Bridges and Articulation Points

Oliver-Matis Lill

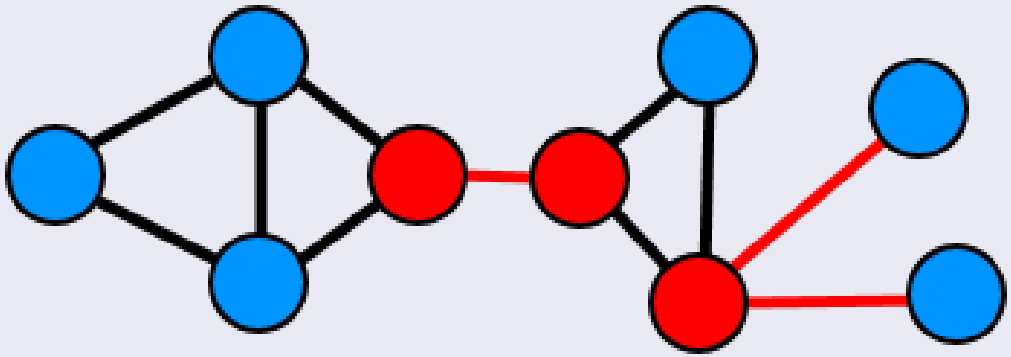
June 19, 2017

Bridges and Articulation Points Definition

- We are dealing with a connected undirected graph
- A bridge is an edge that increases the number of connected components when removed
- An articulation point is a vertex that does the same

Example

Articulation points are red vertices and bridges are red edges



The Problem

- You have a connected undirected graph with $N < 3 \cdot 10^5$ vertices and $M < 3 \cdot 10^5$ edges
- Find all bridges and articulation points
- How would you approach this?

Solution Idea

- Use Depth-First Search to pick a subset of $N - 1$ edges, that form a tree
- All other edges are "extra" edges that create cycles
- Notice that only the tree edges can be bridges
- Each extra edge creates a simple cycle with the simple path between its endpoints. No edge in that cycle can be a bridge.

Tree Construction Code

Code

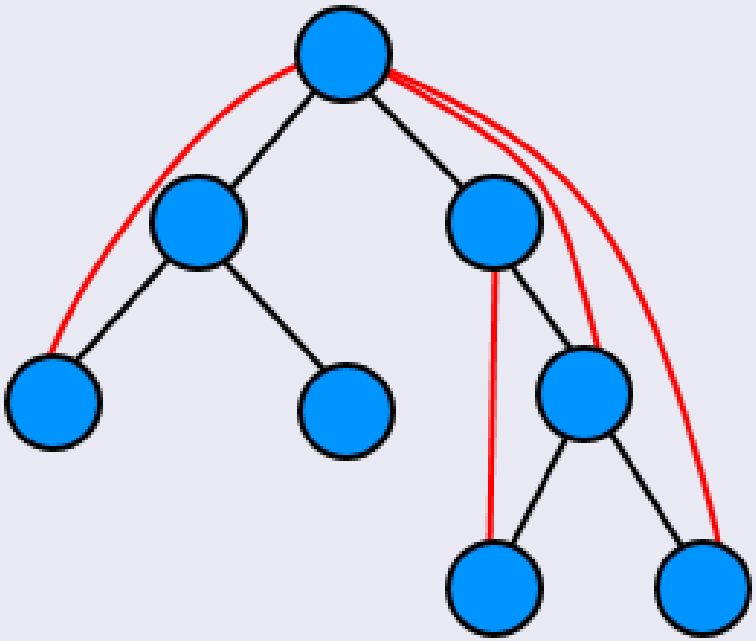
```
void dfsTreeConstruct(Node* cur) {
    cur->explr = true;

    for(Edge* edge : cur->edges) if(!edge->used) {
        edge->used = true;
        Node* next = edge->from(cur);
        if(next->explr)
            cur->extra.push_back(next);
        else {
            next->depth = cur->depth + 1;
            dfsTreeConstruct(next);
            cur->children.push_back(next);
        }
    }
}
```

Tree Example

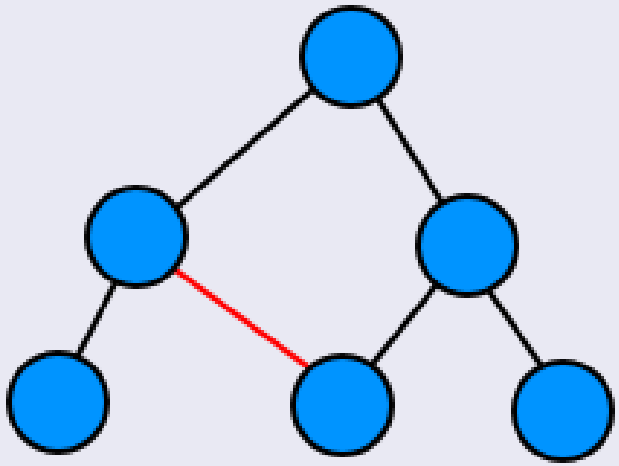
Example

Extra edges are in red:



Tree Caveat

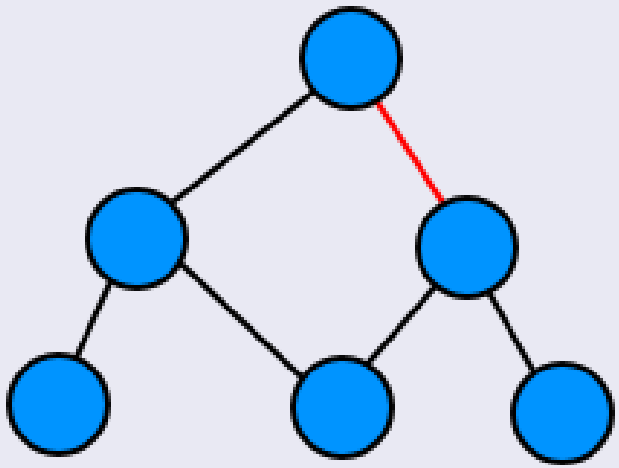
- In an extra edge, if one endpoint is an ancestor of the other, then things are simple
- What about a case where neither endpoint is an ancestor of the other:



- How would we deal with such cases?

Tree Caveat Resolution

- Turns out that case doesn't appear since the tree was constructed with DFS
- In DFS, all node descendants are traversed before moving up
- As such, the constructed tree in that case would look like this instead:



Finding Bridges

- That makes finding bridges really simple
- For each vertex, the edge to its parent is a bridge if and only if there is no extra edge to any of its ancestors in its subtree
- In fact we only care about the upmost depth reachable through the extra edges in its subtree

Finding Bridges Code

Code

```
void countBridges(Node* cur) {
    cur->upperDepth = cur->depth;

    for(Node* next : cur->children) {
        countBridges(next);
        cur->upperDepth = min(cur->upperDepth, next->upperDepth);
    }
    for(Node* next : cur->extra)
        cur->upperDepth = min(cur->upperDepth, next->depth);

    if(cur != root && cur->upperDepth == cur->depth)
        numberOfBridges += 1;
}
```

Conclusion

- We have presented a fairly simple algorithm to find bridges
- Finding articulation points has been left as an exercise
- This decomposition into tree and extra edges could be useful for proving various things