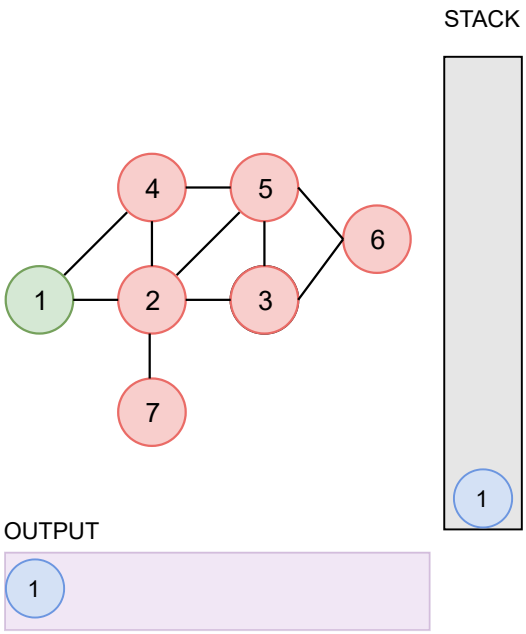
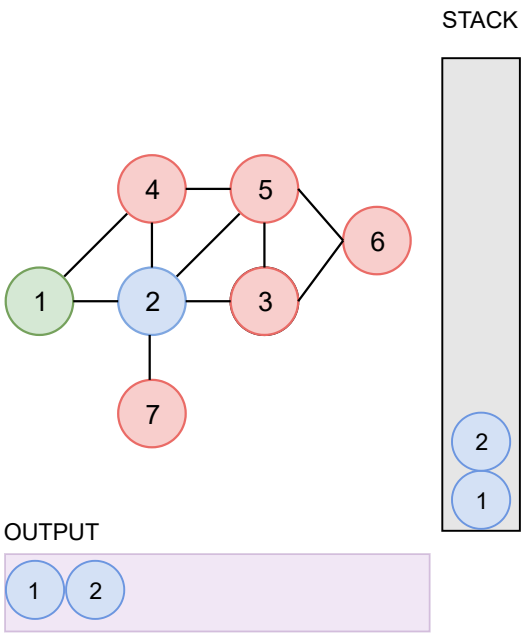
Graphs (Depth-first traversal)

The nodes are traversed from top to bottom. (Reading time: under 2 minutes)

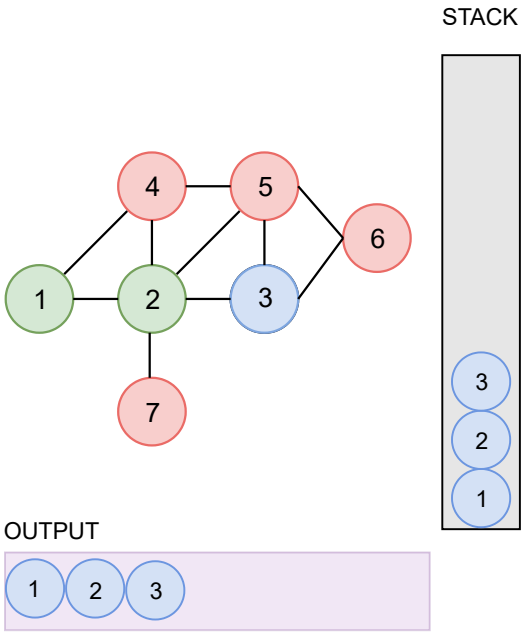
To traverse a graph depth-first, we need to use a **stack**. We manually pick one node to start with, as there is no specific root node like you have with a binary search tree, and go down each of the children nodes. In this example, **green nodes are marked as visited nodes, blue nodes are currently being visited**.



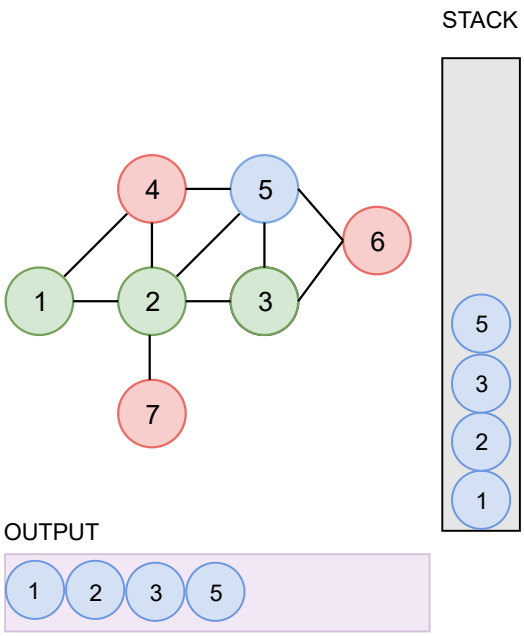
Let's say we used the node with the value 1 as our starting node. Right now, the stack contains the node with the value 1, and gets marked as visited as it gets added to the output sequence.



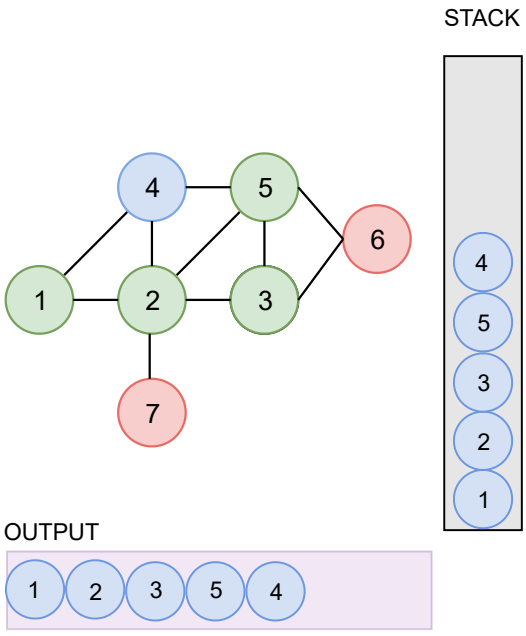
We go past its unvisited child nodes: 2 and 4. We choose the nodes numerically, and 2 is lower than 4, so we go to node 2. Node 2 is now added to the stack, pushed to the output sequence, and marked as visited.



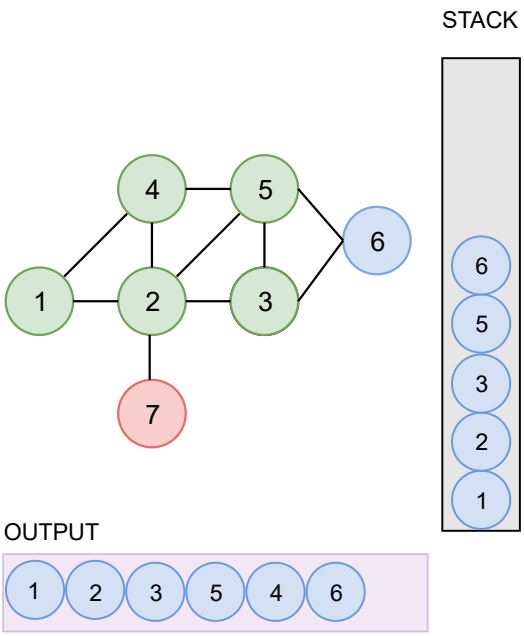
Node 2 has 3 unvisited child nodes: 7, 3 and 4. 3 is the smallest value, so the node with the value 3 gets added to the stack, pushed to the output sequence, and marked as visited.



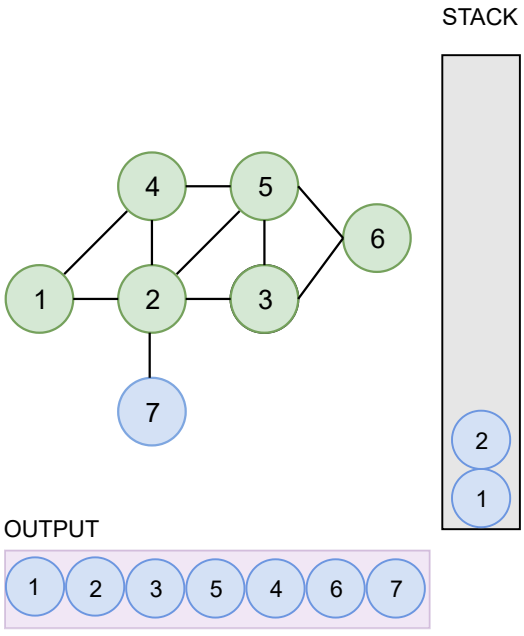
5 is the smallest unvisited node's value, so it gets pushed to stack and output sequence, and marked as visited.



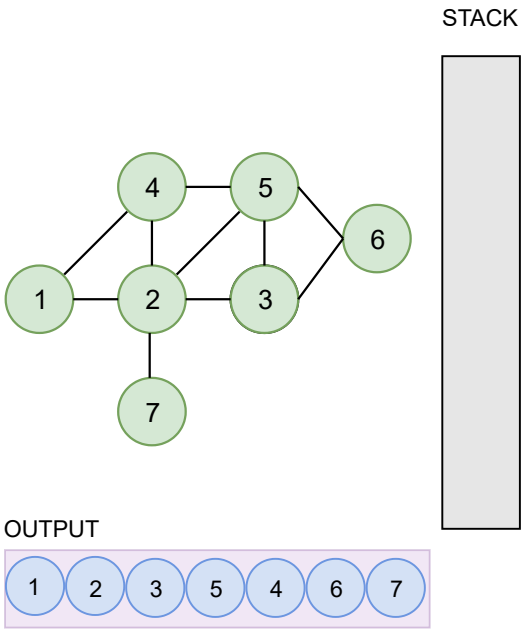
4 is the smallest unvisited node: it gets pushed to the stack and output array, and marked as visited. However, 4 doesn't have any unvisited child nodes! Now, the nodes get popped off the stack until it reaches a node that has unvisited child nodes.



After 4 has been popped off the stack, 5 is the node on top. 5 has an unvisited child node, node 6, so 6 gets added. This node again doesn't have any unvisited nodes. We pop nodes off the stack, until we find a node with unvisited child nodes, node 2!



The stack is empty: this is the sign that we have successfully traversed the entire graph!



The stack is empty: this is the sign that we have successfully traversed the entire graph!

In the next lesson, I will talk about the breadth-first traversal of a graph.