Another BFS Forest (starting from l)

When not strongly connected there may be multiple roots.

DFS Forest

When not strongly connected there may be multiple roots.

DFS Forest

tree arcs \( a=(u,v) \) have

\[ n(u) < n(v) \]
\[ \textcolor{red}{f(u)} > \textcolor{blue}{f(v)} \]
\[ p(v) = u \]

\( v \) is white when \( a \) is examined

DFS Forest

back arcs \( a=(u,v) \) have

\[ n(u) > n(v) \]
\[ \textcolor{red}{f(u)} < \textcolor{blue}{f(v)} \]

\( v \) is gray when \( a \) is examined
DFS Forest cross arcs

Cross arcs \( a = (u, v) \) have
\[ n[u] > n[v] \]
\[ f[u] > f[v] \]
\( v \) is black when \( a \) is examined.

DFS Forest forward arcs

Forward arcs \( a = (u, v) \) have
\[ n[u] < n[v] \]
\[ f[u] > f[v] \]
\( v \) is black when \( a \) is examined.

Directed Acyclic Graphs (DAGs)

A topological sort of a directed graph is a numbering \( f[.]. \) of its vertices satisfying \( f[u] < f[v] \) for any arc \( a = (u, v) \).

Thm A directed graph has a topological sort if and only if it is acyclic.

Proof: 
1. Suppose \( f[.]. \) is a topological sort for \( D \).
2. Suppose \( D \) has a cycle: \( C = v_0, v_1, \ldots, v_k, v_0 \).
3. Since there is an arc from \( v_i \) to \( v_{i+1} \) for each \( 0 \leq i < k \),
\[ f[v_0] < f[v_1] < \ldots < f[v_{i-1}] < f[v_i] < f[v_{i+1}] \]
So \( D \) cannot have a cycle: it is acyclic.
Directed Acyclic Graphs (DAGs)

**Proof** (continued): Suppose \( D \) is acyclic.
Perform a DFS of \( D \) to obtain the post-DFS numbering \( f[\cdot] \).

No back arcs will be found since a back arc from \( u \) to \( v \) along with the tree path from \( v \) to \( u \) would form a cycle.
All tree, cross and forward arcs satisfy \( f[u] > f[v] \), so \(-f[\cdot]\) is a topological sort.

**Corollary** A directed graph is acyclic if a DFS finds no back arcs. All DFSs find no back arcs.

Finding Strongly Connected Components

The root of a SCC has no path to a proper ancestor in the tree.

A vertex \( u \) is not a root if it has a descendant \( v \) such that either

1. \( v \) has a back arc to a proper ancestor of \( u \), or
2. \( v \) has a cross arc to a vertex \( w \) whose root is a proper ancestor of \( u \).

**Thm** \( u \) is the root of a SCC if and only if \( dlow[u] = n[u] \).

**Proof:** Suppose \( u \) is a root of a SCC.
If \( dlow[u] \neq n[u] \), then \( dlow[u] < n[u] \).
So \( \exists w \) with \( n[w] = dlow[u] < n[u] \), and a descendant \( v \) of \( u \) (\( v=\) possible) with a back or cross arc to \( w \) and a path from \( w \) to a proper ancestor of \( u \).
Proof: (continued)

Then w and u are strongly connected. But since \( n[w] < n[u] \), w would not be a root (the first vertex discovered in its SCC).
So \( \text{dlow}[u] = n[u] \).

Now suppose \( \text{dlow}[u] = n[u] \).
Suppose the root of u’s SCC, r, is not u.
Since r and u are strongly connected, there is a path P from u to r.
Since \( n[r] < n[u] \), r is not a descendant of u.

Proof: (continued)
Let w be the first vertex on P that is not a descendant of u.
Let v be the vertex on P just before w. (\( u = v \) and \( w = r \) are possible)
The arc from v to w is either a cross or back arc.
If the (v, w) arc is a back arc, \( n[w] < n[u] \)
since w is not a descendant of u.
If the (v, w) arc is a cross arc, \( n[w] < n[v] \) and
\( n[w] < n[u] \) since w is not discovered within the DFS of u, but v is.

Proof: (continued)
So there is a descendant v of u with a cross or back arc to w, and w has a path to u. (Follow P from w to r then the tree from r to u.)
\[ \text{dlow}[u] \leq n[w] < n[u] \]
So u must be the root of u’s SCC.

How to calculate \( \text{dlow}[\cdot] \) ????

\[ \text{dlow}[u] = \min\{ n[u] \} \]
\[ n[w] \]
there is a descendant v with either a back arc to w, or with a cross arc to w and w has a path to a proper ancestor of u.

Case 1: cross arc and w is not a descendant of u.
black vertices have \( \text{dlow}[\cdot] \) calculated and of those, which are roots, is known.
w has a path to a proper ancestor of u if w’s root is a proper ancestor of u.

Proof: (continued)
Case 2: back arc and \( w \) is not a descendant of \( u \).

\[
dow[u] = \min\{n[u] \cup \{n[w]\} \}
\]

There is a descendant \( v \) with either a back arc to \( w \), or with a cross arc to \( w \) and \( w \) has a path to a proper ancestor of \( u \).

\( w \) will have a path to \( u \) since it is an ancestor of \( u \).

\( w \)'s root has not yet been found.

Case 3: \( w \) is a descendant of \( u \).

\[
dow[u] = \min\{n[u] \cup \{n[w]\} \}
\]

If \( n[u] < n[w] \) so \( n[w] \) does not influence the value of \( dow[u] \).

Use a stack \( P \) to store discovered vertices.

Pop \( P \) whenever a root is identified removing all of the root's descendants still in \( P \).

If \( w \)'s root is discovered during DFS(\( u \)) then \( n[w] > n[u] \).

\[
dow[u] = \min\{n[u] \cup \{n[v] \ | \ u \ has \ a \ back \ or \ cross \ arc \ to \ v \ and \ v \'}s \ root \ has \ not \ yet \ been \ found \}
\]

Then \( w \)'s root has not been found if \( w \) is still on \( P \).