

Algorithms and Data Structures



COMP261

Articulation Points 1: Idea

Yi Mei

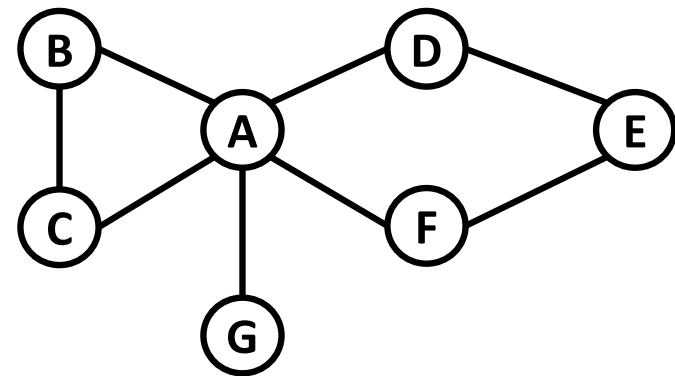
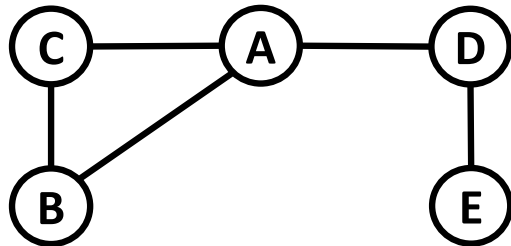
yi.mei@ecs.vuw.ac.nz

Outline

- Articulation Points
- How to find articulation points
 - A bad algorithm
- A good (faster) algorithm

Articulation Point

- In a **connected** graph, an **articulation point** is such a point that the graph will become **disconnected** (split into 2 separate pieces) when it is **removed** from the graph (along with the edges associated to it)
- Represents the **vulnerability** of a connected graph
- Which are articulation points?



Why Articulation Points

- Many real-world applications
 - Social network
 - Wireless sensor network
 - Road network
 - Military
 - ...



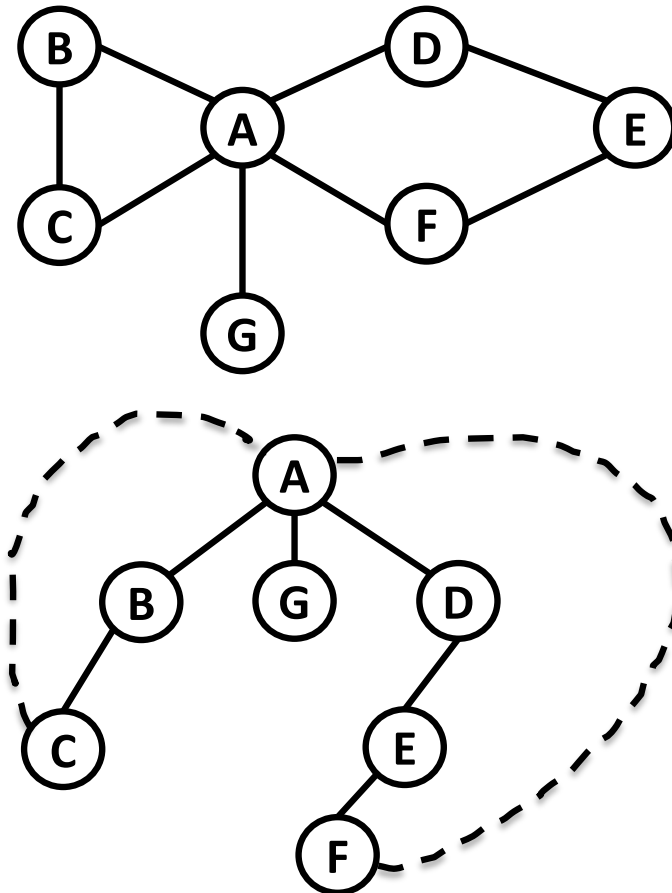
- How to find all the articulation points?

Finding Articulation Points

- **A simple way**
 - For each node, **remove it** from the graph, and **test whether the graph will become disconnected**
 - Depth-First Search (DFS) rooted from the node to be removed
 - If the graph will become disconnected, then the **DFS tree will have more than one branches**

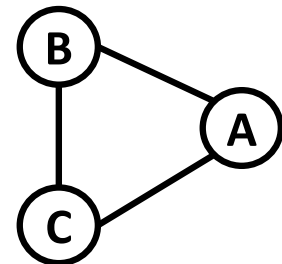
```
All nodes are unvisited, A is visited;  
recDFS(first neighbour of A);  
if (there exists a neighbour of A unvisited)  
    A is an articulation point;
```

```
recDFS(node) {  
    if (node is unvisited) {  
        set node to visited;  
        for (each neighbour of node) {  
            if (neighbour is unvisited)  
                recDFS(neighbour);  
        }  
    }  
}
```



Finding Articulation Points

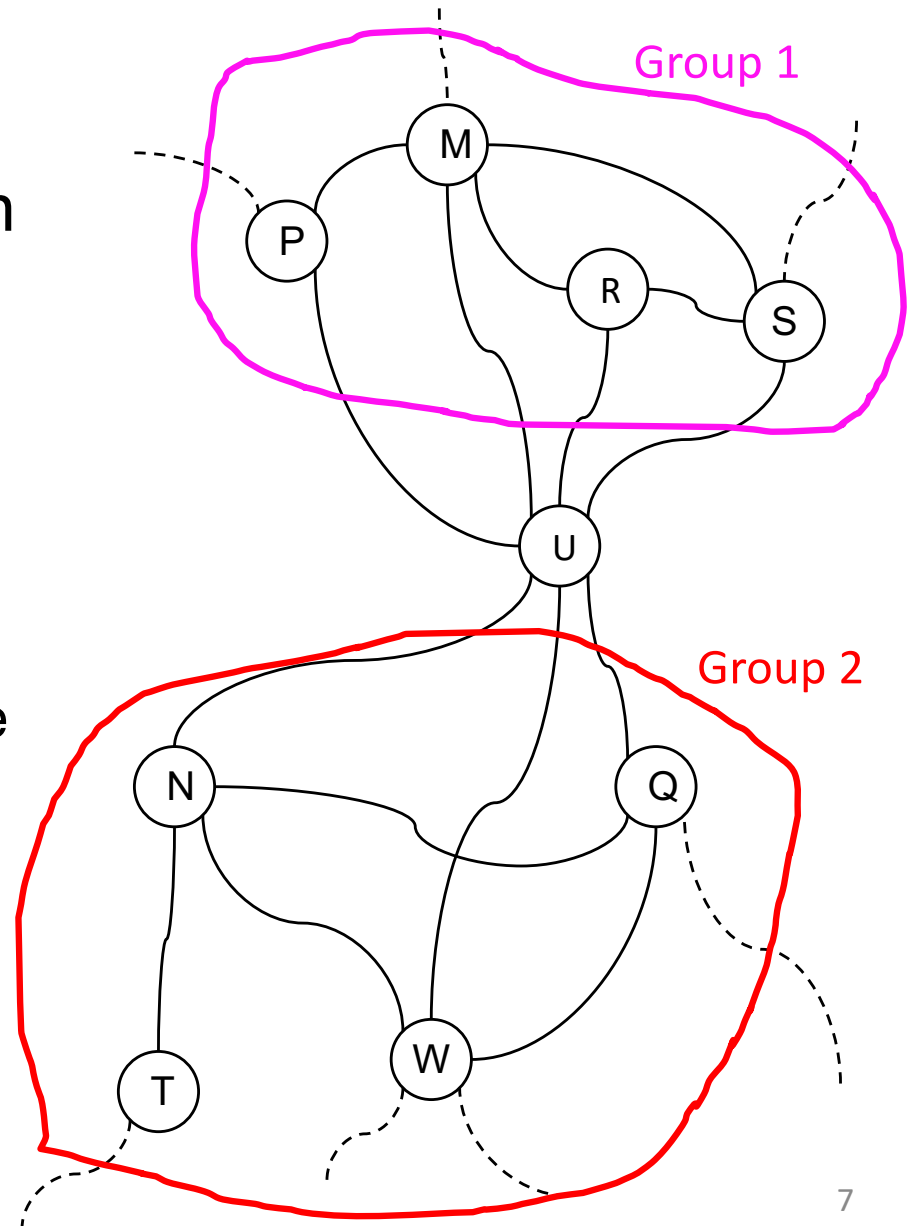
- But this way is very inefficient
 - Cost of DFS: $O(e)$ or $O(n^2)$ for very dense graphs
 - Cost of algorithm: $O(ne)$ or $O(n^3)$ for very dense graphs
- Many visits are **repeated many times unnecessarily**
 - 2 visits for A, B and C as root, **6 visits** in total
 - But **3 visits are enough** to tell none of A, B and C is articulation point



- A new efficient algorithm that finds all the articulation points in **a single DFS**?

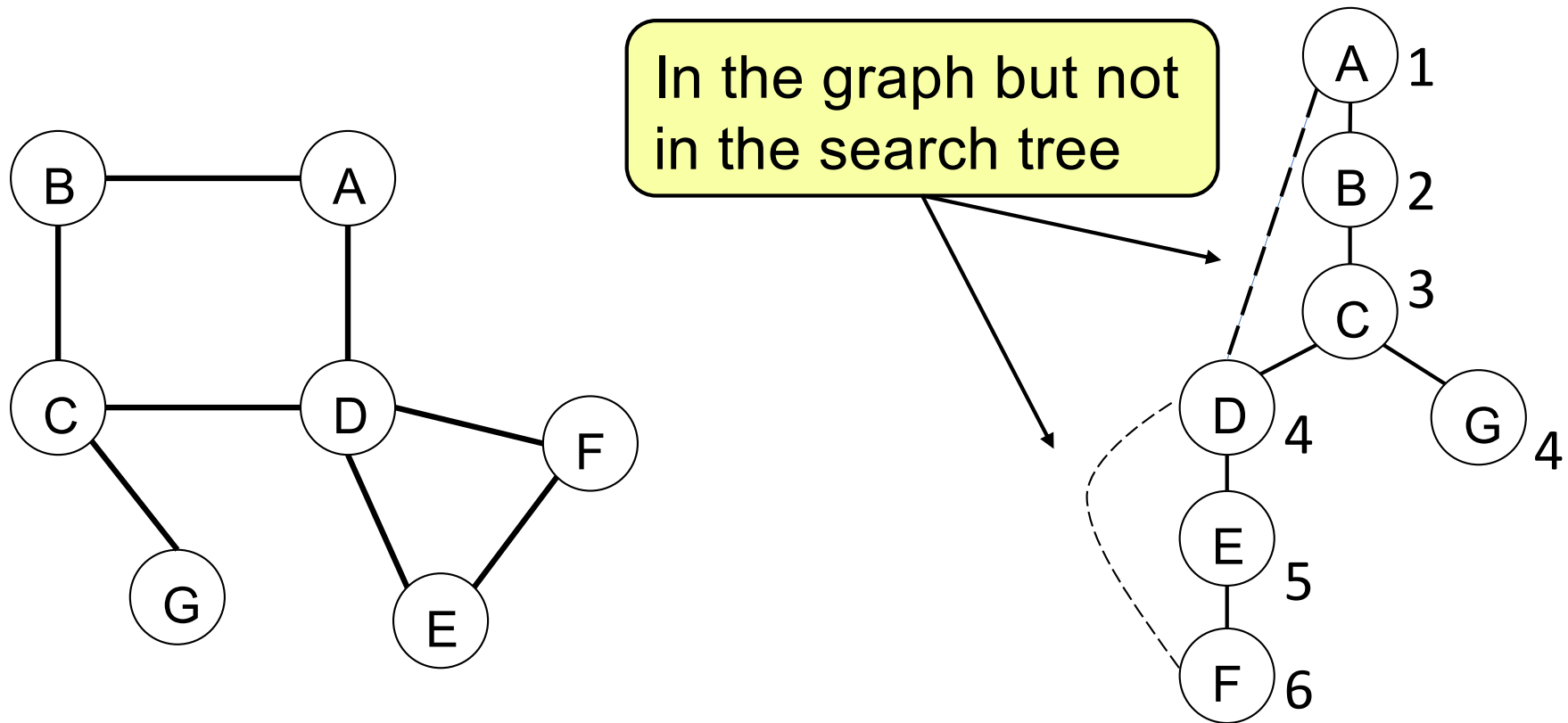
A New Efficient Algorithm

- **Idea:** an articulation point separates the graph into two groups, so that all paths from nodes in one group to nodes in the other group **MUST** go through the node.
- **Example:**
 - node U is an articulation point, since all paths between any node in group 1 and any node in group 2 must go through U.



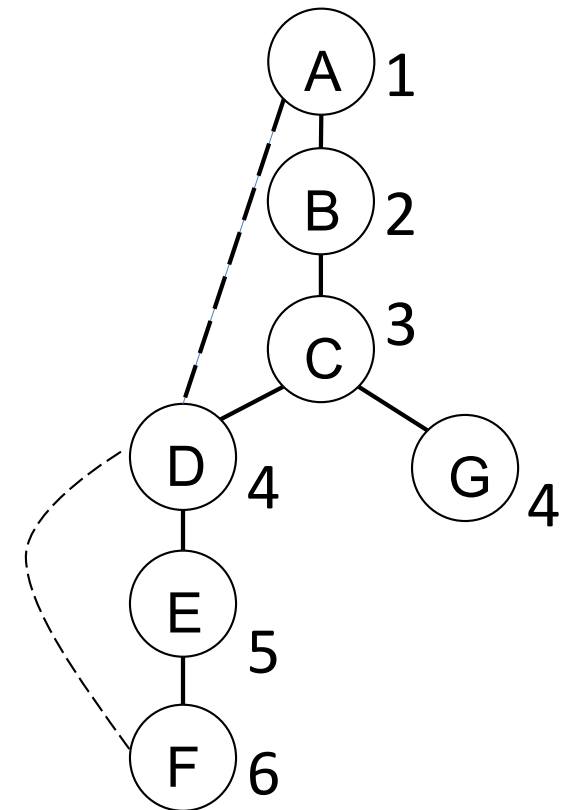
Articulation Points Algorithm

- Traverse the graph using DFS, and index the nodes through the search
 - Assign a search **depth** to each node



Articulation Points Algorithm

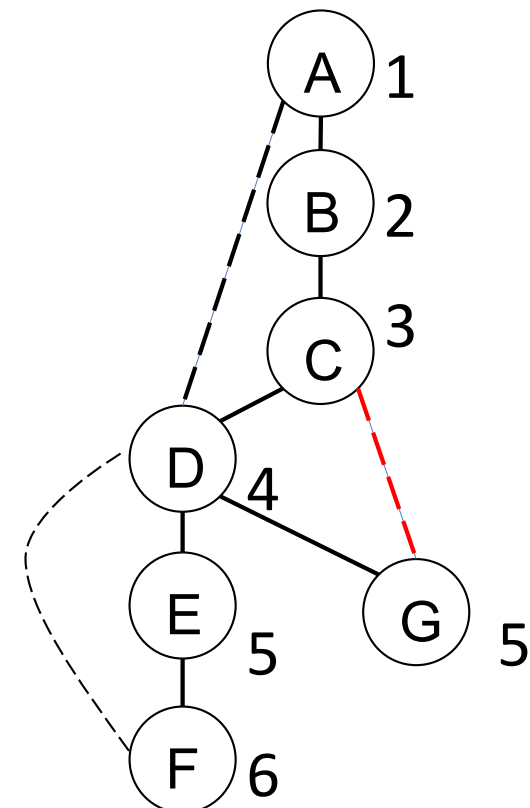
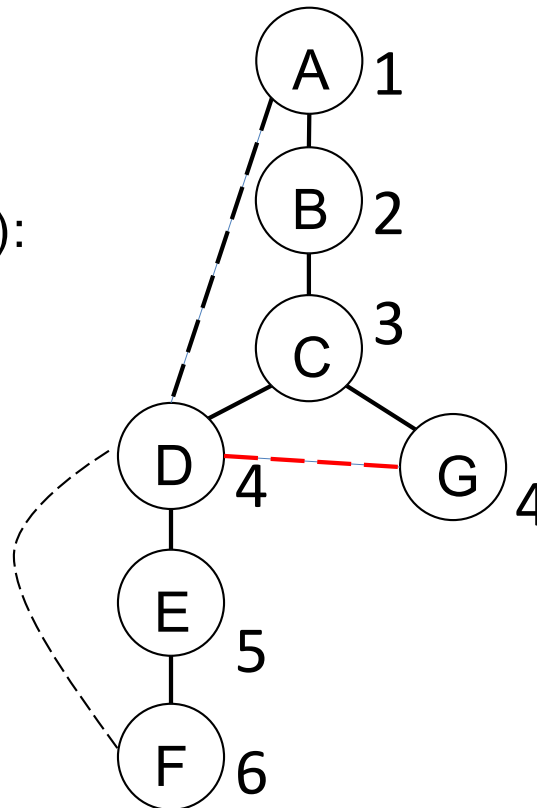
- In the search tree, each node separates the nodes into two subsets
 - **Children set:** Nodes **in its subtree**
 - **Parents set:** Nodes **not in its subtree**
- Example:
 - For node D: {E, F} and {A, B, C, G}
 - For node C: {D, E, F, G} and {A, B}
 - For node A: {} and {B, C, D, E, F, G}
- Check if **Children** and **Parents** are separated after removing the node
 - The node is an articulation point if at least one **child** node and **parents** are separated after removing it
 - There is no alternative path
- Should we check nodes in different subtrees?
E.g. {D,E,F} and {G} for C?



Articulation Points Algorithm

- **Theorem:** no matter **which root node** the DFS starts from, and **in which order the neighbours are checked**, there is **no alternative path between subtrees of a node**
 - **Proof:** (easy) if there is an alternative path between subtrees, then the sub-trees should have been in the same subtree in the first place due to DFS

- **Example:**
 - If alternative path (D,G):
 - G is a child of D, not C

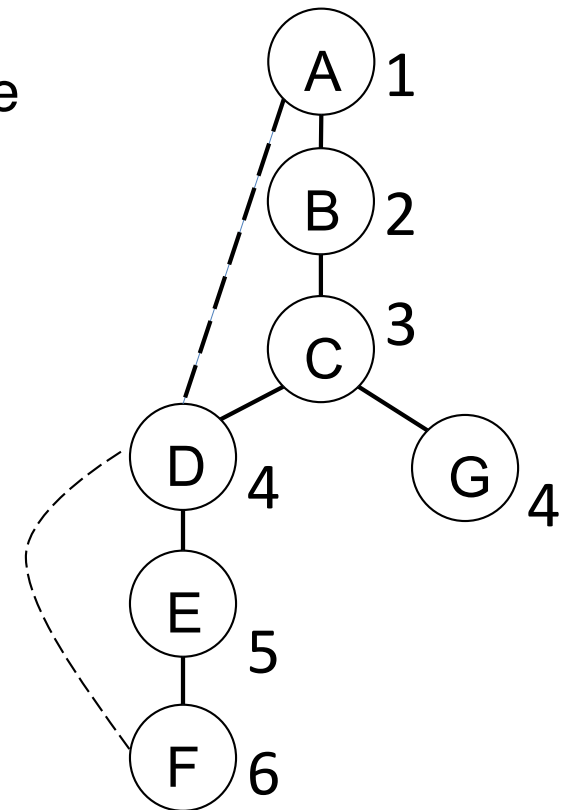


Articulation Points Algorithm

- **Theorem:** a node A is an articulation point, if and only if there exists a child node B, for which there is no alternative path from B to any of the parents
 - Removing node A will separate B and the parents
 - This is independent of the root node of the DFS and order that the neighbours are checked
- Checking alternative paths for a child node B of node A
 - An edge in the graph, but not in the DFS tree
 - Directly link node B to a parent node
 - Link another child node C in the same subtree of B to a parent node
 - All the child node in the same subtree are connected without node A

Articulation Points Algorithm

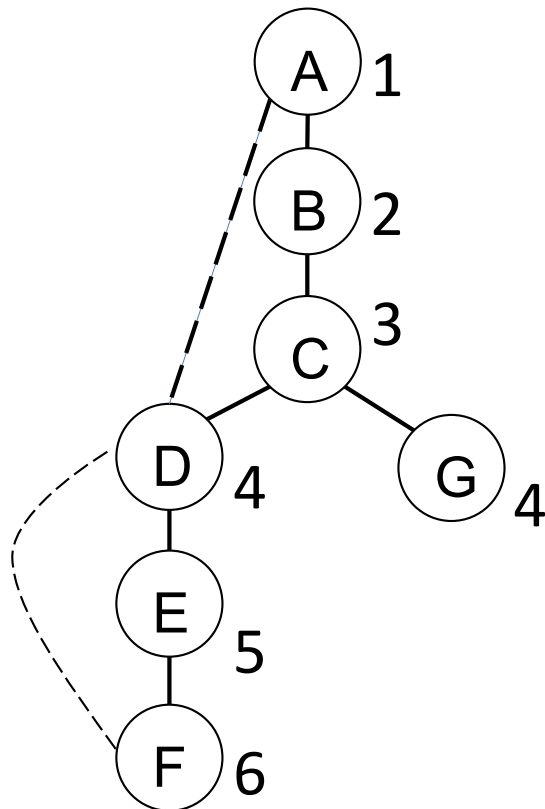
- Example:
 - 2 alternative paths (A, D) and (D, F)
 - C is an articulation point, because there is no alternative path from **the child G** to any of its parents {A, B}
 - B is not an articulation point, because the edge (A, D) offers an alternative path of all its children {C, D, E, F, G}
 - D? E?
- How about the root node?
 - **Parents is empty**
 - Checking alternative path is meaningless (**always no alternative path**)
 - Need to **check the root node separately**



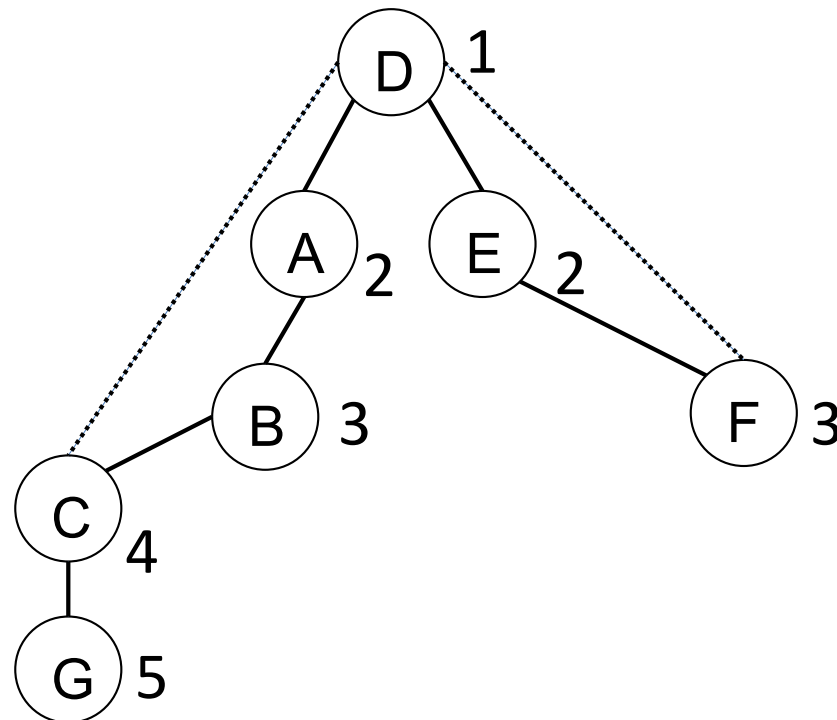
Articulation Points Algorithm

- **Theorem:** a **root node** is an **articulation point** if and only if it has **multiple sub-trees** in the DFS
 - Proof is easy (no alternative path between sub-trees)

A is not an articulation point
(one sub-tree)



D is an articulation point
(two sub-trees)



Summary

- Finding articulation points is important in many applications
 - Social network
 - Cyber-security
 - Wireless sensor network
 - ...
- Brute force search is time consuming and can be much improved
- A new efficient algorithm with **a single DFS**
 - Assign **depth** to each node during DFS (larger depth are children, smaller are parents)
 - Check for **root node**: **number of sub-trees**
 - Check for **other nodes**: **alternative path from children to parents?**
- Next lecture: implementation