NCKU Programming Contest Training Course
Course 9
2015/03/25

Tzu-Yen Chiu(tommy5198)
tommy5198@*gmail.com*

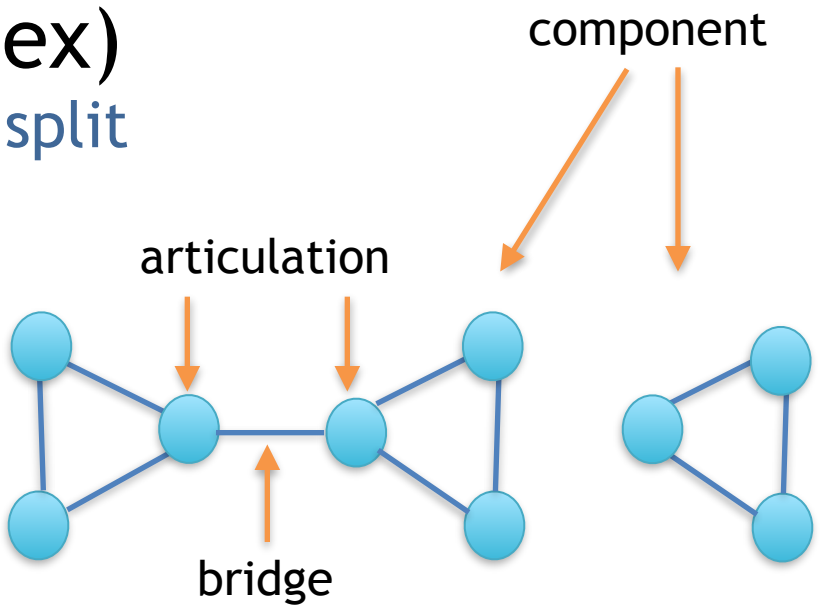
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

Outline

- Articulation/Bridge
- Strongly Connected Component(SCC)

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



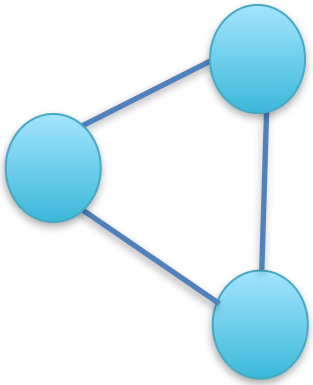
made by tommy5198

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)



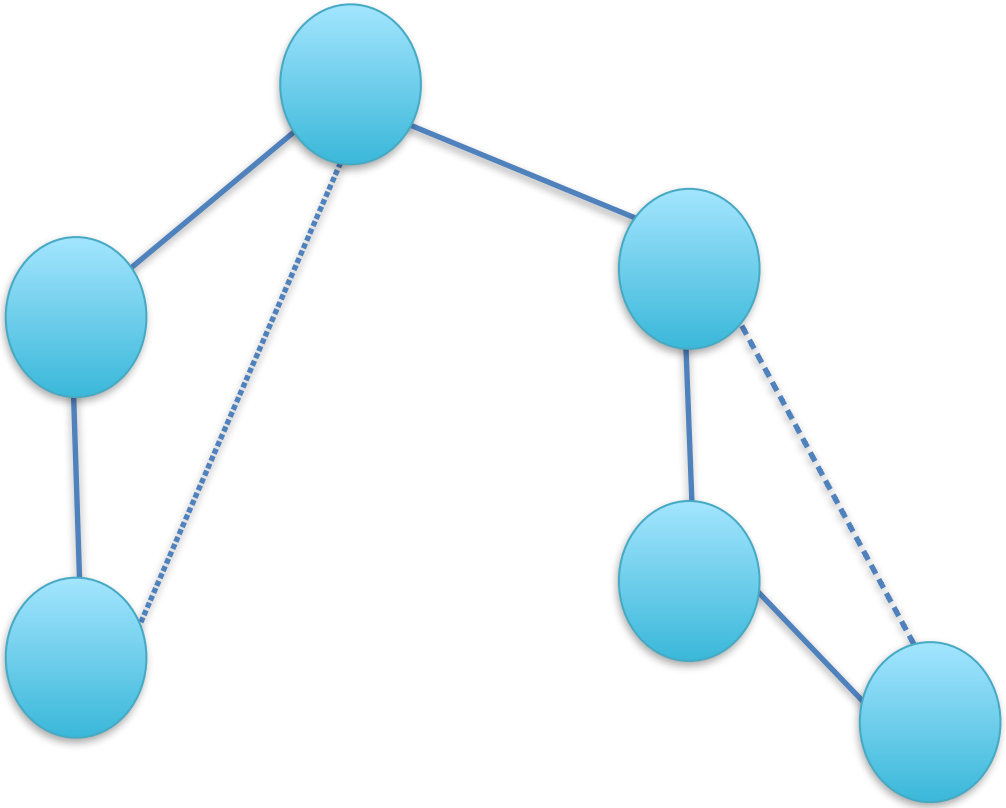
Articulation/Bridge

- Concept
  - 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

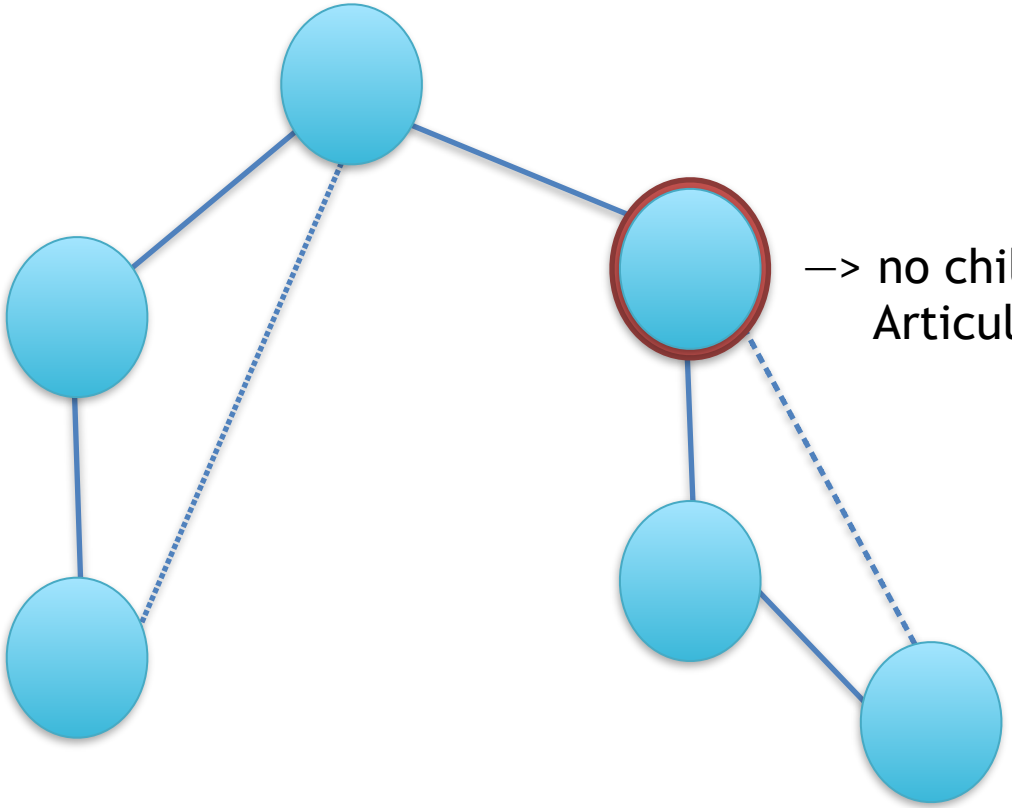
- Bridge?
  - two Articulation u, v have an edge —> (u, v) is Bridge!

Articulation/Bridge



made by tommy5198

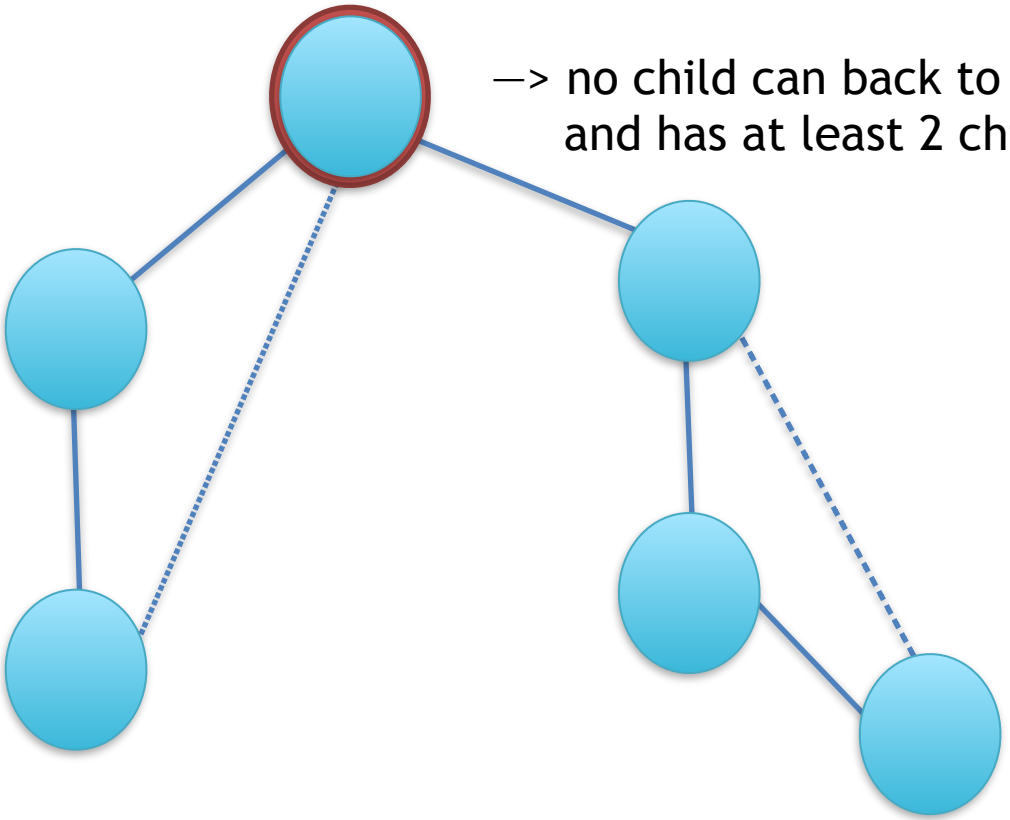
Articulation/Bridge



—> no child can back to ancestor.
Articulation!

made by tommy5198

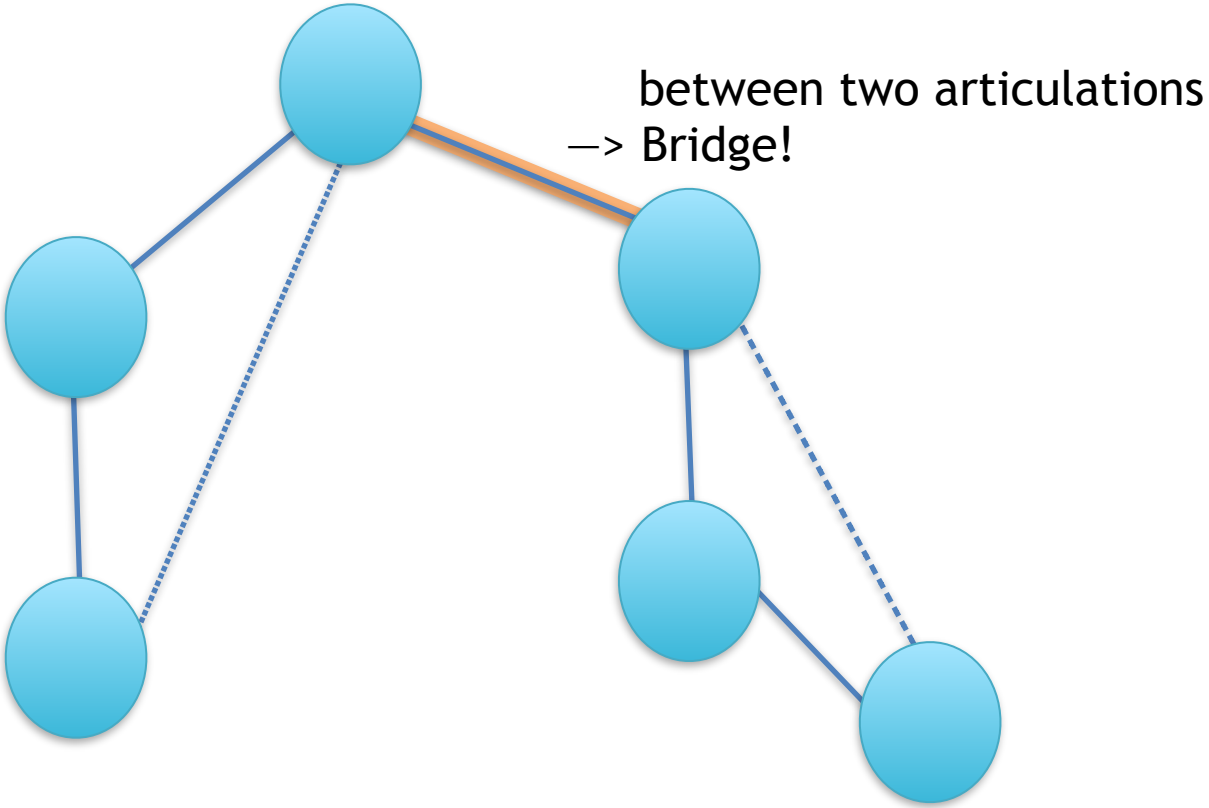
Articulation/Bridge



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

made by tommy5198

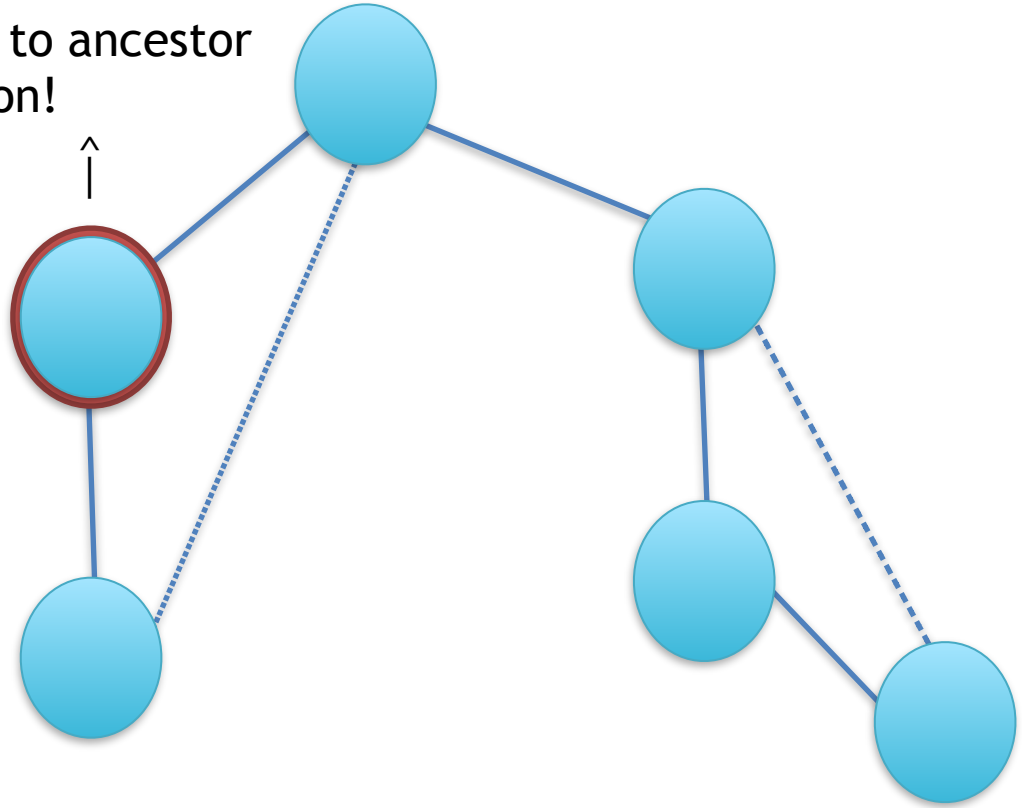
Articulation/Bridge



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

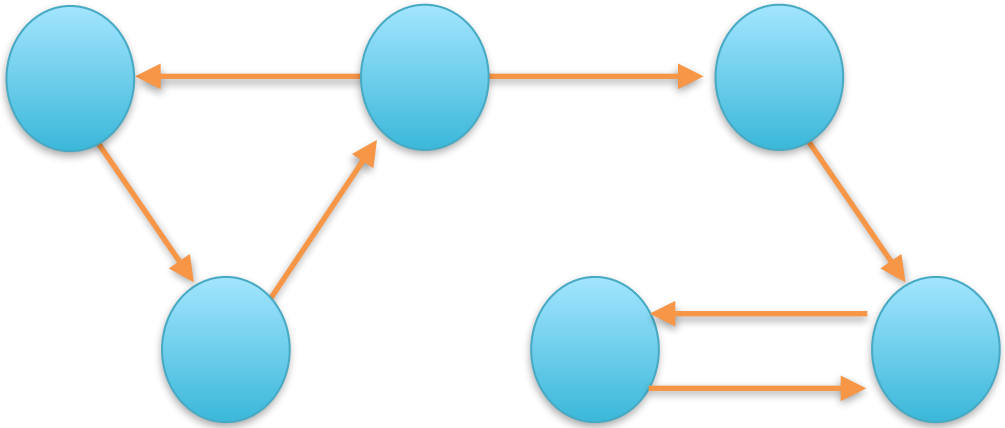
Practice

UVA - 315

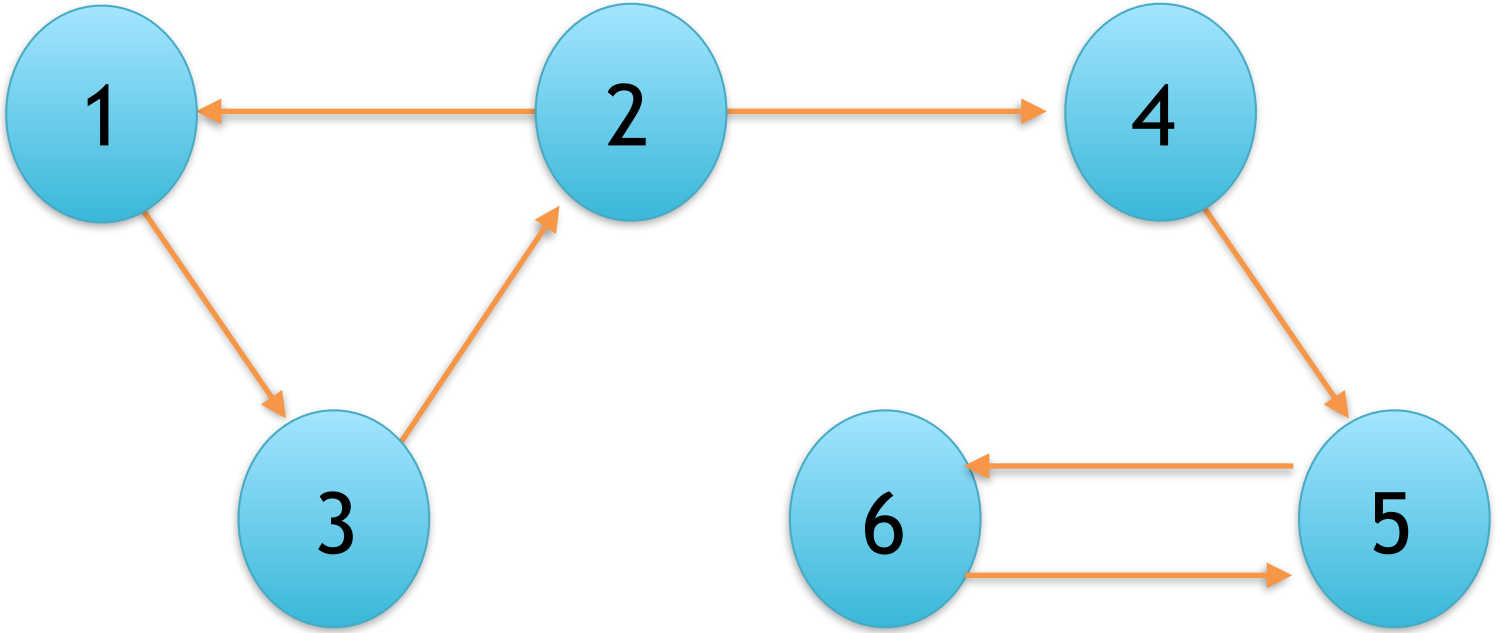
made by tommy5198

SCC

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

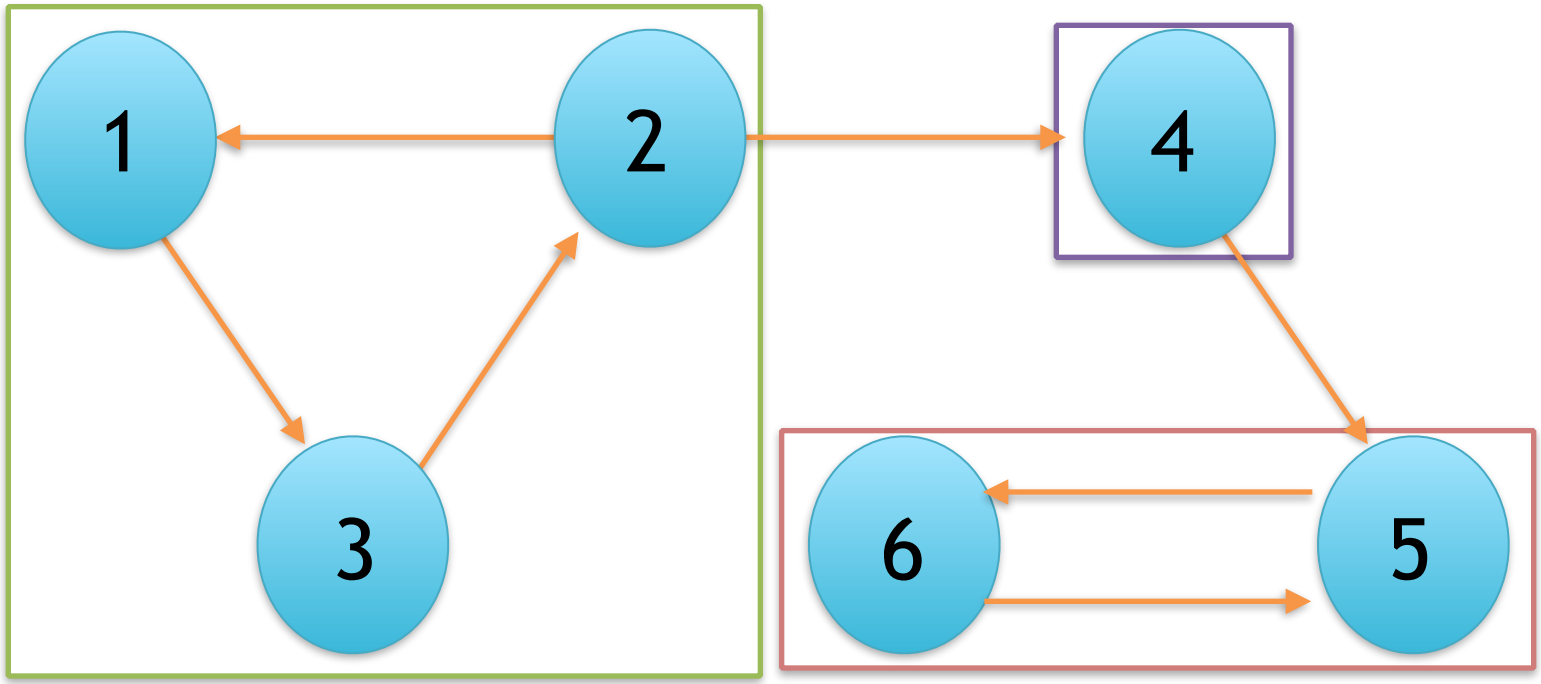


SCC



made by tommy5198

SCC

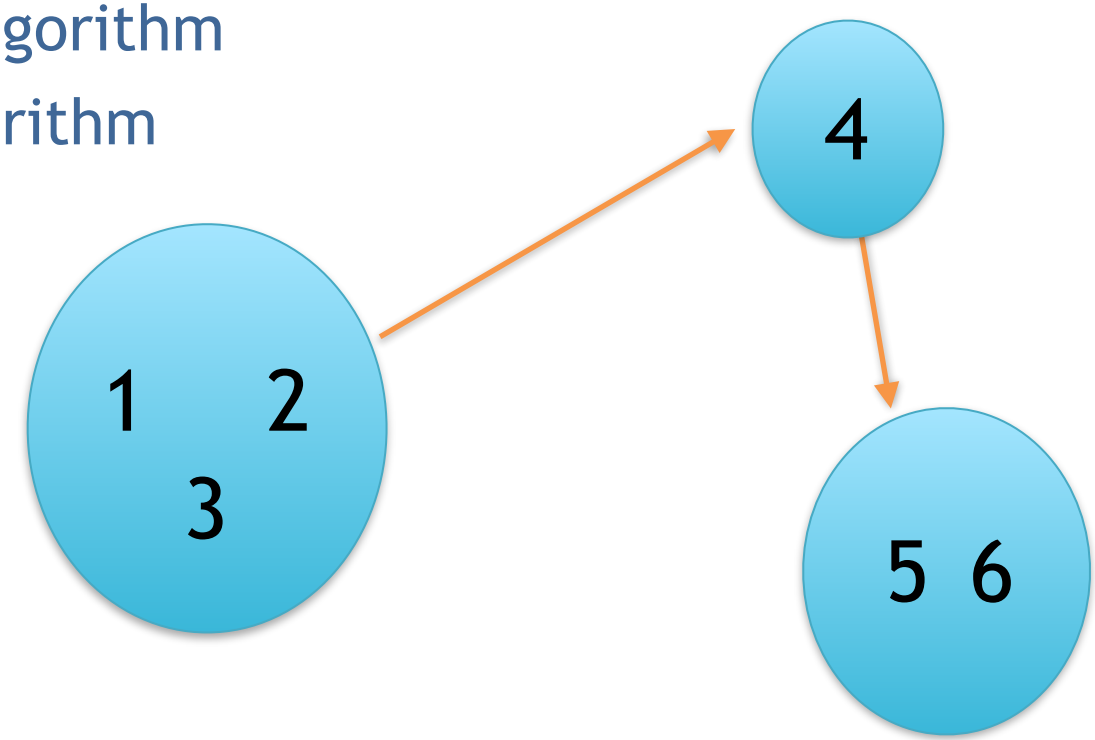


made by tommy5198

SCC

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

- Kosaraju's Algorithm
- Tarjan's Algorithm



made by tommy5198

SCC

- Kosaraju's algorithm

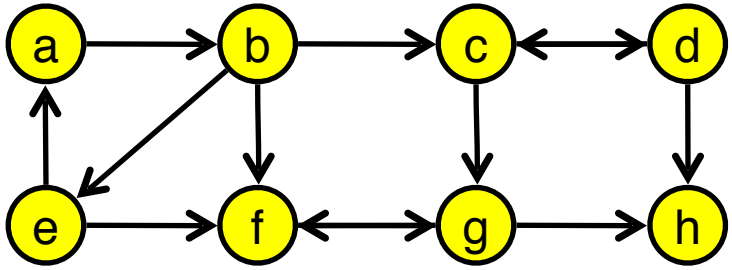
STRONGLY-CONNECTED-COMPONENTS(G)

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)

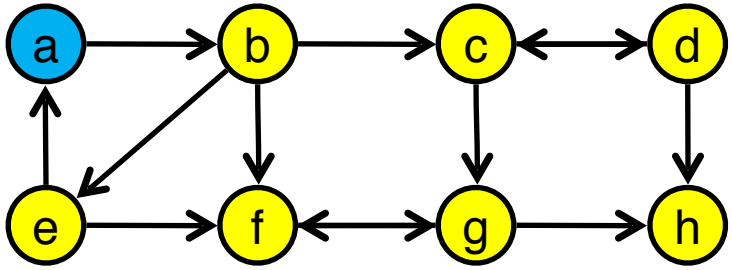
SCC

- Algorithm



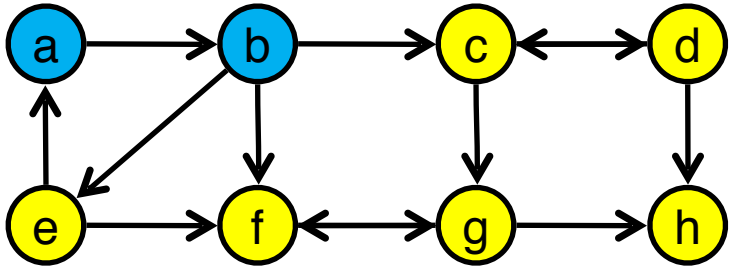
SCC

- Algorithm



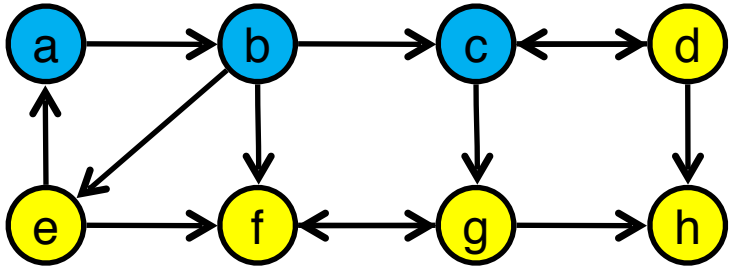
SCC

- Algorithm



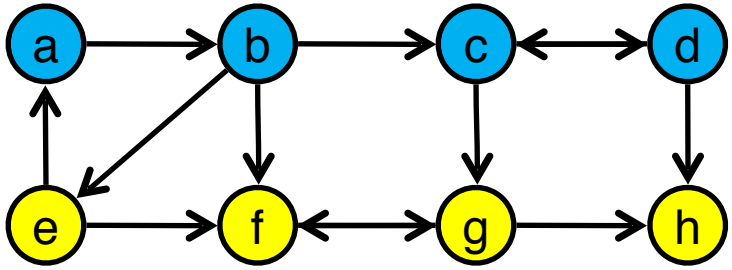
SCC

- Algorithm



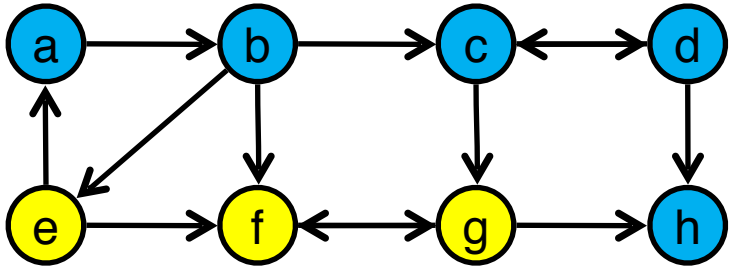
SCC

- Algorithm



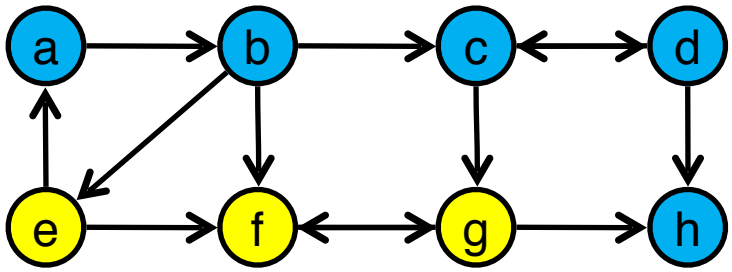
SCC

- Algorithm



SCC

- Algorithm

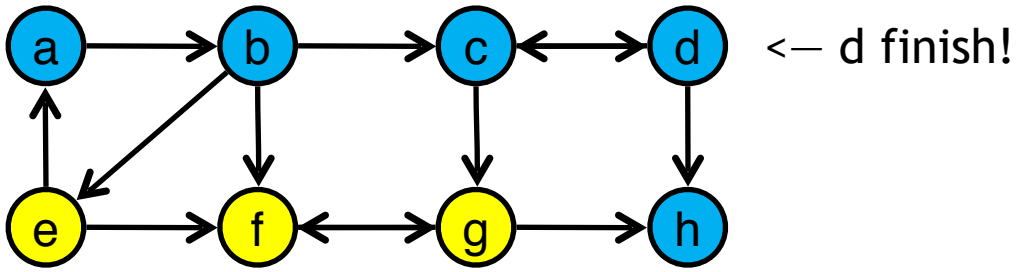


<— h finish!

h									

SCC

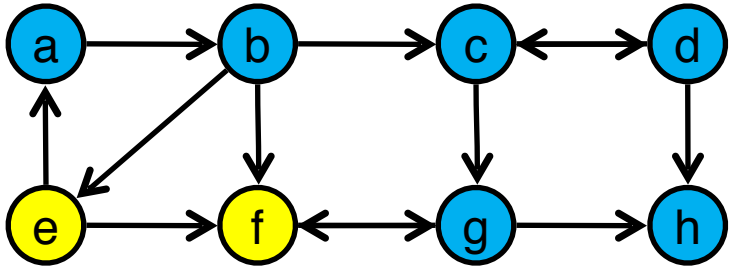
- Algorithm



h	d								

SCC

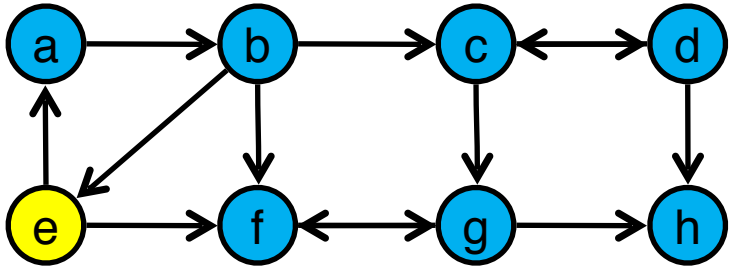
- Algorithm



h	d								

SCC

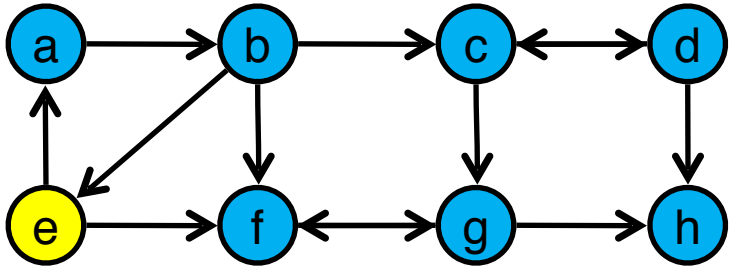
- Algorithm



h	d								

SCC

- Algorithm

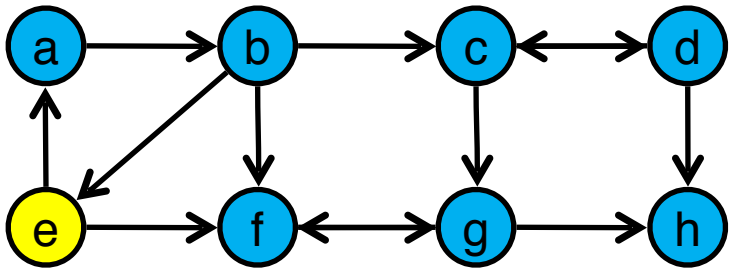


^ f finish!

h	d	f							

SCC

- Algorithm

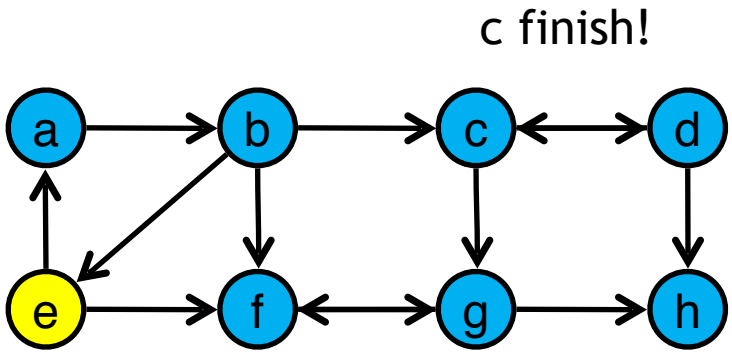


^ g finish!

h	d	f	g						

SCC

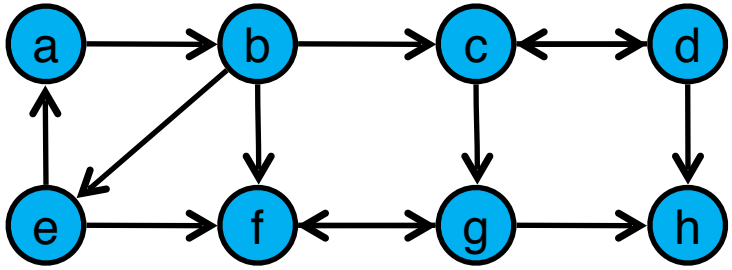
- Algorithm



h	d	f	g	c					

SCC

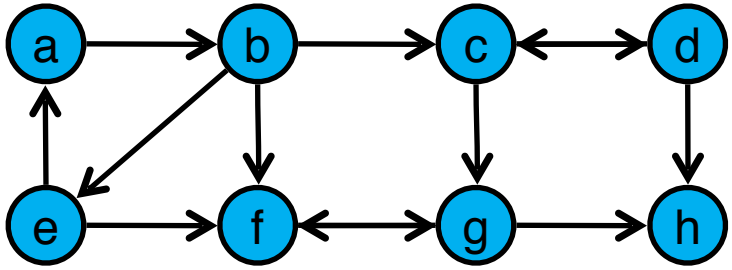
- Algorithm



h	d	f	g	c					

SCC

- Algorithm



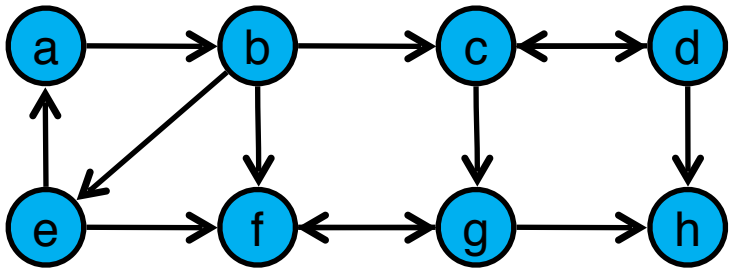
^ e finish!

h	d	f	g	c	e				

SCC

- Algorithm

b finish!

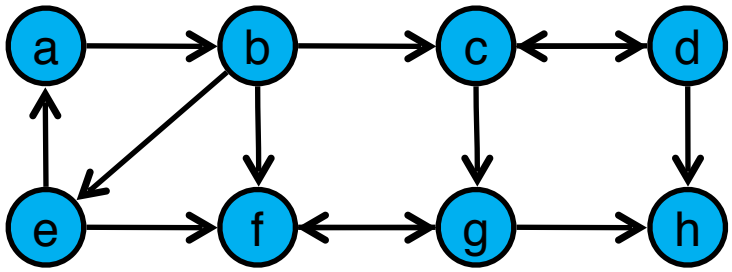


h	d	f	g	c	e	b			

SCC

- Algorithm

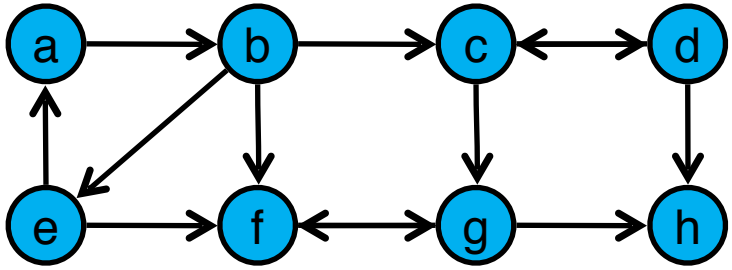
a finish!



h	d	f	g	c	e	b	a		

SCC

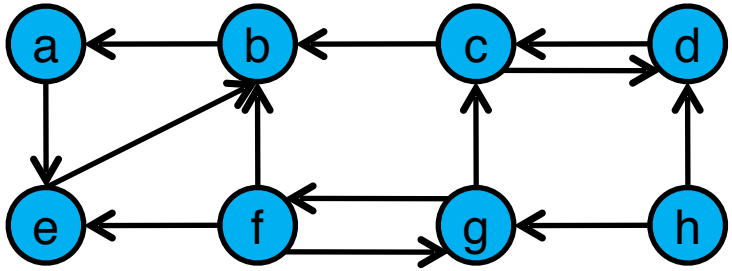
- Algorithm
  - Reverse the graph



h	d	f	g	c	e	b	a		

SCC

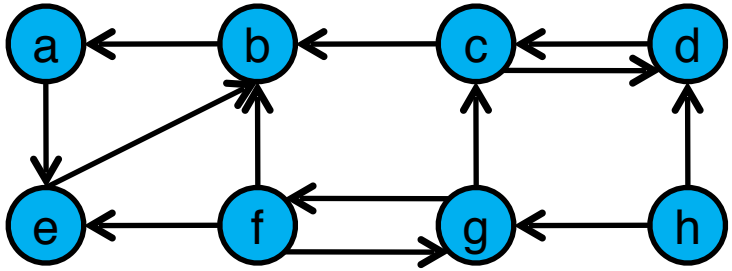
- Algorithm
  - Reverse the graph



h	d	f	g	c	e	b	a		

SCC

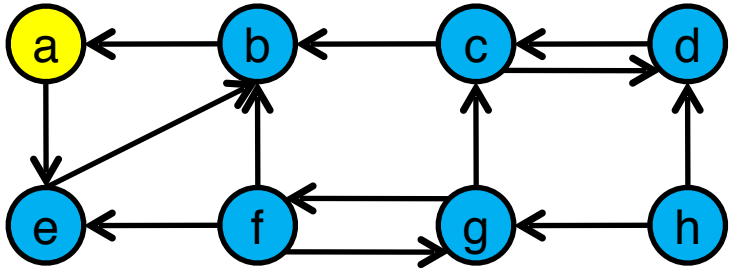
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

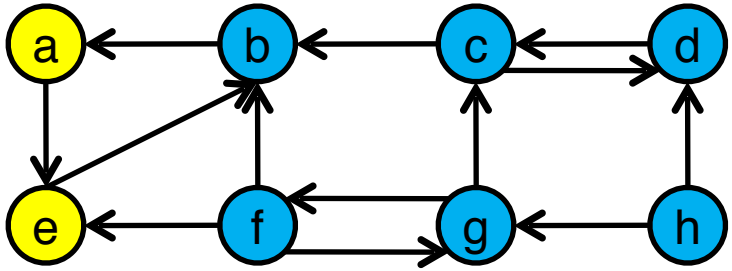
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

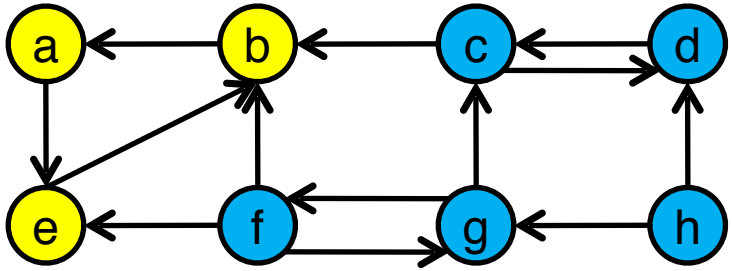
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

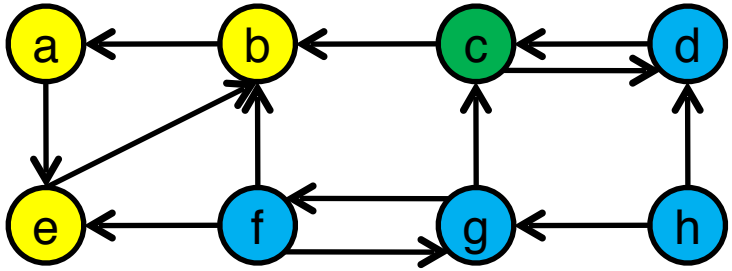
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

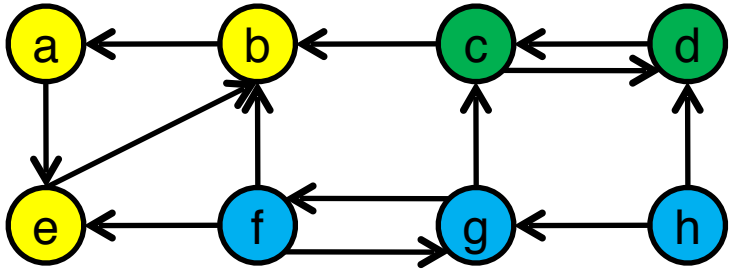
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

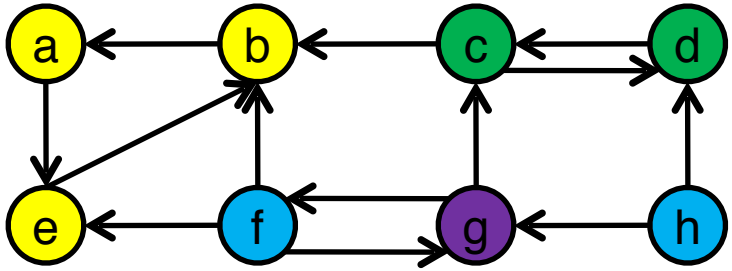
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



h	d	f	g	c	e	b	a		

SCC

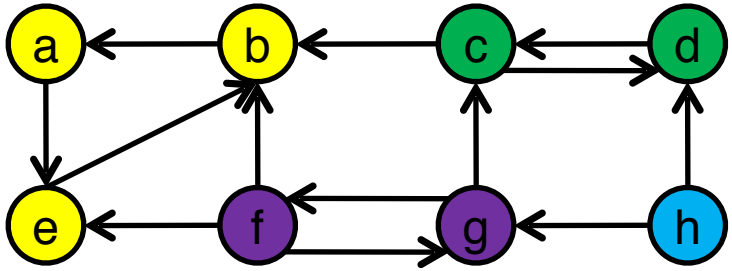
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



made by free999 & electron

SCC

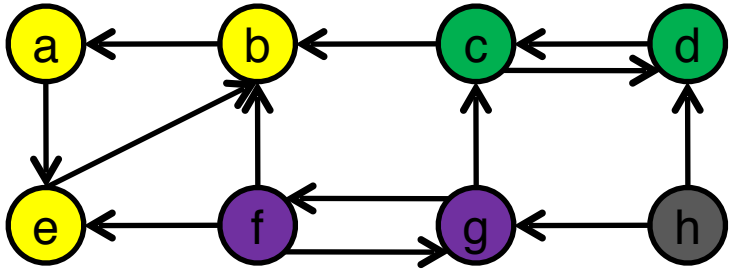
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



made by free999 & electron

SCC

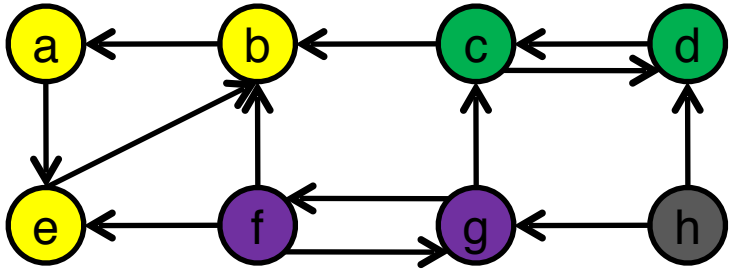
- Algorithm
  - Reverse the graph
  - Re-search by the ending time



SCC

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

4 components



Practice

ICPC - 4262

made by tommy5198

Learn more

- Tarjan's algorithm
  - only one DFS

made by tommy5198

Homework

- UVA (total 14 problems)
  - 247, 315, 459, 610, 796, 10199, 10731, 10765, 11324, 11504, 11709, 11770, 11838, 12783
- POJ (total 5 problems)
  - 1236, 1523, 2117, 2186, 2553
- ICPC (total 3 problems)
  - 4262, 4839, 5135

基本門檻 5 題，第二次修課同學請從橘色的題號選擇