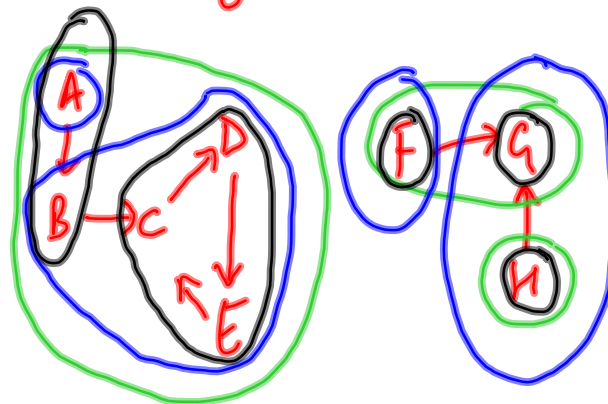


Connectivity

Undirected



Directed

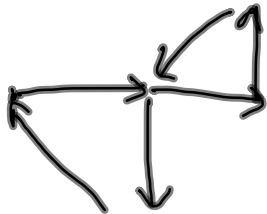


Notion of a component based purely on DFS/BFS seems unstable

When is DFS/BFS guaranteed to identify "same" component for a set of vertices

e.g. $\{C, D, E\}$ is always in same component in previous example

but $\{C, B\}$, $\{A, B\}$, $\{F, G\}$ -- are not



Symmetry: If $u \rightsquigarrow v$ is a path, so is $v \rightsquigarrow u$

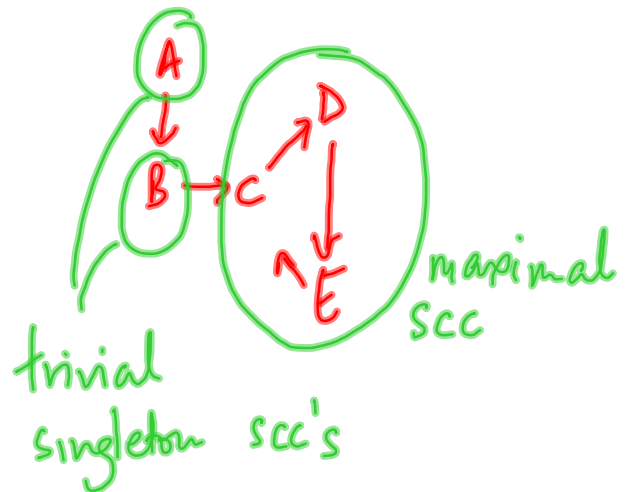
"Strongly connected components"

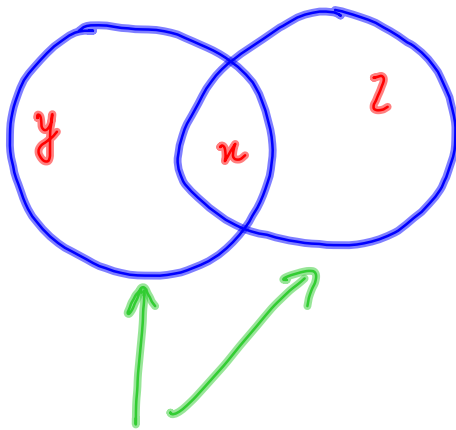
$X \subseteq V$ is an s.c.c. if $\forall u, y \in X \quad u \xrightarrow{x \neq y} y \ \& \ y \rightarrow u$

Typically we are interested in maximal scc's



max.
Can scc's overlap?





max scc's cannot
overlap

We know $y \rightsquigarrow x$
 $x \rightsquigarrow y$

$z \rightsquigarrow x$
 $x \rightsquigarrow z$

$\therefore y \rightsquigarrow z$
 $z \rightsquigarrow y$

Clearly maximal scc's (a) are disjoint
i.e. max scc's partition the graph (b) cover the graph

How to identify & label the scc's in a graph?

We know:

scc's partition the graph

$$G = (V, E) \quad V = C_1 \dot{\cup} C_2 \dot{\cup} \dots \dot{\cup} C_k$$

each C_i is a (max) scc

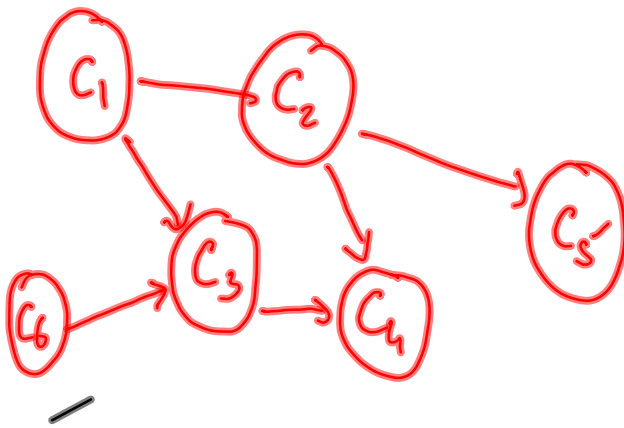
Construct SCC graph

$$\text{Vertices} = \{C_1, C_2, \dots, C_k\}$$

$$\text{Edges: } C_i \rightarrow C_j \text{ if } \exists v_i \in C_i, v_j \in C_j \quad v_i \rightarrow v_j$$

Claim: The SCC graph is a dag

If $C_i \rightarrow C_j$ & $C_j \rightarrow C_i$ then $C_i \cup C_j$ is also an SCC $\Rightarrow C_i, C_j$ are not maximal!



If we start DFS/BFS in C_5 or C_4 , the component would be an SCC

Trick:

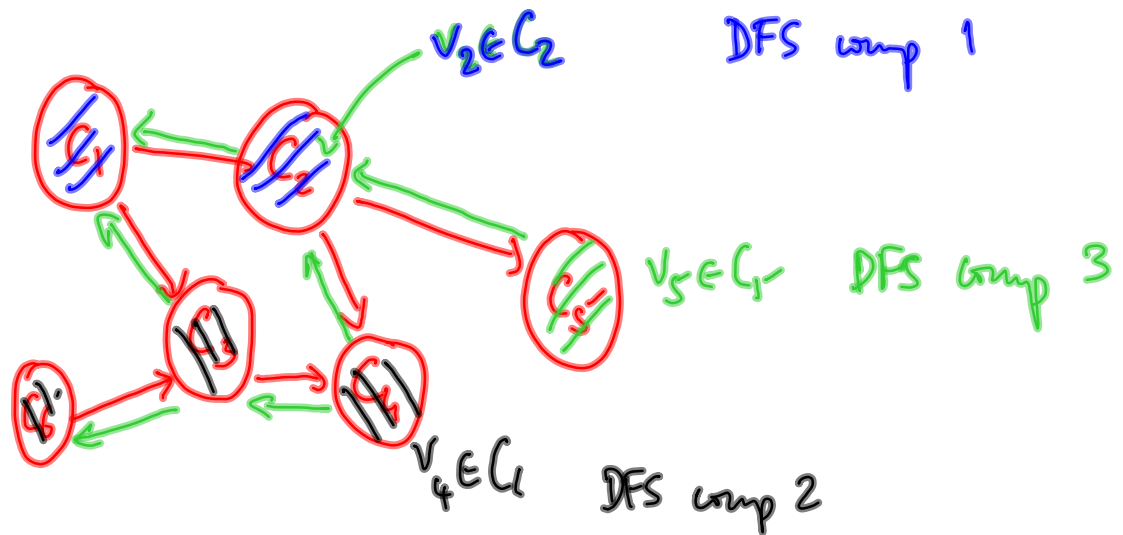
Reverse all edges in original G

Same SCC's

Top structure of SCC graph is reversed

Run DFS on G^{reverse}

Associates $\text{post}(v)$ to each v wrt. G^{reverse}



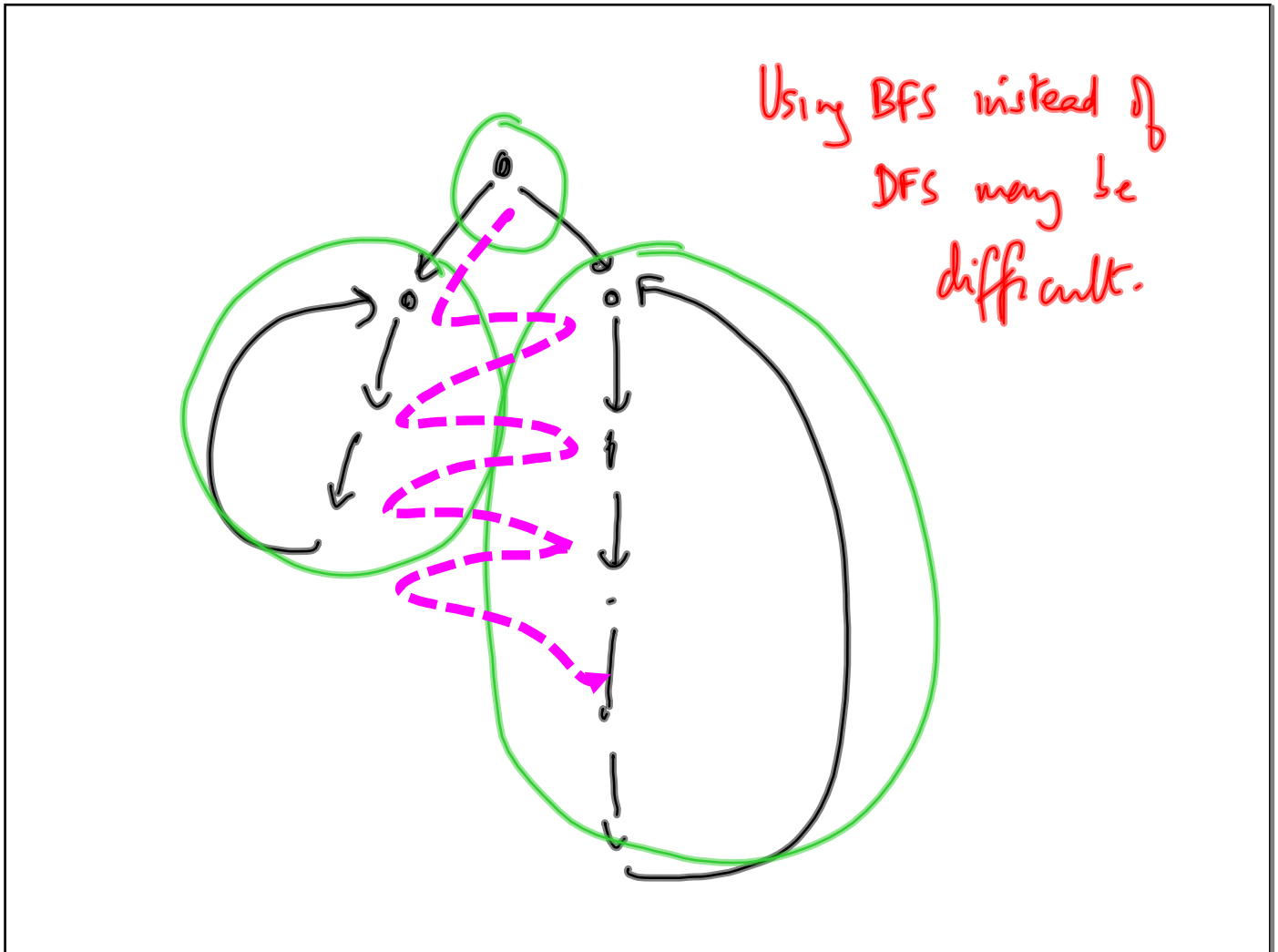
Within each component, start vertex has highest $post(v)$

$$\begin{array}{ccc}
 post(v_2) & \stackrel{(scc)}{<} & post(v_4) & \stackrel{}{<} & post(v_5) \\
 \text{max in Comp 1} & & \text{max in Comp 2} & & \text{max in Comp 3}
 \end{array}$$

Go back to original G

Run DFS in descending order of $\text{post}(v)$
as computed by DFS on G^{rev}

Each component identified by this DFS
will correspond exactly to an SCC



Back to topological sort

Given a DAG $G = (V, E)$

Scan G & compute $\text{indegree}(v)$ for each v

"preprocessing"

$O(n+m)$
with adj lists

$X = V$

while $X \neq \emptyset$

print any $x \in X$ of indegree 0

takes $O(n)$
time

remove x from X (& outgoing edges)

updates only $\text{outdegree}(x)$

Maintain and update a list of vertices of
indegree 0

More conventional to maintain a queue

If DAG is not empty, queue of indegree 0
vertices is not empty

Modified topological sort

$Q = \emptyset$

for each $v \in V$, compute $\text{indegree}(v)$

for each $v \in V$, if $\text{indegree}(v) = 0$, add v to Q

while Q is not empty

extract head of Q as q & print it

reduce $\text{indegree}(v)$ for all v s.t. $(q, v) \in E$

if $\text{indegree}(v) = 0$, add v to Q