Competitive Programming

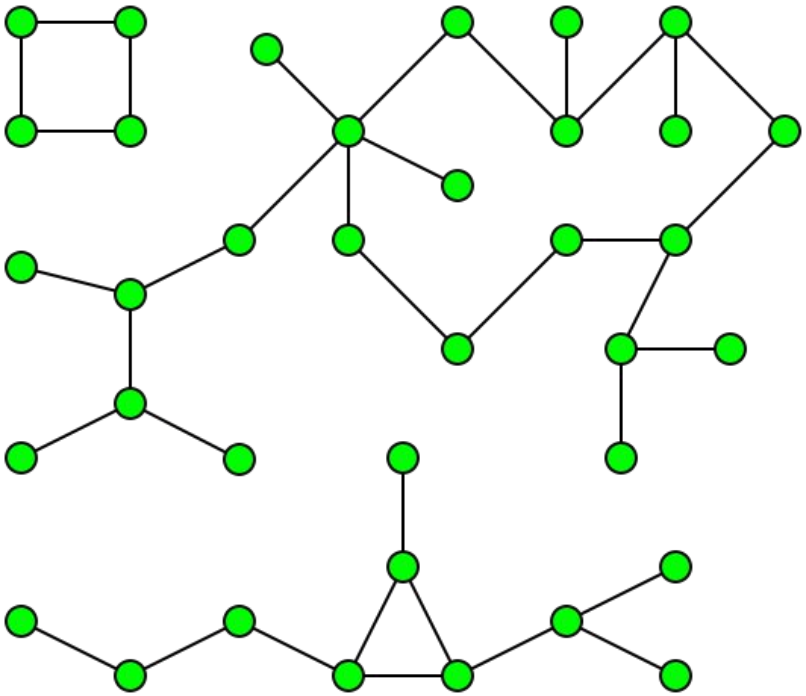
From Problem 2 Solution in O(1)

Graph Theory

Strongly CC using Tarjan - 1

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

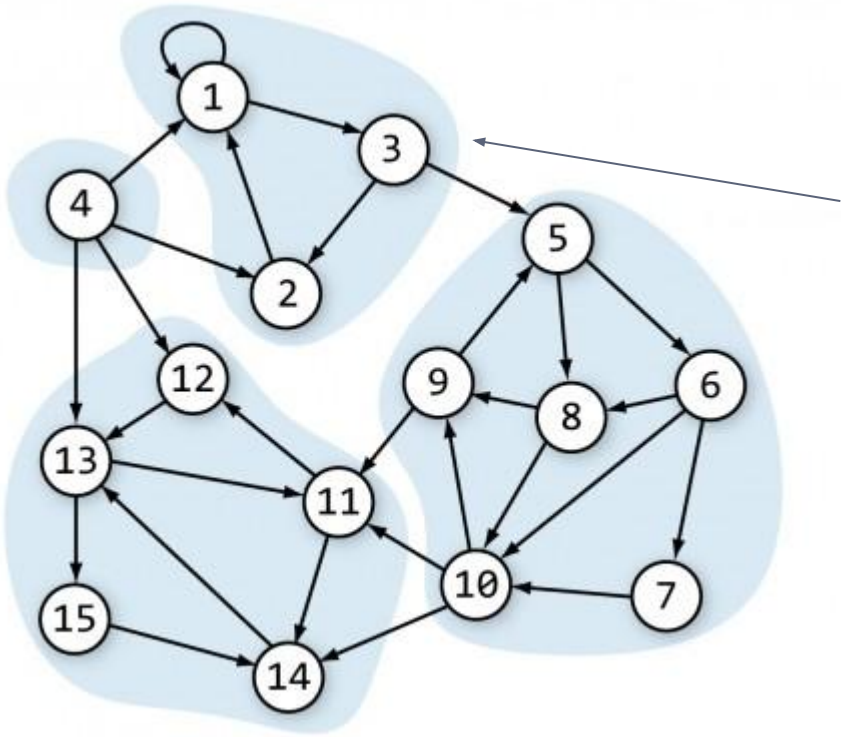
Connected Components



- **Undirected** Graph
- 3 CC (connected components)
- Each CC nodes can reach each other
- 1 DFS can find them
  - Call from each unvisited nodes

Src:https://en.wikipedia.org/wiki/Connected_component_%28graph_theory%29#/media/File:Pseudoforest.svg

Strongly Connected Components



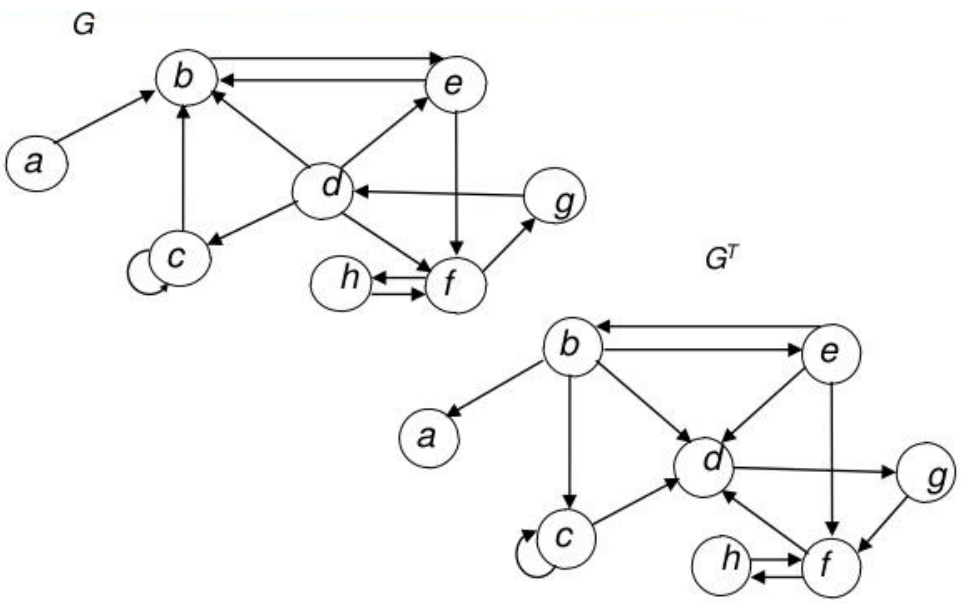
- **Directed** Graph
- 4 SCC
  - {4}
  - {1, 2, 3}
  - {5, 6, 7, 8, 9,. 10}
  - {11, 12, 13, 14, 15}
- Every pair of nodes can reach each others
- You can't add extra node to them
- Graph is SCG, if all nodes are 1 SCC

Src: https://www.greatandlittle.com/studios/index.php?post/2013/03/26/Strongly-Connected-Components-and-the-2-SAT-Problem-in-Dart

SCC Definition

- Strongly Connected Component of (SCC) a directed graph is a **maximal** set of **vertices** such that for **every pair** of vertices u are **reachable** from each other.
- Recall: Graph transpose: Switched each edge
  - E = (3, 7) $\Rightarrow$ E' = (7, 3)
- Graph and its transpose have the same SCCs

Graph Transpose

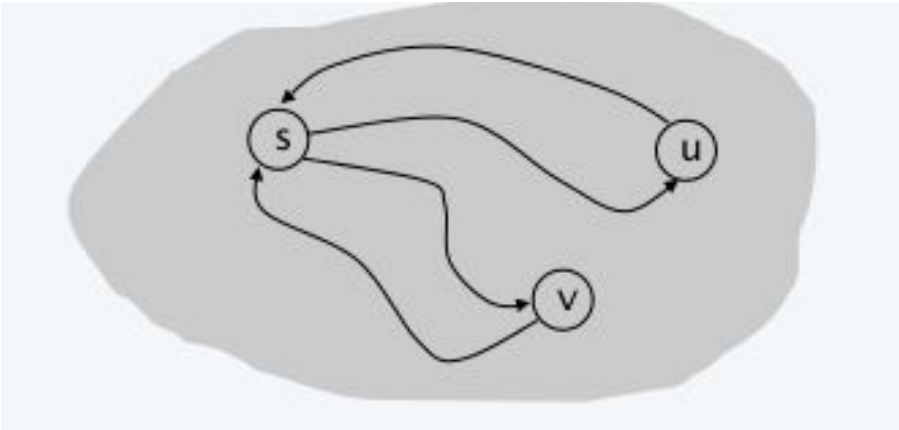


- G: SCC all nodes except a
- G' has same SCC

Src: http://image.slidesharecdn.com/lecture16-140420071804-phpapp02/95/topological-sort-22-638.jpg?cb=1397978457

Strongly Connected Graph

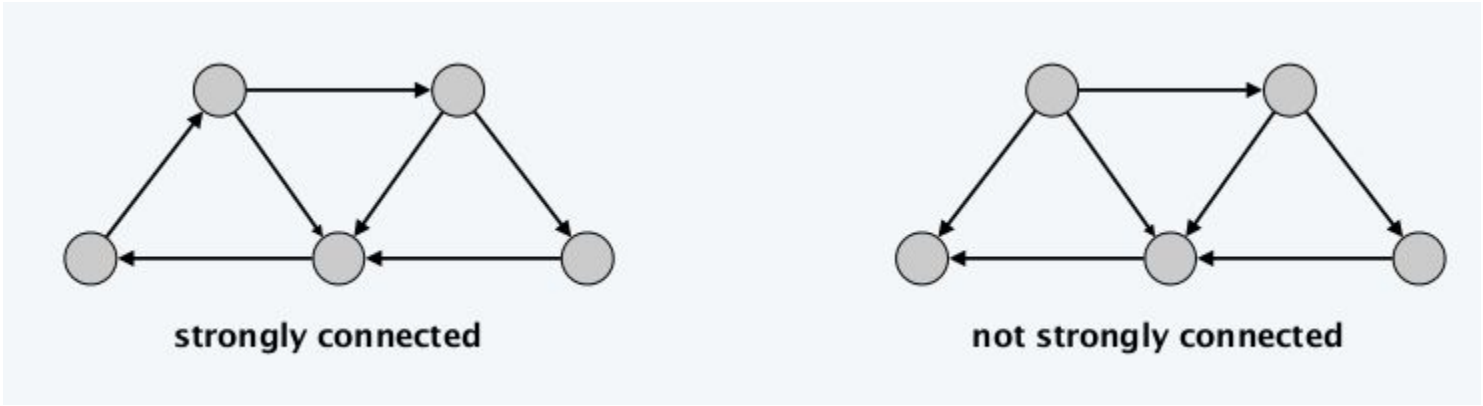
- IFF for some node s
  - S reaches every node...every node reaches S
  - Then for any pair (u, v): (u, s, v) and (v, s, u) exist



Src:http://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/03Graphs.pdf

Strongly Connected Graph

- Test SCG?
  - Pick any node s.
  - Run DFS from s in G.
  - Run DFS from s in **G Transpose**.
  - Are all nodes reachable in both cases?

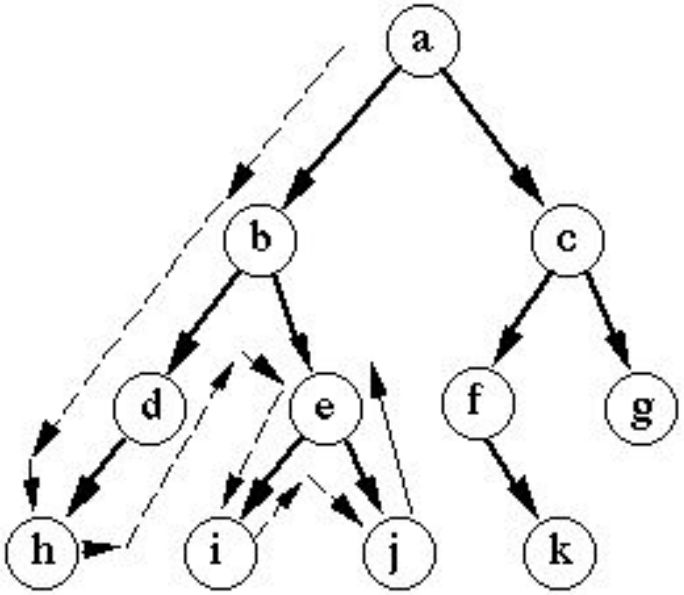


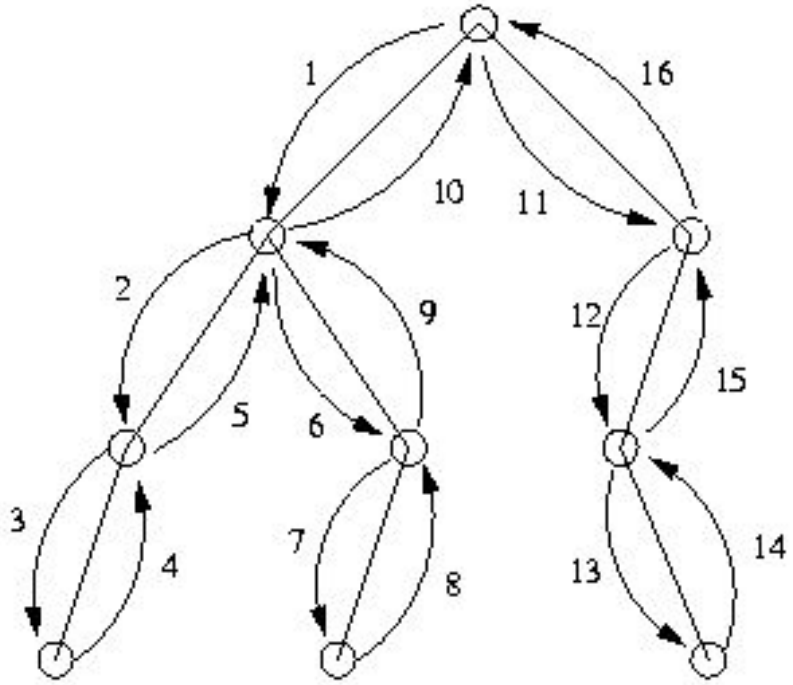
Src:http://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/03Graphs.pdf

Algorithms for SCC

- Run DFS from each node i, and mark in 2D array each node j you reach: **reach**[i][j]
  - Then if reach[i][j] AND reach[j][i] => (i, j) in same SCC
  - O(V * (V+E)) => Worst $O(V^3)$
- Run Floyd warshal to compete **reach** in $O(V^3)$
- Kosaraju's algorithm
  - 2 DFS calls. Easy to code/understand. Won't explain.
- **Tarjan**
  - 1 DFS only. Compute **Also** bridges and articulation
  - Little harder to get. But worth studying that Kosaraju

Depth first search traversal

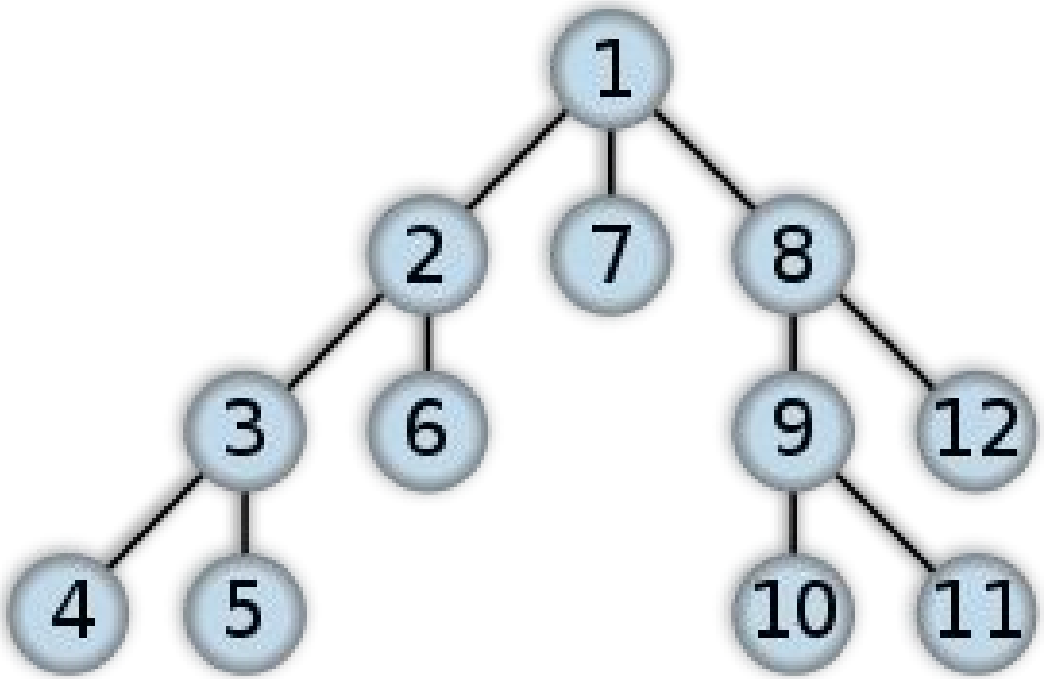




Src:
http://www.cse.unsw.edu.au/~billw/Justsearch.html
https://codersupremo.files.wordpress.com/2013/11/rec2.gif

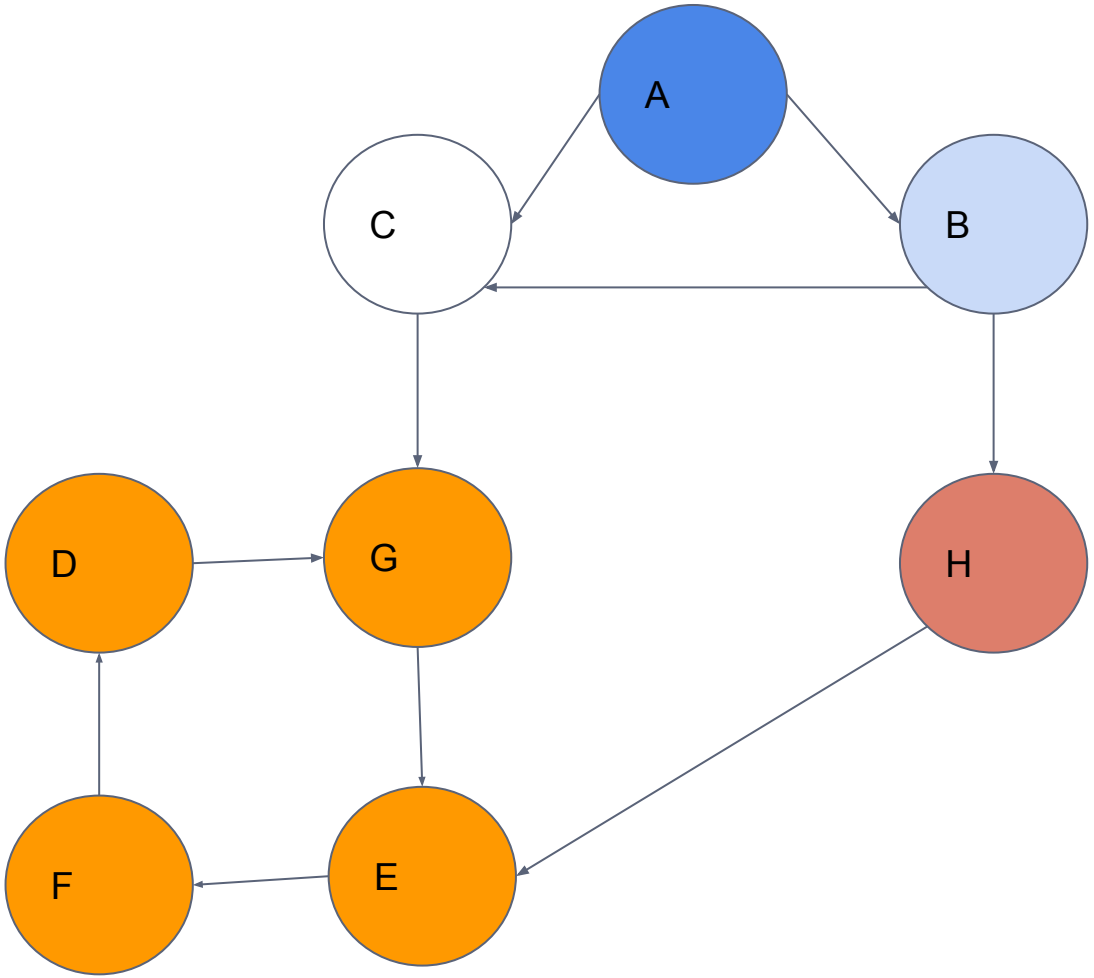
Depth first search number (DFS#)

Assigned number (index) based on visiting time for node



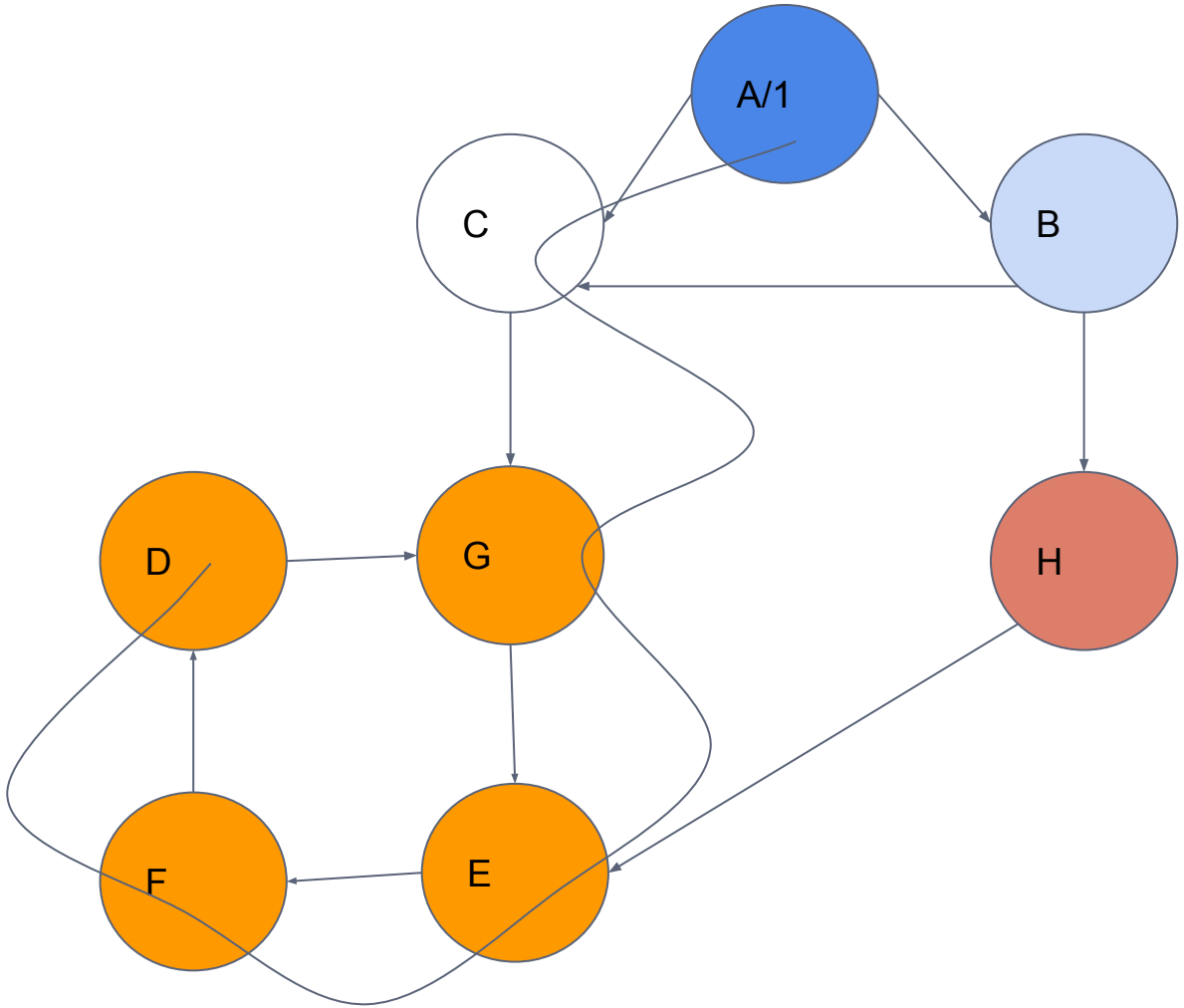
Src:
https://upload.wikimedia.org/wikipedia/commons/thumb/1/1f/Depth-first-tree.svg/300px-Depth-first-tree.svg.png

What are the SCC?



Graph Src:
http://www.programming-algorithms.net/image/id/44234

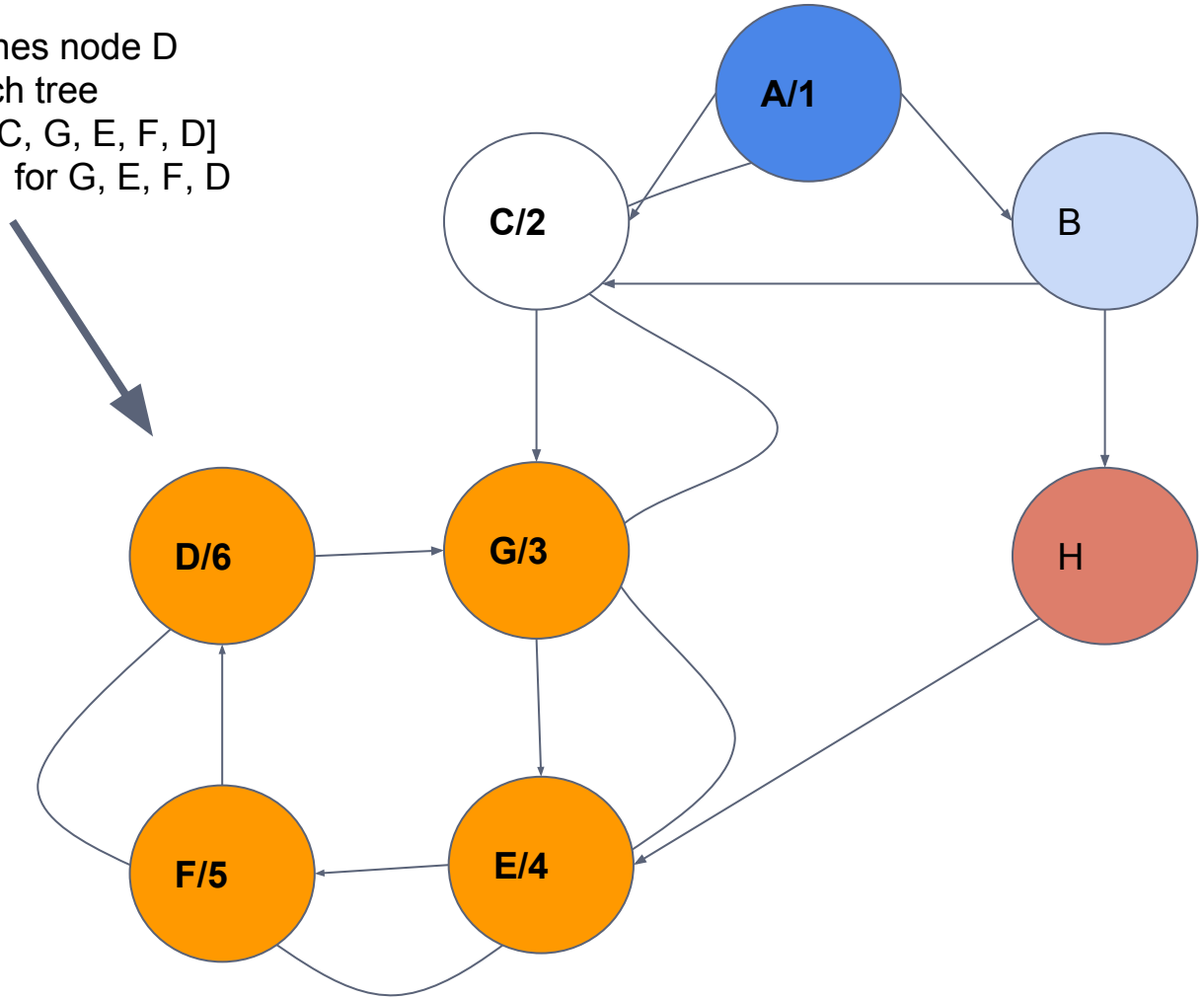
What are the DFS#?



Graph Src:
http://www.programming-algorithms.net/image/id/44234

What are the DFS#?

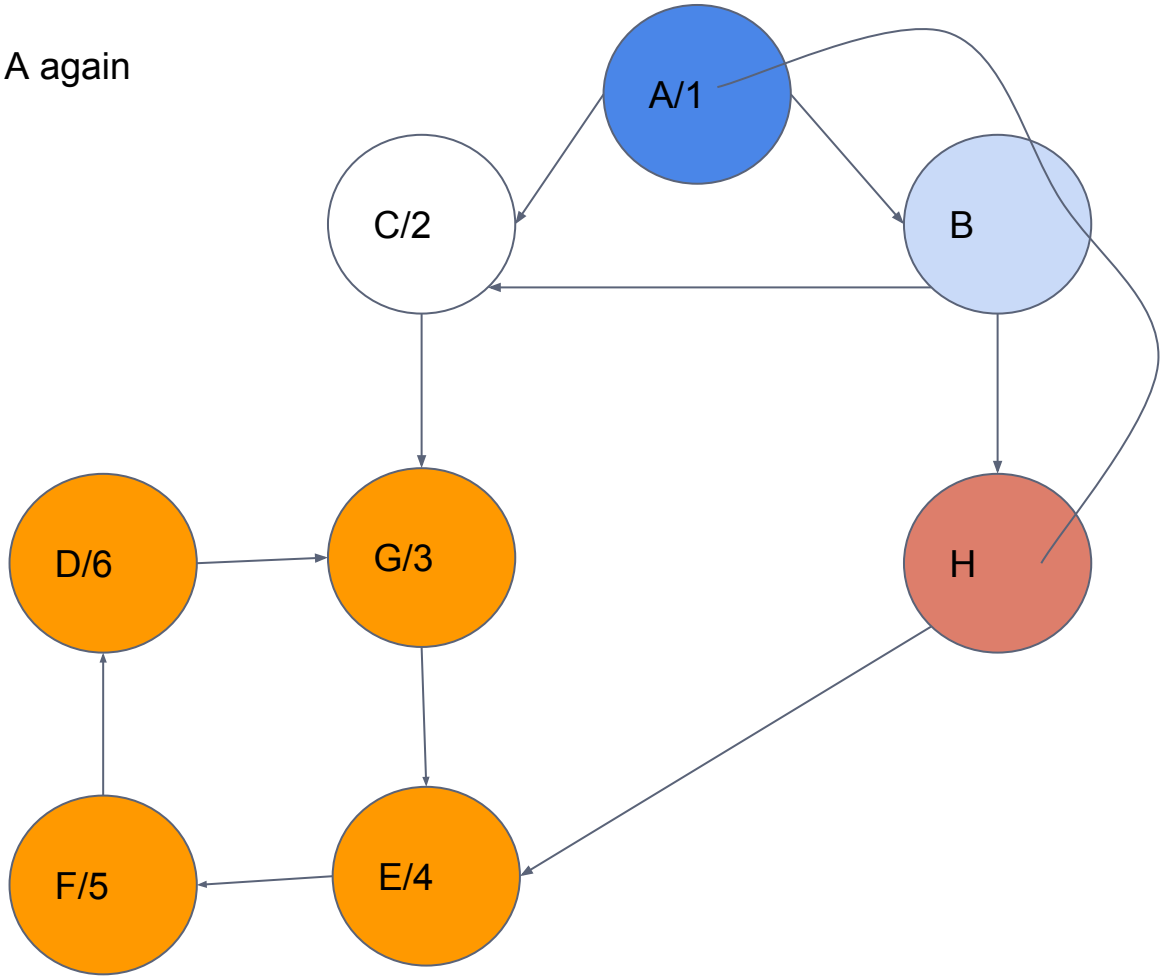
When DFS reaches node D
We have a search tree
Stack nodes [A, C, G, E, F, D]
E.g. C **ancestor** for G, E, F, D



Graph Src:
http://www.programming-algorithms.net/image/id/44234

What are the DFS#?

Now, search backtrack to A again
to start NEW search tree
Stack nodes [A]

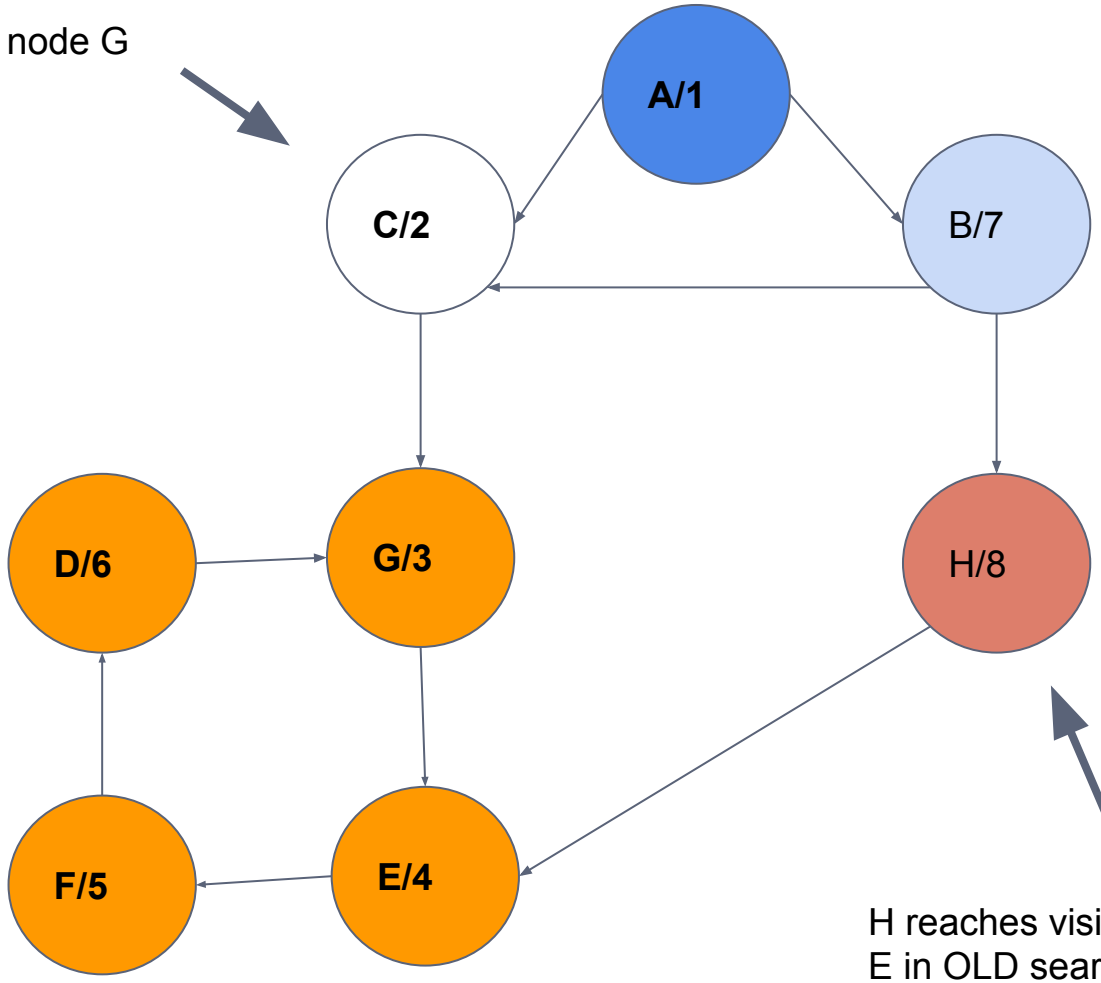


Graph Src:
http://www.programming-algorithms.net/image/id/44234

Observe DFS# between nodes

C is parent of unvisited node G
DFS#(C) < DFS#(G)

D reaches visited node G
G in same search tree
G is ancestor to D
DFS#(G) < DFS#(G)
G and F are in **cycle**



H reaches visited node E
E in OLD search tree
E is NOT ancestor to H
E and H are **NOT** in cycle

Graph Src:
http://www.programming-algorithms.net/image/id/44234

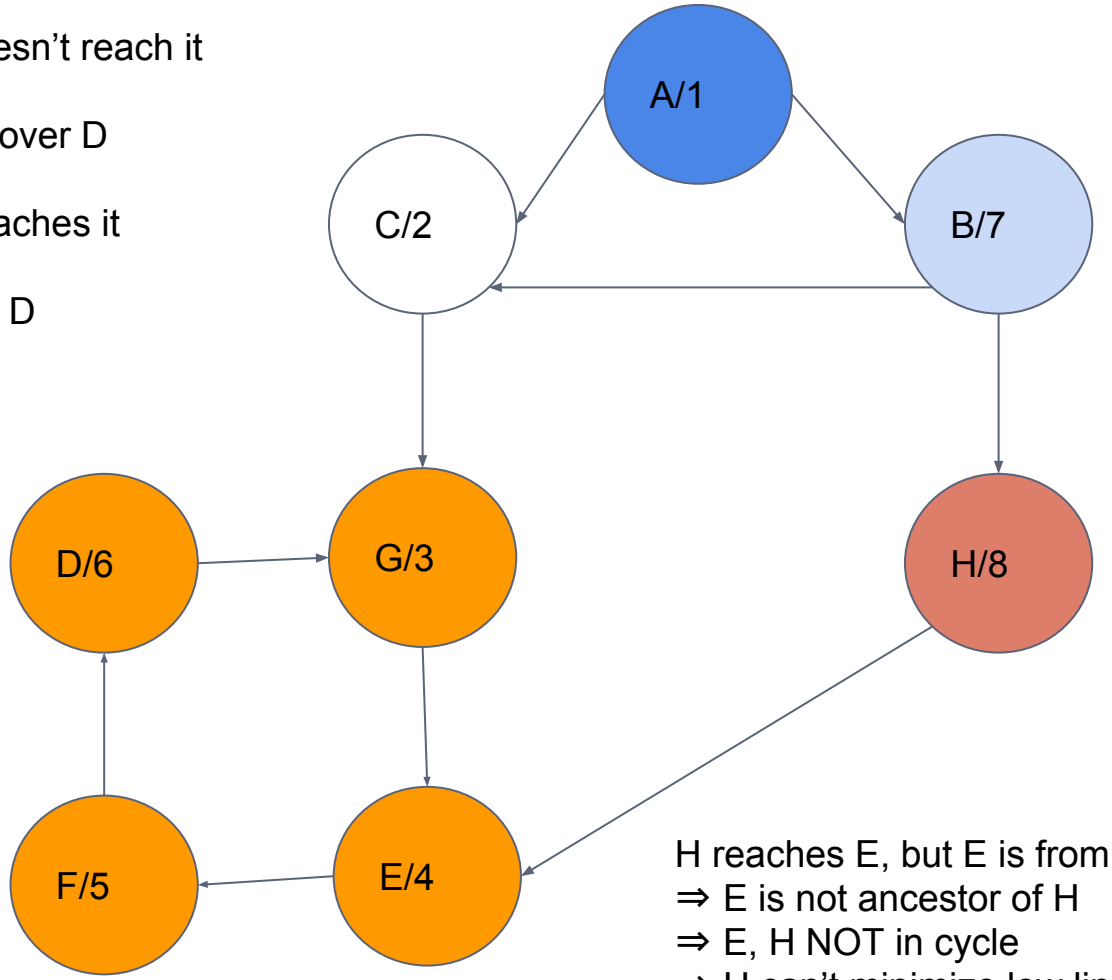
What are the LowestLink #?

- Each node A can reach some other nodes B
  - If B is higher **DFS#**, then it is my child
  - **If B lower, in current search tree = B is ancestor to A**
  - If B lower, in old search tree =backtracked already
- Lowest Link number =
  - min number of my **ancestors** nodes i **reach** including me
  - In other words, **upper nodes** in current DFS **search tree**
- If A is **ancestor** of B & B **reaches** A = Cycle
- Recall: In SCC, all nodes are in a cycle
  - Only 1 node will have DFS# = LowestLink#
  - It has no ancestors that it can reach

What are the LowestLink #?

C is ancestor of D, but D doesn't reach it
⇒ C, D NOT in cycle
⇒ C can't minimize low link over D

E is ancestor of D, and D reaches it
⇒ E, D in cycle
⇒ E minimizes low link over D

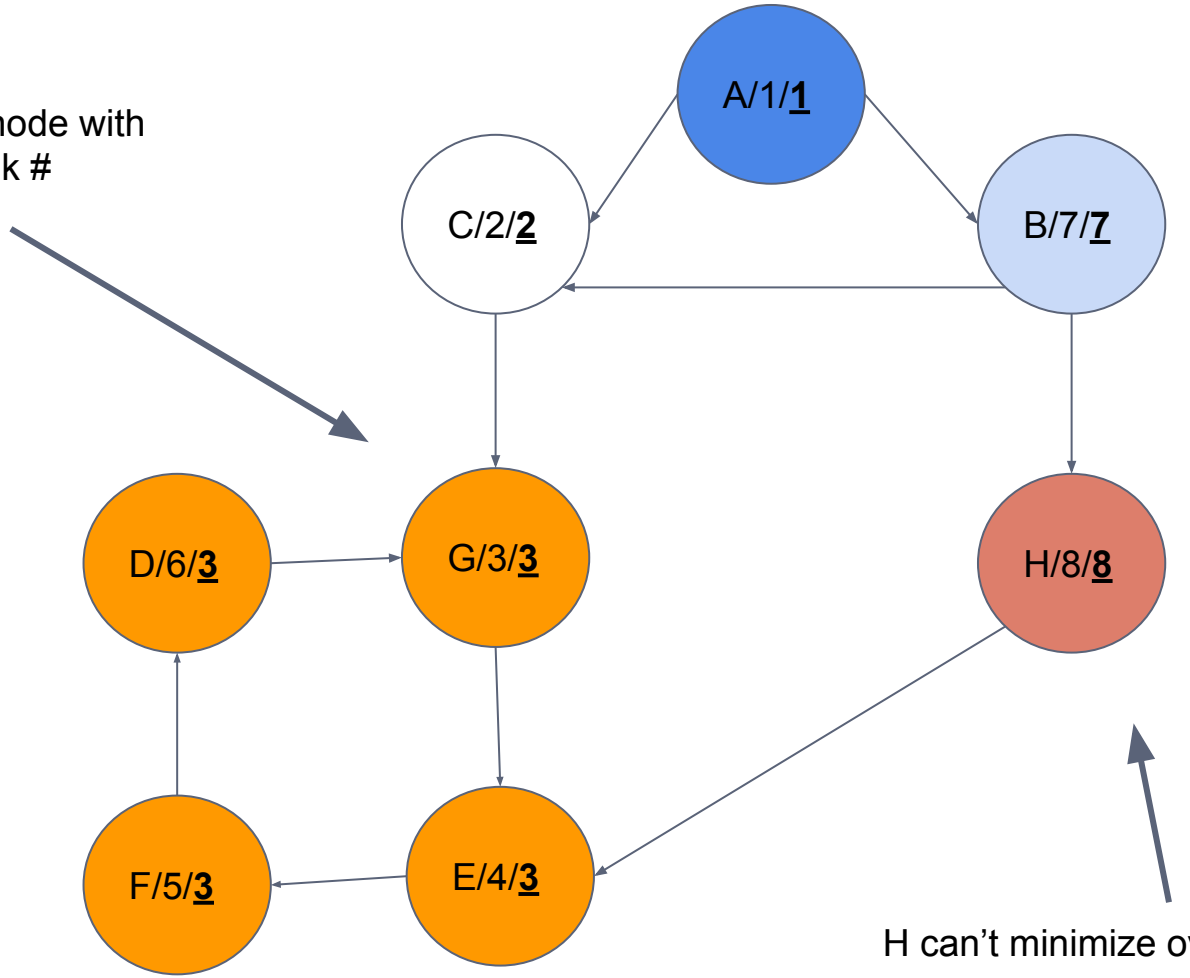


H reaches E, but E is from old search tree
⇒ E is not ancestor of H
⇒ E, H NOT in cycle
⇒ H can't minimize low link over E

Graph Src:
http://www.programming-algorithms.net/image/id/44234

What are the LowestLink #?

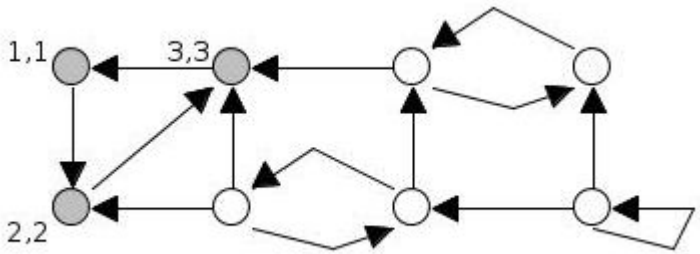
In SCC, only 1 node with
DFS # = LowLink #

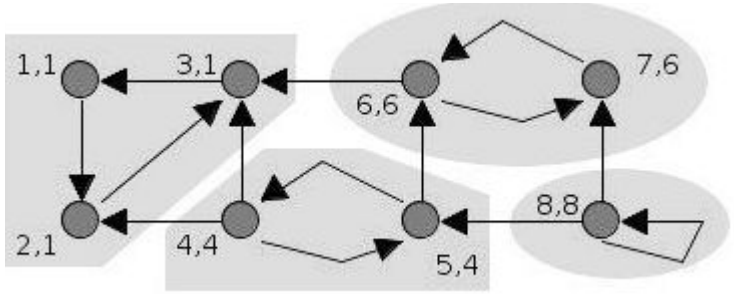


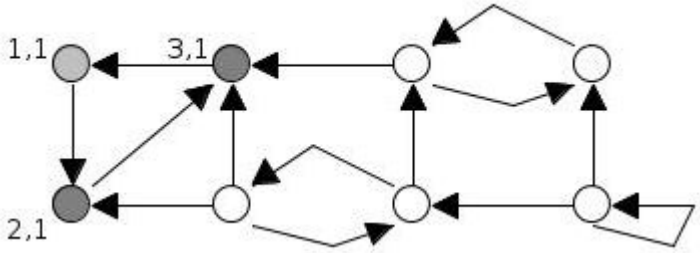
H can't minimize over E low link

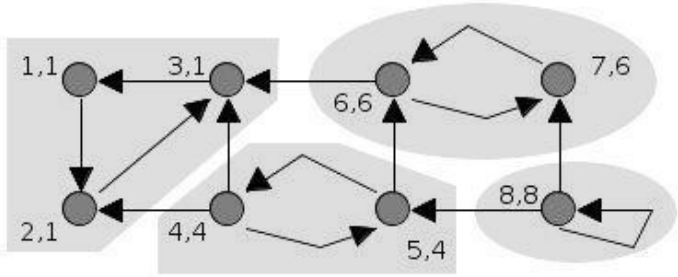
Graph Src:
http://www.programming-algorithms.net/image/id/44234

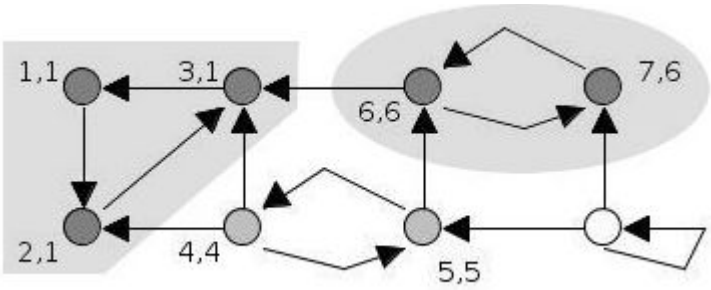
Wiki example





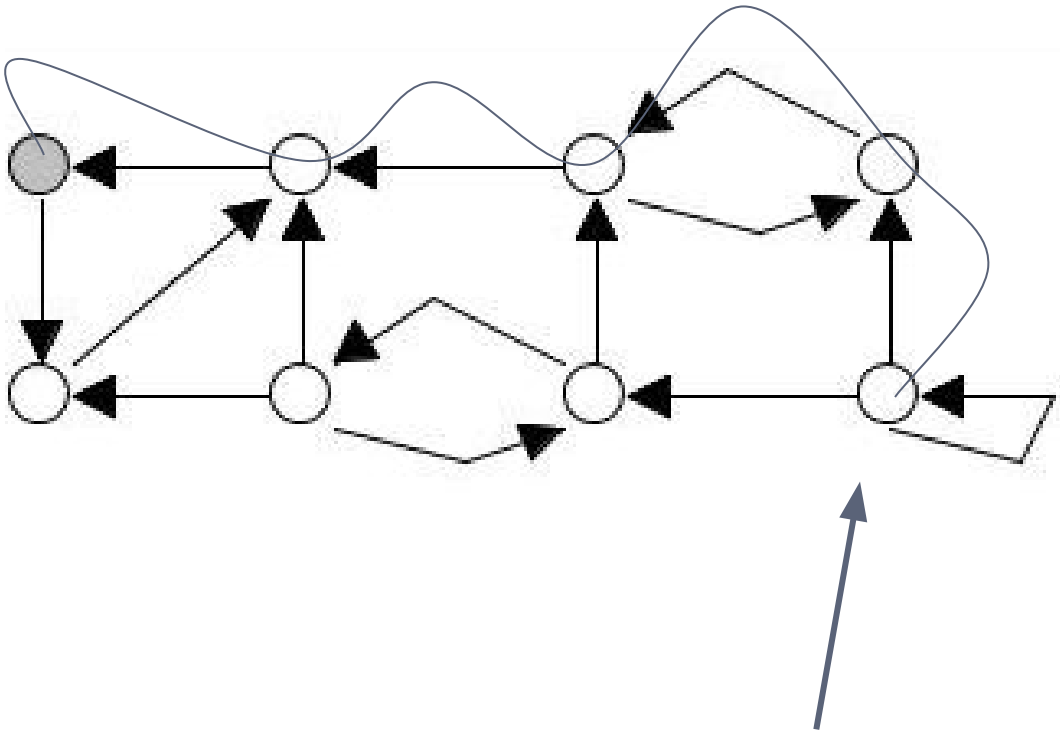






Notice:
Search starts from 1
Restart from 4
Restart from 8

Your turn



LowestLink # by DFS

- Implement the 3 cases in a DFS
- If child node is unvisited?
  - Let ur child find recursively its LowLink #
  - Then simply minimize parent LowLink# with child one
- If child node is visited?
  - Either in current stack (ancestor = cycle) ⇒ minimize
  - Or it is not in stack ⇒ Old search tree ⇒ Ignore

Getting the SCC

- After computing LowLink#, you can simply group each set of nodes of same value as SCC
- Tarjan has a stack invariant to get SCC
  - Have a stack..push node in it during search
  - Before going out of dfs, check if DFS# = LowLink#
  - If yes, all nodes in stack from this node is SCC
  - pop them all
  - Note, in normal DFS we will pop always after dfs end

Tarjan Algorithms

- Based on observations on DFS
- Run a DFS
- Index the numbers by visit time
- Observe relations between DFS #
  - unvisited case, visited back edge, visited previous tree
- Introduce Lowest Link Value
  - All cycle members has same value = min DFS #
  - Value express min reachable ancestor
- Each group of same LinkLow is SCC

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً