TUTORIAL 10: Topological sort

Instance: Directed acyclic graph (DAG)
Question: What is an ordering of vertices 1, ..., |V| such that for every
edge (u,v), u appears before v in the ordering?

Algorithm:
```
TOPOLOGICAL-SORT(G)
1  call DFS(G) to compute finishing times f[v] for each vertex v
2  as each vertex is finished, insert it onto the front of a linked list
3  return the linked list of vertices
```

Example of topological sort:
Demonstrate an example of topological sort using a directed graph with
about 6 vertices. I've drawn one possible example to show. (The dots
represent the pointed end of the arrow.)

```
        a
      /   \
     .     .
     b     c
     \   /   \
      . .     .
       d  .-   e
        \    /
         .  .
          f
```

Running time of topological sort: Theta(n + m)
Why? Depth first search takes Theta(n + m) time in the worst case, and
inserting into the front of a linked list takes Theta(1) time.

Edge classification:

There are four types of edges in a graph G. Run DFS on G and consider the
resulting depth-first forest:

1) Tree edges = edges in the depth-first forest.

2) Back edges = nontree edges (u, v) in G connecting a vertex u to an ancestor
v in a depth-first tree.

3) Forward edges = nontree edges (u, v) connecting a vertex u to a descendant
v in a depth-first tree.

4) Cross edges = all other edges.
a) nontree edges (u, v) connecting vertices in the same depth-first tree,
   as long as one vertex is not an ancestor of the other, and
b) nontree edges (u, v) connecting vertices in different depth-first trees.

Draw a picture to show the different types of edges

Notice: If there is a back edge, there must be a cycle in G
Why? If there is a back edge (u,v), then vertex v is an ancestor of vertex u
     in the depth-first forest. Thus, there is a path from v to u and an
     edge from u to v.
     -> there is a cycle

Theorem 22.12
TOPOLOGICAL-SORT(G) produces a topological sort of a directed acyclic graph G.

Proof:

First run DFS on G to determine the finishing time for each vertex.

Claim: for any u,v \in V, if n \neq v and (u,v) \in E, then f[v]<f[u].

Proof of claim:
Consider when the edge (u,v) is explored by the DFS.

i) If v is gray, then v is an ancestor of u
   Thus (u,v) is a back edge
   -> G has a cycle
   But G is acyclic, so v cannot be gray

ii) If v is white, it becomes a descendant of u, and so f[v] < f[u]
    i.e. we finish examining the descendants of v before those of u

iii) If v is black, it is already finished.
     -> f[v] is already set
     We are still exploring descendants of u, so f[u] is not set
     -> f[v] < f[u]

So the claim is true: if (u,v) \in E, f[v] < f[u].

TOPOLOGICAL-SORT(G) places vertices in a linked list from highest to lowest
finishing time. Therefore, if (u,v) \in E, u will be before v in the list.

Theorem 22.9

In a depth-first forest of a (directed or undirected) graph G = (V, E),
vertex v is a descendant of vertex u if and only if at the time d[u] that
the DFS discovers u, vertex v can be reached from u along a path consisting
entirely of white vertices.

Proof:

Direction #1: If vertex v can be reached from u along a path consisting
entirely of white vertices at time d[u], then vertex v is a descendant
of vertex u in the depth-first forest.

We will prove direction #1 using a proof by contradiction:
Without loss of generality, let v be the first vertex in the path of white
vertices which does not become a descendant of u.

Let w be the predecessor of v in the path
w is either a descendant of u or u itself (f[w] \leq f[u])
v must be discovered after u, since v is still white (d[v] > d[u])
v must be discovered before w is finished (d[v] < f[w])
Thus: d[u] < d[v] < f[w] \leq f[u]

But: if v is discovered after u, v must be finished before u (f[v] < f[u])
Thus: d[u] < d[v] < f[v] < f[u]
-> v is a descendant of u

Direction #2: If vertex v is a descendant of vertex u in the depth-first
forest, then vertex v can be reached from u along a path consisting entirely
of white vertices at time d[u].

We will prove direction #2 using a direct proof:
Let v be a descendant of u in the depth-first tree.
Let w be any vertex on the path between u and v in the depth-first tree.
-> w is a descendant of u
-> d[w] > d[u]
-> w is white at time d[u]

Lemma 22.11

A directed graph G is acyclic iff a DFS of G yields no back edges.

Proof:

Direction #1: If directed graph G is acyclic, then a DFS of G yields no
back edges.
Contrapositive: If a DFS of G yields some back edge, then the directed
graph G contains a cycle.

We showed this earlier.

Direction #2: If the a DFS of G yields no back edges, then the directed graph
G is acyclic.
Contrapositive: If the graph contains a cycle, then the DFS of G yields a
back edge.

Proof of the contrapositive:
Suppose that G contains a cycle c.
Let v be the first vertex discovered in c during the DFS.

Let u be the parent of v in the cycle.
Since v is the first to be discovered in c, the path from v to u is formed
by all white vertices.

By the white-path theorem (Theorem 22.9), vertex u is a descendant of vertex
v iff at time d[v], vertex v can be reached from u along a path consisting
entirely of white vertices.
-> u is a descendant of v
-> edge (u,v) is a back edge.