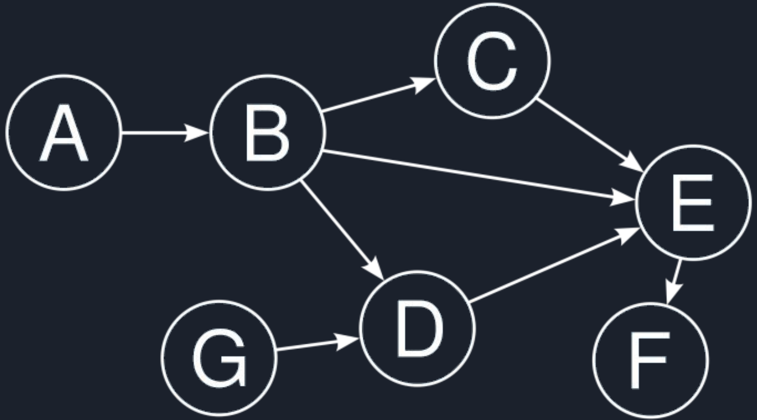
ProgTeam Spring Week 7

Topological Sort

Review: DAGs

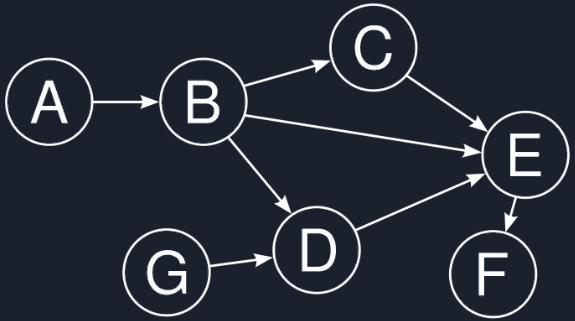
Directed Acyclic Graph:

- Directed: All edges are one way
- Acyclic: There are no cycles



Topological Sort

- One rule: all ancestors come before their descendants
- Example:
  - A-B-G-D-C-E-F is valid
  - A-G-B-C-D-E-F is valid
  - A-G-D-B-C-E-F is not!
    - B is an ancestor of D



Algorithm: BFS (Kahn’s Algorithm)

```
Queue Ready;
// Initialize to all “roots” of the DAG (here, A,G)
while Ready is not Empty:
    v <- Ready.poll(); // Remove first element from queue
    Remove v from graph
    for u in Adj[v]:
        if v was u’s last parent:
            add u to the queue
        endif
    endfor
endwhile
```

Example: Longest Path to a Vertex from “Root”

Must be finite because there are no cycles!

```
while Ready is not Empty:
    v <- Ready.poll(); // Remove first element from queue
    Remove v from graph
    for u in Adj[v]:
        $Longest[u] = max(Longest[u], Longest[v] + 1);$
        if v was u’s last parent:
            add u to the queue
        endif
    endfor
endwhile
```

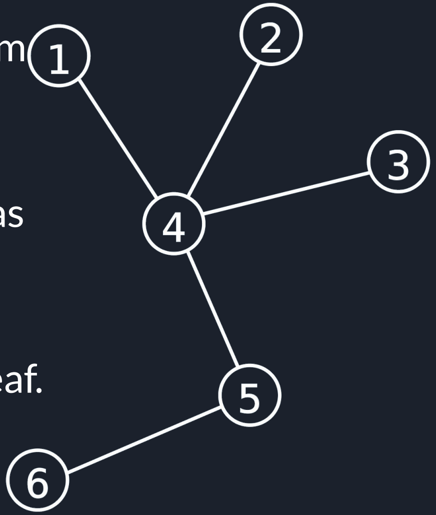
Example: Centroid of a Tree

Definition: The centroid of the tree is the vertex 'v' such that the furthest distance to 'v' is minimized.

Here '4' and '5' are both centroids; the furthest distance from either is length=2.

'6' is not a centroid because the furthest distance from '6' has length=3.

Approach: Clearly the furthest vertex from any vertex is a leaf. Let's remove all of the leafs, and then repeat this until there are one/two vertices left. These are the centroids.



Algorithm: Remove the leaves of the tree

```
Queue Ready;
// Initialize to all leaves  of the tree (here, A,G)
while Ready is not Empty:
    v <- Ready.poll(); // Remove first element from queue
    if Ready is Empty: return v; // Last element must be a centroid!
    Remove v from graph
    for u in Adj[v]:
        if u is now a leaf:
            add u to the queue
        endif
    endfor
endwhile
```