NCKU Programming Contest Training Course
Strong Connected Component(SCC)
2018/04/25

Syuan-Yi Lin(petermouse)
petermouselin@gmail.com

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

made by
tommy5198 & free999 &
electorn & petermouse

Outline

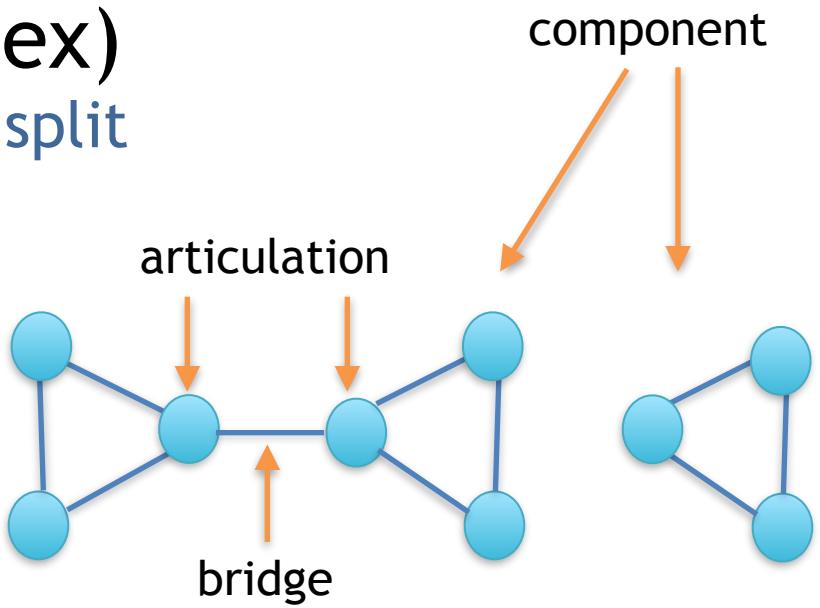
- Articulation/Bridge
  (in undirected graph)

- Strongly Connected Component(SCC)
  (in directed graph)

made by tommy5198
& petermouse

Graph

- connected graph/component
  connected graph/component iff all pairwise vertices exist at least one path & no more vertices can be added
- articulation(cut-vertex)
  remove articulation vertex split one component to two
- bridge(cut-edge)
  same as articulation



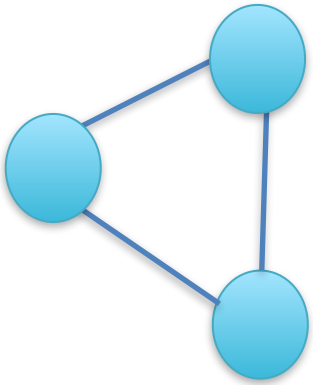
made by tommy5198

Articulation/Bridge

- Find Articulation in Graph
  - Graph become non-connected if remove a Articulation.
    V times DFS = O(V*(V+E)) —> too slow!

  - Vertex is not Articulation if can find alternative path
    —> find cycle!

  - Use DFS —> O(V+E)



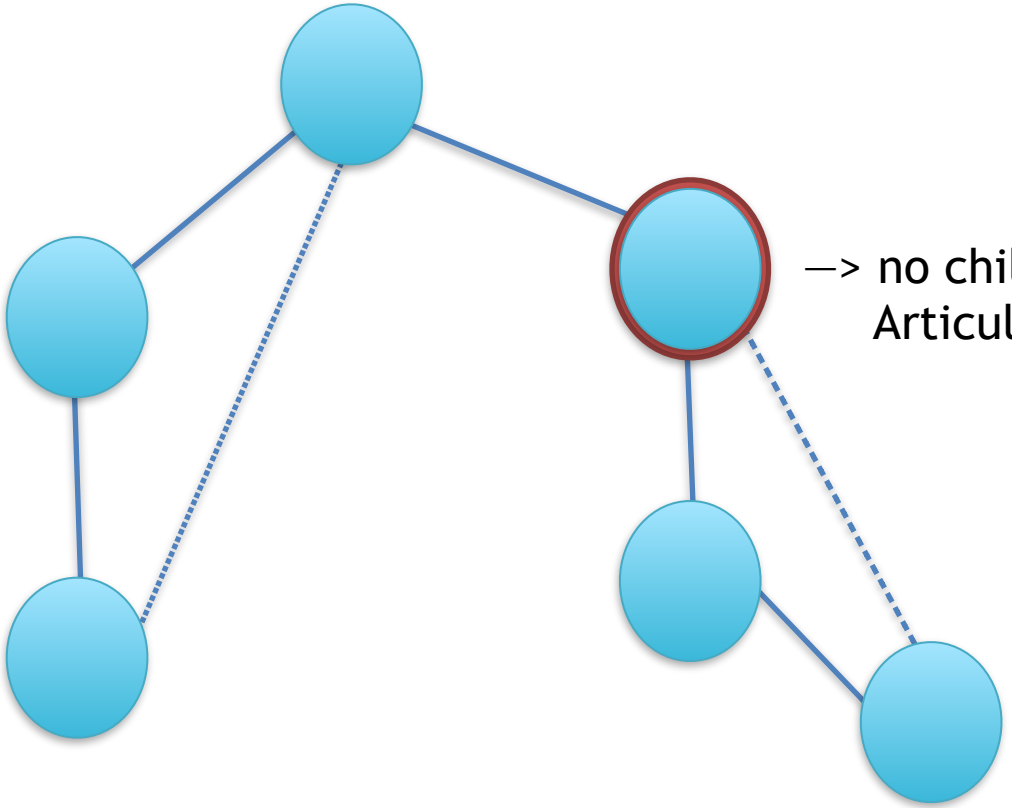
Articulation/Bridge

- Concept
  - DFS traversel will construct a DFS tree
  - if vertex u's children can't back to u's ancestors
    —> u is Articulation
  - if vertex u is root and has at least 2 child
    —> u is Articulation

- Bridge?
  - two Articulation u, v have an edge —> (u, v) is Bridge!

made by tommy5198
& petermouse

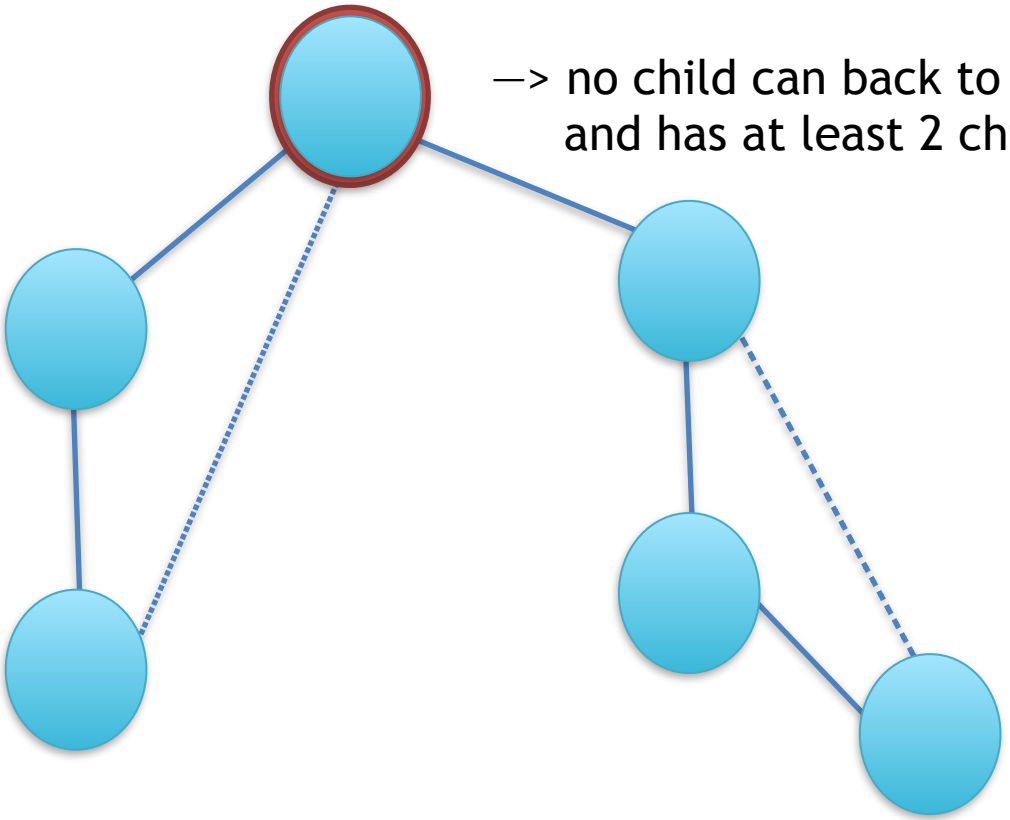
Articulation/Bridge



—> no child can back to ancestor.
Articulation!

made by tommy5198

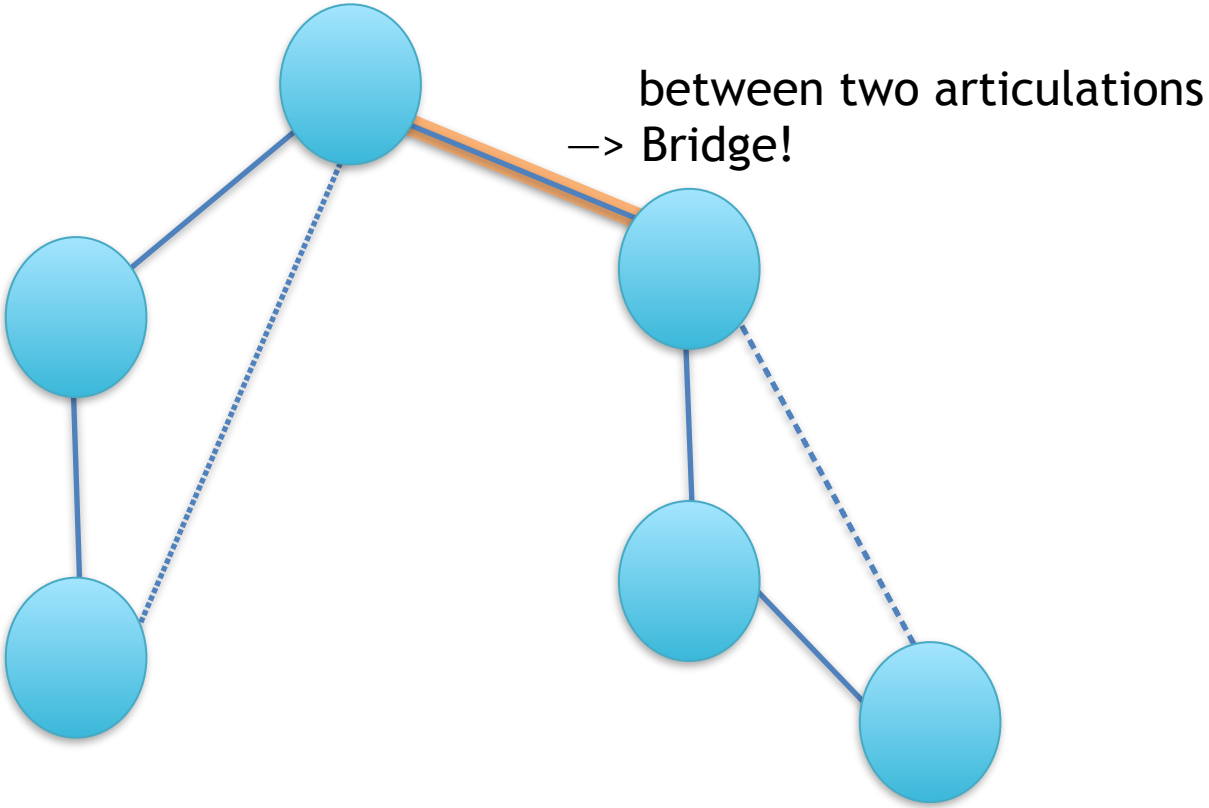
Articulation/Bridge



—> no child can back to ancestor
and has at least 2 child. Articulation!

made by tommy5198

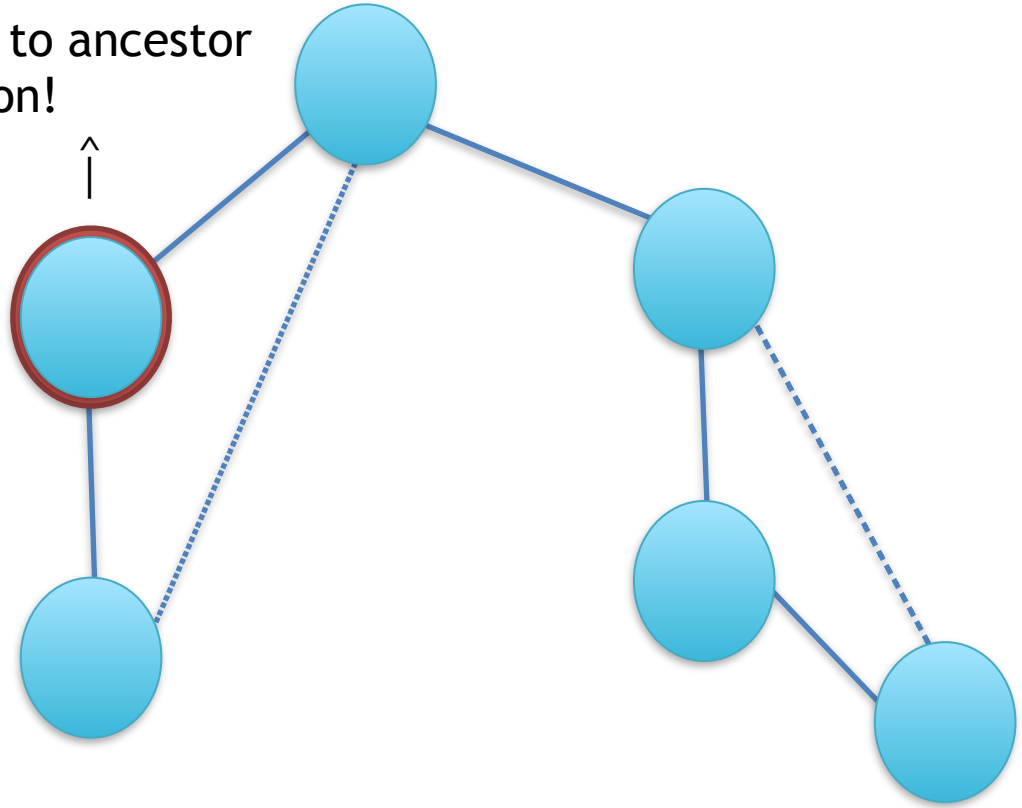
Articulation/Bridge



made by tommy5198

Articulation/Bridge

child can back to ancestor
NOT Articulation!



made by tommy5198

Articulation/Bridge

- dfn[u] = DFS traversal order
  - first visit time each vertex u in DFS

- low[u] = min(dfn[u], lowest low[v])
  - if edge (u,v) exist and v is not u's parent

Articulation/Bridge

- Articulation
  - if vertex u's children can't back to u's ancestors
  —> dfn[u] <= low[v], v is u's child

  - if vertex u is root and has at least 2 child
  —> count child >= 2
- Bridge?
  - two Articulation u, v —> dfn[u] < low[v], v is u's child

made by tommy5198

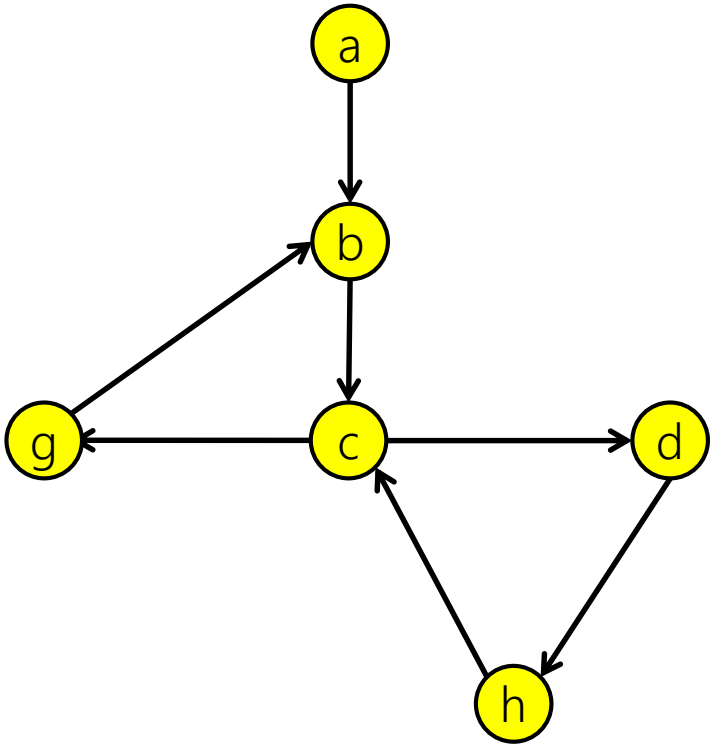
Cut vertex

- code

```
23  void DFS(int prev,int cur)
24  {
25      bool cut = false;
26      int child = 0;
27      low[cur] = dfn[cur] = ++dfsn;
28      for(int idx = adj_list[cur]; ~idx; idx = edge[idx].next) {
29          if(!dfn[edge[idx].to]) {
30              DFS(cur, edge[idx].to);
31              low[cur] = min(low[cur], low[edge[idx].to]);
32              if(low[edge[idx].to] >= dfn[cur])
33                  cut = true;
34              ++child;
35          } else if(edge[idx].to != prev)
36              low[cur] = min(low[cur], dfn[edge[idx].to]);
37      }
38      if((child > 1 || prev != -1) && cut)
39          ans++;
40  }
```

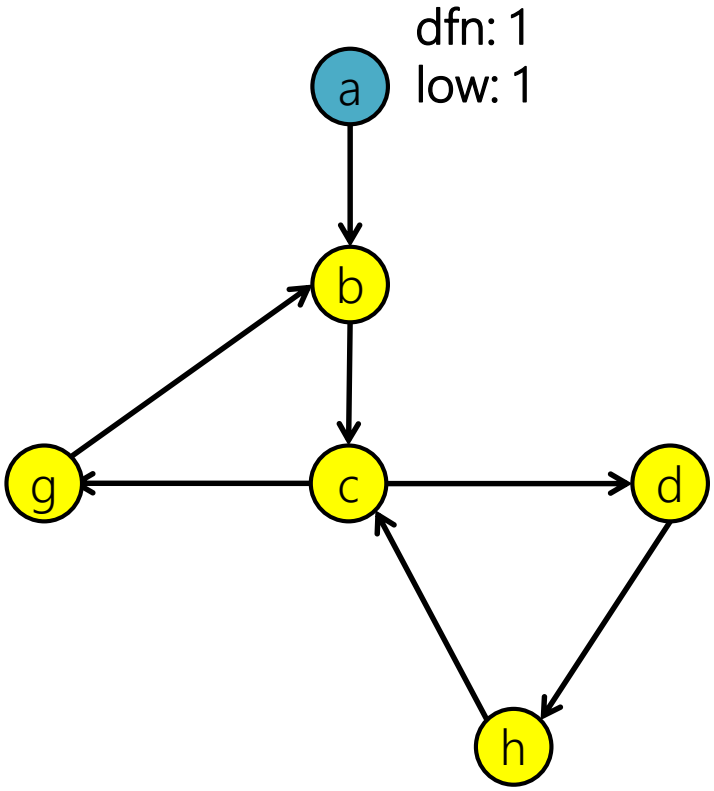

made by petermouse

Cut vertex

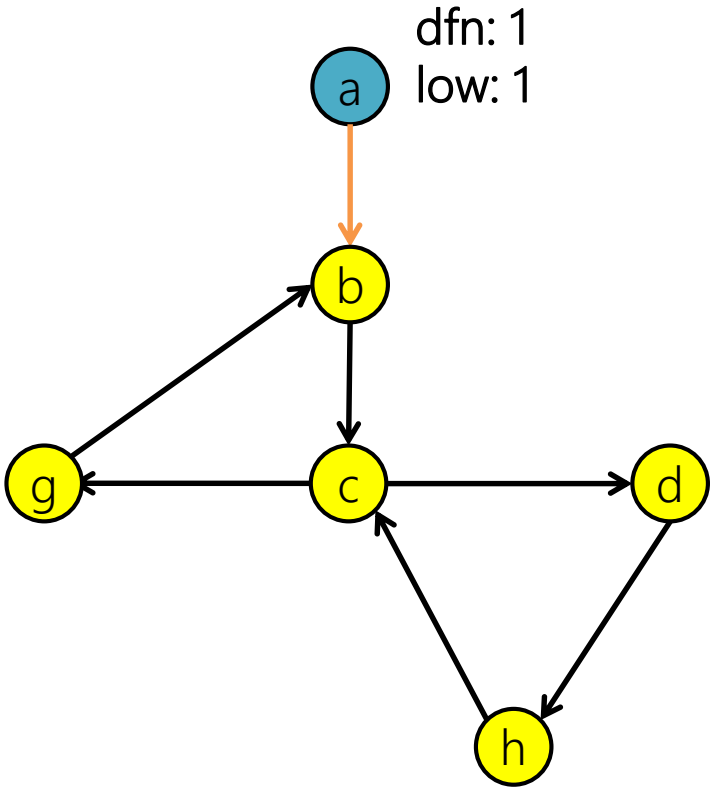


Note: This is an undirected graph

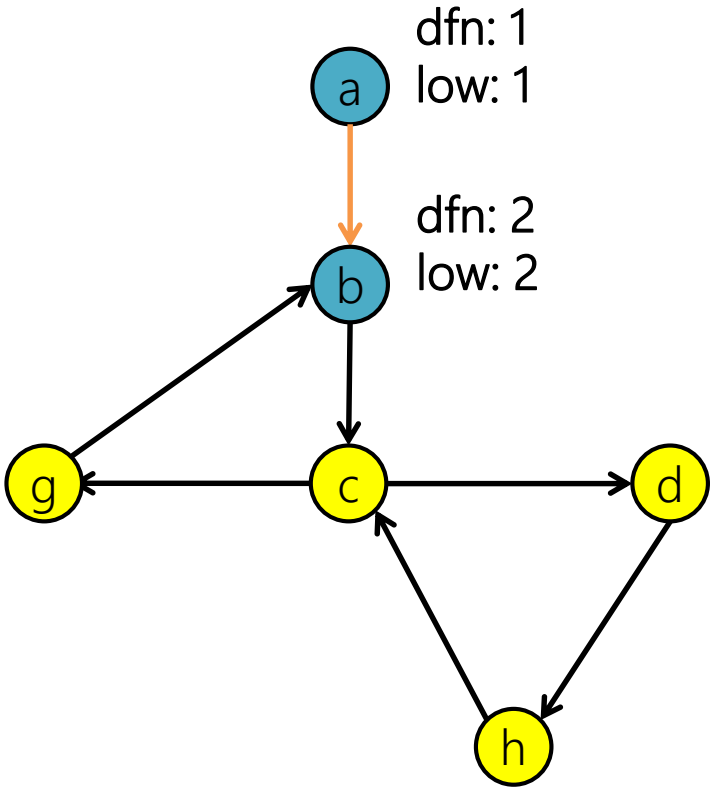
Cut vertex



Cut vertex

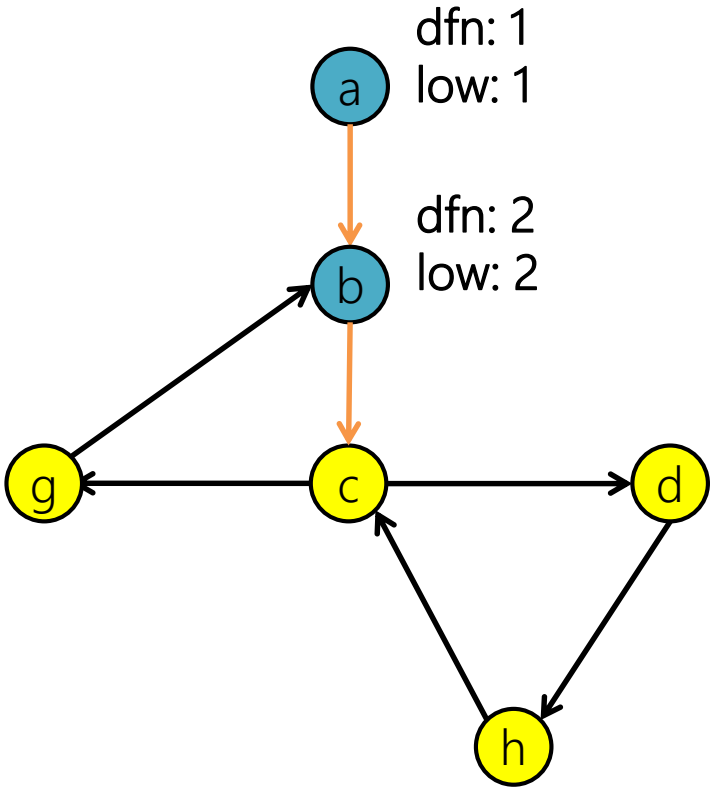


Cut vertex



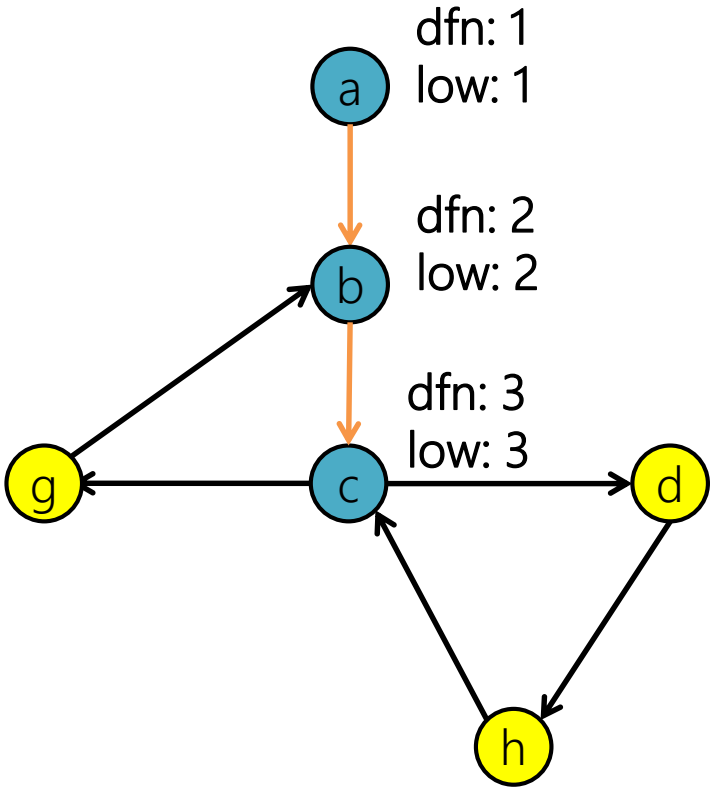
made by riljian

Cut vertex

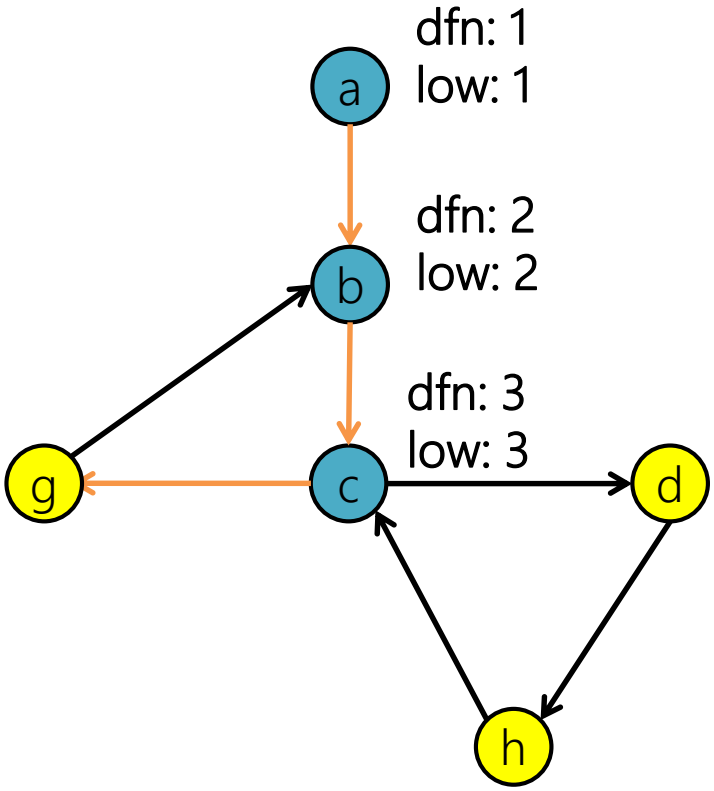


made by riljian

Cut vertex

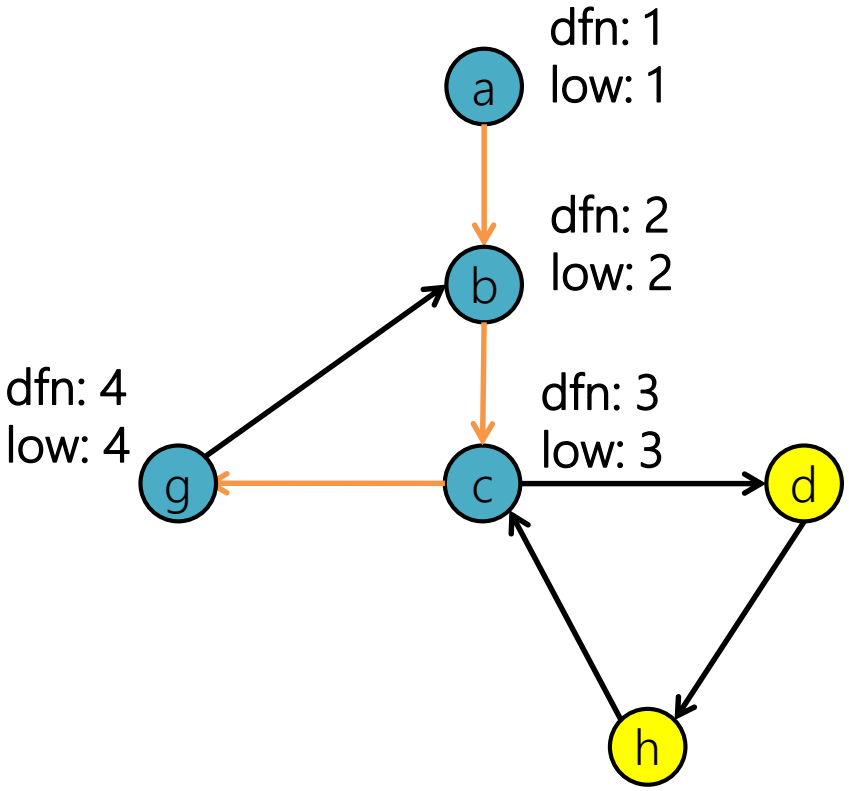


Cut vertex

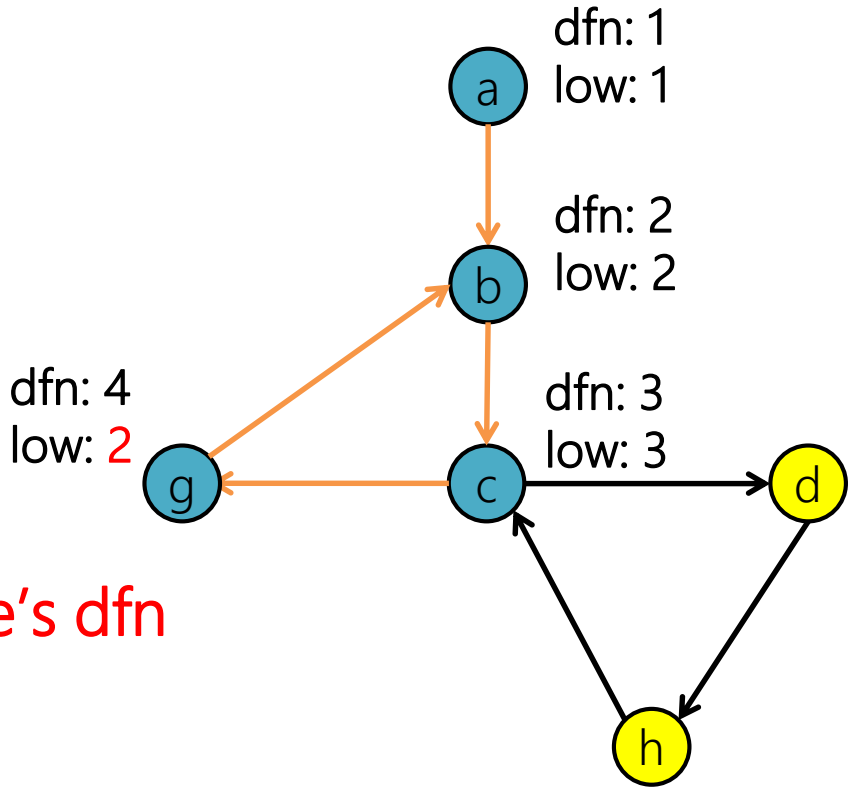


made by riljian

Cut vertex



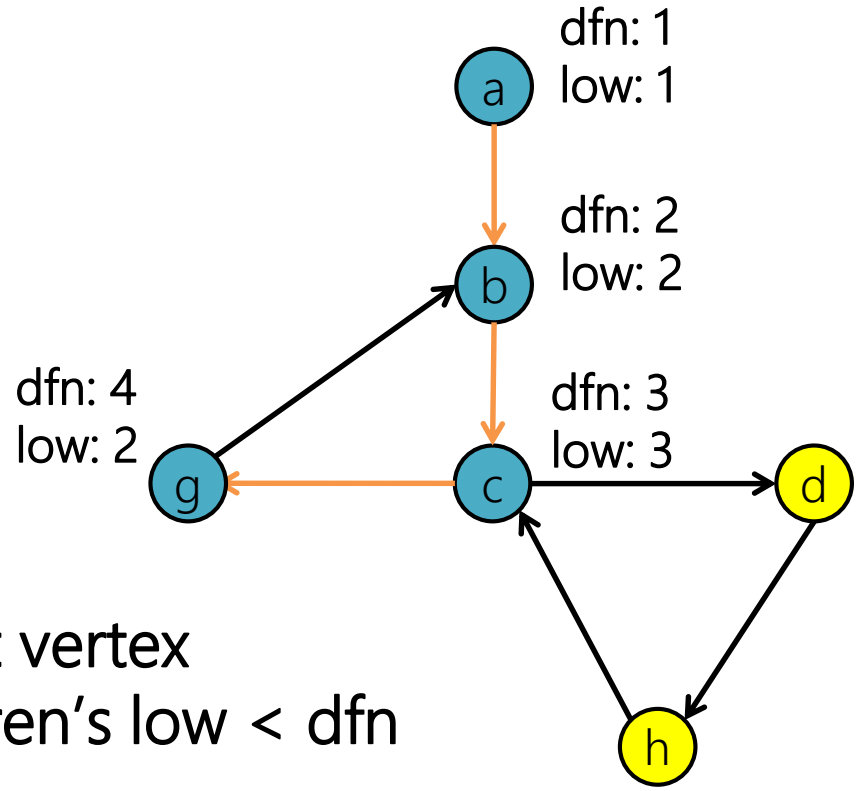
Cut vertex



child node's dfn

made by riljian

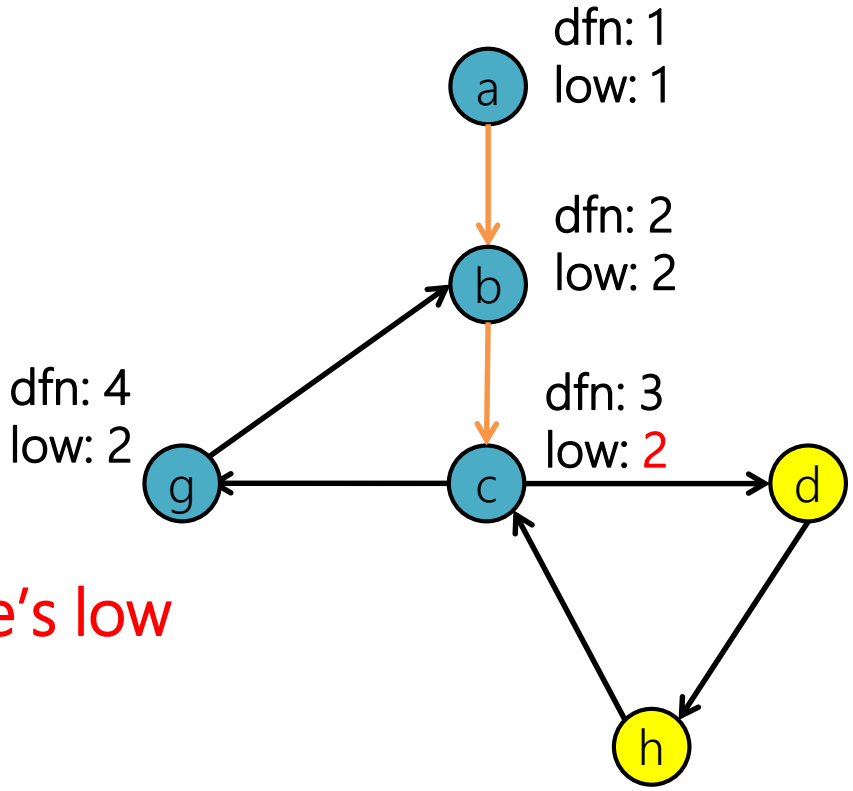
Cut vertex



g is not cut vertex
since children's low < dfn

made by riljian

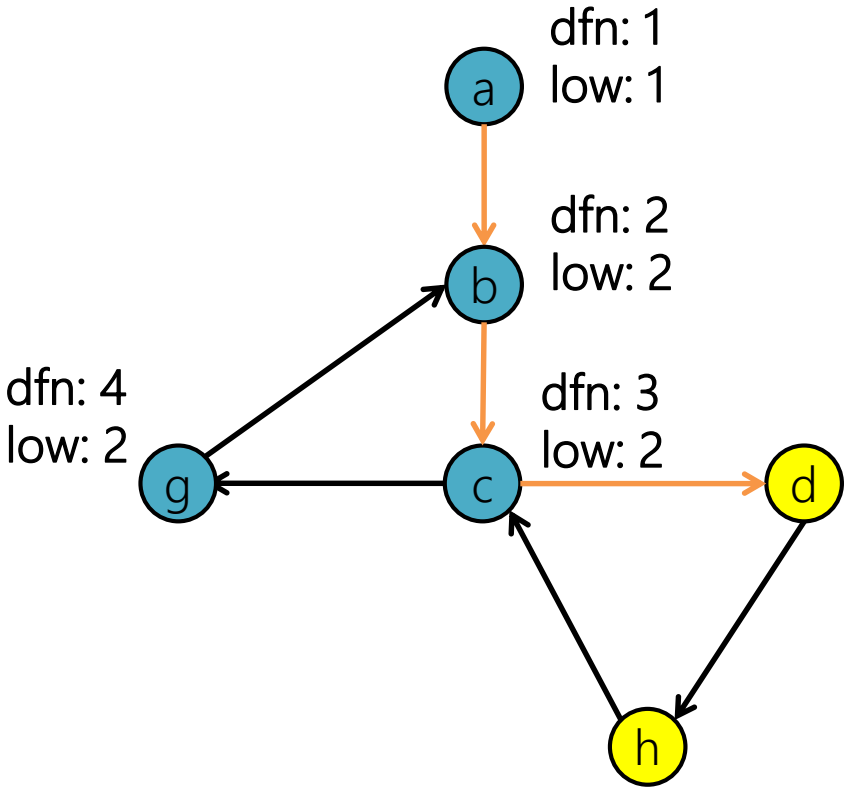
Cut vertex



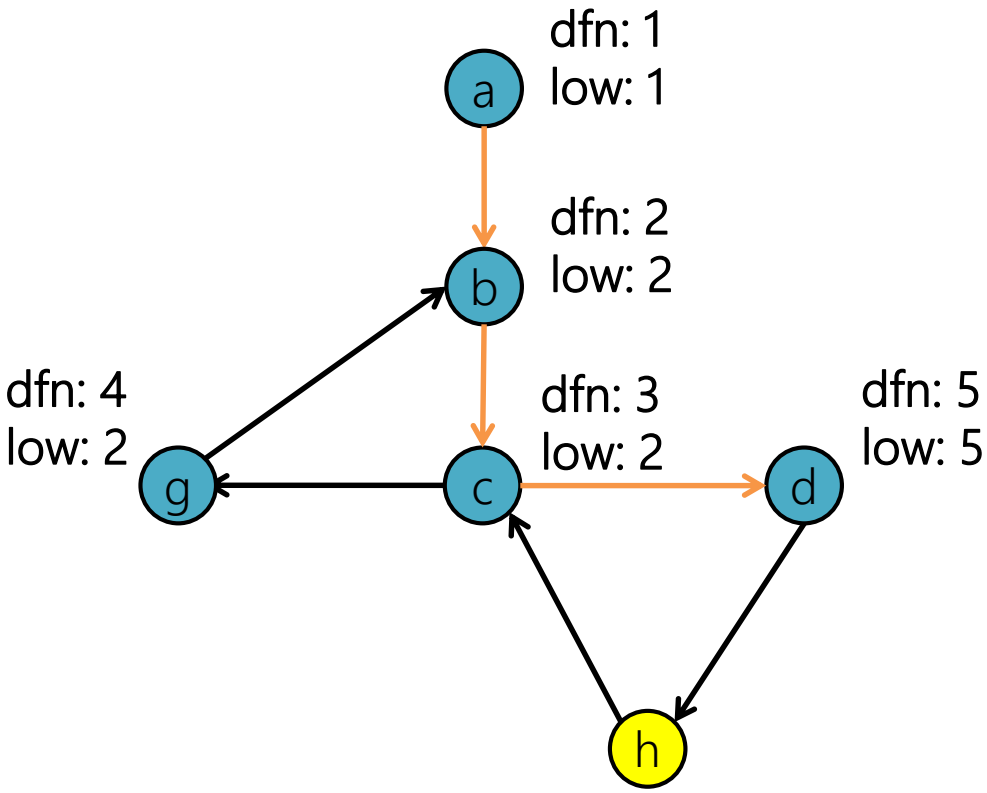
child node's low

made by riljian

Cut vertex

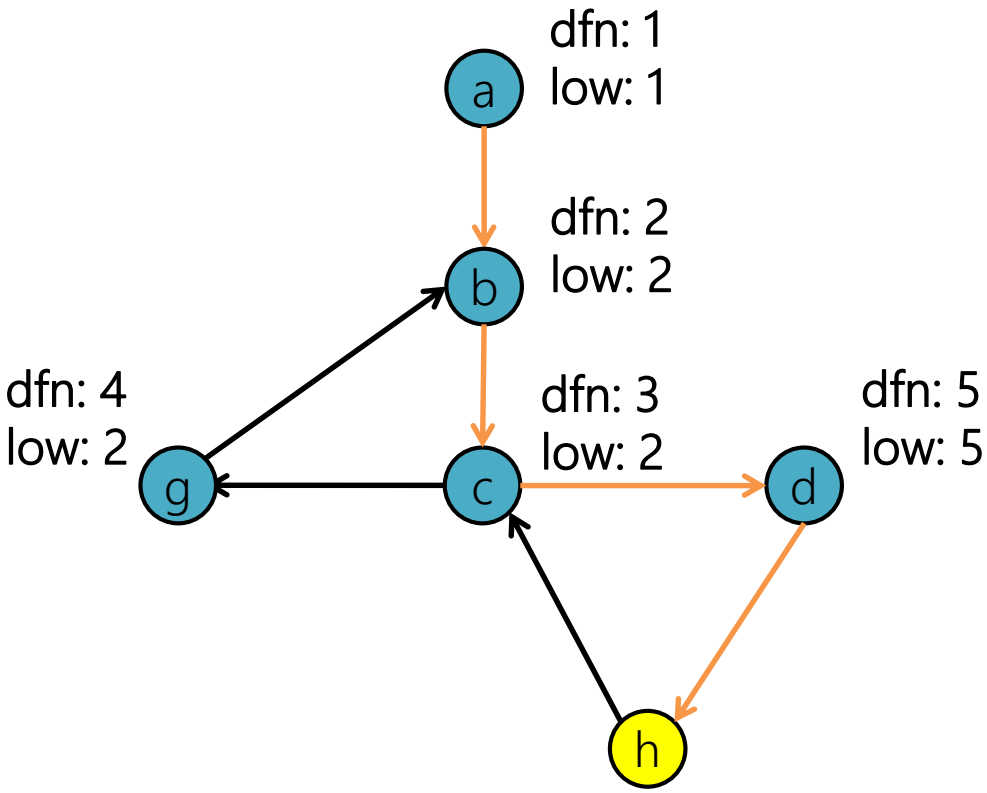


Cut vertex



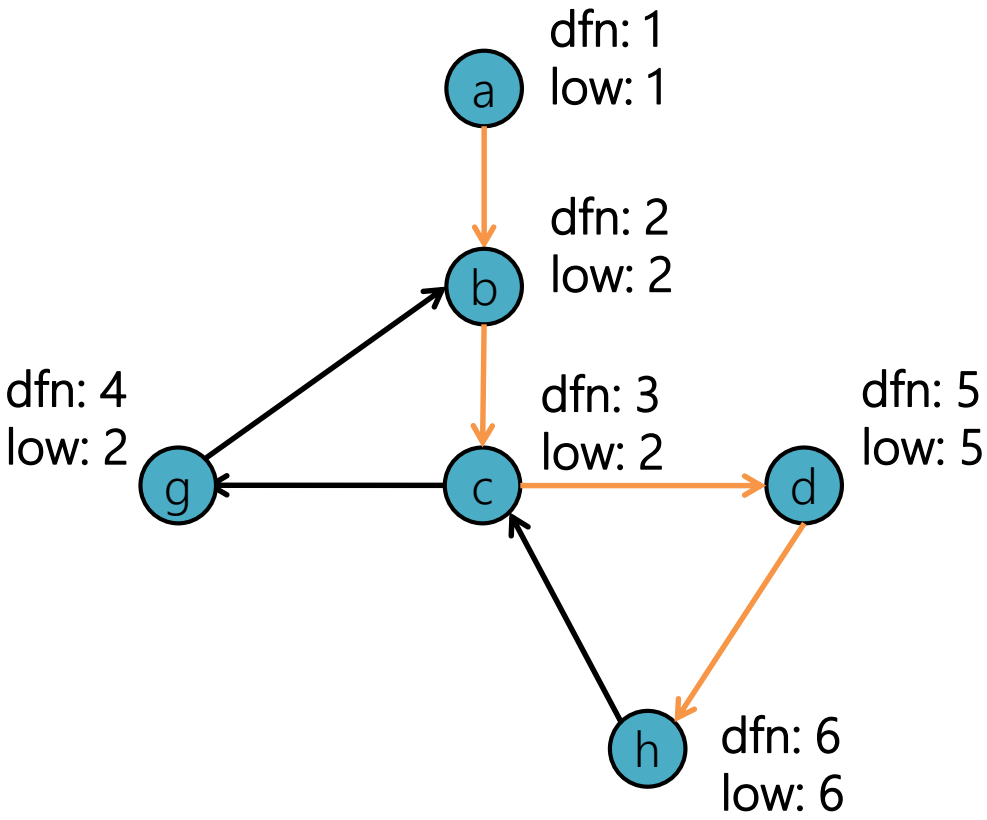
made by riljian

Cut vertex



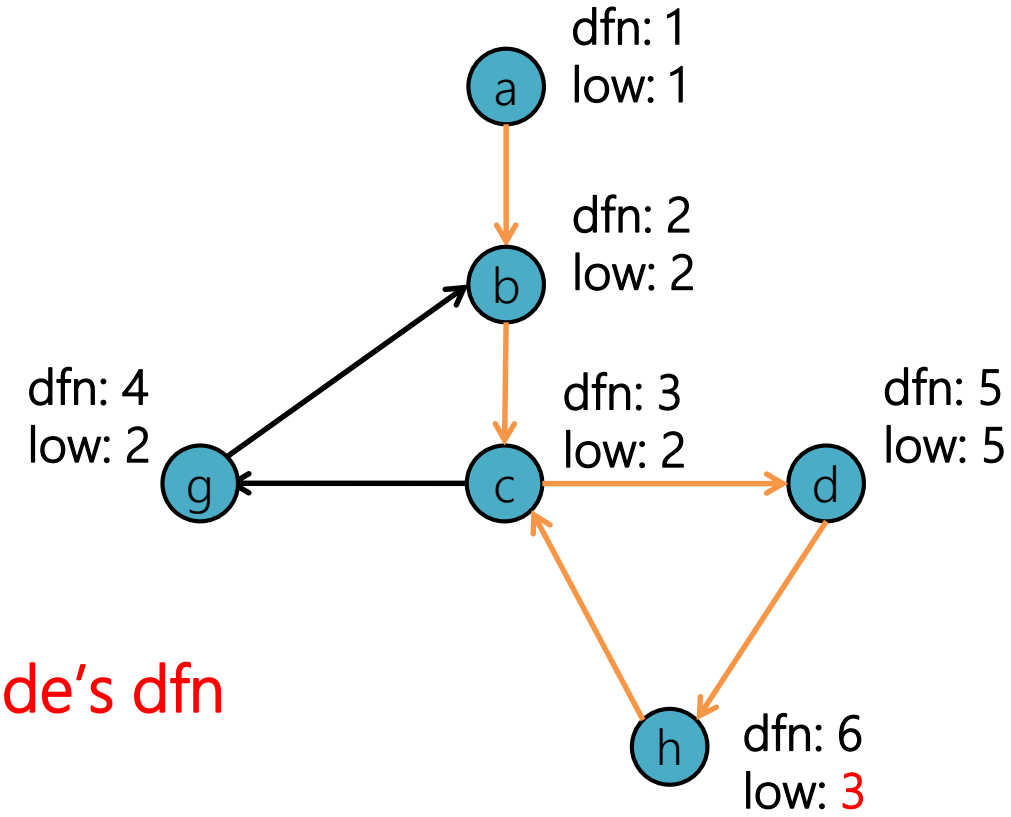
made by riljian

Cut vertex



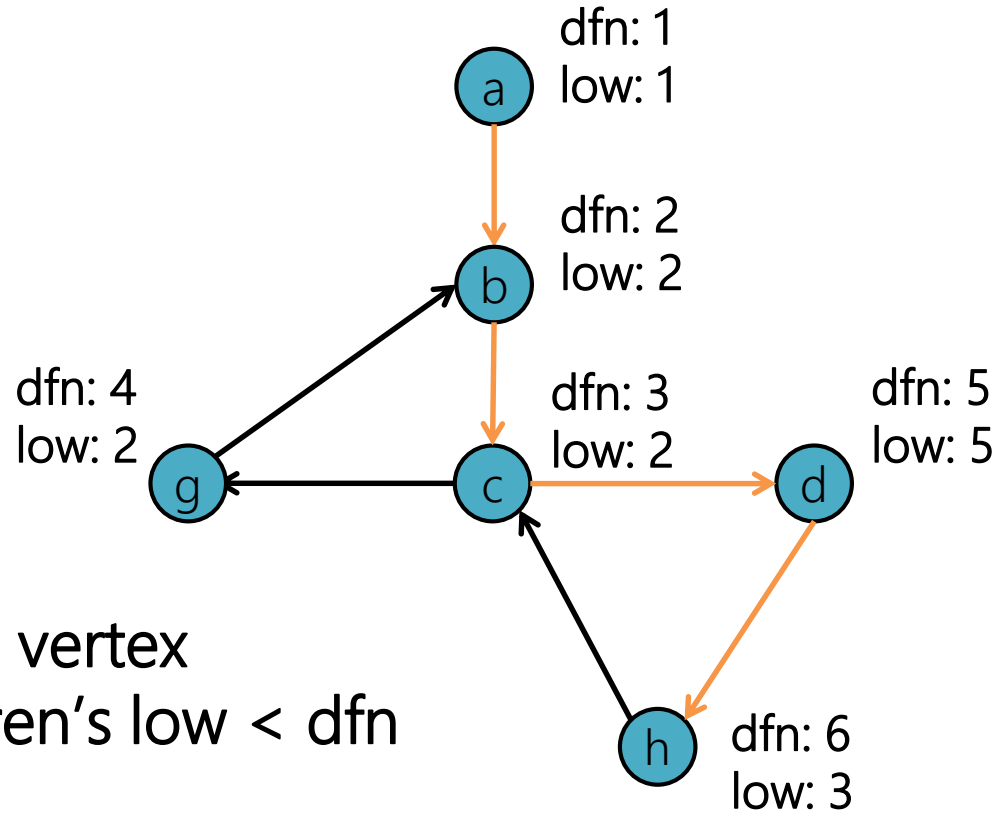
made by riljian

Cut vertex



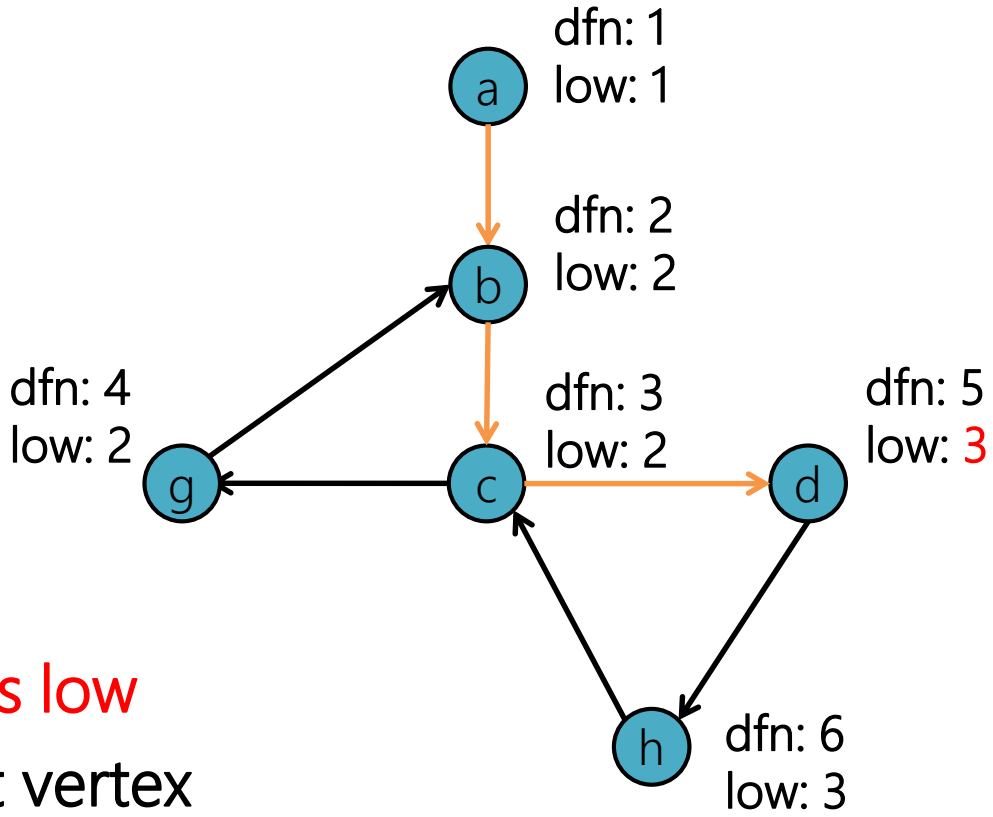
!!! child node's dfn

Cut vertex



h is not cut vertex
since children's low < dfn

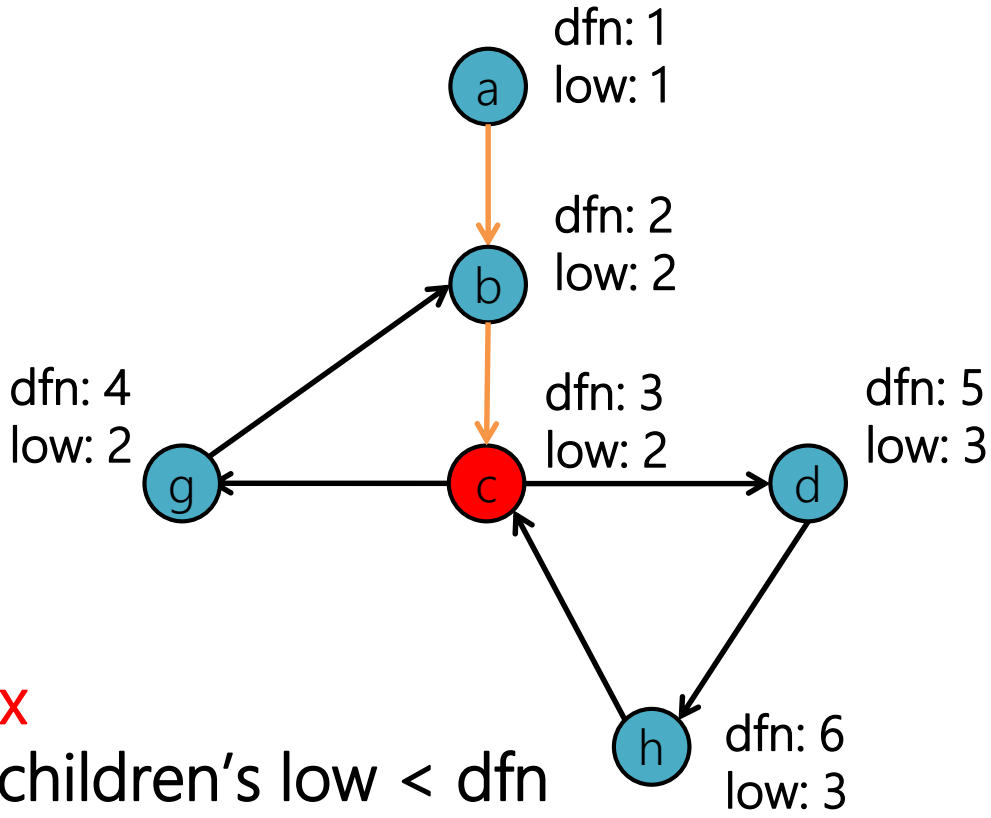
Cut vertex



child node's low

d is not cut vertex
since children's low < dfn

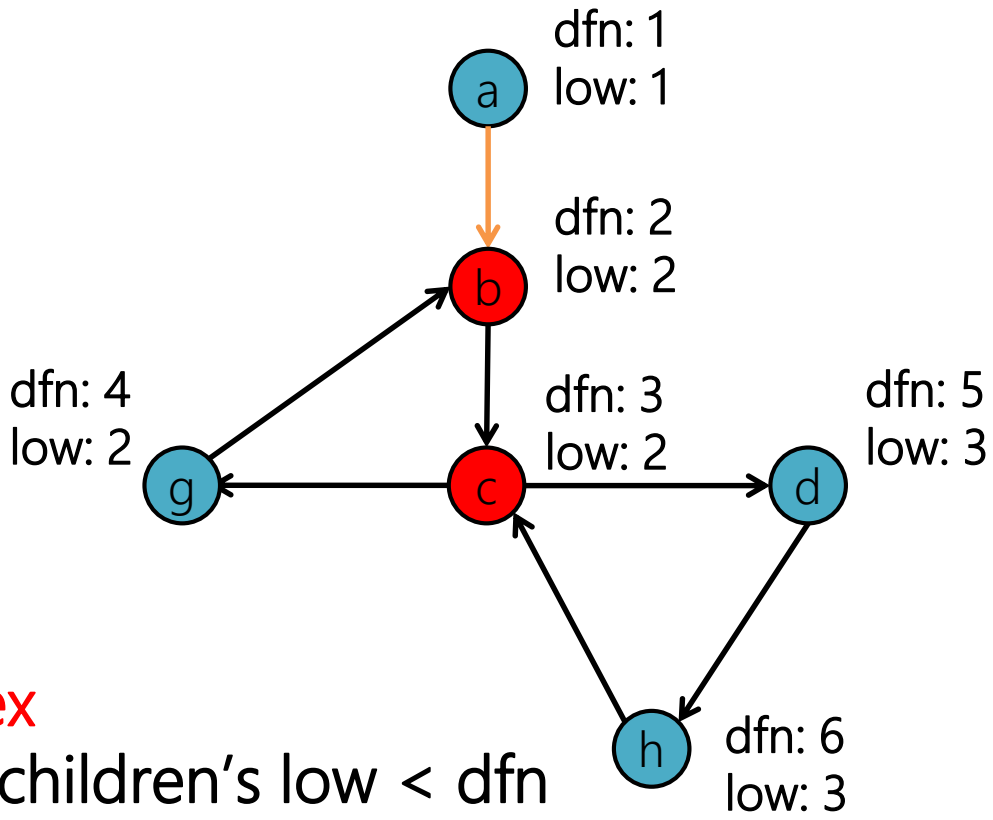
Cut vertex



c is cut vertex
since not all children's low < dfn

made by riljian

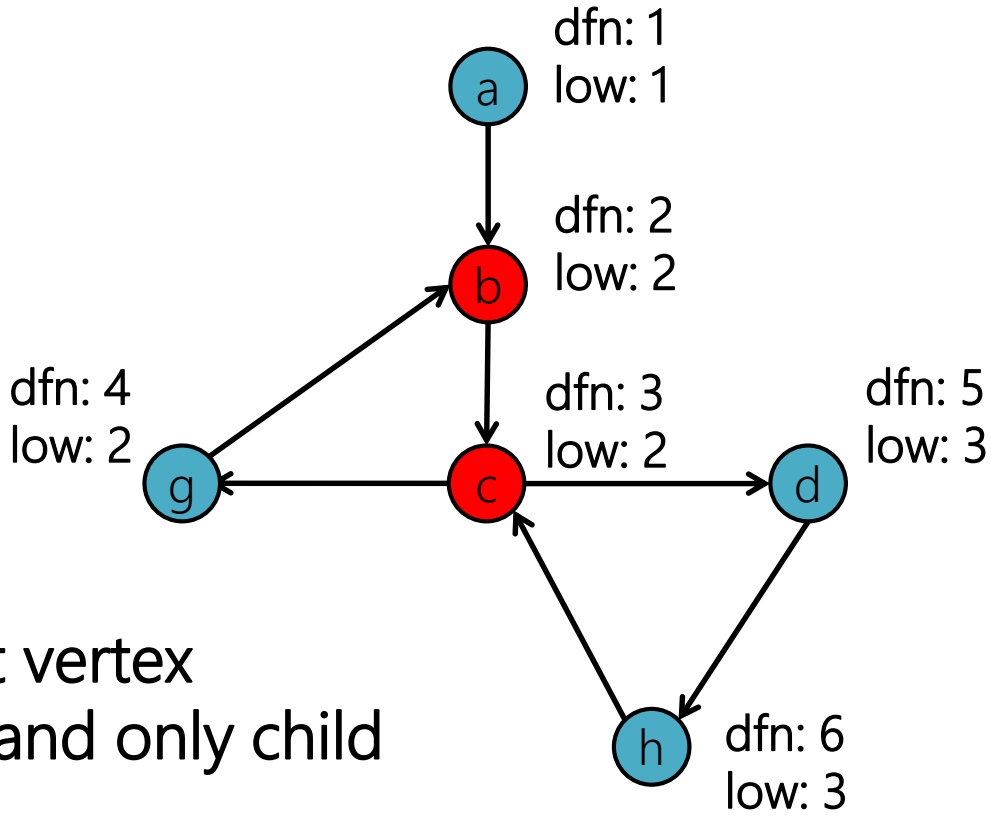
Cut vertex



d is cut vertex
since not all children's low < dfn

made by riljian

Cut vertex



a is not cut vertex
since root and only child

made by riljian

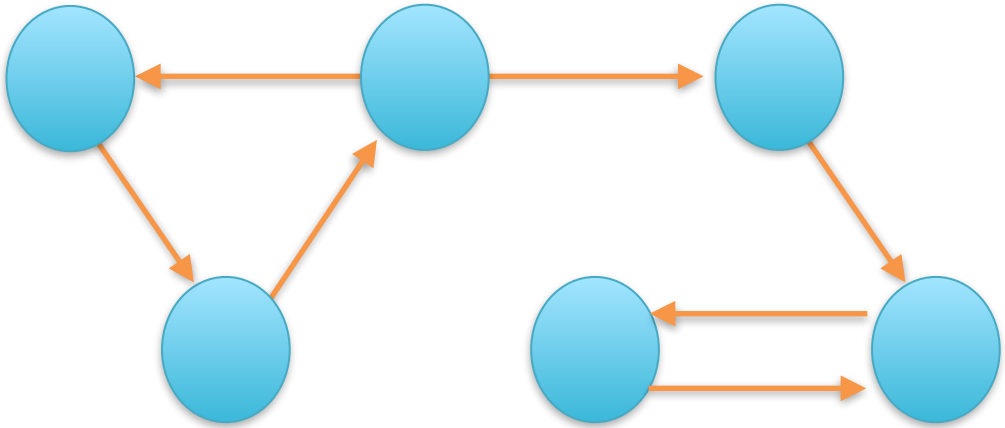
Practice

UVA - 315

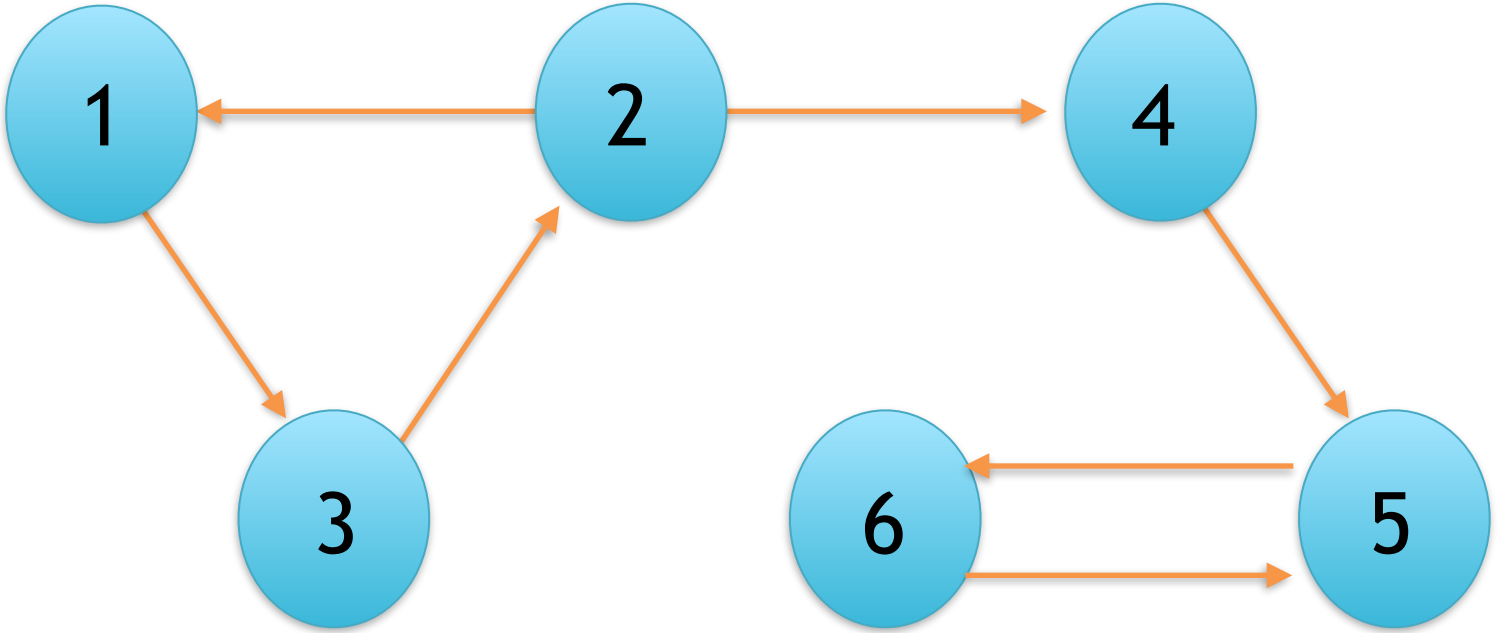
made by tommy5198

SCC

- connected component in directed graph
  - same definition in undirected graph

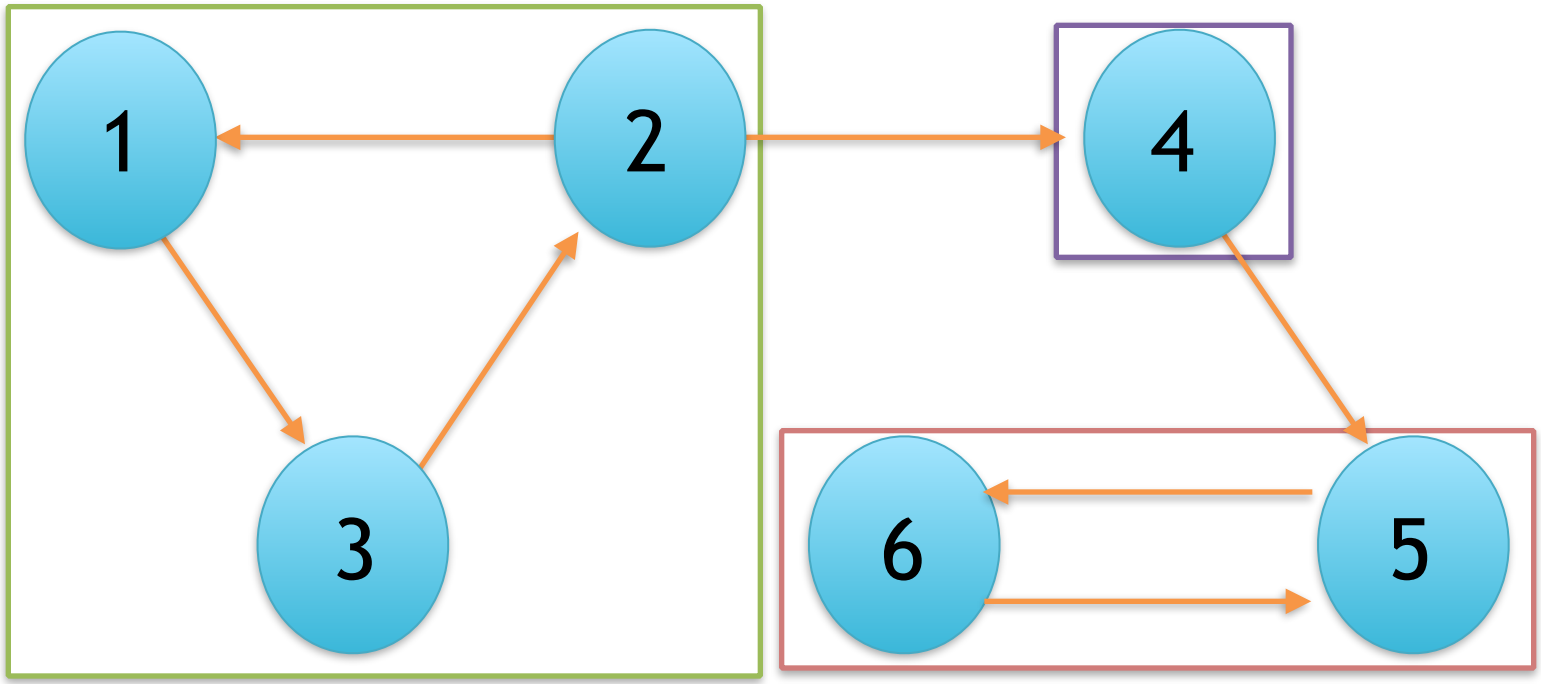


SCC



made by tommy5198

SCC

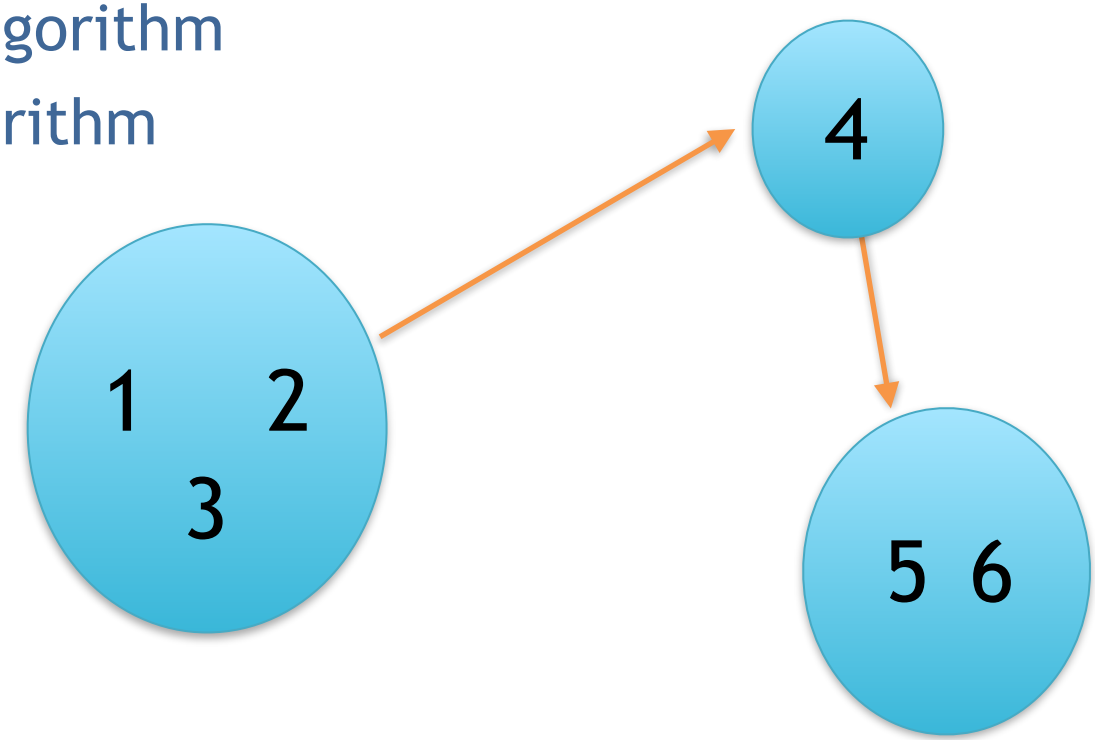


made by tommy5198

SCC

find all SCCs, contract all cycles —> DAG (directed acyclic graph)

- Kosaraju's Algorithm
- Tarjan's Algorithm



made by tommy5198

SCC

- Kosaraju's algorithm

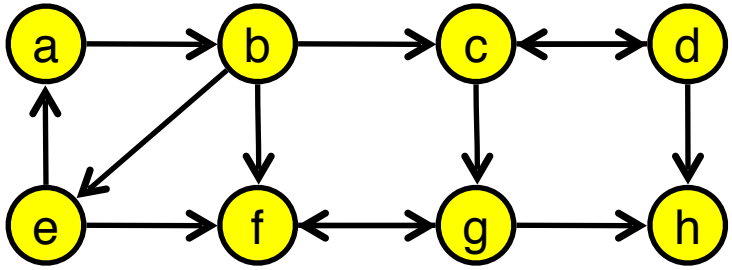
STRONGLY-CONNECTED-COMPONENTS(G)

1. Call DFS(G) to compute finishing time for each vertex.
2. Compute transpose of G i.e., G^T.
3. Call DFS(G^T) but this time consider the vertices in order of decreasing finish time.
4. Out the vertices of each tree in DFS-forest.

twice DFS —> total complexity: O(V+E)

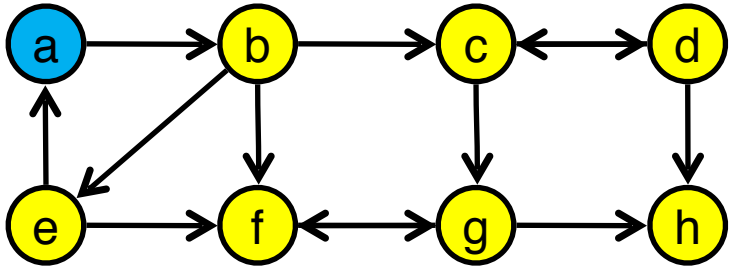
SCC

- Algorithm



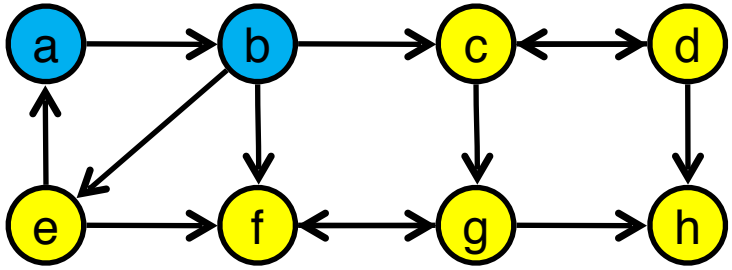
SCC

- Algorithm



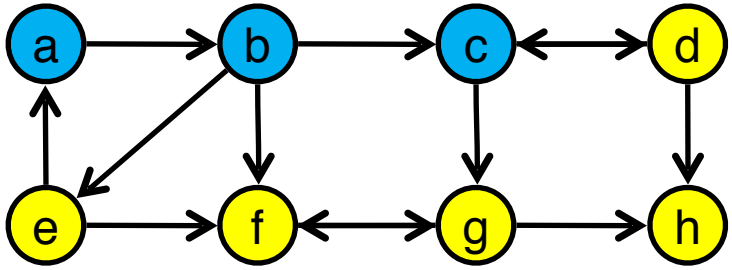
SCC

- Algorithm



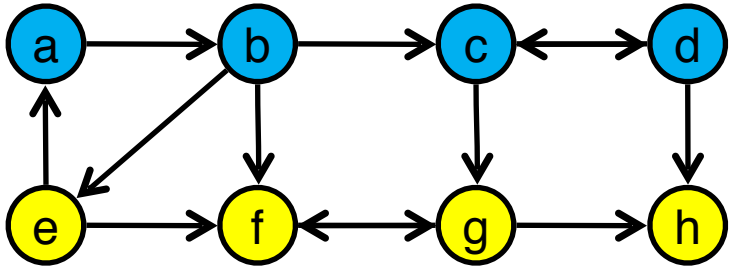
SCC

- Algorithm



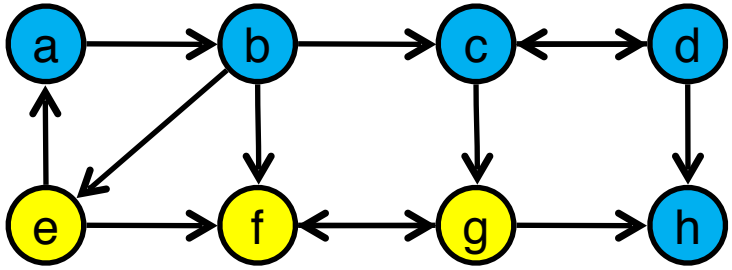
SCC

- Algorithm



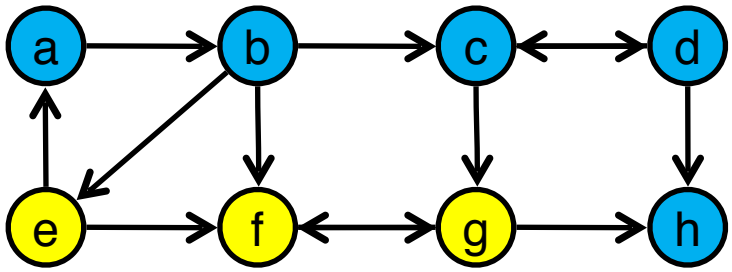
SCC

- Algorithm



SCC

- Algorithm

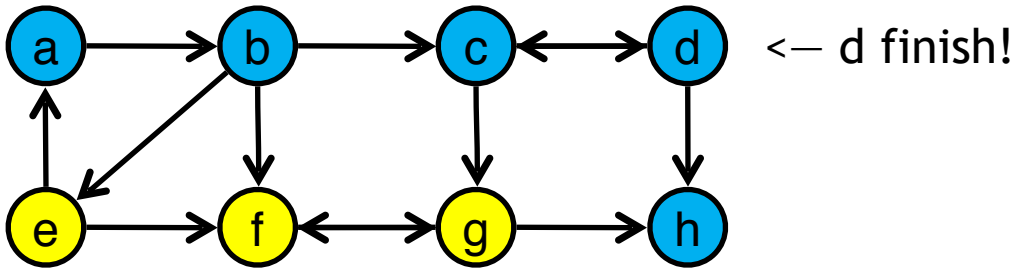


<— h finish!

h									

SCC

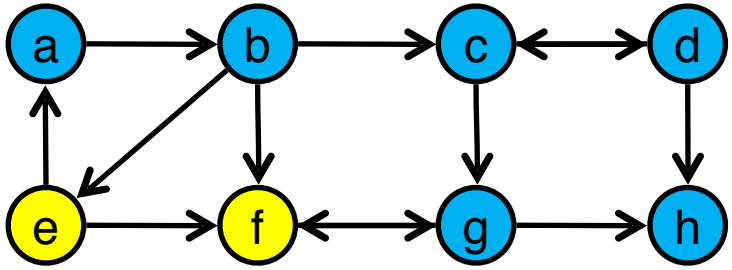
- Algorithm



h	d								

SCC

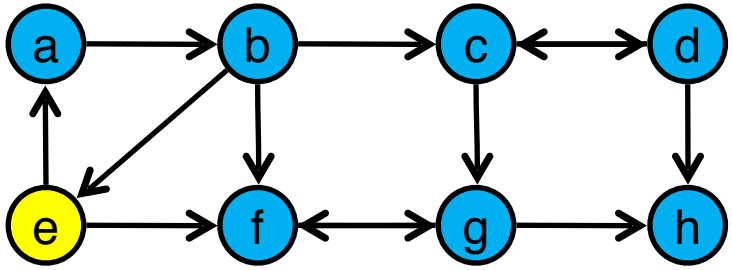
- Algorithm



h	d								

SCC

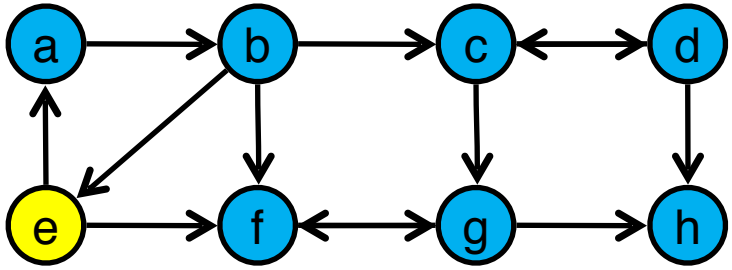
- Algorithm



h	d								

SCC

- Algorithm

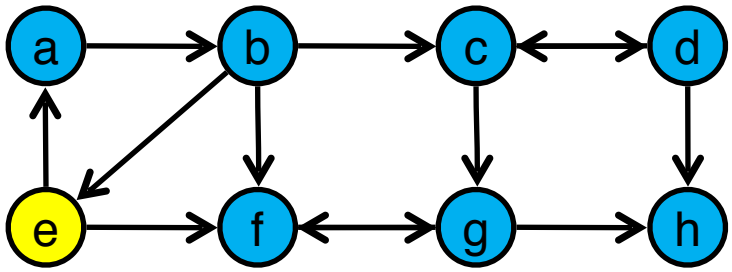


^ f finish!

h	d	f							

SCC

- Algorithm

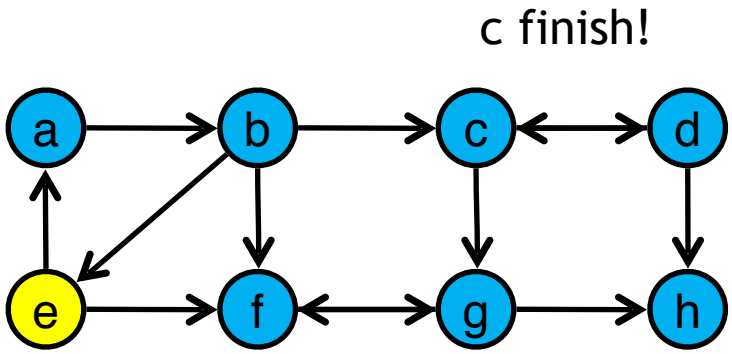


^ g finish!

h	d	f	g						

SCC

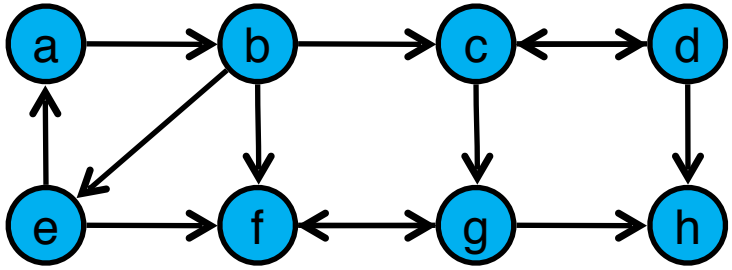
- Algorithm



h	d	f	g	c					

SCC

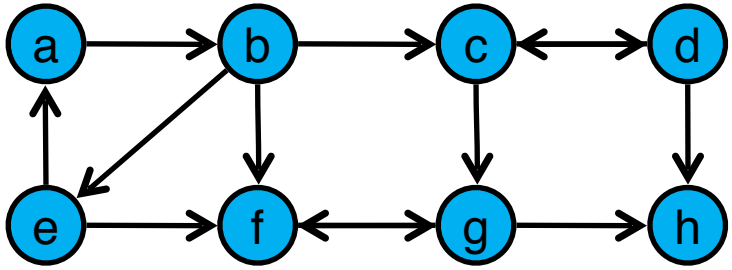
- Algorithm



h	d	f	g	c					

SCC

- Algorithm



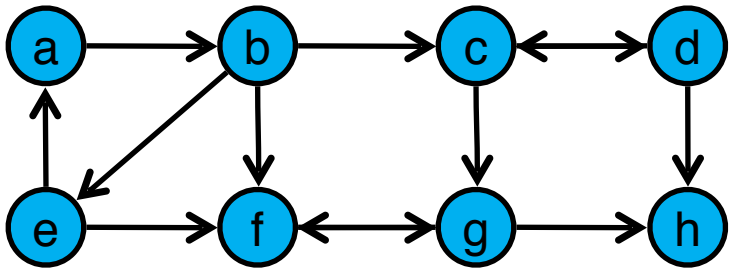
^ e finish!

h	d	f	g	c	e				

SCC

- Algorithm

b finish!

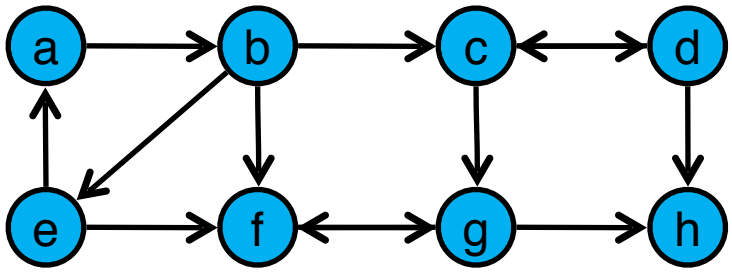


h	d	f	g	c	e	b			

SCC

- Algorithm

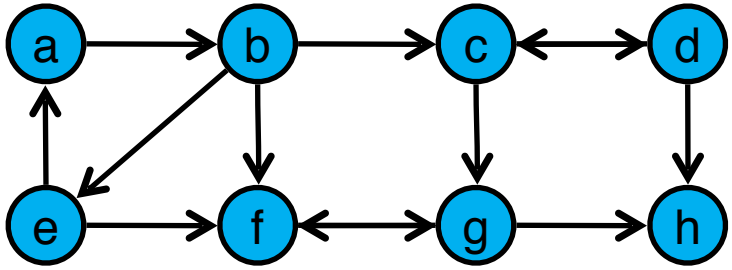
a finish!



h	d	f	g	c	e	b	a		

SCC

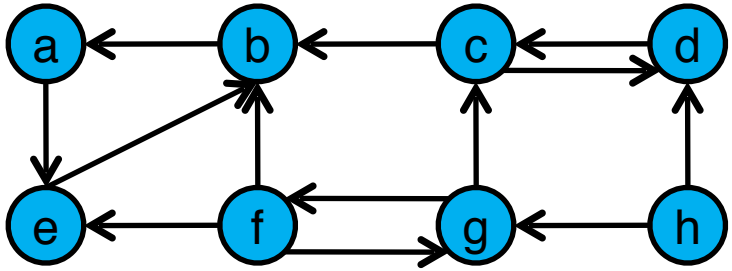
- Algorithm
  - Reverse the graph



h	d	f	g	c	e	b	a		

SCC

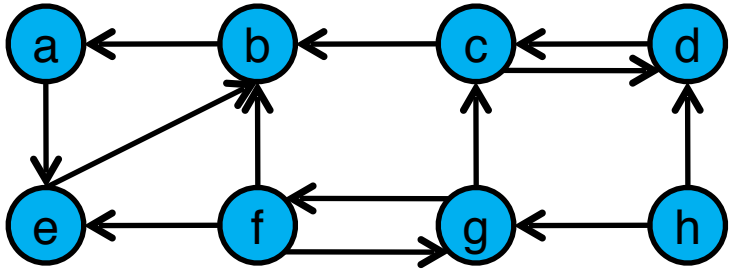
- Algorithm
  - Reverse the graph



h	d	f	g	c	e	b	a		

SCC

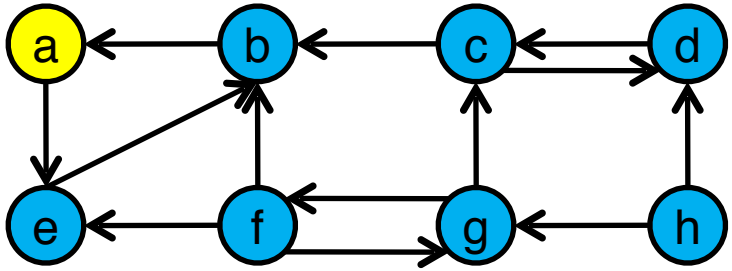
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

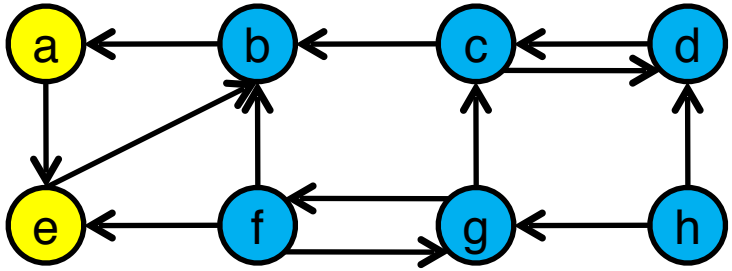
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

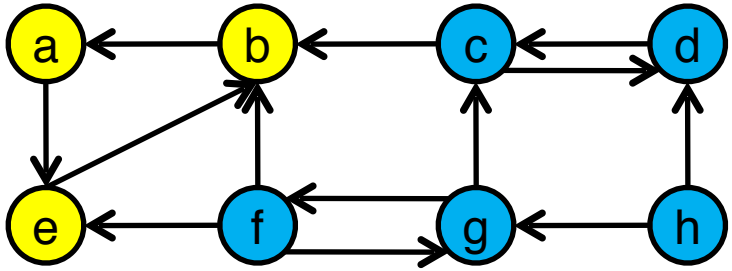
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

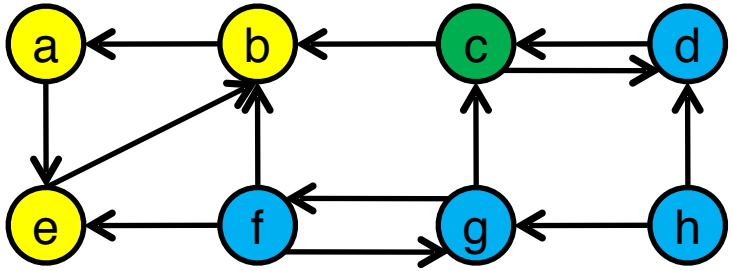
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

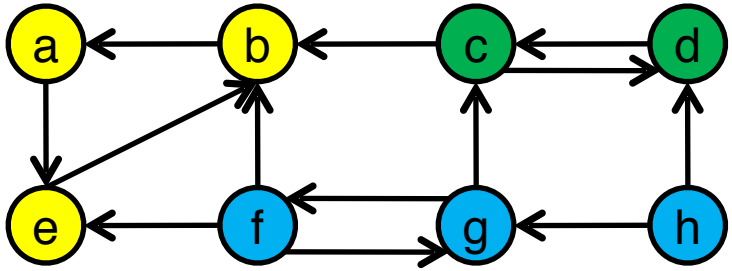
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

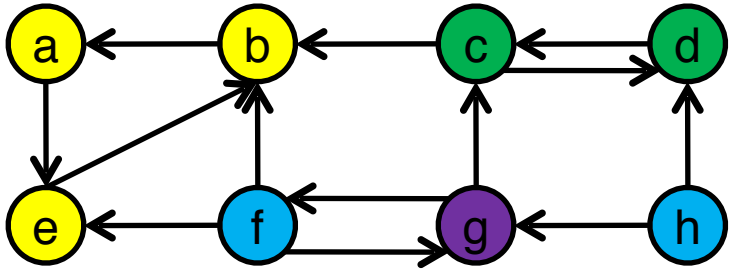
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

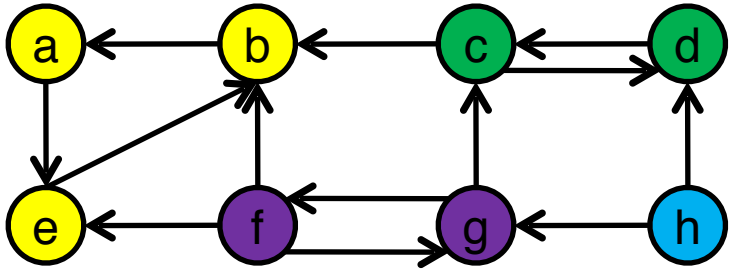
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



made by free999 & electron

SCC

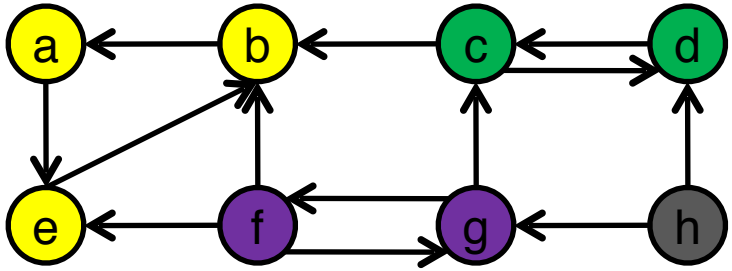
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



made by free999 & electron

SCC

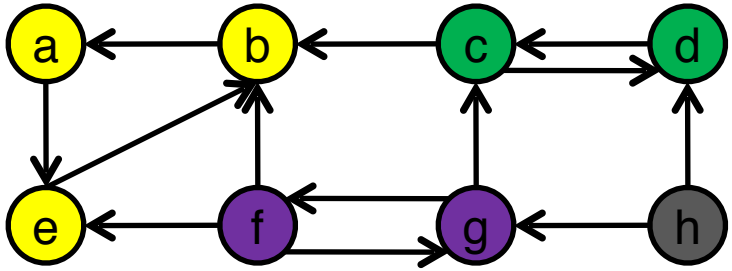
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



SCC

- Algorithm
  - Reverse the graph
  - Re-search by the ending time

4 components



SCC

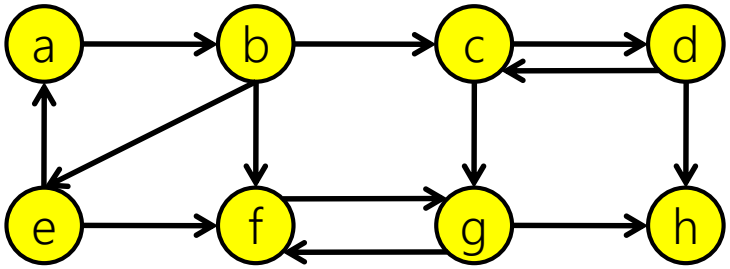
- Tarjan

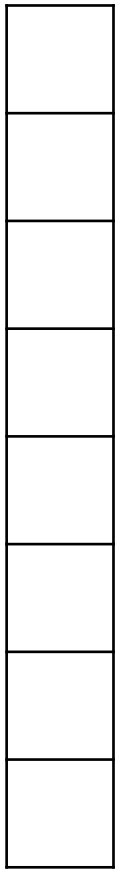
```
42 void DFS(int v) {
43     int top;
44     dfn[v] = low[v] = ++dfn_cnt;
45     stk.push(v);
46     in_stk[v] = true;
47     for (int idx = adj_list[v]; ~idx; idx = edge[idx].next) {
48         if (!dfn[edge[idx].to]) {
49             DFS(edge[idx].to);
50             low[v] = min(low[v], low[edge[idx].to]);
51         } else if (in_stk[edge[idx].to]) {
52             low[v] = min(low[v], dfn[edge[idx].to]);
53         }
54     }
55
56     if (dfn[v] == low[v]) {
57         do {
58             top = stk.top();
59             stk.pop();
60             in_stk[top] = false;
61         } while (top != v);
62         ++ans;
63     }
64 }
```


made by riljian

SCC

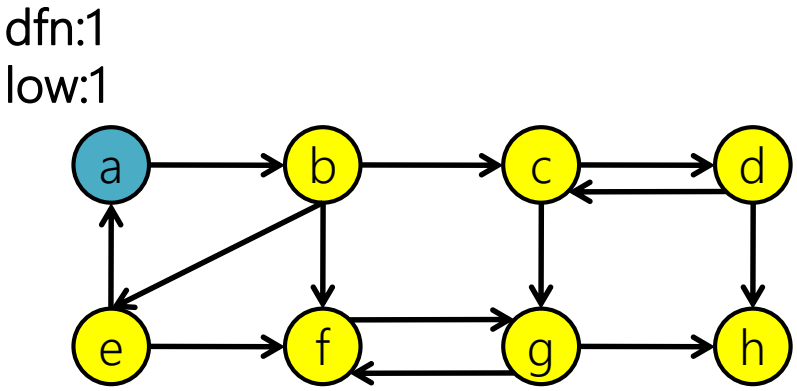
- Tarjan

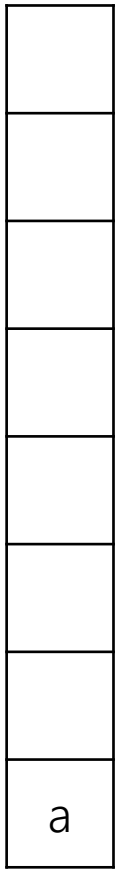




SCC

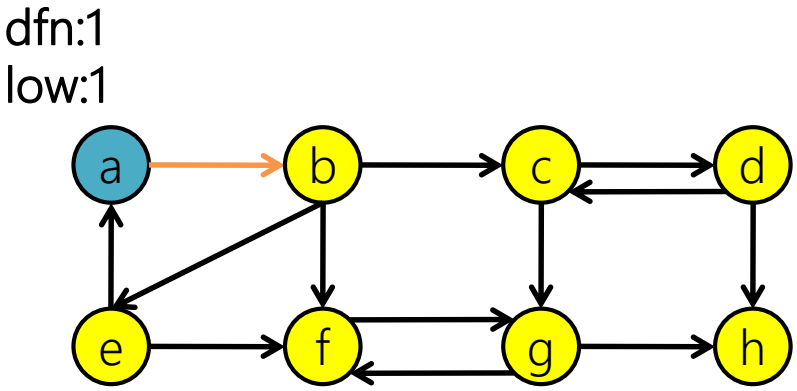
- Tarjan

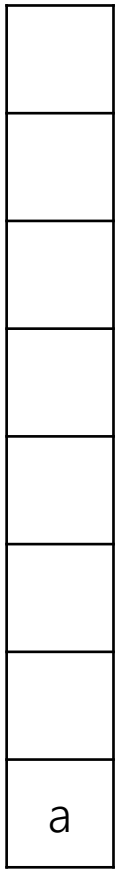




SCC

- Tarjan

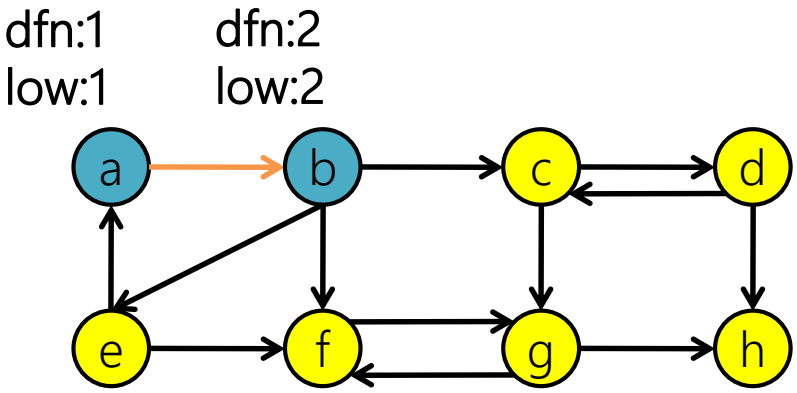


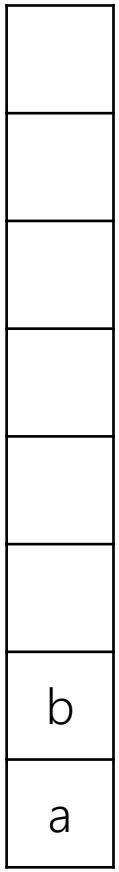


made by riljian

SCC

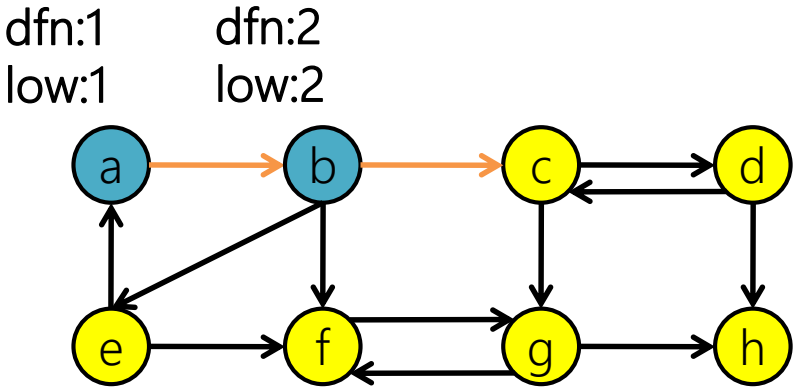
- Tarjan

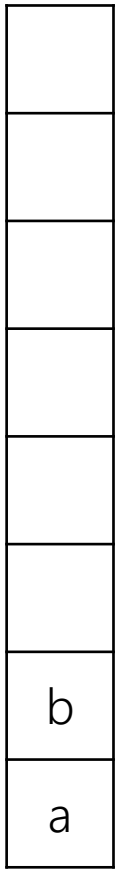




SCC

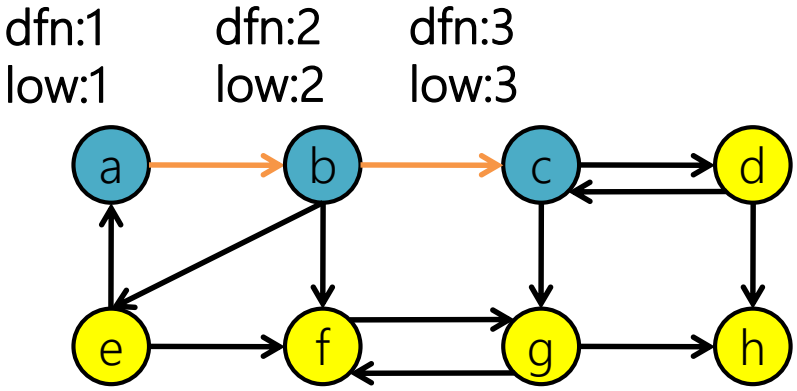
- Tarjan

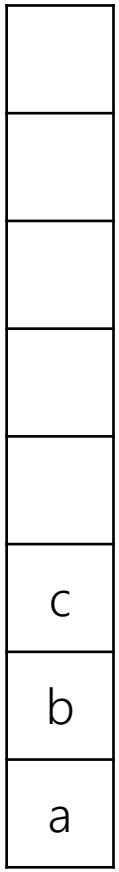




SCC

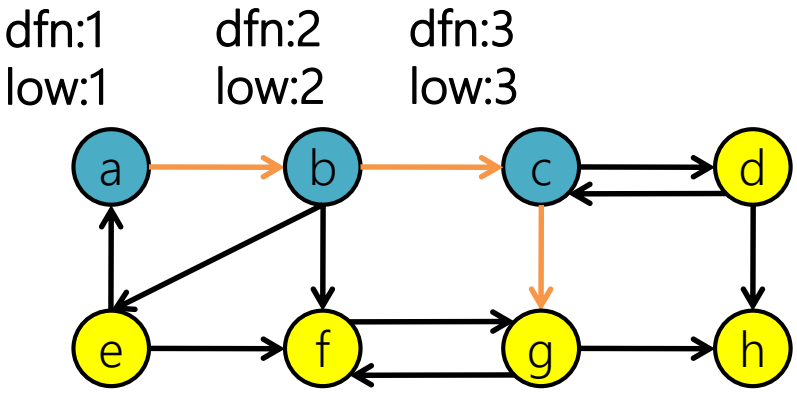
- Tarjan

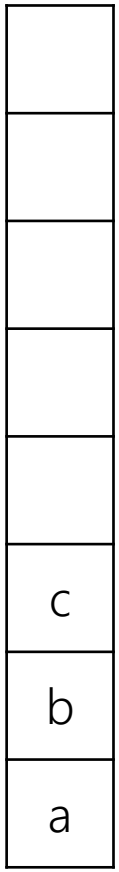




SCC

- Tarjan

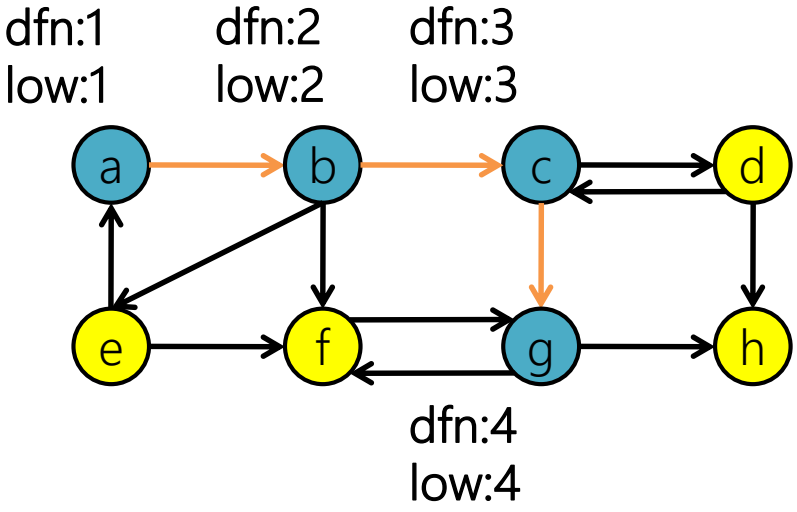


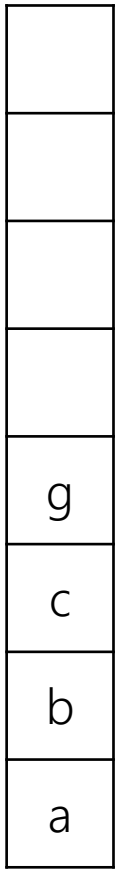


made by riljian

SCC

- Tarjan

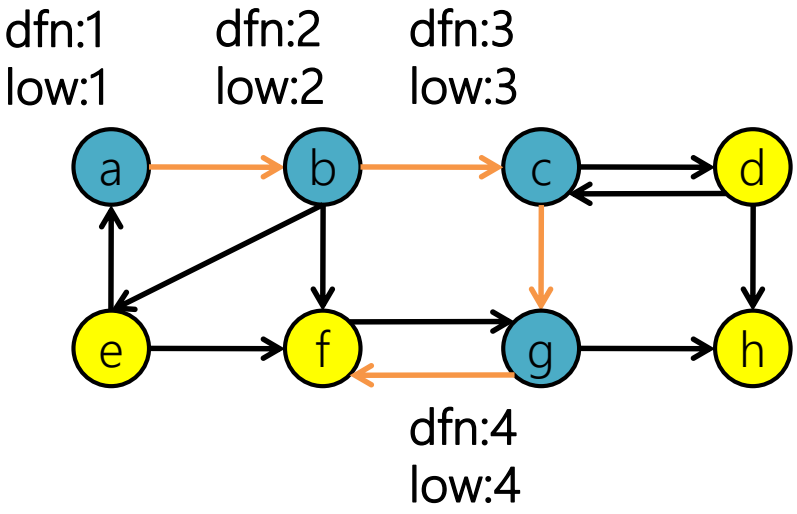


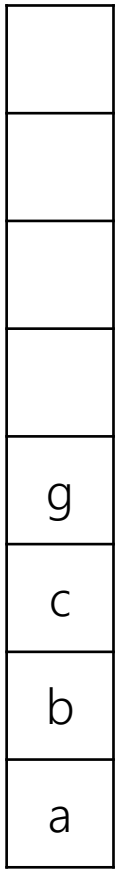


made by riljian

SCC

- Tarjan

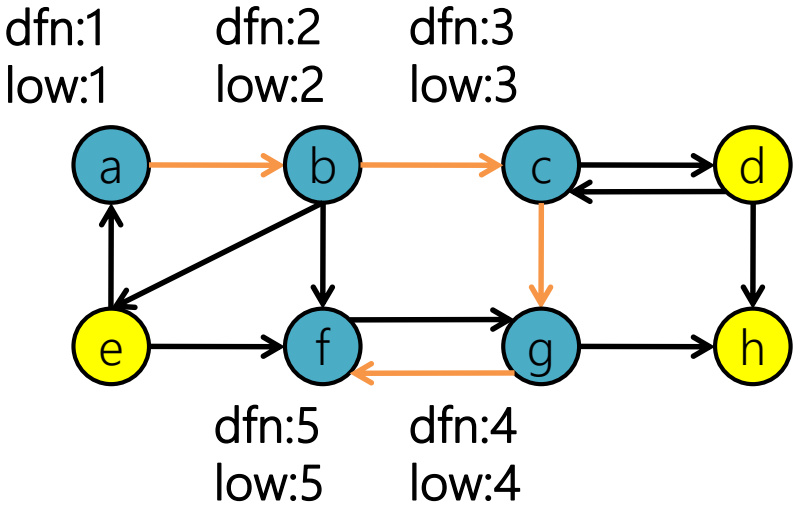


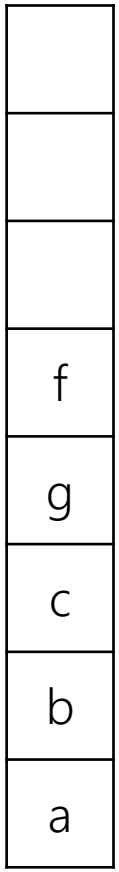


made by riljian

SCC

- Tarjan

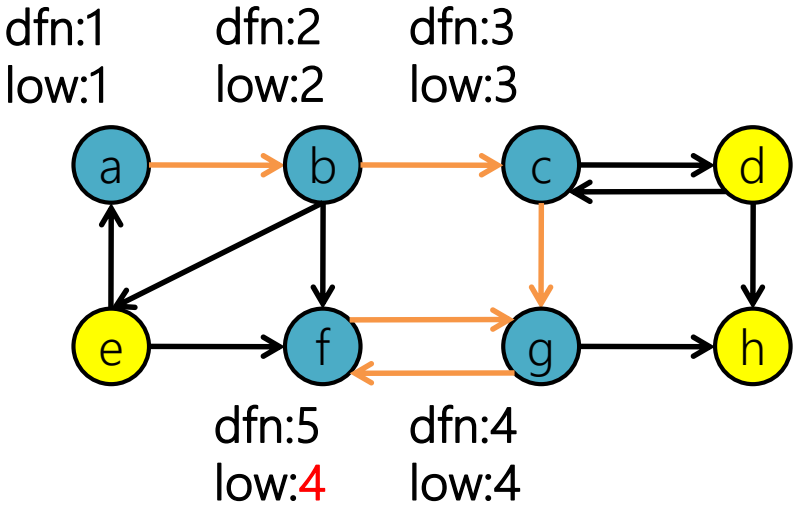




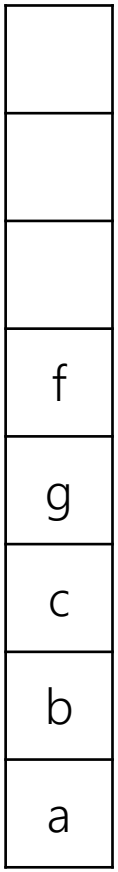
made by riljian

SCC

- Tarjan

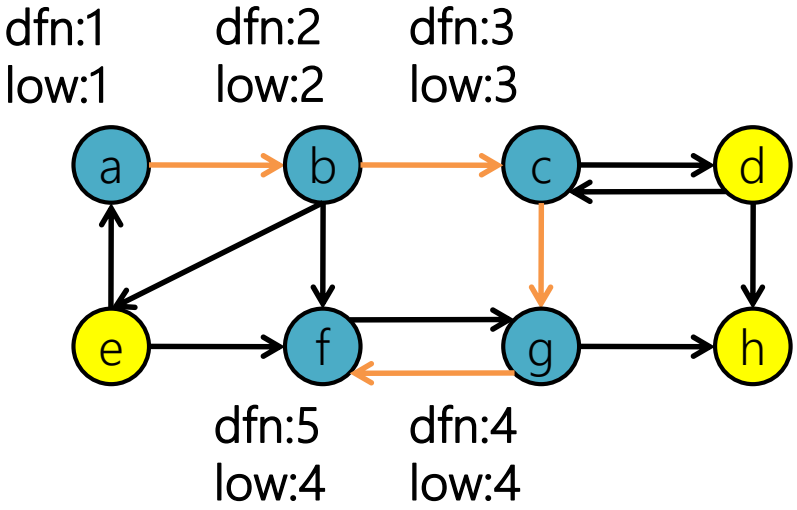


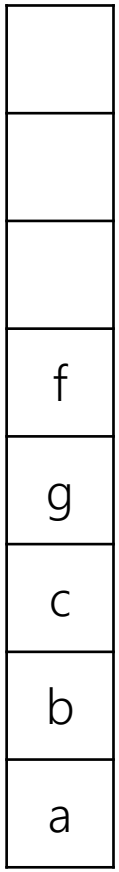
In stack



SCC

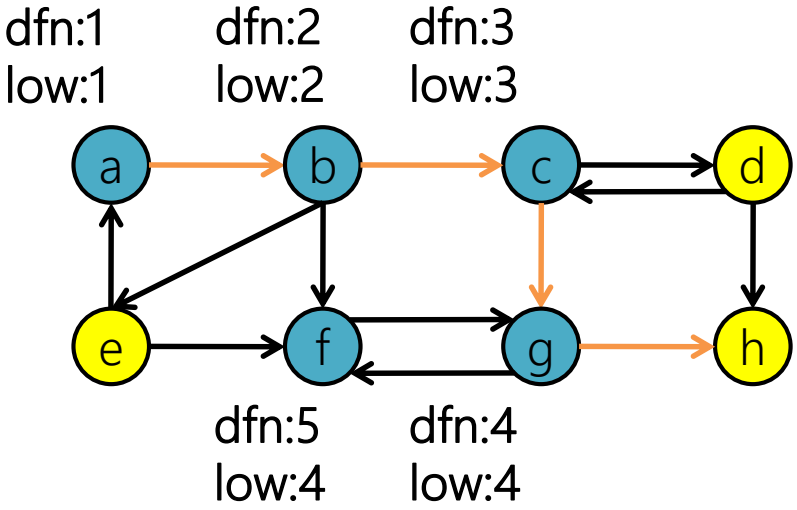
- Tarjan

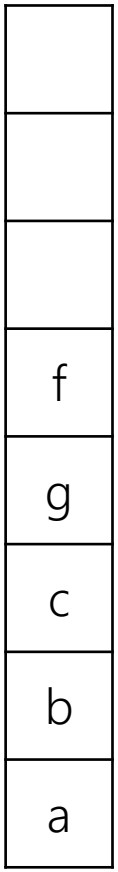

dfn:1
low:1
dfn:2
low:2
dfn:3
low:3
a
b
c
d
e
f
g
h
dfn:5
low:4
dfn:4
low:4


f
g
c
b
a

SCC

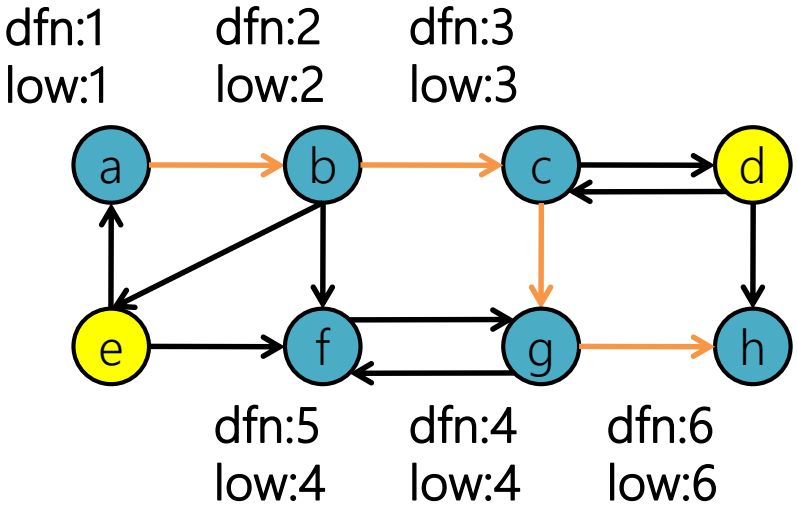
- Tarjan

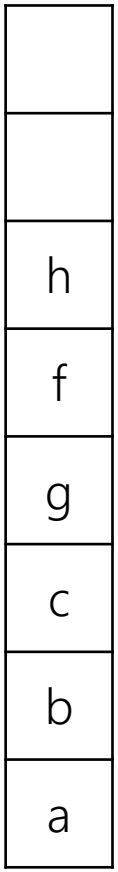

dfn:1
low:1
dfn:2
low:2
dfn:3
low:3
a
b
c
d
e
f
g
h
dfn:5
low:4
dfn:4
low:4


f
g
c
b
a

SCC

- Tarjan

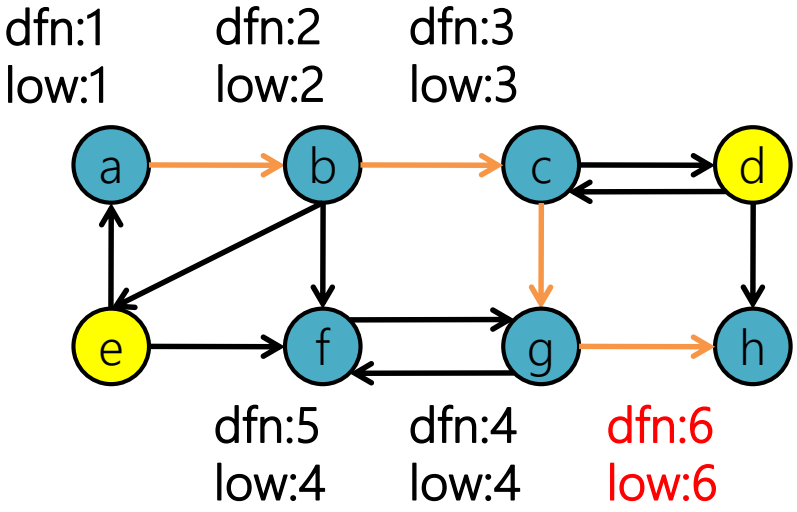




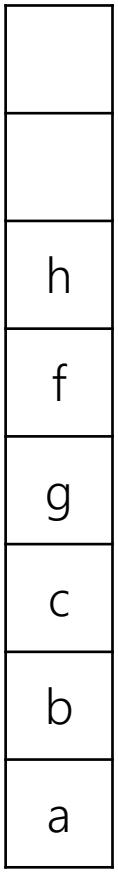
made by riljian

SCC

- Tarjan



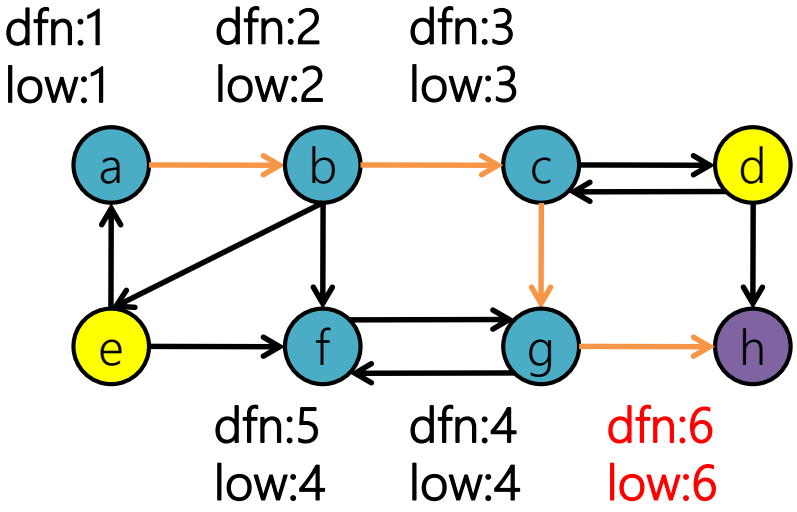
dfn == low

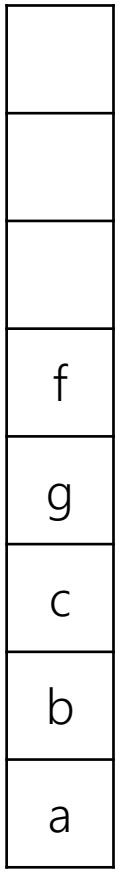


made by riljian

SCC

- Tarjan

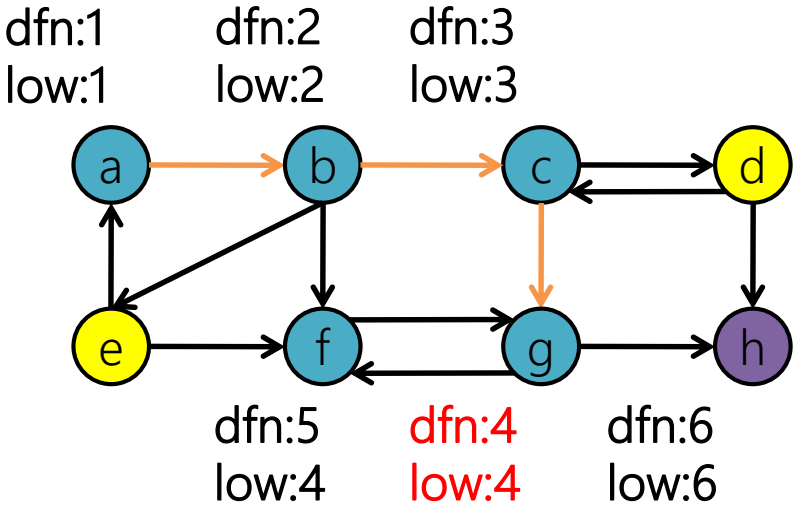




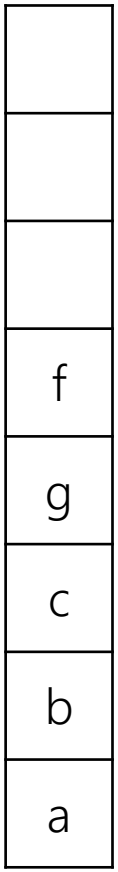
made by riljian

SCC

- Tarjan

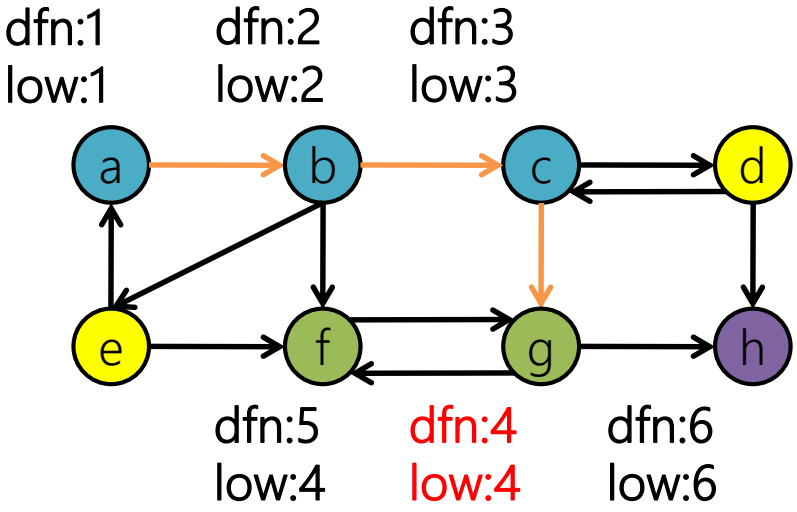


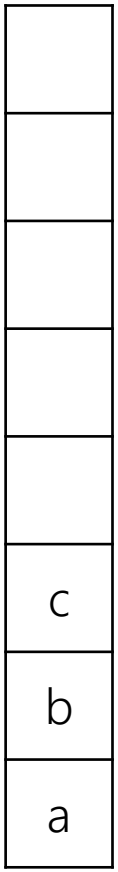
dfn == low



SCC

- Tarjan

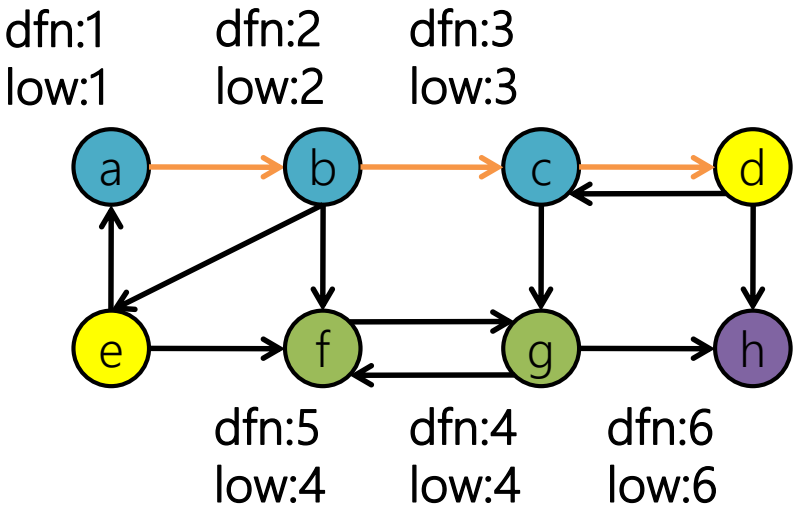


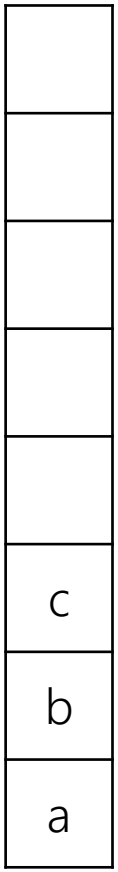


made by riljian

SCC

- Tarjan

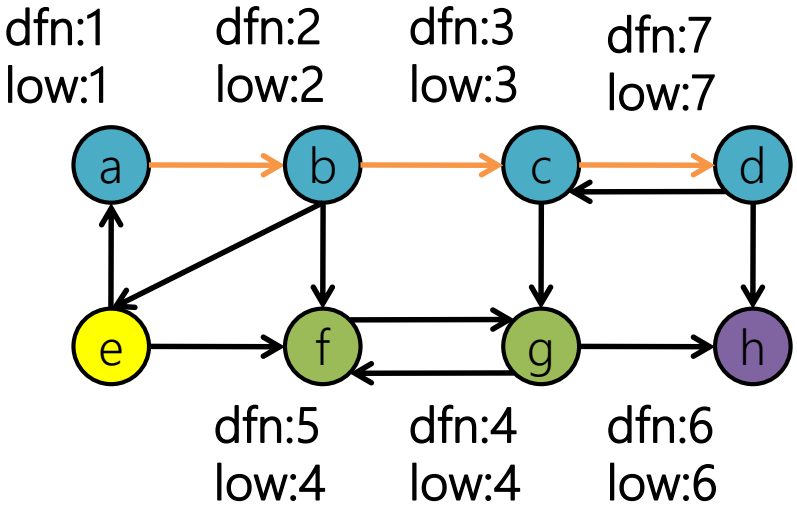


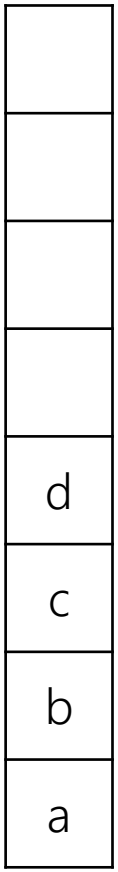


made by riljian

SCC

- Tarjan

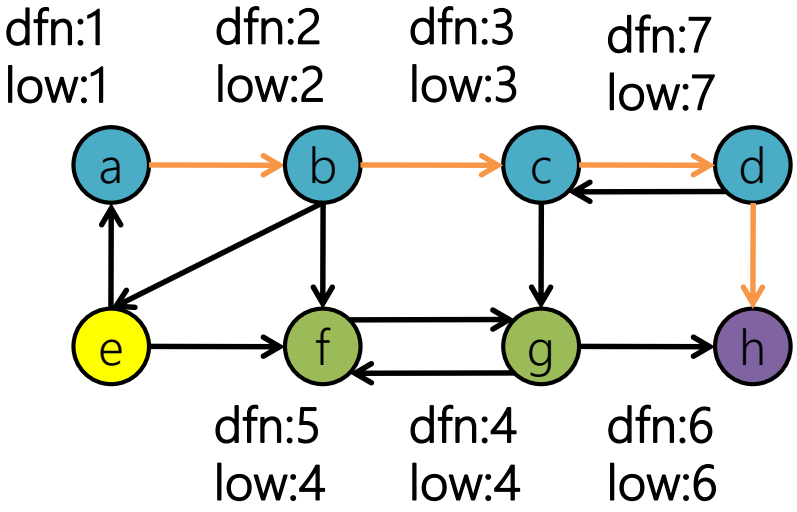


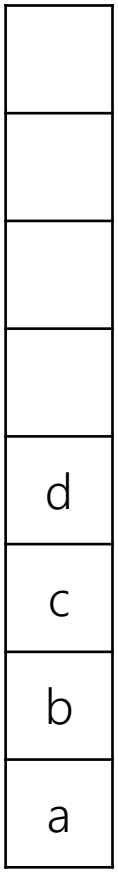


made by riljian

SCC

- Tarjan

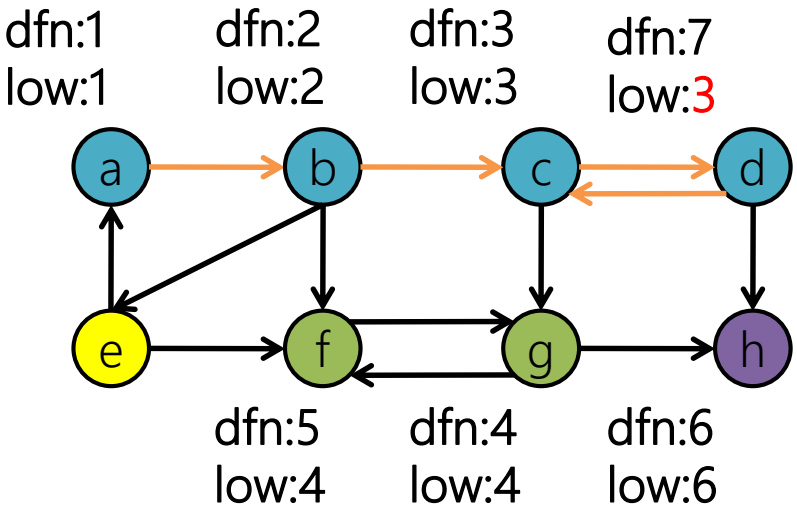




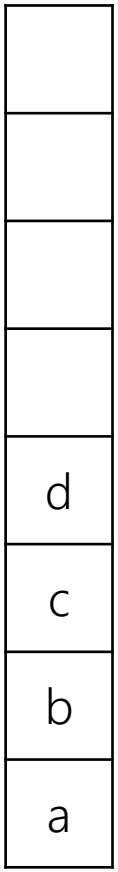
made by riljian

SCC

- Tarjan

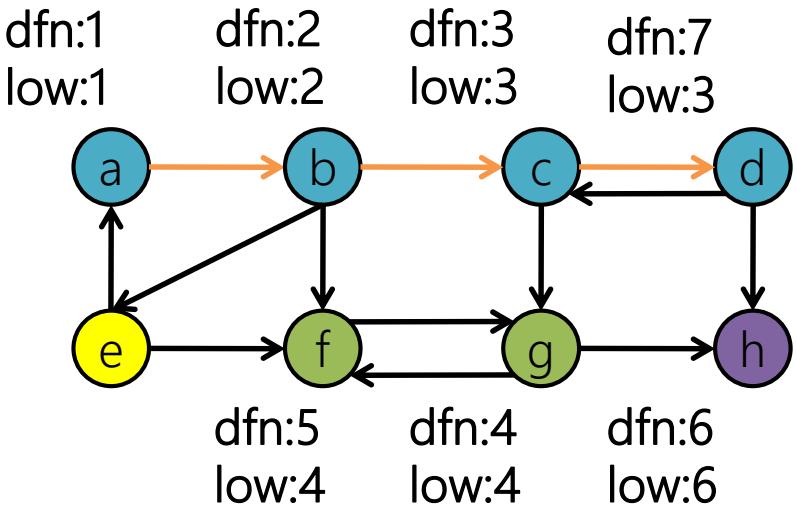


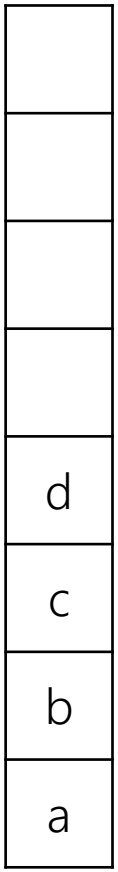
In stack



SCC

- Tarjan

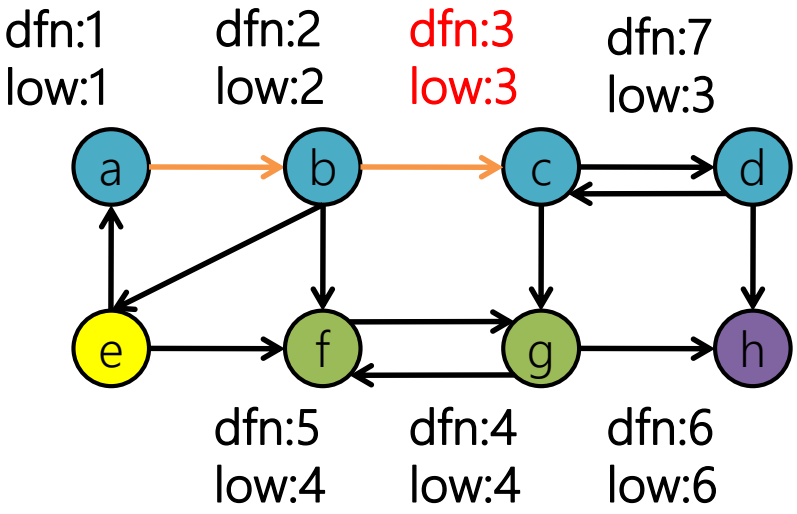




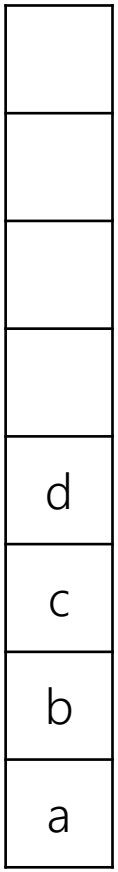
made by riljian

SCC

- Tarjan

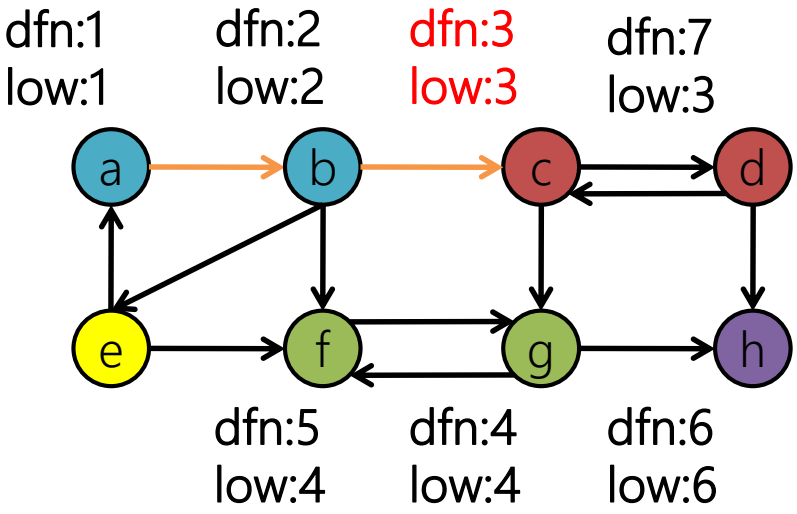


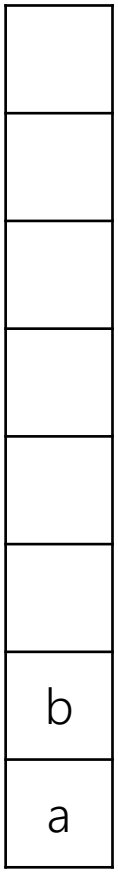
dfn == low



SCC

- Tarjan

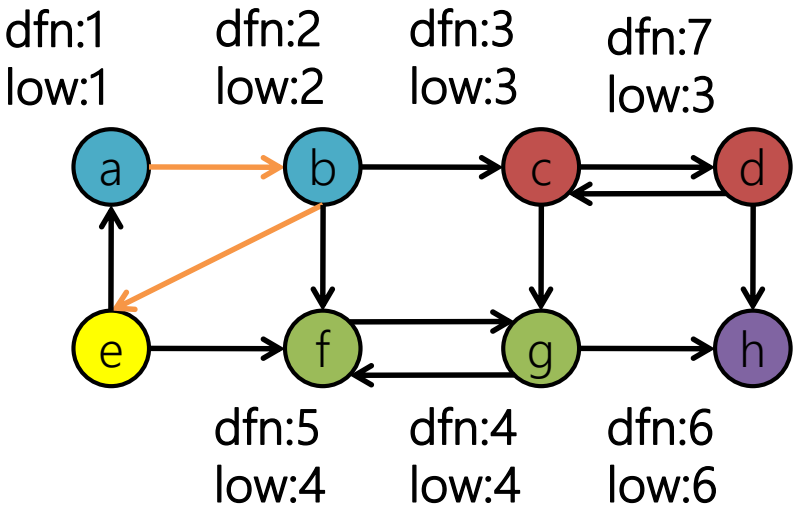


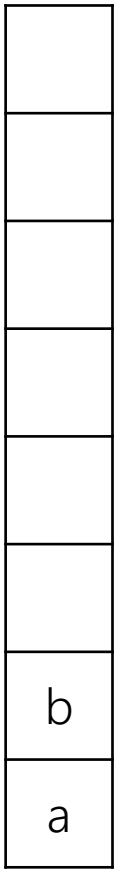


made by riljian

SCC

- Tarjan

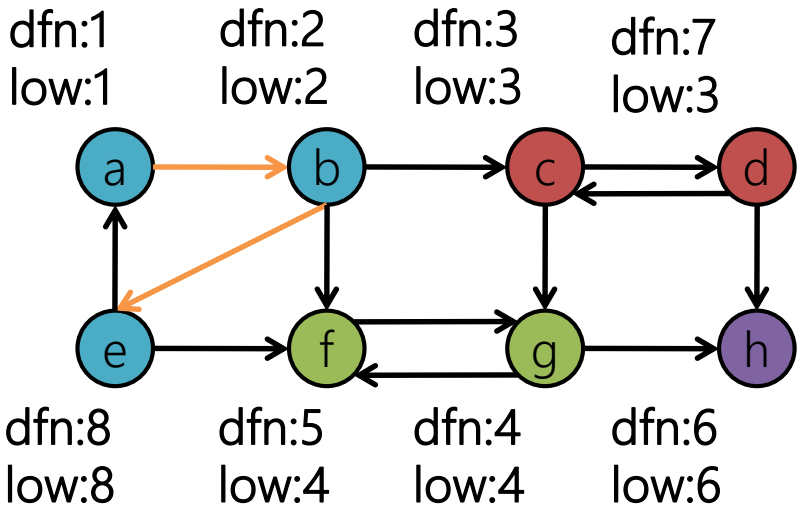


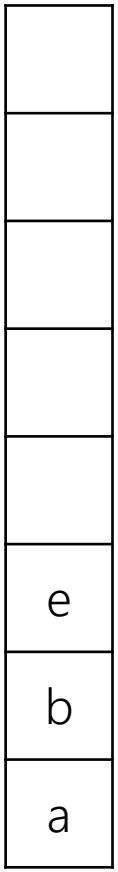


made by riljian

SCC

- Tarjan

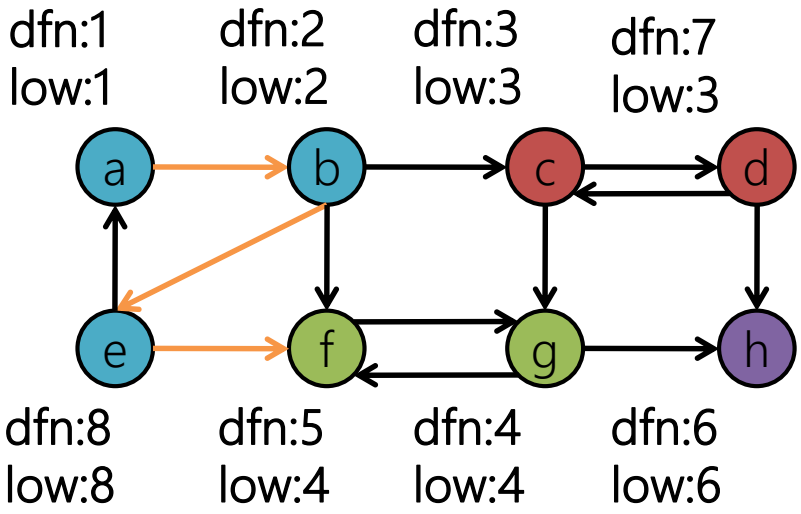


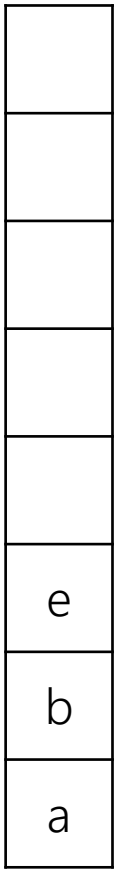


made by riljian

SCC

- Tarjan

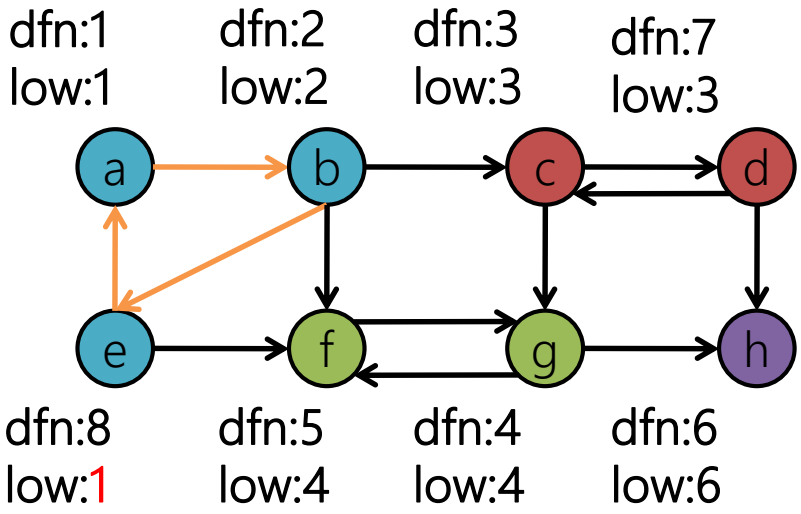




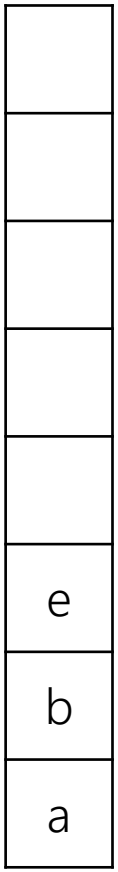
made by riljian

SCC

- Tarjan



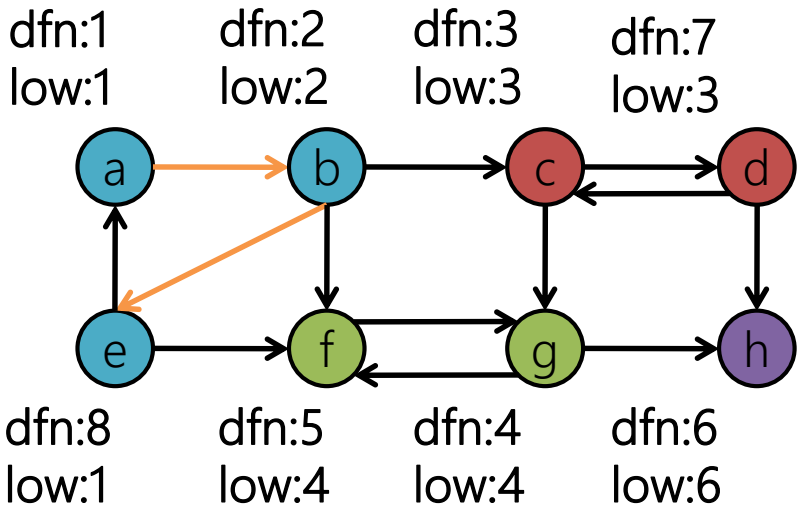
In stack

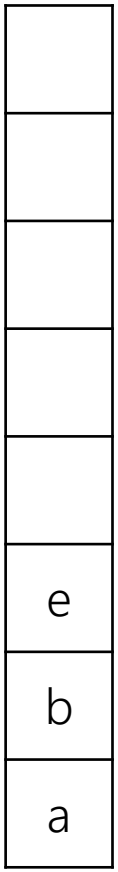


made by riljian

SCC

- Tarjan

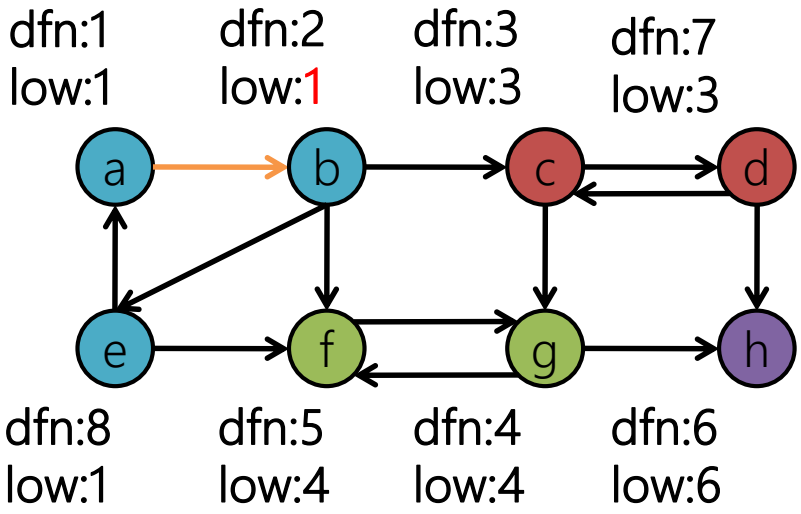


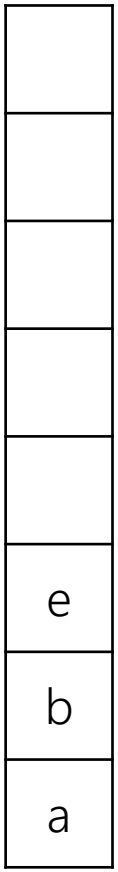


made by riljian

SCC

- Tarjan



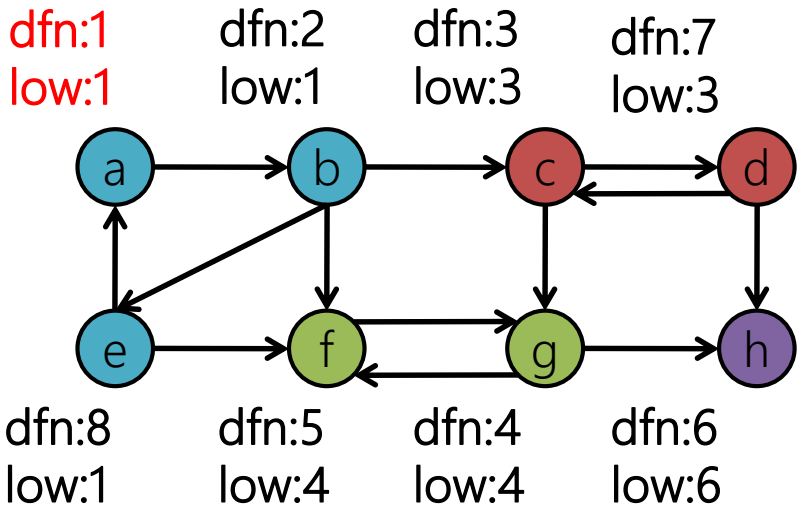


children's low < low

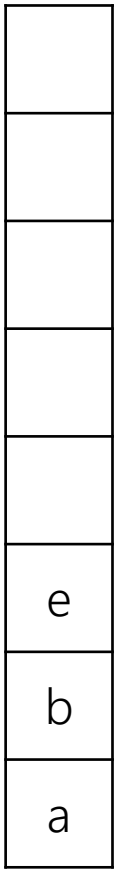
made by riljian

SCC

- Tarjan



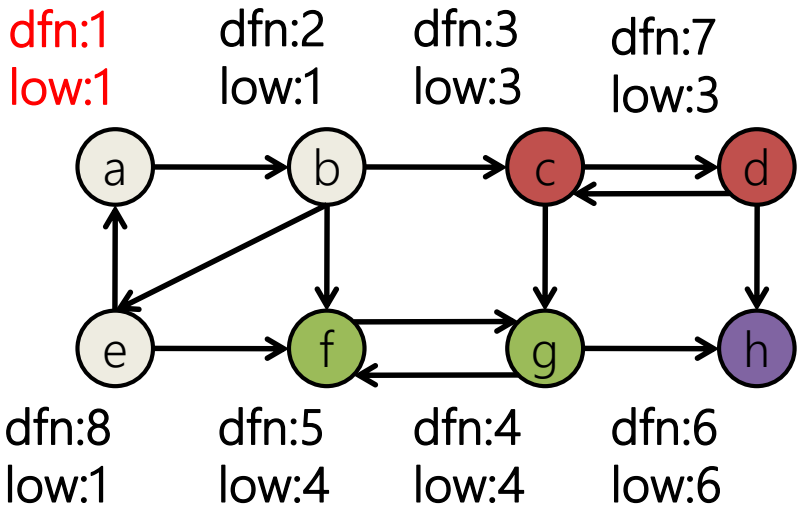
dfn == low

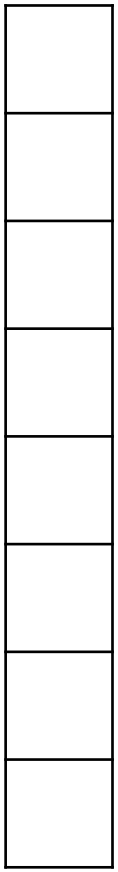


made by riljian

SCC

- Tarjan

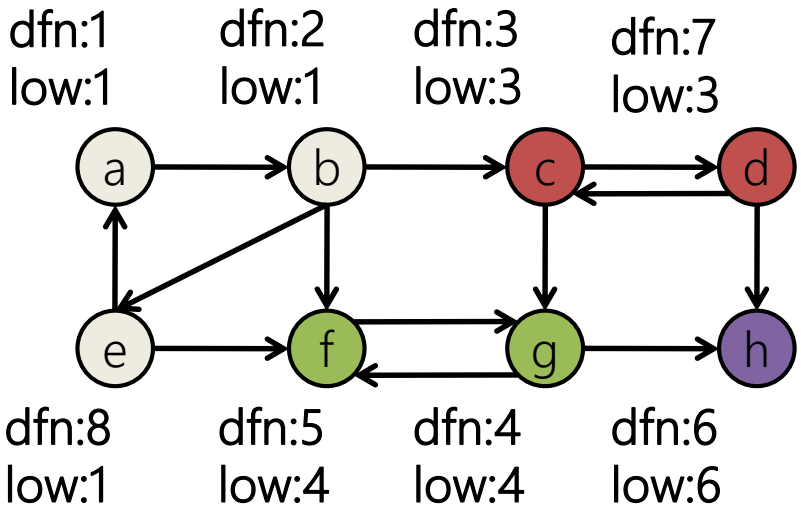


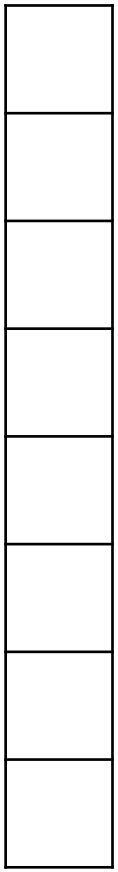


made by riljian

SCC

- Tarjan





4 strongly connected components

made by riljian

Practice

Uva - 11838

made by petermouse

Practice

acm International Collegiate Programming Contest

IBM event sponsor

UVa 11504

made by riljian