We run the same algorithm as in the text, with the following small modification. If in every iteration we find a node with no incoming edges, then we will succeed in producing a topological ordering. If in some iteration, it transpires that every node has at least one incoming edge, then we saw in the text a proof that G must contain a cycle. We can turn this proof into an algorithm with running time $O(n)$ to actually find a cycle: we repeatedly follow an edge into the node we're currently at (choosing the first one on the adjacency list of incoming edges, so that this can run in constant time per node). Since every node has an incoming edge, we can do this repeatedly until we re-visit a node v for the first time. The set of nodes encountered between these two successive visits is a cycle C (traversed in the reverse direction).

[1] ex947.5.72