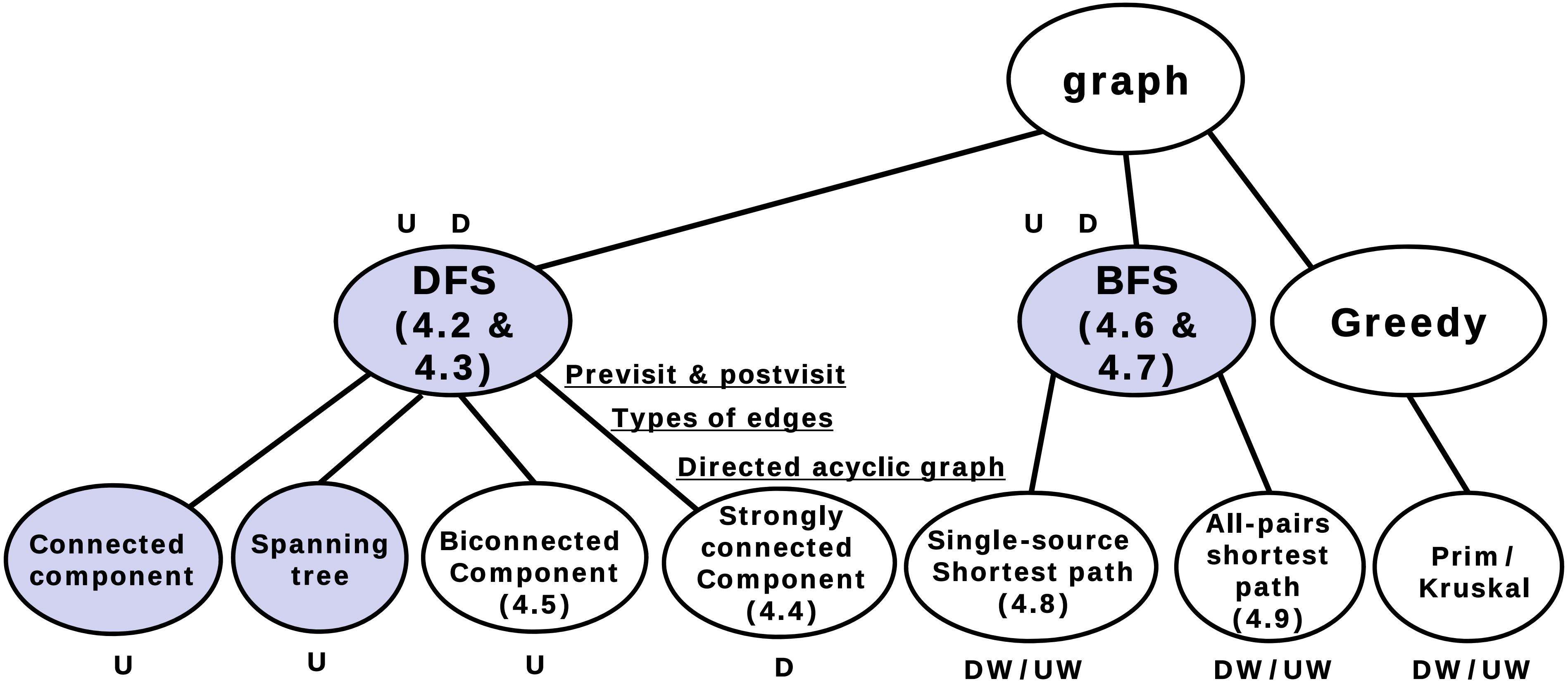
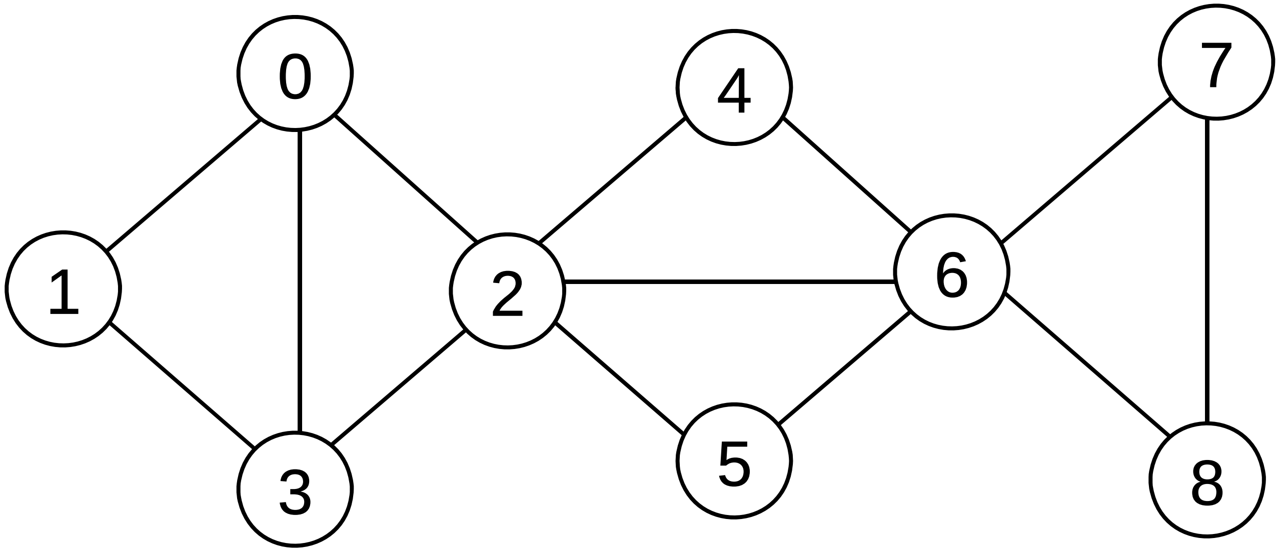
4.5 Biconnected Components

Classification of graph algorithms



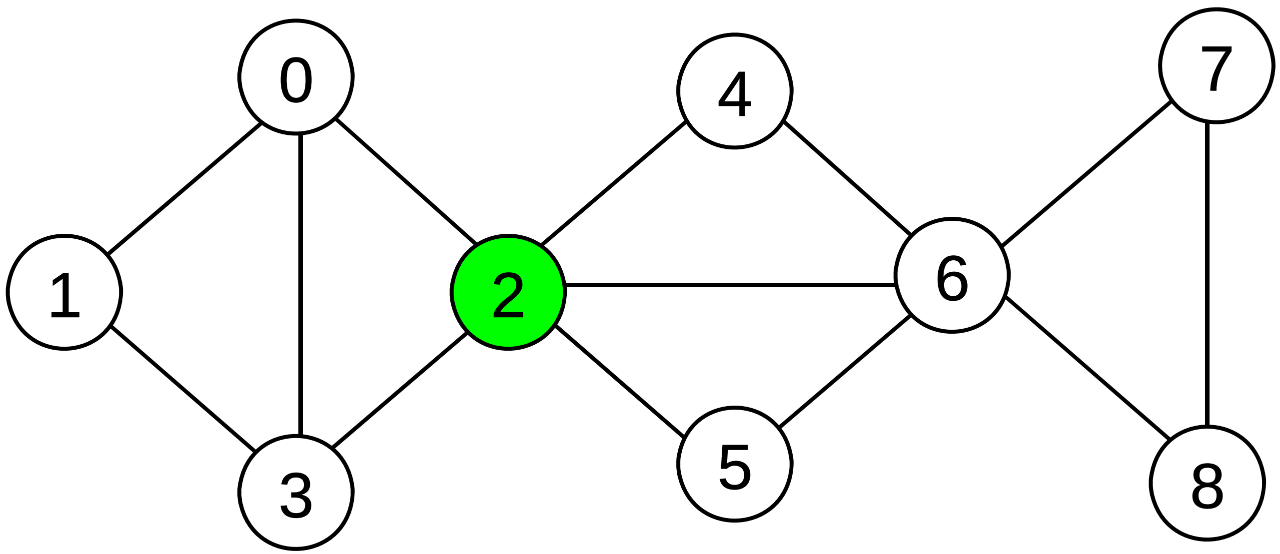
4.5 Biconnected Components

- Articulation point
  - A vertex v of G such that the deletion of v, together with all edges incident to v, produces a graph G’ that has at least two connected components



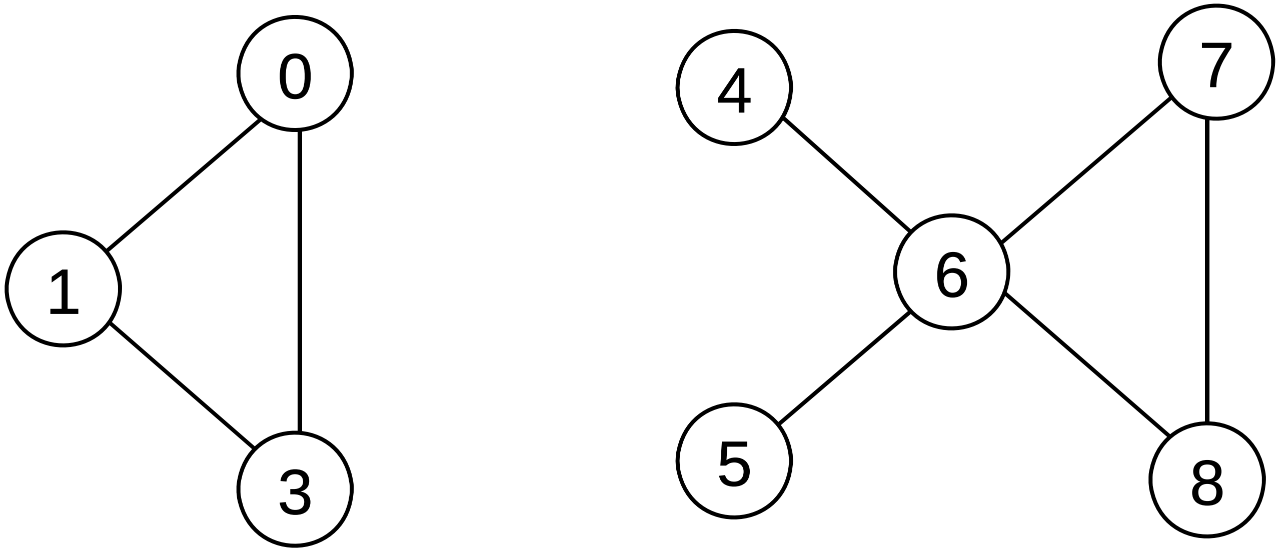
4.5 Biconnected Components

- Articulation point
  - A vertex v of G such that the deletion of v, together with all edges incident to v, produces a graph G’ that has at least two connected components



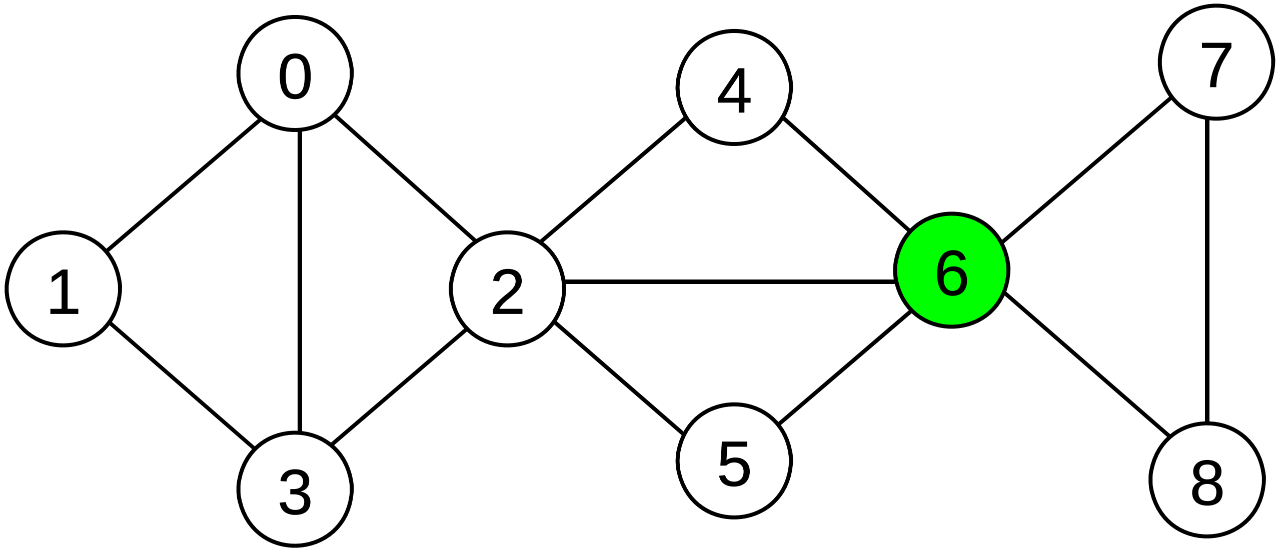
4.5 Biconnected Components

- Articulation point
  - A vertex v of G such that the deletion of v, together with all edges incident to v, produces a graph G’ that has at least two connected components



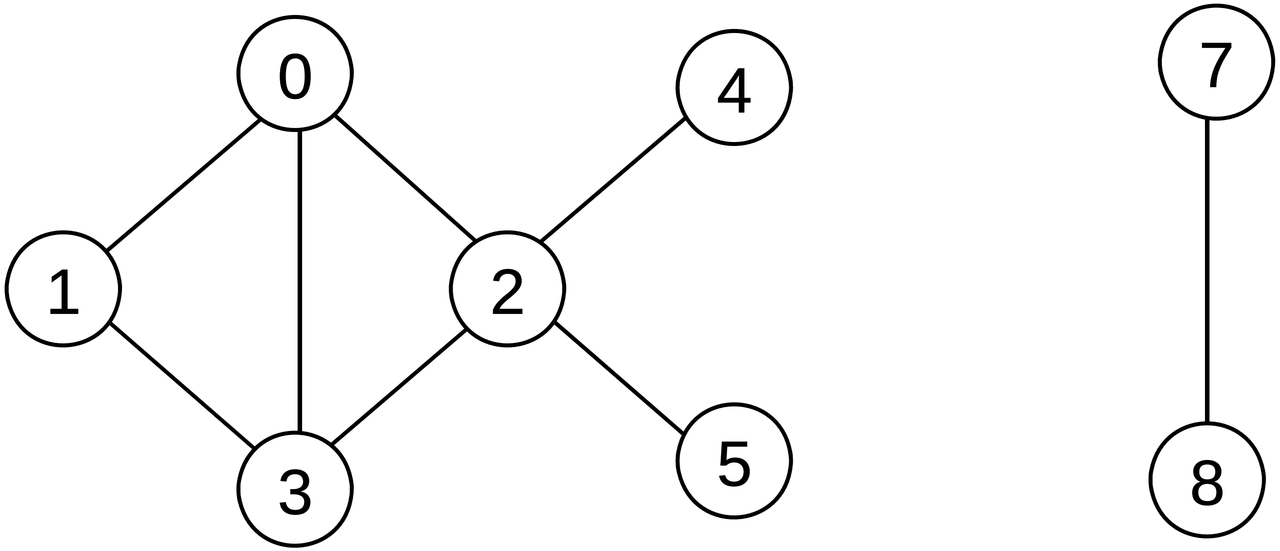
4.5 Biconnected Components

- Articulation point
  - A vertex v of G such that the deletion of v, together with all edges incident to v, produces a graph G' that has at least two connected components



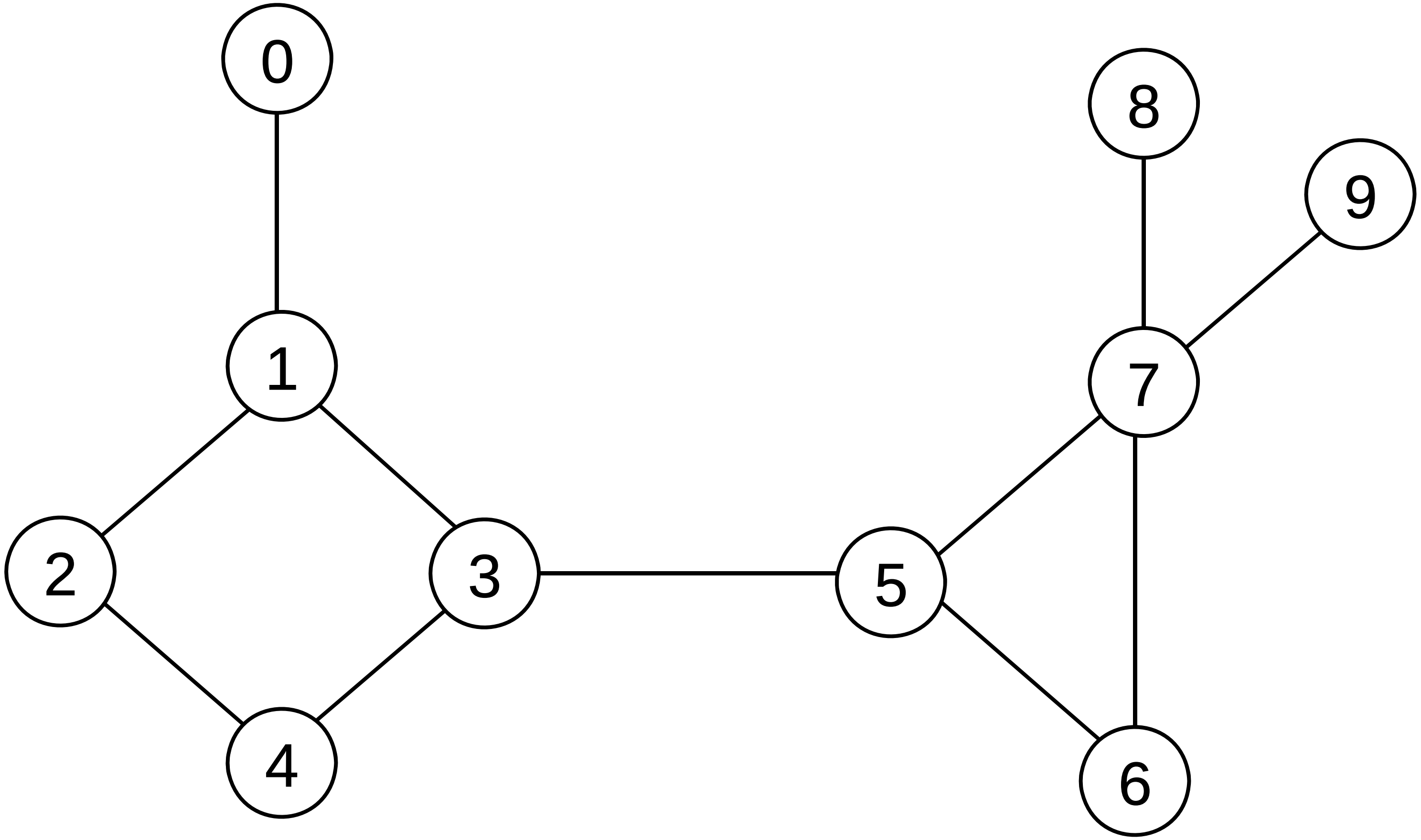
4.5 Biconnected Components

- Articulation point
  – A vertex v of G such that the deletion of v, together with all edges incident to v, produces a graph G’ that has at least two connected components



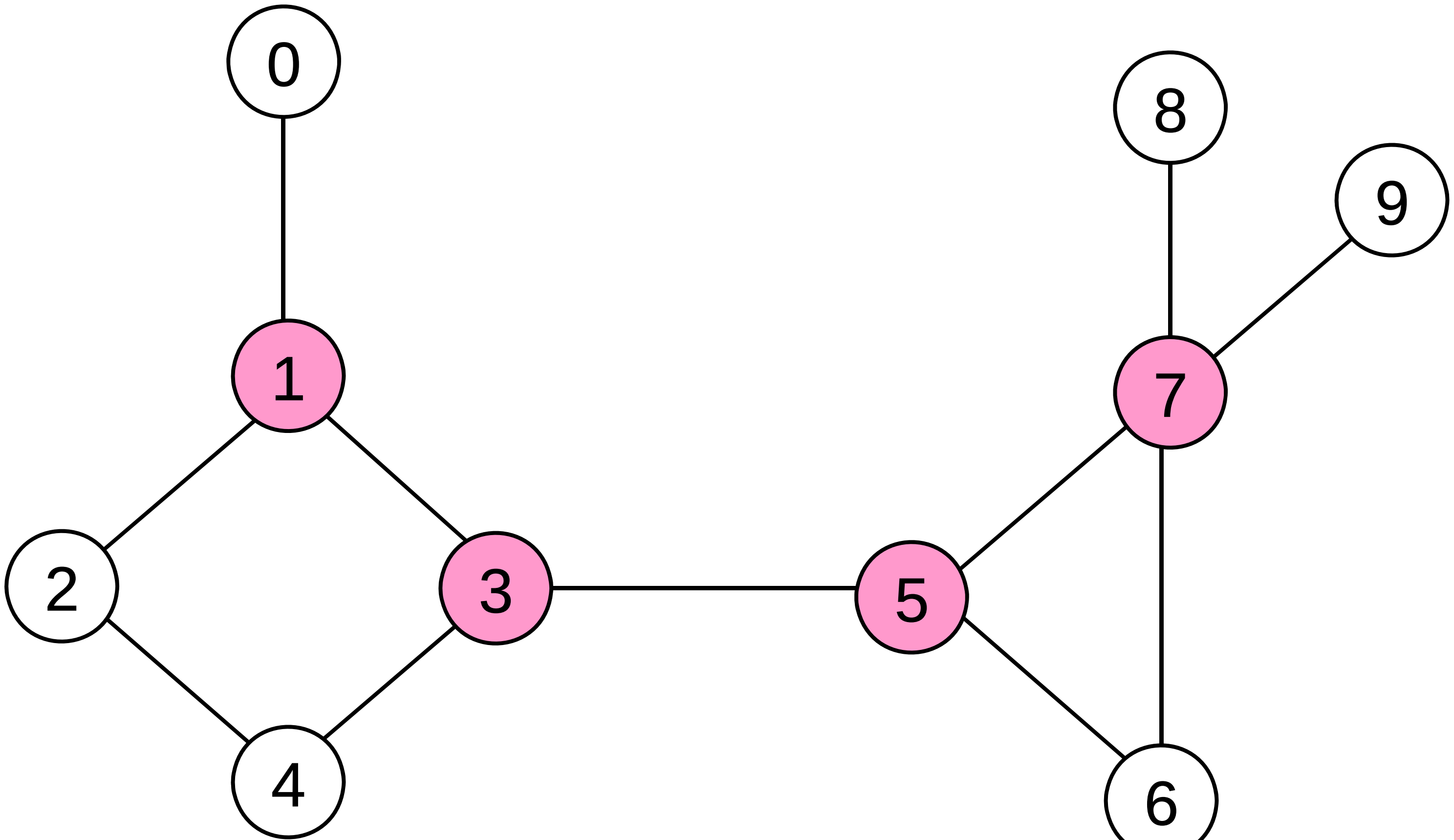
4.5 Biconnected Components

- Ex) What is the articulation point on this graph?



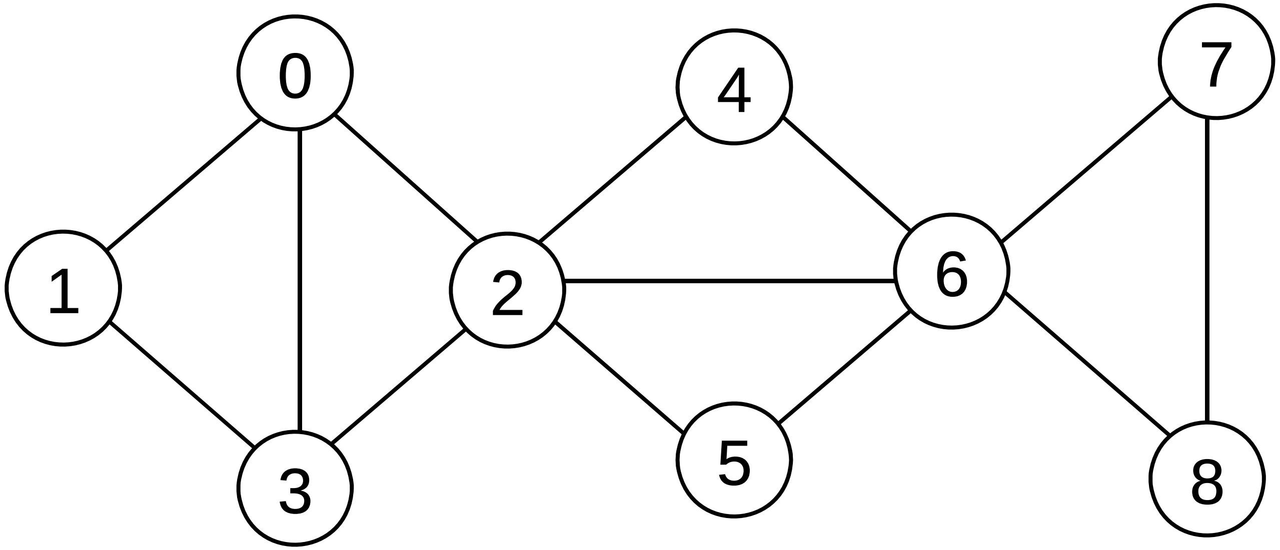
4.5 Biconnected Components

- Ex) What is the articulation point on this graph?



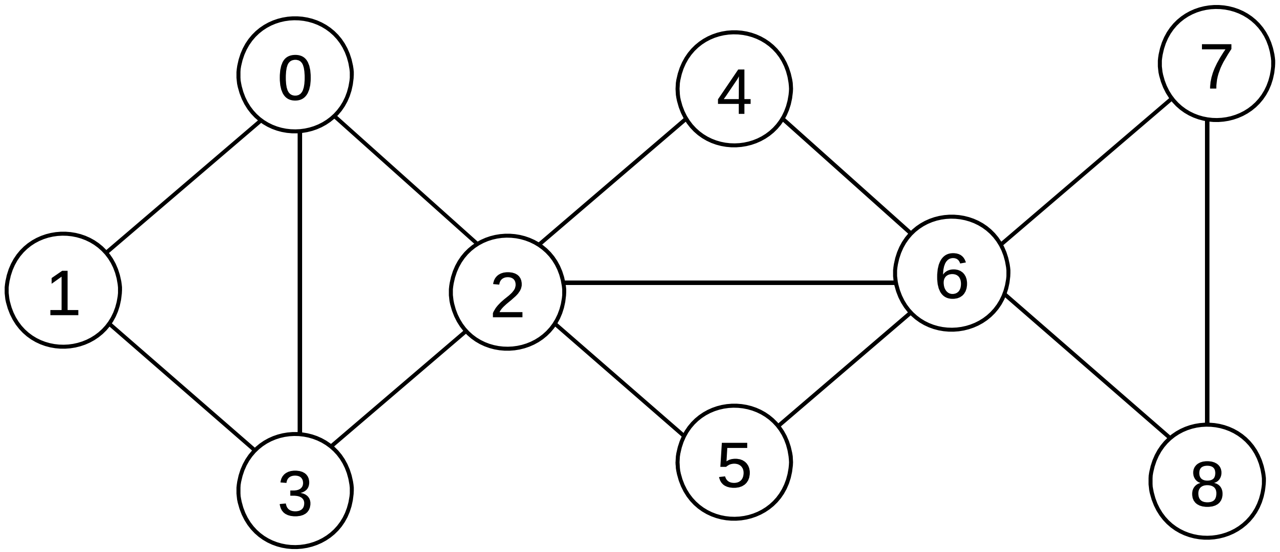
4.5 Biconnected Components

- Biconnected graph
  – A connected graph that has no articulation points



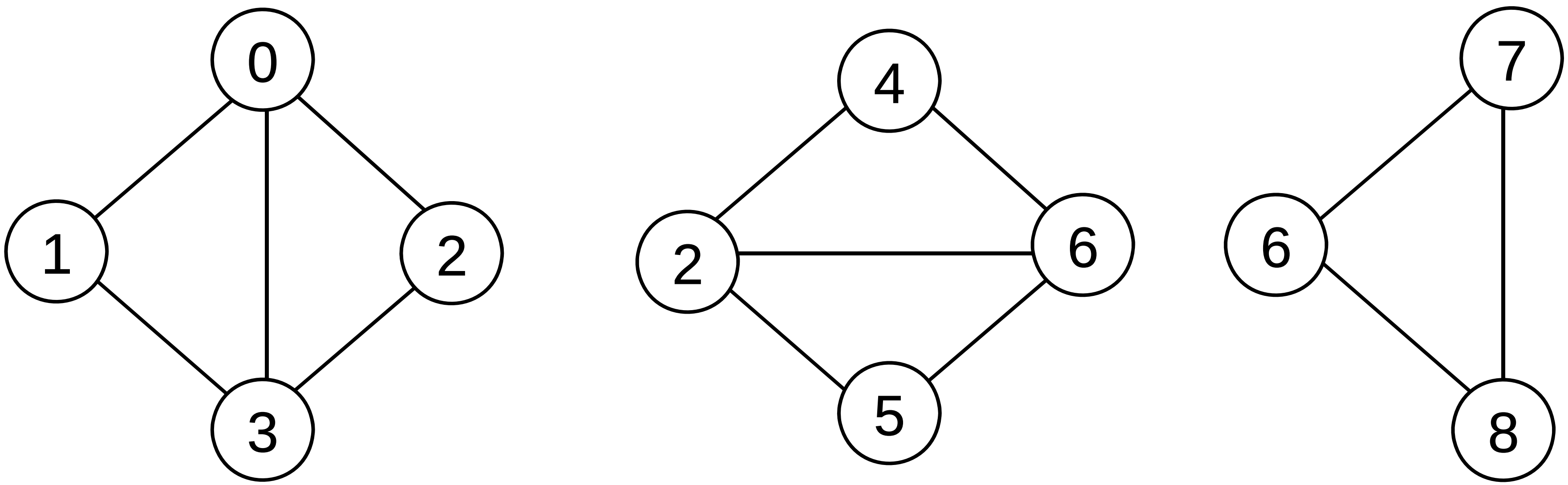
4.5 Biconnected Components

- Biconnected graph
  - A connected graph that has no articulation points
- Biconnected component
  - A maximal biconnected subgraph



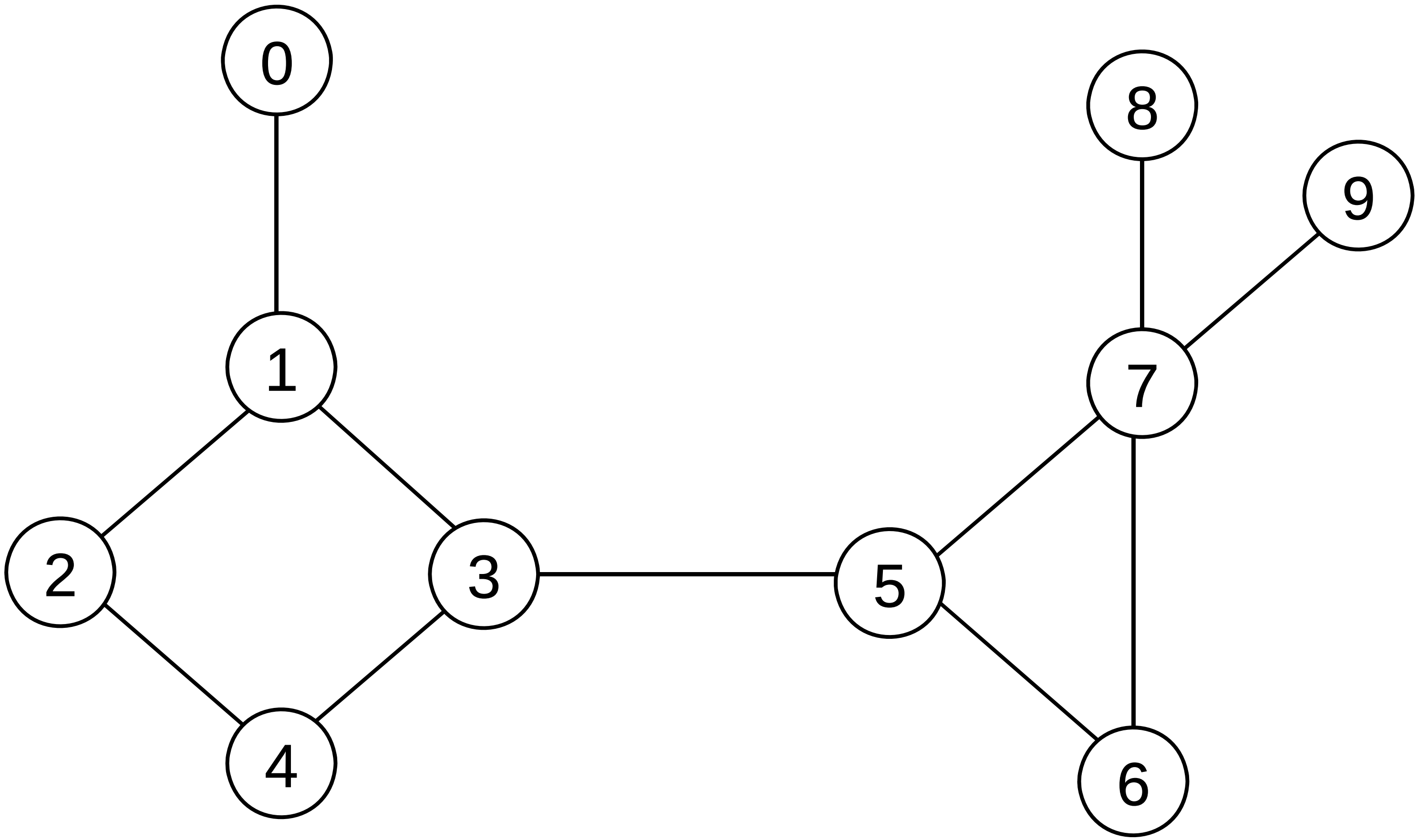
4.5 Biconnected Components

- Biconnected graph
  - A connected graph that has no articulation points
- Biconnected component
  - A maximal biconnected subgraph



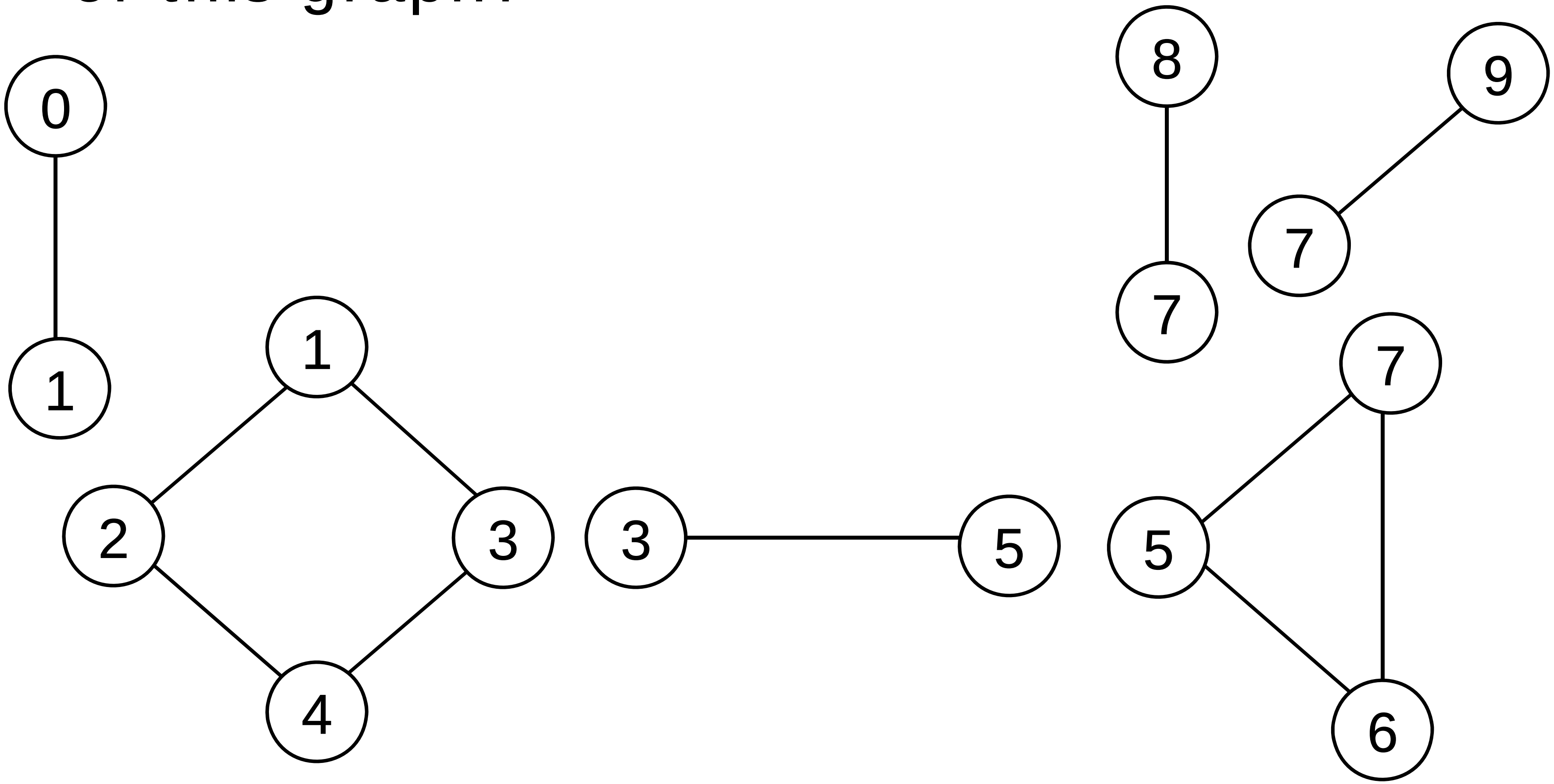
4.5 Biconnected Components

- Ex) What are the biconnected components of this graph?



4.5 Biconnected Components

- Ex) What are the biconnected components of this graph?

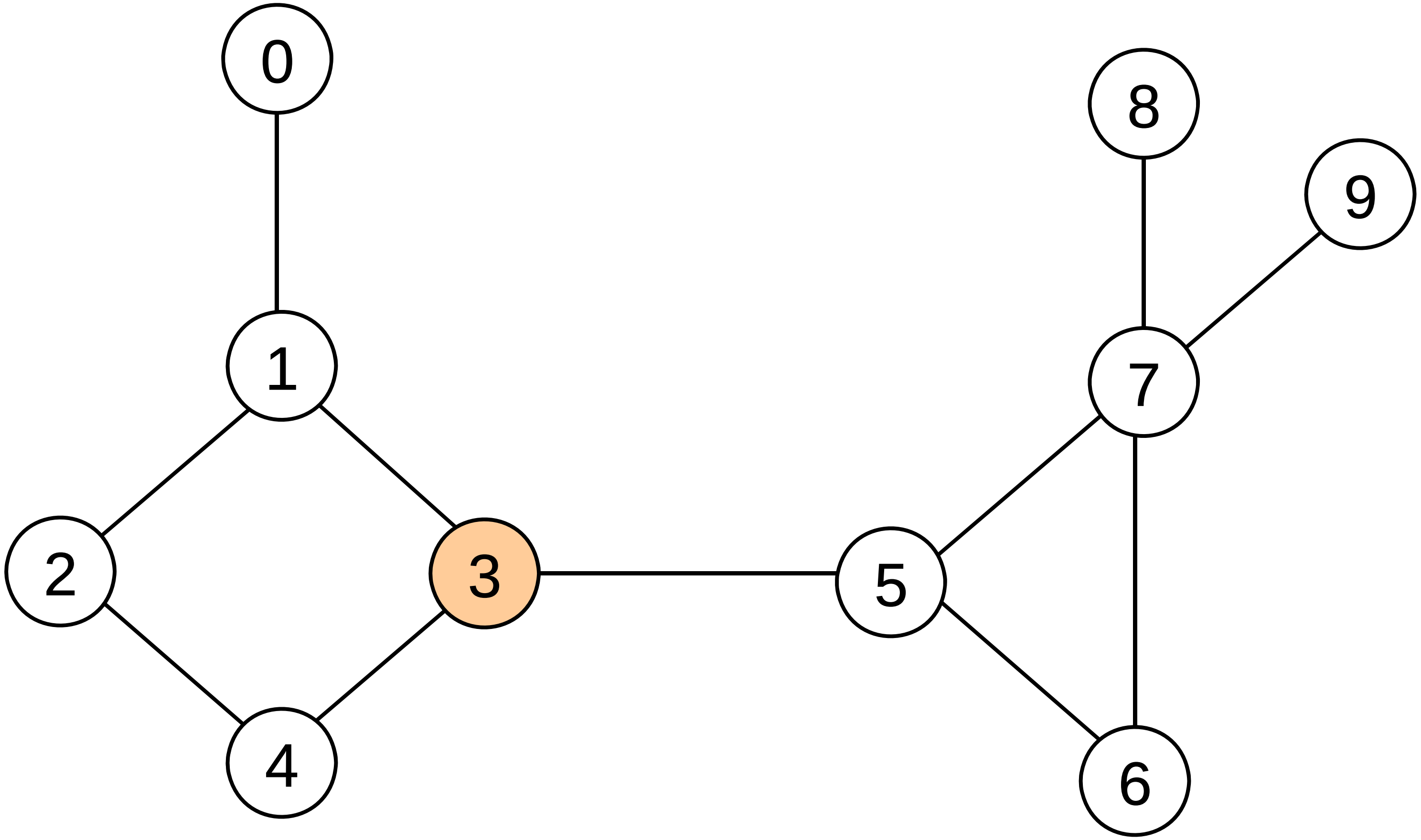


4.5 Biconnected Components

- How to find biconnected components of a graph
  - Use dfs ( )
  - dfn (depth-first number) of a vertex
    - The sequence in which the vertices are visited during depth-first search

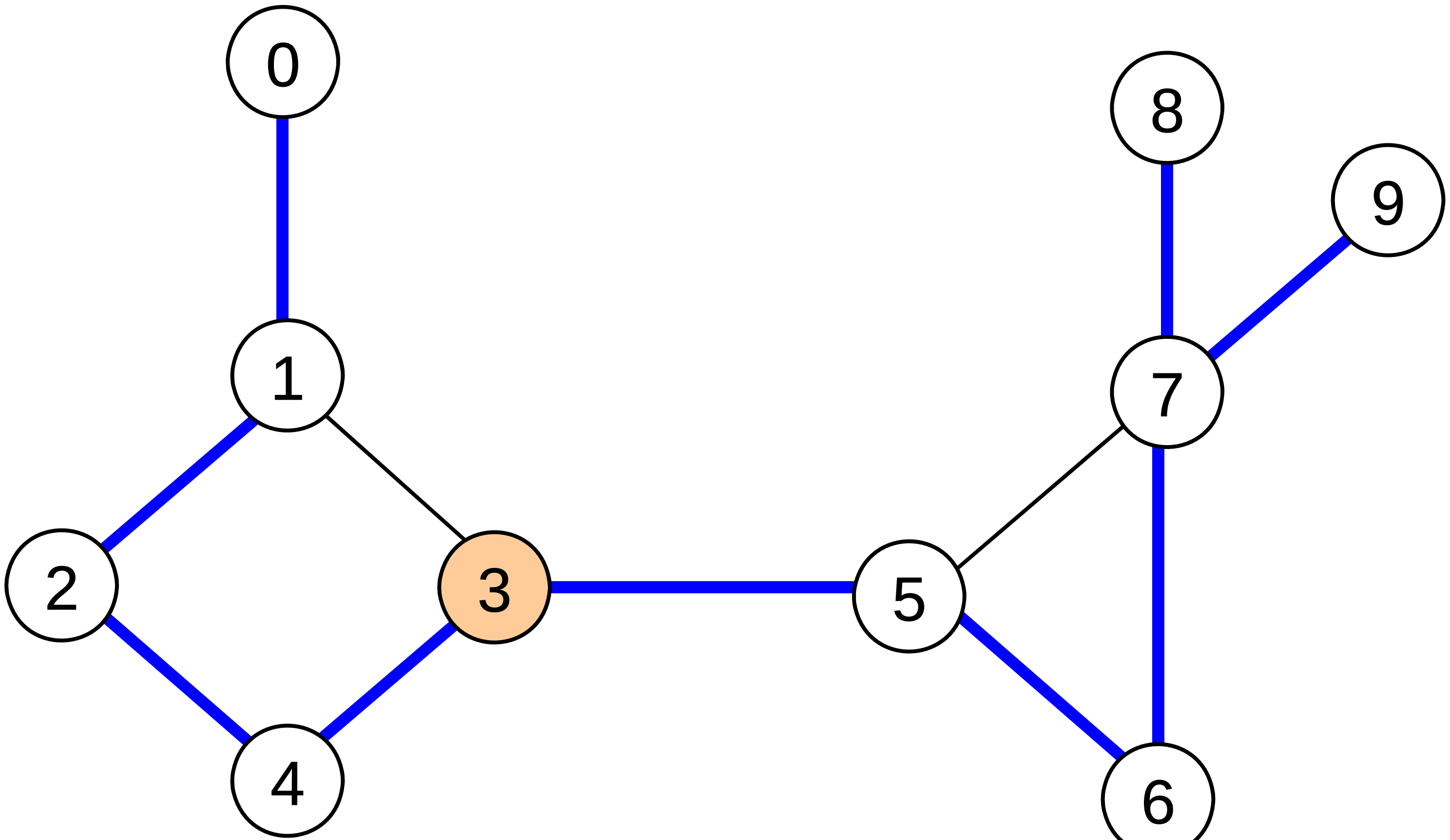
4.5 Biconnected Components

- Ex) What is dfn, if 3 is a root of depth-first spanning tree?



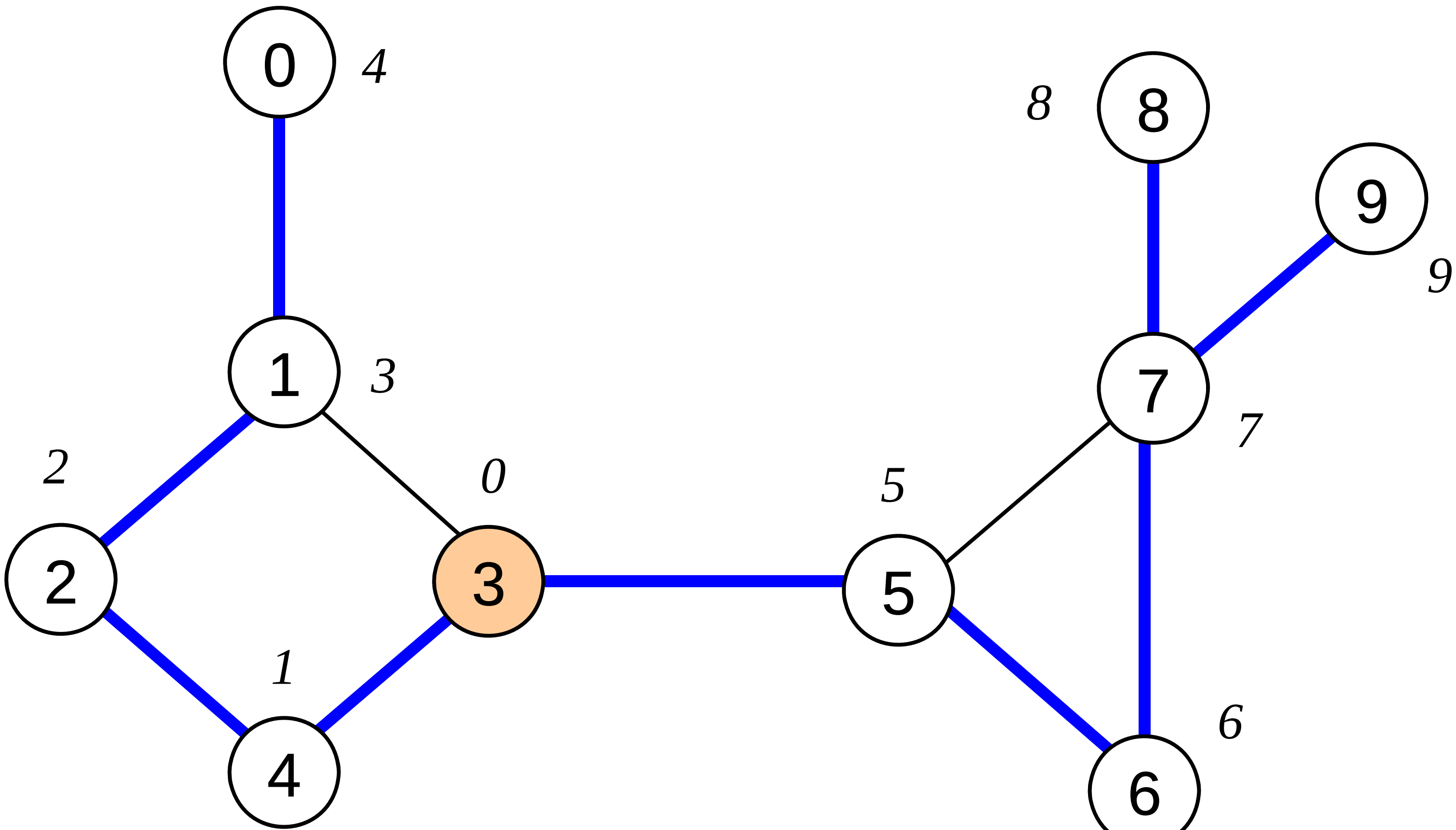
4.5 Biconnected Components

- Ex) What is dfn, if 3 is a root of depth-first spanning tree?



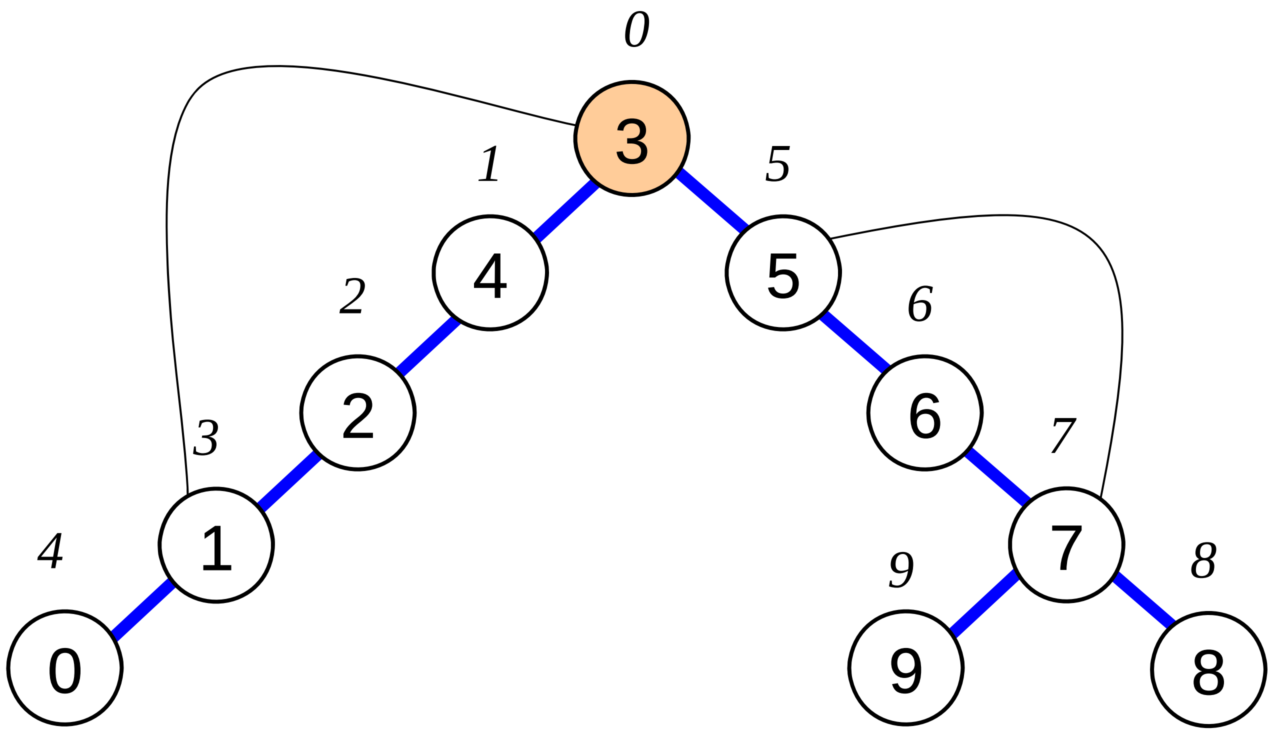
4.5 Biconnected Components

- Ex) What is dfn, if 3 is a root of depth-first spanning tree?



4.5 Biconnected Components

- Ex) What is dfn, if 3 is a root of depth-first spanning tree?



4.5 Biconnected Components

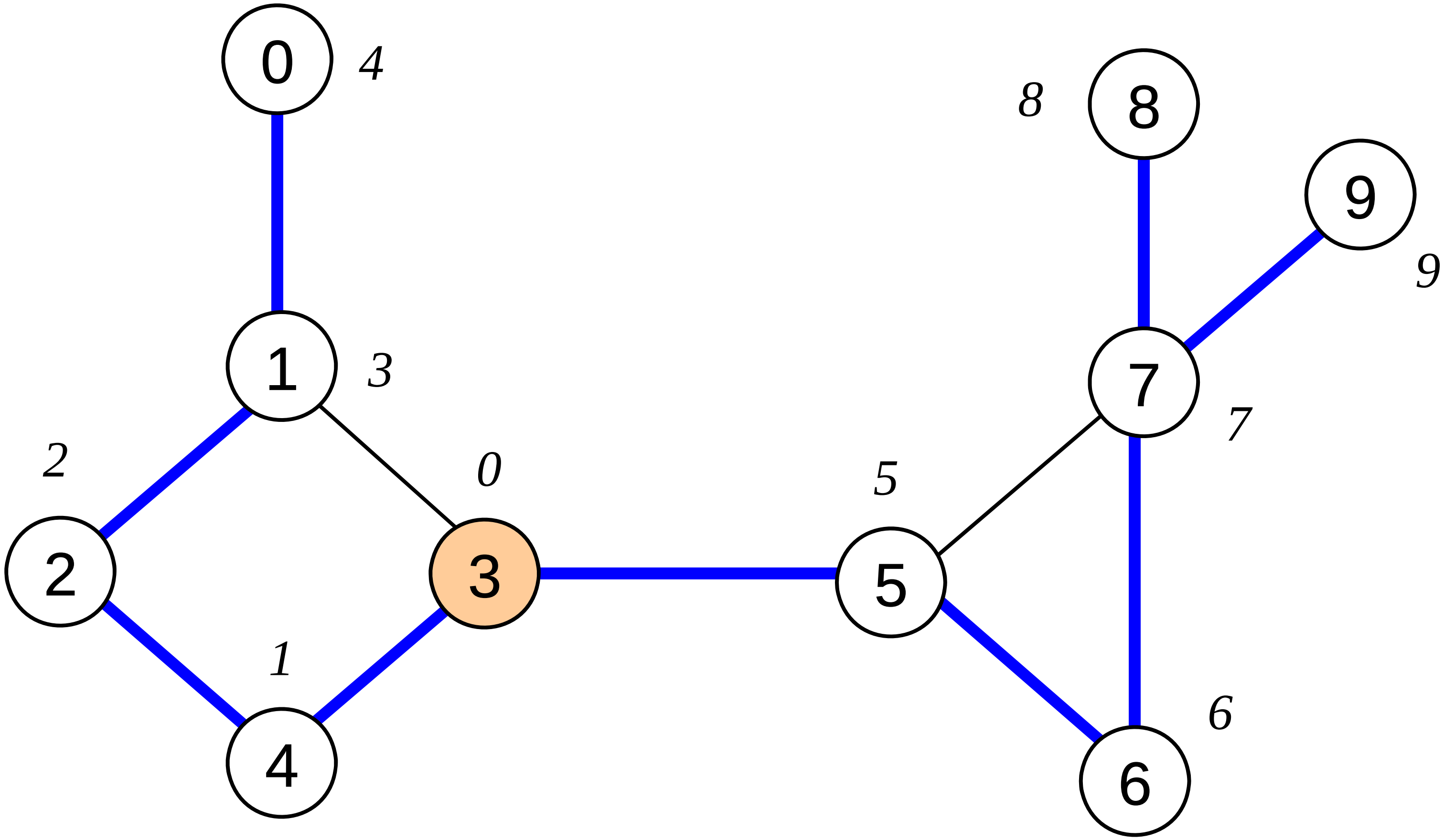
- Property
  - u is an ancestor of v in the depth-first spanning tree → $dfn(u) < dfn(v)$

4.5 Biconnected Components

- Back edge
  - Edges in G = edges in the spanning tree + nontree edges
  - Back edge:
    - A nontree edge (u, v), if u is an ancestor of v or vice versa
  - In depth-first spanning tree, all the nontree edges are back edges

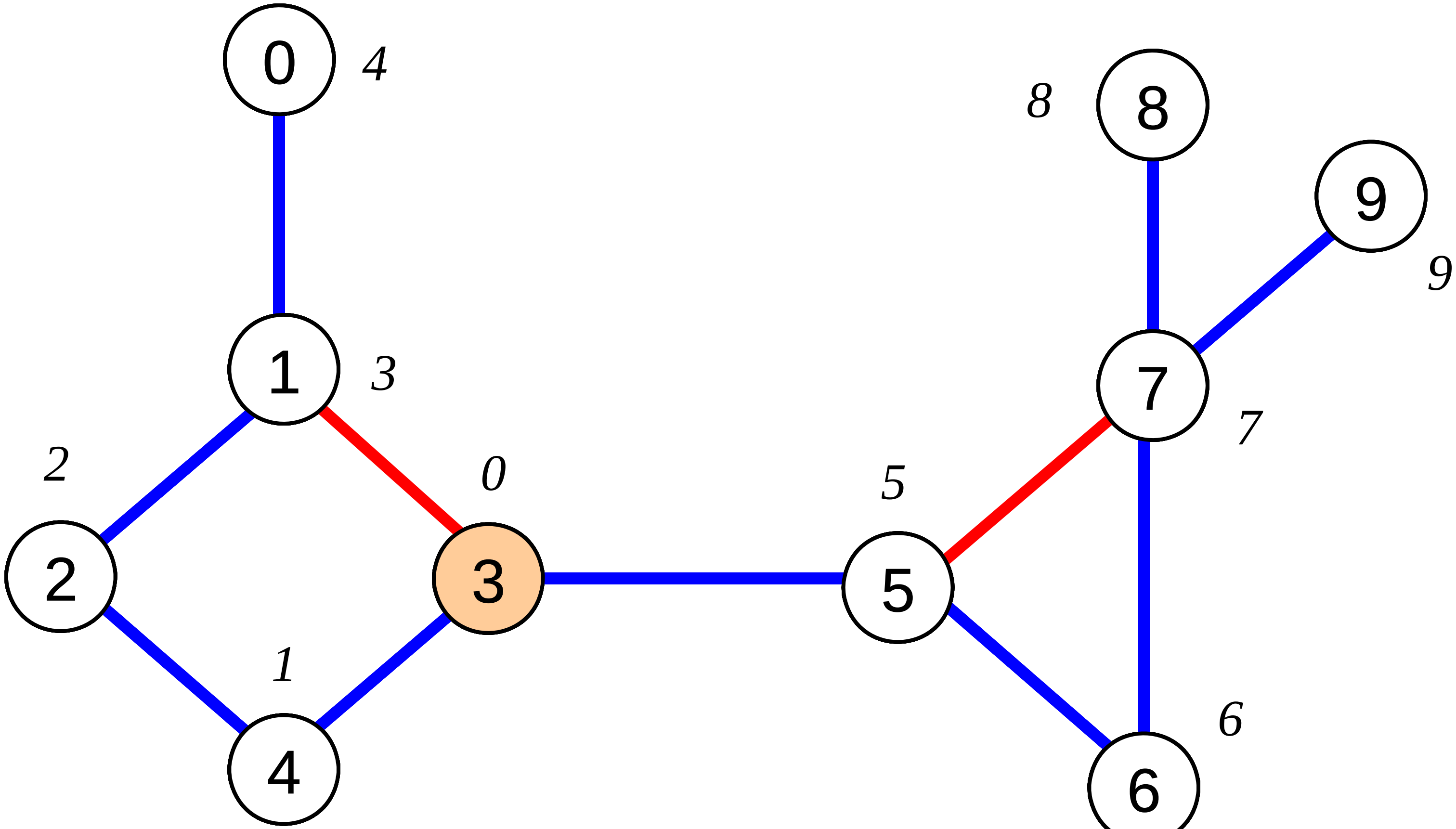
4.5 Biconnected Components

- Ex) What are the back edges of this graph?



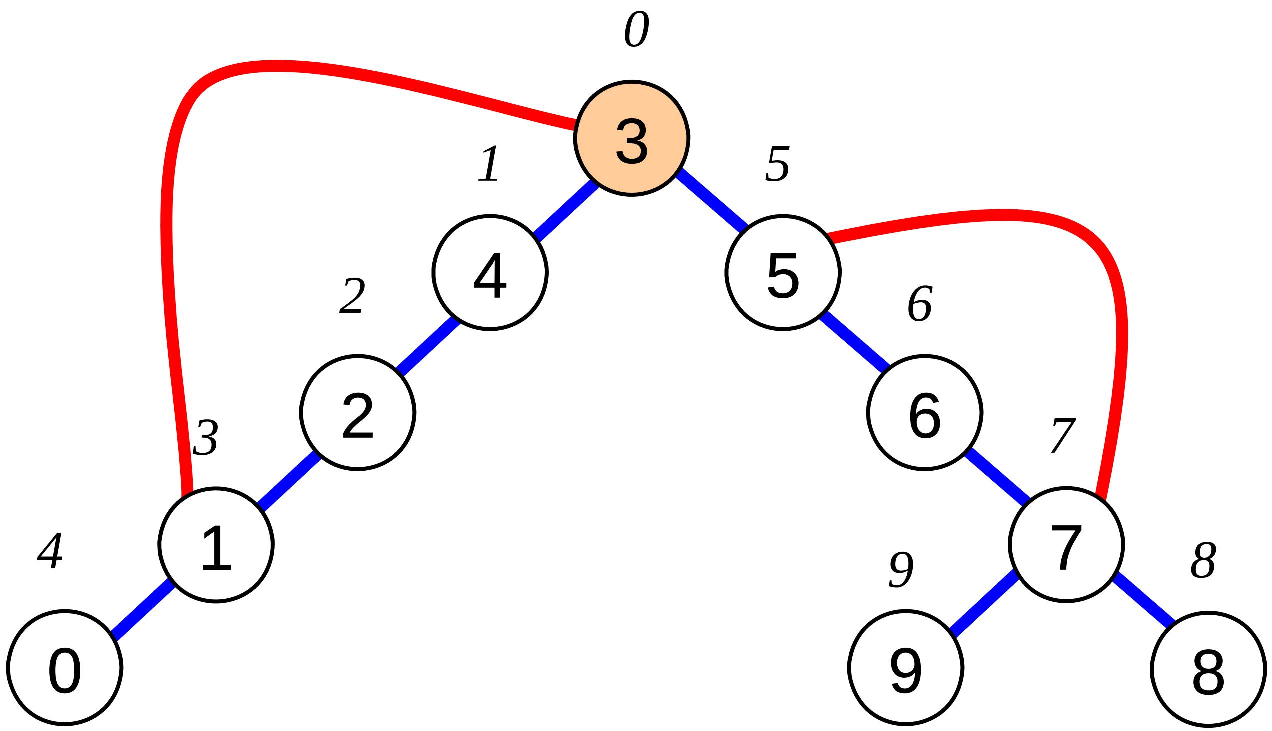
4.5 Biconnected Components

- Ex) What are the back edges of this graph?



4.5 Biconnected Components

- Ex) What are the back edges of this graph?

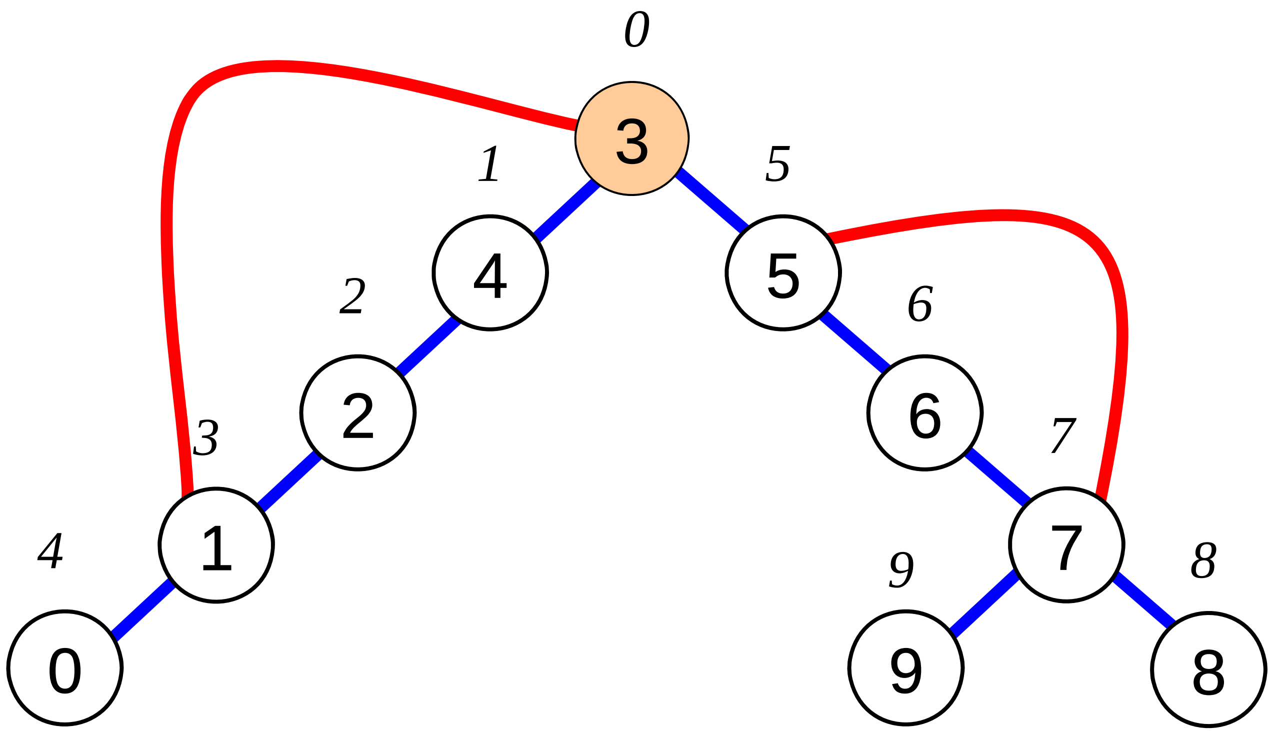


4.5 Biconnected Components

- Articulation points (1)
  - A root of a depth-first spanning tree is an articulation point, if it has at least two childs

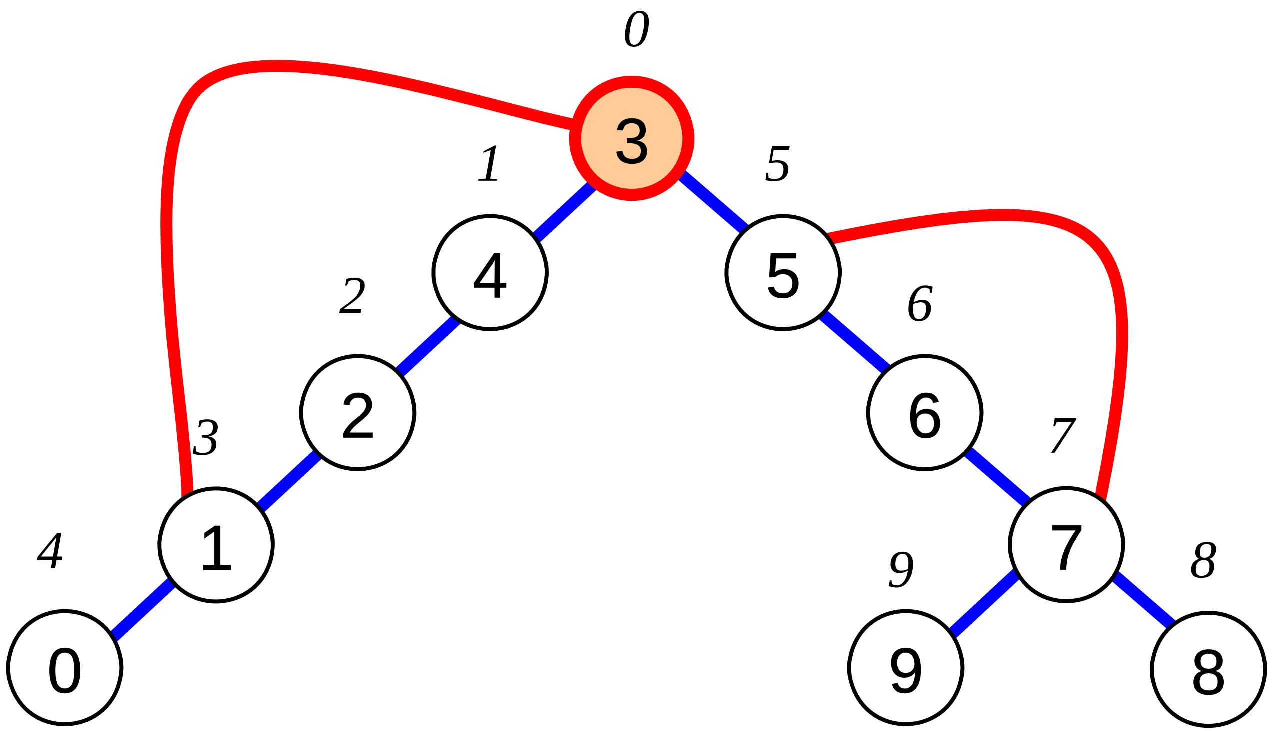
4.5 Biconnected Components

- Articulation points
  - Ex) What are the articulation points?



4.5 Biconnected Components

- Articulation points
  - Ex)

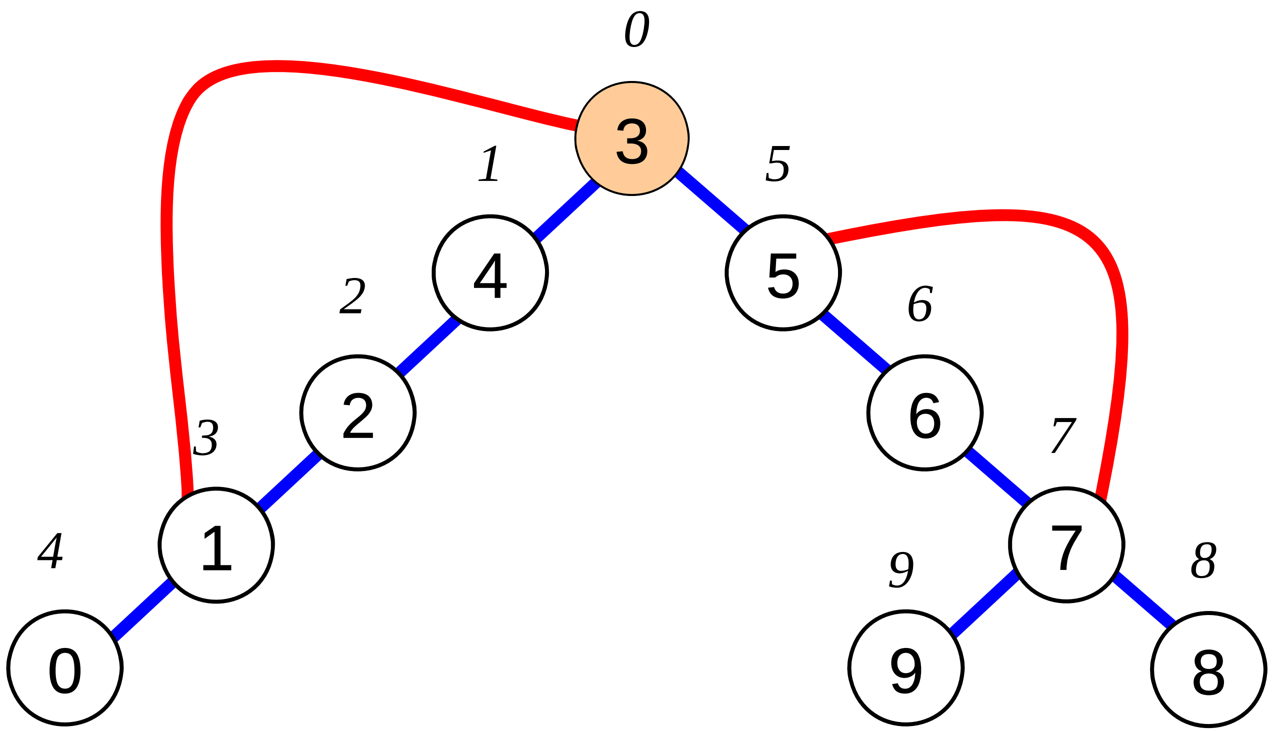


4.5 Biconnected Components

- Articulation points (2)
  - A vertex u, if it has at least one child w such that a path (w, descendants of w, and a single back edge, ancestor of u) does not exist

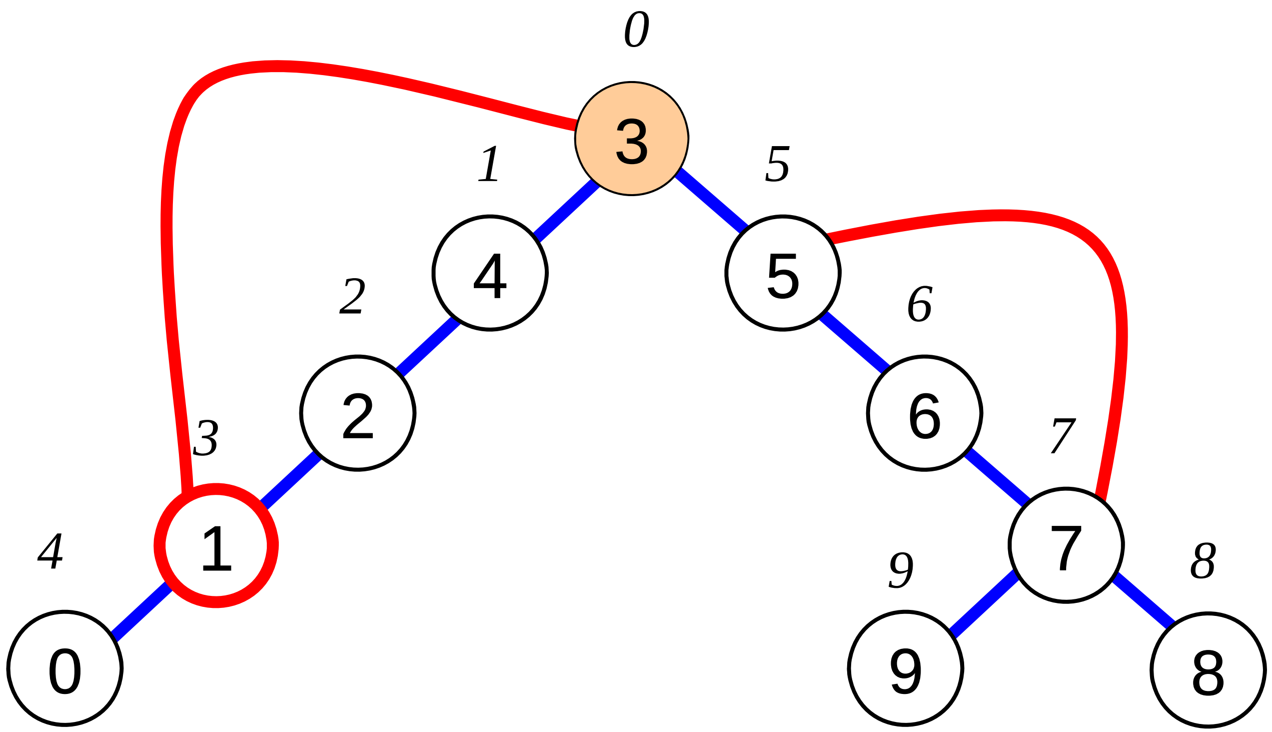
4.5 Biconnected Components

- Articulation points
  - Ex) What are the articulation points?



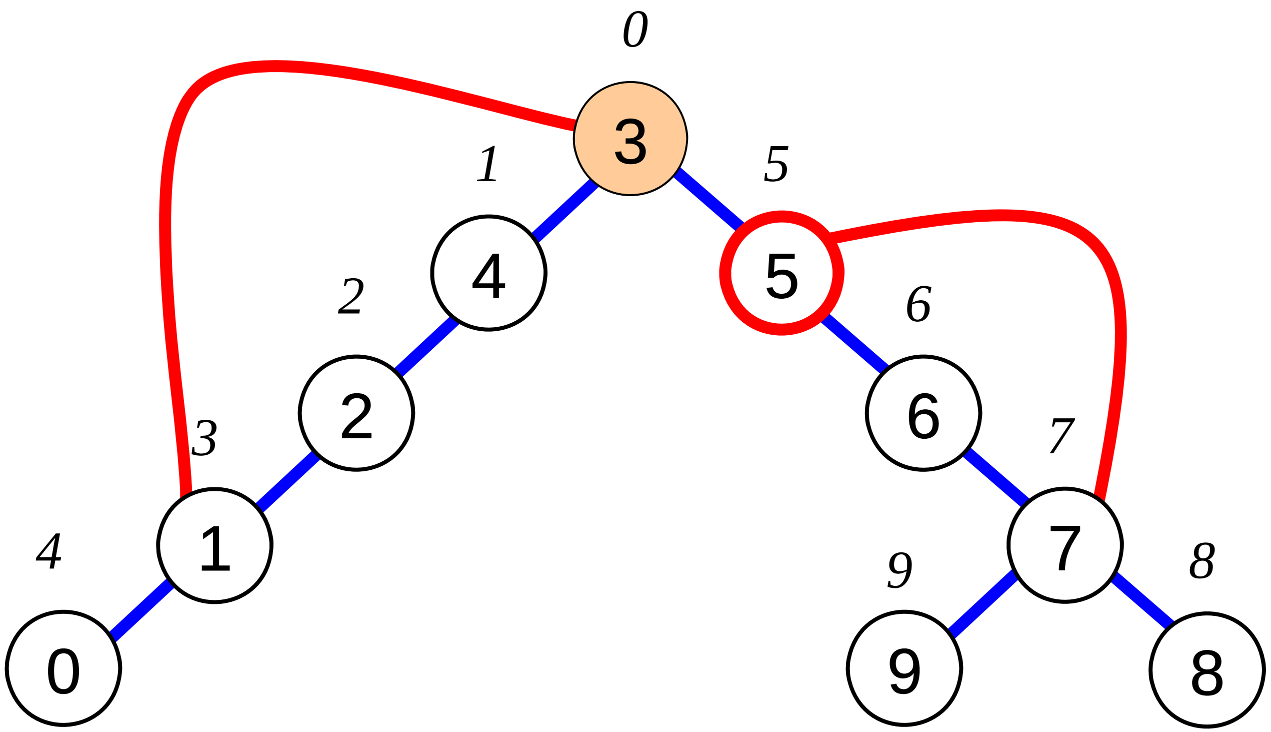
4.5 Biconnected Components

- Articulation points
  - Ex) What are the articulation points?



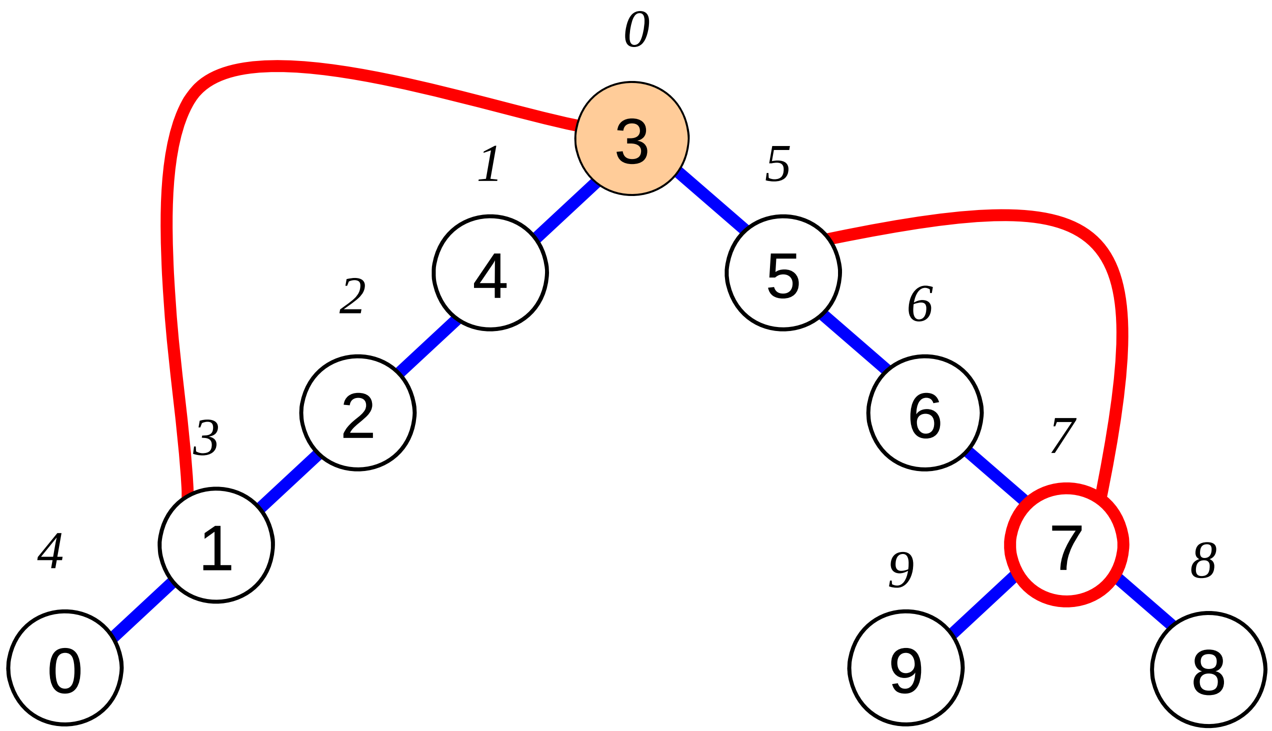
4.5 Biconnected Components

- Articulation points
  - Ex) What are the articulation points?



4.5 Biconnected Components

- Articulation points
  - Ex) What are the articulation points?



4.5 Biconnected Components

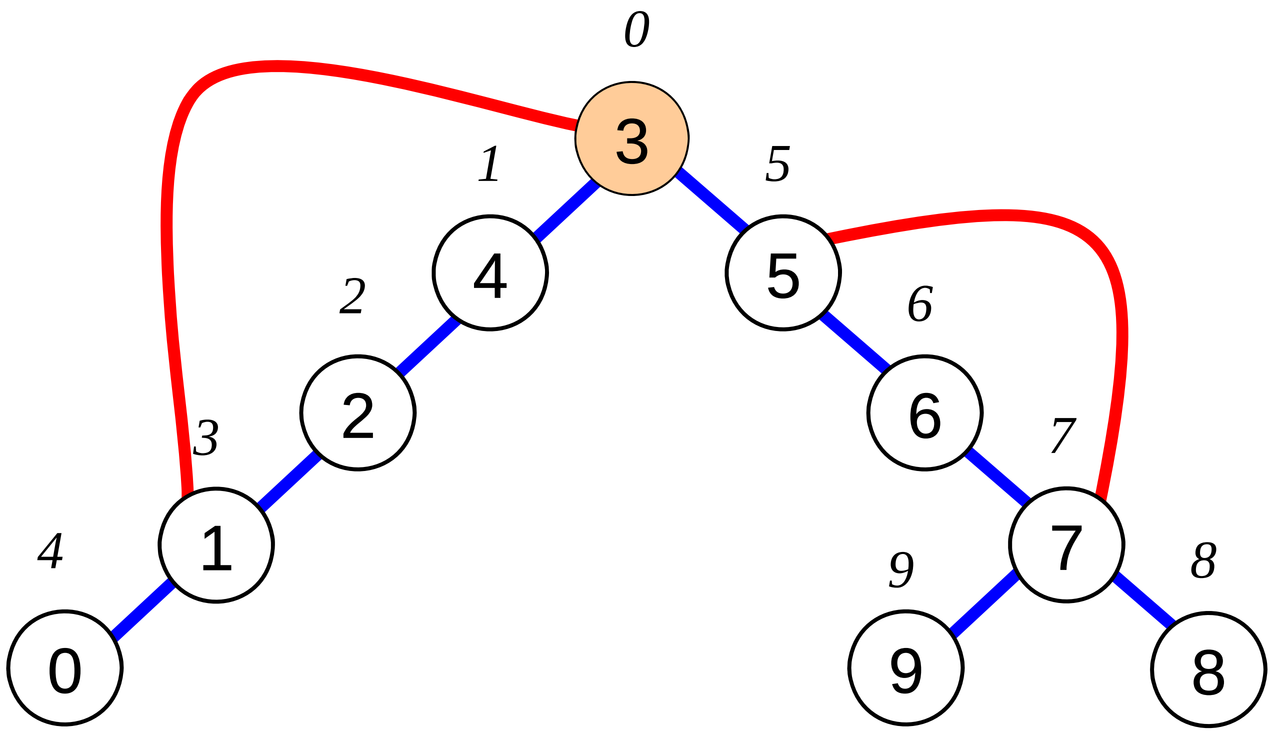
- How to find articulation points?
  - Define a new value low for each vertex u, such as low(u)
  - low(u)
    - The lowest depth first number that we can reach from u using a path of descendants followed by at most one back edge

$$
\begin{aligned}
low(u) = \min\{ & dfn(u), \\
& \min\{low(w) \mid w \text{ is a child of } u\}, \\
& \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\ \}
\end{aligned}
$$

4.5 Biconnected Components

- low(u)
  - Ex) What are low(u)?

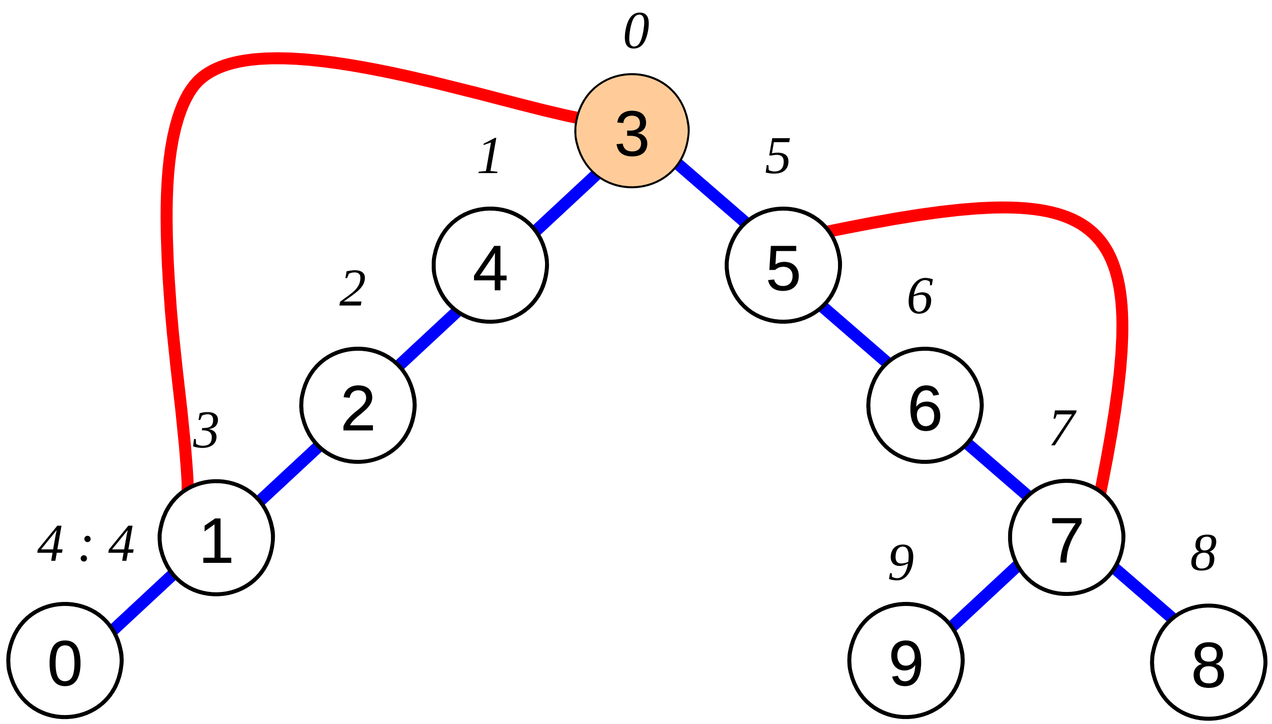
$$low(u) = \min\{dfn(u), \min\{low(w) \mid w \text{ is a child of } u\}, \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(0)?

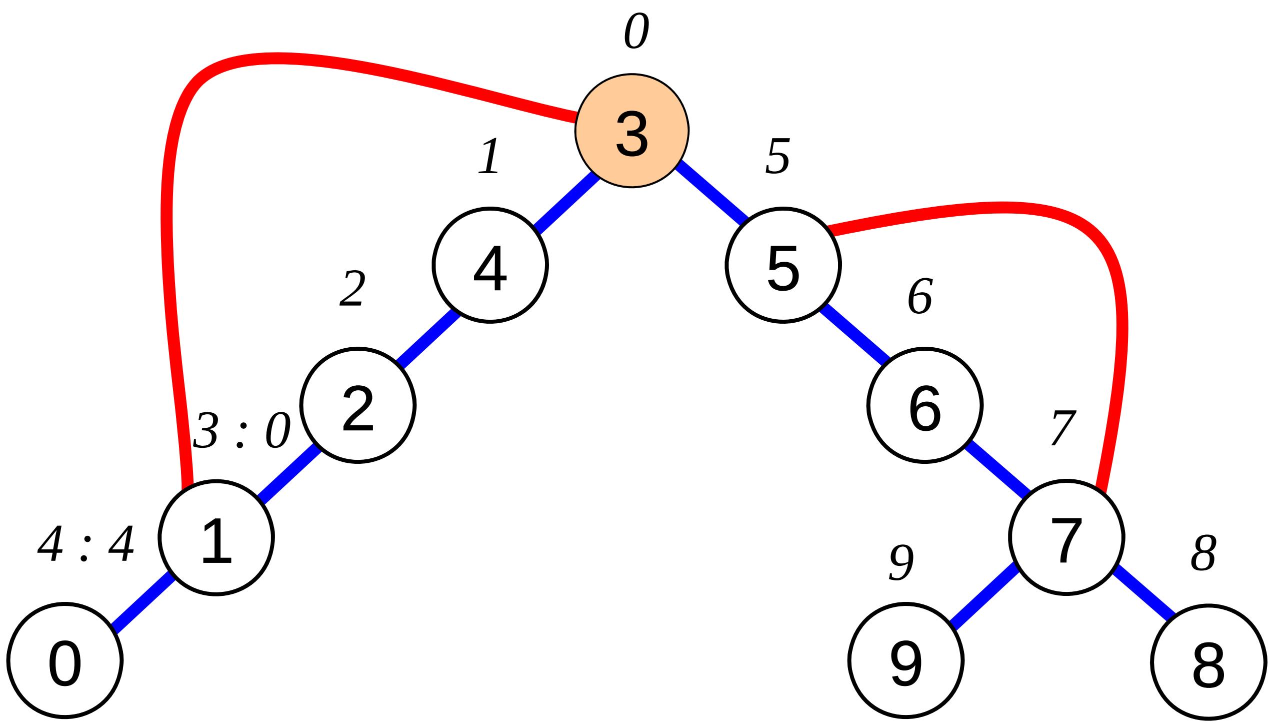
$$low(u) = \min\{dfn(u), \min\{low(w) \mid w \text{ is a child of } u\}, \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(1)?

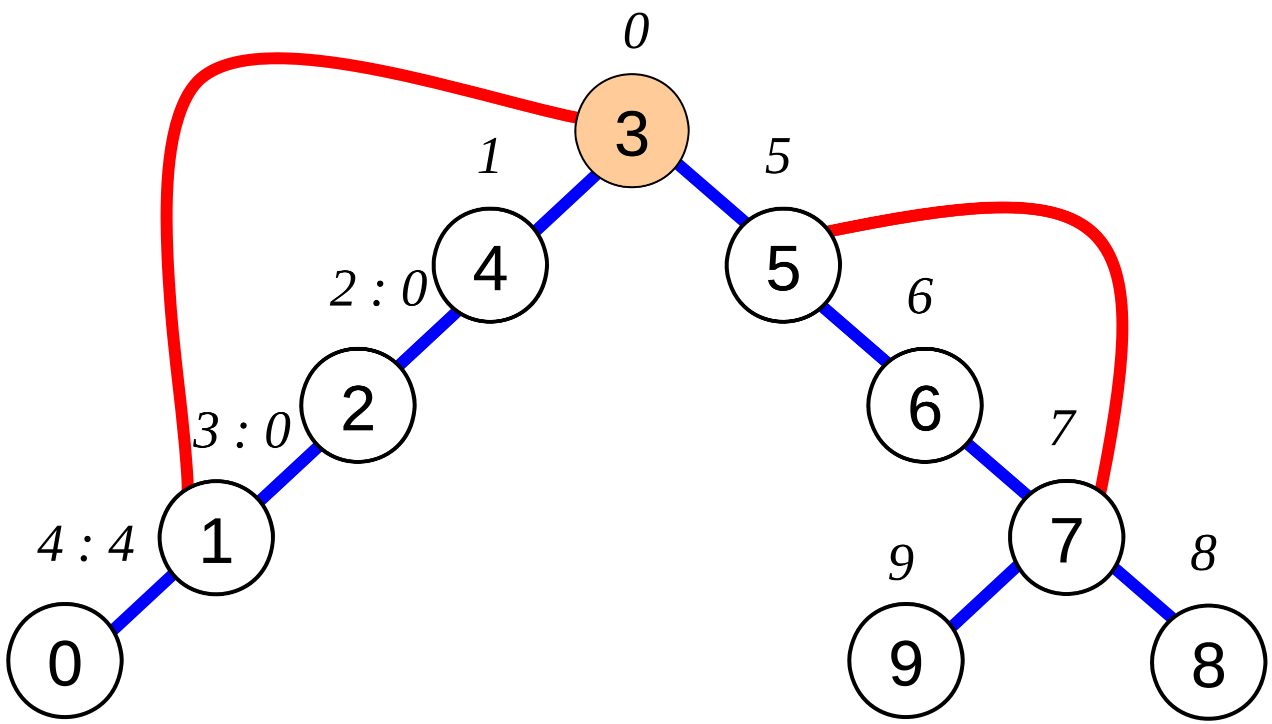
$$low(u) = \min\{dfn(u), \min\{low(w) \mid w \text{ is a child of } u\}, \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(2)?

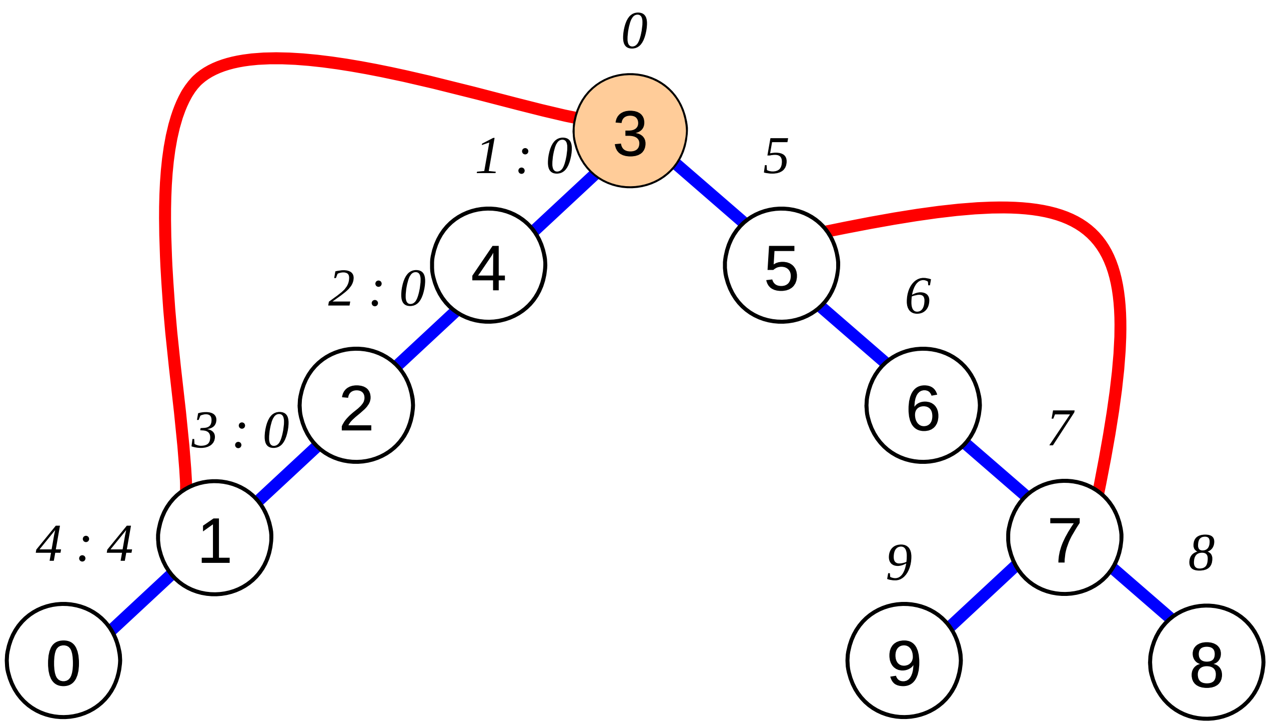
$$low(u) = \min\{dfn(u),$$
$$\min\{low(w) \mid w \text{ is a child of } u\},$$
$$\min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(4)?

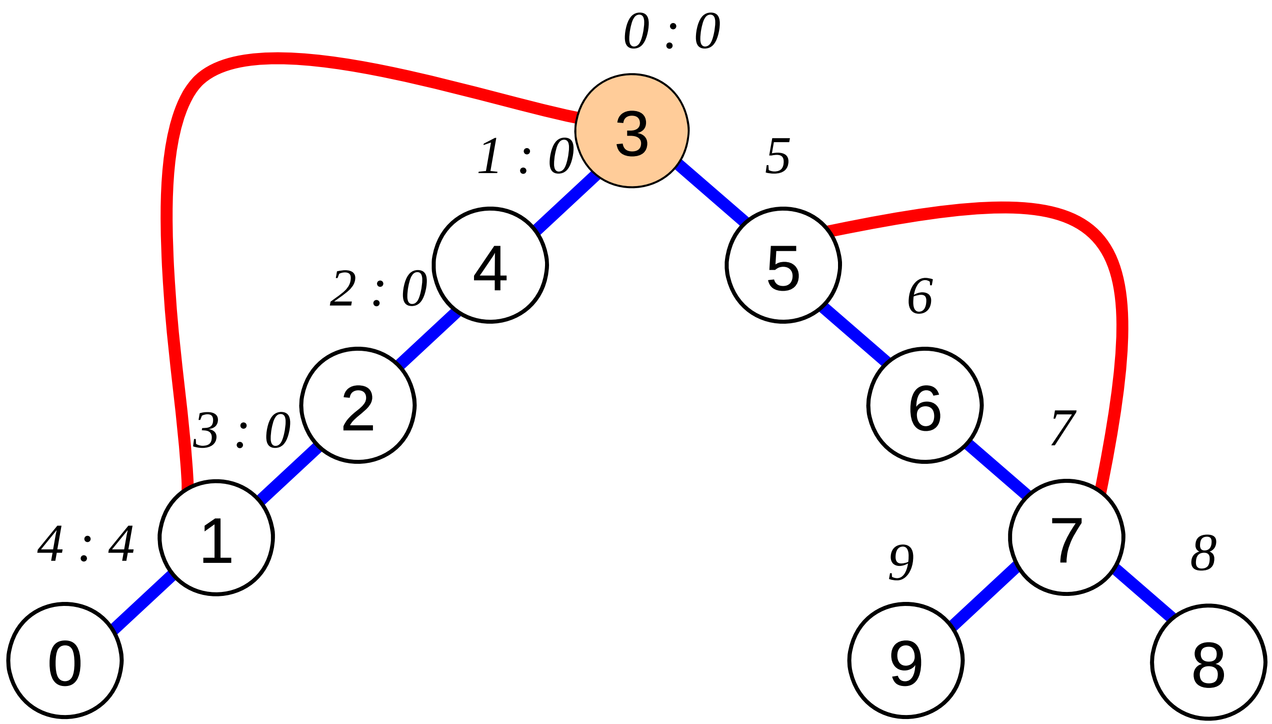
$$low(u) = \min\{dfn(u), \min\{low(w) \mid w \text{ is a child of } u\}, \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(3)?

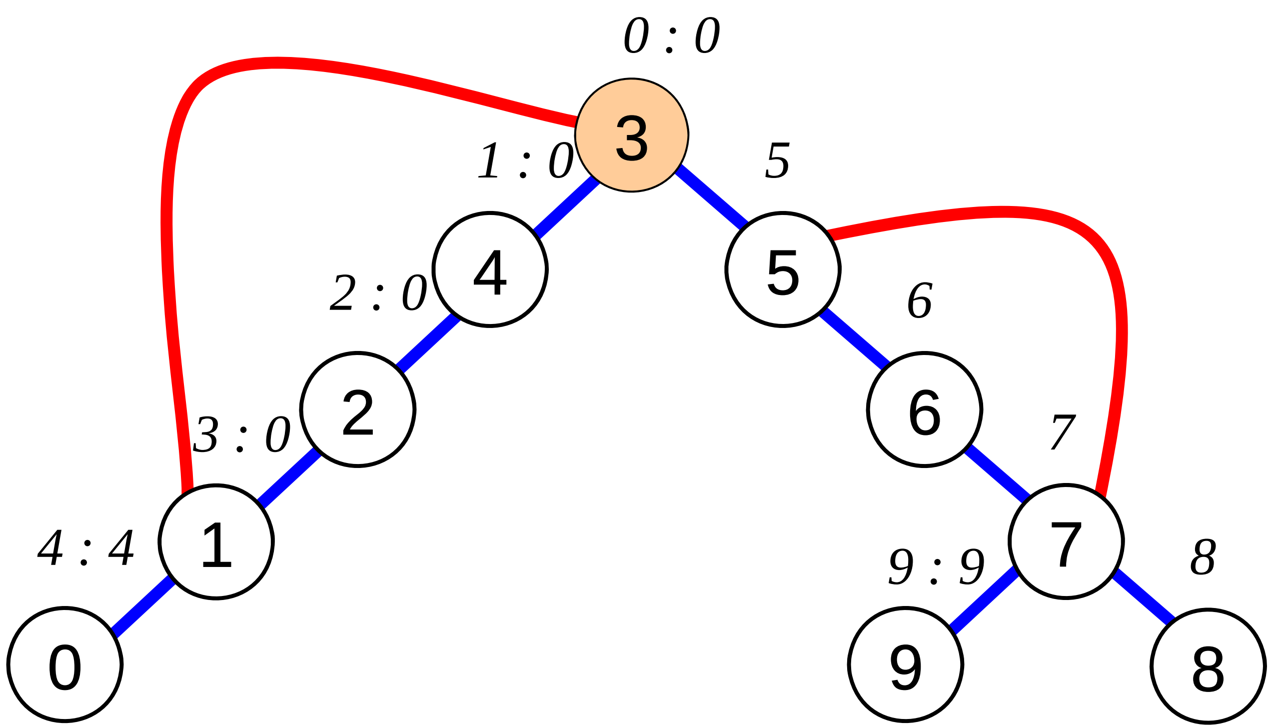
$$low(u) = \min\{dfn(u), \min\{low(w) \mid w \text{ is a child of } u\}, \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(9)?

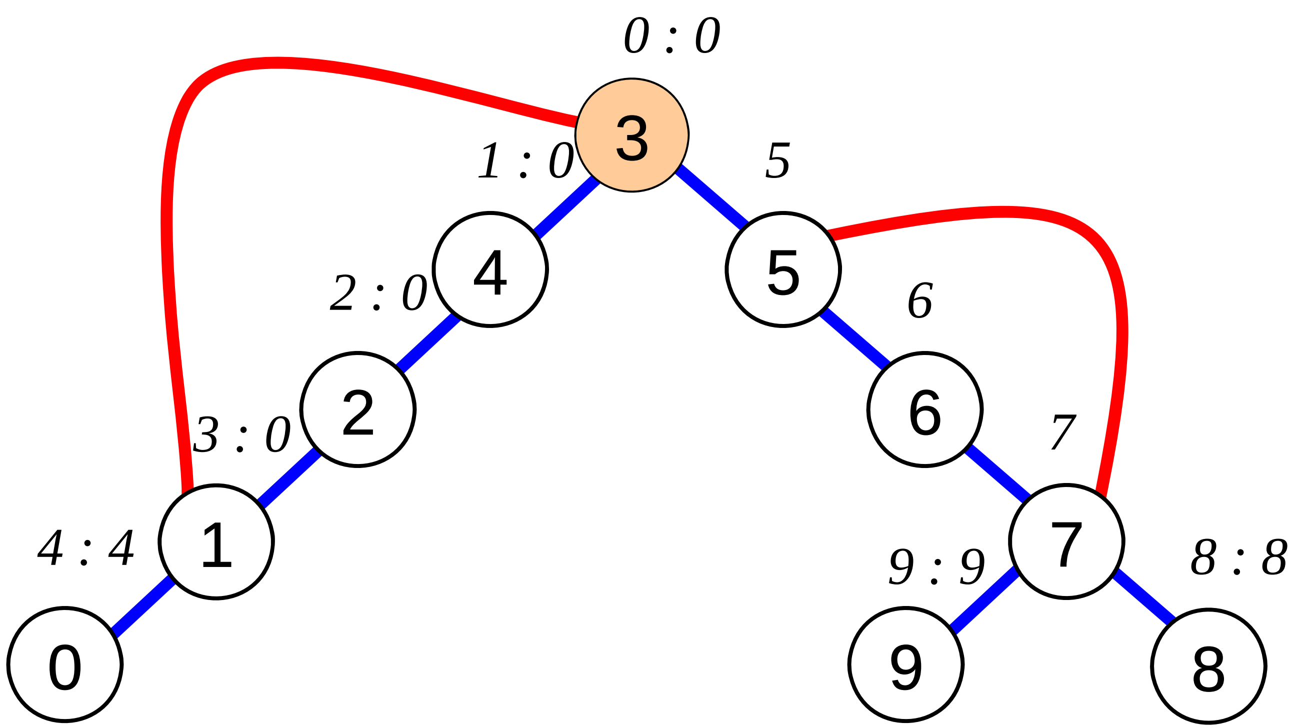
$$low(u) = \min\{dfn(u),\ \min\{low(w) \mid w \text{ is a child of } u\},\ \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(8)?

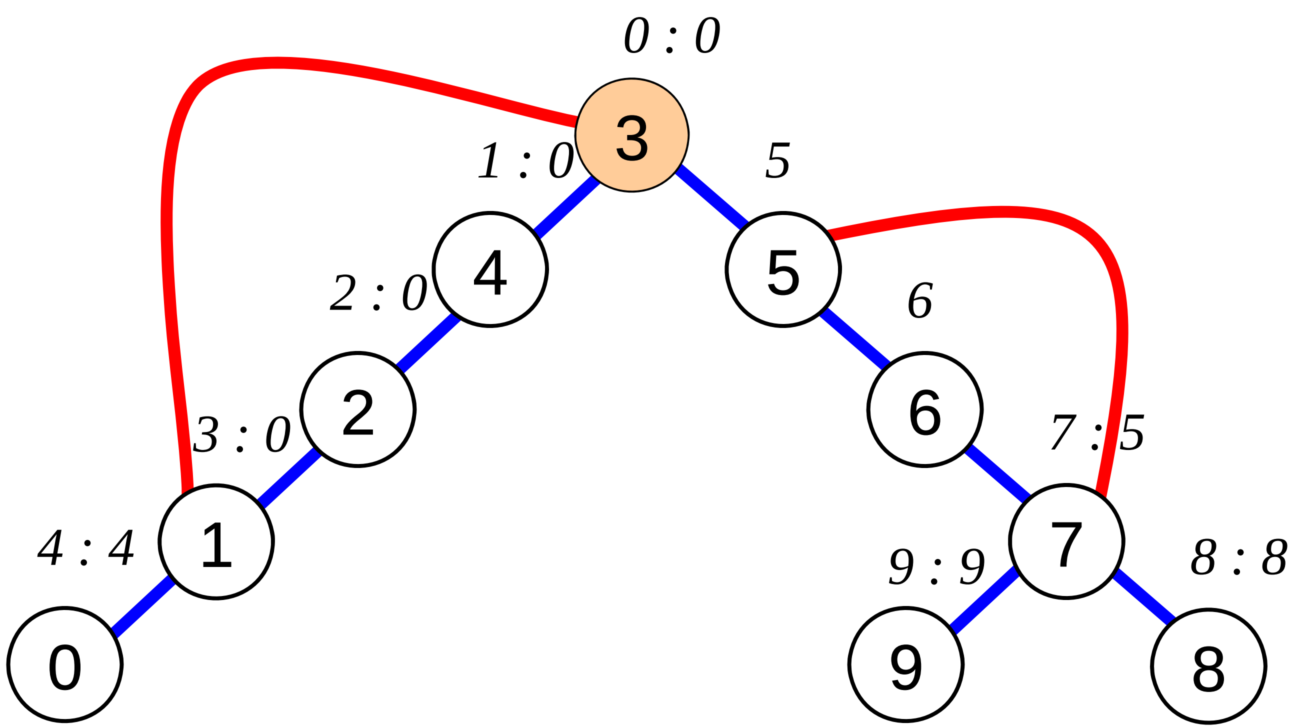
$$low(u) = \min\{dfn(u),\ \min\{low(w) \mid w \text{ is a child of } u\},\ \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(7)?

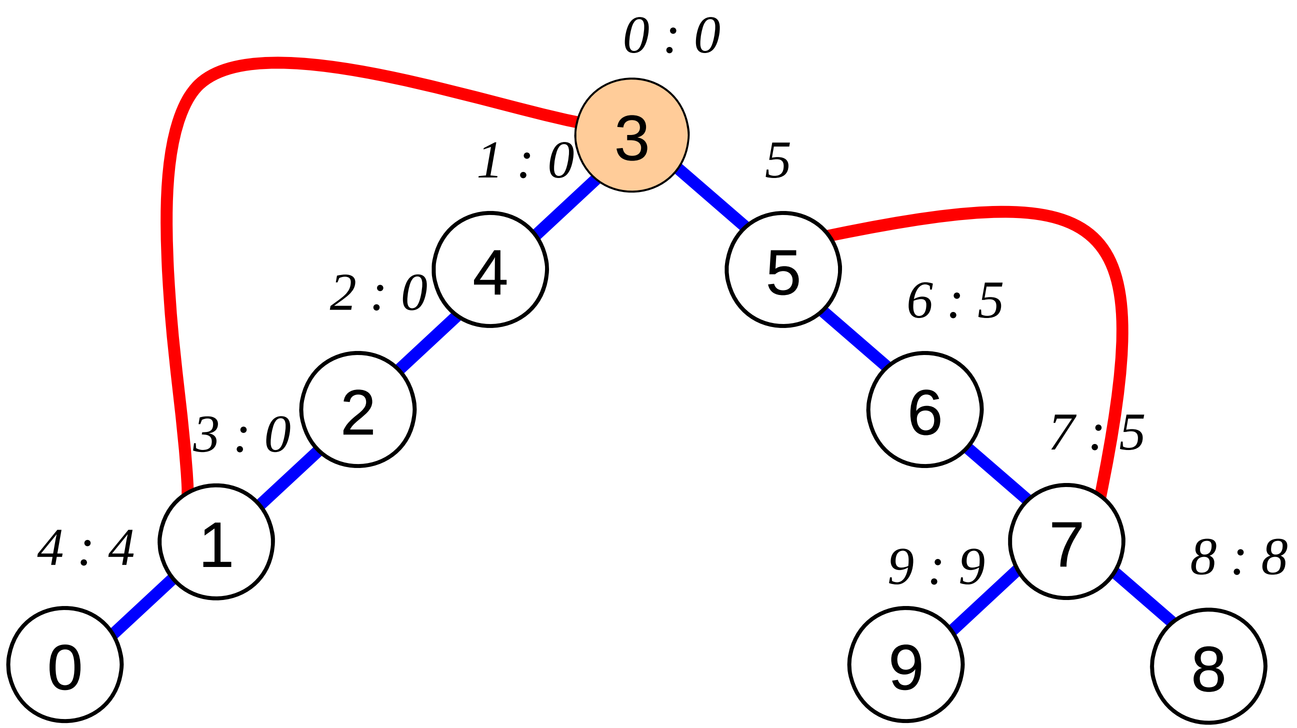
$$low(u) = \min\{dfn(u), \\ \min\{low(w) \mid w \text{ is a child of } u\}, \\ \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(6)?

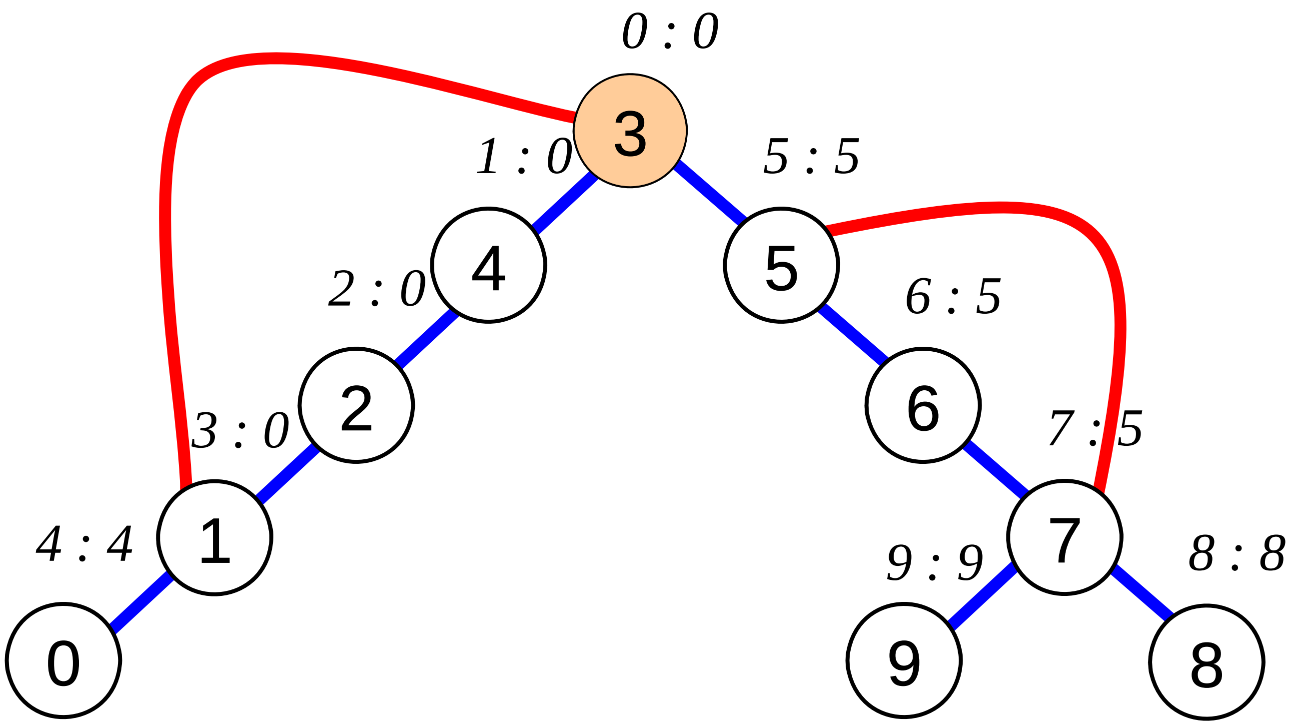
$$low(u) = \min\{dfn(u), \min\{low(w) \mid w \text{ is a child of } u\}, \min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



4.5 Biconnected Components

- low(u)
  - Ex) What are low(5)?

$$low(u) = \min\{dfn(u),$$
$$\min\{low(w) \mid w \text{ is a child of } u\},$$
$$\min\{dfn(v) \mid (u,v) \text{ is a back edge}\}\}$$



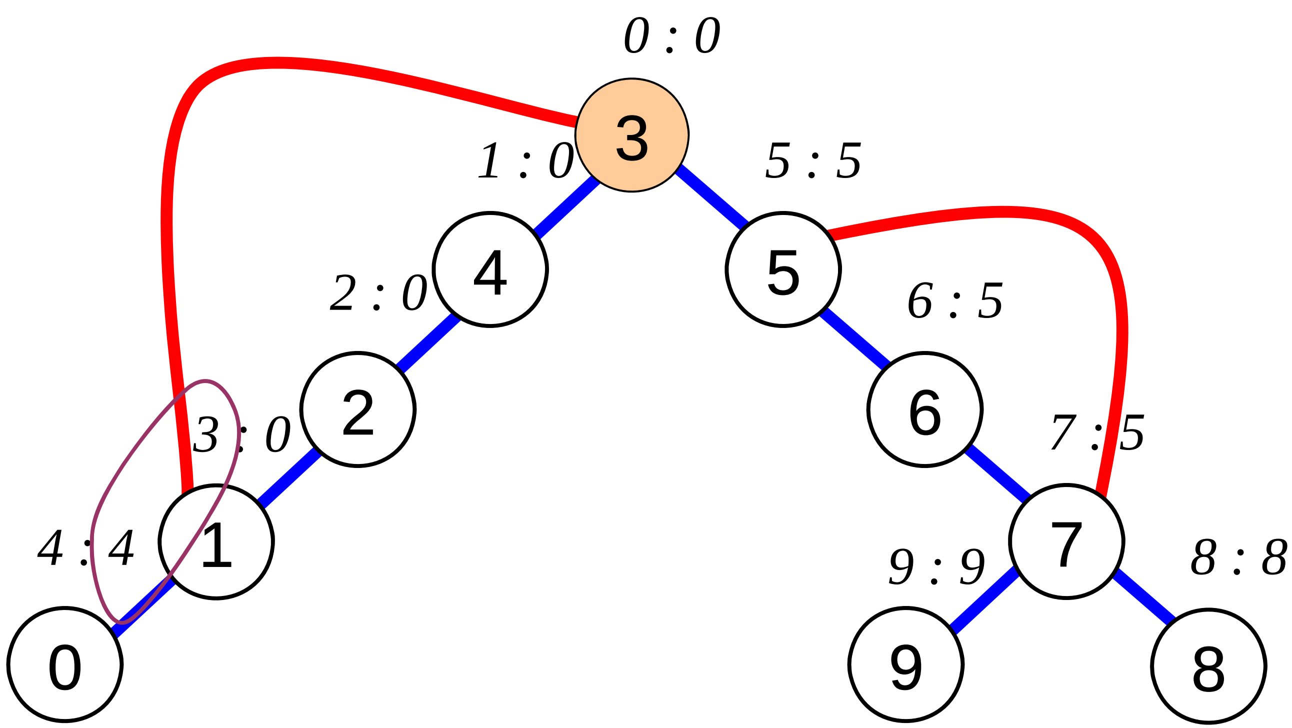
4.5 Biconnected Components

- Articulation points
  - u is an articulation point,
    - if u is either the root of the spanning tree with two or more childs,
    - or u is not a root and has a child w such that low(w) ≥ dfn(u)

	0	1	2	3	4	5	6	7	8	9
dfn	4	3	2	0	1	5	6	7	8	9
low	4	0	0	0	0	5	5	5	8	9

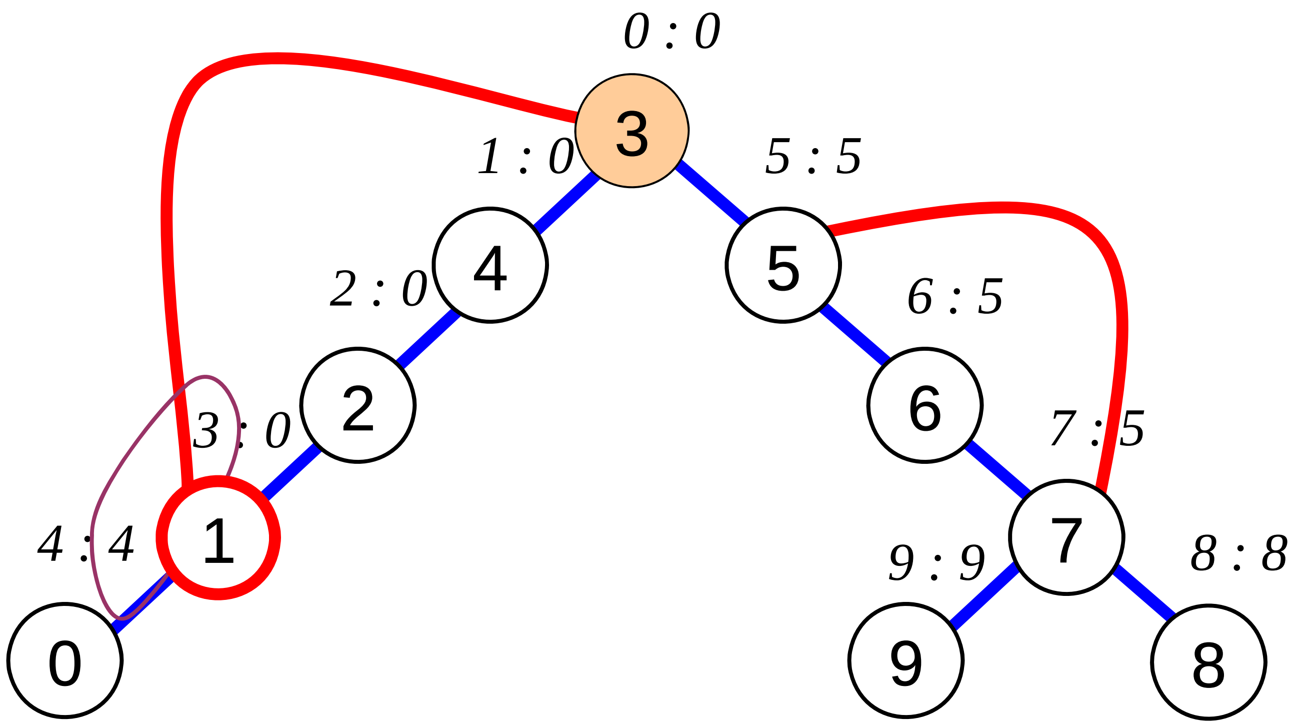
4.5 Biconnected Components

- Determine articulation points
  - At 1, low(w) ≥ dfn(u)?



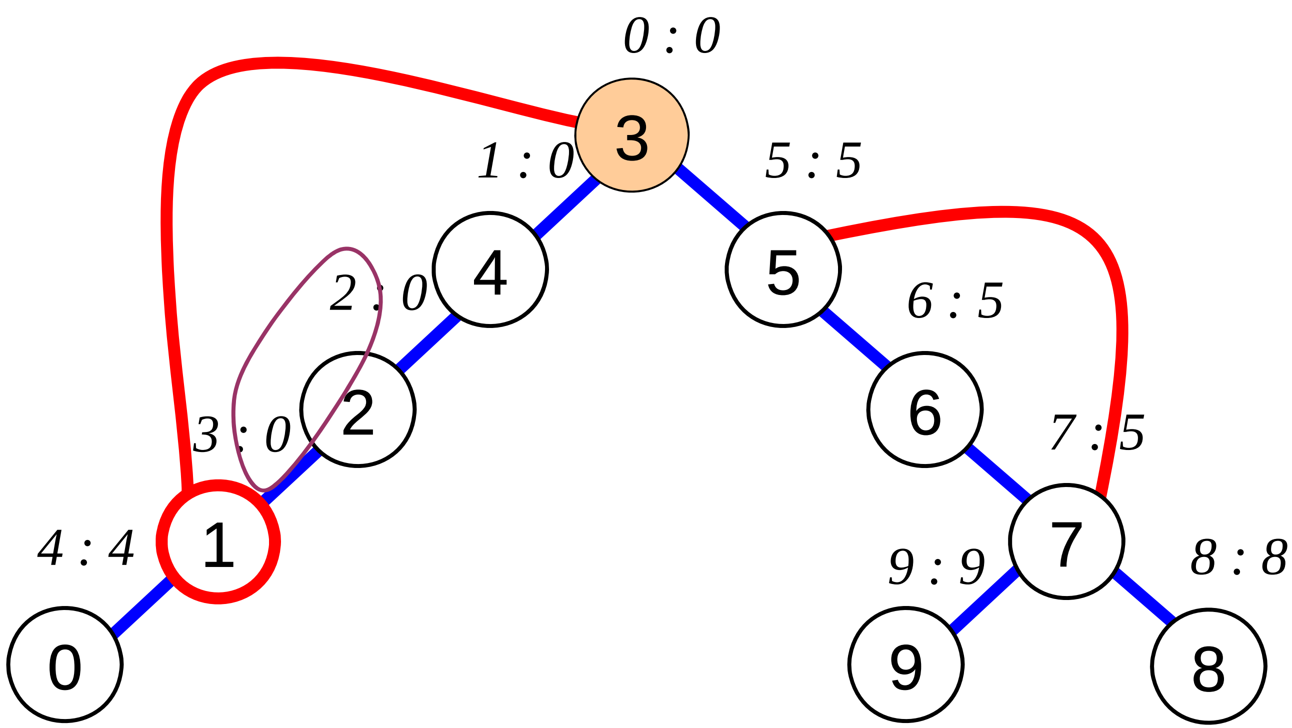
4.5 Biconnected Components

- Determine articulation points
  - At 1, low(w) ≥ dfn(u)?



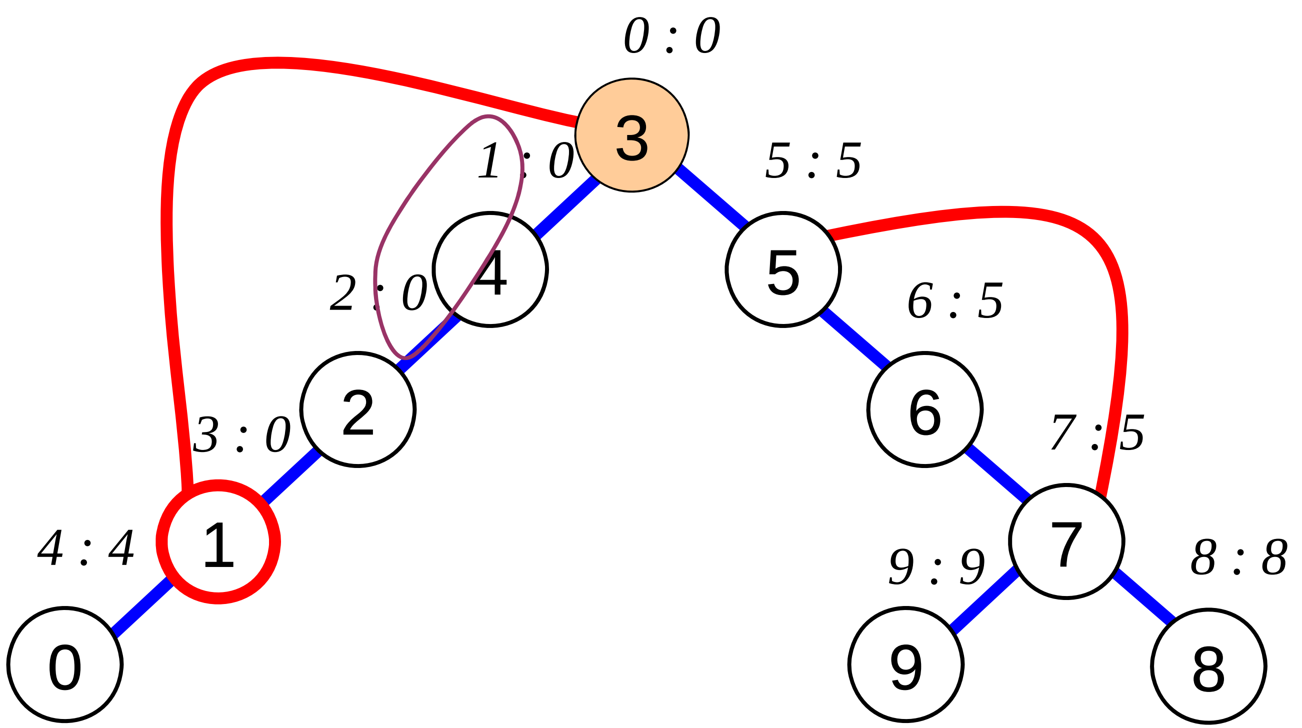
4.5 Biconnected Components

- Determine articulation points
  - At 2, low(w) $\geq$ dfn(u)?



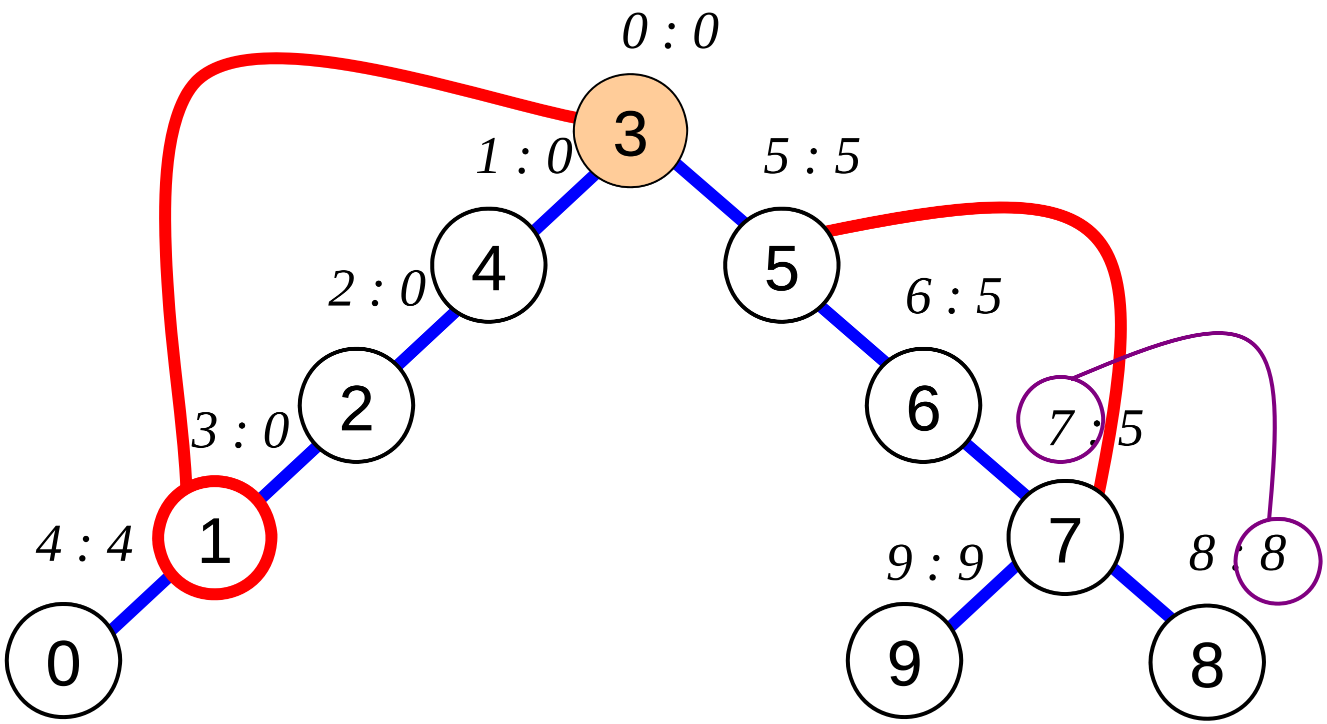
4.5 Biconnected Components

- Determine articulation points
  - At 4, low(w) ≥ dfn(u)?



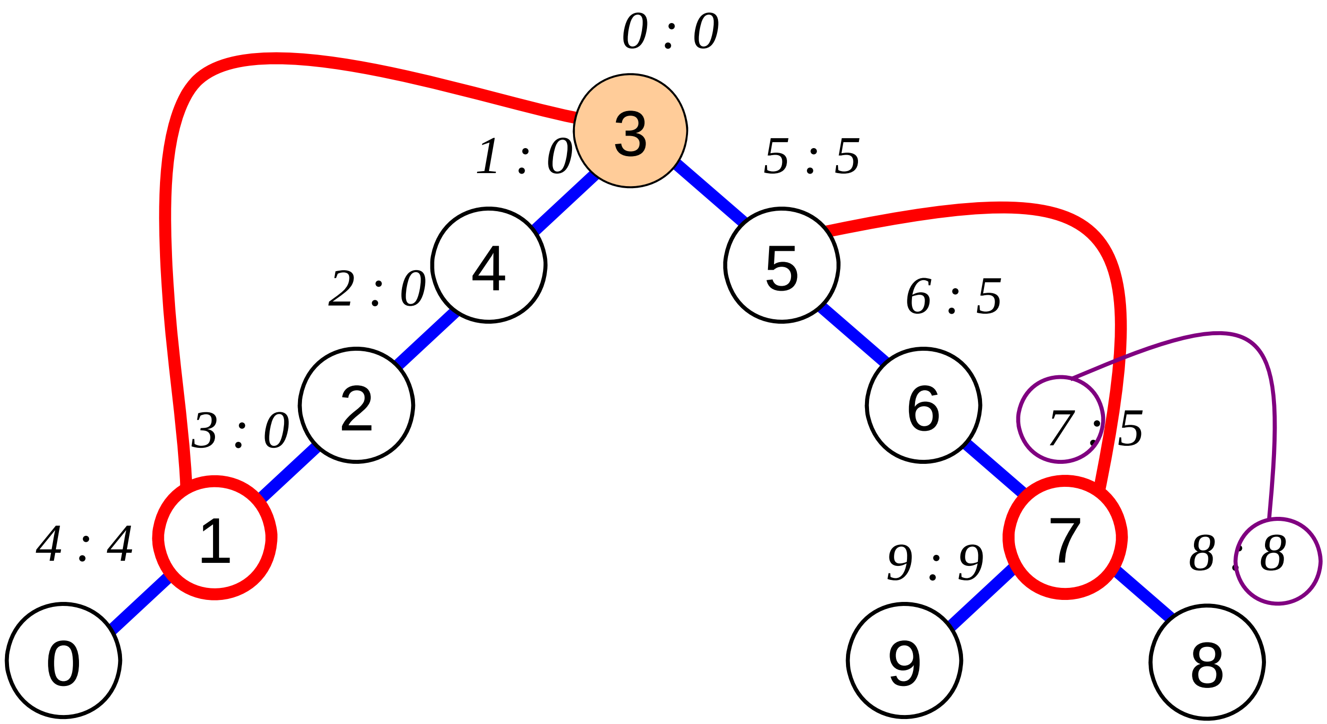
4.5 Biconnected Components

- Determine articulation points
  - At 7, low(w) ≥ dfn(u)?



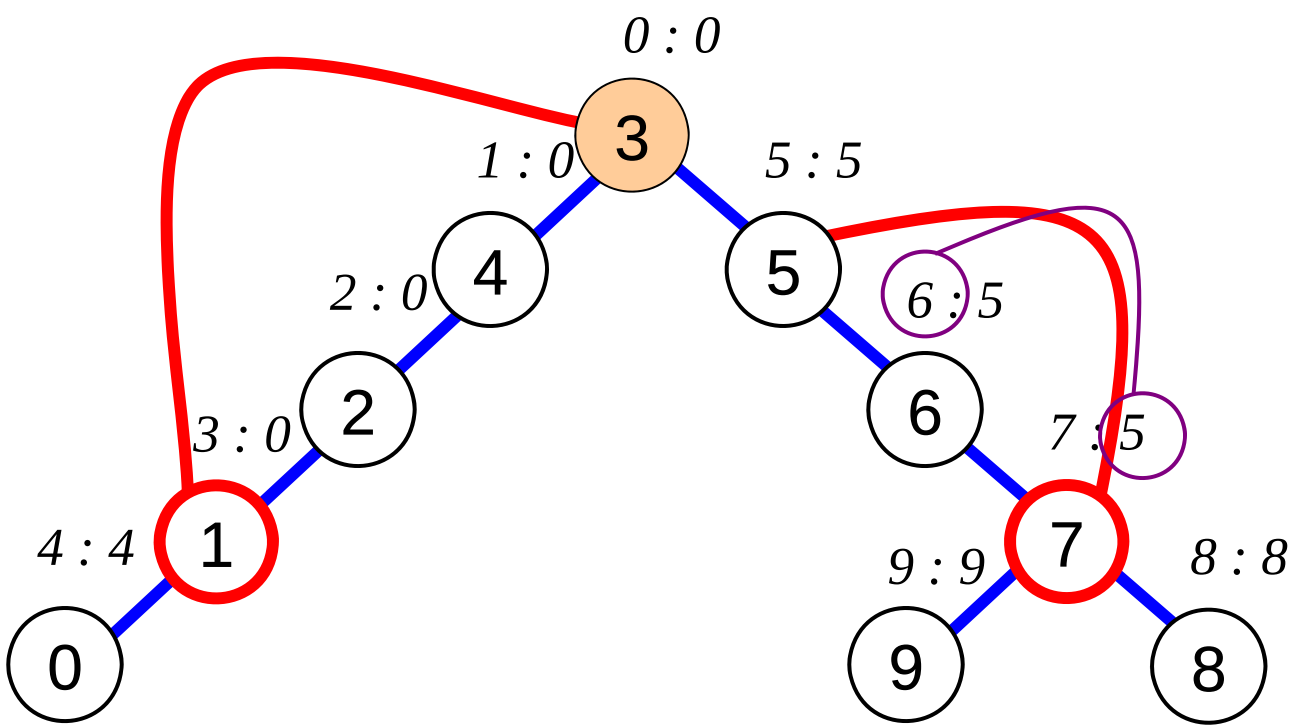
4.5 Biconnected Components

- Determine articulation points
  - At 7, low(w) $\geq$ dfn(u)?



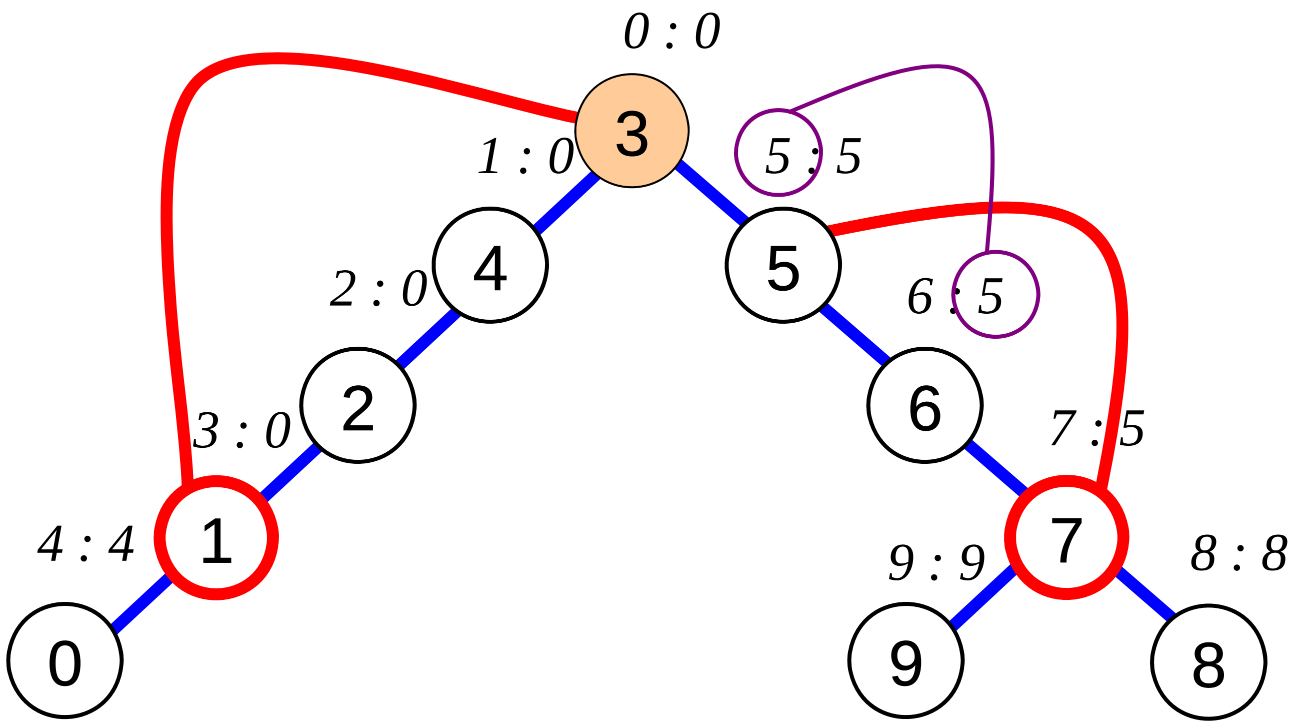
4.5 Biconnected Components

- Determine articulation points
  - At 6, low(w) ≥ dfn(u)?



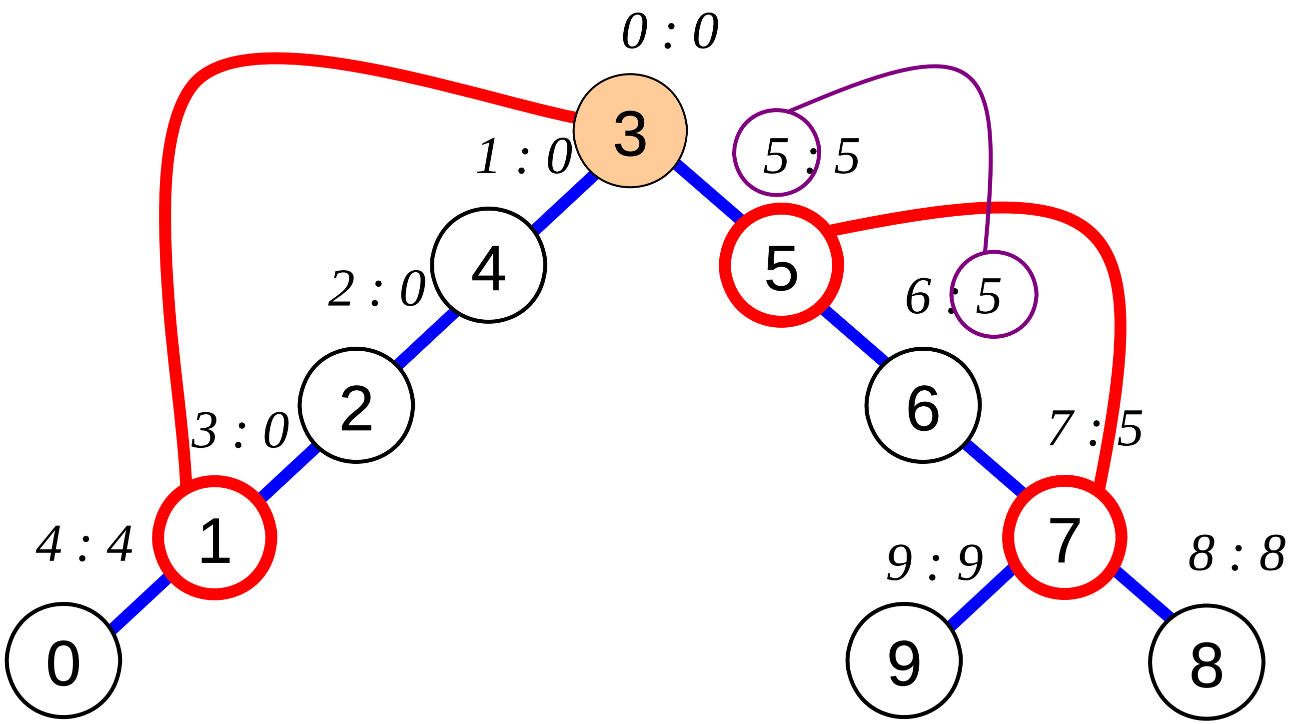
4.5 Biconnected Components

- Determine articulation points
  - At 5, low(w) $\geq$ dfn(u)?



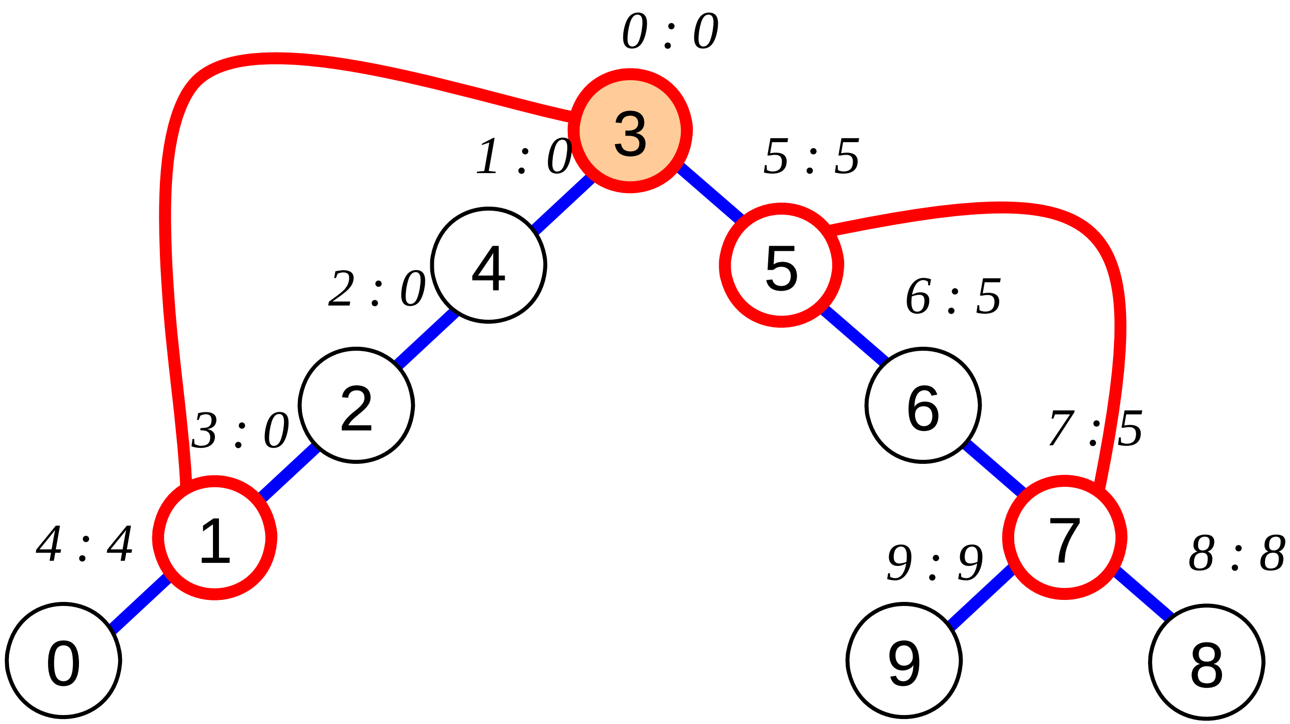
4.5 Biconnected Components

- Determine articulation points
  - At 5, low(w) ≥ dfn(u)?



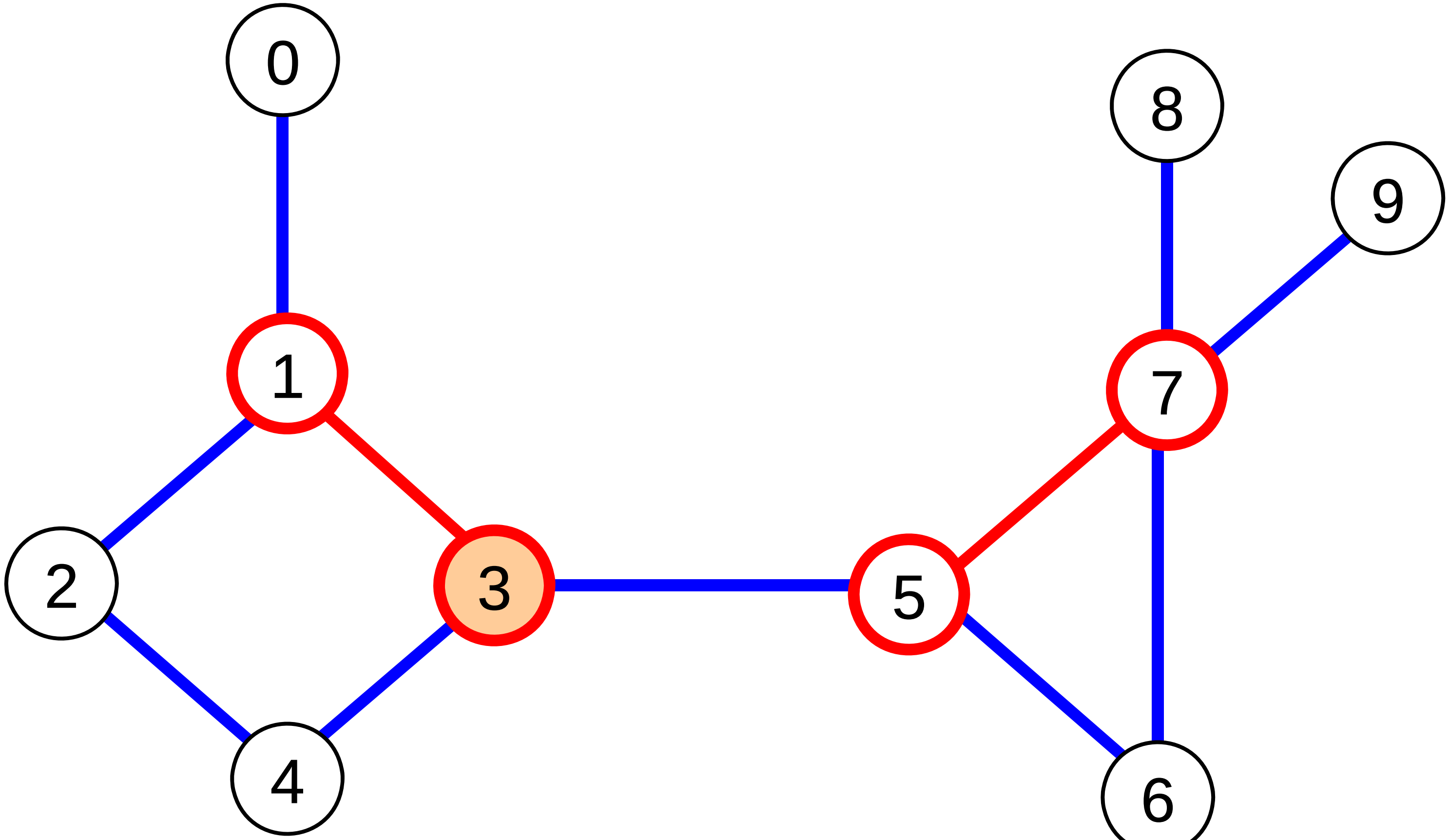
4.5 Biconnected Components

- Total articulation points
  - {u| low(w) ≥ dfn(u)} + root



4.5 Biconnected Components

- Total articulation points
  - $\{u \mid low(w) \geq dfn(u)\}$ + root



4.5 Biconnected Components

- DFS (1) → computing dfn
  - Build an array for dfn & low, instead of visit
  - Initialize dfn[v] = -1 (visit[v] = FALSE)
  - Call dfs (u, v) instead of dfs (u) (v: parent of u)
  - Declare num and initialize to 0

```
int visit[MAX_VERTEX]; // FALSE로 초기화
void dfs ( u )
{
  visit[u] = TRUE;

  for ( w = graph[u]; w; w = w->link )
    if ( !visit[w] )
      dfs ( w );
  }
}
```

```
int dfn[MAX_VERTEX]; // -1로 초기화
int low[MAX_VERTEX]; // -1로 초기화
int num = 0;
void dfs1 ( u, v ) // v는 u의 부모
{
  dfn[u] = num++;

  for ( w = graph[u]; w; w = w->link )
    if ( dfn[w] < 0 )
      dfs1 ( w, u );
  }
}
```

4.5 Biconnected Components

- DFS (2) → computing low

```
void dfs2 ( u, v ) // v는 u의 부모
{
  dfn[u] = low[u] = num++;

  for ( w = graph[u]; w; w = w->link )
    if ( dfn[w] < 0 ) {
      dfs2 ( w, u );
      low[u] = min (low[u], low[w]);
    }
    else if ( w != v )
      low[u] = min (low[u], low[w]);
  }
}
```

4.5 Biconnected Components

- DFS (3) → computing articulation point

```
void dfs3 ( u, v ) // v는 u의 부모
{
  dfn[u] = low[u] = num++;

  for ( w = graph[u]; w; w = w->link )
    if ( dfn[w] < 0 ) {
      dfs3 ( w, u );
      low[u] = min (low[u], low[w]);
      if ( low[w] >= dfn[u] )
        print ("u: articulation point");
    }
    else if ( w != v )
      low[u] = min (low[u], low[w]);
  }
}
```

4.5 Biconnected Components

- DFS (4) → biconnected component

```
void dfs4 ( u, v ) // v는 u의 부모
{
  dfn[u] = low[u] = num++;

  for ( w = graph[u]; w; w = w->link )
    if ( dfn[w] < 0 ) {
      push (u, w);
      dfs4 ( w, u );
      low[u] = min (low[u], low[w]);
      if ( low[w] >= dfn[u] ) {
        print("new bicon:");
        do {
          pop (&x, &y);
          print ( <x, y> );
        } while ( x != u || y != w );
      }
    }
    else if ( w != v )
      low[u] = min (low[u], low[w]);
  }
}
```

4.5 Biconnected Components

- Time complexity
  - Time complexity of DFS → $O(n + m)$
  - Time complexity of BCC → $O(n + m)$

All about graph

Type	Purpose	Operations	Performance
DFS	Traverse all vertices	Visiting all vertices & visiting all edges	O(n) + O(m)
SCC	Finding SCC	DFS on G^R and G	O(DFS)
BCC	Finding BCC	DFN & Low → Articulation Point → BCC in DFS	O(DFS)
BFS			
Dijkstra			
Floyd			
Kruskal (Greedy)			
Prim (Greedy)			
MultiStage (Dynamic)			