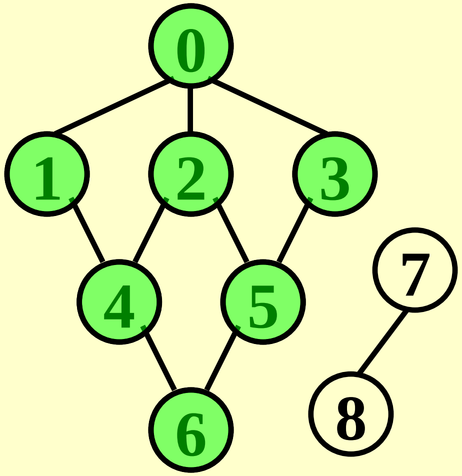
§6 Applications of Depth-First Search

/* a generalization of preorder traversal */

```
void DFS ( Vertex V )  /* this is only a template */
{   visited[ V ] = true;  /* mark this vertex to avoid cycles */
    for ( each W adjacent to V )
        if ( !visited[ W ] )
            DFS( W );
} /* T = O( |E| + |V| ) as long as adjacency lists are used */
```

1. Undirected Graphs

DFS (0)




```
void ListComponents ( Graph G )
{   for ( each V in G )
        if ( !visited[ V ] ) {
            DFS( V );
            printf("\n");
        }
}
```

0 1 4 6 5 2 3
7 8

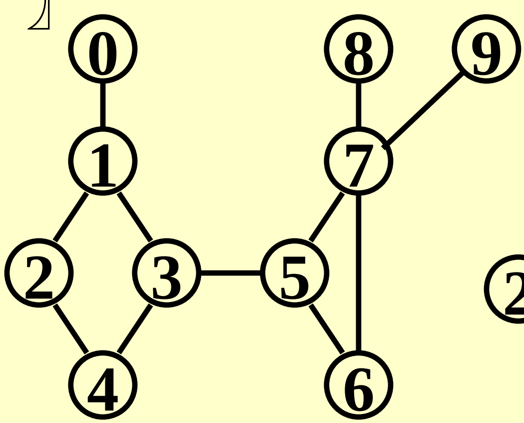
2. Biconnectivity

- *v* is an **articulation point** if G' = DeleteVertex(G, *v*) has **at least 2** connected components.
- G is a **biconnected graph** if G is connected and has no articulation points.
- A **biconnected component** is a maximal biconnected subgraph.

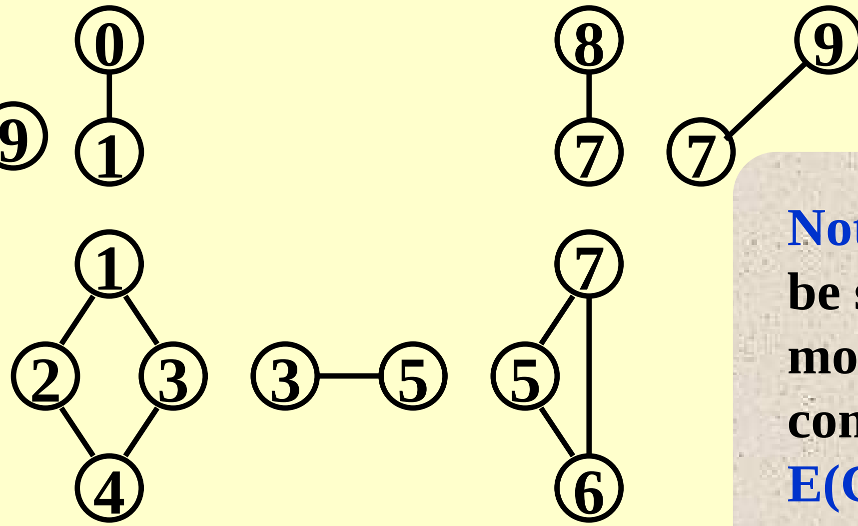
Biconnected graph

iculation point

〖Example〗



Connected graph



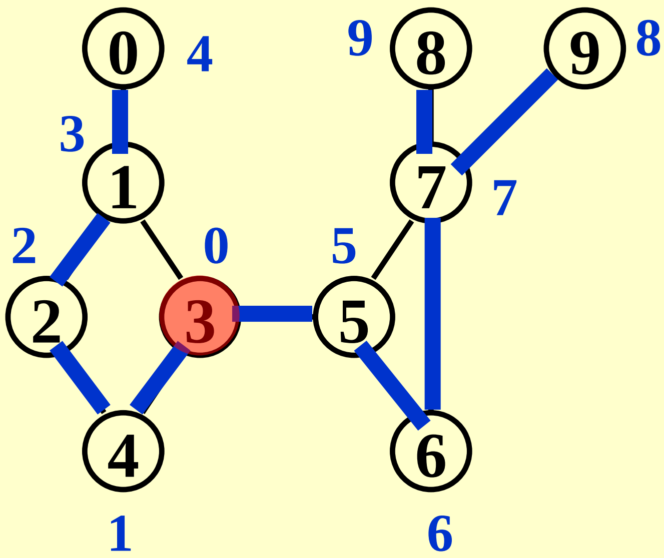
Biconnected components

Note: No edges can be shared by two or more biconnected components. Hence **E(G) is partitioned** by the biconnected components of G.

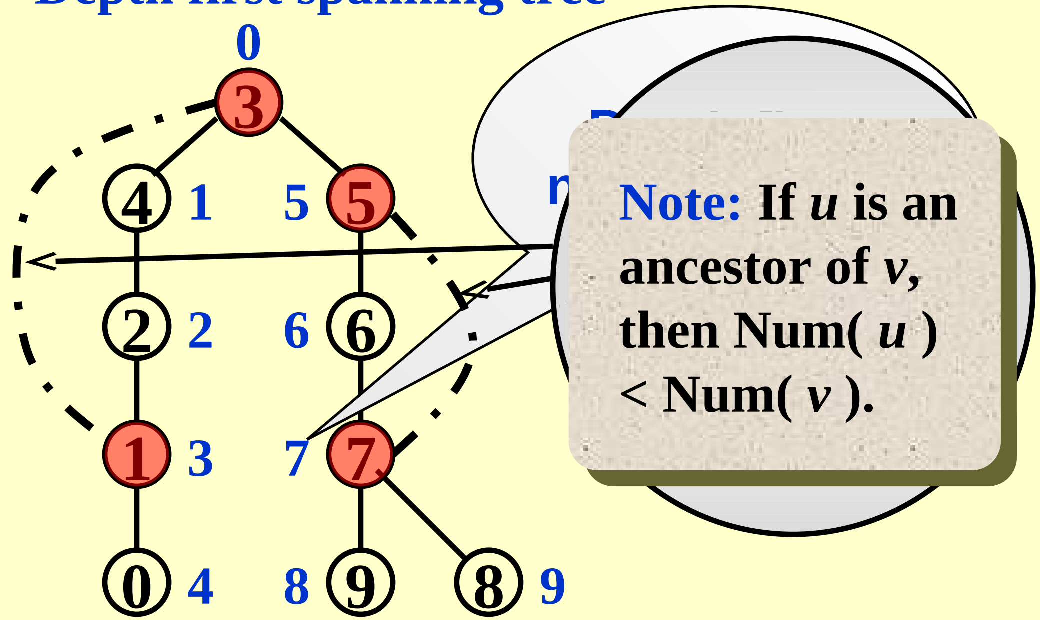
Finding the biconnected components of a connected undirected G

- Use **depth first search** to obtain a spanning tree of G

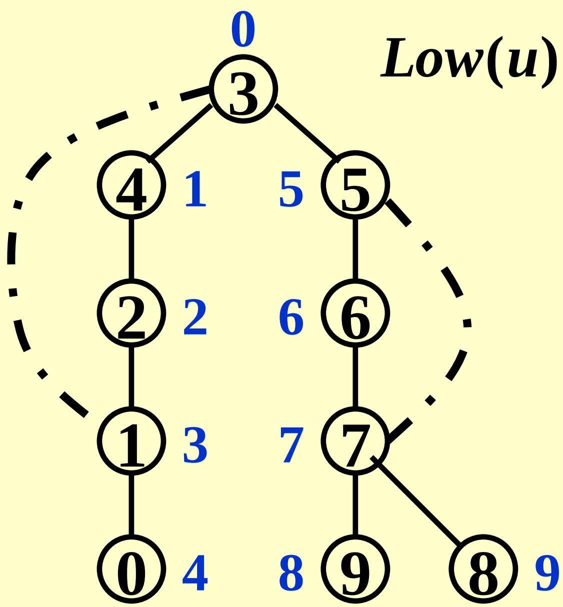
DFS (3)



Depth first spanning tree



- Find the **articulation points** in G
  - The **root** is an articulation point iff it has **at least 2 children**
  - Any **other vertex u** is an articulation point iff u has **at least 1 child**, **and** it is impossible to **move down at least 1 step and then jump up to u's ancestor**.



$Low(u) = \min\{Num(u),$
$\quad \min\{Low(w) \mid w \text{ is a child of } u\},$
$\quad \min\{Num(w) \mid (u, w) \text{ is a back edge}\}\}$

vertex	0	1	2	3	4	5	6	7	8	9
Num	4	3	2	0	1	5	6	7	9	8
Low	4	0	0	0	0	5	5	5	9	8

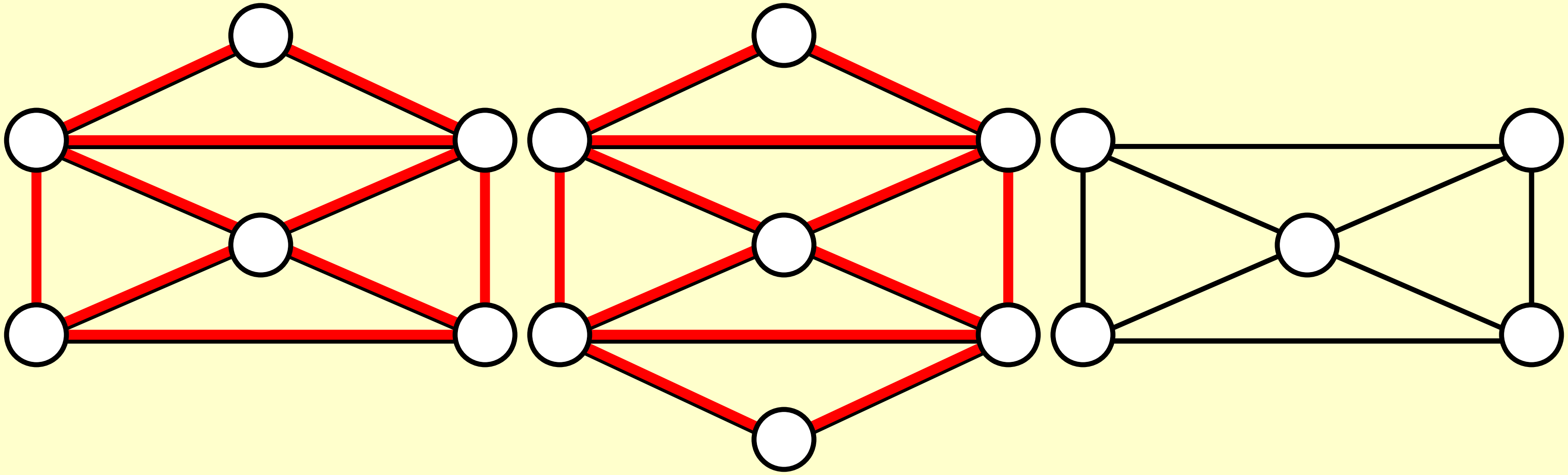
Therefore, *u* is an articulation point iff

(1) *u* is the root and has at least 2 children; or

(2) *u* is not the root, and has at least 1 child such that *Low(child)* ≥ *Num(u)*.

Please read the pseudocodes on p.327 and p.329 for more details.

3. Euler Circuits

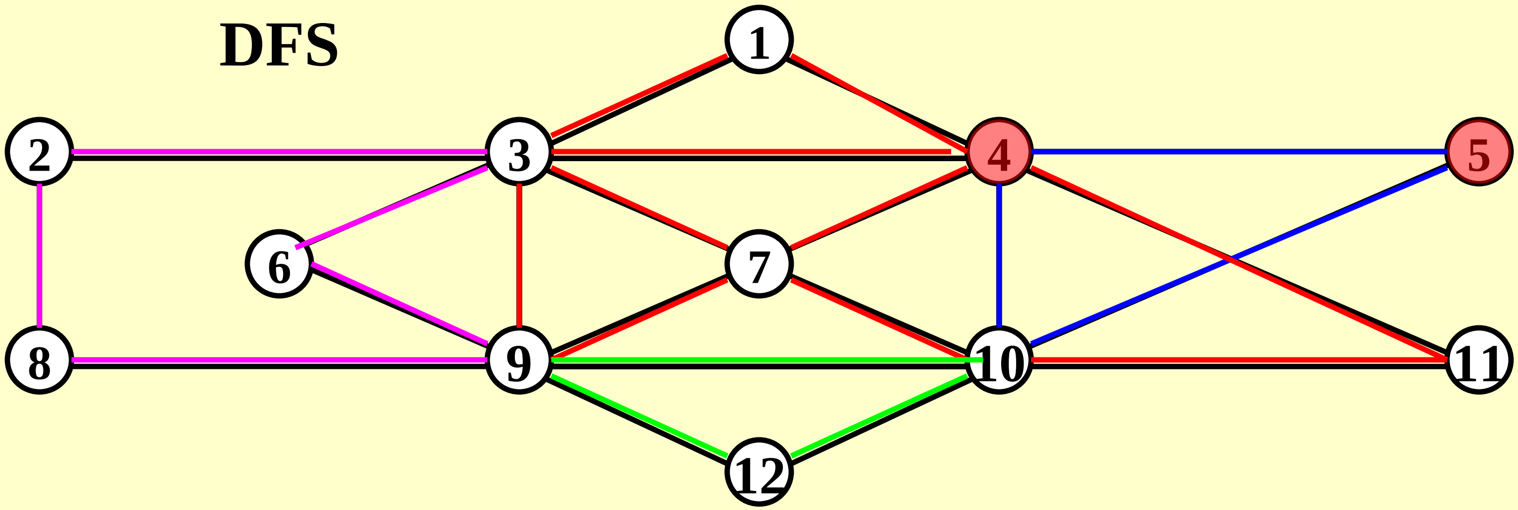


- **Draw each line exactly once without lifting your pen from the paper – *Euler tour***
- **Draw each line exactly once without lifting your pen from the paper, AND finish at the starting point – *Euler curcuit***

〖Proposition〗 **An Euler circuit is possible only if the graph is connected and each vertex has an even degree.**

〖Proposition〗 **An Euler tour is possible if there are exactly two vertices having odd degree. One must start at one of the odd-degree vertices.**

DFS



Note:

- **The path should be maintained as a linked list.**
- **For each adjacency list, maintain a pointer to the last edge scanned.**
- $T = O(|E| + |V|)$

Find a simple cycle in an undirected graph that visits every vertex – ***Hamilton cycle***