Introduction to Computational Thinking and Data Science

Recitation Hour #02: **Dynamic programming & Graph Theory**

Somaia Zabihi

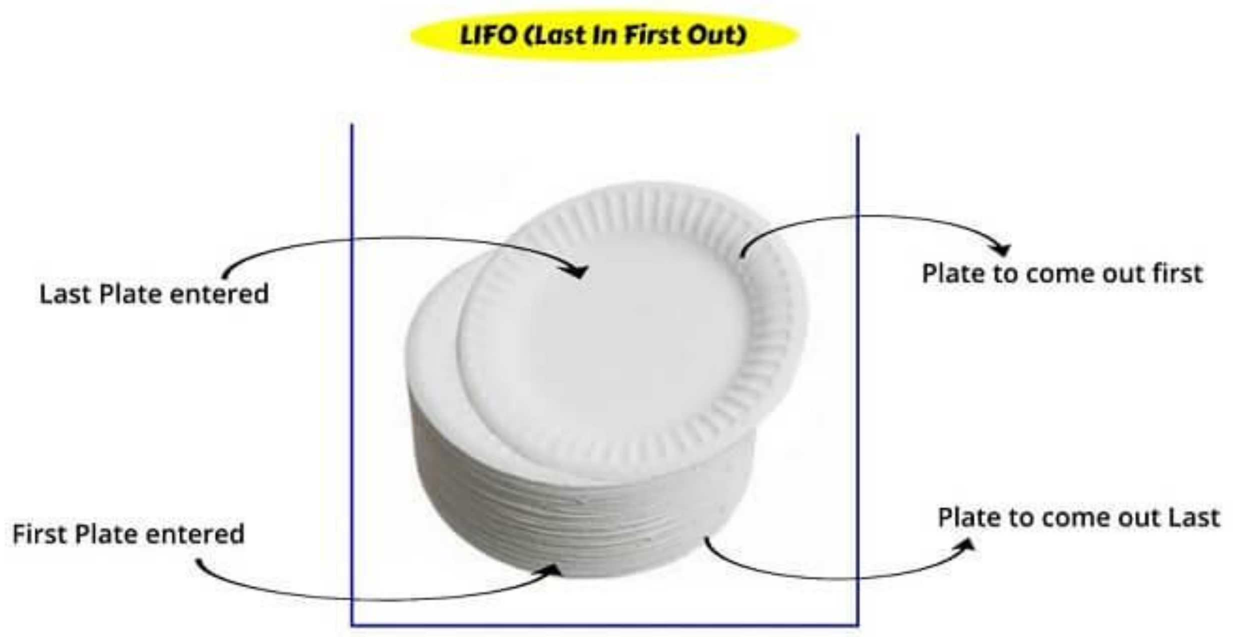
BFS & DFS

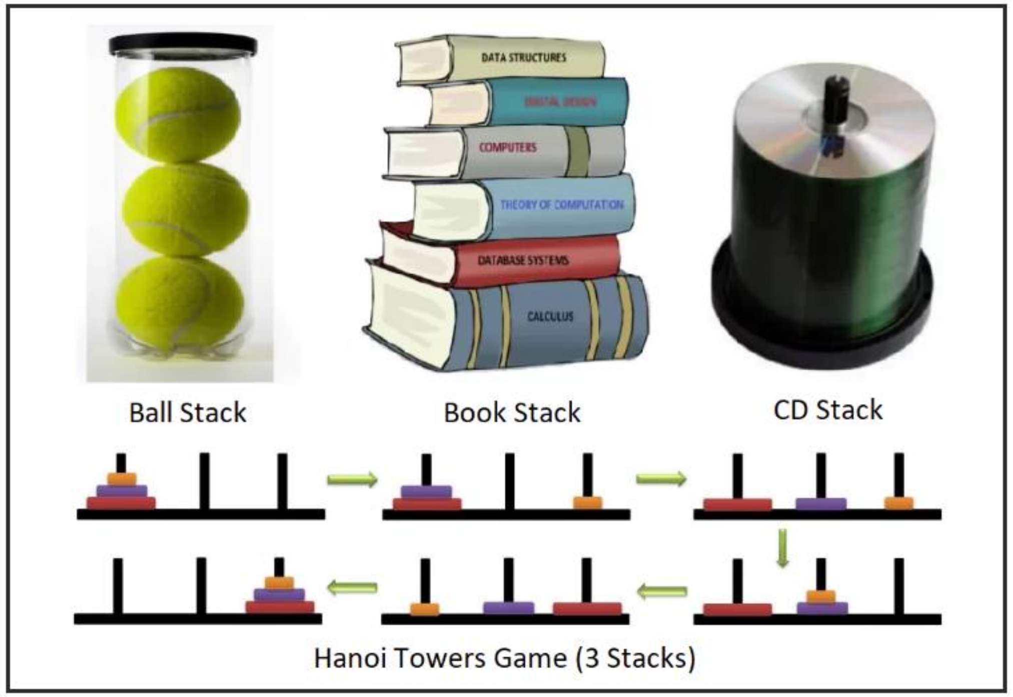
- BFS (Breadth-First Search) and DFS (Depth-First Search) are two fundamental **graph traversal** algorithms used in graph theory and computer science to explore and search through the nodes of a graph or a tree.
- The choice between BFS and DFS depends on the specific problem you are trying to solve.
- Use BFS when you need to find the **shortest path** or explore nodes level by level, and use DFS when you need to explore deeply into a graph or solve certain types of **problems** like finding cycles or paths.
- Both algorithms have their advantages and limitations, and the choice of which to use will depend on the context and requirements of your graph-related task.

Depth-First Search (DFS)

- DFS is an algorithm that starts at the root node and explores as far as possible along each branch before backtracking.
- It is often implemented using a **stack** data structure or recursion.
- DFS is useful for tasks like topological sorting, cycle detection, and finding paths, but it does **not** necessarily find the shortest path in an **unweighted graph**.

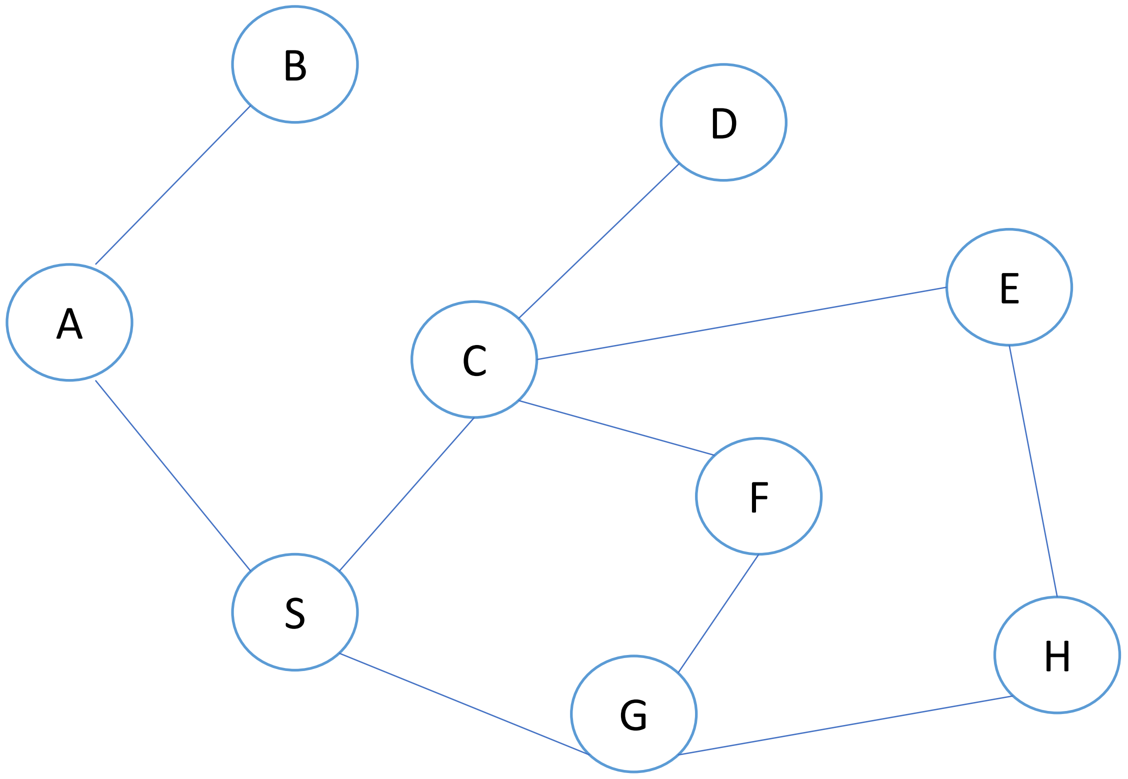
Stack Data Structure in Our Daily Life



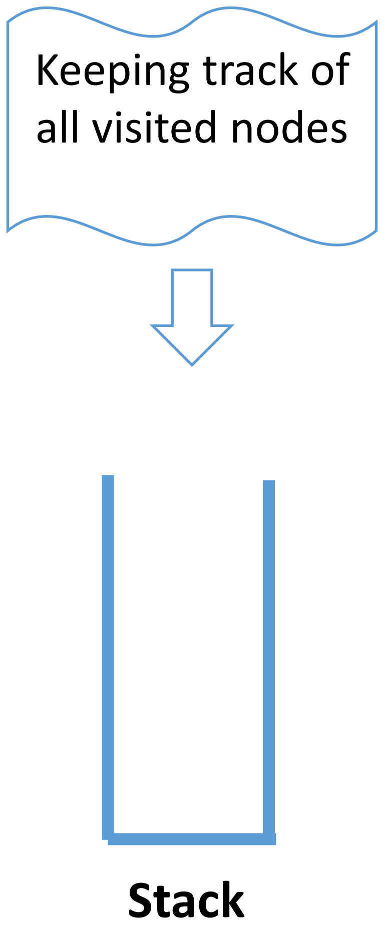


Depth First Search (DFS)

🟨 Visited node

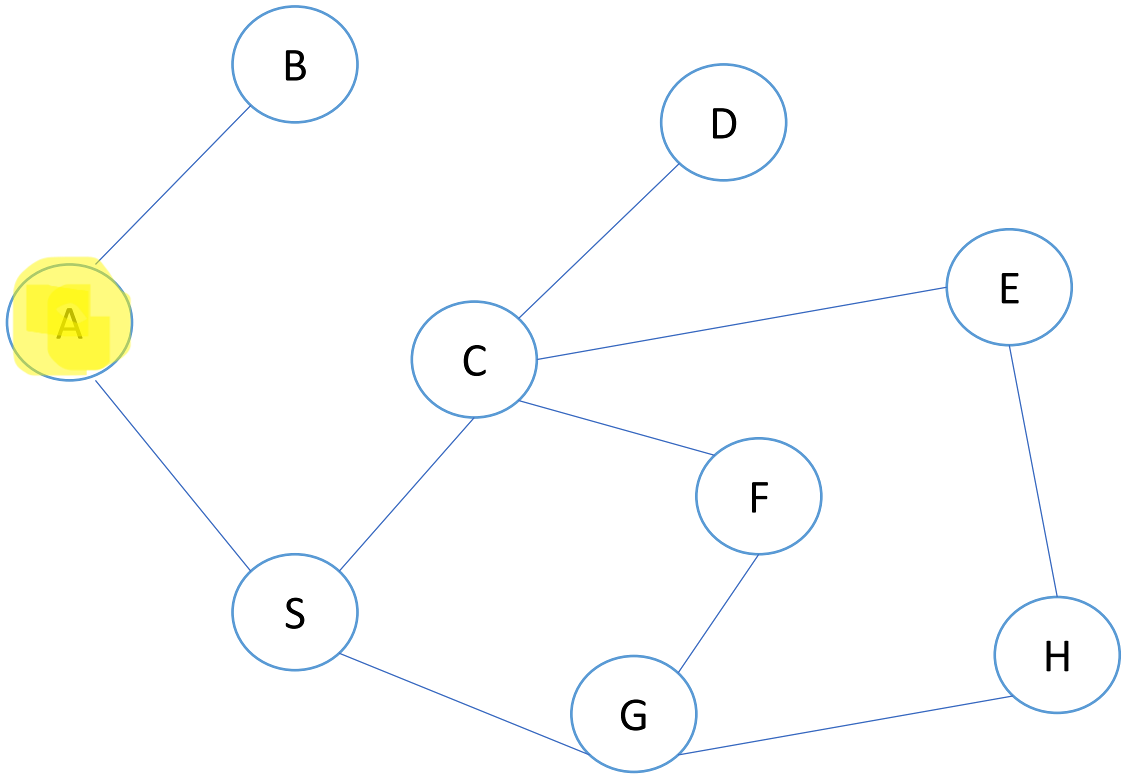


Output:



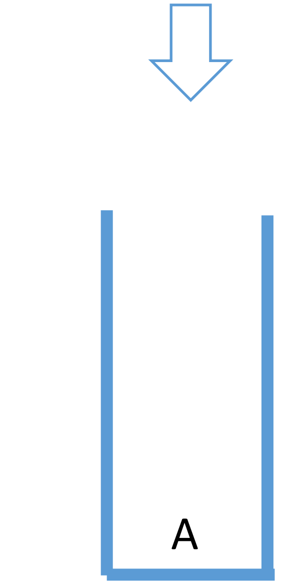
Depth First Search (DFS)

🟨 Visited node



Output: A

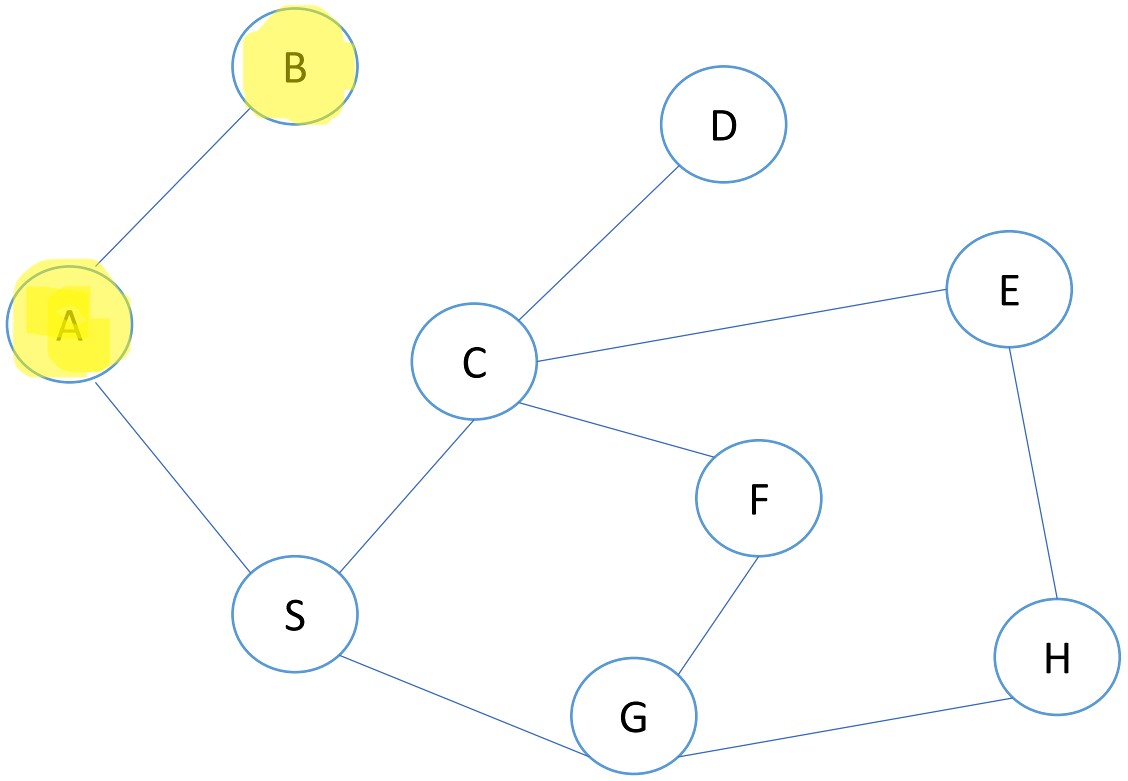
Keeping track of all visited nodes



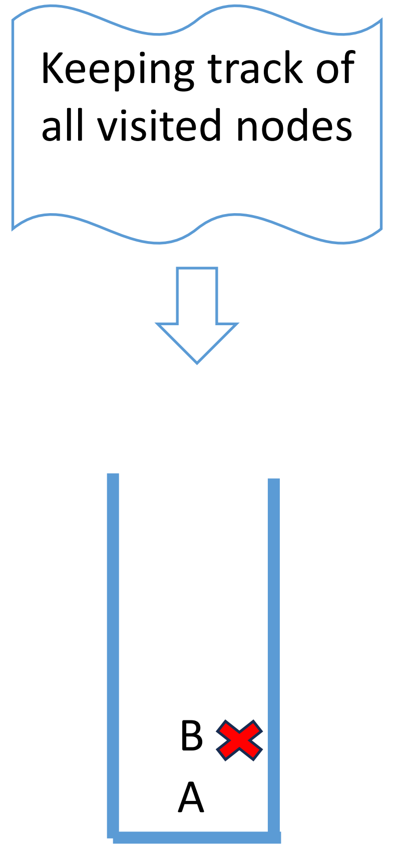
Stack

Depth First Search (DFS)

🟨 Visited node



Keeping track of all visited nodes

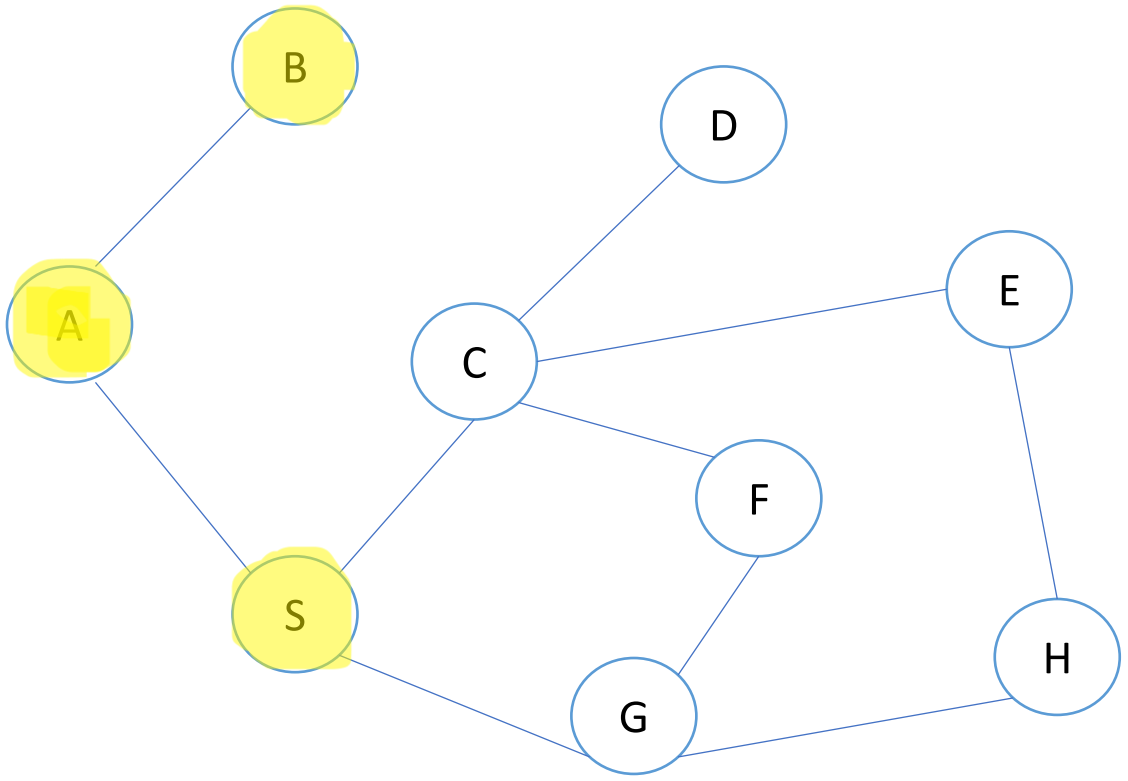


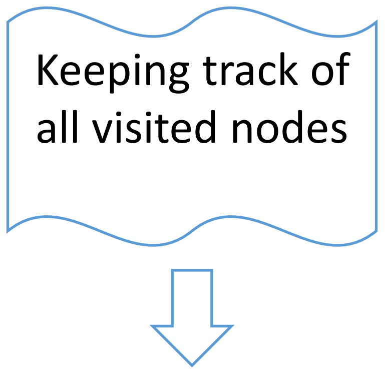
Stack

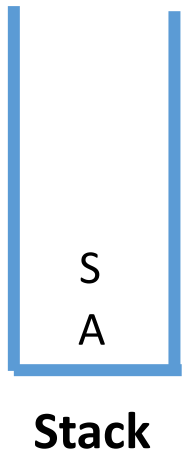
Output: A,B

Depth First Search (DFS)

🟨 Visited node



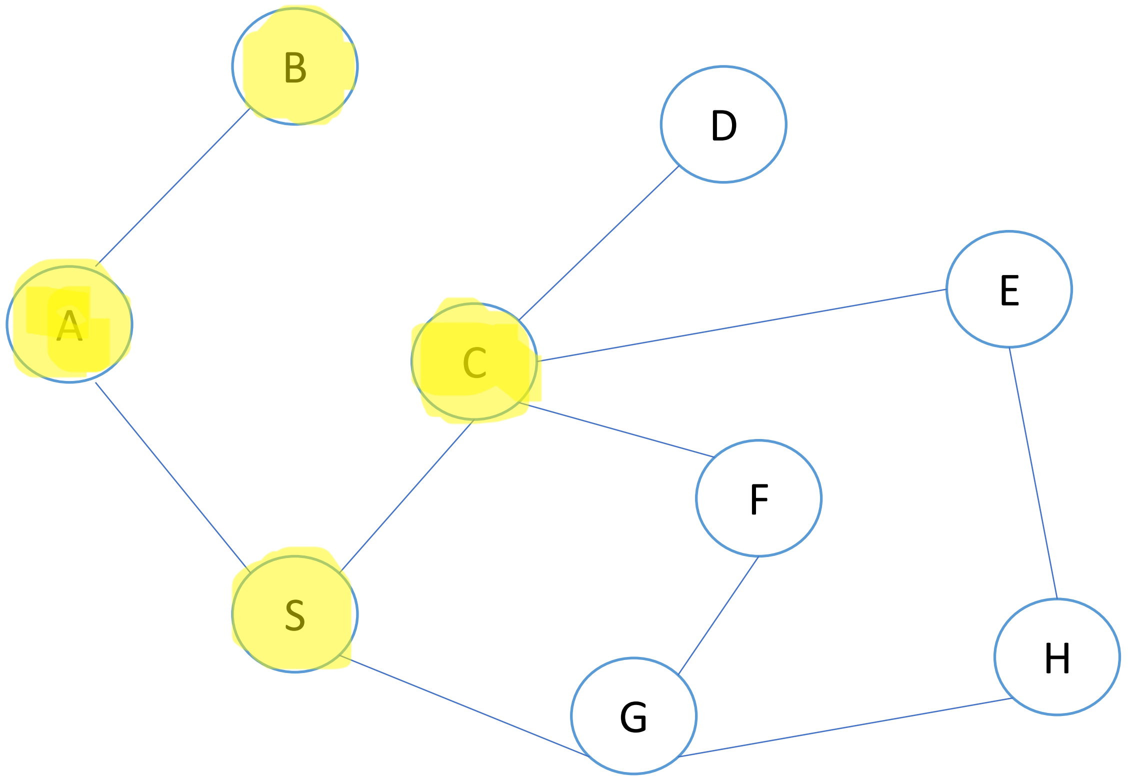




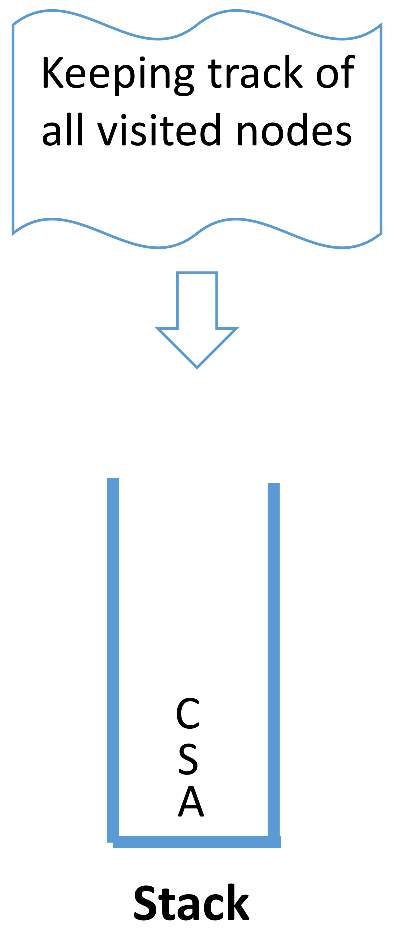
Output: A,B,S

Depth First Search (DFS)

🟨 Visited node

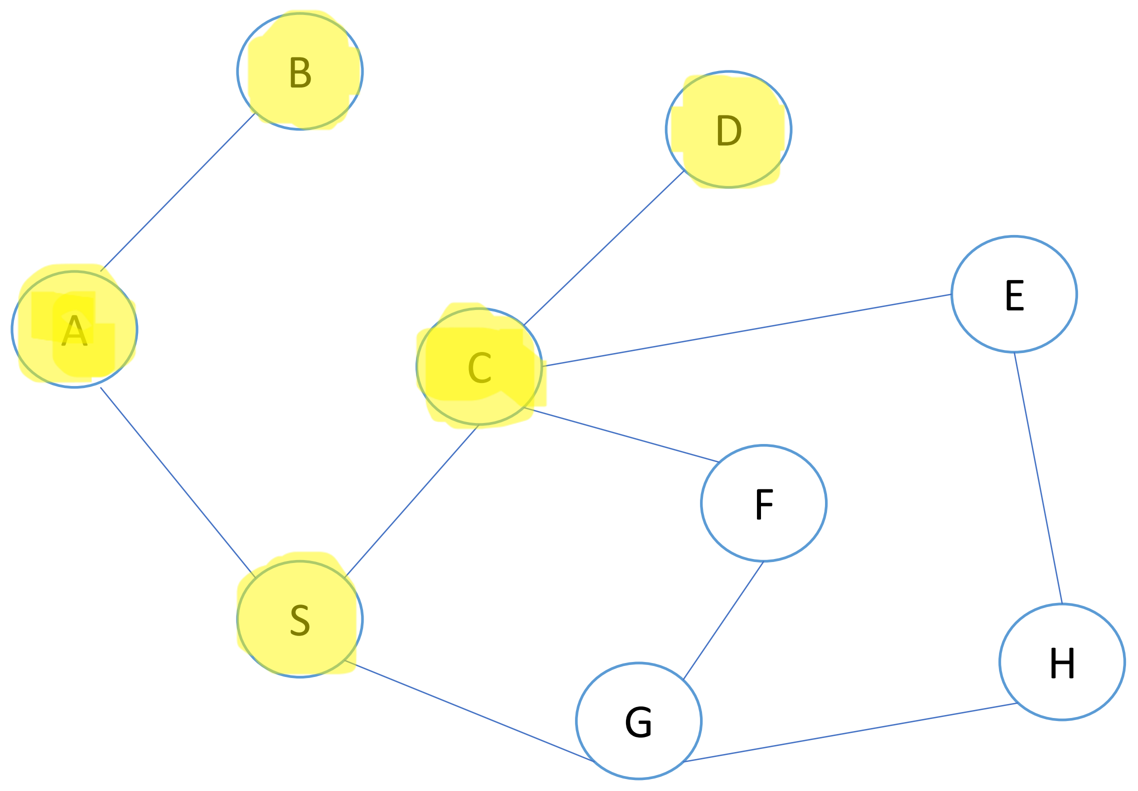


Output: A,B,S,C

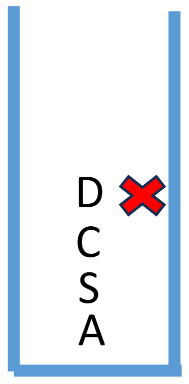


Depth First Search (DFS)

🟨 Visited node



Keeping track of all visited nodes

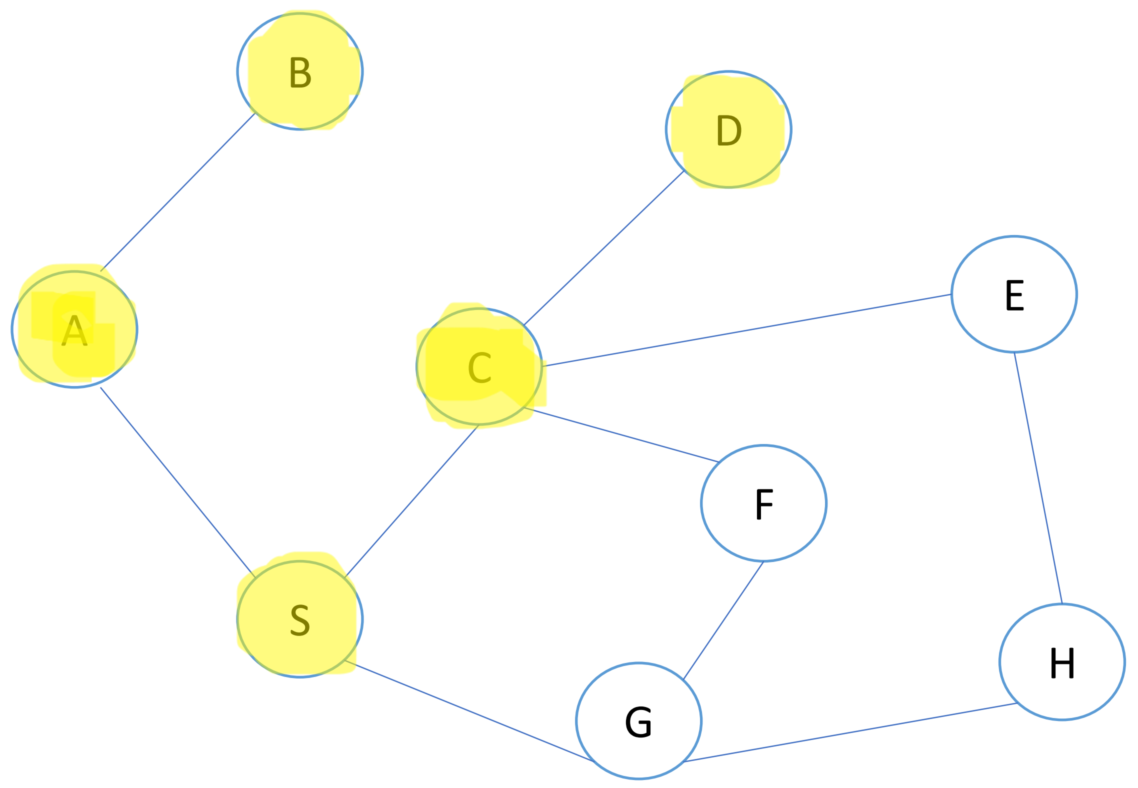


Stack

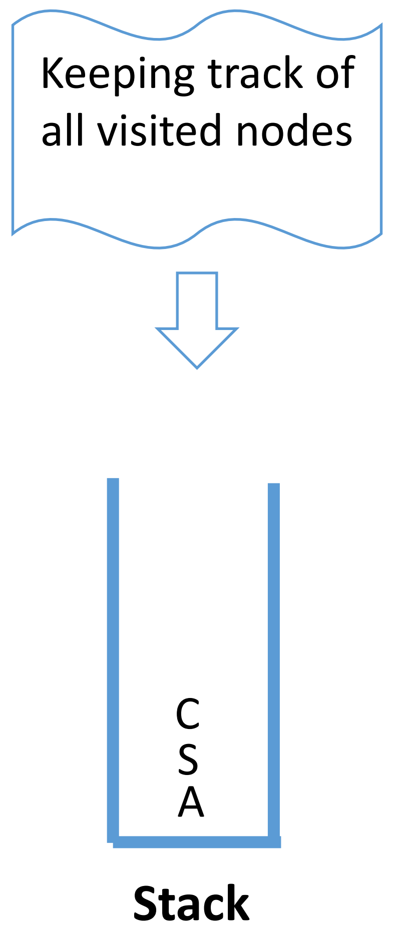
Output: A,B,S,C,D

Depth First Search (DFS)

🟨 Visited node

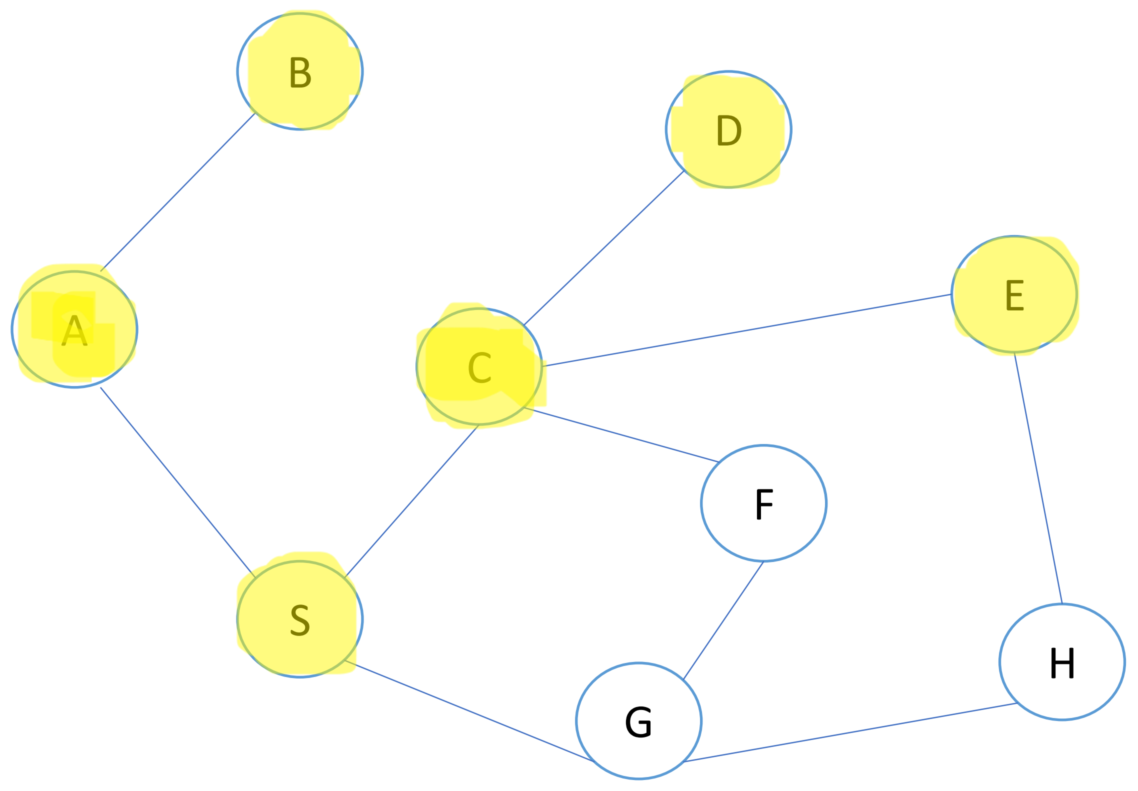


Output: A,B,S,C,D

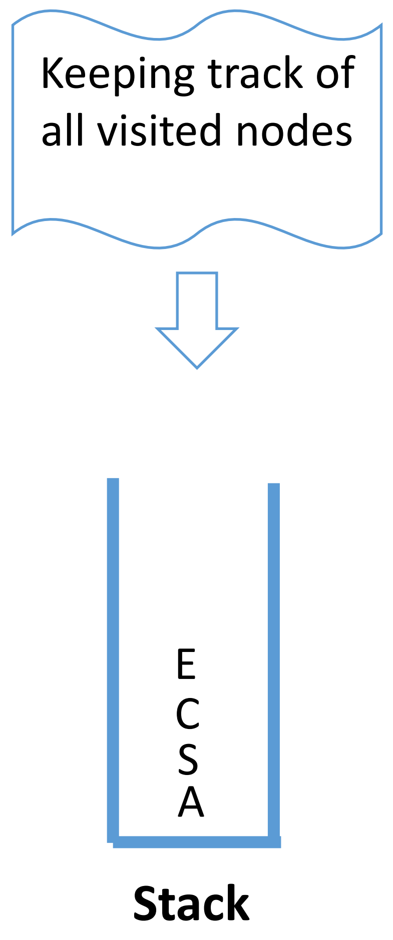


Depth First Search (DFS)

🟨 Visited node

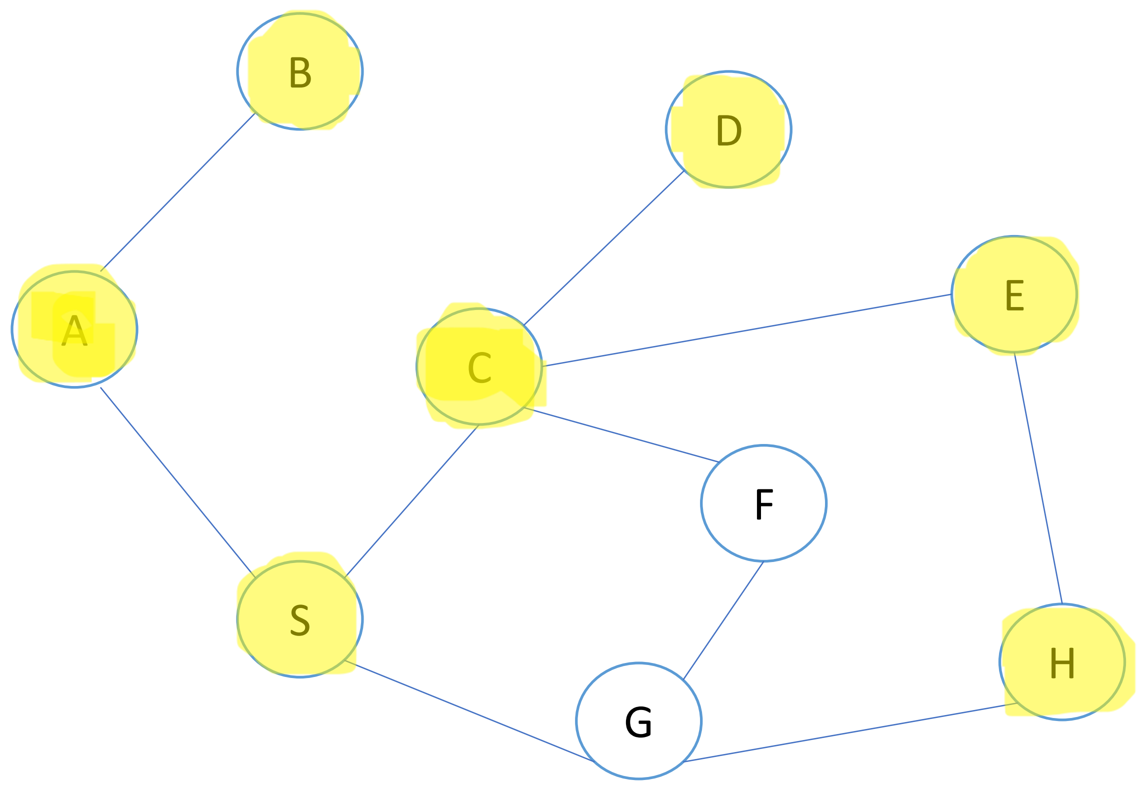


Output: A,B,S,C,D,E

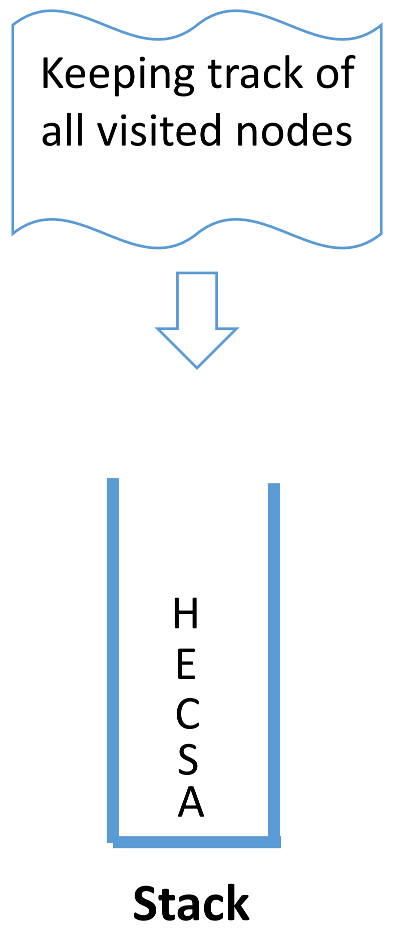


Depth First Search (DFS)

🟨 Visited node

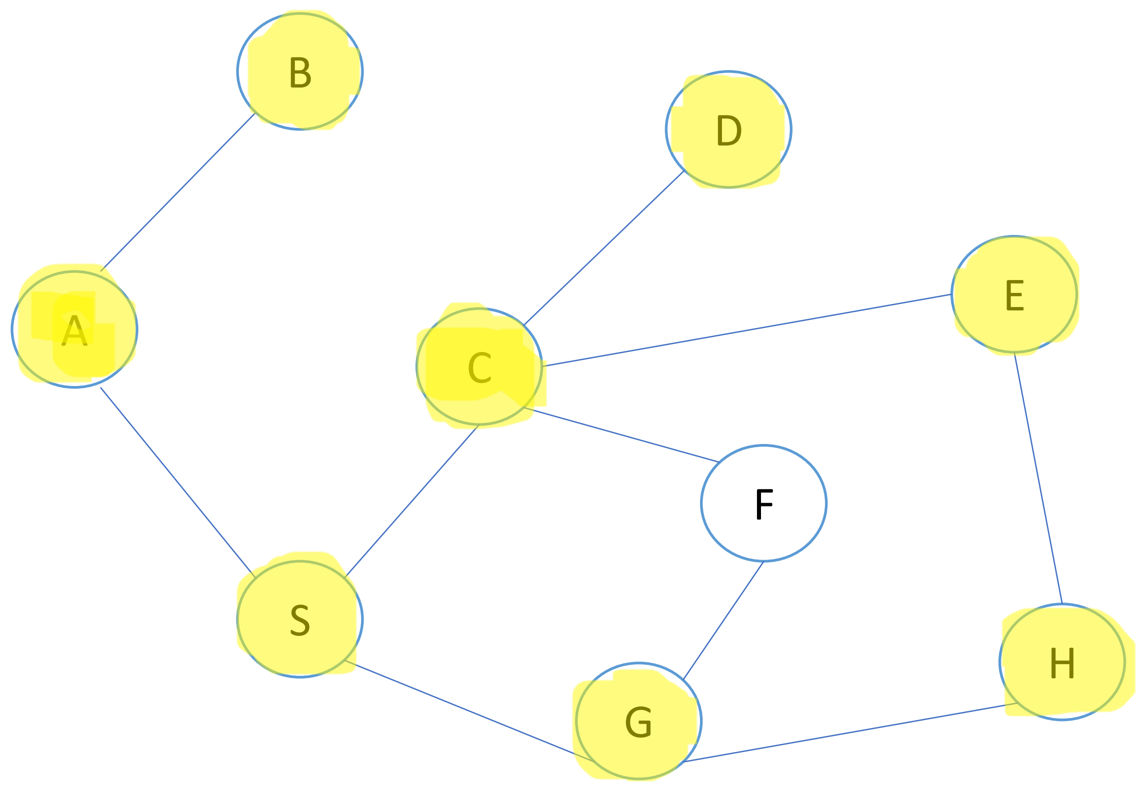


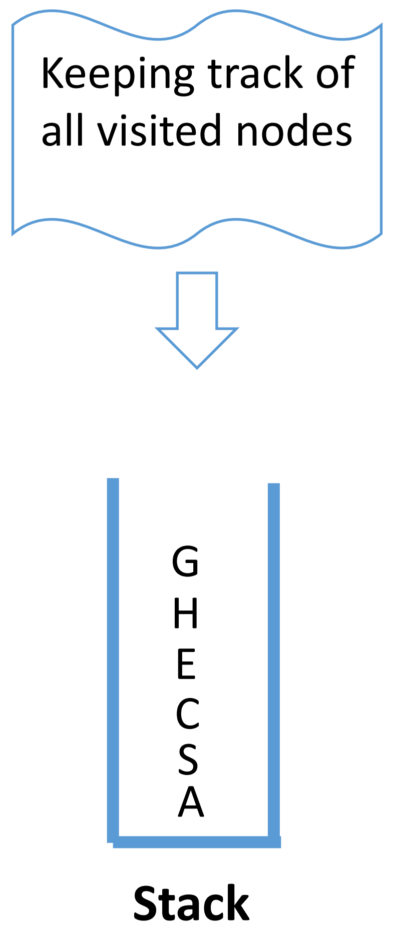
Output: A,B,S,C,D,E,H



Depth First Search (DFS)

🟨 Visited node

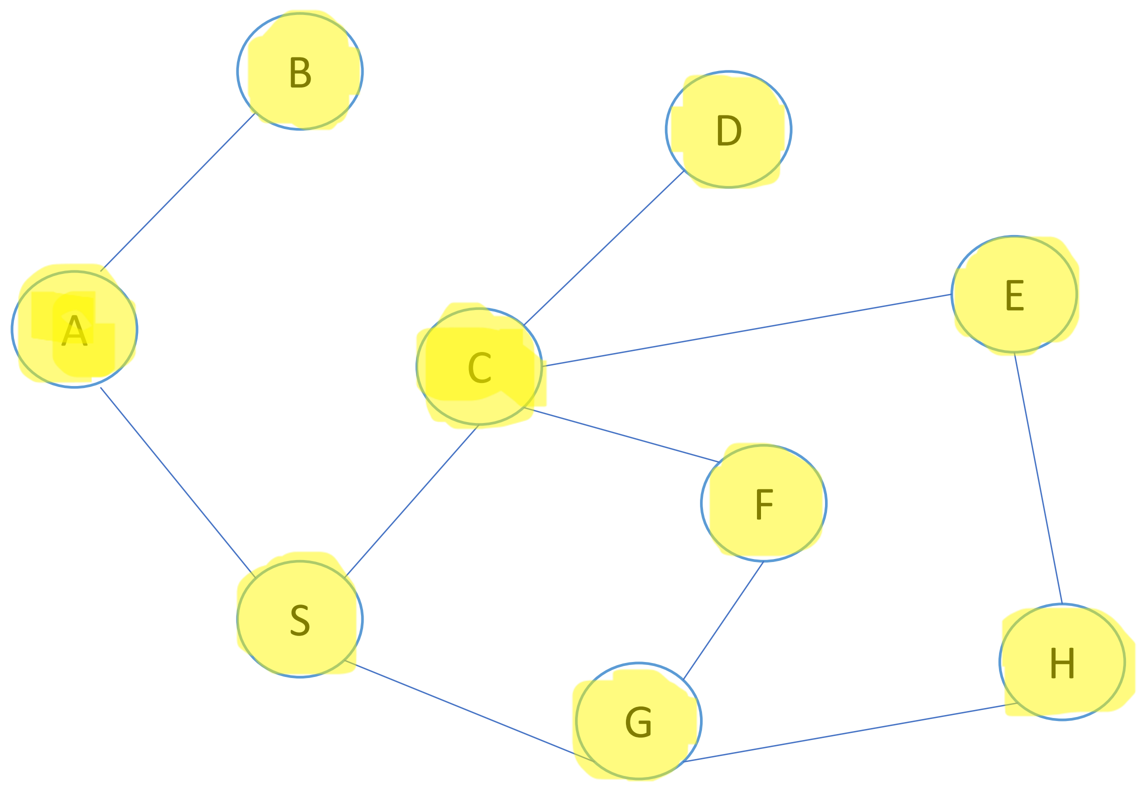




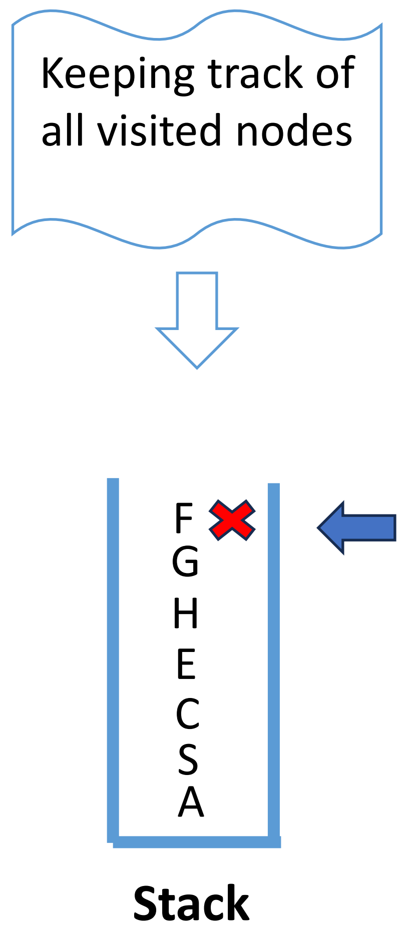
Output: A,B,S,C,D,E,H,G

Depth First Search (DFS)

🟨 Visited node

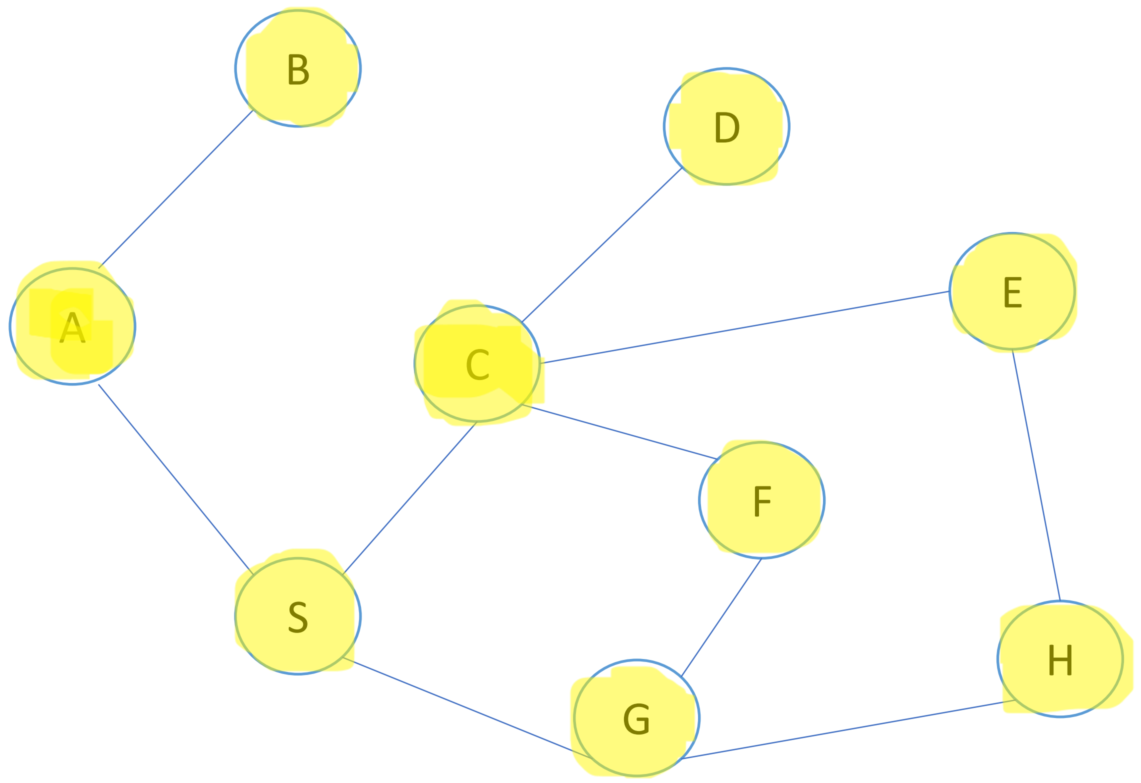


Output: A,B,S,C,D,E,H,G,F

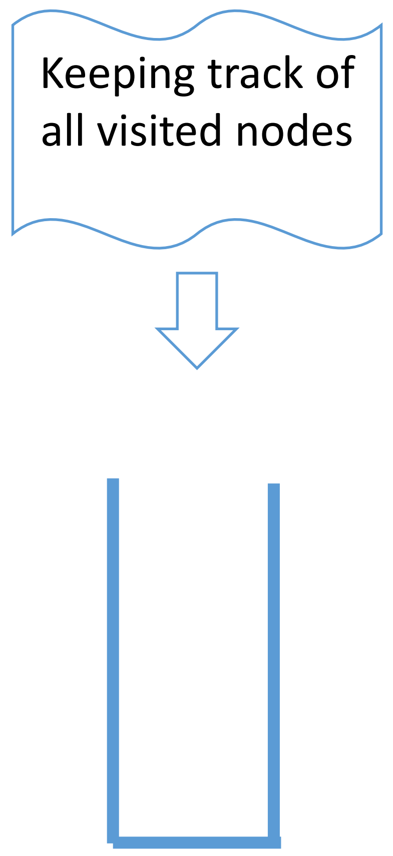


Depth First Search (DFS)

🟨 Visited node



Output: A,B,S,C,D,E,H,G,F

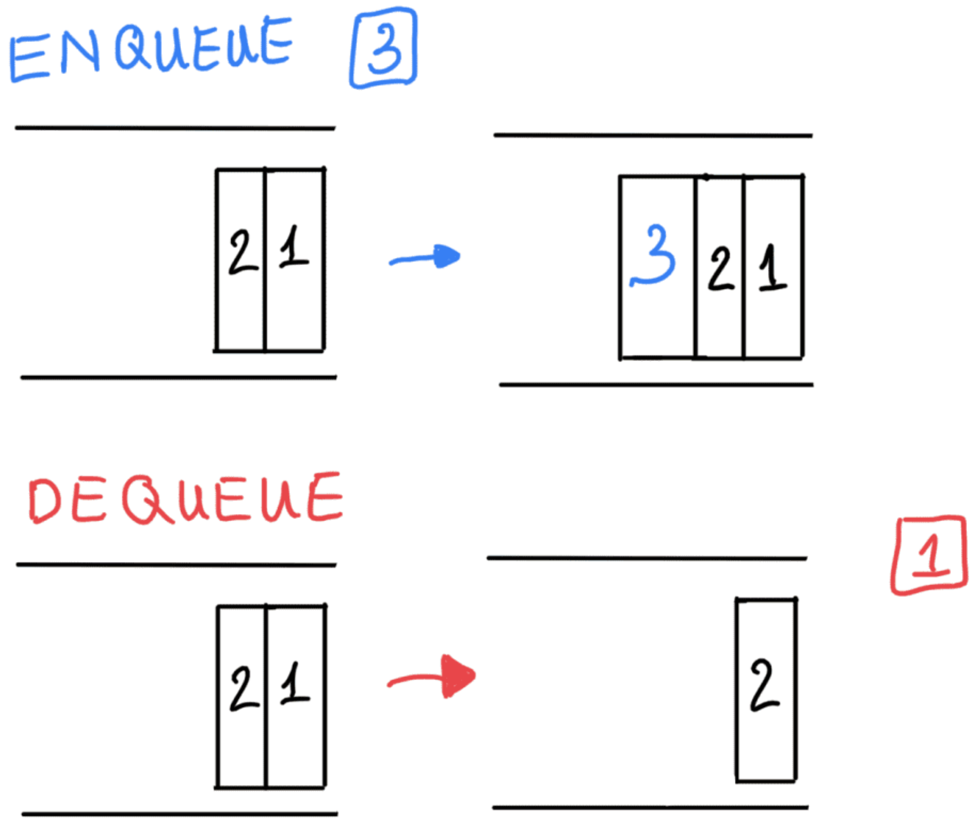


Stack

Breadth-First Search(BFS)

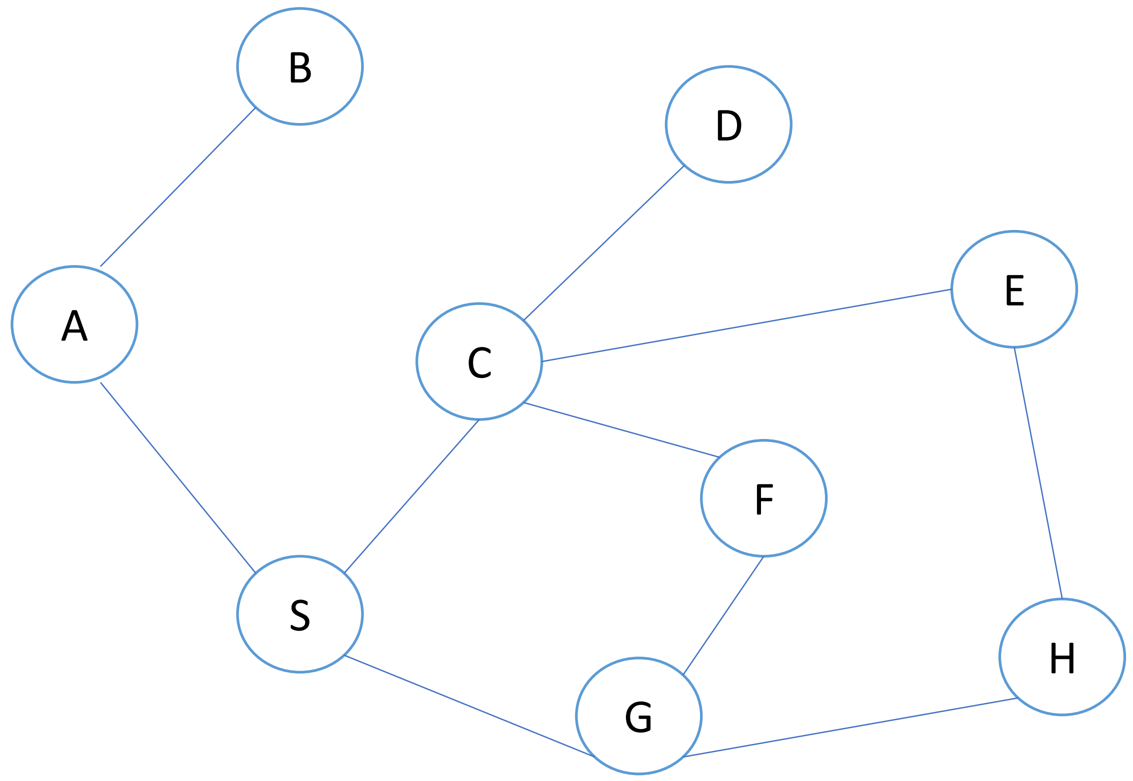
- BFS is an algorithm that starts at the root node (or any arbitrary node) and explores all the **neighbor nodes** at the **current depth level** before moving on to nodes at the next depth level.
- It is often implemented using a **queue** data structure.
- BFS is particularly useful for finding the **shortest path** between two nodes in an unweighted graph since it explores nodes level by level.

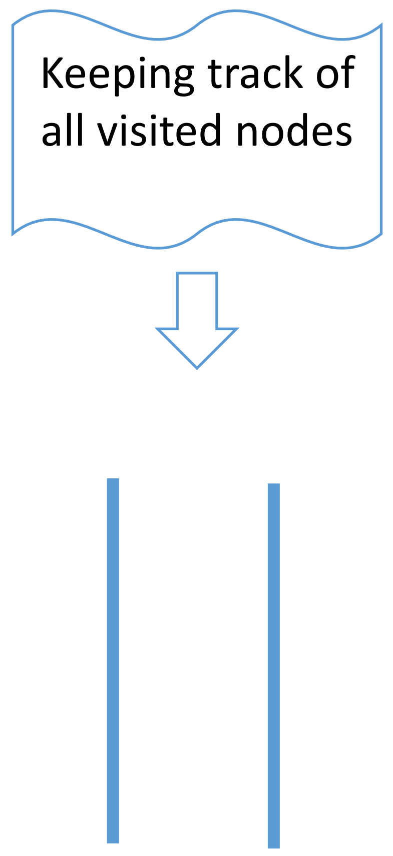
Queue Data Structure



Breadth First Search (BFS)

🟩 Visited node



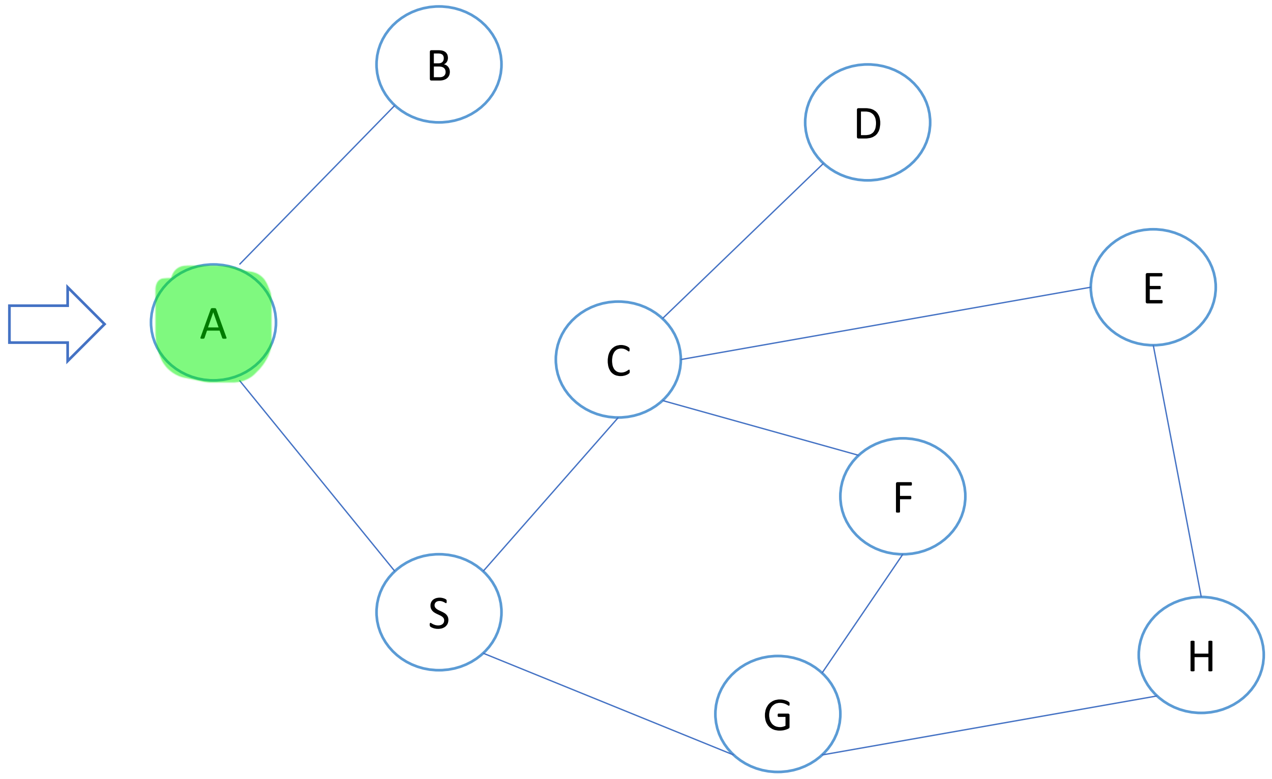


Queue

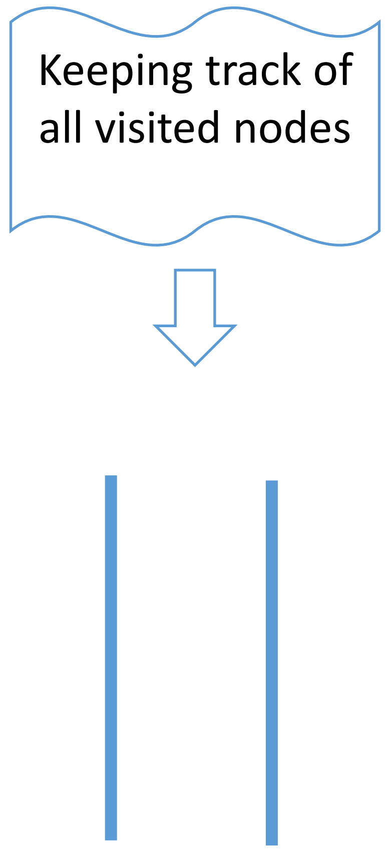
Output:

Breadth First Search (BFS)

🟩 Visited node

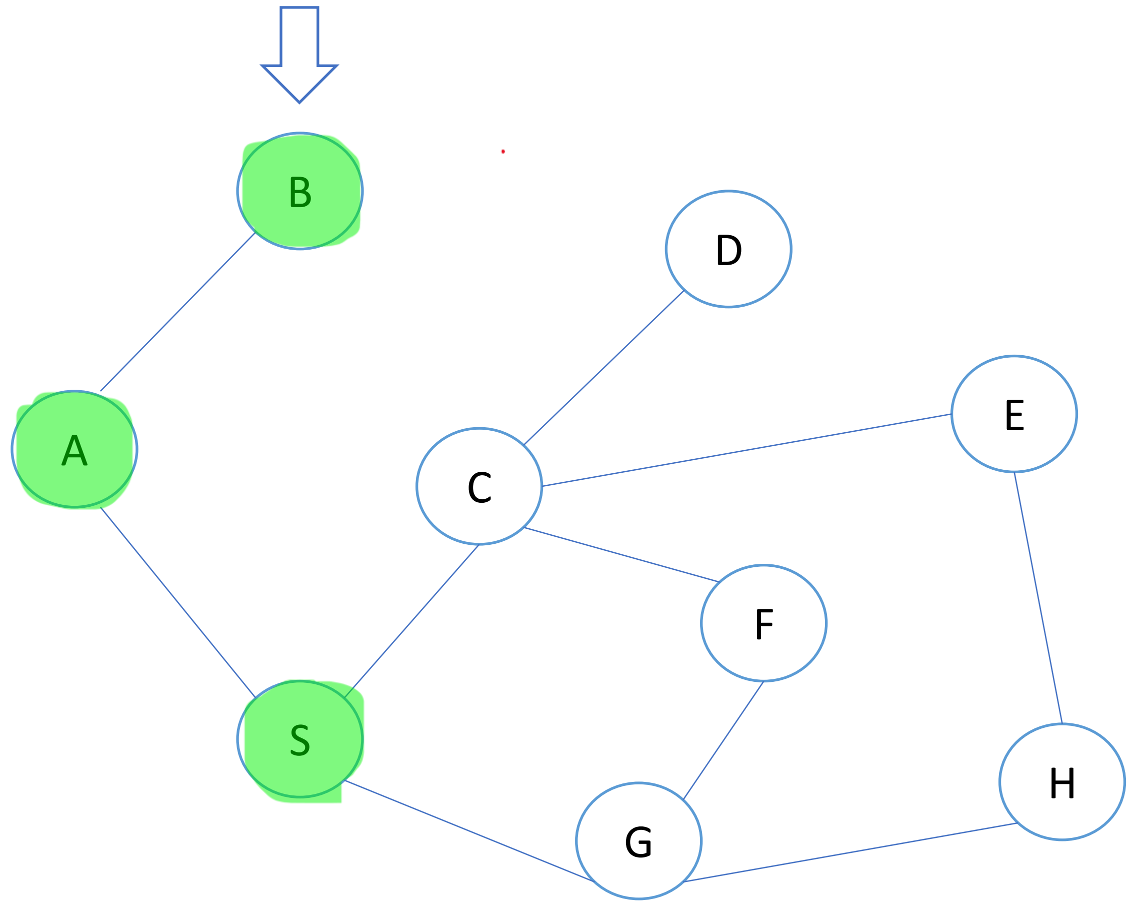


Output: A, B, S



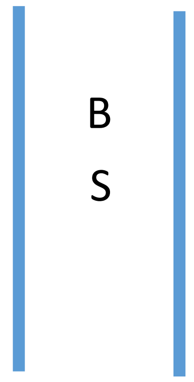
Queue

Breadth First Search (BFS)



🟩 Visited node

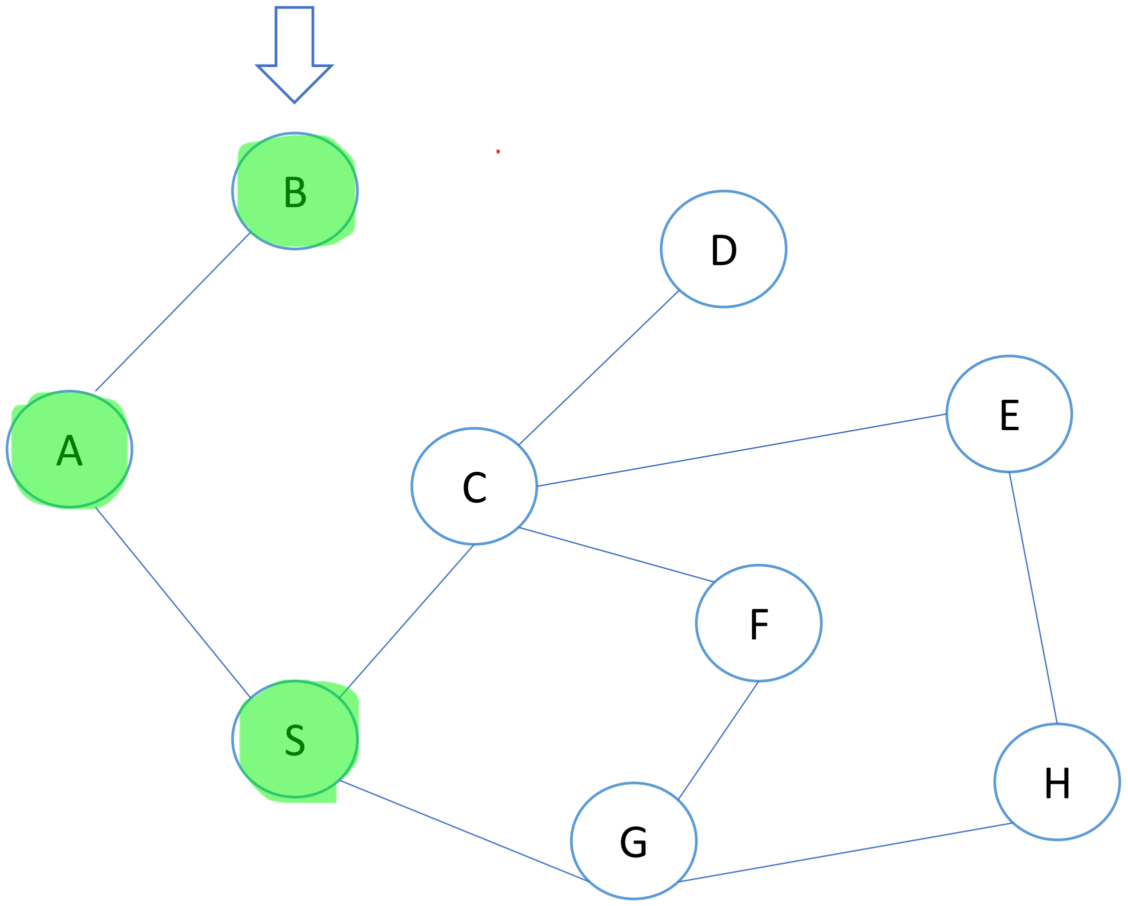
Keeping track of all visited nodes



Queue

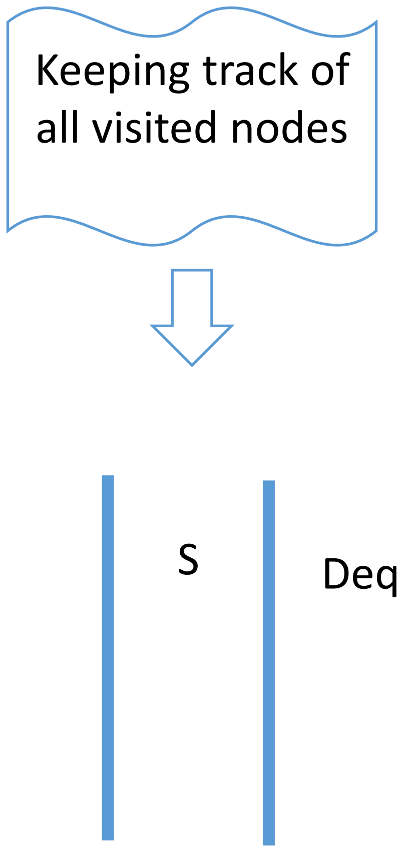
Output: A, B,S

Breadth First Search (BFS)



Output: A, B,S

🟩 Visited node



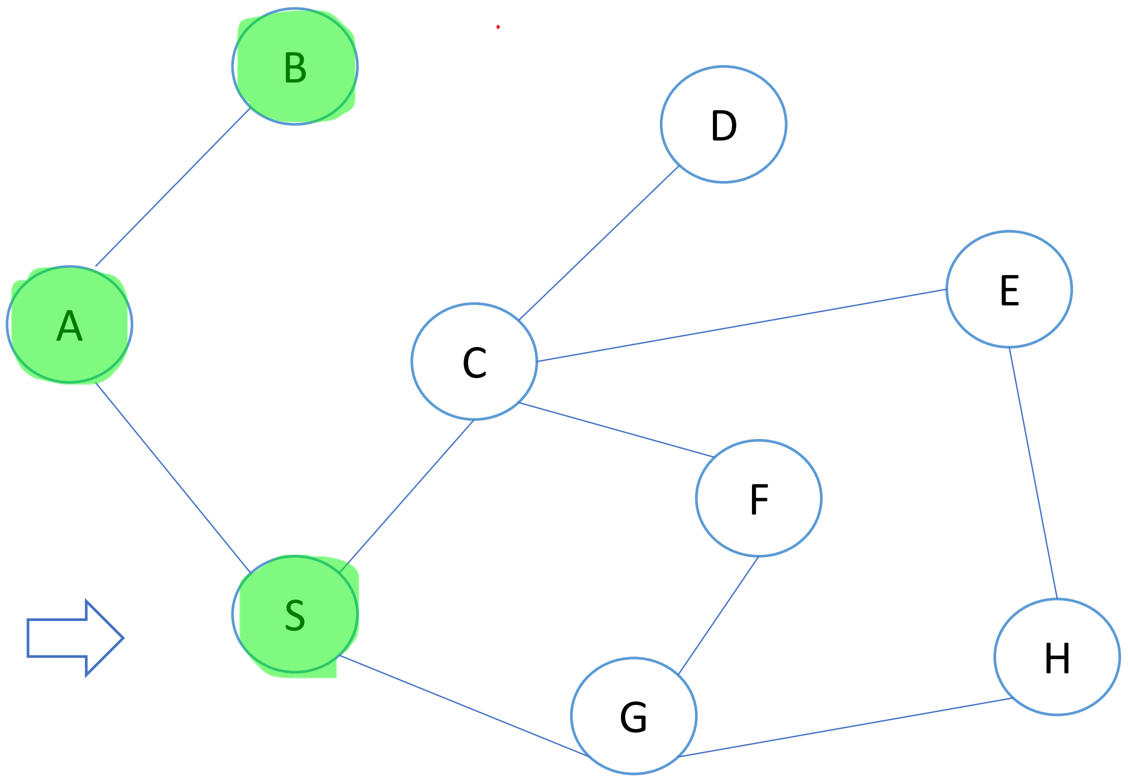
Keeping track of all visited nodes

Dequeue B

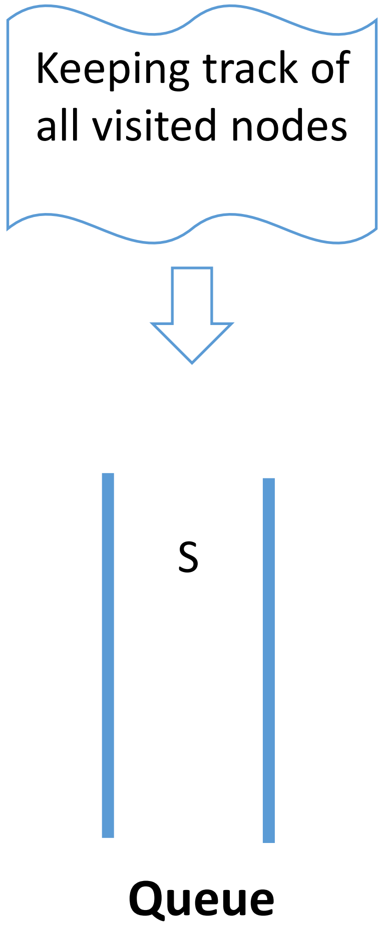
Queue

Breadth First Search (BFS)

🟩 Visited node

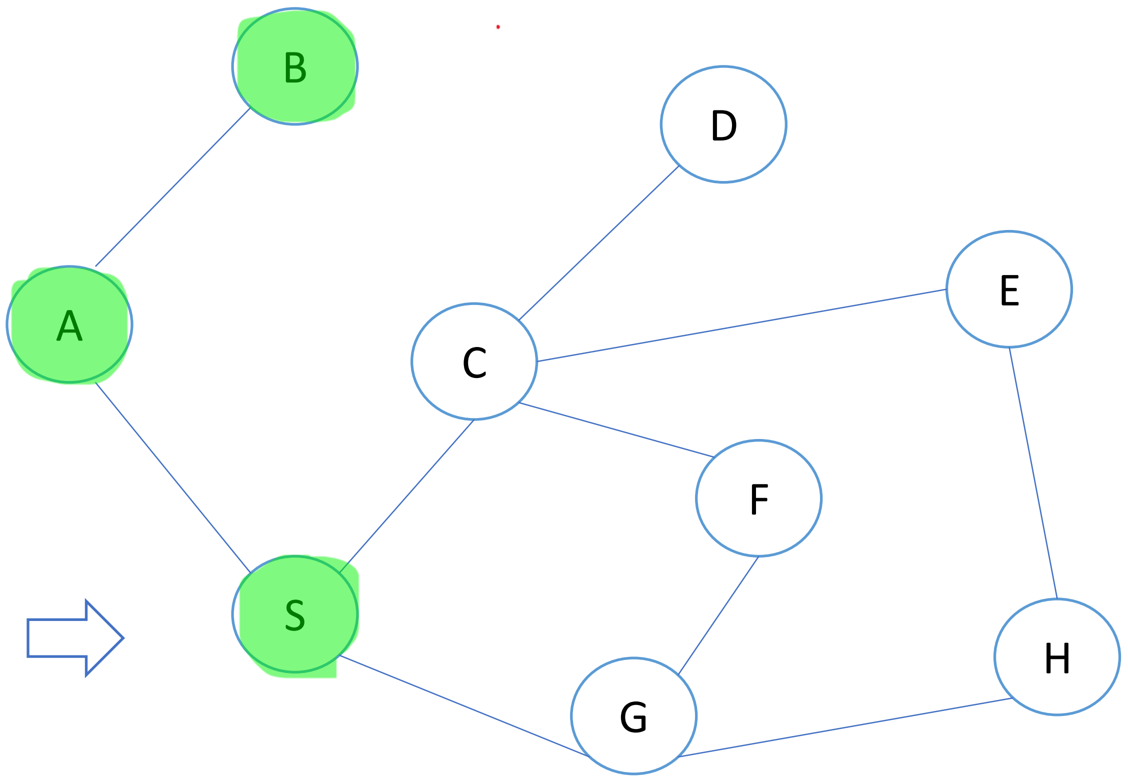


Output: A, B,S

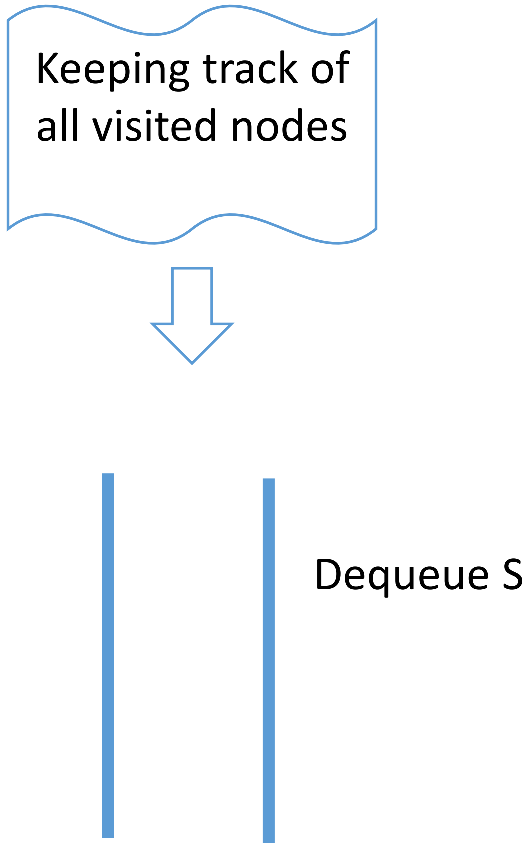


Breadth First Search (BFS)

🟩 Visited node



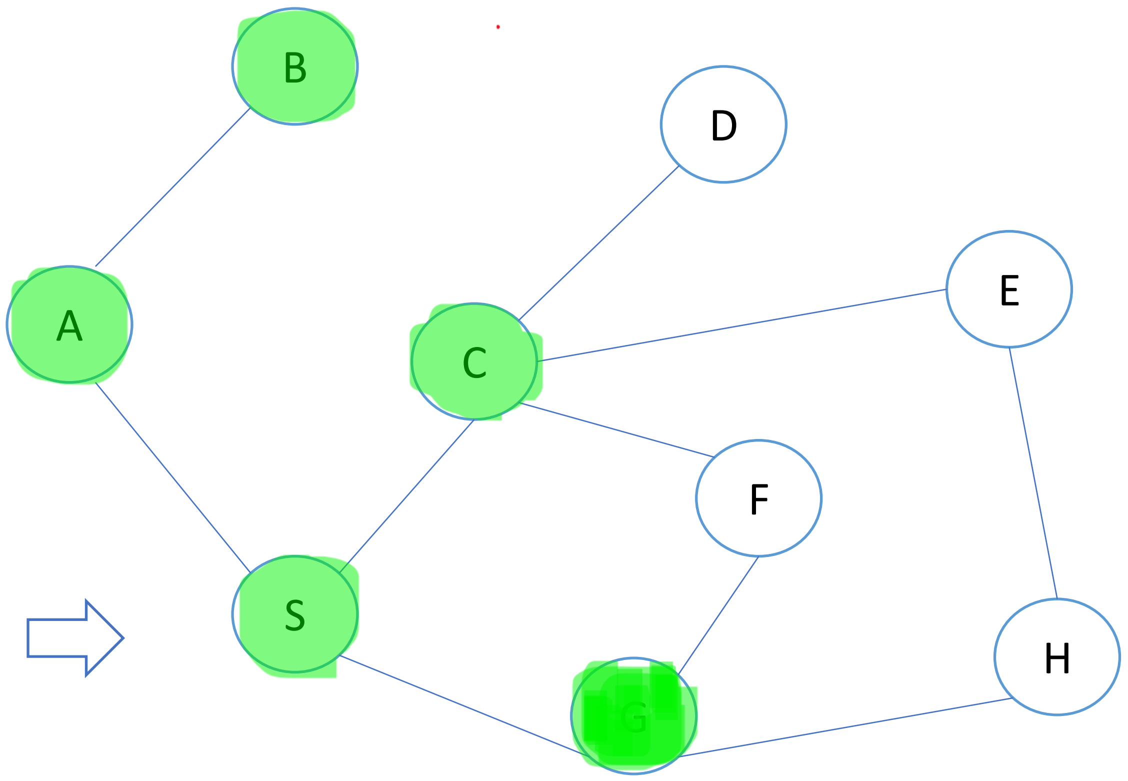
Output: A, B, S



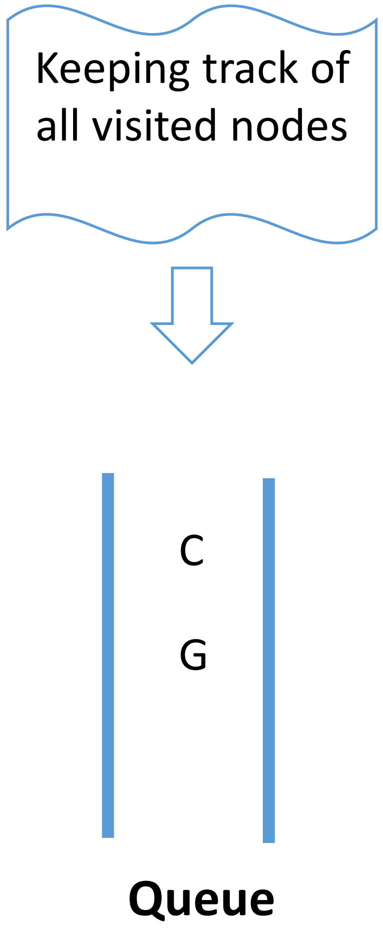
Queue

Breadth First Search (BFS)

🟩 Visited node

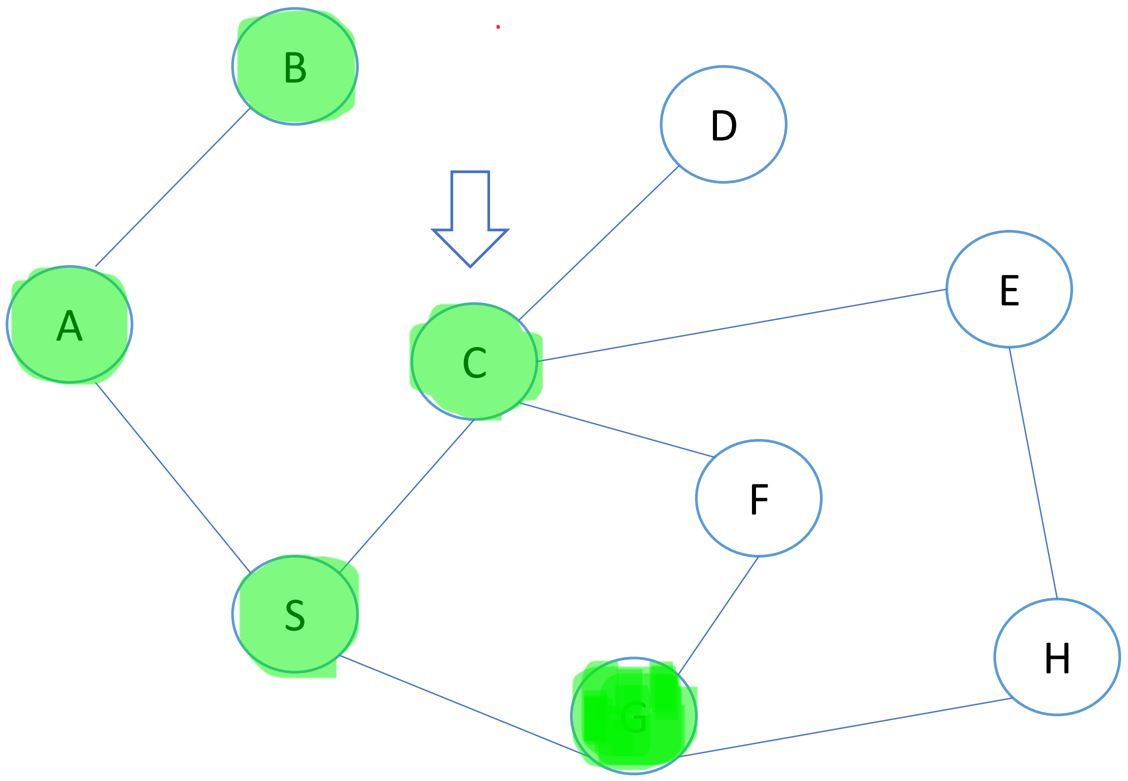


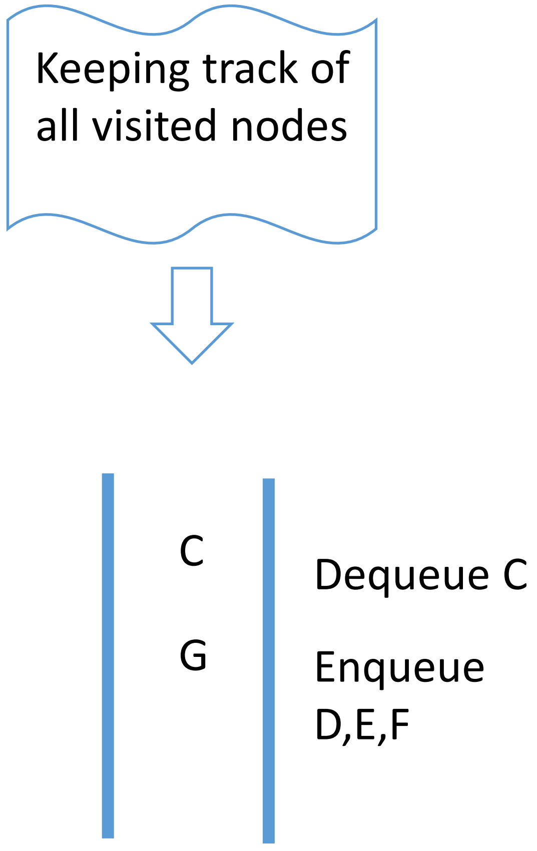
Output: A, B, S, C, G



Breadth First Search (BFS)

🟩 Visited node



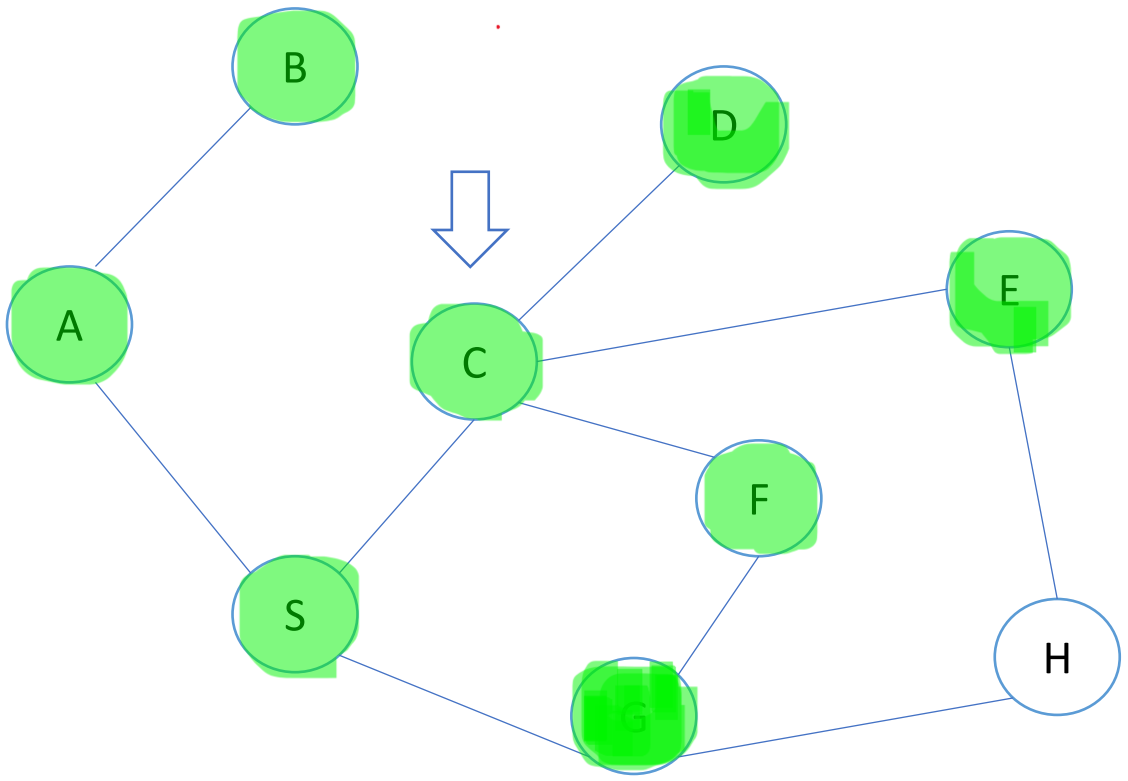

Keeping track of all visited nodes

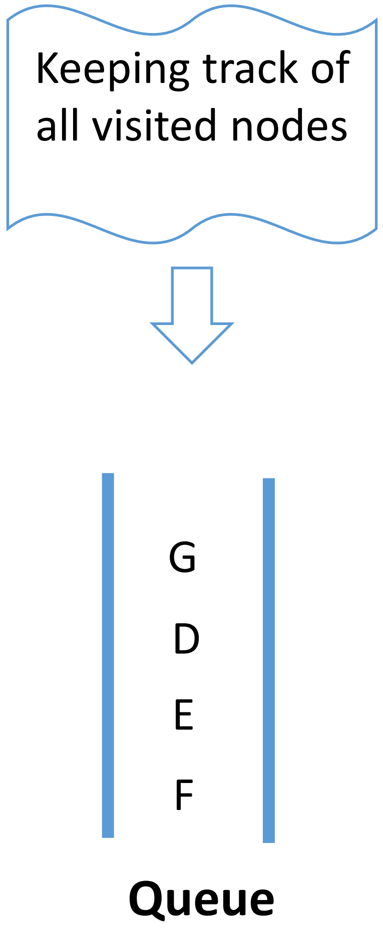
Queue

Output: A, B, S, C, G

Breadth First Search (BFS)

🟩 Visited node



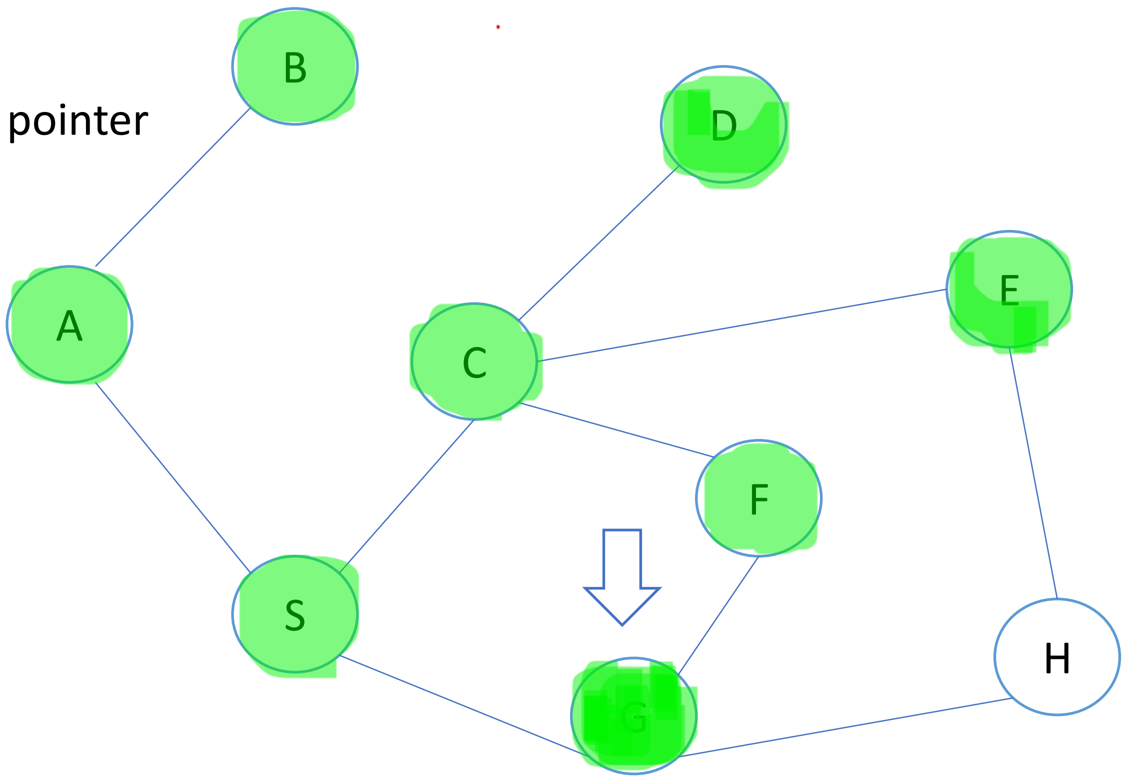


Output: A, B, S, C, G, D, E, F

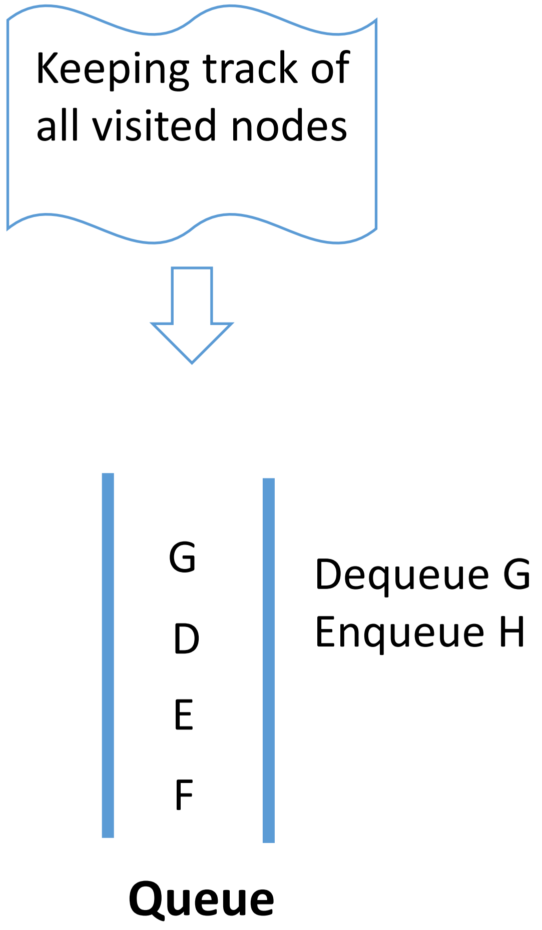
Breadth First Search (BFS)

🟩 Visited node

Updated the pointer



Keeping track of all visited nodes

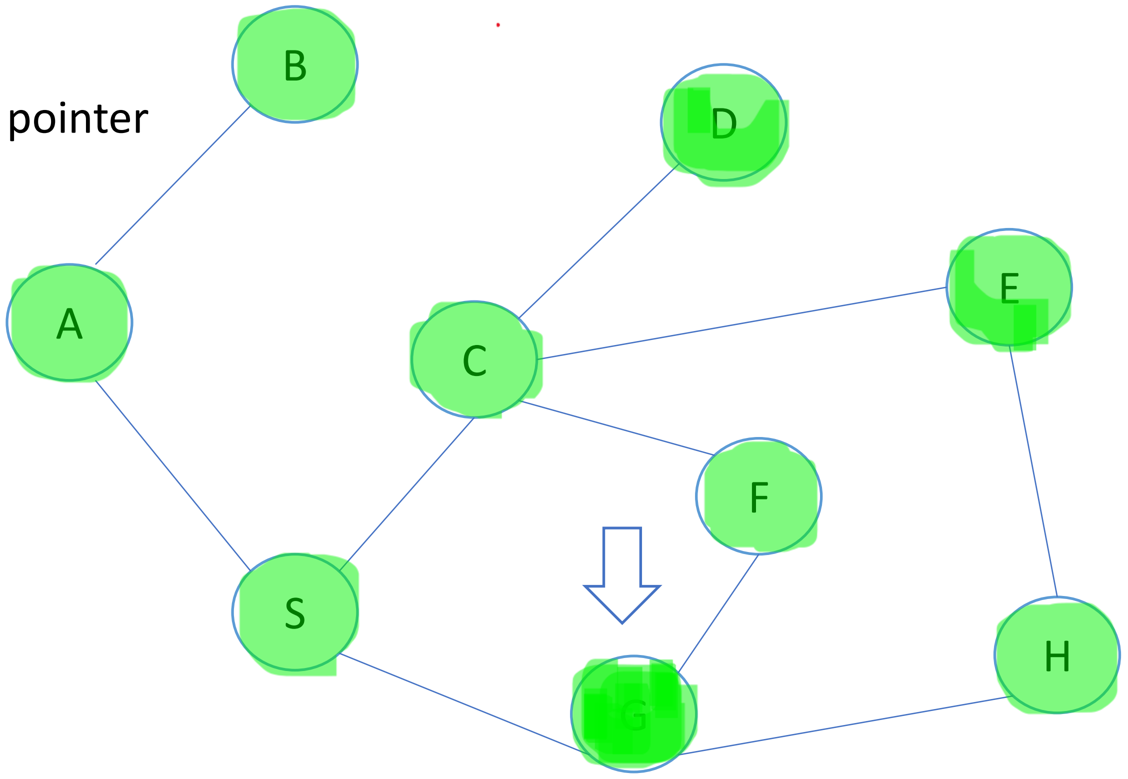


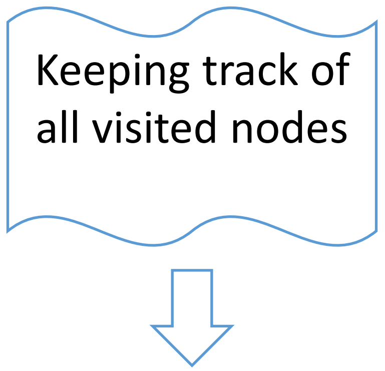
Output: A, B, S, C, G, D, E, F

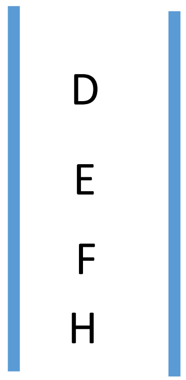
Breadth First Search (BFS)

🟩 Visited node

Updated the pointer







Dequeue D
And others

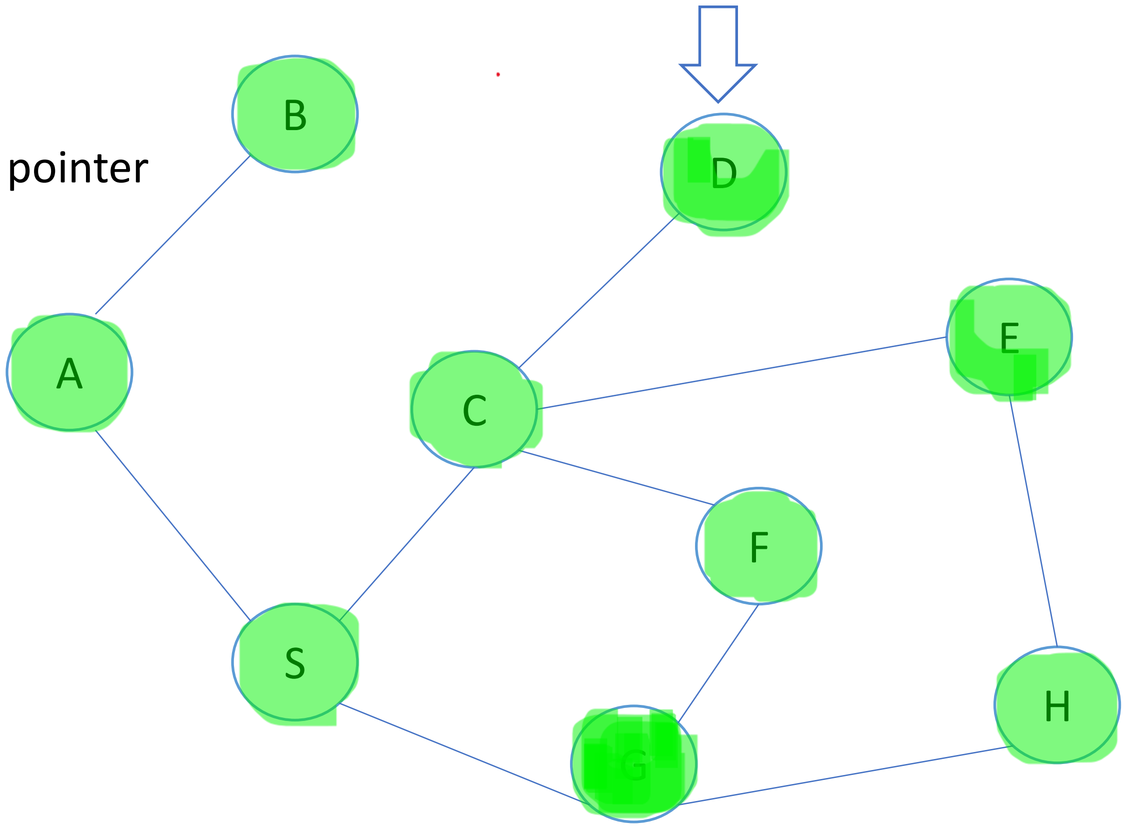
Queue

Output: A, B, S, C, G, D, E, F, H

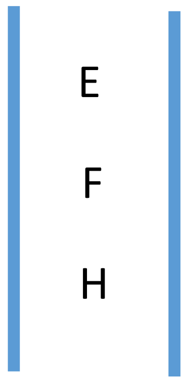
Breadth First Search (BFS)

🟩 Visited node

Updated the pointer



Keeping track of all visited nodes



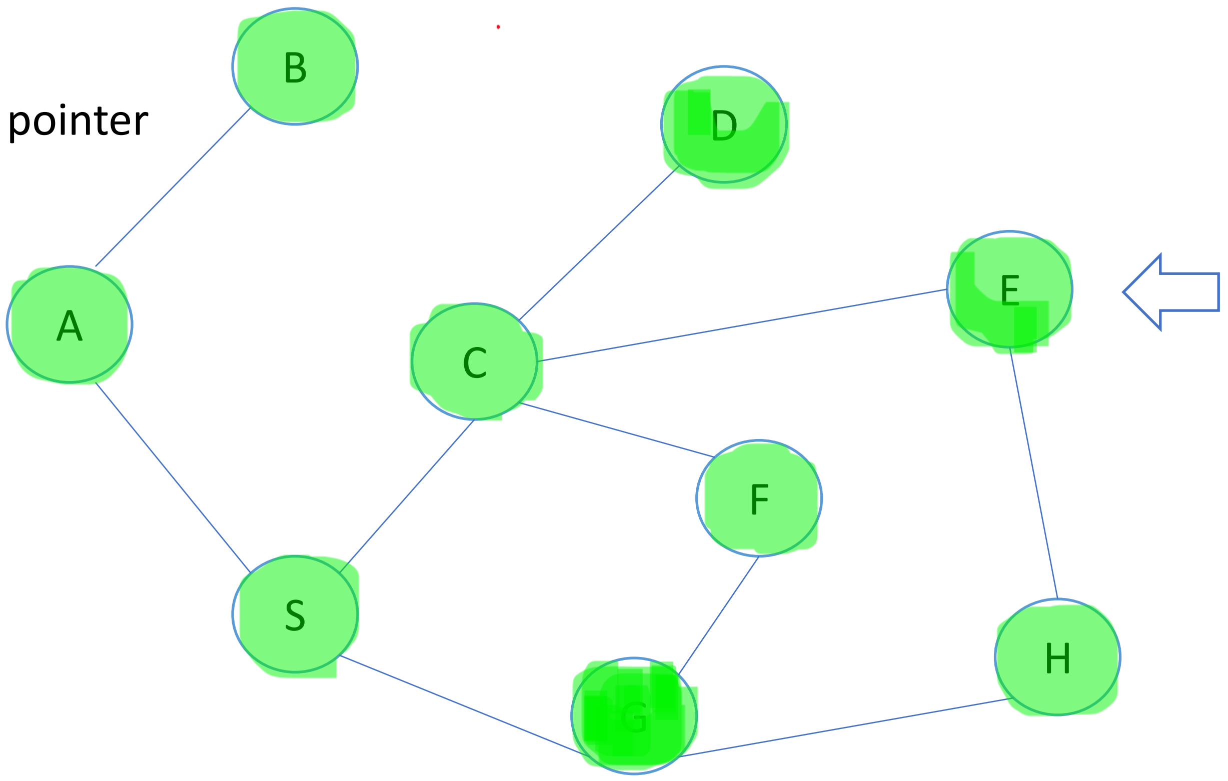
Queue

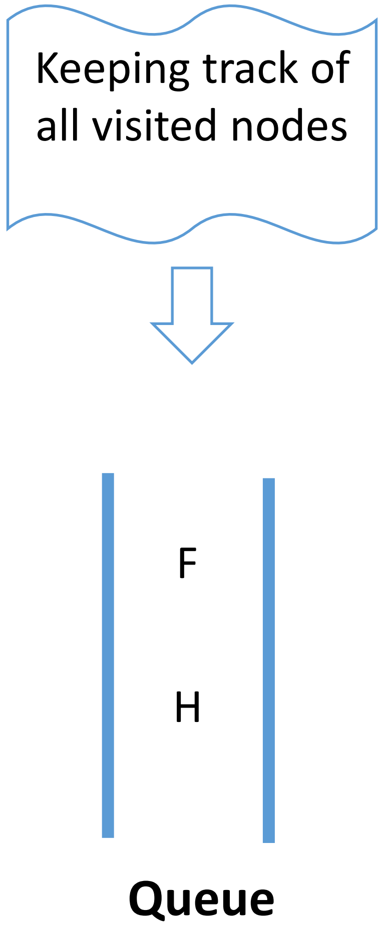
Output: A, B, S, C, G, D, E, F, H

Breadth First Search (BFS)

🟩 Visited node

Updated the pointer



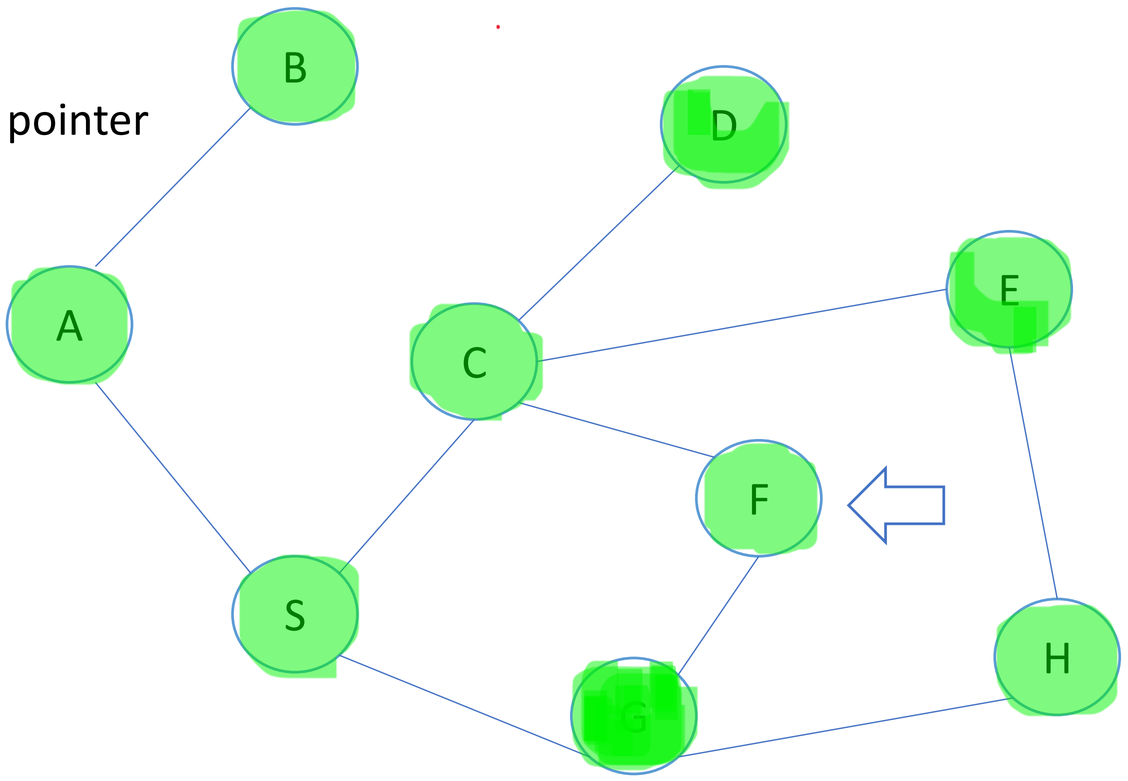


Output: A, B, S, C, G, D, E, F, H

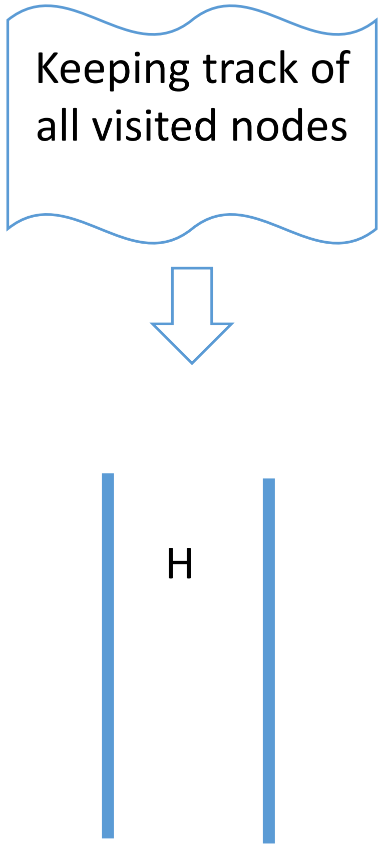
Breadth First Search (BFS)

🟩 Visited node

Updated the pointer



Keeping track of all visited nodes



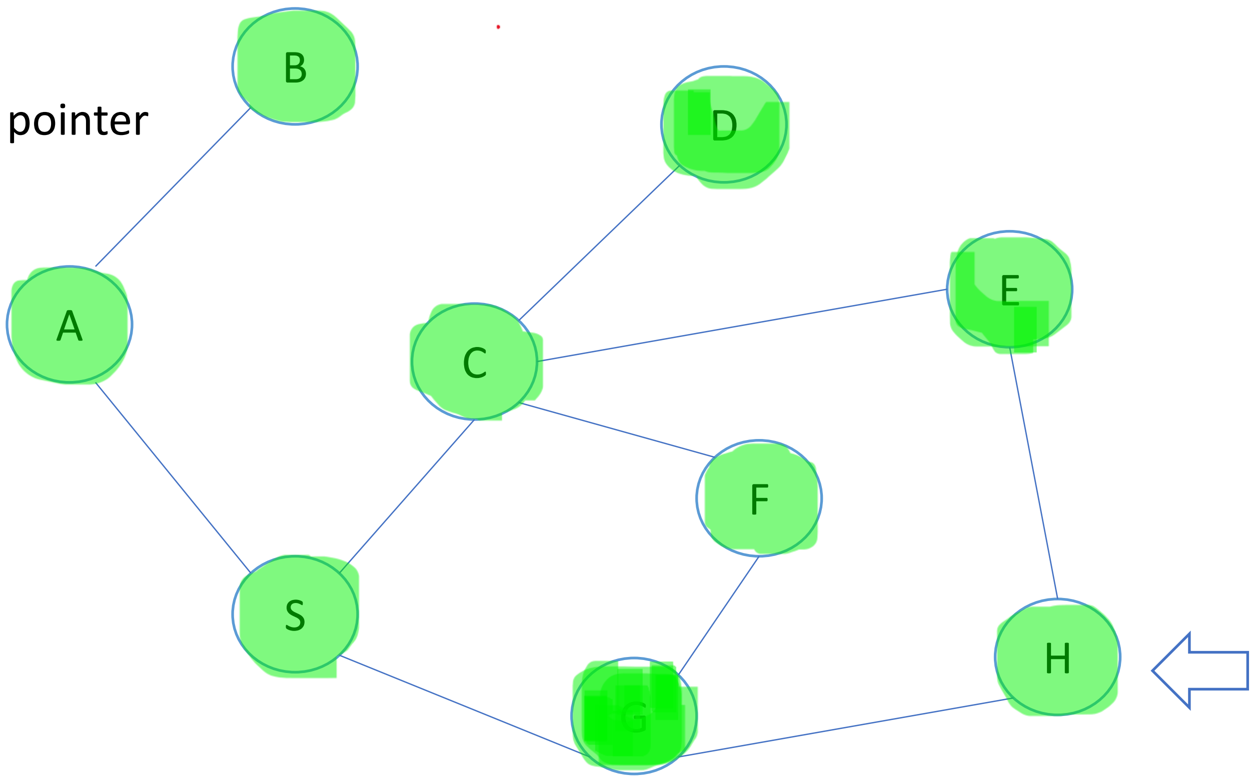
Queue

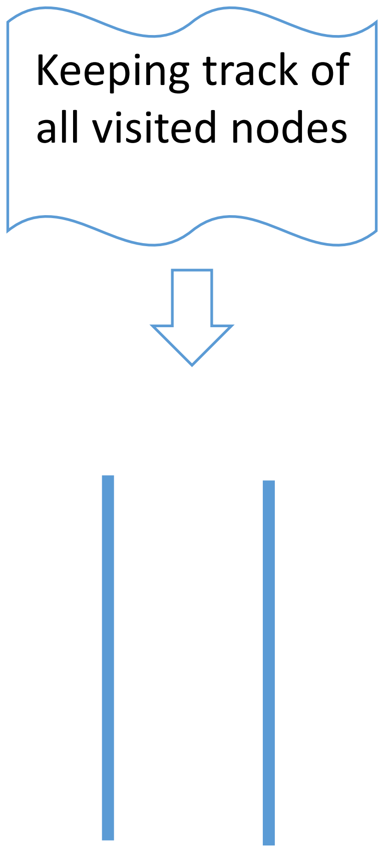
Output: A, B, S, C, G, D, E, F, H

Breadth First Search (BFS)

🟩 Visited node

Updated the pointer

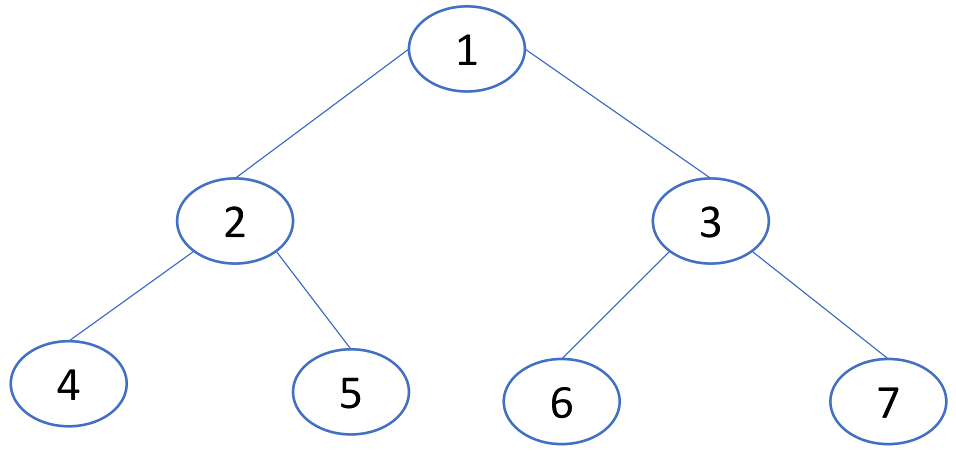




Output: A, B, S, C, G, D, E, F, H

Queue

DFS & BFS in Tree



BFS: 1,2,3,4,5,6,7

DFS: 1,2,4,5,3,6,7