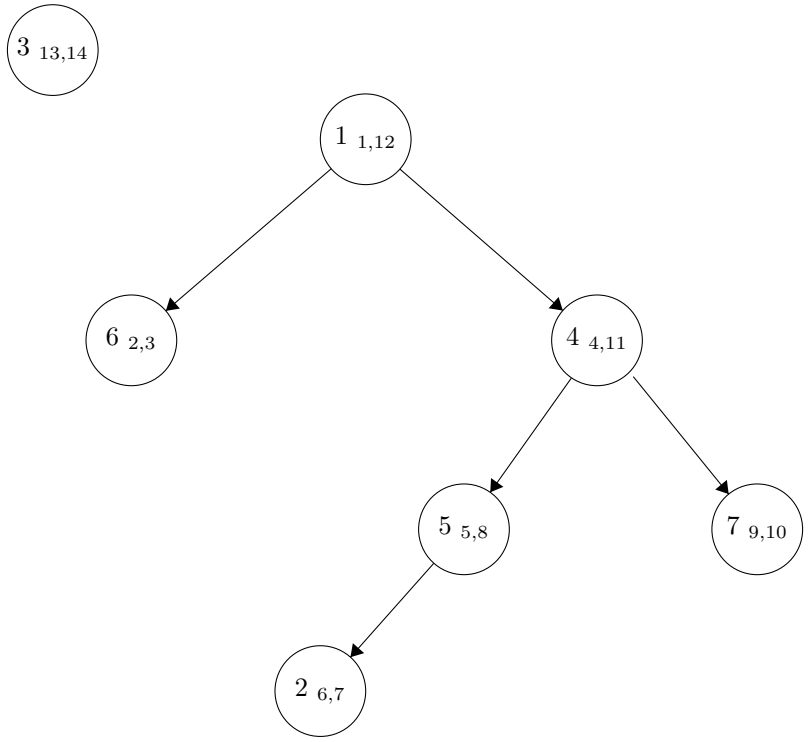
By: Eric Koehli, Jacob Chmura, Conor Vedova

Question 1. [1 MARK]

Let $G = (V, E)$ be a directed graph and let $V = \{1, 2, 3, 4, 5, 6, 7\}$.

a.



In the DFS of G above, the discovery time d[u] and finish time f[u] of each node are in the subscripts.

b. How many back edges, forward edges, and cross-edges are found by the above DFS?

In the DFS of G, there are:

- Back edges: 0
- Forward edges: 2
- Cross edges: 4

c. Suppose the graph G above represents seven courses and their prerequisites. For example, $A(1) = 6, 4, 7$ means that course 1 is a prerequisite for (i.e., it must be taken before) courses 6, 4 and 7; and $A(6) = \texttt{NIL}$ means that course 6 is not a prerequisite for any course.

Using Part (b) above and a theorem that we learned in class, prove that it is possible to take all the courses in a sequential order that satisfies all the prerequisite requirements. State the theorem that you use in your argument; do not give a specific course order here.

Proof. Since the DFS of G produced no back edges, it follows that there are no cycles (by the negation of the theorem proven in class that was an application of the White Path Theorem). Thus, since there are

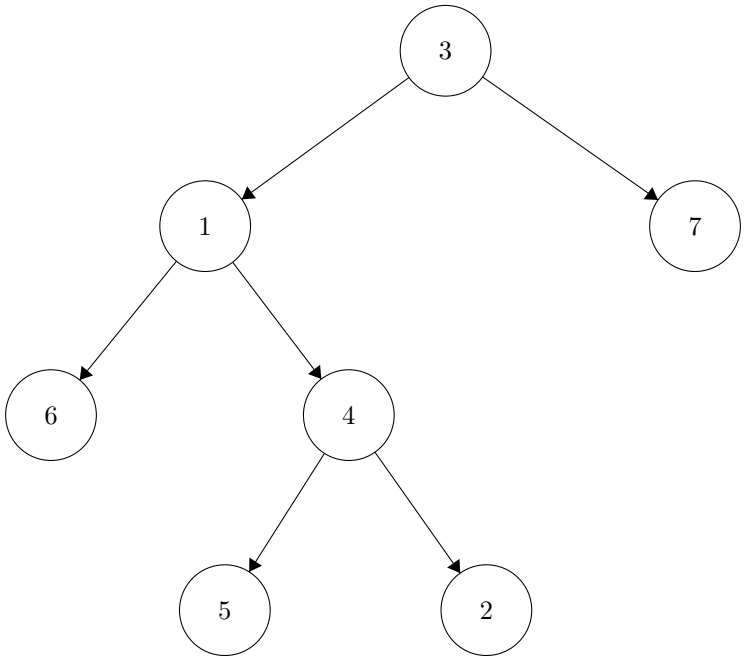
no cycles in the DFS of G, we cannot have a preprequisite deadlock. Since we cannot have a prerequisite deadlock, there exists at least one such path where it is possible to take all the courses in a sequential order that satisfies all the prerequisite requirements. □

d. Now list the courses in an order they can be taken without violating any prerequisite. To do so you must use your DFS of Part (a) and an algorithm that we covered in a tutorial.

We will use the DFS of G from Part (a) and list the nodes from left to right such that for all $(u, v) \in E$, if `f[u]` $>$ `f[v]`, then course u will be taken before course v. In other words, we are just going to order the courses above by exploration completion time in descending order. Thus, the course order from Part (a) is as follows,

$$3, 1, 4, 7, 5, 2, 6$$

e. Draw a Breadth-First Search tree of G that starts at node 3 and explores the edges in the order of appearance in the above adjacency lists.



Answered by: Eric
Verified by: Jacob, Conor

Question 2. [1 MARK]

Let $G = (V, E)$ be an undirected graph. Each node of the graph G represents x_i for some i. Create an edge for each equality constraint. Perform DFS on as many nodes as needed in order to fully traverse the graph [1]. While performing a single DFS operation, assign every node that you arrive at the value of the starting node, the value of the starting node for x_j should be j. So, the DFS operation results in a set of nodes with value j. After performing all the necessary DFS operations, simply loop through the inequality constraints, and see if the value of the two nodes being compared are equal. If they are equal, return `NIL`. Else, `continue`. Then, return the values of the nodes.

The time complexity of this algorithm is $\mathcal{O}(m + n)$ because there are at most n DFS operations on one node each, or, on the other extreme, there are at most m DFS comparisons (on a fully connected graph). Otherwise, the number of steps is less than or equal to $m + n$. Now, the number of steps of the ending loop is at most $m - i$ where i is the number of equality constraints. So, in total, there are no more than $m - i + i + n = m + n$ steps. i.e., the algorithm is $\mathcal{O}(m + n)$.

Answered by: Conor
Verified by: Jacob, Eric

[1] The graph may not be fully connected

Question 3. [1 MARK]

Let $G = (V, E)$ be an undirected, connected graph, and w: $E \to R$ be an edge weight function such that edges have distinct weights. Suppose that for every edge $e \in E$ there is a cycle of G that contains e. Let e_{max} be the edge with the maximum weight in G. Prove that no minimum spanning tree of G contains e_{max}.

Proof. Suppose for contradiction that e_{max} belongs to a Minimum Spanning tree T. Then deleting edge e_{max} will result in two different subtrees S_1, S_2, with each end of e_{max} in a different subtree. By the assumption that for every edge there is a cycle that contains that edge, there must be a cycle C that contains e_{max}. Therefore there must be some edge e' that reconnects subtrees S_1, S_2. By our assumption that edges have distinct weights, and the definition of e_{max}, this edge e' must have a smaller weight then the weight of e_{max}, so we have a spanning tree T', with less total edge weight than T. $\to\leftarrow$ CONTRADICTION.

Therefore, T is not a minimum spanning tree, and e_{max} cannot belong to a minimum spanning tree of G. $\square$

Answered by: Jacob
Verified by: Conor, Eric