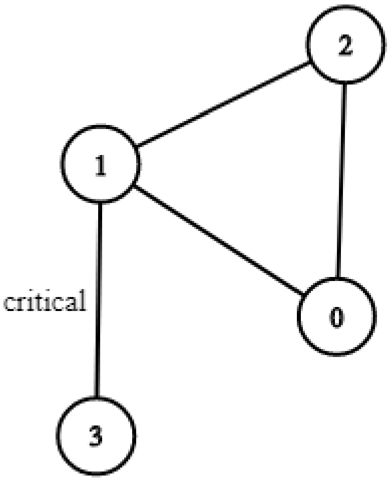
1192. Critical Connections in a Network

There are `n` servers numbered from `0` to `n - 1` connected by undirected server-to-server `connections` forming a network where `connections[i] = [ai, bi]` represents a connection between servers `ai` and `bi`. Any server can reach other servers directly or indirectly through the network.

A *critical connection* is a connection that, if removed, will make some servers unable to reach some other server.

Return all critical connections in the network in any order.

Example 1:



```
Input: n = 4, connections = [[0,1],[1,2],[2,0],[1,3]]
Output: [[1,3]]
Explanation: [[3,1]] is also accepted.
```

Example 2:

~~~
Input: n = 2, connections = [[0,1]]
Output: [[0,1]]
``````Python
from collections import defaultdict
class Solution:
    def criticalConnections(self, n: int, connections: List[List[int]]) -> List[List[int]]:
        graph = defaultdict(list)
        for u,v in connections:
            graph[u].append(v)
~~~

```
            graph[v].append(u)
        visited = [False]*n
        parent = [-1]*n
        disc = [-1]*n
        low = [-1]*n
        res = []
        time = 0
        self.dfs(0,graph,visited,parent,disc,low,res,time)
        return res


    def dfs(self,u,graph,visited,parent,disc,low,res,time):
        disc[u] = time
        low[u] = time
        time =time+1
        visited[u] = True

        for v in graph[u]:
            #First condition:
            if parent[u]==v:
                continue
            elif visited[v]==True:
                low[u] = min(low[u],disc[v])
            else:
                parent[v] = u
                self.dfs(v,graph,visited,parent,disc,low,res,time)
                low[u] = min(low[u],low[v])
                if low[v]>disc[u]:
                    res.append((u,v))
```

Applicaton of Tarjan's algorithm. See the difference between articulation and bridges problem.

Articulation : low[v]>=dis[u]

Bridges: low[v]>dis[u]