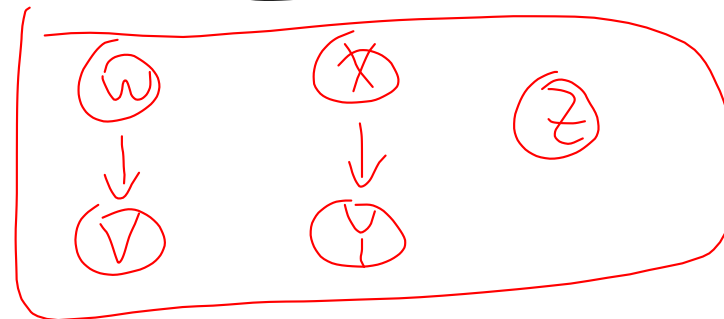
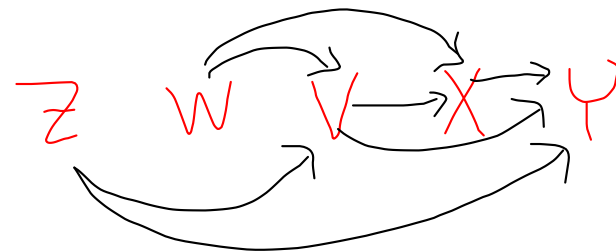
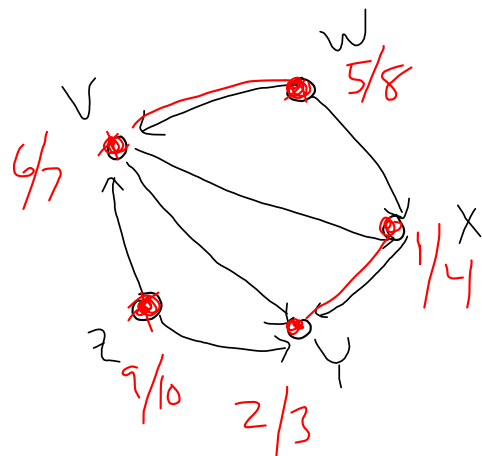


Then: G^{scc} has no cycles

Corr: For any digraph G , G^{scc} is a dag.

If we sort the vertices of a dag by finishing times, highest-to-lowest, we get a topological sort of the dag.

$(x (y y) x) (w (v v) w) (z z)$



Start at V .



$W \ Z \ V \ X \ Y$

Thm: This works!

Proof: It suffices to

prove that this sort,

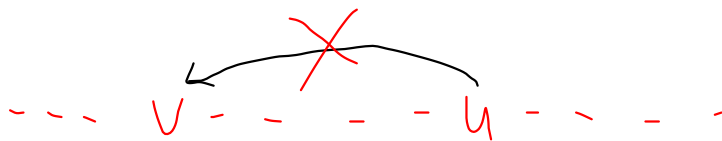
(finishing times high to low)

will not have any left-bound edges.

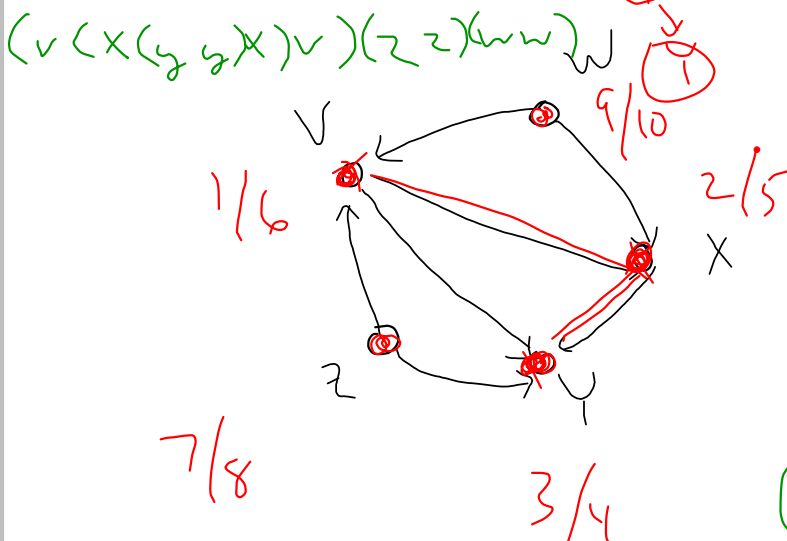
That is, there will not be an edge

from vertex u to vertex v if

$$f[u] < f[v]$$



topological sort

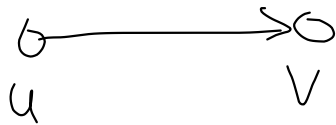


Let's see what it would mean for a dag

To have an edge $u \rightarrow v$, but $f[u] < f[v]$

~~Case 1: u and v are in the same tree in our DFS.~~ Don't need case 1

Case (a): Suppose we discover u before v .



discover u

discover v

finish u

finish v must go there

Contradiction!

(get a cycle)

Case (b):

discover v before u

discover v Since we have

discover u

finish u

finish v

But this implies the dag contains a path from v to u .

Contradiction!

Theorem (Nested Parentheses Theorem)

The discovery "(" and finishing ")" times of a DFS form a properly-formed parenthesized expression.

good (() ())

bad ()) (

good (a (b b) (c c) a)

bad (a (b a) (c c) b)

Thm: The following algorithm finds SCCs in a digraph.

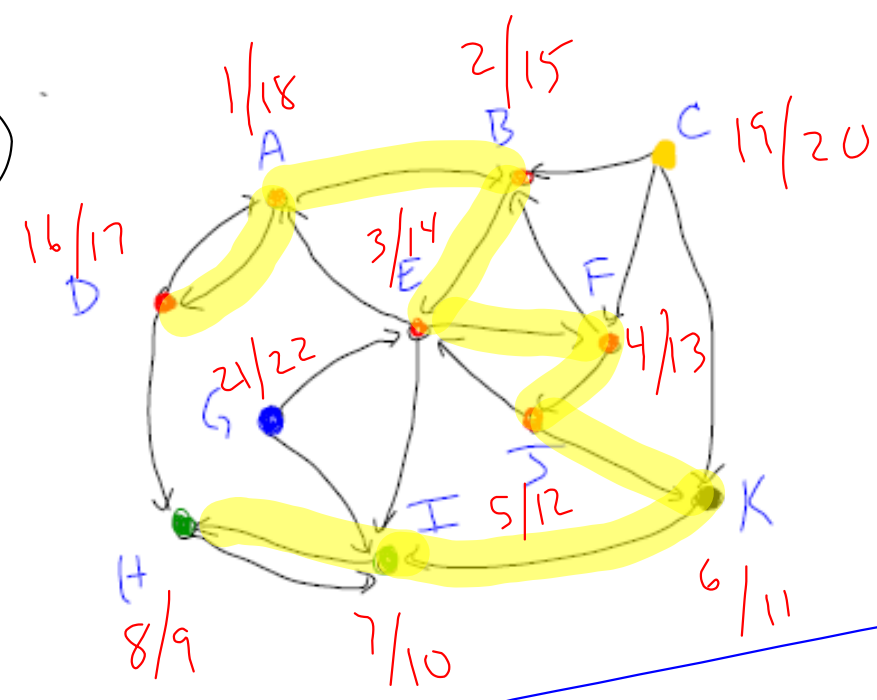
D ,

$D^T = \text{"D Transpose"}$

Find-SCC

- Do a DFS on D
- Sort vertices by finishing times, high-to-low.
- Reverse every arc (edge), call it D^T
- Do a DFS on D^T , with the outer loop "for each white vertex ..." run according to the above sort.
- Each tree created by this last DFS is a SCC of D .

graphviz



Show Steps

Finish the Alg +10 x
 +15 if it's really nice!
 Hand in Monday
 Turn, Mike.