Last class:

DAG = directed acyclic graph

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

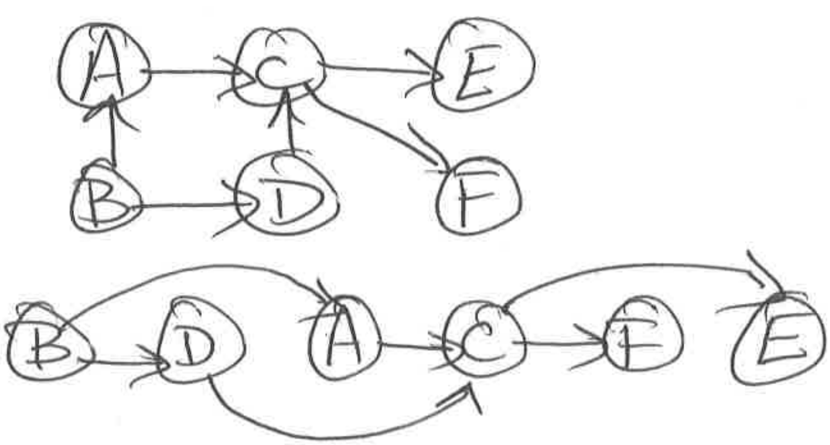
Key: for a DAG, its DFS tree has no backedges, hence for all other edges $\overrightarrow{vw}$:

$$\text{post}(v) > \text{post}(w)$$

Topological sorting alg.:

Run DFS & order by $\downarrow$ post #.

Example:



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

DAG has ≥ 1 source & ≥ 1 sink

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

last must be a sink.

Alternative topological sorting algorithm:

1. Find a sink, output it, delete it.
2. Repeat (1) until the graph is empty.

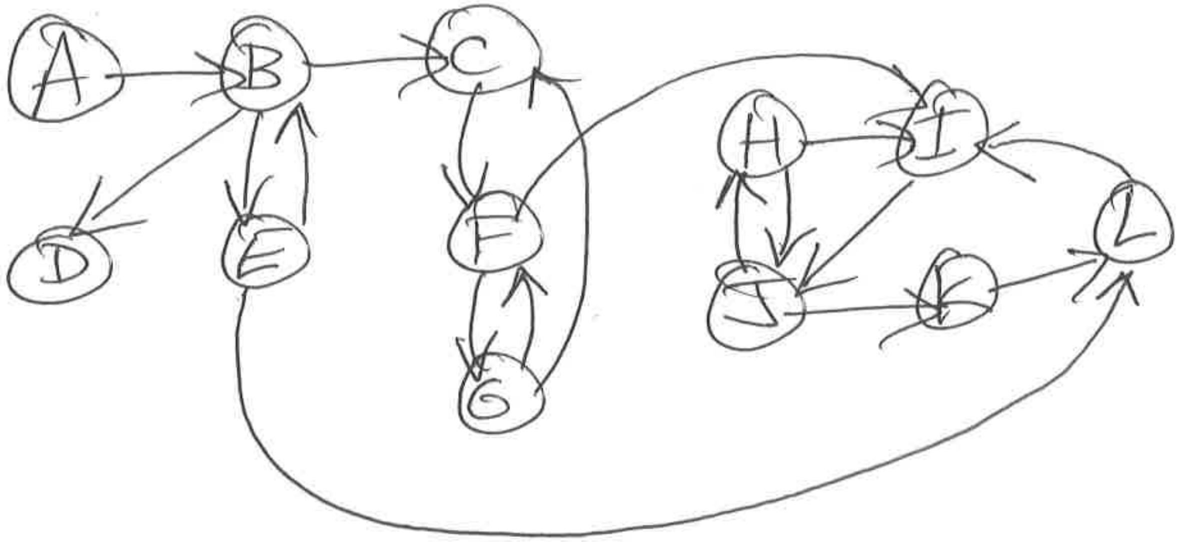
(can do with source instead of sink)

Connectivity in directed graphs:

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

SCC = maximal set of strongly connected vertices.

Example:



SCCs: {A}, {B,E}, {C,F,G}, {D}, {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 $\overrightarrow{wz}$

Then:

It's a DAG!

Property: Every directed graph is a DAG of its SCC's.

Why? If path from $S \rightsquigarrow S'$ & path $S' \rightsquigarrow S$ then $S \cup S'$ is a SCC, so S & S' are not maximal sets. hence, the meta-graph has no cycles.

Our goal: find SCC's & find topological ordering of SCC's.

High-level approach: Find sink SCC, output it, delete it, & repeat

How to find a sink SCC?

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

(Earlier Example: if v is $\underbrace{H, I, J, K \text{ or } L}$, then we visit these ↑ & nothing else.)

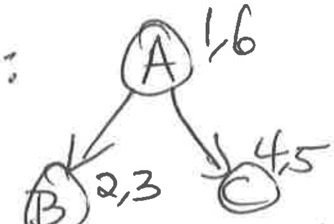
How do we find a vertex in a sink SCC?

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

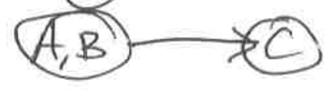
In a general directed graph, is the vertex with lowest post # guaranteed to lie in a sink SCC?

No, consider:

DFS tree:



SCCs:



but B has lowest post #.

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 $E^R=\{\overrightarrow{vw} : \overrightarrow{wv}\in E\}$

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 $G=(V,E)$,

1. Construct G^R.
2. Run DFS on G^R.
3. Order V by decreasing post # 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 [cc++ ; Explore(w)]

Explore(w):

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

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 SCC's

& an edge $\vec{yz} \in E$ where $y \in S$, $z \in S'$,

then max post # in S > max post in S'

hence can topologically sort the SCC's 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 $S \to S'$ so there is a path $S \rightsquigarrow S'$ & hence there is no path $S' \rightsquigarrow S$.

Let z be the 1st vertex in $S \cup S'$ visited by DFS.

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

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

If $z \in S'$, then we see all of S' & finish exploring S' before seeing any of S. So $post(S') < post(S)$. ∎