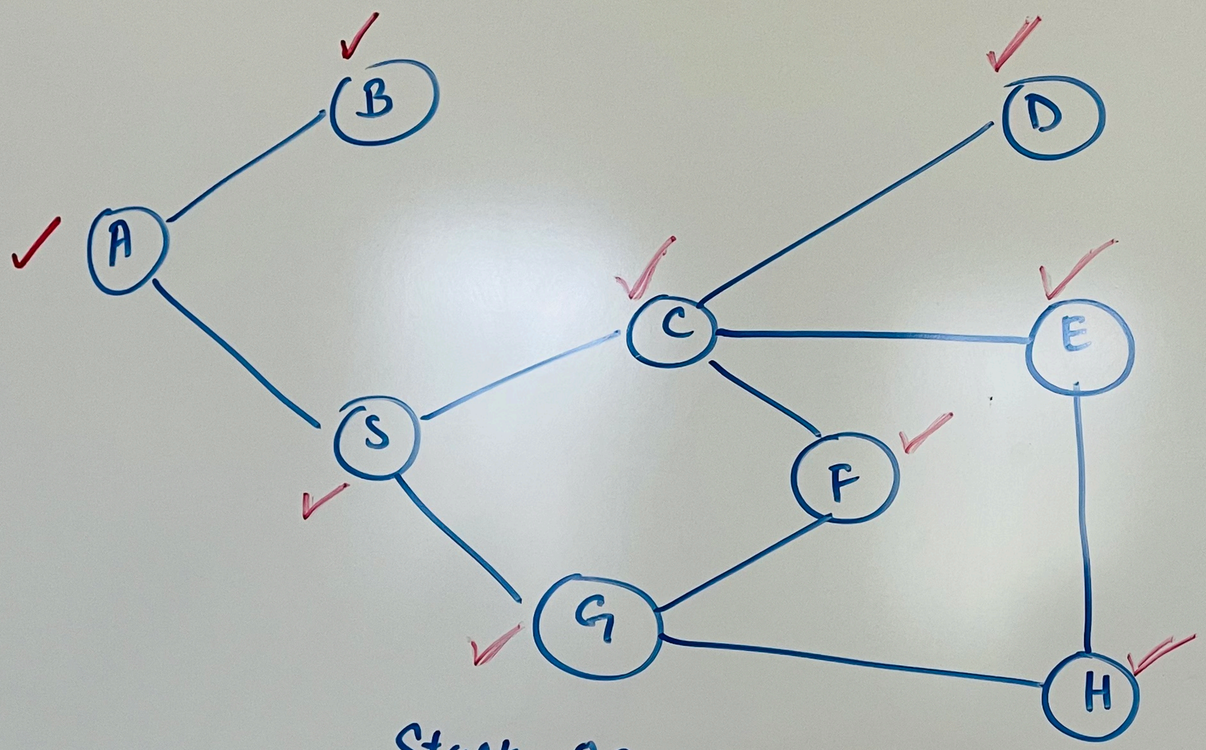
Depth First Search (DFS) → Stack (LIFO)

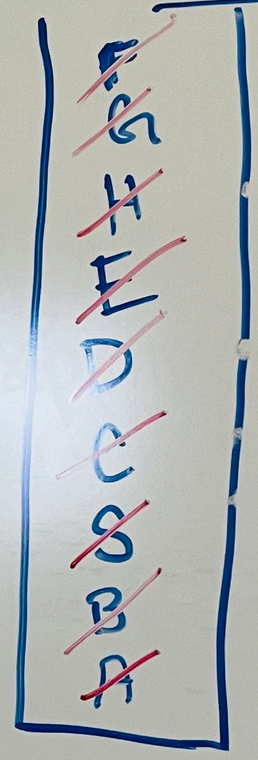


Stack Operations:

Push : inserting

Pop : removing

Stack Status



Output: A B S C D E H G F

k Status

Our starting node is A. We will push A to the top of the stack, add it to output, and mark as visited.

According to DFS, we will check the top of the stack (A), and then explore all of the adjacent, unvisited nodes. We prioritize depth over its siblings on the same level. We have B and S.

We will push B to the top of the stack, add it to output, and mark as visited. We will explore all adjacent, unvisited nodes B (prioritize depth). Since there aren't any, we will pop B from the stack. Now, we have A on top of the stack. We will move next to S, push S to the top of the stack, add it to output, and mark as visited.

CDEHGF

Now S is on top of the stack. As adjacent, unvisited nodes, we have C and G. We will push C to the stack, add it to output, and mark as visited. Then, C has D as adjacent unvisited node. We will push D to the stack, add it to output, and mark as visited.

Now beco

Next, we

Now E

Now,

visited

And

Now

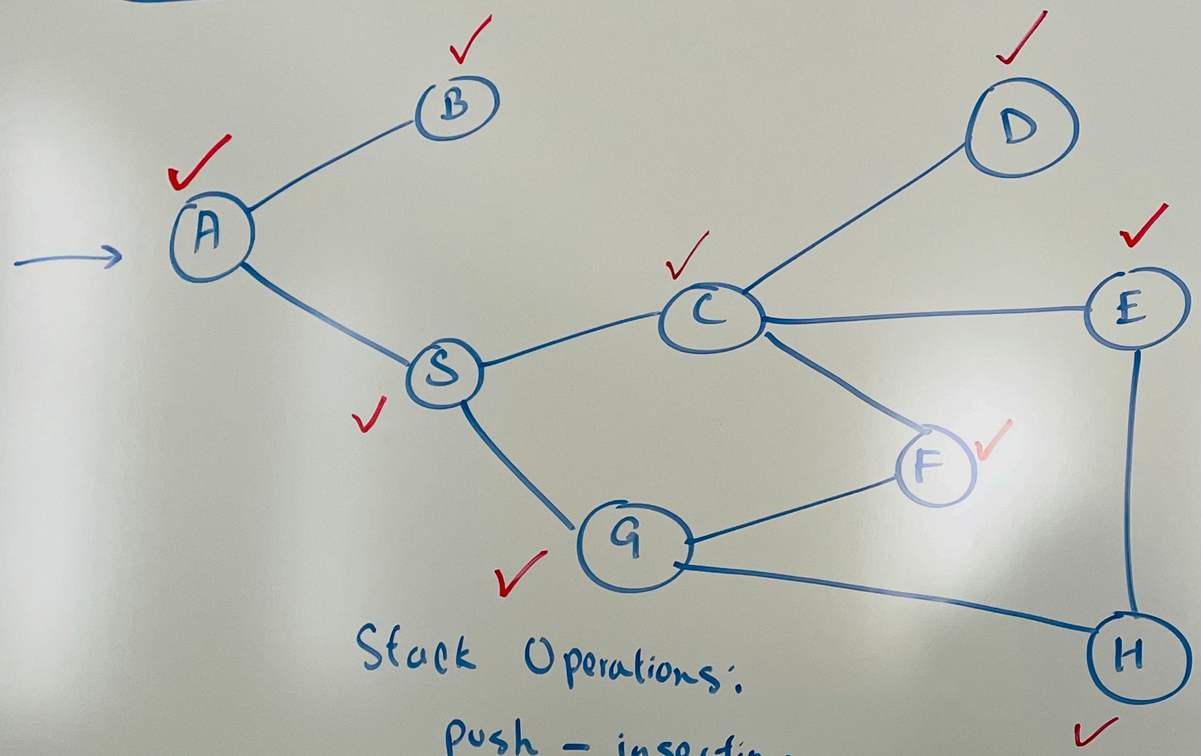
So

Si

Now because D doesn't have any adjacent unvisited node, we pop D.
Next, we will push E, add it to output, and mark as visited.
Now E has H. We will push H, add it to output, and mark as visited.
Now, H has G. We will push G, add it to output, and mark as visited. After that, G has F. We will push F, add it to output, and mark as visited.

Now all of F's adjacent unvisited nodes have been explored.
So we will pop F, G, H, E, C, S, and A one after another.
Since stack is empty, the algorithm terminates.

Depth-First Search (DFS) → Stack (LIFO)

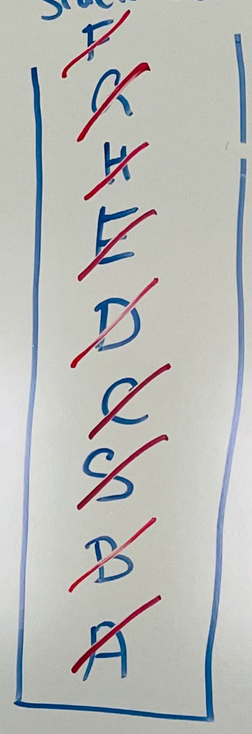


Stack Operations:

Push - inserting

Pop - removing

Stack Status



Output: A B S C D E H G F

Our start
Add it
According
And the
und
add

Now,
have
pop
sta
ad
No
C
T

FO)

ack Status

F
R
H
E
D
C
S

A BSCDEHGF

Our starting node is A. We will push A to the stack, add it to output, and mark as visited.

According to DFS, we will check the top of the stack (A) and then all of its adjacent, unvisited node. We have B and S. Following alphabetical order, we will push B, add it to output, and mark as visited.

Now, we will look at the top of the stack (B). Does B have any adjacent, unvisited Node? No, that's why we will pop B from the stack. Now we have A on top of the stack. S is the other adjacent unvisited node. We will push S, add it to output, and mark as visited.

Now S is on top of the stack. S has C and G. We will push C, add it to output, and mark as visited.

Then C has D, E and F. We will push D, add it to output, and mark as visited.

Since D doesn't have any adjacent, unvisited node, we will pop D.

Next, we will push E, add it to output, and mark as visited.

Now, E has H. So we will push H, add it to output, and mark as visited. Then H has G. We will push G to the stack, add it to output, and mark as visited.

Lastly, G has F. We will push F, add it to output, and mark as visited.

Now, all of F's adjucent unvisited nodes have been explored. So, we will pop F, G, H, E, C, S and A. Since stack is empty, the algorithm terminates.

DFS Flowchart

① Push starting node to stack, add to output, and mark it as visited.

② Does the top of the stack have any adjacent, unvisited node? If yes, select alphabetically, Push node, add to output, and mark as visited.

③ After that, rather than moving to its sibling in the same level, we will prioritize depth. We will check if the top node of the stack has any adjacent, unvisited node, and select alphabetically.

④ Once we have explored the full extent of depth, and there are no more nodes where the pointer can go, we will backtrack, and explore the first sibling on its way.

⑤ Once all adjacent nodes of a node is explored, we will pop the node and repeat the process for the node on top of the stack. Once all the nodes have been popped and the stack is empty, the algorithm terminates.

DFS Flowchart

① Push starting node to stack, add to output, and mark it as visited.

② Does the top node of the stack have any adjacent unvisited node? If yes, we select alphabetically. We push node, add it to output, and mark as visited.

③ After that, rather than turning to its sibling in the same level, we will prioritize depth. We will check if the top node of the stack has any unvisited node, and select alphabetically.

④ Once we have explored the full extent of depth, and there are no more nodes where the pointer can go to, we will backtrack and explore the first sibling on its way.

⑤ Once all adjacent nodes of a node is explored, we will pop the node and repeat the process for the top of the stack. Once all nodes have been popped and the stack is empty, the algo terminates.