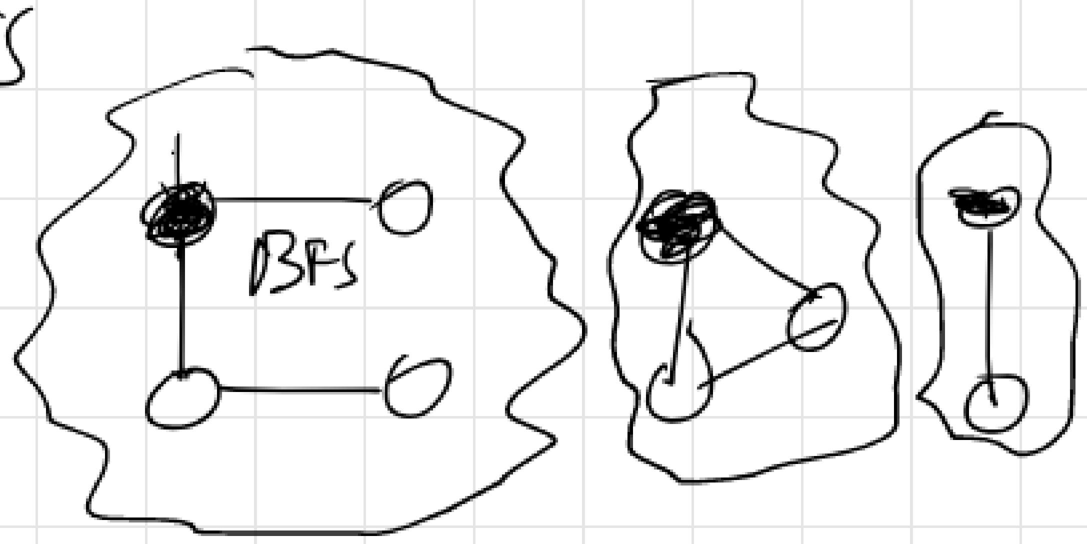
Application of DFS

→ Find the set of all connected components in an undirected graph.

→ Did it by BFS

BFS(i) = find the connected component containing i



Run DFS(i) = find the connected component containing i

1) whether there exists a cycle in undirected graph.

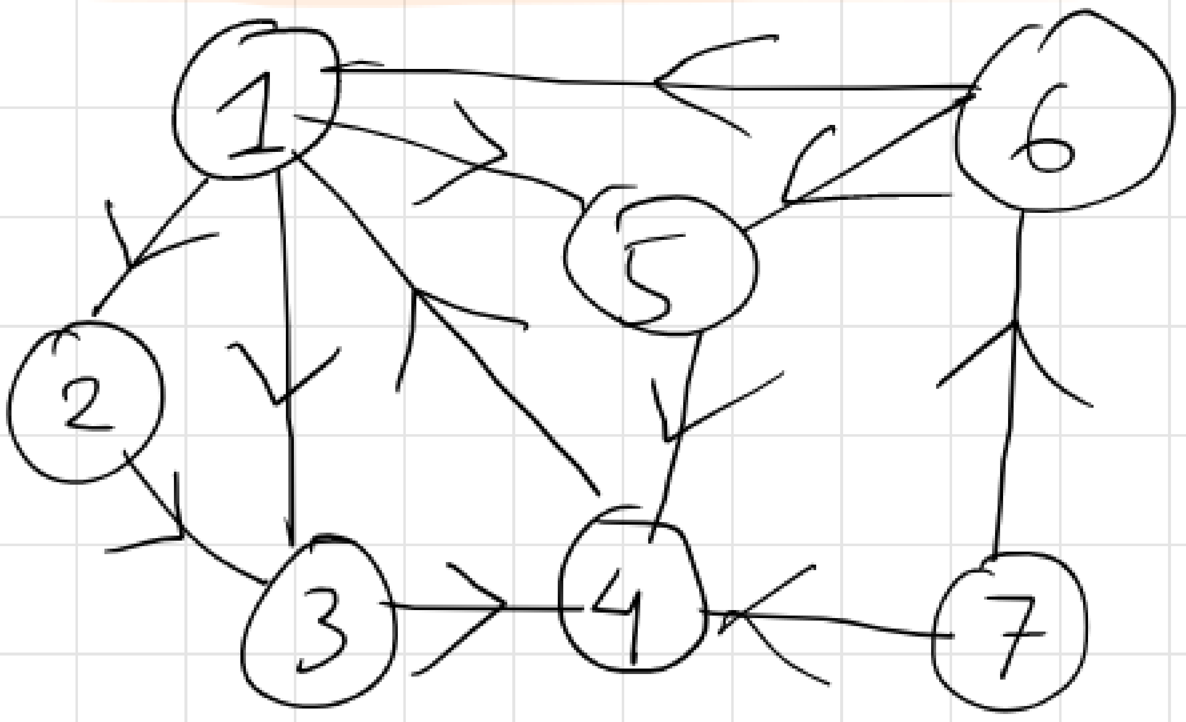
2) Whether there exists a cycle in directed graph.

3) Given a directed graph G and a node u, find the set of nodes in G from which there exists a path to u.

4) Given a directed graph G, whether G is strongly connected?

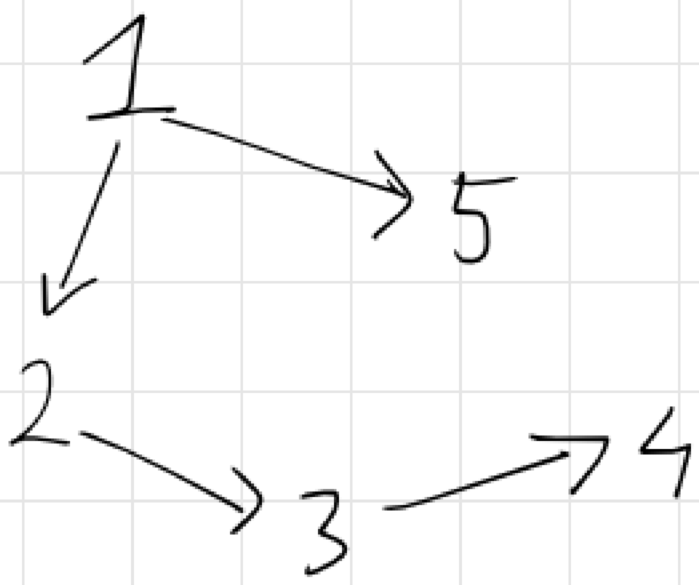
5) Given a directed graph, find all strong connected components.

DFS on directed graphs



DFS(1)

=

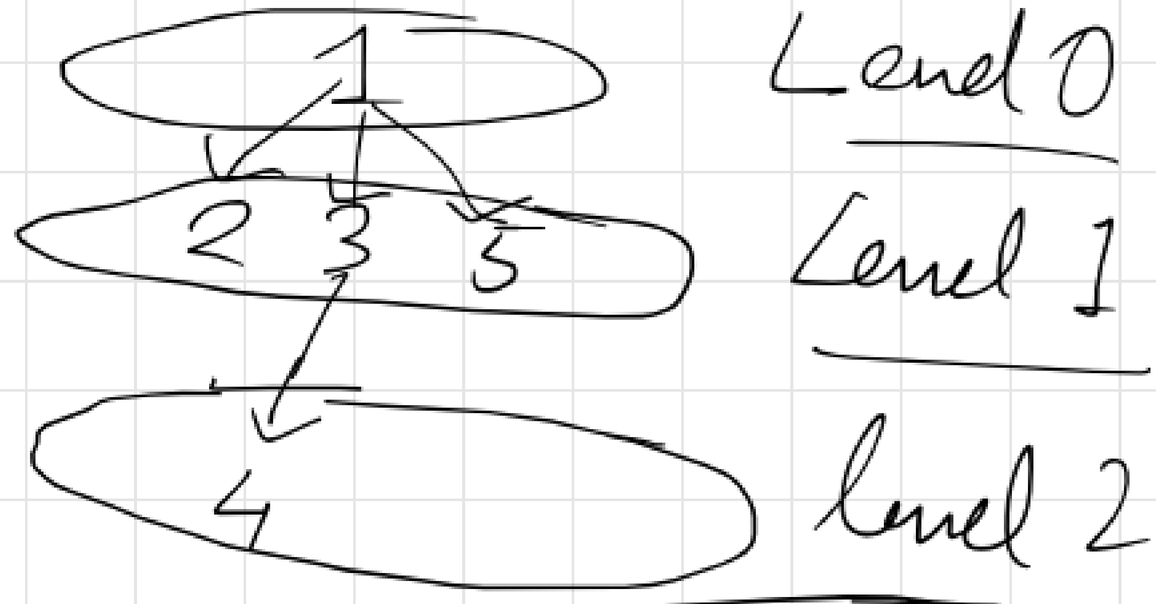


(6, 7 → undiscovered)

BFS on directed graphs

BFS(1)

=



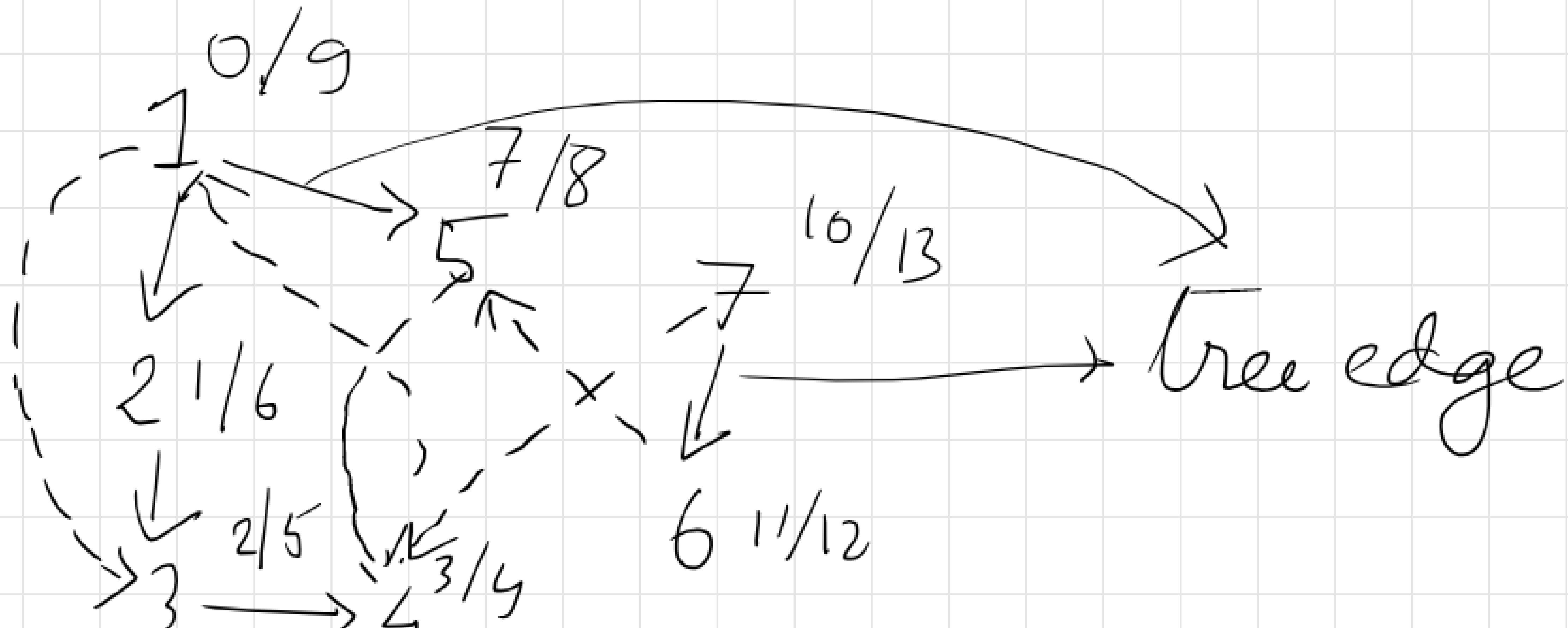
$\boxed{\text{6, 7 undiscovered}}$

Both DFS & BFS discover all nodes to which there is a path from node 1

The procedures can be extended from an unvisited node

BFS(7) 7 → 6   DFS(7) 7 → 6

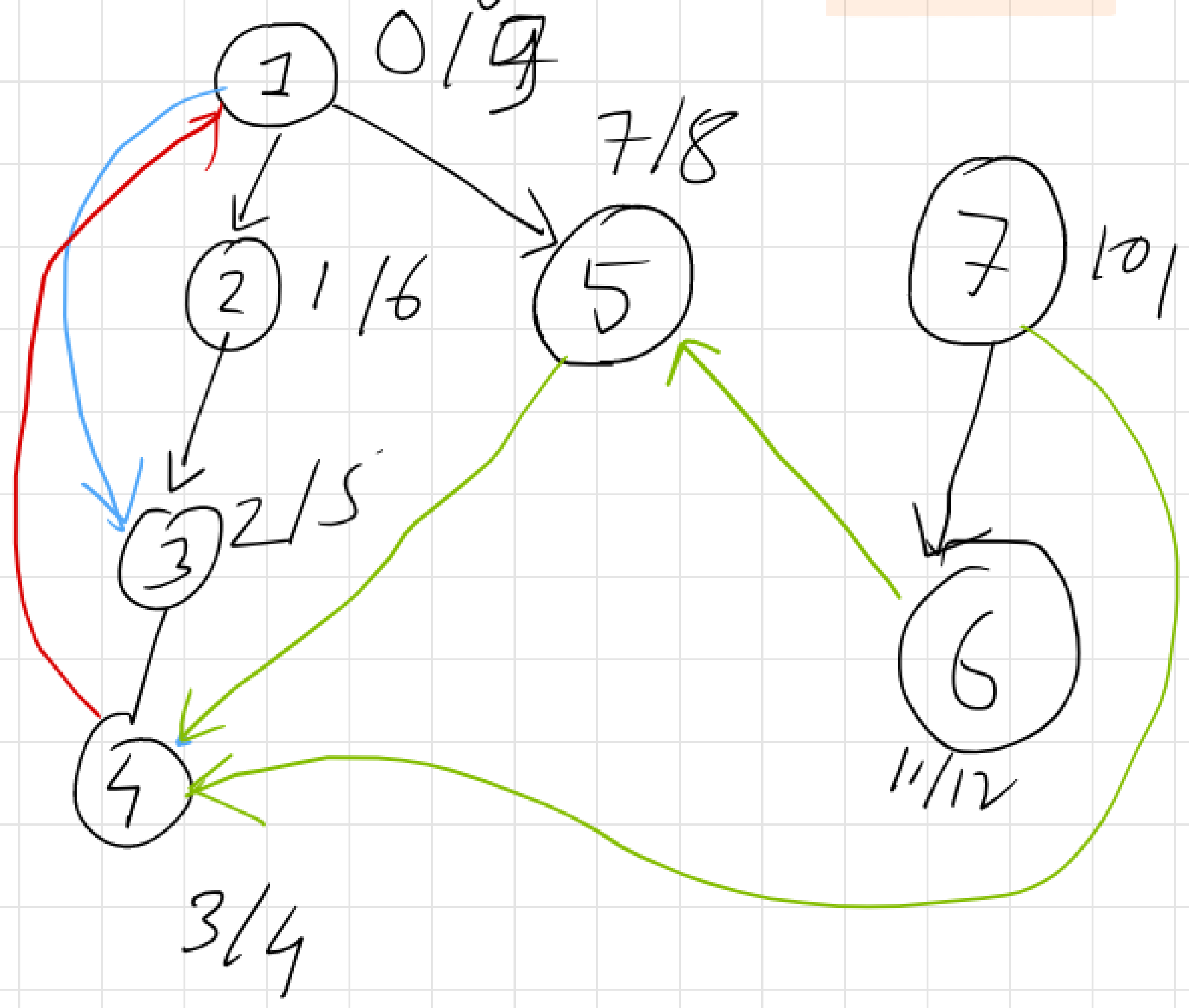
DFS Tree



non tree edge (Read from Cormen, Tardos)

— tree edge
— back edge
— forward edge
— cross edge

[not connecting any ancestor descendant



Tree edge (u, v) → belong to tree

$$d(u) < d(v) < f(v) < f(u)$$

forward edge (u, v) → edges going forward in tree

$$d(u) < d(v) < f(v) < f(u)$$

Back edge (u, v) $(4 \to 1)$

edges going back in tree

$$d(u) > d(v)$$
$$f(u) < f(v)$$

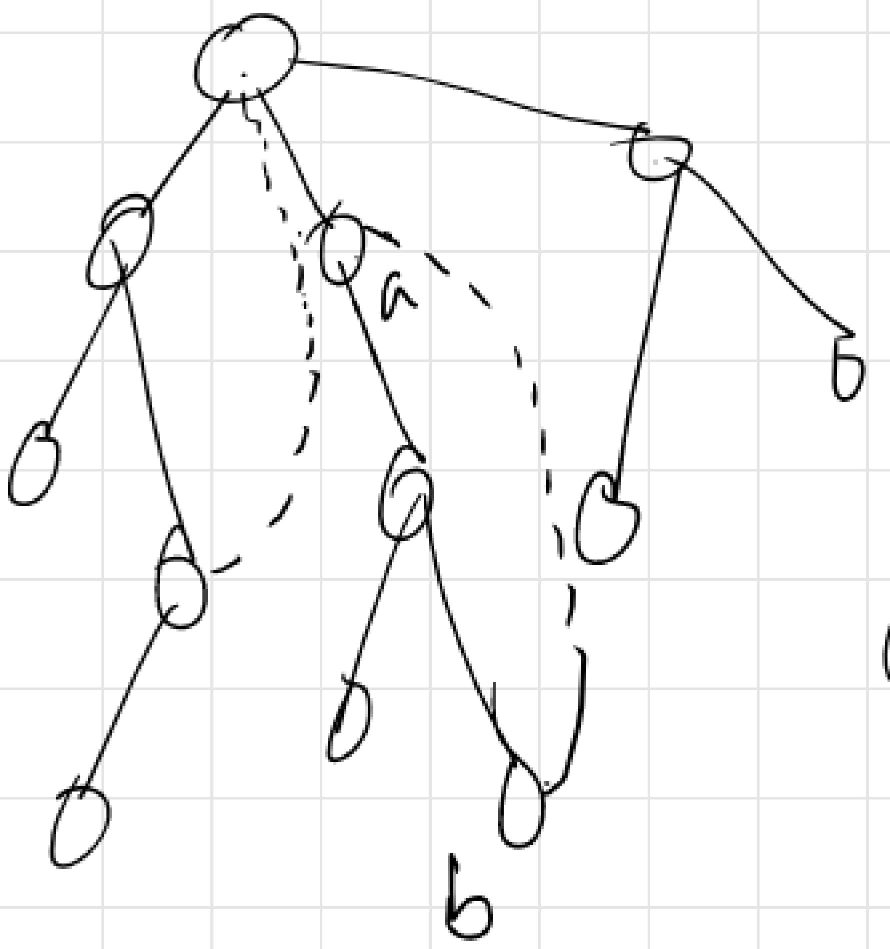
Cross edge (u, v) $(5 \to 4)$

not connecting any desc ance.

$$d(v) < f(v) < d(u) < f(u)$$

[Check if a given undirected graph has a cycle.]

DFS Tree



[let $b\,a$ back edge]

(if there is a back edge, then there is a cycle)

"no back edge $\longrightarrow$ no cycle" ??

Yes

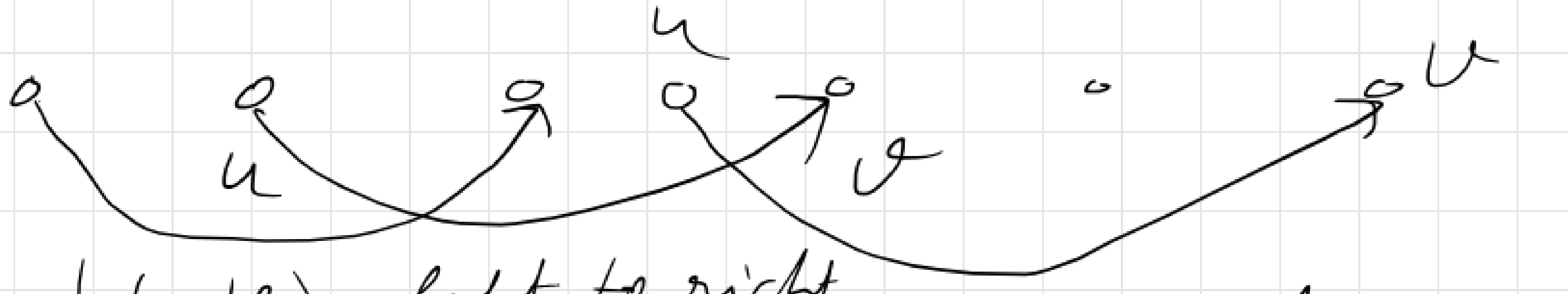
Check if a given directed graph G has a cycle

back edge $\longrightarrow$ cycles ✓

no back edge $\longrightarrow$ no cycles ?

→ Do a DFS (on any node) and order the vertices in decreasing order of finish time.

largest finish time



Smallest finish time

(u, v) left to right
(tree edge)

forward edge

[Cross edge (u, v)]

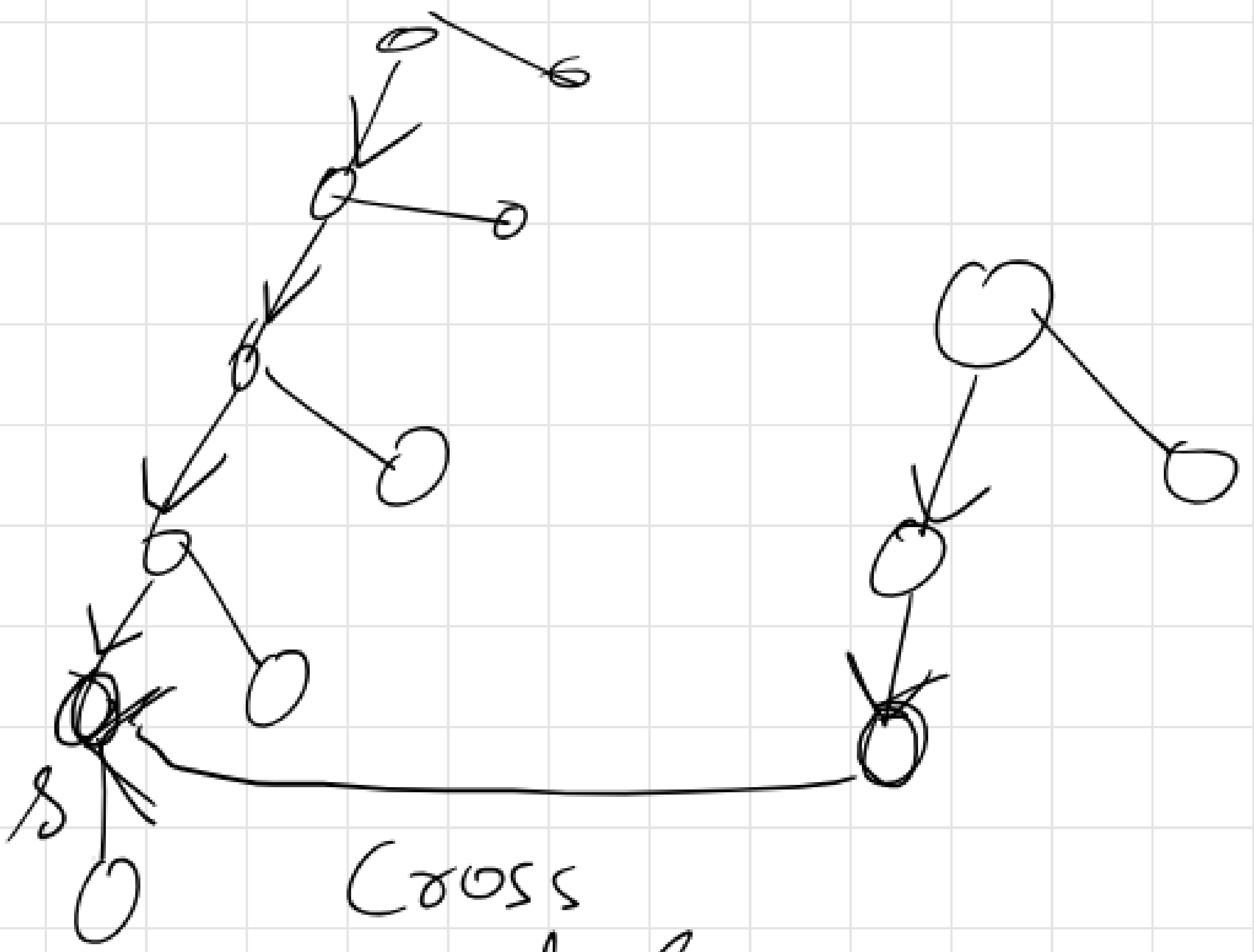
" Can they form a cycle?

$(u, v) \equiv$ back edge
right to left.

(no back edge $\rightarrow$ no cycle) Yes

(If there is a cycle, there must be a back edge.)

Given a directed graph $G=(V,E)$, and a node s, find the set of nodes having path to s



ancestor(s) has a path to s
other ??

Algo $G_{Rev} = (V, E_{Rev})$

DFS(s) in G_{Rev}, all the nodes reachable from s in the reverse graph

Claim 1: A node has a path from s in G_{Rev} iff it has a path to s in G.

$\boxed{\text{Proof HW}}$

Claim 2: If u and v are mutually reachable, v and w are mutually reachable, then u and w are mutually reachable (MR: mutually reachable)

Pf: Homework

Read Tardos for both the claims

Q) Given a directed graph, is it strongly connected.

Naive DFS(u) — all nodes reachable from u

Is every nodes reachable from u?

no $\longrightarrow$ not strongly connected.

yes $\rightarrow$ u has a path to every vertex.

repeat for all vertices $v \in V$
$DFS(v)$

$O((m+n)\,n)$

Can we do better than this?

$DFS(u)$: all nodes reachable from u?

no = not strongly connected

yes

$G_{rev} = (V, E^{Rev}) \rightarrow DFS(u)$ in G^{Rev}

all nodes reachable from u? NO $\rightarrow$ not strongly connected

Yes

(all nodes have a path to u in G) (by claim 1)

$\exists\, s \rightsquigarrow u$ & $\exists\, u \rightsquigarrow s$ $\forall s \in V$

Consider $v_1 \,\&\, v_2 \in V$ = u and v_1 are MR
u and v_2 are MR

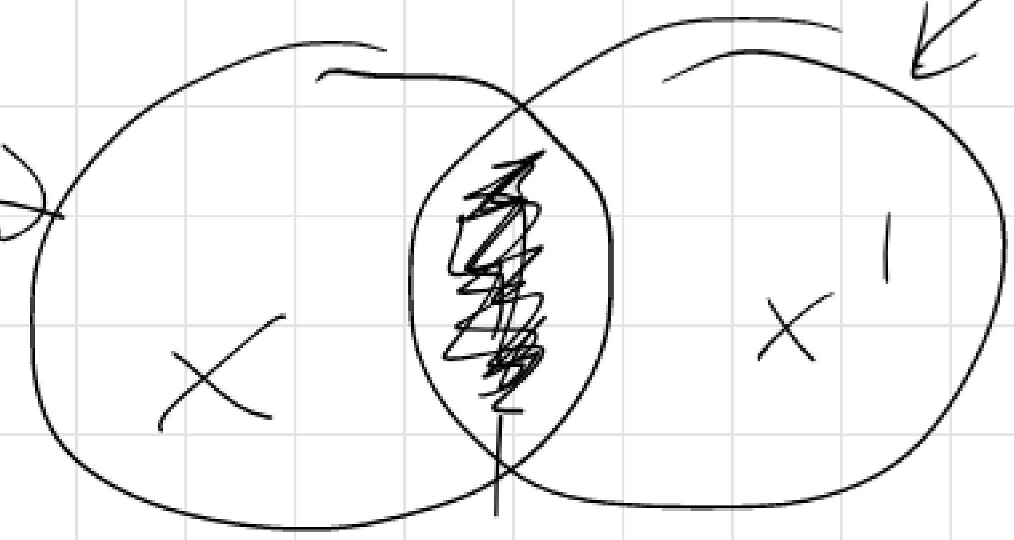
(Claim 2) v_1 and v_2 are MR

Strongly connected component

Connected component of (s)
set of all nodes v
s.t. s and v are
mutually reachable

$DFS(s) = \{\text{reachable from } s\}$

$DFS(s)$ on $G^{Rev} = \{s \text{ reachable from other node}\}$



Strong component of s $\left\{ X \cap X' = \text{nodes mutually reachable with } s \right.$

Claim: For any two nodes s and t in a directed graph, their strong component are either identical or disjoint

Two Cases

Proof:

case1

s, t mutually reachable

Case 2

s, t are not mutually reachable

→ s, t are mutually reachable → Case 1

To prove: Strong components s & t are identical

Consider $v \in SC(s)$

Also s and t are MR

By claim 2 v and t are MR

$v \in SC(t)$

$\forall v \in SC(s) \rightarrow v \in SC(t)$

$SC(s) \subseteq SC(t)$

Slly

$SC(t) \subseteq SC(s)$

$SC(s) = SC(t)$

→ s and t are not mutually readable

To prove $SC(s) \cap SC(t) \neq \emptyset$

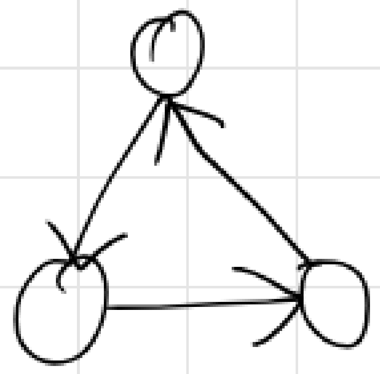
let $\exists u \in SC(s) \cap SC(t)$

u and s are MR
u and t are MR

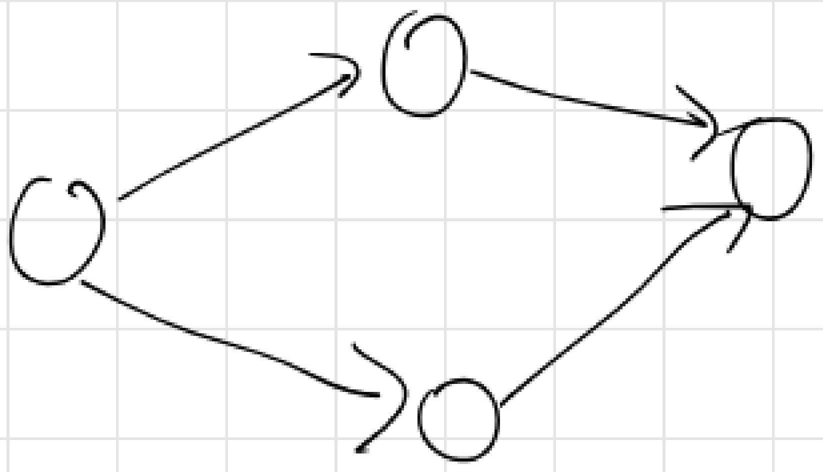
By claim s and t are MR

so $SC(s) \cap SC(t) = \emptyset$

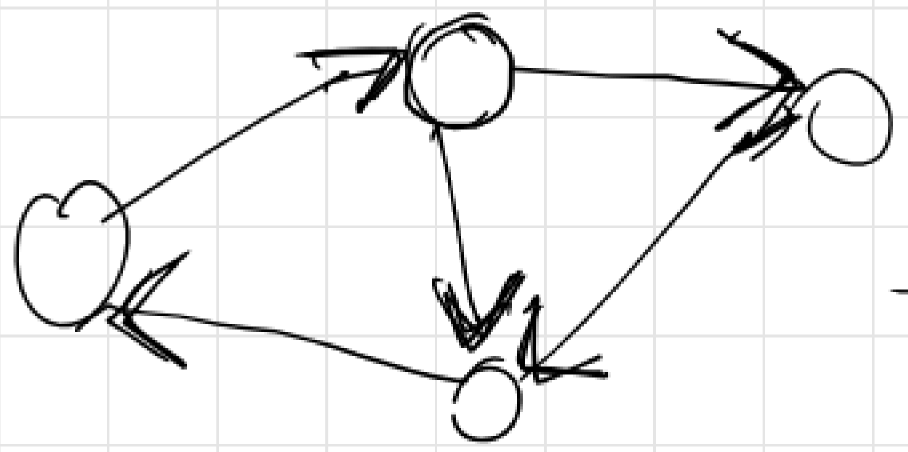
Example of strongly connected



→ strongly connected

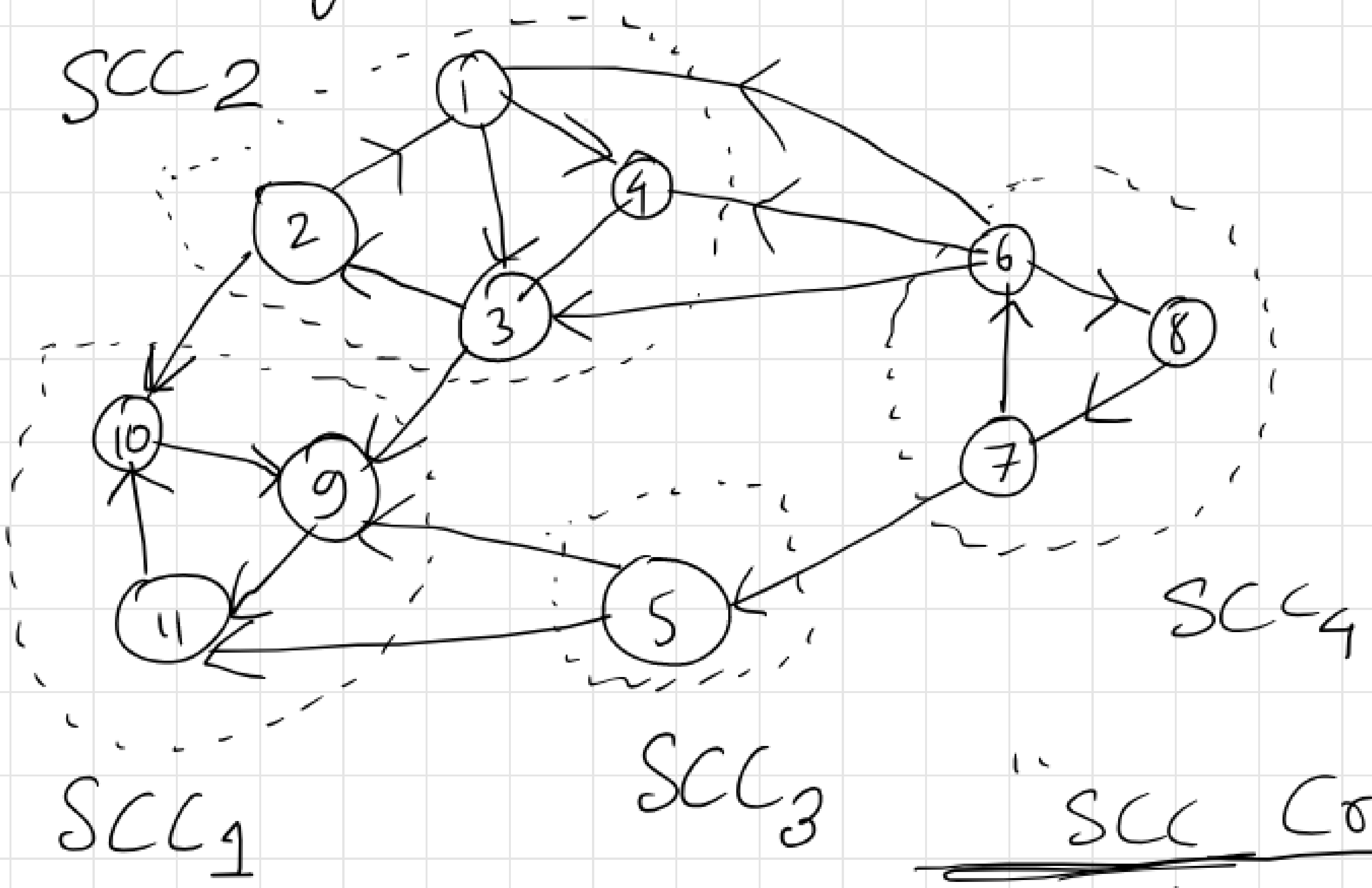


→ not strongly connected



→ strongly connected

Strongly connected components in a graph



"SCC Criteria"

→ $\forall u, v$, $\exists$ $u \rightsquigarrow v$ path
$\exists$ $v \rightsquigarrow u$ path

→ maximal too

nodes {1, 2, 3} is not SCC, because it is not maximal, Whereas {1, 2, 3, 4} is a SCC

There are 4 SCC in the above graph

$SCC_1 = \{1, 2, 3, 4\}$

$SCC_2 = \{5\}$

SCC_4 $\{6, 7, 8\}$

$SCC_1 = \{9, 10, 11\}$

Find the strong connected components of a directed graph G.

~~Let~~ $DFS(10) = \{9, 10, 11\}$

↓

SCC_1

$DFS(8) = \{8, 7, 6, 1, 4, 3, 2, 9, 10, 11, 5\}$

$SCC_1 \cup SCC_2 \cup SCC_3 \cup SCC_4$

$DFS(1) = \{1, 4, 3, 2, 9, 10, 11\}$

$SCC_1 \cup SCC_2$

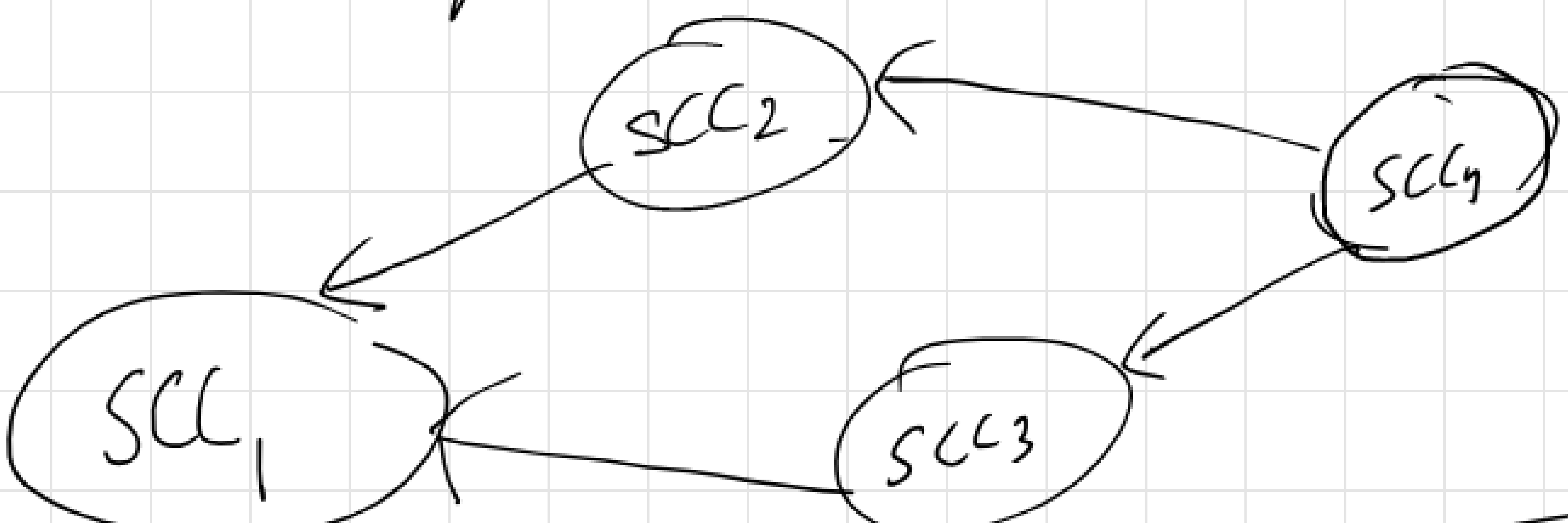
[Merely doing DFS will not give much idea of strongly connected components]

In previous example, invoking DFS on node 10 gave you SCC_1,
but invoking DFS on node 8, gave all the nodes

- right place to invoke DFS. ??

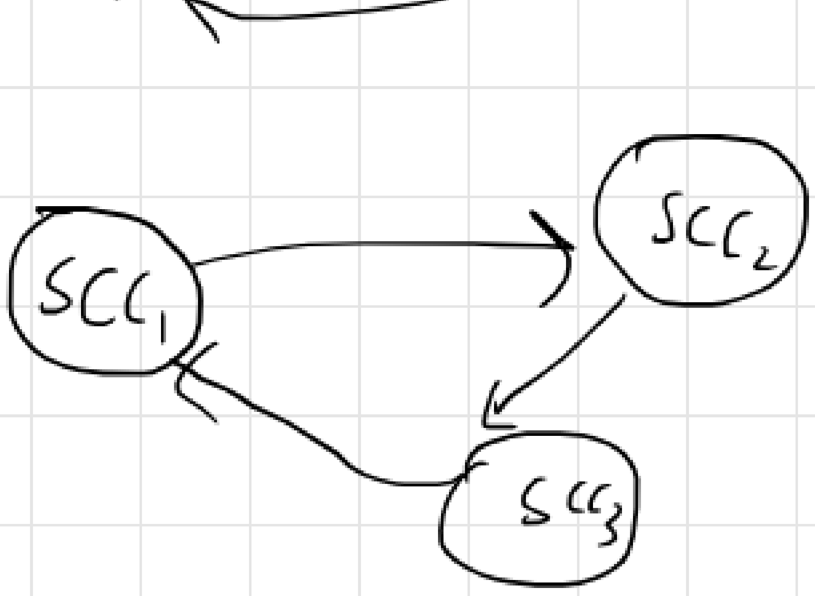
<u>Observation</u> no outgoing edge from the component to other component

<u>Component graph (MetaGraph)</u> (nodes correspond to components



(1) $SCC_i \rightleftarrows SCC_i$ X not possible

(2)



<u>no cycle possible</u>

> Can we claim that if $\nexists$ any cycle in component graph, $\exists$ a component having no out-edges?

$\rightarrow$ Yes !!

Pf: Homework.

How to figure out which component has no out edge?

$\rightarrow$ $G = (V, E)$ $\qquad G_{rev} = (V, E_{rev})$

What is the relation between the strong components of G & G_{rev}?

Claim Strong connected components of G and G_{rev} are same.

Pf: Homework

Let C_i and C_j are two components of G_{rev}.

If there exists an edge between these two components C_i and C_j

$$C_i \longrightarrow C_j$$

$$f(C_j) = \max_{u \in C_j} f(C_j)$$

$$f(C_i) = \max_{u \in C_i} f(C_i)$$

In graph G, (not in G_{rev})

$$C_i \longleftarrow C_j$$

What can you say about the component graph ~~on~~ after doing DFS on G_{rev}?

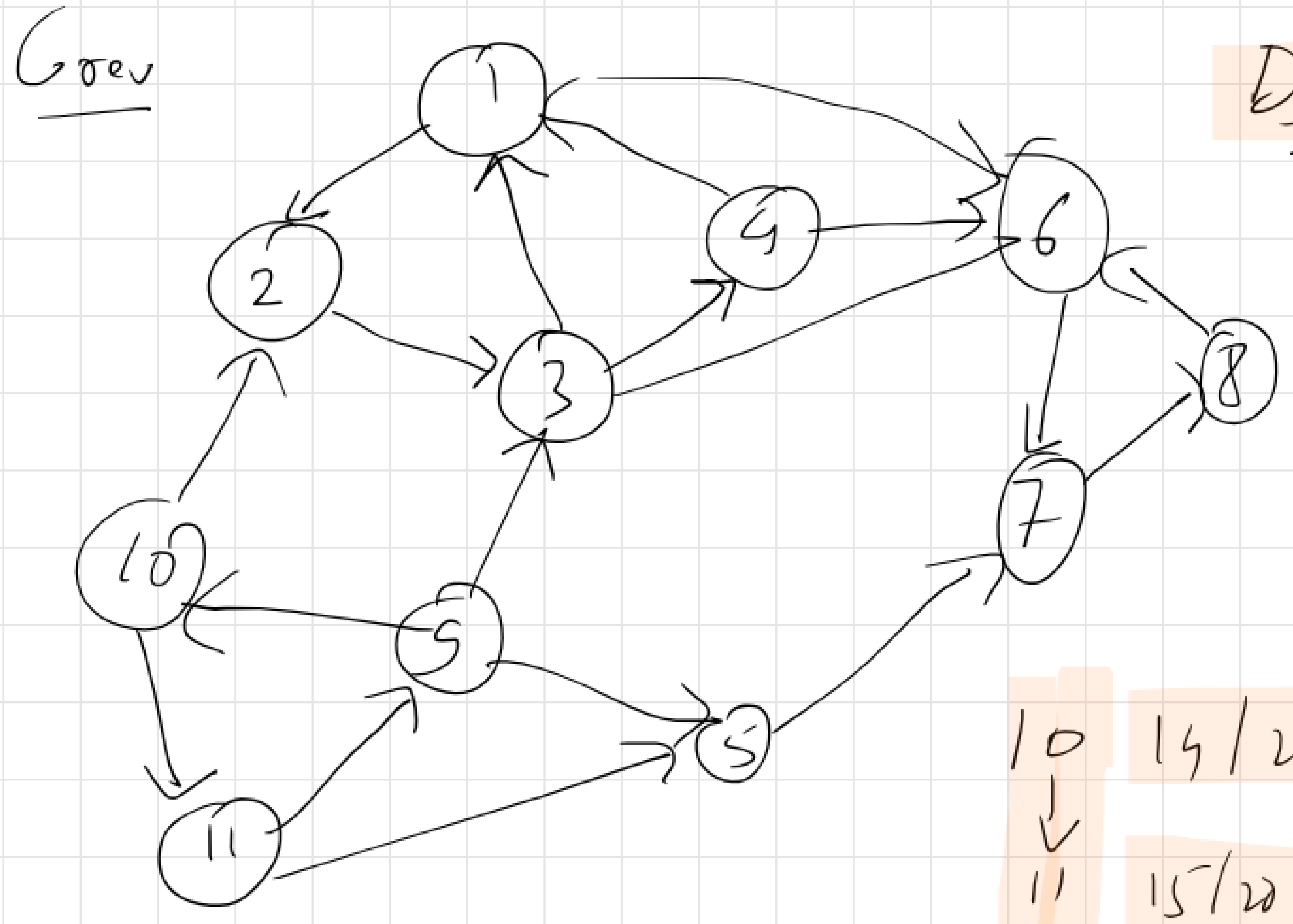
- Component graph is acyclic
- a component with zero indegree (proof)
- If indegree is zero, $f($for that component$)$ is maximum.

There will be a vertex in this component with max finish time

→ [for the same components in G
the out degree for that component
is zero]

▷ [We were searching this component
We got this] → So, finally we got a component whose out degree is zero

---



DFS(G^rev)

1	0/13
↓	
2	1/12
↓	
3	2/11
↓	
4	3/10
↓	
6	4/9
↓	
7	5/8
↓	
8	6/7

10	14/21
↓	
11	15/20
↓	
9	16/19
↓	
5	17/18

Sorted order finish time

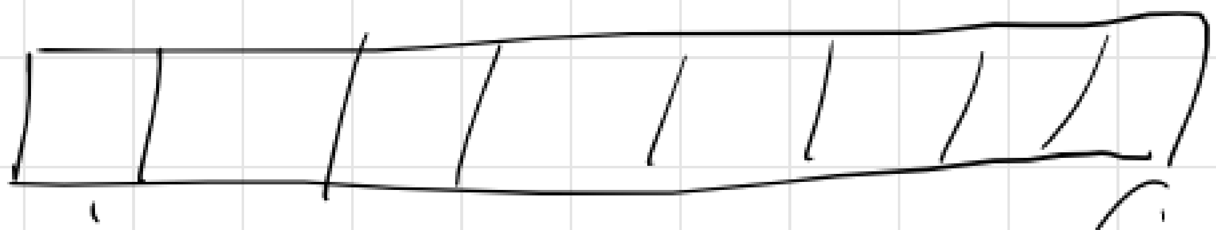
10	11	9	5	1	2	3	4	6	7	8

Do DFS (10) on G = 10, 9, 11 → SCC_1
DO DFS (5) on G : 5 → SCC_2
Do DFS (1) on G : 1, 2, 3, 4 → SCC_3
Do DFS (6) on G : 6, 7, 8 → SCC_4

Algorithm

→ DFS on G^{Rev}

→ Store nodes with respect to finish time (decreasing order)



nodes in finish order

→ Do DFS on G starting from node with highest finish time and continue.

The resultant is a DFS tree.
\# trees = # SCC

"BOTH ALGO ARE SAME"!

Cormen book has algorithm where DFS is done on G, then finish time collected, DFS on G^{Rev} is done w.r.t finish time.