Last class:

DAG = Directed acyclic graph

Topological ordering = order vertices of a DAG
so that all edges go left → right
(low → high)

Key: for a DAG, its DFS tree has no
backedges, hence for all other edges \( \overrightarrow{vw} \),
\( \text{Post}(v) > \text{post}(w) \)

Topological sorting alg:
Run DFS & order by \( V \) Post #.

Example:
Source vertex = vertex with no incoming edges.
Sink vertex = no outgoing edges.

DAG has $\geq 1$ source & $\geq 1$ sink

1st vertex in top ordering last must be a source
must be a source (might be others)

Alternative topological sorting algorithm:
1. Find a sink, output it, delete it.
2. Repeat (1) until the graph is empty.
(can do with source instead of sink)
Connectivity in \textit{directed} graphs:

Vertices \( v \) \& \( w \) are strongly connected if there is a path \( v \to w \) \& \( w \to v \).

\( \text{SCC} = \text{maximal set of strongly connected vertices} \)

\textbf{Example:}

\( \text{SCCS:} \ T,A,B,E,F,G,H,I,J,K,L \)

Think of meta-vertex for each SCC & edge \( S \to S' \) if some \( w \in S \) \& \( z \in S' \) has edge \( wz \)

Then:\n
\( \text{It's a \textit{DAG!}} \)
Property: Every directed graph is a DAG of its SCCs.

Why? If path from $S \rightarrow S'$ & path $S' \rightarrow S$ then $SUS$ is a SCC
So $S & S'$ are not maximal sets.

hence the meta-graph has no cycles.

Our goal: find SCCs & find topological ordering of SCCs.

High-level approach: Find sink SCC, output it, delete it, & repeat
How to find a sink SCC?

If we find a vertex \( v \in \text{sink SCC} \) then run \( \text{Explore}(v) \), this will visit all vertices in this sink SCC & no other vertices.

(Example: if \( v \) is \( H, I, J, K \) or \( L \), then we visit these \& nothing else.)

How do we find a vertex in a sink SCC?

In topological ordering of a DAG vertex with lowest postorder \# is at the end so it's a sink.

In a general directed graph, is the vertex with lowest post \# guaranteed to lie in a sink SCC?

No, consider: B \rightarrow A \rightarrow C

DPS: tree

\[ \begin{align*}
& A & \rightarrow & 1, 6 \\
& B & \rightarrow & \text{tree}, \quad 2, 3 \\
& C & \rightarrow & 4, 5 \\
& \text{SCC's:} & A, B \rightarrow & C
\end{align*} \]
But notice that A has the highest post & it lies in a source SCC.

Key lemma: In a general directed graph, for any DFS tree, vertex \( v \) with highest postorder \( \# \) lies in a source SCC.

So we can get a vertex in a source SCC but we need a vertex in a sink SCC.

Flip the graph.

For \( G = (V,E) \), let \( G^R = (V,E^R) \)

where \( E^R = \{ vw : \overrightarrow{vw} \in E \} \)

So reverse every edge.

Source SCC in \( G \) = Sink SCC in \( G^R \)
Sink SCC in \( G \) = Source SCC in \( G \).
**Scc algorithm:**

For input $G = (V, E)$,

2. Run DFS on $G^R$.
3. Order $V$ by decreasing Post #1 from step 2.
4. Run the (undirected) connected component algorithm on directed $G$ (with $V$ ordered as in 3).

**DFS-cc(\(G\))**:

for all $v$, visited($v$) = False
cc = 0
for all $w \in V$ (ordered by )
    if not visited($w$) then $\text{cc}++$
        Explore($w$)

**Explore($w$)**:

visited($w$) = True
ccnum($w$) = cc
for all $\ell : \overrightarrow{w\ell} \in E$:
    if not visited($\ell$) then Explore($\ell$)

**Running time**: $O(n+m)$.
Proof of key Lemma: (vertex $v^*$ with highest Post # lies in a source SCC)

Claim 1: if $S$ & $S'$ are SCCs & an edge $\overline{YZE}E$ where YES, ZES, then max Post # in $S > max$ Post in $S'$.

hence can topologically sort the SCCs by the max Post # in each SCC.

So SCC with max Post # will be 1st & hence is a source SCC.

So the vertex $v^*$ with max Post # will be in this source SCC.
Proof of claim:

There is an edge $S \rightarrow S'$ so there is a path $S \rightarrow S'$ and hence there is no path $S' \rightarrow S$.

Let $z$ be the 1st vertex in $SUS'$ visited by DFS.

If $z \in S$ then all of $SUS'$ is reachable from $z$ so all of $SUS'$ is in $z$'s subtree in the DFS tree. Hence, $\text{post}(z) > \text{post}(y)$ for $y \in SUS'-zz$.

So $z \in S$ has max post #.

If $z \in S$, then we see all of $S'$ before seeing any of $S$. So $\text{post}(s') > \text{post}(s)$.

A finish explaining.