DFS = Depth first search

Run DFS on undirected graphs to get connected components.

Consider directed graphs.

Look at DFS with pre- & postorder numbering.

DFS(G):

for all $v \in V$, visited(v) = FALSE

clock = 1

for all $v \in V$

if not visited(v) then Explore(v)

Explore(z):

visited(z) = TRUE

pre(z) = clock

clock++

for each $(z, w) \in E$:

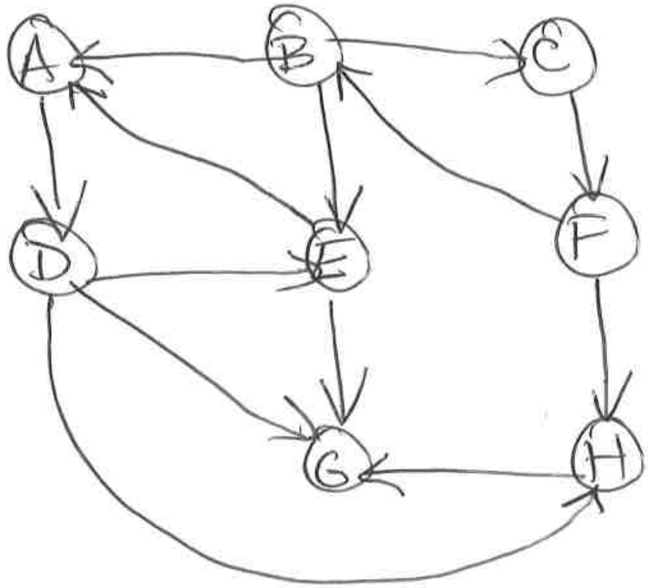
if not visited(w) then Explore(w)

post(z) = clock

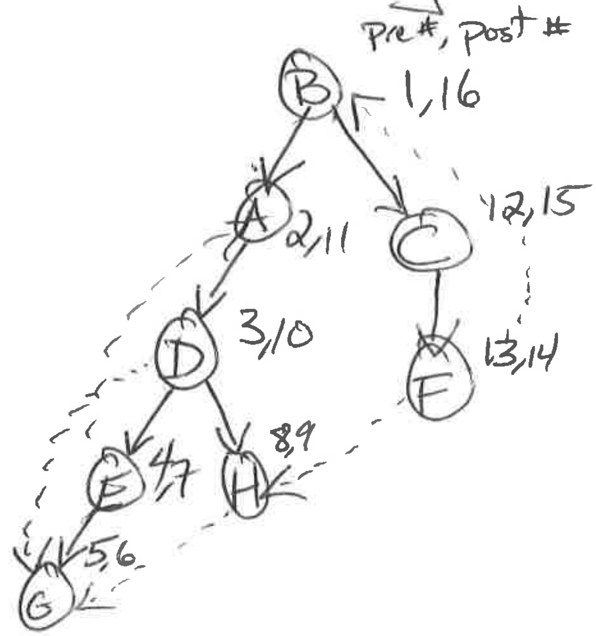
clock++

Running time: $O(|V| + |E|) = O(n+m)$ $\quad n = |V|$, $m = |E|$

Example:



Run DFS starting at B



types of edges: $v \to w$

tree edges $post(v) > post(w)$

back edges – example $E \to A$, $F \to B$

$post(w) > post(v)$

forward edges – $D \to G$

$post(v) > post(w)$

cross edges – $F \to H$, $H \to G$

$post(v) > post(w)$

G has a cycle iff its DFS tree has a back edge

DAG = directed acyclic graph

Topologically sorting a DAG
= order vertices so that all edges go lower → higher (left → right)

For DFS on a DAG,
NO back edges
So for every edge $v \to w$,
$post(v) > post(w)$

Hence, topological sorting algorithm = sort vertices by ↓ postorder #

In a DAG,
lowest post # is a source = no incoming edges
highest post # is a sink = no outgoing edges

Alternative algorithm:

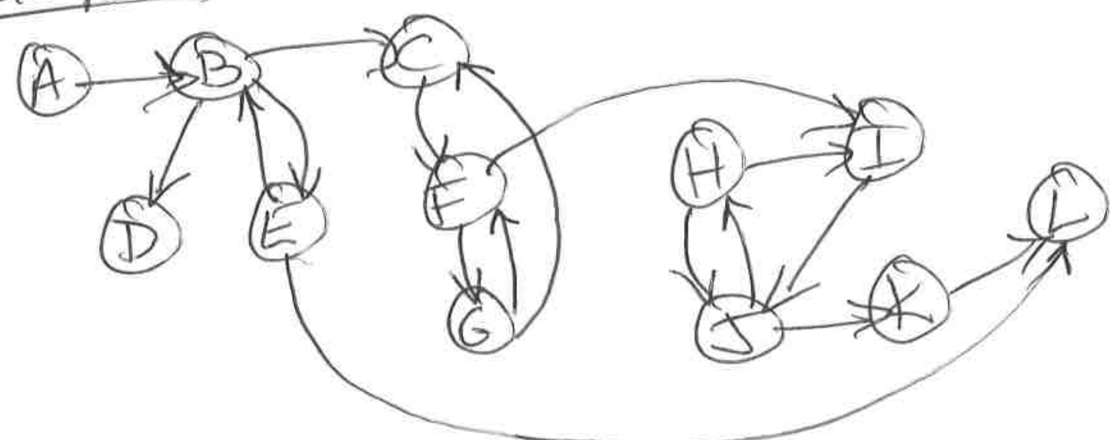
(1) Find a sink, output it, & delete it
(2) Repeat (1) until empty graph.

For a general directed graph $G=(V,E)$,
↖ may have cycles

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

SCC = maximal set of strongly connected vertices
↖ strongly connected components

Example:



SCCs are $\{A\}, \{B,E\}, \{C,F,G\}, \{D\}, \{H,I,J,K,L\}$

Make a meta vertex for each SCC
add edge from SCC S to S'
if some $v \in S$ & $w \in S'$ have edge $v \to w$.

The metagraph on SCC's is a DAG.

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

We'll find the SCC's & the DAG on SCC's.

Approach: Find sink SCC, output it, remove it & repeat

How to find a sink SCC?

Take a vertex v in a sink SCC S.

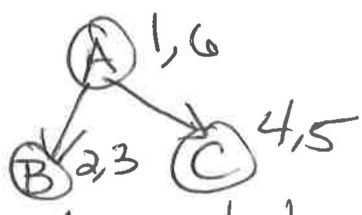
Run Explore(v) then we visit S & nothing else.

How to find v in sink SCC?

Maybe vertex with lowest post #?

NO example:

DFS from A:



B has lowest post # but $\{A, B\}$ is not a sink SCC.

<u>Lemma</u>: Vertex with highest post # lies in a <u>source</u> SCC.

How to get a vertex in a <u>sink</u> SCC?

Look at G^R = reverse of G.

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

where $E^R = \{\overrightarrow{wv} : \overrightarrow{vw} \in E\}$ = reverse of every $e \in E$

source SCC in G = sink SCC in G^R

sink SCC in G = source SCC in G^R

<u>SCC algorithm</u>:

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

1) Construct G^R
2) Run DFS on G^R
3) Order V by ↓ post # from step (2).
4) Run the (undirected) connected components algorithm on (directed) G.

<u>Running time</u>: $O(n+m) = O(|V|+|E|)$

Undirected connected components algorithm: (for step (4))

DFS-cc(G):

For all $v \in V$, visited(v) = FALSE

cc = 0

for all $w \in V$, (where v is ordered by ↓ Post # from (2))

if not visited(w) then:

cc++

Explore(w)

Explore(w):

visited(w) = TRUE

ccnum(w) = cc

for each $(w, z) \in E$:

if not visited(z) then Explore(z)

Proof of key lemma: Vertex with highest post # lies in a source SCC.

Claim: if S & S' are SCCs and if there is an edge from $v \in S$ to $w \in S'$ then max post # in S > max post # in S'

Assuming the claim we can topologically sort the SCC's by the max post # in each SCC.
The first SCC in the order is a source SCC & it contains the vertex with the maximum post #.
Thus this proves the lemma, just need to prove the claim.

Proof of claim:

There is a path $S \to S'$
(thus no path $S' \to S$ – otherwise S & S' are a single SCC)

Let z be the 1^{st} vertex in $S \cup S'$ that's visited by DFS.

Case 1: $z \in S'$

We'll see all of S' before visiting any of S,
so all post #'s in S' $<$ all post #'s in S

Case 2: $z \in S$

We'll do Explore(z) & see all of $S \cup S'$ before finishing z, thus:

post # of z $>$ post # of $w \in S \cup S' - \{z\}$

Since $z \in S$ we have the claim.

∎