Week 6 Workshop

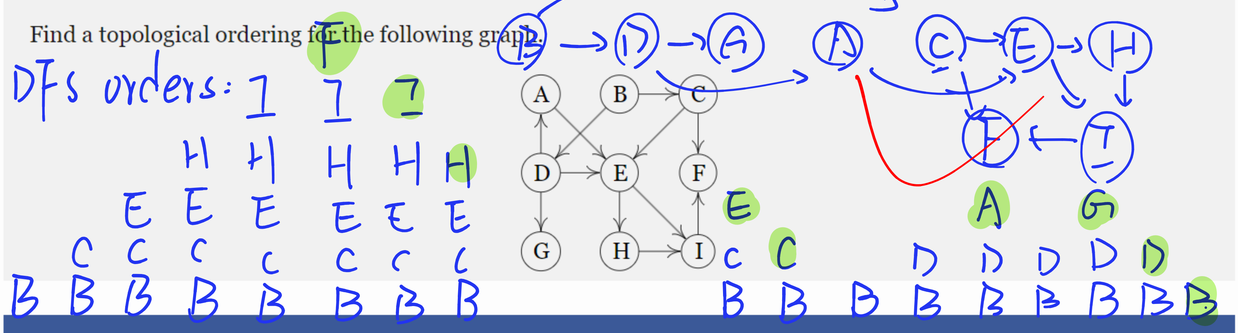
Tutorial

1. 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.

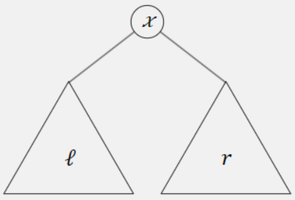
One of the algorithms discussed in lectures involves running a DFS on the DAG and keeping track of the order in which the vertices are popped from the stack. The topological ordering will be the reverse of this order.

Find a topological ordering for the following graph:



2. Binary tree traversals

When traversing a binary tree we have a decision to make about order. If we represent a general binary tree as follows, where x is the root, and ℓ and r are the (possibly empty) left and right sub-trees respectively, we can discuss different traversal orders more concretely.



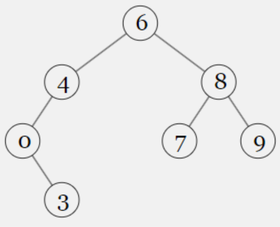
Now, we can define the traversal orders like so:

Inorder: traverse ℓ, visit x, traverse r

Preorder: visit x, traverse ℓ, traverse r

Postorder: traverse ℓ, traverse r, visit x

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



3. Level-order traversal of a binary tree

A level-order of a binary tree first visits the root, then the left child of the root, then the right child of the root, then the left child of the left child (the leftmost grandchild of the root) *etc.*, going left to right at each level.

In which order are the nodes of the binary tree from Question 2 visited in a level-order traversal.

Which graph traversal algorithm could we modify to perform a level-order traversal of a binary tree? Write the pseudo-code.

4. Binary tree sum

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

Confirm that your algorithm works on the tree from Question 2.

Which order does your algorithm process nodes in? *Inorder*, *preorder* or *postorder*?