1. Describe a linear time algorithm that takes as input a directed graph and partitions its edges into two subsets such that each of the two subsets forms a directed acyclic graph.

   Solution: Arbitrarily label the vertices from 1 to n. For each edge $v \to w$, add it to the first set if $v < w$, and add it to the second set otherwise. The labels of the vertices form a topological order for the first set, and the reverse order forms a topological order on the second set. So each set of edges forms a DAG.

   Alternative solution: do a DFS traversal and classify the edges as DFS edges, forward edges, back-edges or cross-edges. Forward and DFS edges should be in the same set and back-edges should be in the other set. The cross-edges can be placed in either set.

2. Suppose that G is a directed acyclic graph with n vertices, with the special property that there is only one way to place the vertices of G into a topological order.

   (a) How many edges can the transitive closure of G have, as a function of n?

   Solution: First, we show that there is only one way to place the vertices of G into a topological order if and only if there is a simple path that goes through all the vertices: ($\Leftarrow$) The path determines the topological order of the vertices: if the path is $v_1 \to v_2 \to \cdots \to v_n$, then v_1 must be before v_2, which must be before v_3, etc. There is no other possible topological order. ($\Rightarrow$) Suppose that there are two possible topological orders. In particular there must be two vertices v and w such that v appears before w in one case, and v appears after w in the other case. For this to be possible, there can't be a path between v and w.

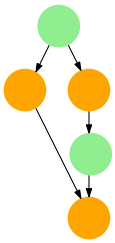
   Since a graph with this special property has a path through all its vertices in topological order, every vertex can reach all later vertices. So the transitive closure of the graph has an edge $v \to w$ if v is earlier than w, for a total of $1 + 2 + \cdots + (n-1) = (n-1)n/2$ edges.

   (b) How many edges can the transitive reduction of G have, as a function of n?

   Solution: Since there is a simple path through all the vertices, that path determines the topological order. All other edges are redundant, which means that the transitive reduction must have $n - 1$ edges.

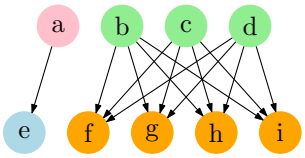
3. (For 163 students:) If G is a directed acyclic graph without any redundant edges (it is its own transitive reduction), is it always possible to assign the vertices to layers in such a way that each edge goes downwards from one layer to the next consecutive layer? For this problem, there is no limit on the number of vertices per layer. If the answer is yes, explain why. If the answer is no, find a counterexample.

Solution: No, here's a counterexample.

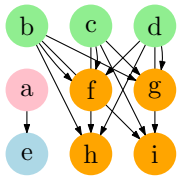


(For 265 students:) Find an example of a directed acyclic graph for which the Coffman-Graham algorithm, with three vertices per layer, can use more layers than the optimum layering. Show both the optimum layering and the layering found by the algorithm.

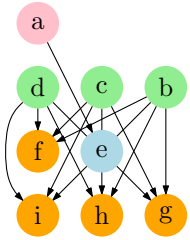
Solution:



Here's an example, courtesy of Balint.



The optimal layering contains three layers.



But one possible output of Coffman-Graham is the following, assuming that the order of the nodes in the list of nodes with no predecessors was a, b, c, d. However, if a appears after b, c, d, we get an optimal layering again.

4. Suppose we are given as input a directed graph (not necessarily acyclic) and we want to add one edge to the graph so that the augmented graph is strongly connected. Describe a linear time algorithm for finding such an edge, if it exists. (Hint: use the condensation, and look at the numbers of incoming and outgoing edges of the nodes in the condensation.)

Solution: The condensation is a DAG, and a DAG can be made strongly connected by the addition of a single edge if and only if there is exactly one source node and one sink node that are distinct from each other. (A source is a node with no incoming edges, and a sink is a node with no outgoing edges.)

Proof: ($\Rightarrow$) A DAG must have at least one source and one sink (otherwise it would have a cycle). If the DAG has only one source and only one sink and the two are equal, then the graph is already strongly connected. So assume that this is not the case. To make the DAG strongly connected, we must add enough edges to connect every sink to every source, which is at least the minimum of the number of sinks and the number of sources. So if the DAG can be made strongly connected by adding just one edge, there can only be one source and one sink.
($\Leftarrow$) Let v be a vertex of the DAG that is neither the source nor the sink. We claim that v is on a path from the source to the sink. Since v has an outgoing edge, follow that edge to the next vertex w. If w is not the sink, it also has an outgoing edge. By following outgoing edges, we will eventually reach the sink. Thus there is a path from v to the sink. Similarly, by following incoming edges starting at v and working backwards, we will eventually reach the source, which mean we can find a path from the source to v. So every vertex in a path from the source to the sink. Now, if we add an edge from the sink to the source, every vertex can reach every other vertex. In other words, the DAG with that additional edge is strongly connected. This concludes the proof.

Here's the algorithm:

1. Compute the condensation of the input graph.
2. Compute the set of sources and the set of sinks of the condensation.
3. If there is exactly one source and one sink:
4. if the source is different from the sink:
5. choose a vertex v in the sink and a vertex w in the source
6. return $v \to w$.
7. return "the graph is already strongly connected"
8. Return "no such edge."