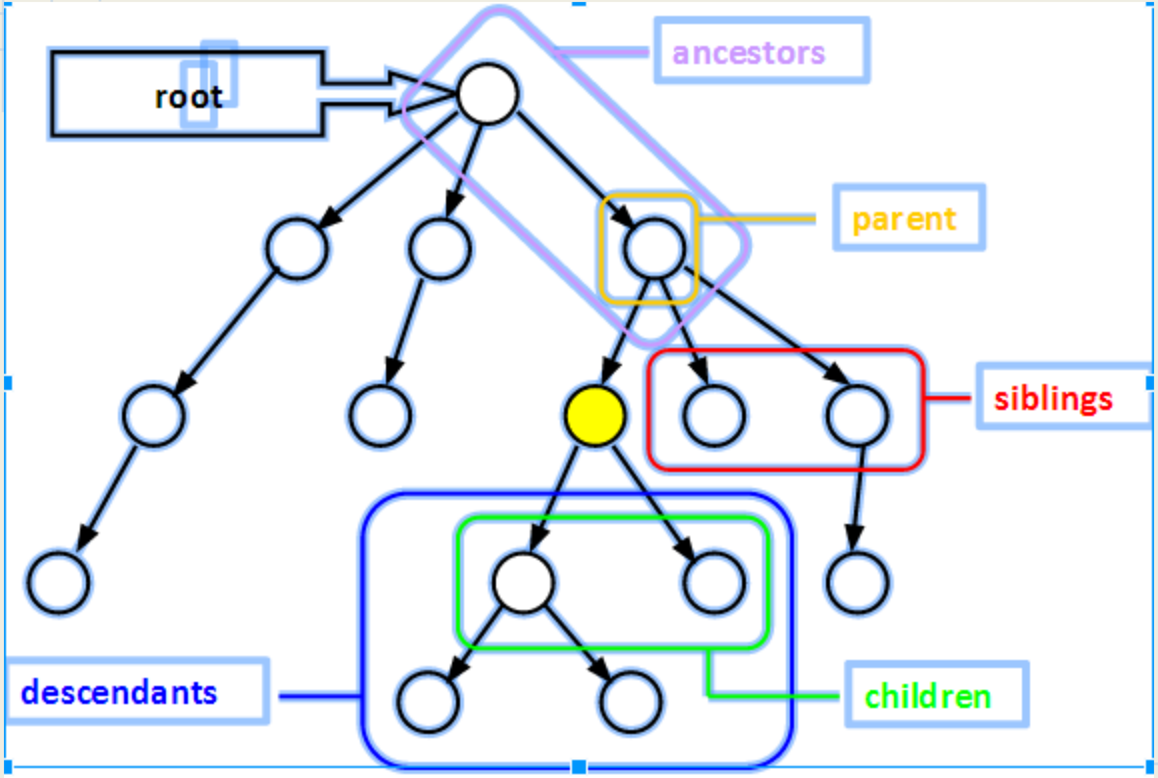
Graph (III)

Theo

Weapon

- Stack
- DFS/BFS

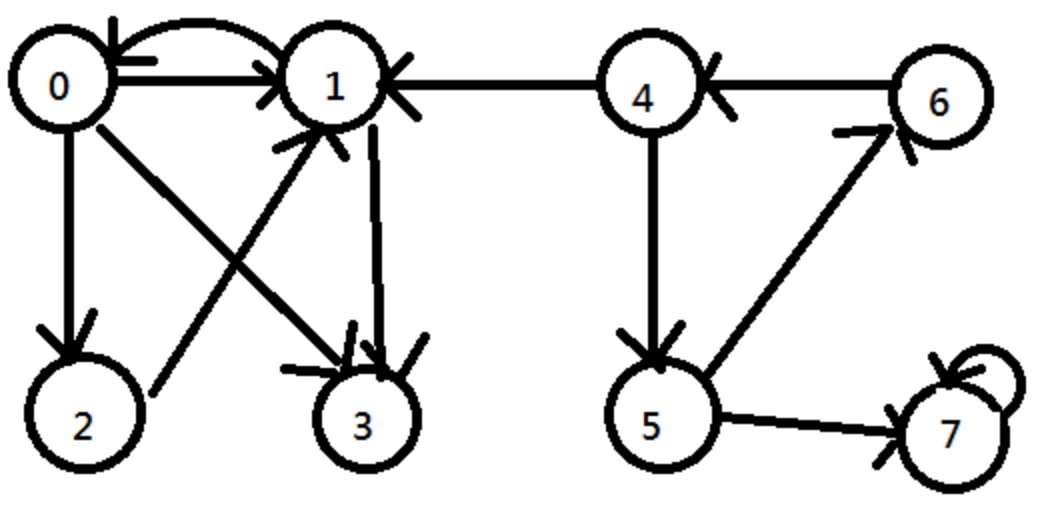
Terms on directed tree



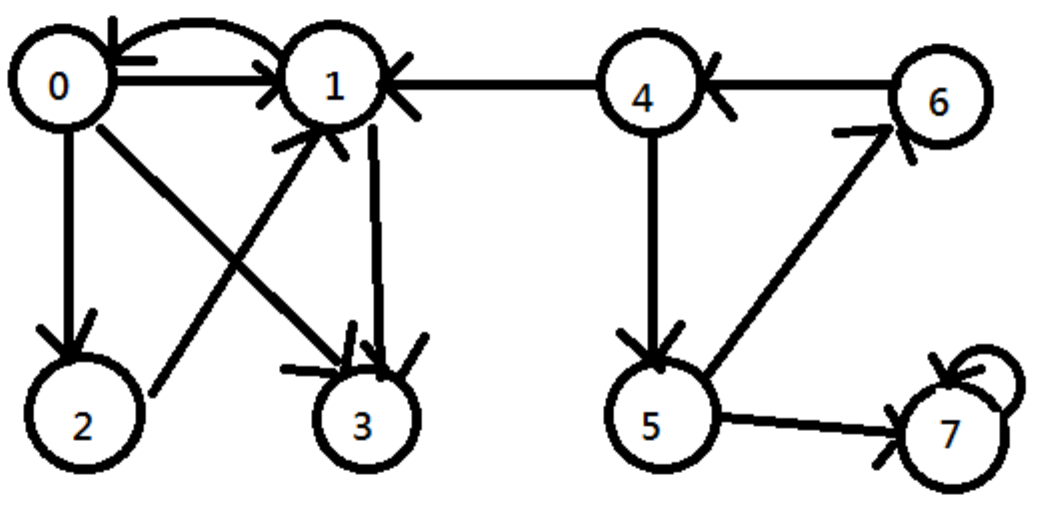
DFS Forest

- if you do a DFS on a graph (not necessary a tree!) you obtained a tree
- important information:
  - arrival(birth) time of a node
  - leaving(death) time of a node
  - parent of a node

Sample



Sample

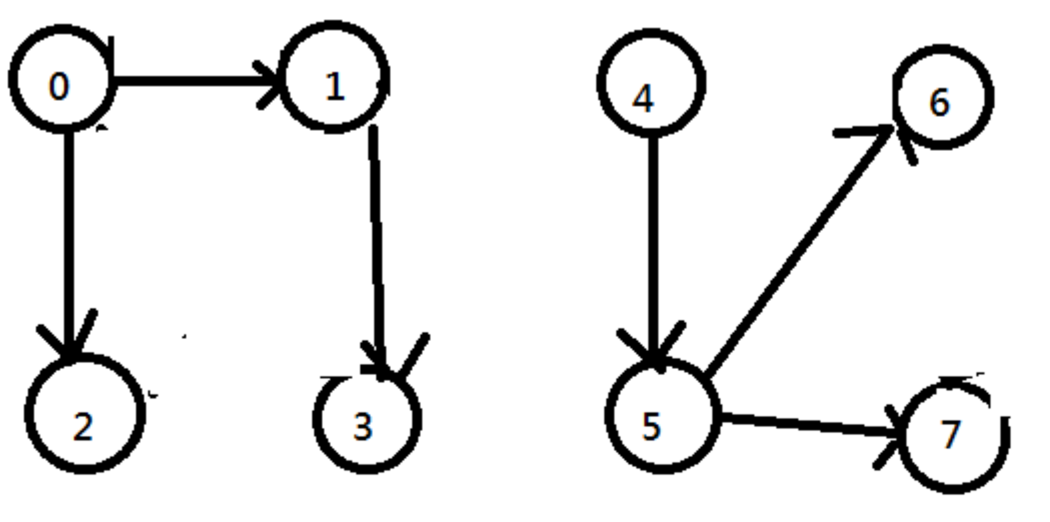


node	0	1	2	3	4	5	6	7
birth	1	2	6	3	9	10	11	13
death	8	5	7	4	16	15	12	14
parent	/	0	0	1	/	4	5	5

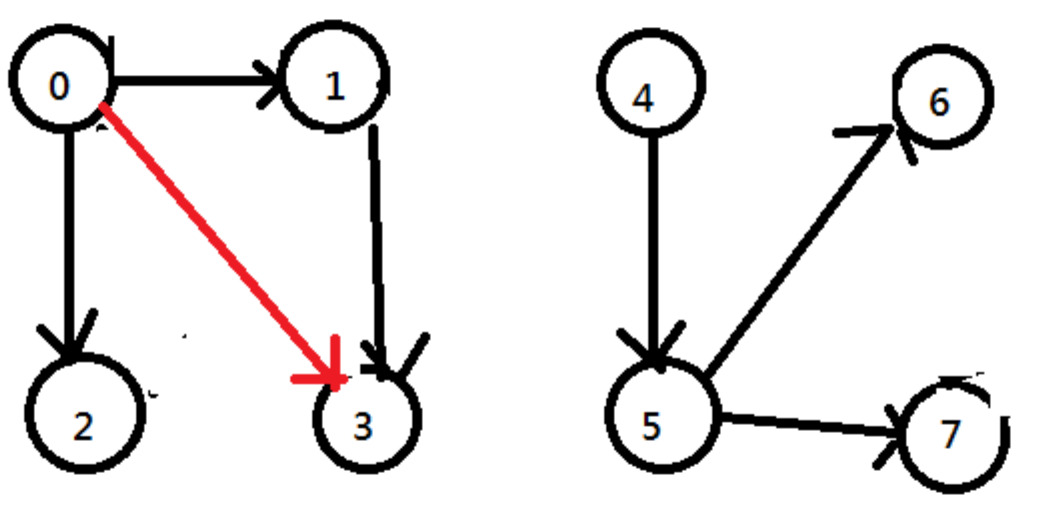
Types of edges

- Tree edges
- Forward edges
- Back edges
- Cross edges

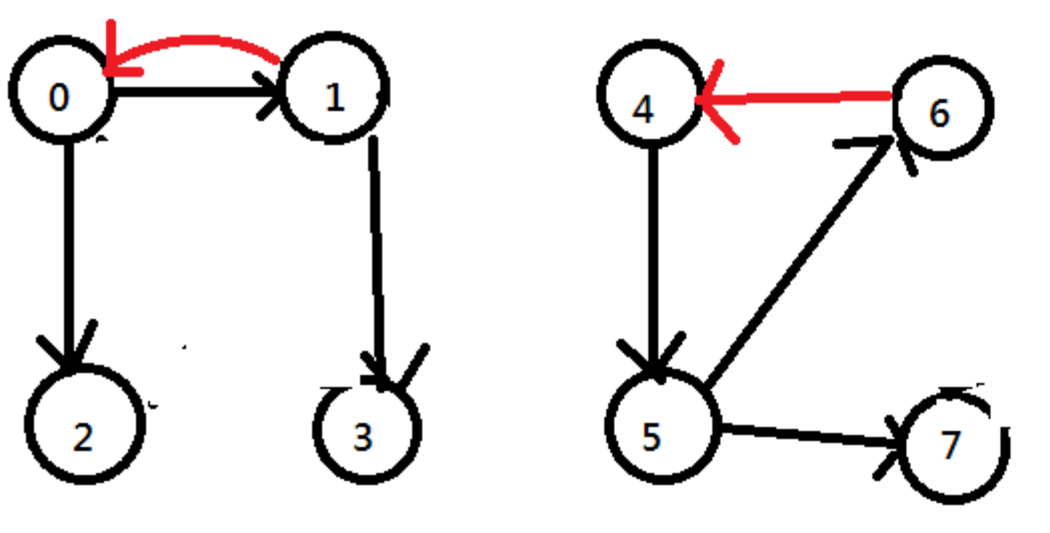
Forest built (tree edges)



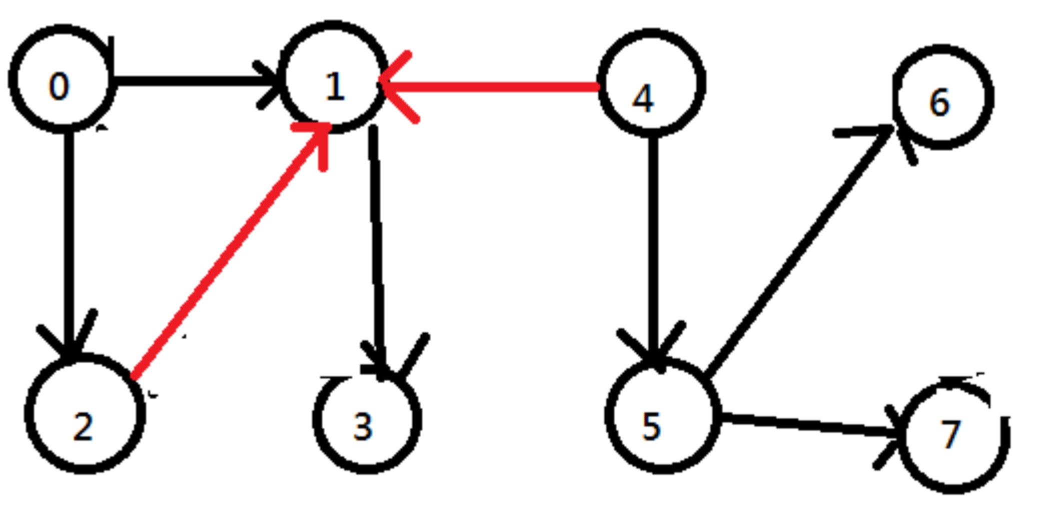
Forward edges



Back edges



Cross edges



Notice that it does not exist in undirected graph(why?)

Determination of edges

Tree edge:

    parent[v] = u

Forward edge:

    parent[v] != u
    birth[v] > birth[u]
    death[v] < death[u]

Determination of edges

Back edge

- birth[v] < birth[u]
- death[v] > death[u]

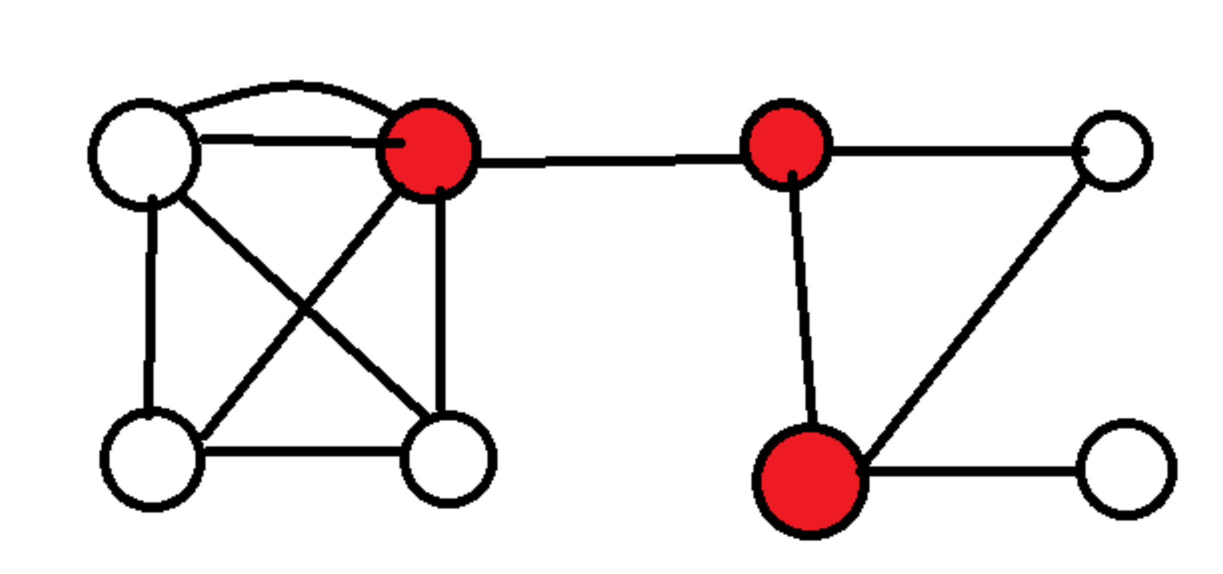
Cross edge

- birth[v] < birth[u]
- death[v] < death[u]

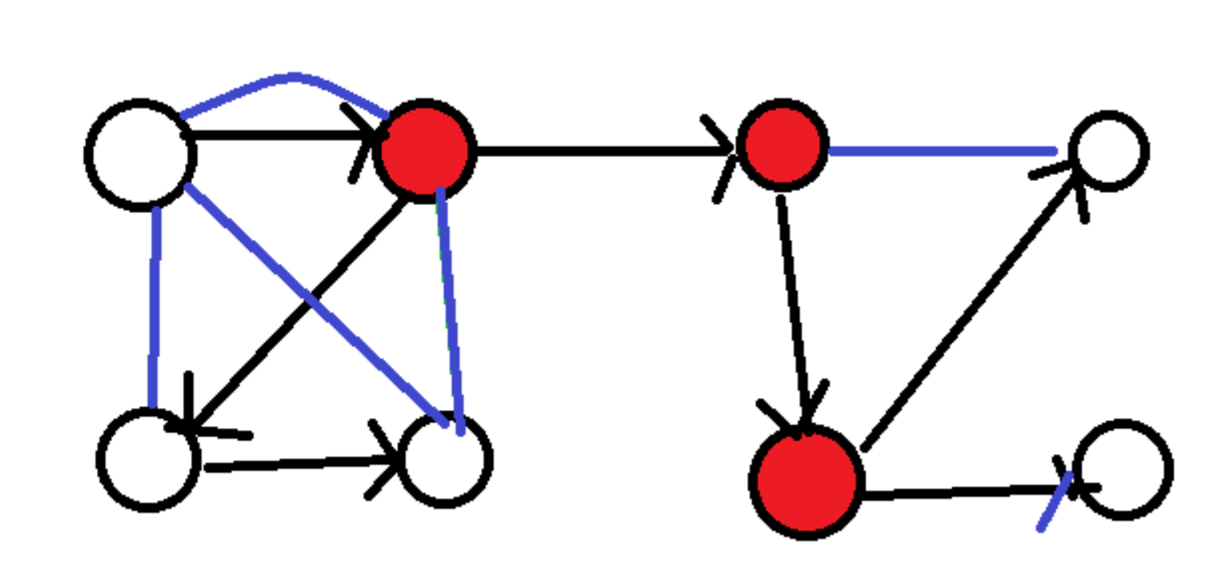
Articulation Point

In a undirected graph, articulation point is a vertex such that if it is removed, number of component will increase

Articulation Point



Relationship to DFS forest



Articulation Point

A vertex, say u, is articulation point if and only if no blue(back) edges by u’s descendatns links to u’s ancestors

How to check?
Brute Force-->O(N+M)
Brute Force for all nodes --> O(NM)

Articulation Point

Define, for all vertex, low[]: the lowest birth time node that it can find from DFSing this vertex.

u is AP --> for children v, low[v] >= birth[u]
OR for root, child[u] > 1

how to find low[]?

Articulation Point

low[u] = min(birth[u],
low[v]: (u, v) is a tree edge,
birth[v]: (u, v) is a back edge)

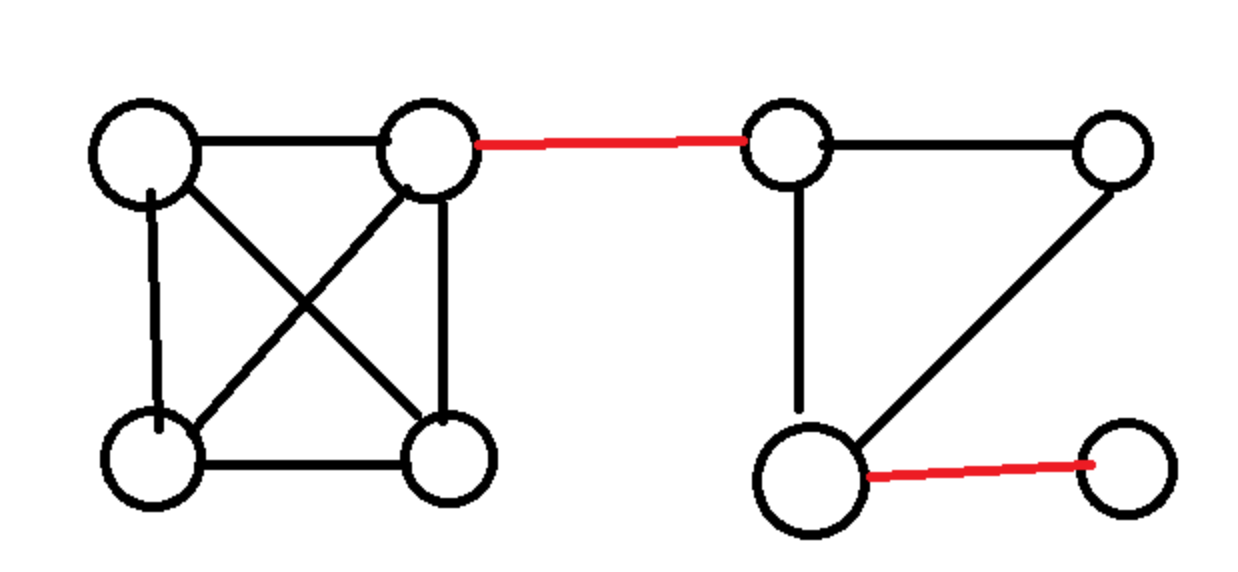
Example

http://poj.org/problem?id=1523

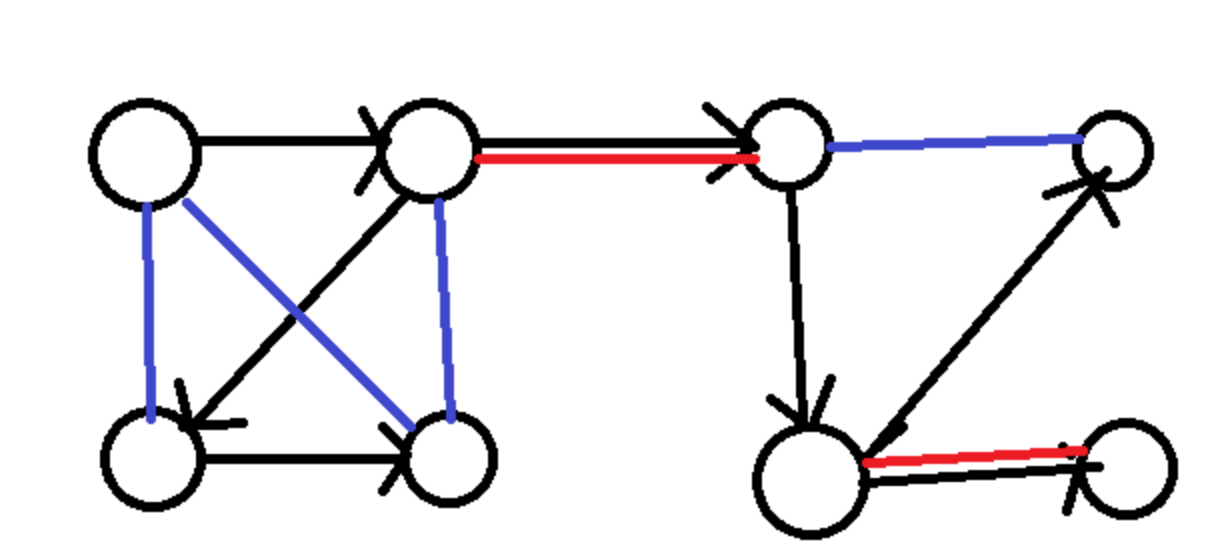
Bridge

Similar to AP
A edge is a bridge if removing it will increase the number of components

Bridge



Relationship to DFS forest



Bridge

Back edges can not be bridge (why?)

A edge(u, v) is a bridge if and only if (u, v) is a tree edge and we cannot find any path from v(and v’s decendants) to u(and u’s ancestors) besides (u,v)

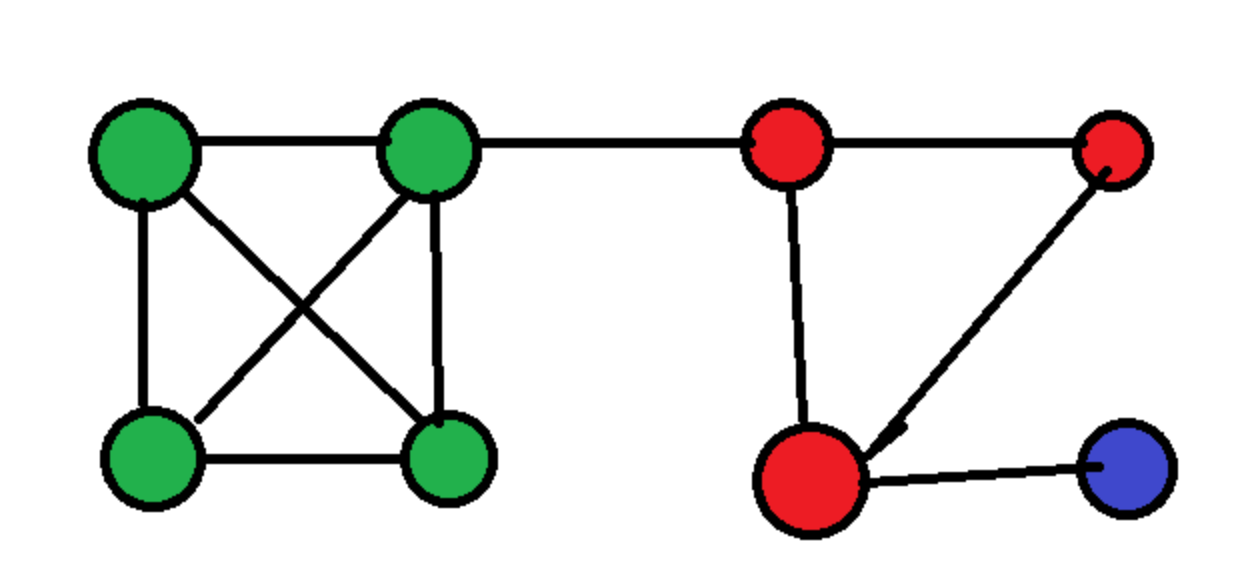
--> low[v] > birth[u]

Bi-connected component

Consider a undirected graph, if one of the connected subgraph contains no bridge, it is said to be a bi-connected subgraph

Bi-connected component is bi-connected subgraph that cannot extend anymore

Bi-connected component



Bi-connected component

Facts:

- Bridges connect bi-connected components
- AP belongs to 2+ bi-connected components

Bi-connected component

u and v are in one bi-connected component

  if low[u] == low[v] (why?)

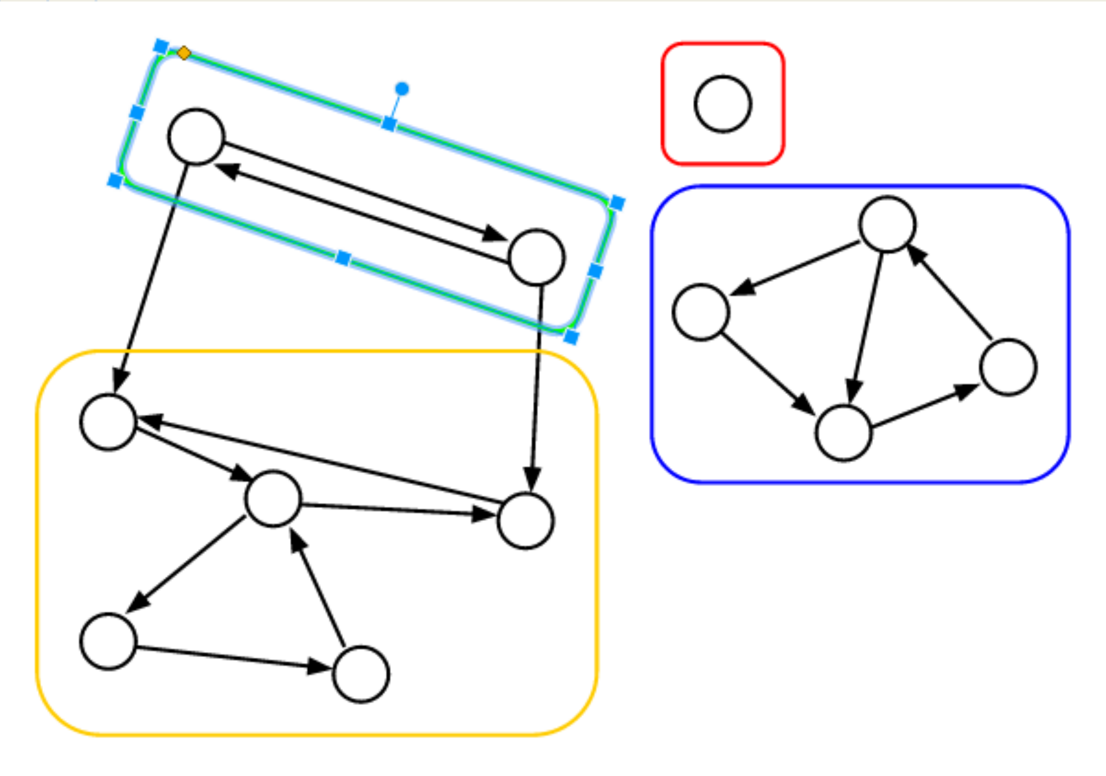
Example

http://poj.org/problem?id=3352

Strongly Connected Component

Consider a directed graph, a inextendable connected subgraph is said to be strongly connected component such that for every pairwise vertex u, v(u and v in component) there exists a path from u to v and from v to u

Strongly Connected Component



Strongly Connected Component

Let S be a empty stack
Perform a DFS on G (start from any node)
  each time we leave from a node, push that node into S
Reverse the direction of edges
Perform DFS on G by poping nodes from S
each DFS visits a set of node and that set of node is a SSC

Examples

How about “weak connected components” ?

http://poj.org/problem?id=2762