16.4.2

DFS and cycle detection:
Topological sorting using DFS
Cycles in graphs

**Question:** Given an **undirected** graph how do we check whether it has a cycle and output one if it has one?

**Question:** Given an **directed** graph how do we check whether it has a cycle and output one if it has one?
Cycles in graphs

**Question:** Given an **undirected** graph how do we check whether it has a cycle and output one if it has one?

**Question:** Given an **directed** graph how do we check whether it has a cycle and output one if it has one?
Cycle detection in directed graph using topological sorting

Question

Given $G$, is it a **DAG**?

If it is, compute a topological sort.
If it fails, then output the cycle $C$. 
Topological sort a graph using DFS... And detect a cycle in the process

**DFS** based algorithm:

1. Compute **DFS(\(G\))**
2. If there is a back edge \(e = (v, u)\) then \(G\) is not a **DAG**. Output cycle \(C\) formed by path from \(u\) to \(v\) in \(T\) plus edge \((v, u)\).
3. Otherwise output nodes in decreasing post-visit order. **Note:** no need to sort, **DFS(\(G\))** can output nodes in this order.

Computes topological ordering of the vertices.

Algorithm runs in \(O(n + m)\) time.
Correctness is not so obvious. See next two propositions.
Topological sort a graph using DFS...

And detect a cycle in the process

**DFS** based algorithm:

1. Compute **DFS**$(G)$

2. If there is a back edge $e = (v, u)$ then $G$ is not a **DAG**. Output cycle $C$ formed by path from $u$ to $v$ in $T$ plus edge $(v, u)$.

3. Otherwise output nodes in decreasing post-visit order. **Note**: no need to sort, **DFS**$(G)$ can output nodes in this order.

Computes topological ordering of the vertices.

Algorithm runs in $O(n + m)$ time.

Correctness is not so obvious. See next two propositions.
Topological sort a graph using DFS...

And detect a cycle in the process

DFS based algorithm:

1. Compute $\text{DFS}(G)$
2. If there is a back edge $e = (v, u)$ then $G$ is not a DAG. Output cycle $C$ formed by path from $u$ to $v$ in $T$ plus edge $(v, u)$.
3. Otherwise output nodes in decreasing post-visit order. Note: no need to sort, $\text{DFS}(G)$ can output nodes in this order.

Computes topological ordering of the vertices.

Algorithm runs in $O(n + m)$ time.  
Correctness is not so obvious. See next two propositions.
Back edge and Cycles

Proposition

\( G \) has a cycle \( \iff \) there is a back-edge in \( \text{DFS}(G) \).

Proof.

If: \((u, v)\) is a back edge implies there is a cycle \(C\) consisting of the path from \(v\) to \(u\) in \(\text{DFS}\) search tree and the edge \((u, v)\).

Only if: Suppose there is a cycle \(C = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k \rightarrow v_1\). Let \(v_i\) be first node in \(C\) visited in \(\text{DFS}\). All other nodes in \(C\) are descendants of \(v_i\) since they are reachable from \(v_i\). Therefore, \((v_{i-1}, v_i)\) (or \((v_k, v_1)\) if \(i = 1\)) is a back edge.
Decreasing post numbering is valid

Proposition

If $G$ is a DAG and $\text{post}(v) > \text{post}(u)$, then $(u \rightarrow v)$ is not in $G$.

Proof.

Assume $\text{post}(u) < \text{post}(v)$ and $(u \rightarrow v)$ is an edge in $G$. One of two holds:

- Case 1: $[\text{pre}(u), \text{post}(u)]$ is contained in $[\text{pre}(v), \text{post}(v)]$.
- Case 2: $[\text{pre}(u), \text{post}(u)]$ is disjoint from $[\text{pre}(v), \text{post}(v)]$. 
Decreasing post numbering is valid

**Proposition**

If $G$ is a DAG and $\text{post}(v) > \text{post}(u)$, then $(u \rightarrow v)$ is not in $G$.

**Proof.**

Assume $\text{post}(u) < \text{post}(v)$ and $(u \rightarrow v)$ is an edge in $G$. One of two holds:

- **Case 1:** $[\text{pre}(u), \text{post}(u)]$ is contained in $[\text{pre}(v), \text{post}(v)]$.
- **Case 2:** $[\text{pre}(u), \text{post}(u)]$ is disjoint from $[\text{pre}(v), \text{post}(v)]$. 

□
Decreasing post numbering is valid

Proposition

If $G$ is a DAG and $\text{post}(v) > \text{post}(u)$, then $(u \rightarrow v)$ is not in $G$.

Proof.

Assume $\text{post}(u) < \text{post}(v)$ and $(u \rightarrow v)$ is an edge in $G$. One of two holds:

- **Case 1:** $[\text{pre}(u), \text{post}(u)]$ is contained in $[\text{pre}(v), \text{post}(v)]$. Implies that $u$ is explored during $\text{DFS}(v)$ and hence is a descendent of $v$. Edge $(u, v)$ implies a cycle in $G$ but $G$ is assumed to be DAG!

- **Case 2:** $[\text{pre}(u), \text{post}(u)]$ is disjoint from $[\text{pre}(v), \text{post}(v)]$. This cannot happen since $v$ would be explored from $u$. 

□
Translation

We just proved:

**Proposition**

If \( G \) is a **DAG** and \( \text{post}(v) > \text{post}(u) \), then \((u \rightarrow v)\) is not in \( G \).

\[\Rightarrow \text{ sort the vertices of a **DAG** by decreasing post numbering in decreasing order, then this numbering is valid.}\]
Topological sorting

Theorem

\( G = (V, E) \): Graph with \( n \) vertices and \( m \) edges.

Compute a topological sorting of \( G \) using DFS in \( O(n + m) \) time.

That is, compute a numbering \( \pi: V \rightarrow \{1, 2, \ldots, n\} \), such that

\[(u \rightarrow v) \in E(G) \iff \pi(u) < \pi(v).\]
THE END

... (for now)