Today:

- DAGs
- directed connectivity
- BFS

def: A directed acyclic graphs is a digraph with no cycles

Q: given a digraph G, does G have cycles?

IsAcyclic(G):

run DFS on G, and,

output "DAG" if DFS reveals no **backedges**.

def: $(u,v) \in E$ is backedge

if $[\ [\ \ ]\ ]$

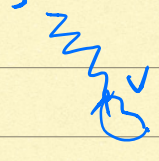
$v\ \ u\ \ u\ \ v$

claim: G is DAG

$\Updownarrow$

DFS(G) reveals no back edges

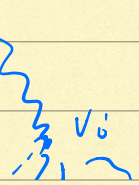
proof: backedge → cycle



cycle → backedge

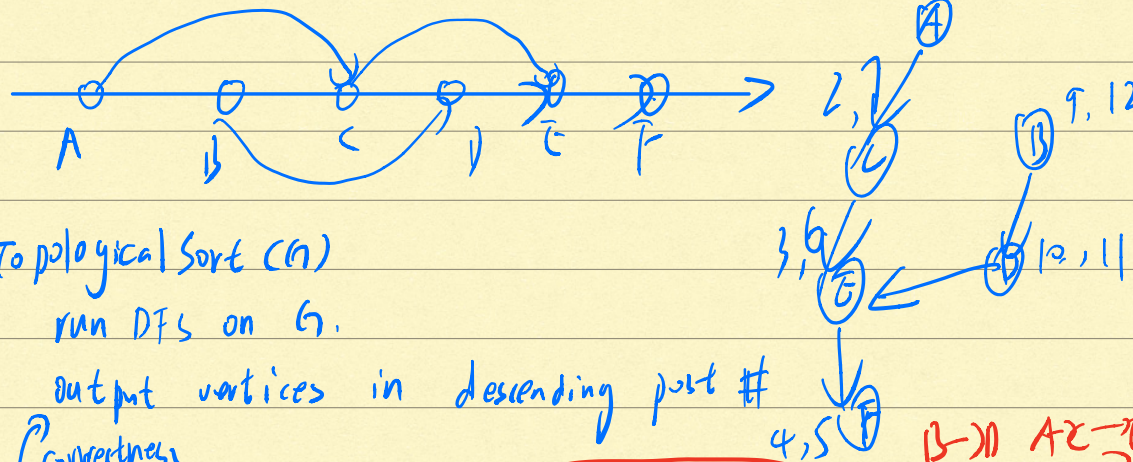
let $v_1, \dots, v_t$ be a cycle.

let v_i be the first visited vertex.



Topological Sort.

Given G that is a DAG, want to order the vertices in such a way that every edge points forward.



Topological Sort (G)

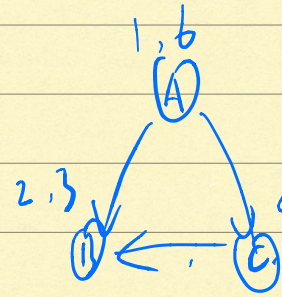
run DFS on G.

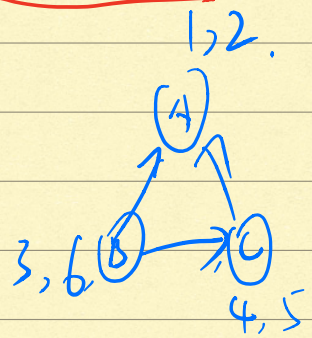
output vertices in descending post #

correctness

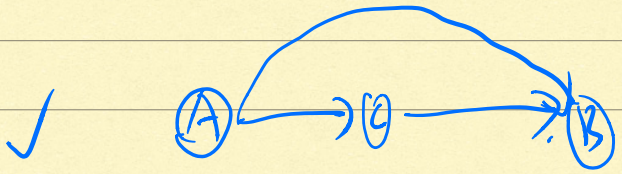
lemma: $(u,v) \in E \rightarrow$ post(u) > post(v)

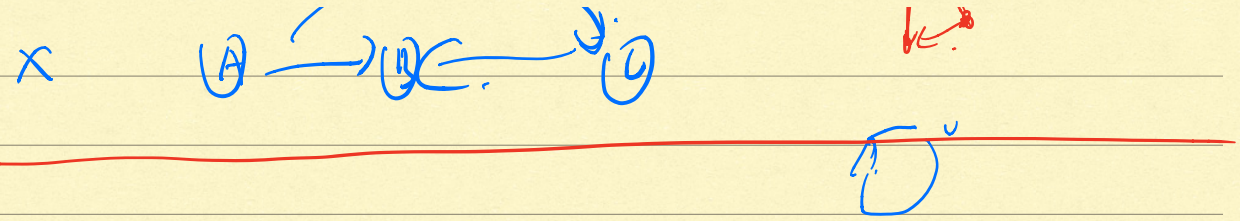
B → A → E → F





A → B → C



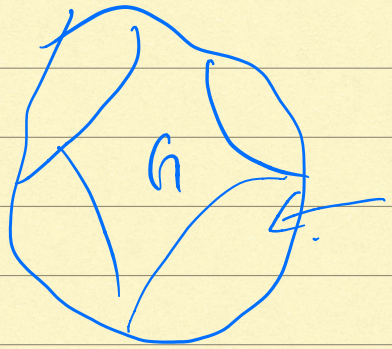


Connectivity for digraphs

def: $u, v \in V$ are (strongly) connected

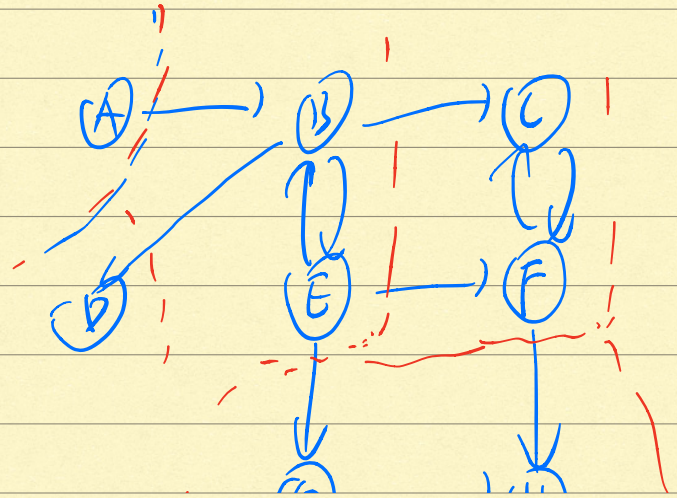
if : $u \rightsquigarrow v$.

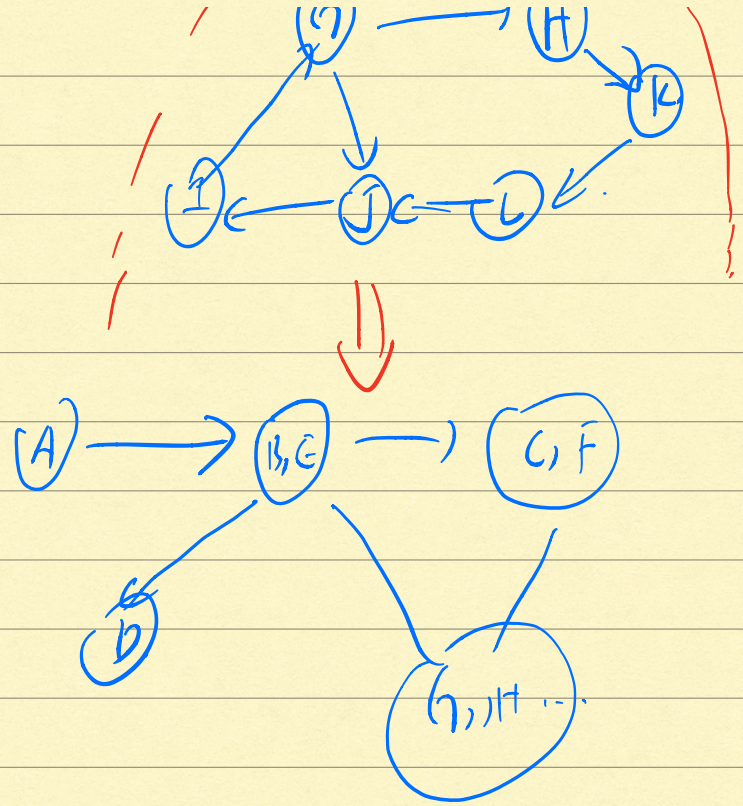
: $v \rightsquigarrow u$



strongly connected components.

claim: every digraph G is a DAG on SCCs.





Side question:

given a digraph G, find any v in source SCC.

A: the vertex with largest post #

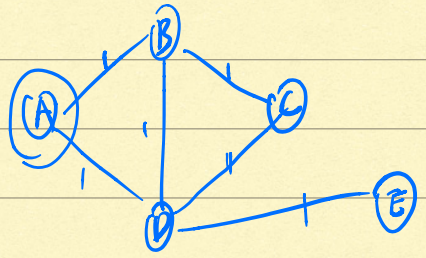
SCC(G):

1. deduce G^R from G.
2. run DFS on G^R to have post #s
3. $\forall v \in V$ in reverse post order of G^R

```
if not visited[v]:
    explore(G, v).
```

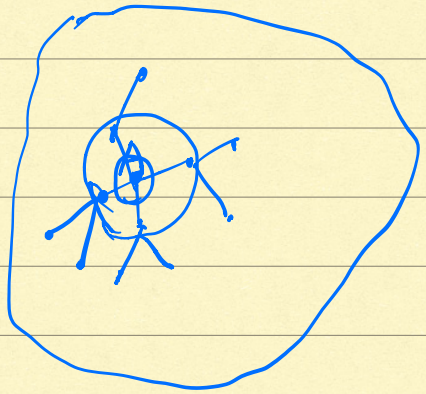
LL scc ff

New topic: shortest paths



	A	B	C	D	E.
A	0	1	2	1	2.

Q: find distance of A to all other vertices



BFS(G, s)

- dist[s] = 0
- dist[V/{s}] = ∞

- Q = [s]

while Q ≠ [ ]

u = eject(Q)

for (u,v) ∈ E

if dist[v] = ∞

enject(Q,v)

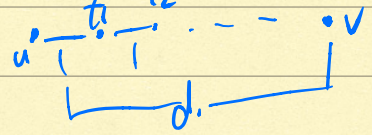
dist[v] = dist[u] + 1

$O(n+m)$

Shortest paths with distances/weights

idea: use BFS again by modifying graph.

Transform G into G' such each edge u — d — v is replaced with



$O(n'+m')$

$\begin{cases} n' = |V'| = n? \\ m' = |E'| = m? \end{cases}$ > big!