Cpts 515. 9/11/2020

Today: SCC-Alg (Tarjan Alg) — Kosaraju's Alg.

← Princeton

John Hopkins

↓

strongly connected component.

← super efficient (linear time)

one of the authors of the linear time selection alg we learned last time.

(Blum, Floyd, Pratt, Rivest, Tarjan)

Intro: whenever you see "loops, cycles..." in a graph, you can use this alg.

⇒ next page

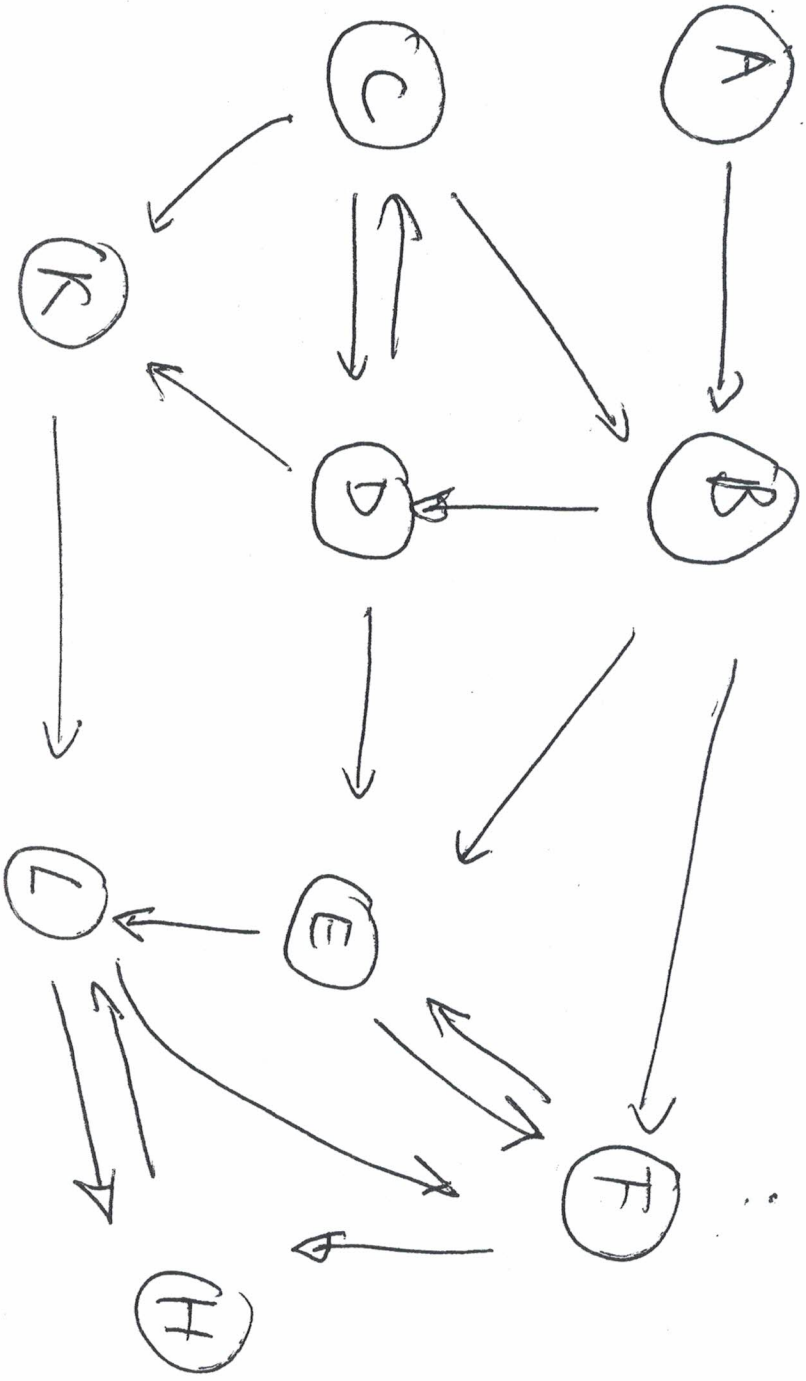
Matrix = Graph.

"connected"

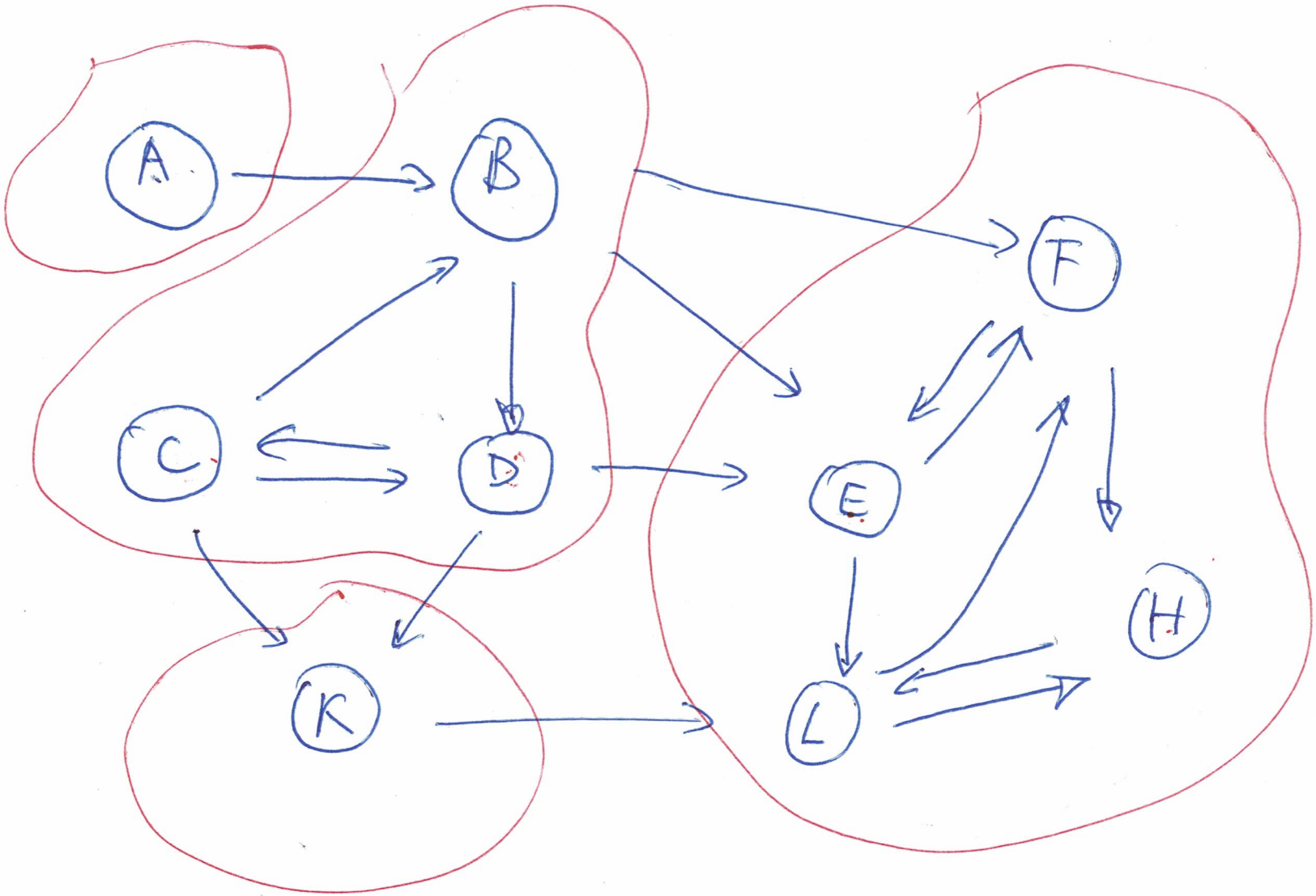
"adjacent":
"neighborhood",
transitive closure

scc of a graph.

Key of using SCC: you must know how to use it!



Exaple G:



This graph has 4 SCC's:

{A}, {B, C, D}, {K}, {E, F, L, H}.

Let G be a directed graph.

An SCC is a maximal subset of nodes in a given graph G sat:

(*) ~~for~~ for each node v and each node v', we have $v \rightsquigarrow v'$.

$\rightsquigarrow$ in 0 or more steps.

you can not add any new node to the subset s.t. the resulting subset still sat. the condition (*).

Observations: ① All SCC's are disjoint, and hence,

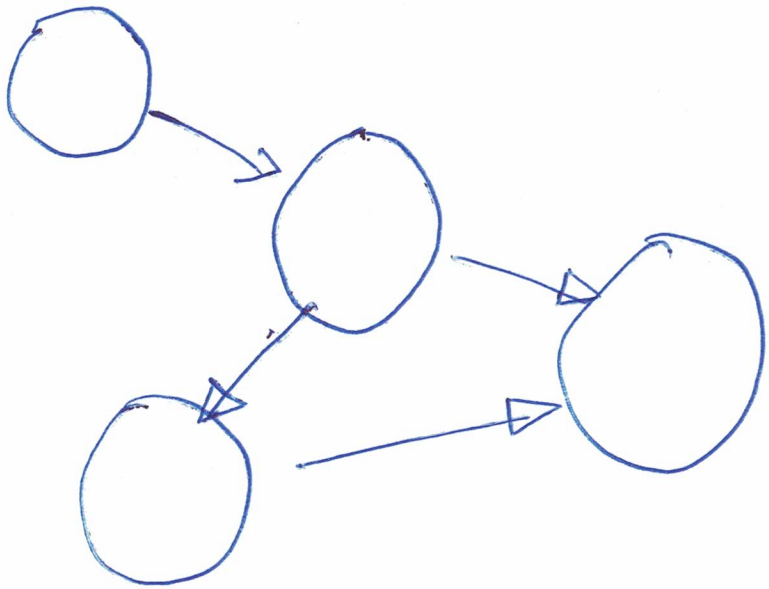
② All SCC's form a partition of all nodes in G.

SCC Alg is to output all SCC's of a graph.

---

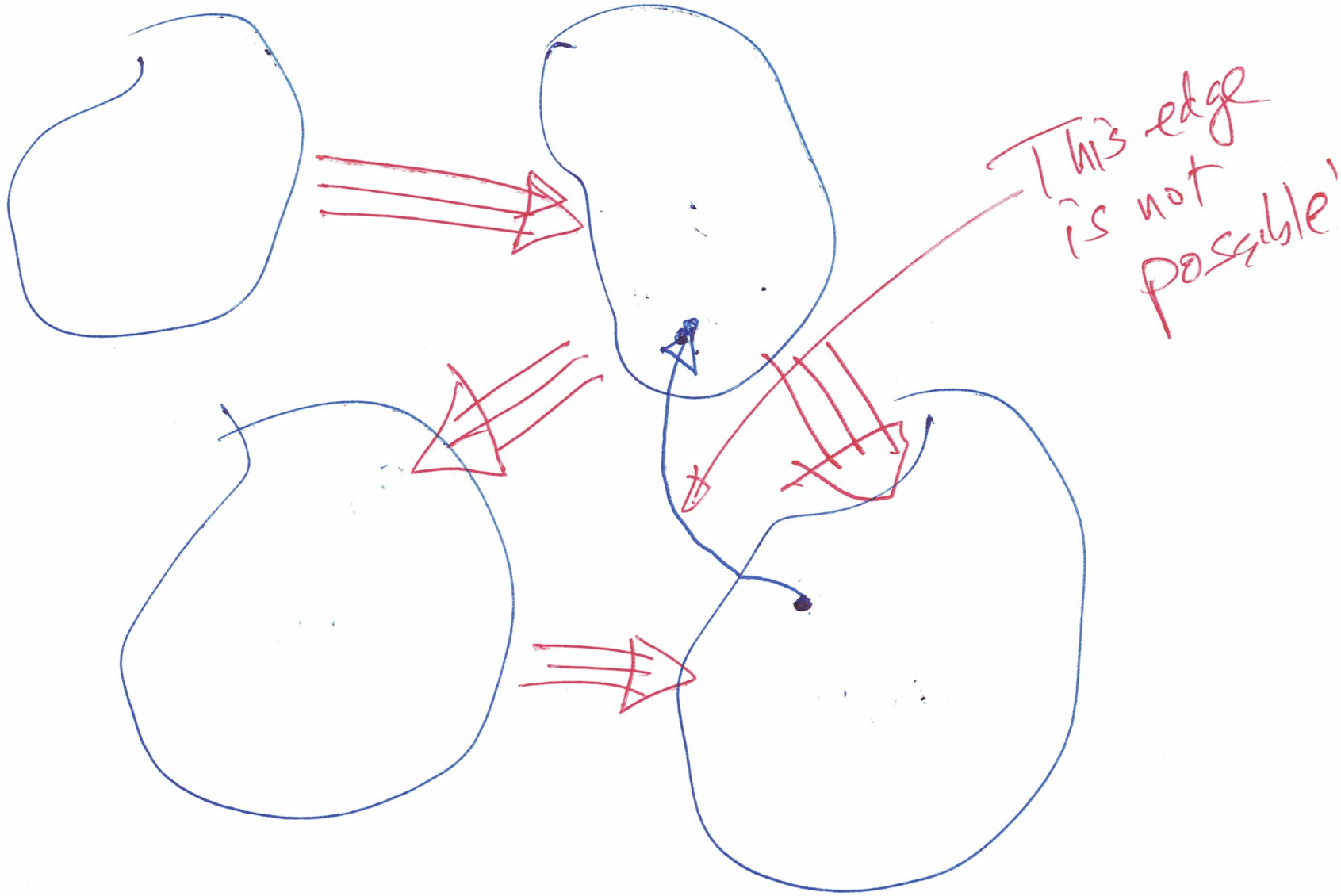
③ if I treat each SCC as a BigNode, then the graph is translated into a graph with BigNodes.

Translation from the Exaple graph G
to a Graph on Big Nodes:



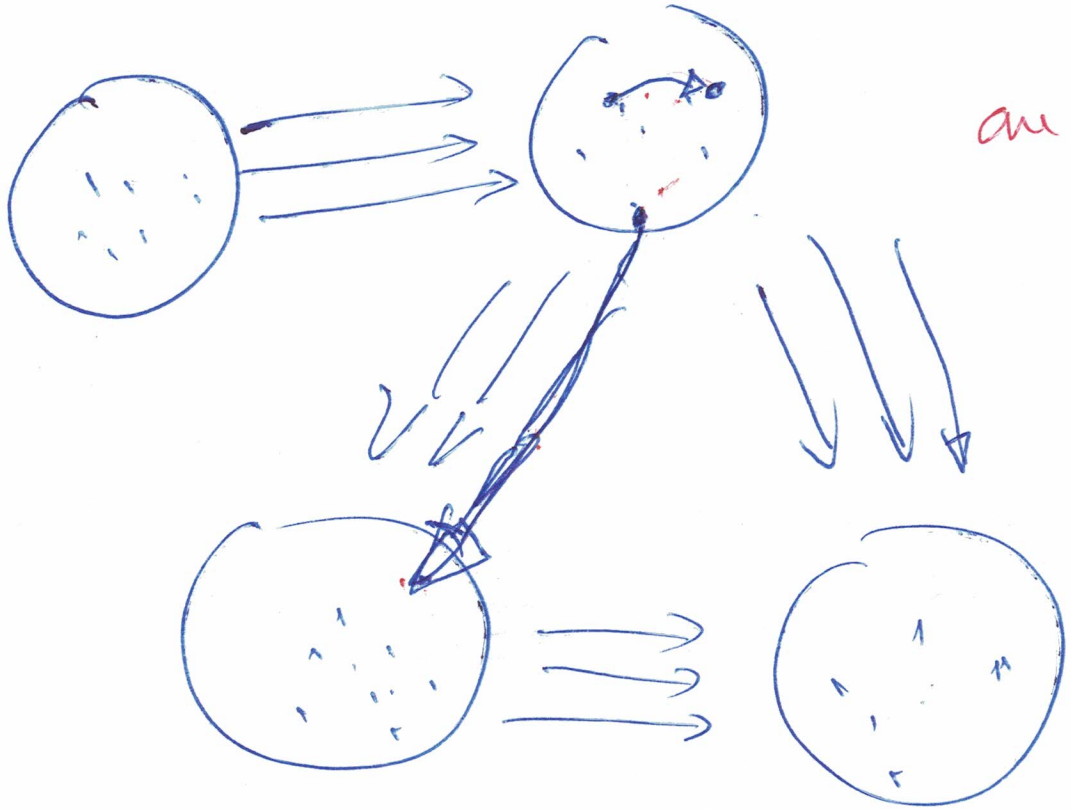
There is no loop in this translated graph!
We call it DAG.

What if you do have a loop in Graph?

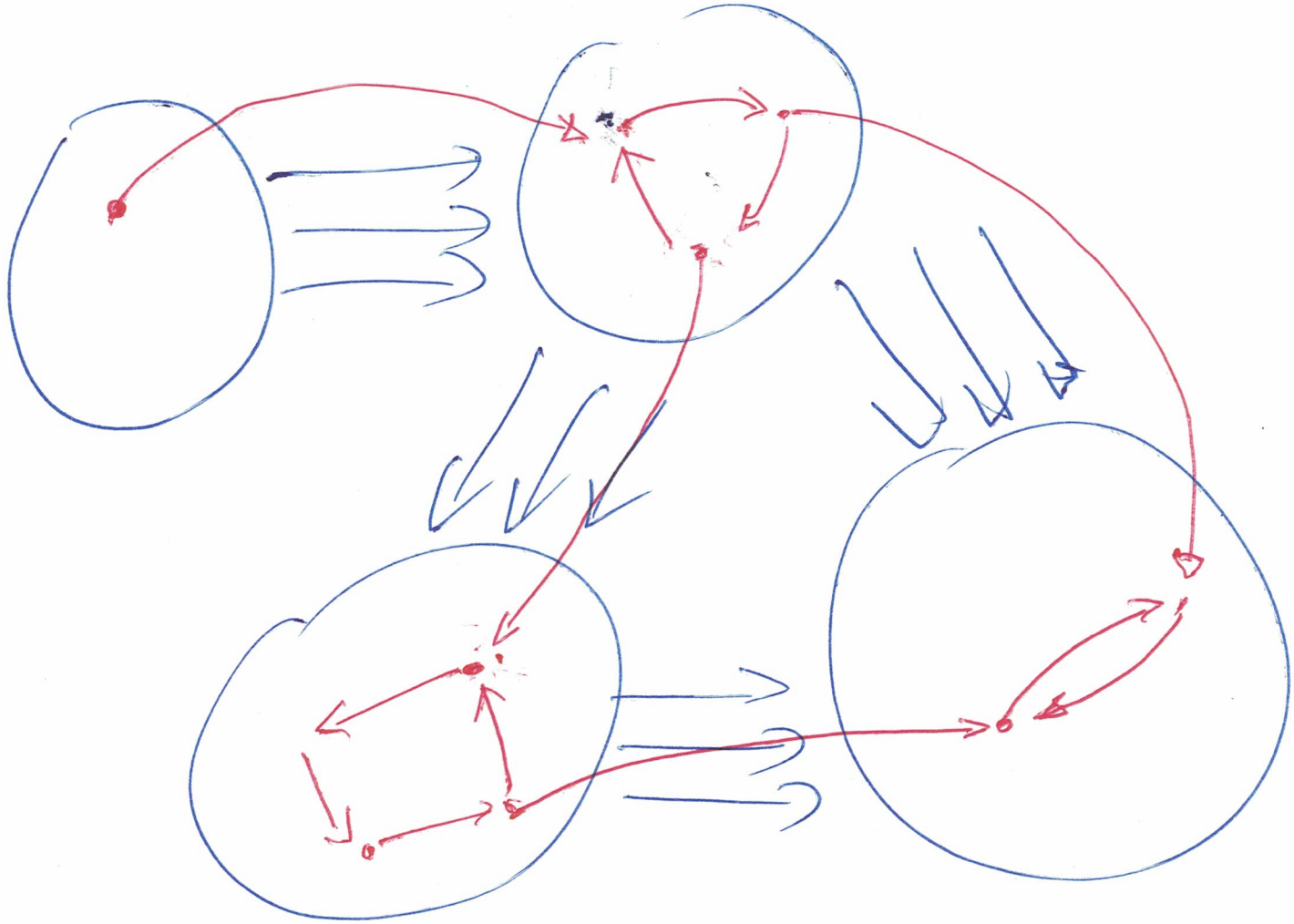


Alg's intuition ?

if you can distinguish between an edge inside an SCC and an edge crossing two SCC's, then you have all the SCC's.



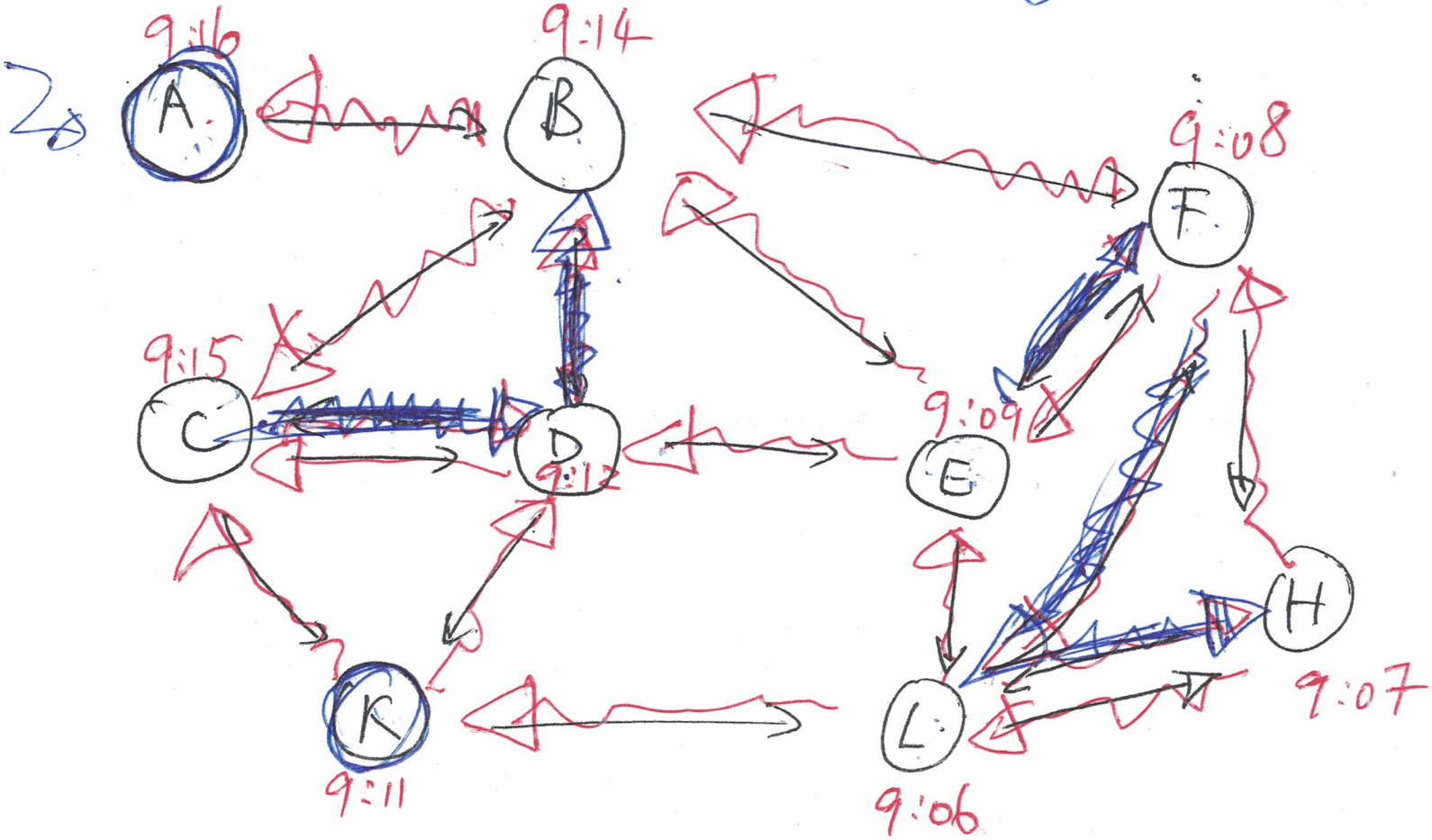
⟹ Figure out a property that can distinguish .....



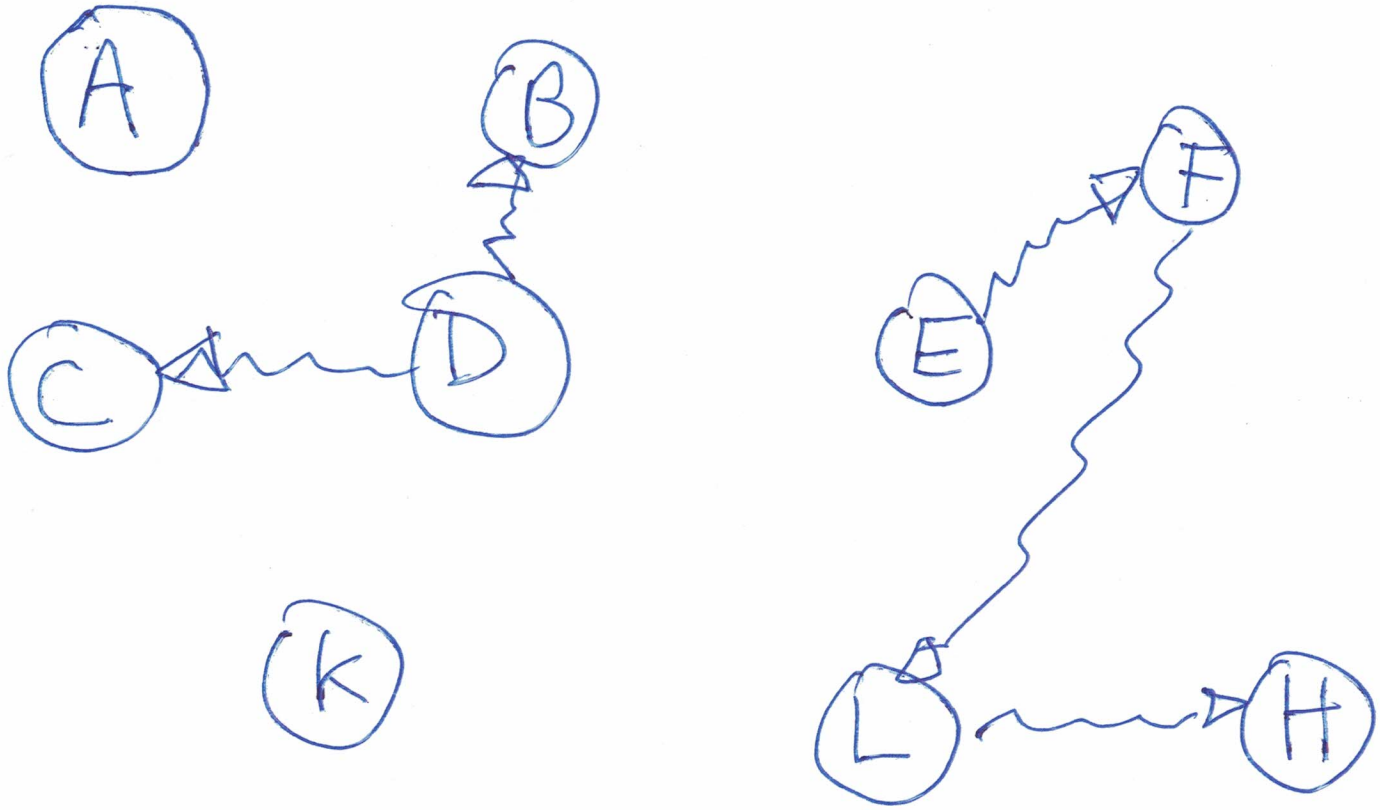
Reverse every edge in G!

A
9:16
C
9:15
K
9:11
9:01
9:13
9:10
D
9:12
B
9:14
9:02
L
9:06
E
9:09
9:03
9:05
F
9:08
H
9:07
9:04

Reverse every edge.
& DFS.



Two rounds of DFS.



C
A
K
D
B
I
E
F
H