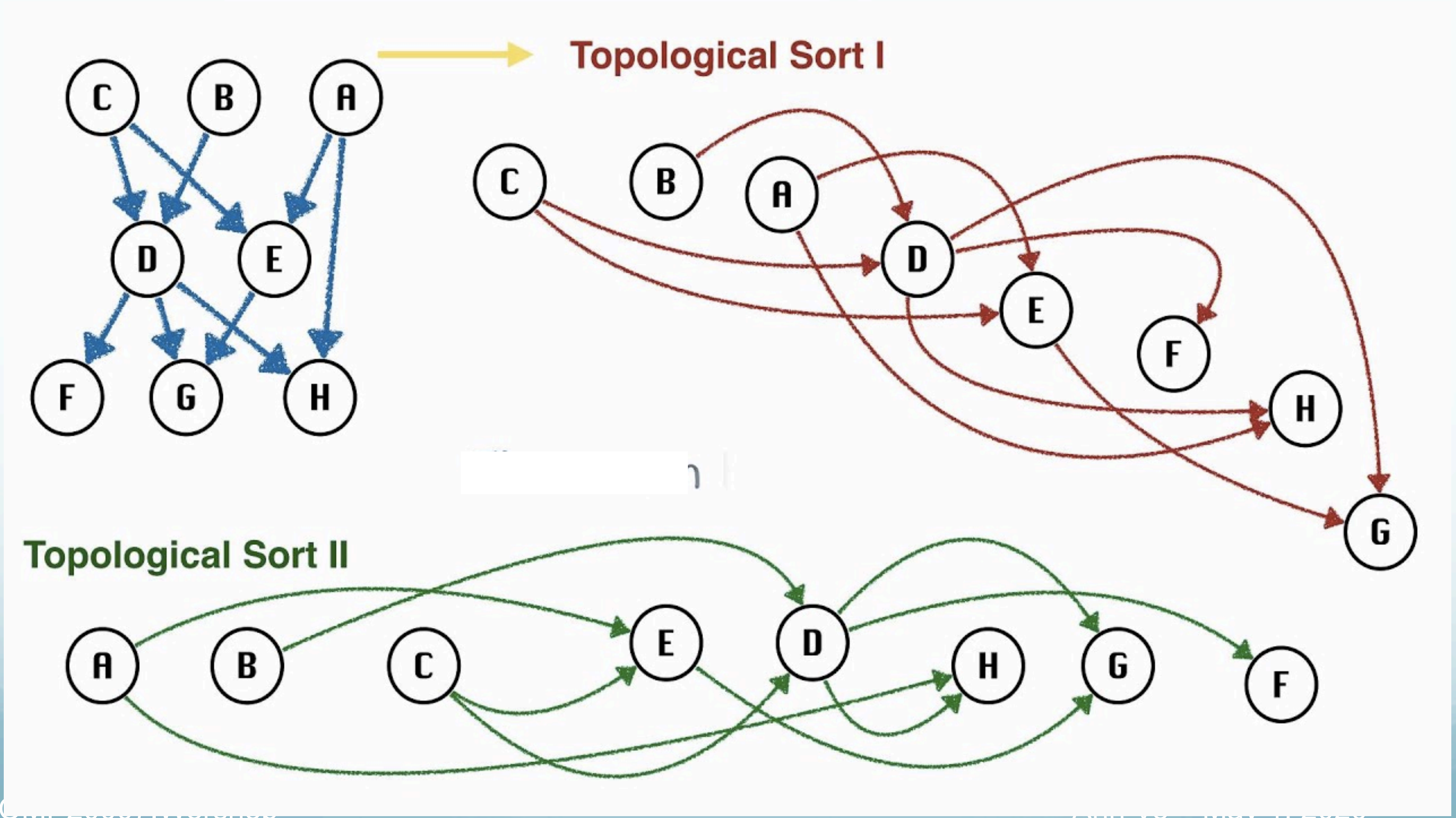
COMP20007 Workshop Week 7

	Preparation: - open `ws7.pptx` from `github.com/anhvir/c207` - open `wokshop7.pdf` (from `LMS`)
1	**Topic 1:** Topological Sorting *Group Work:* Problems T1 (toposort), T2 (Dijkstra's with negative weights)
2	**Topic 3:** Binary Trees & BST *Group Work:* Problems T3, T4, T5

Topological Sorting

A *topological ordering* of a directed acyclic graph (DAG) is a way of sorting the nodes of the graph such that all edges point in one direction: to nodes later in the ordering.

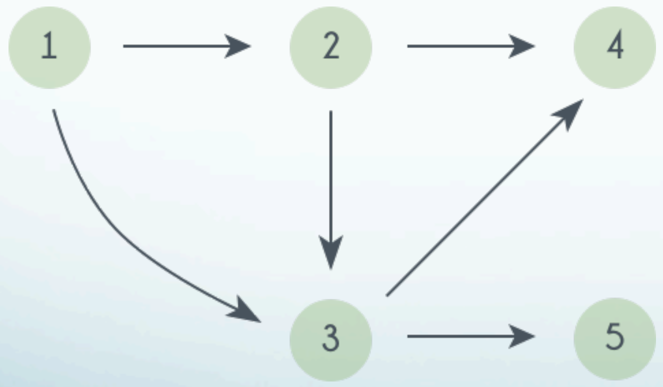


Topological Sorting

A *topological ordering* of a directed acyclic graph (DAG) is a way of sorting the nodes of the graph such that all edges point in one direction: to nodes later in the ordering.

One method: Select a source (node with no incoming edges), then remove this source and all of its incidents. Repeat this process until all nodes have been selected.

What's the complexity of this algorithm if using adjacency matrix? list?



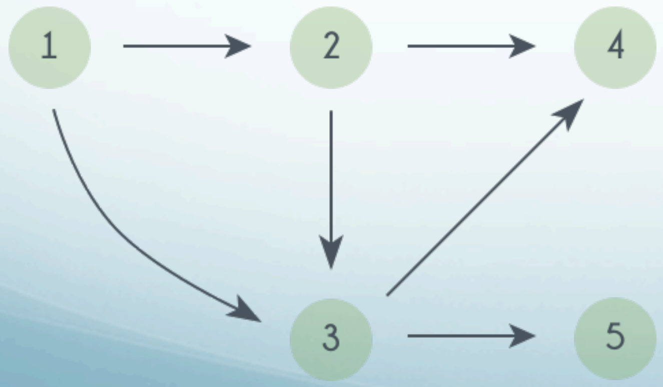
Topological Sorting

A *topological ordering* of a directed acyclic graph (DAG) is a way of sorting the nodes of the graph such that all edges point in one direction: to nodes later in the ordering.

Another Algorithm (discussed in lectures) involves running a DFS on the DAG and keeping track of the order in which the vertices are popped from the stack. The topological ordering will be the reverse of this order.

What's the complexity if using adjacency lists for graphs? adj matrix?

How to have the pop-order?

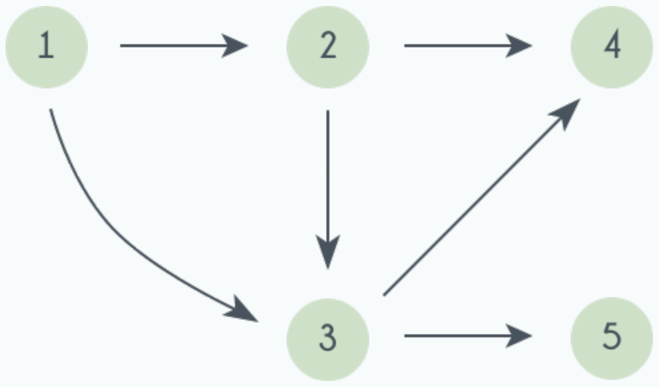


DFS: push- and pop-order (pre- and post-order)

```
function DFS(⟨V, E⟩)
    mark each node in $V$ with 0
    $count \leftarrow 0$
    for each $v$ in $V$ do
        if $v$ is marked 0 then
            DFSEXPLORE($v$)

function DFSEXPLORE($v$)
    $count \leftarrow count + 1$
    mark $v$ with $count$
    for each edge $(v, w)$ do
        if $w$ is marked with 0 then
            DFSEXPLORE($w$)
```

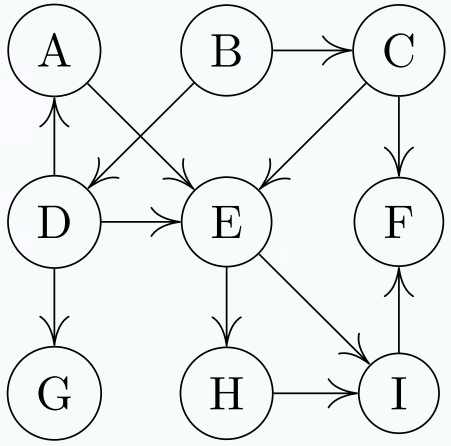
Problem: *Modify the algorithm so that it also builds the arrays* `pre[V]` *and* `post[V]`*to store the push- and the pop-order of the vertices.*



Group Work: problems T1 & T2

T1: *Finding a topological order for the graph by running a DFS.*

YOUR ANSWER:



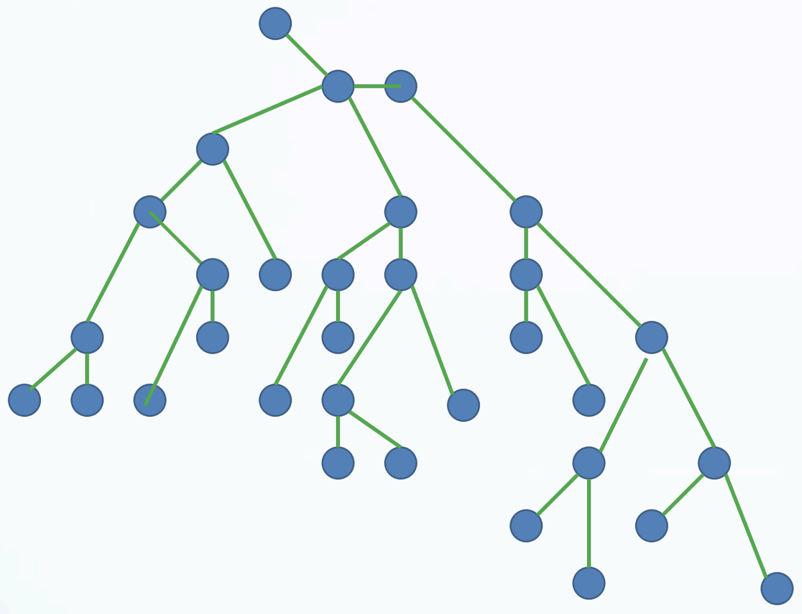
T2: *Dijkstra's algorithm, unmodified, can't handle some graphs with negative edge weights. Your friend has come up with a modified algorithm for finding shortest paths in a graph with negative edge weights:*

1. Find the largest negative edge weight, call this weight **−`w`**.
2. Add **`w`** to the weight of all edges in the graph. Now, all edges have non-negative weights.
3. Run Dijkstra's algorithm on the resulting non-negative-edge-weighted graph.
4. For each path found by Dijkstra's algorithm, compute its true cost by subtracting **`w`** from the weight of each of its edges.

a) Give an example showing that DA can't handle negative weights.

b) Will your friend's algorithm work? Give an example.

Trees as Special Graphs



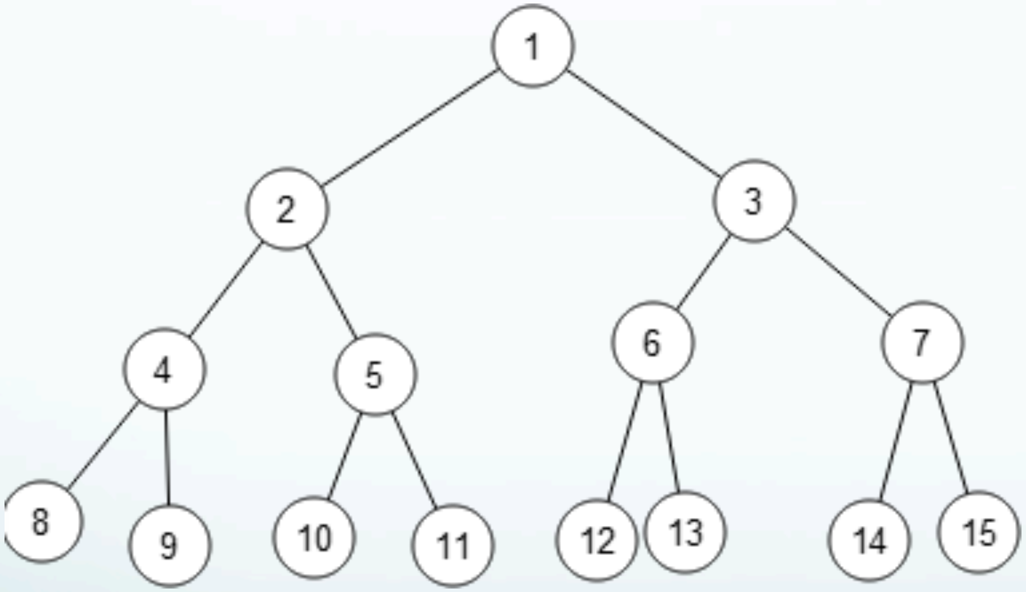
If a graph (V,E) is a tree, then:

-
-
-
-

Binary Tree: Recursive Definition

Binary tree traversal

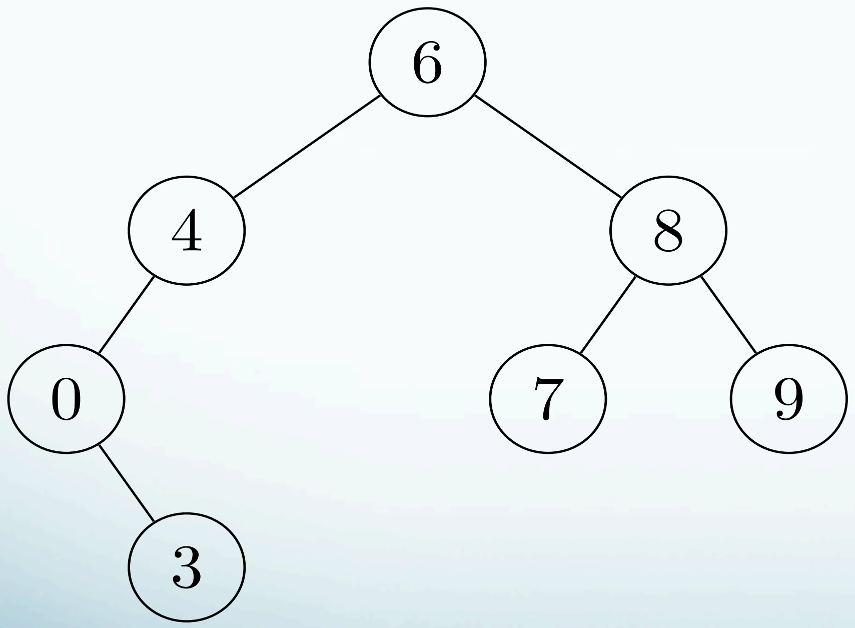
What is *inorder*, *preorder*, *postorder,* and *level-order* traversal? Are they BFS or DFS?



Group Work: Problems T3, T4, T5.

T3: conventional traversal

Write the *inorder*, *preorder* and *postorder* traversals of the following binary tree:



YOUR ANSWER:

In-order:

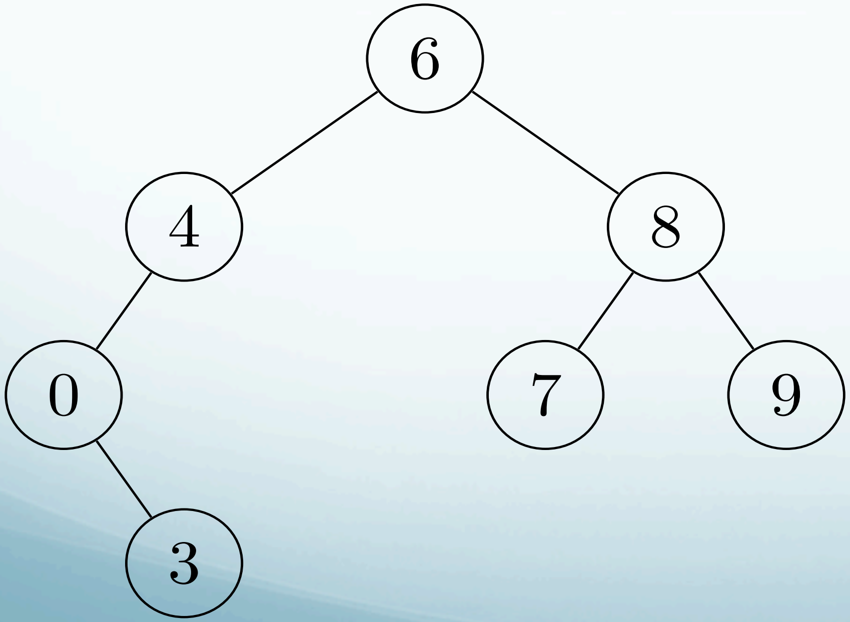
Pre-order:

Post-order:

T4: level-order traversal

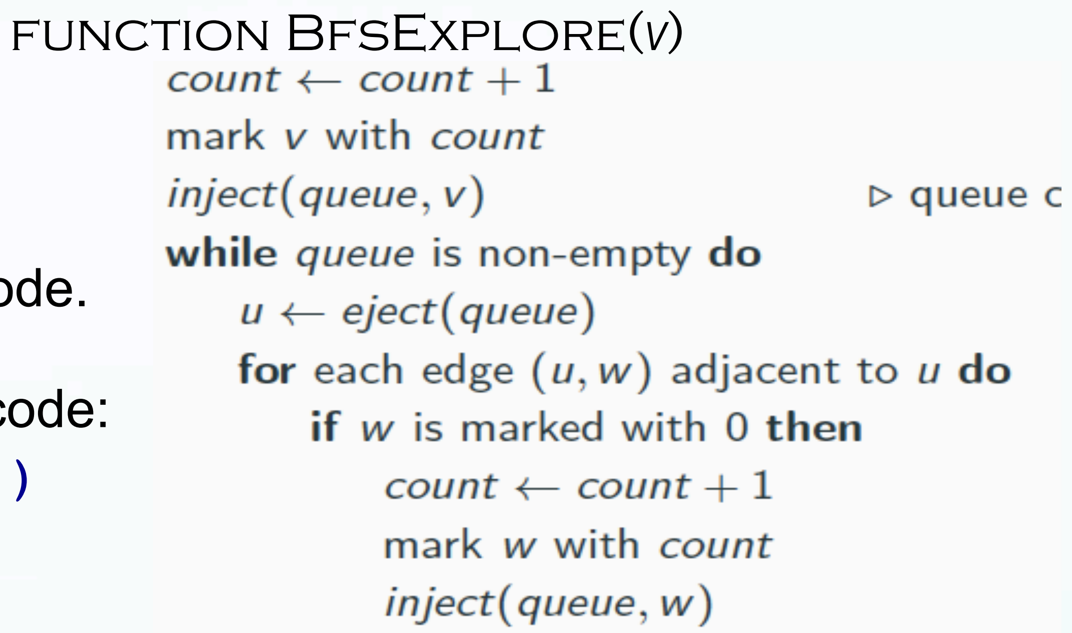
Level-order: visit level-by-level, left-to-right, starting from the root (which is in 0-th level).

a) For the tree below, what's the visited order?
b) Write the level-order pseudo-code.



YOUR ANSWER:
Level-order:

T4: level-order traversal



```
function BfsExplore($v$)
    $count \leftarrow count + 1$
    mark $v$ with $count$
    $inject(queue, v)$                              ▷ queue c
    while $queue$ is non-empty do
        $u \leftarrow eject(queue)$
        for each edge $(u, w)$ adjacent to $u$ do
            if $w$ is marked with 0 then
                $count \leftarrow count + 1$
                mark $w$ with $count$
                $inject(queue, w)$
```

b) Write the level-order pseudo-code.

YOUR ANSWER: The pseudocode:

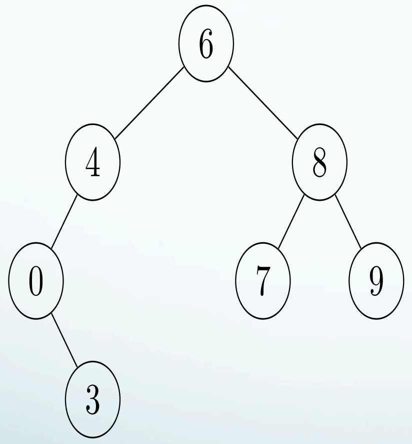
```
function LevelOrder( T )
```

T5: Binary Tree Sum

Write a recursive algorithm to calculate the sum of a binary tree where each node contains a number.

YOUR ANSWER: The pseudocode:

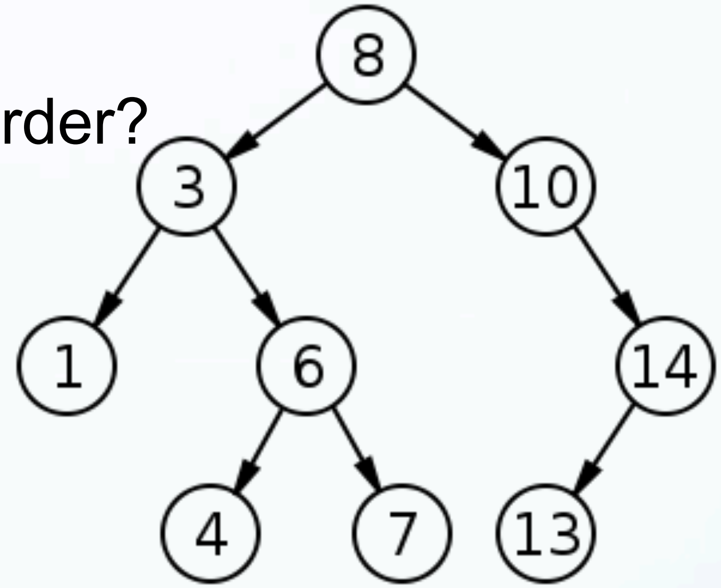
```
function Sum( T )
```



Binary (Search) Tree

How to:

- print the number in increasing order?
- print in decreasing order?
- copy the tree?
- free the tree?


8
3
10
1
6
14
4
7
13