

Introduction to Computational Thinking and Data Science

BFS VS. DFS

Somaia Zabihi

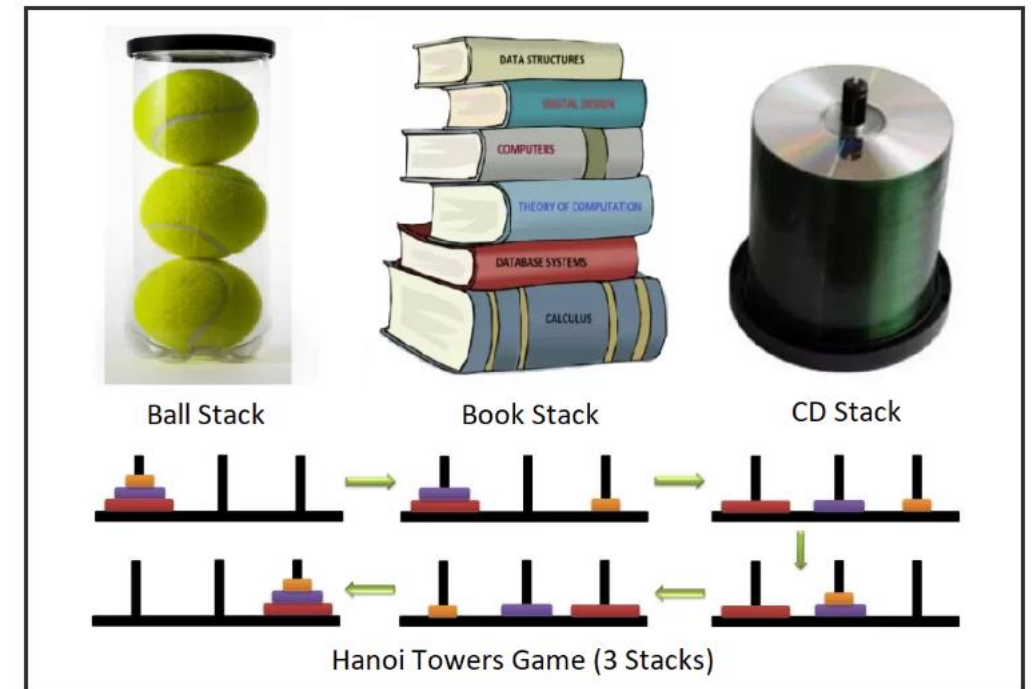
BFS & DFS

- 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.

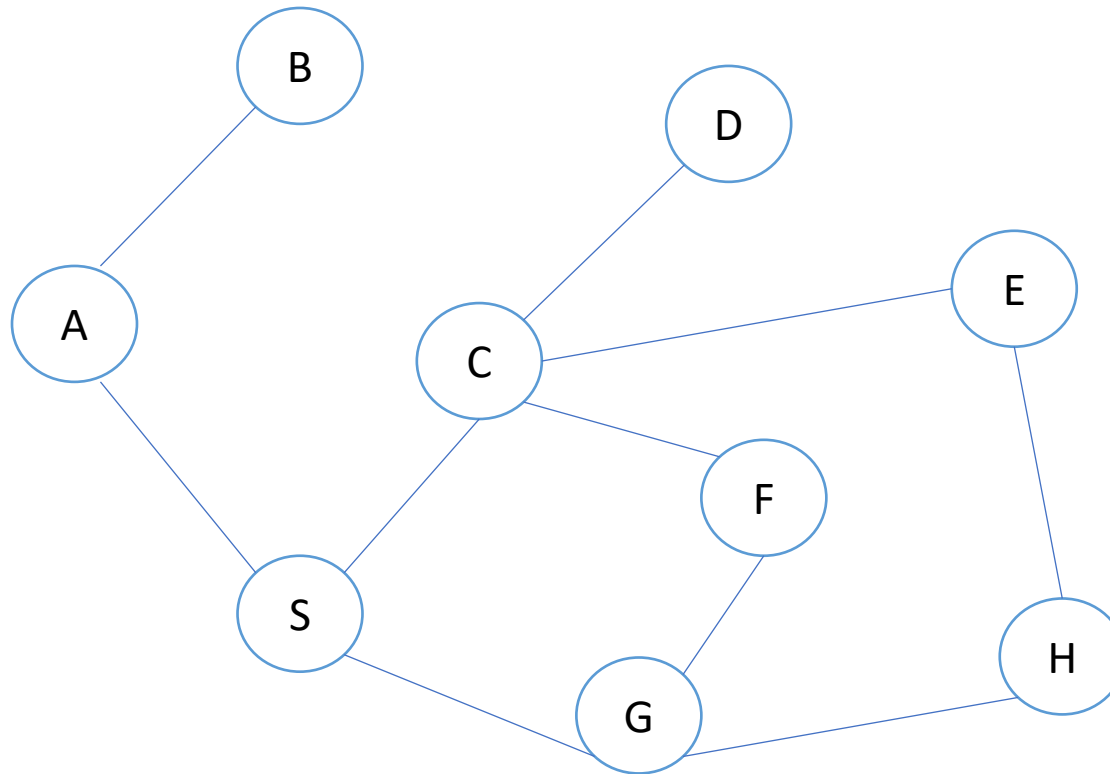
Depth-First Search (DFS)

- 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)



Output:

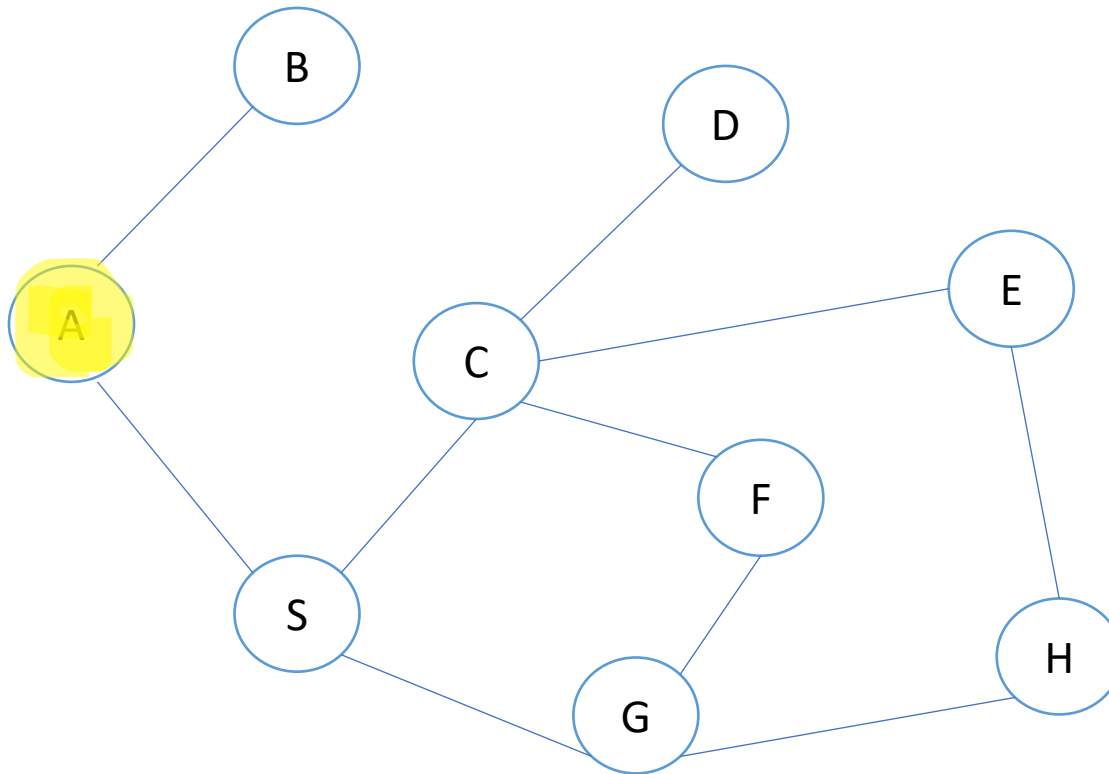
 Visited node

Keeping track of
all visited nodes



Stack

Depth First Search (DFS)



Output: A

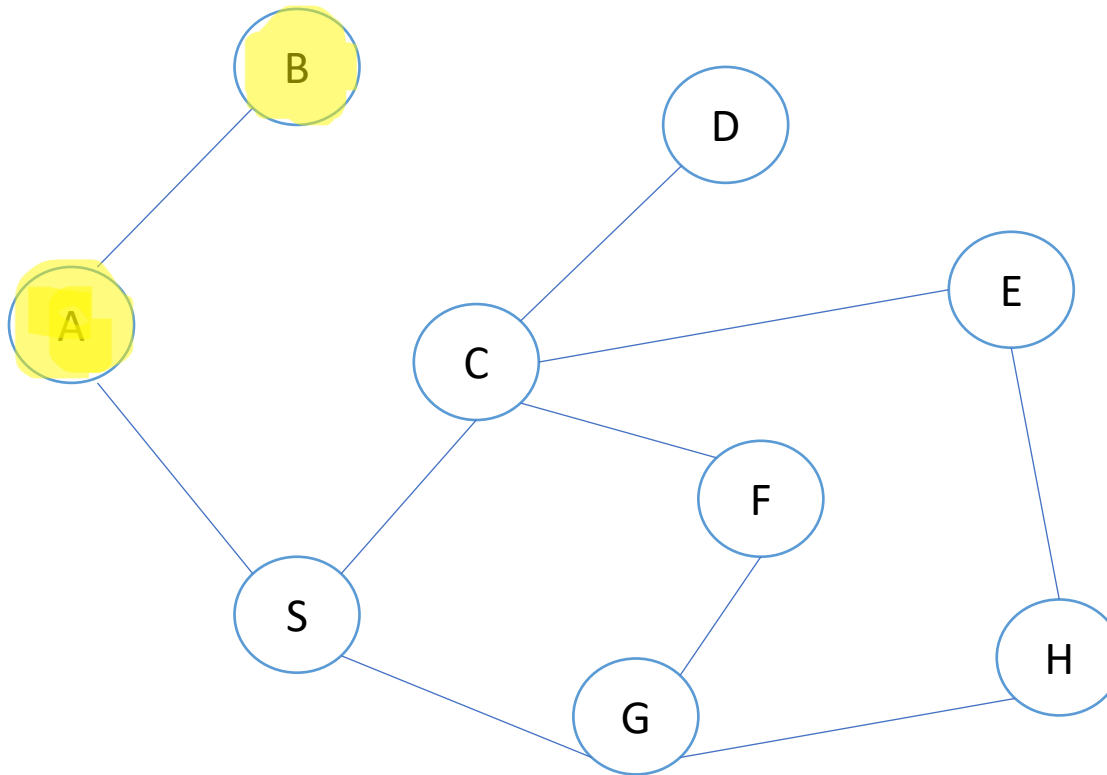
 Visited node

Keeping track of
all visited nodes



Stack

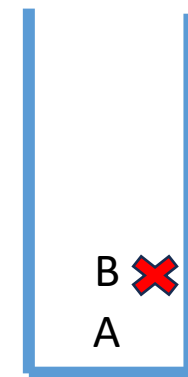
Depth First Search (DFS)



Output: A,B

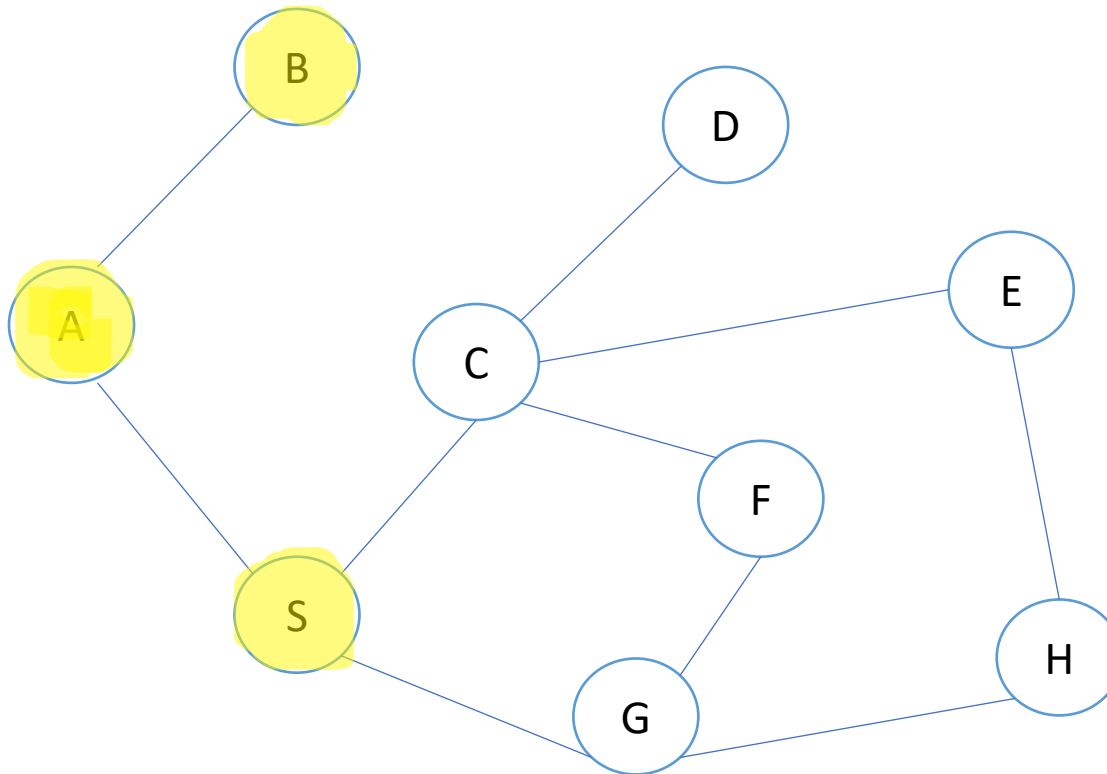
 Visited node

Keeping track of
all visited nodes



Stack

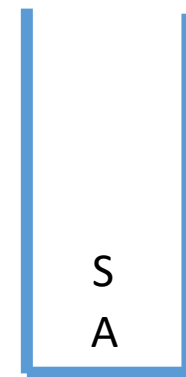
Depth First Search (DFS)



Output: A,B,S

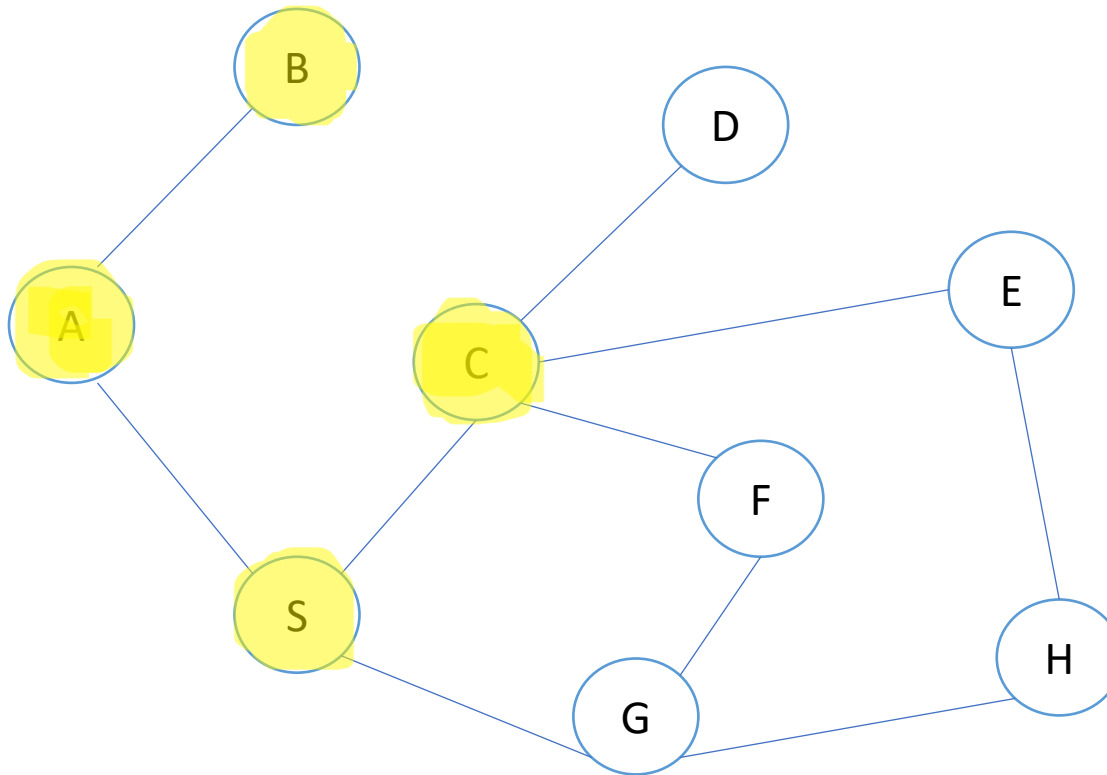
 Visited node

Keeping track of
all visited nodes



Stack

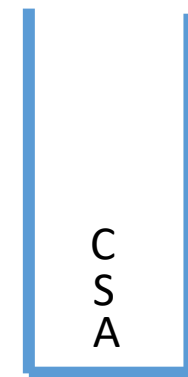
Depth First Search (DFS)



Output: A,B,S,C

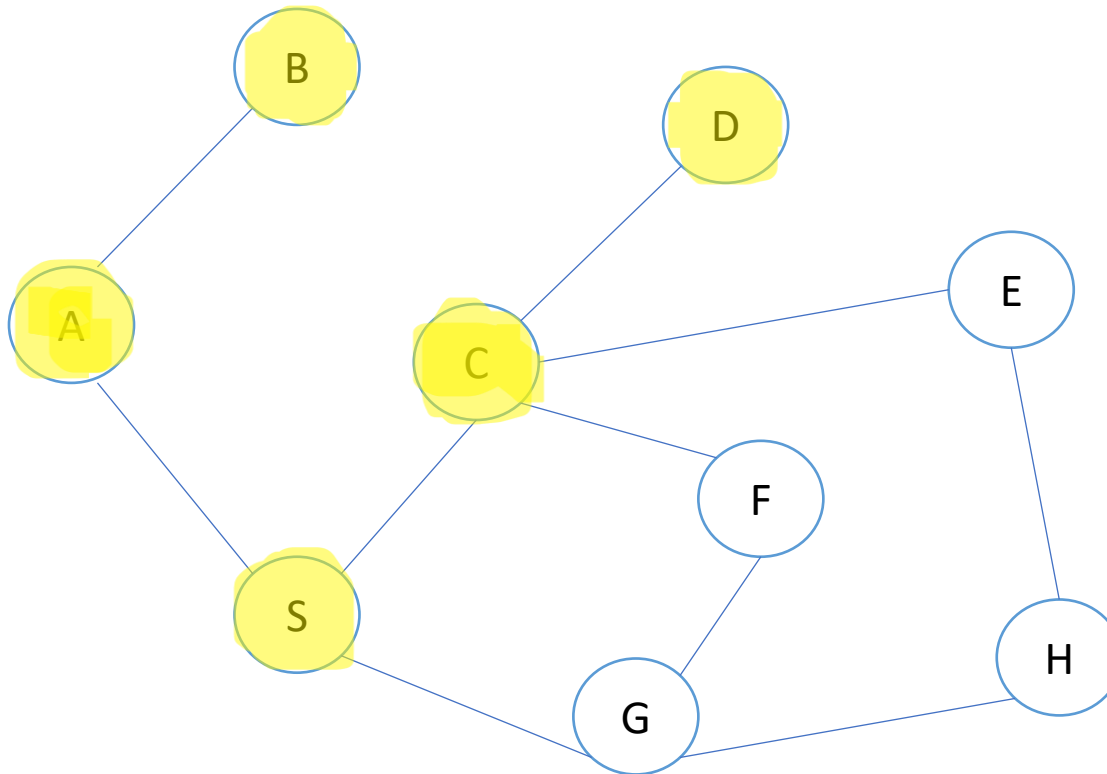
 Visited node

Keeping track of
all visited nodes



Stack

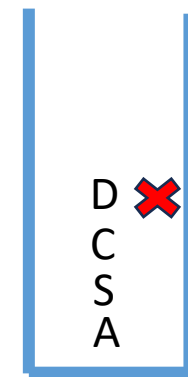
Depth First Search (DFS)



Output: A,B,S,C,D

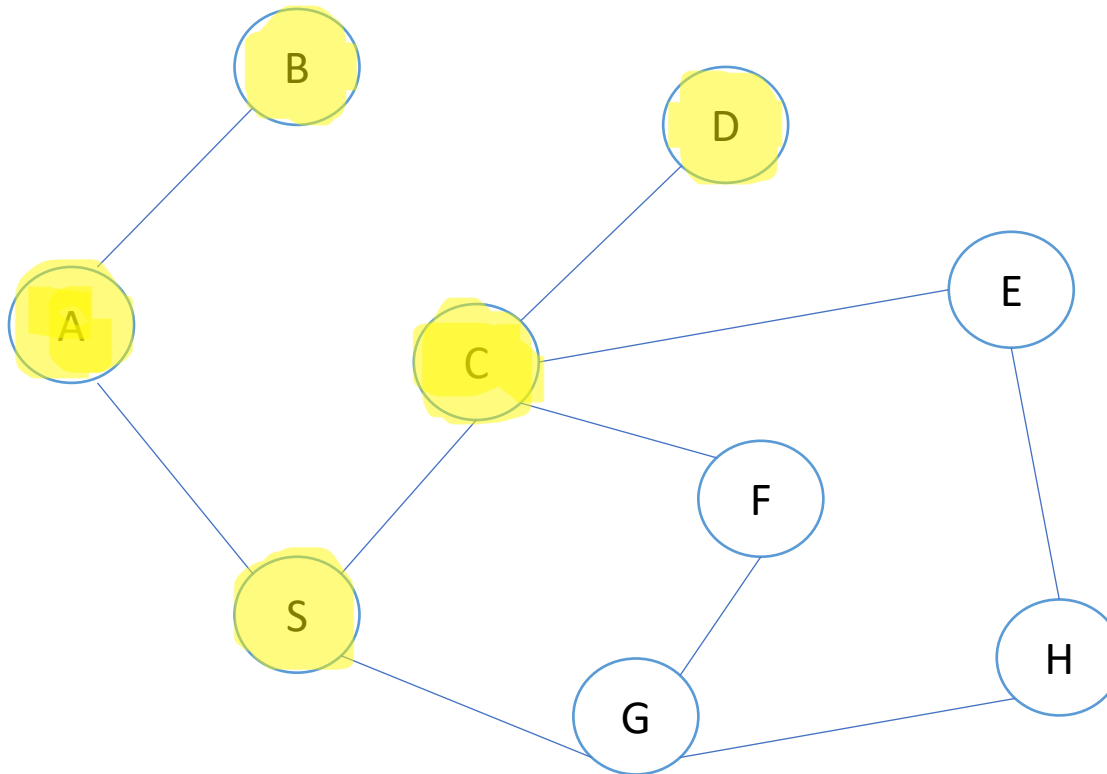
 Visited node

Keeping track of
all visited nodes



Stack

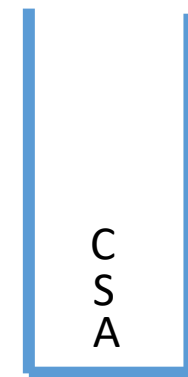
Depth First Search (DFS)



Output: A,B,S,C,D

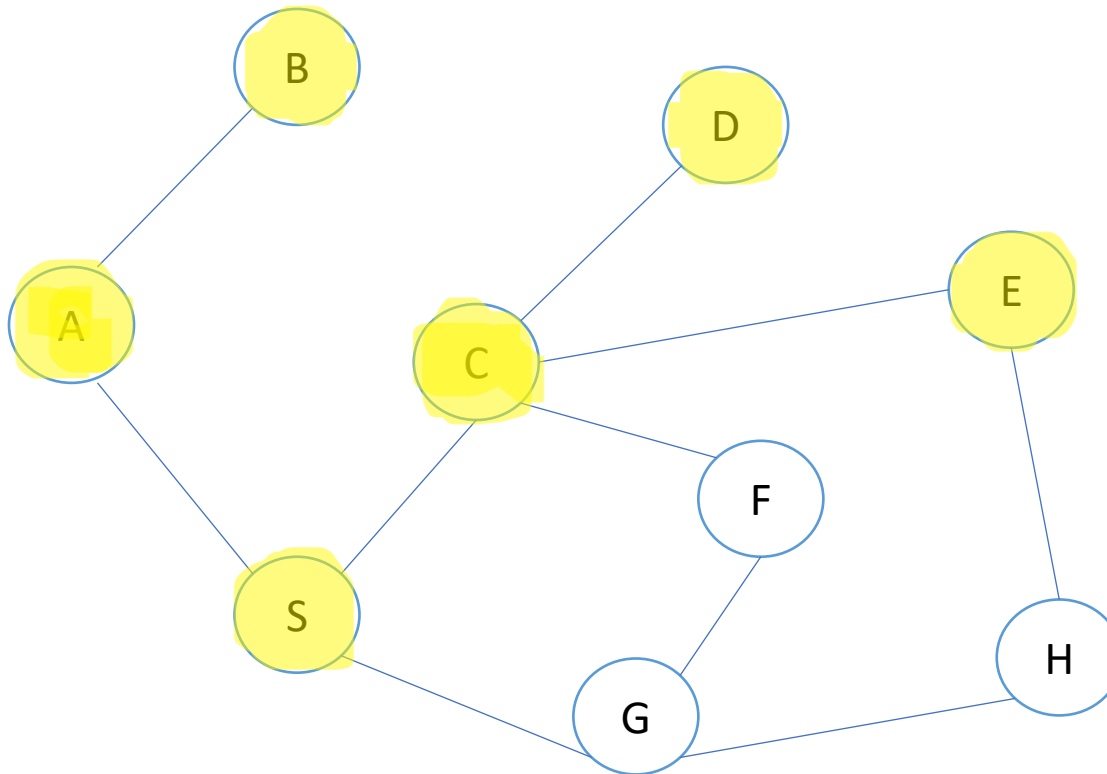
 Visited node

Keeping track of
all visited nodes



Stack

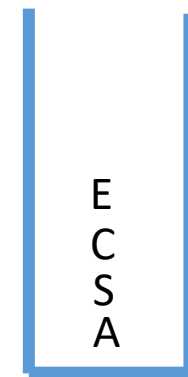
Depth First Search (DFS)



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

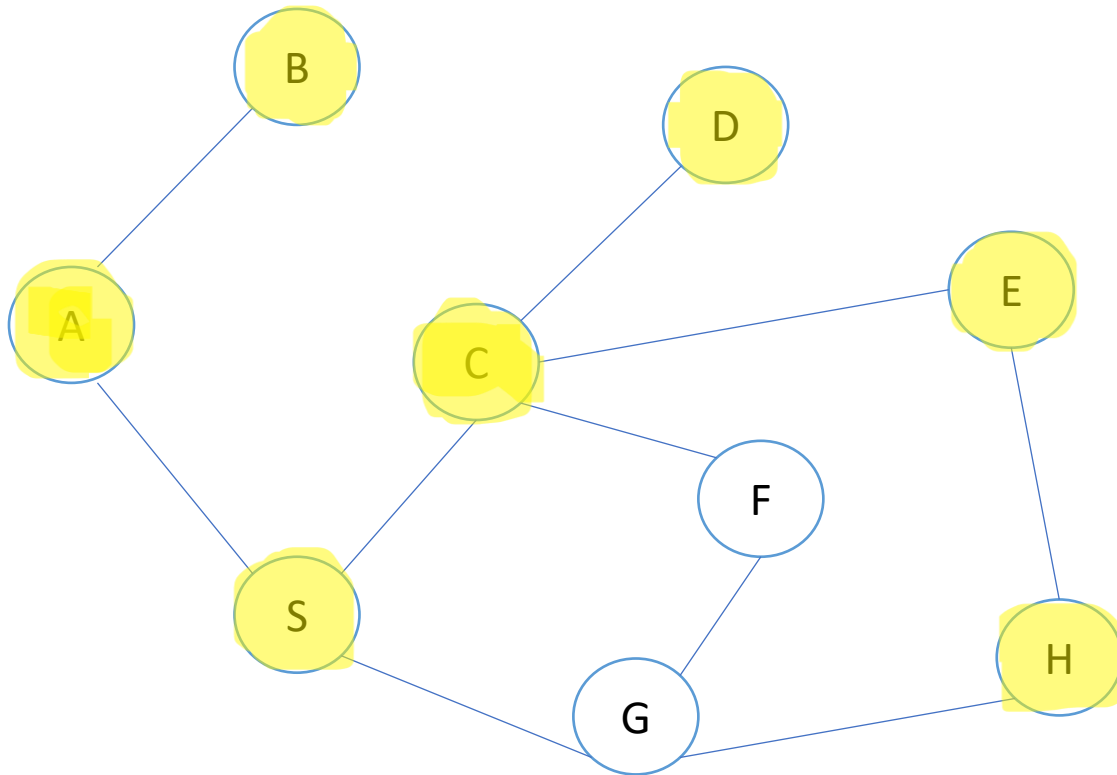
 Visited node

Keeping track of
all visited nodes



Stack

Depth First Search (DFS)



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

 Visited node

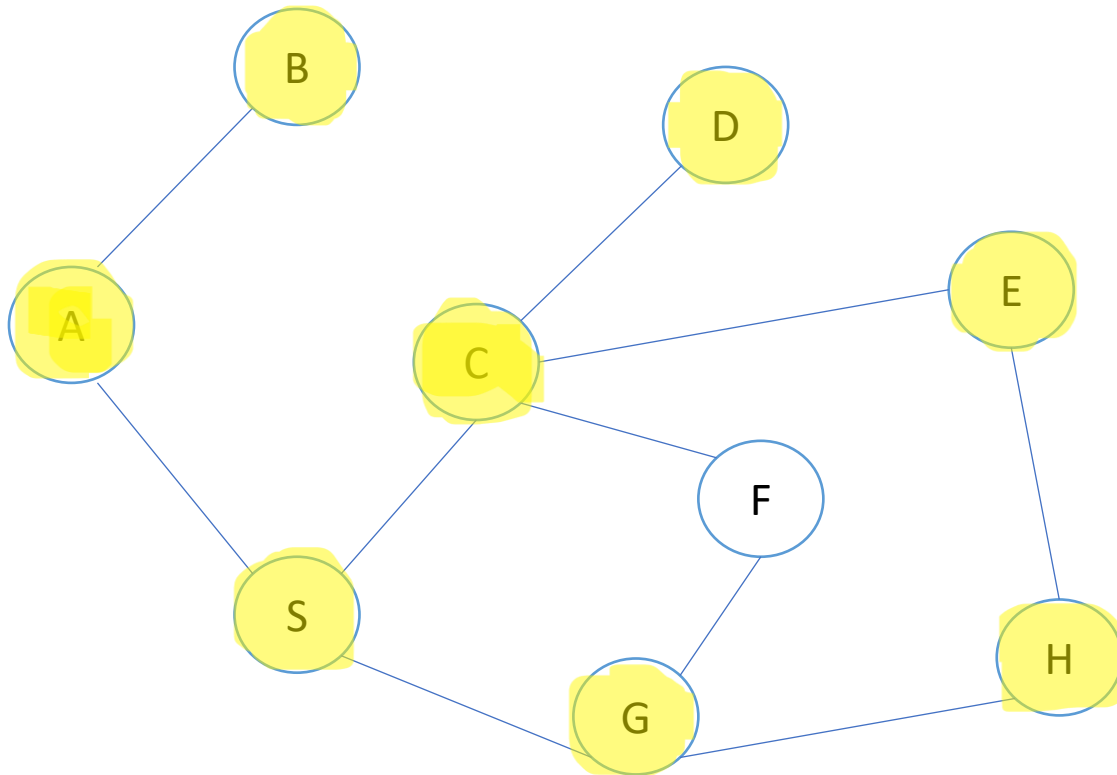
Keeping track of
all visited nodes



H
E
C
S
A

Stack

Depth First Search (DFS)



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

 Visited node

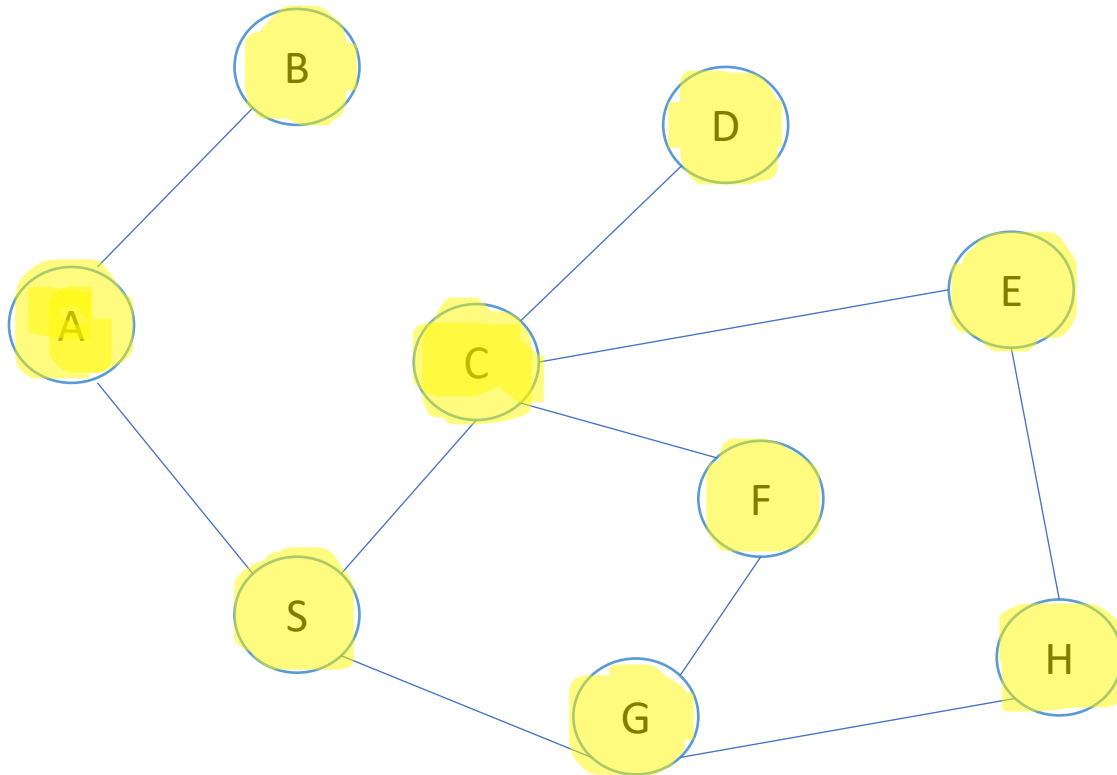
Keeping track of
all visited nodes



G
H
E
C
S
A

Stack

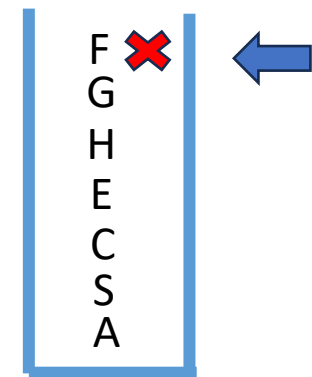
Depth First Search (DFS)



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

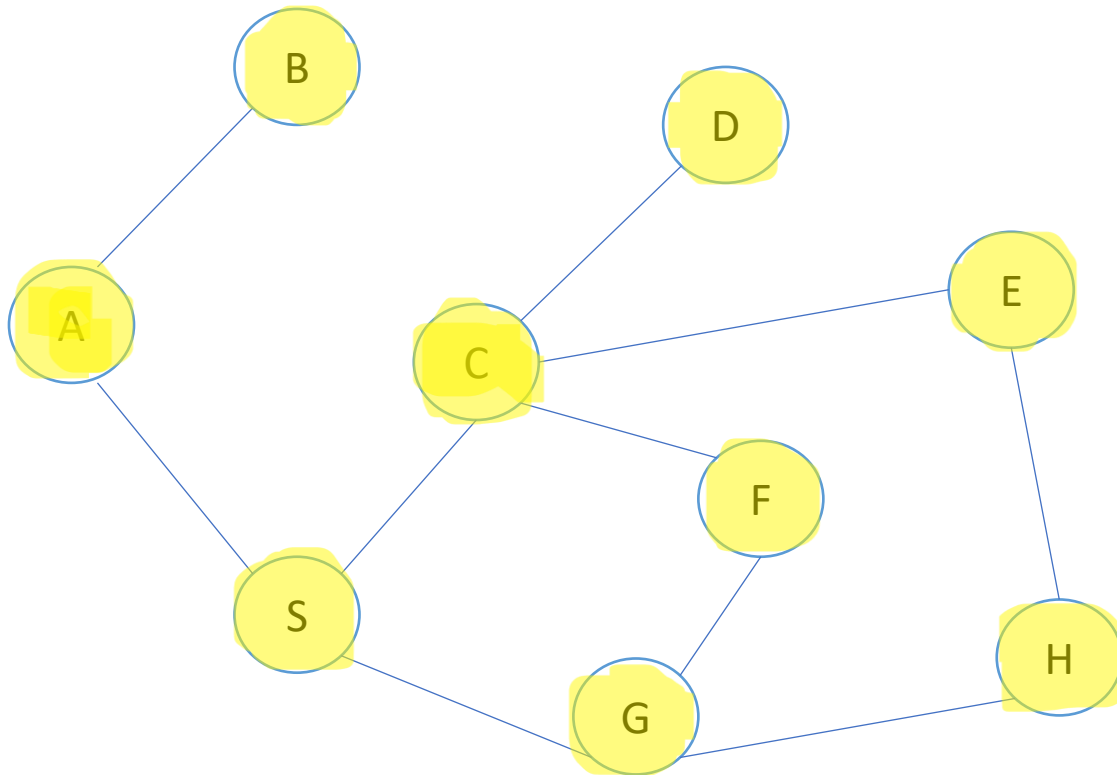
 Visited node

Keeping track of
all visited nodes



Stack

Depth First Search (DFS)



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

 Visited node

Keeping track of
all visited nodes



Stack

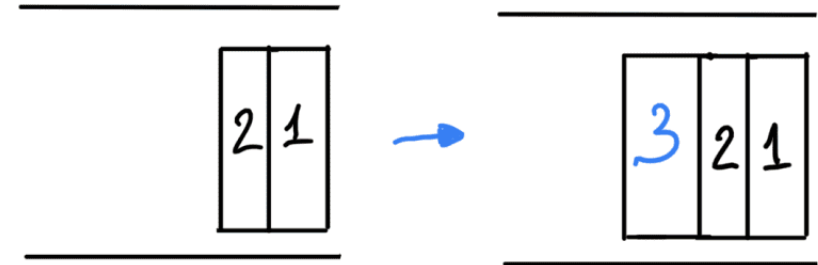
Breadth-First Search(BFS)

- 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.

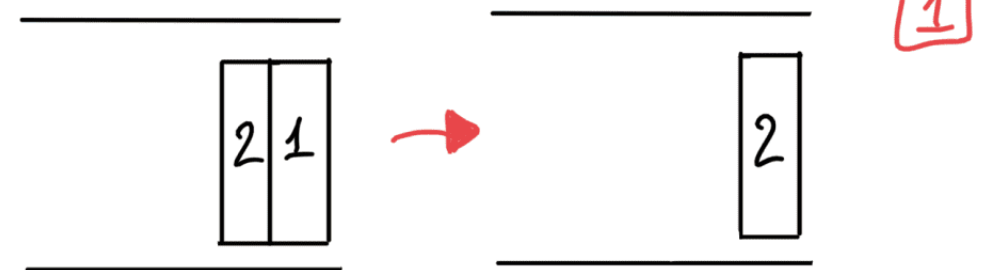
Queue Data Structure



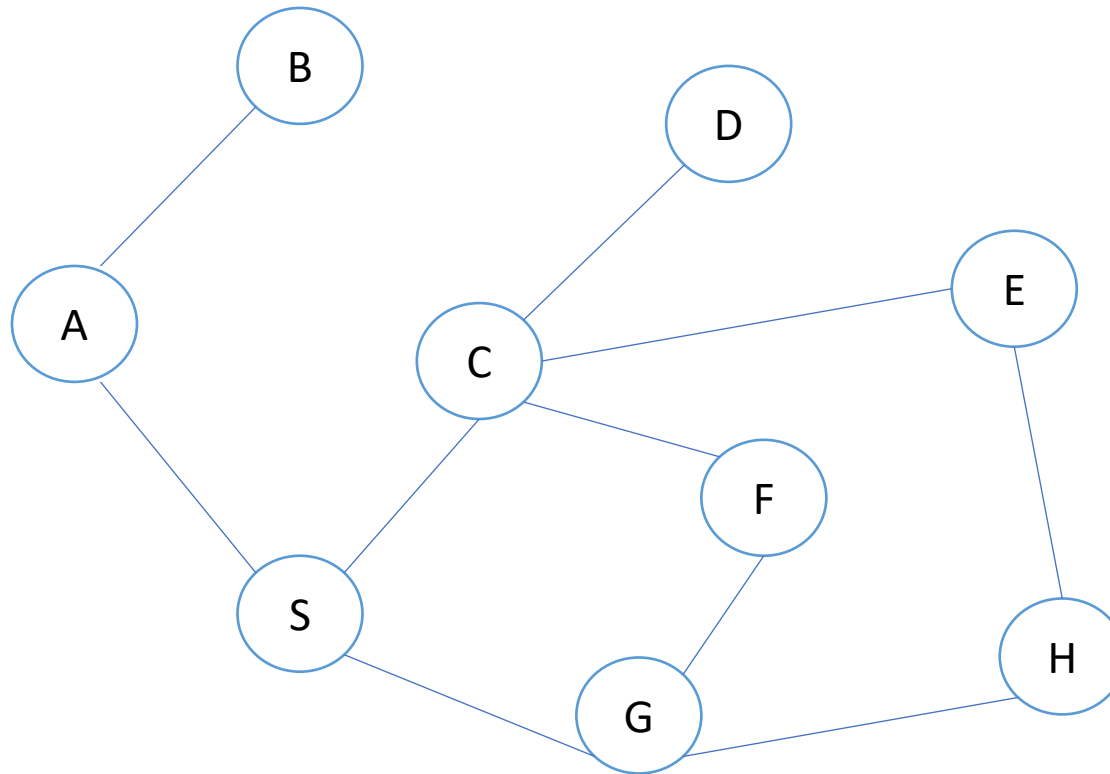
ENQUEUE 3




DEQUEUE



Breadth First Search (BFS)



Output:

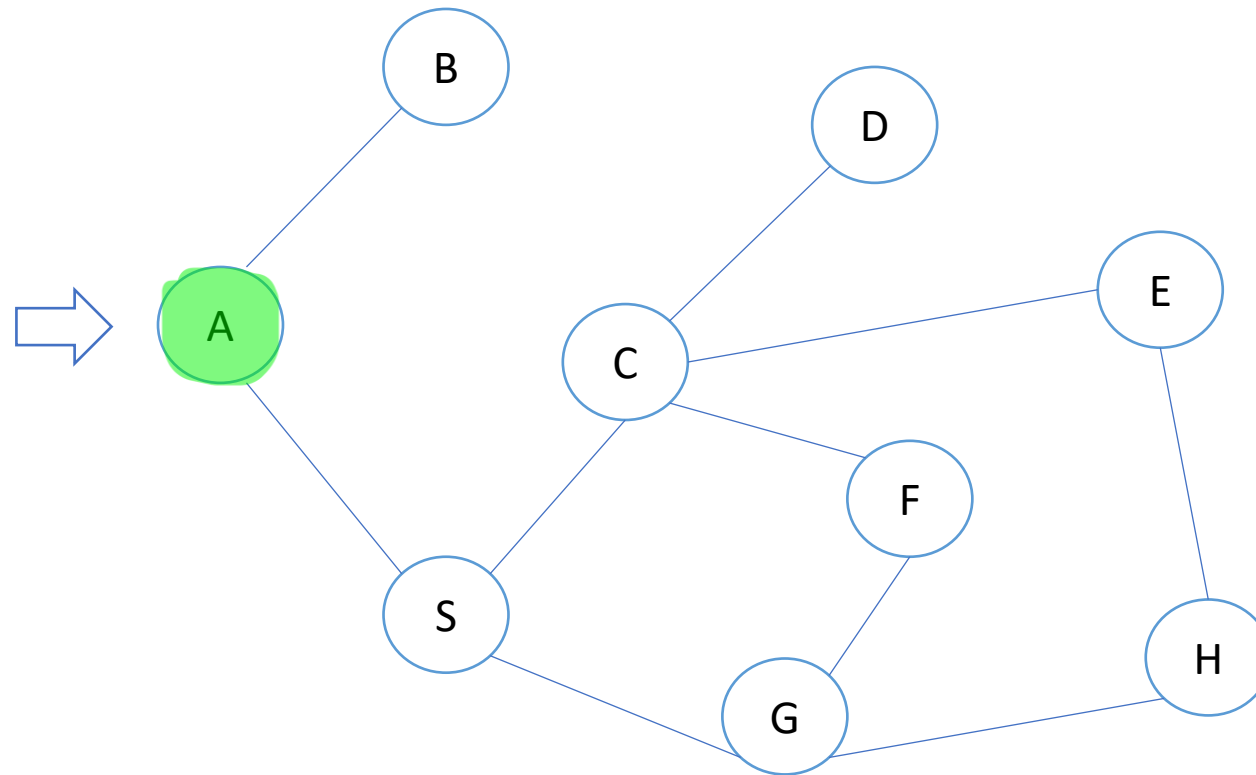
 Visited node

Keeping track of
all visited nodes



Queue

Breadth First Search (BFS)



Output: A, B, S

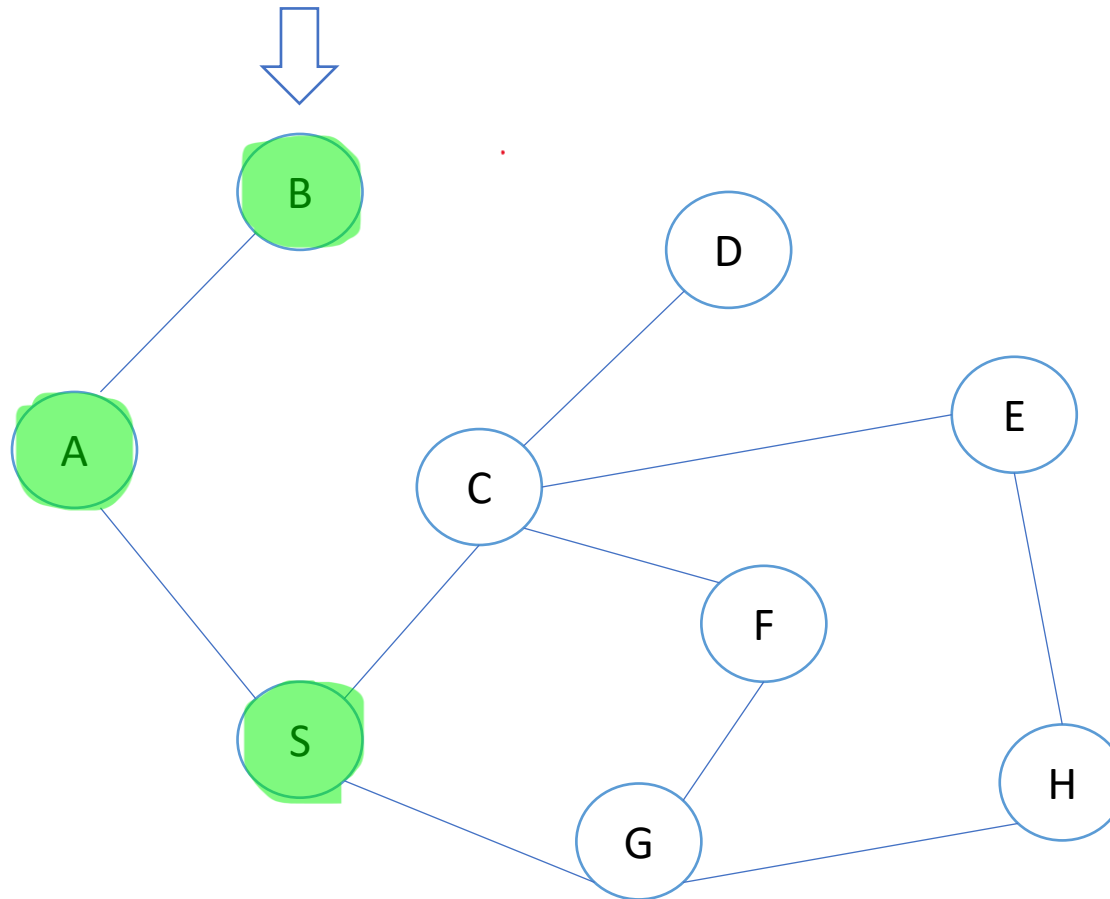
 Visited node

Keeping track of
all visited nodes



Queue

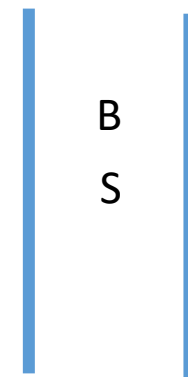
Breadth First Search (BFS)



Output: A, B, S

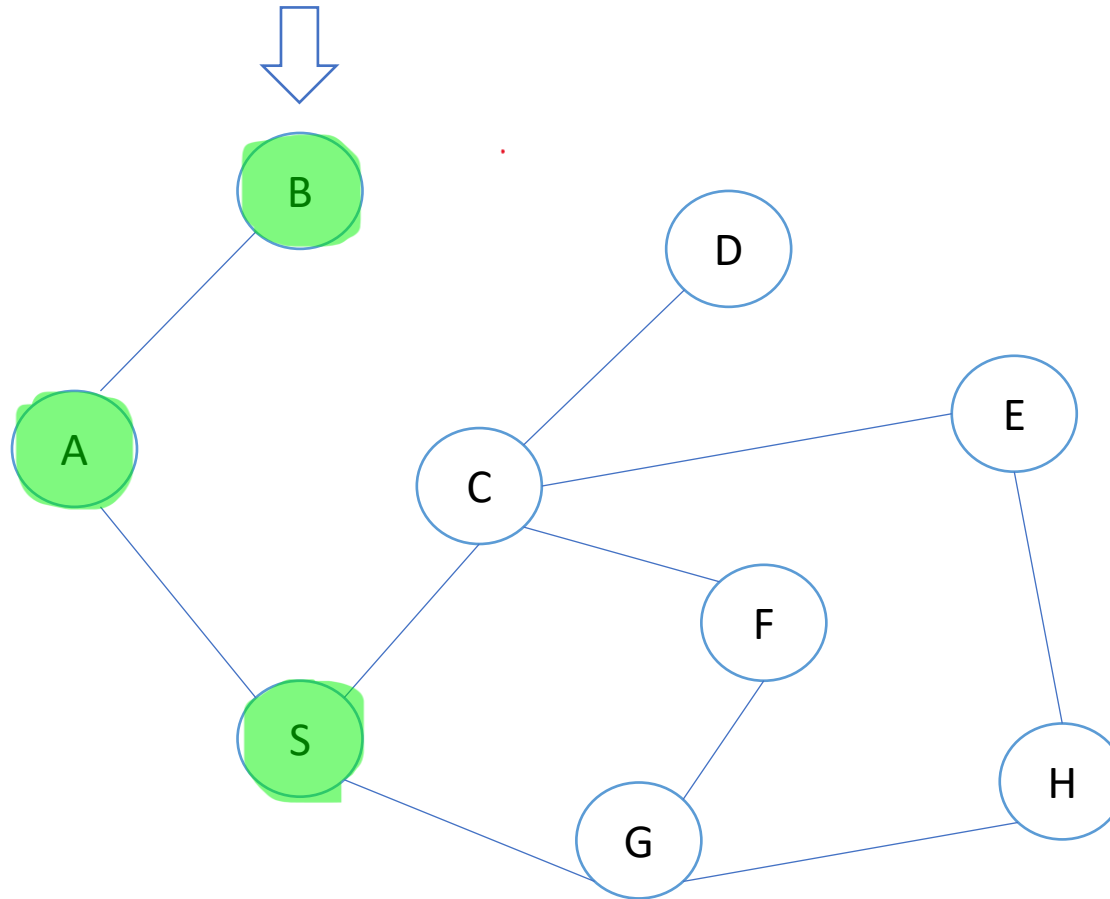
Visited node

Keeping track of
all visited nodes



Queue

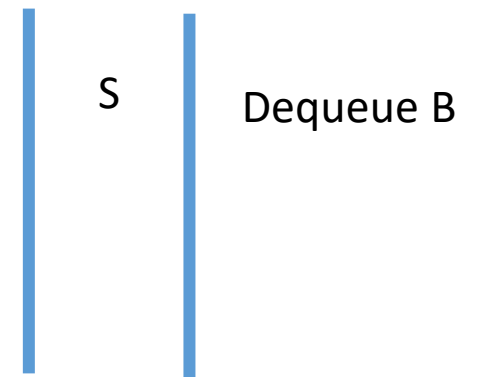
Breadth First Search (BFS)



Output: A, B, S

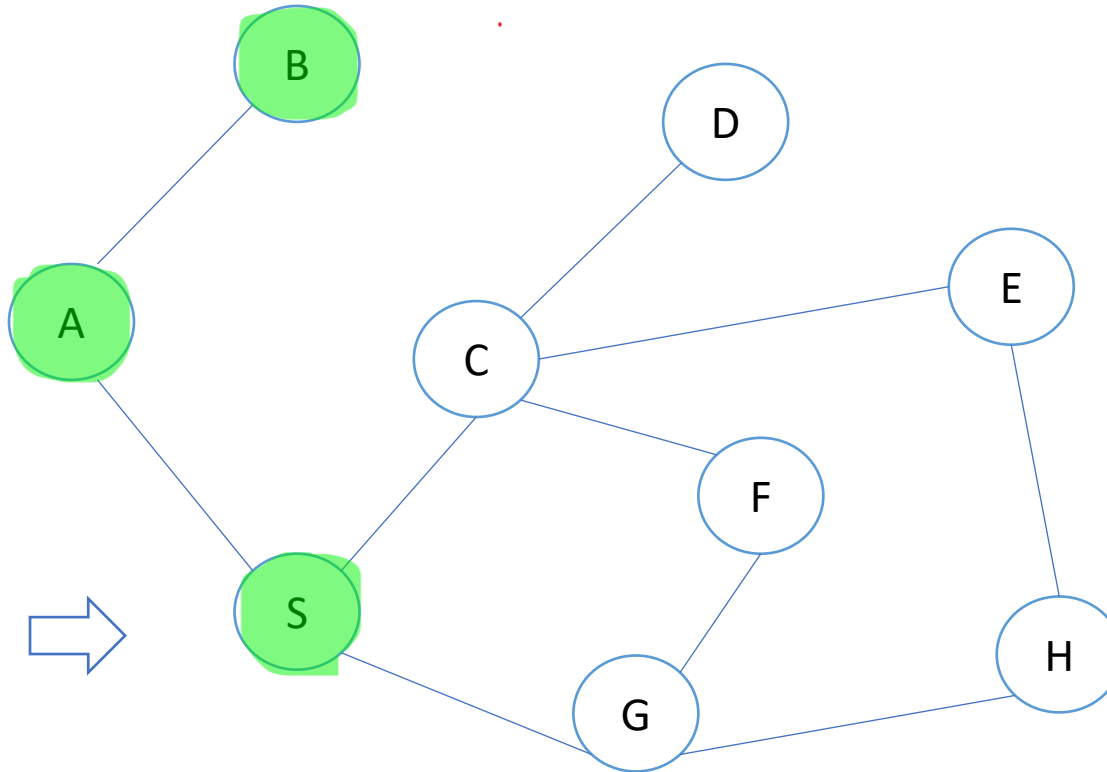
Visited node

Keeping track of
all visited nodes



Queue

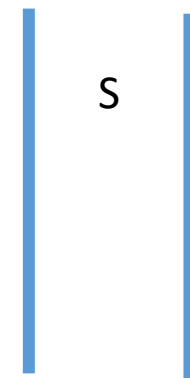
Breadth First Search (BFS)



Output: A, B, S

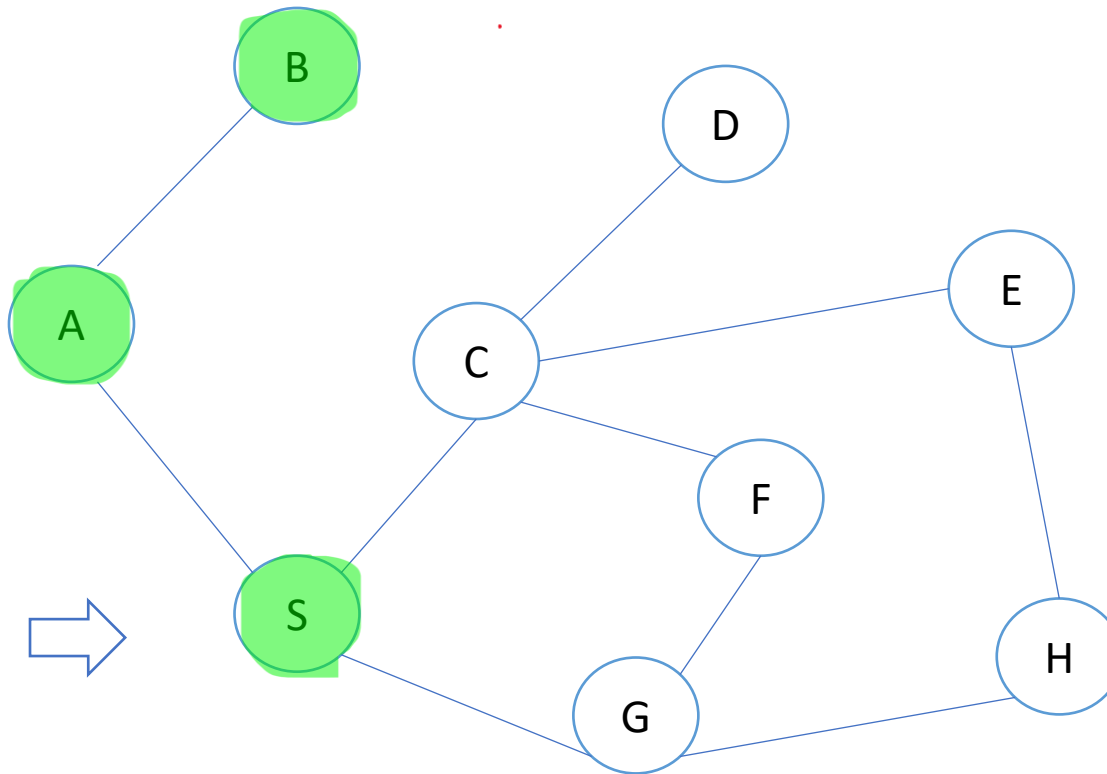
Visited node

Keeping track of
all visited nodes



Queue

Breadth First Search (BFS)



Output: A, B, S

Visited node

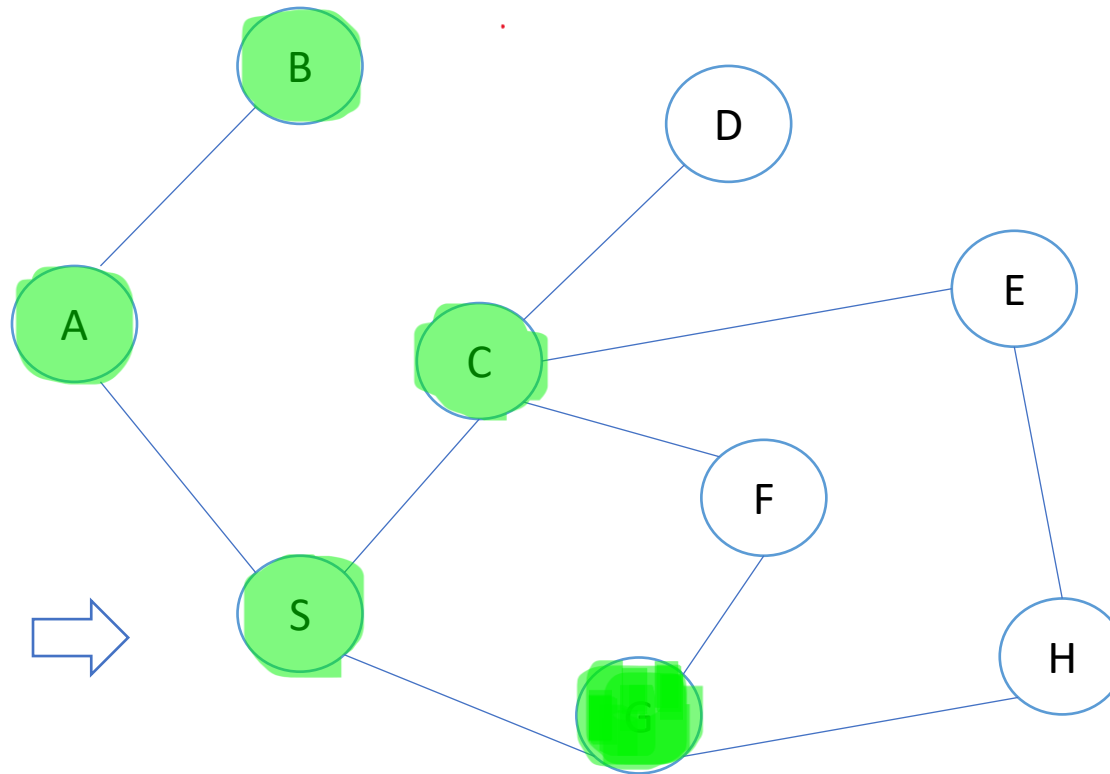
Keeping track of
all visited nodes



Dequeue S

Queue

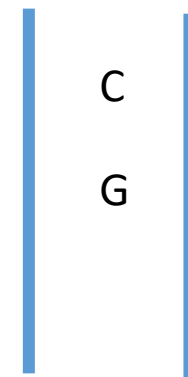
Breadth First Search (BFS)



Output: A, B, S, C, G

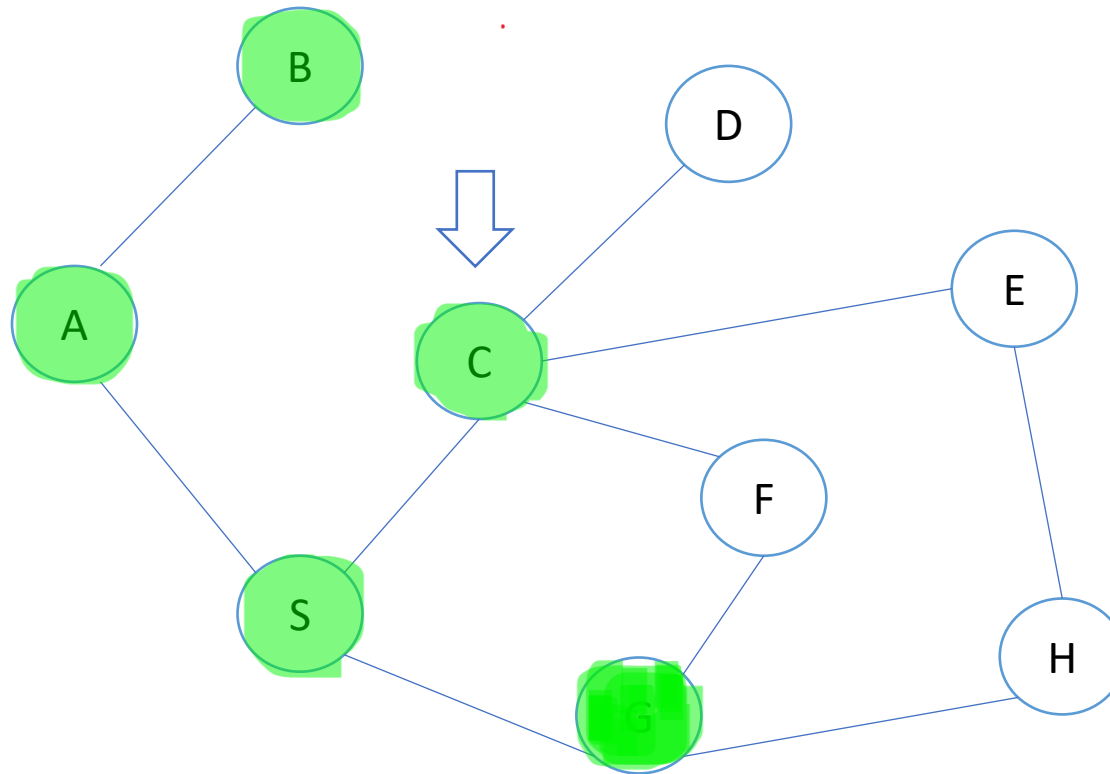
 Visited node

Keeping track of
all visited nodes



Queue

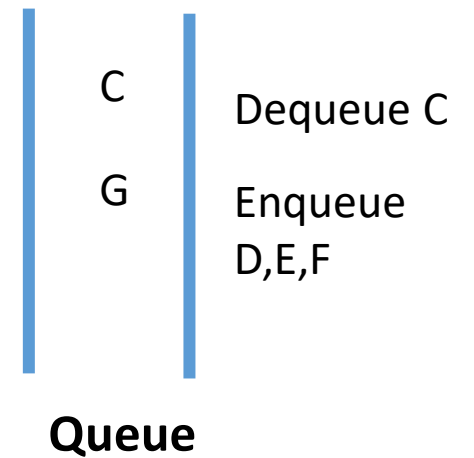
Breadth First Search (BFS)



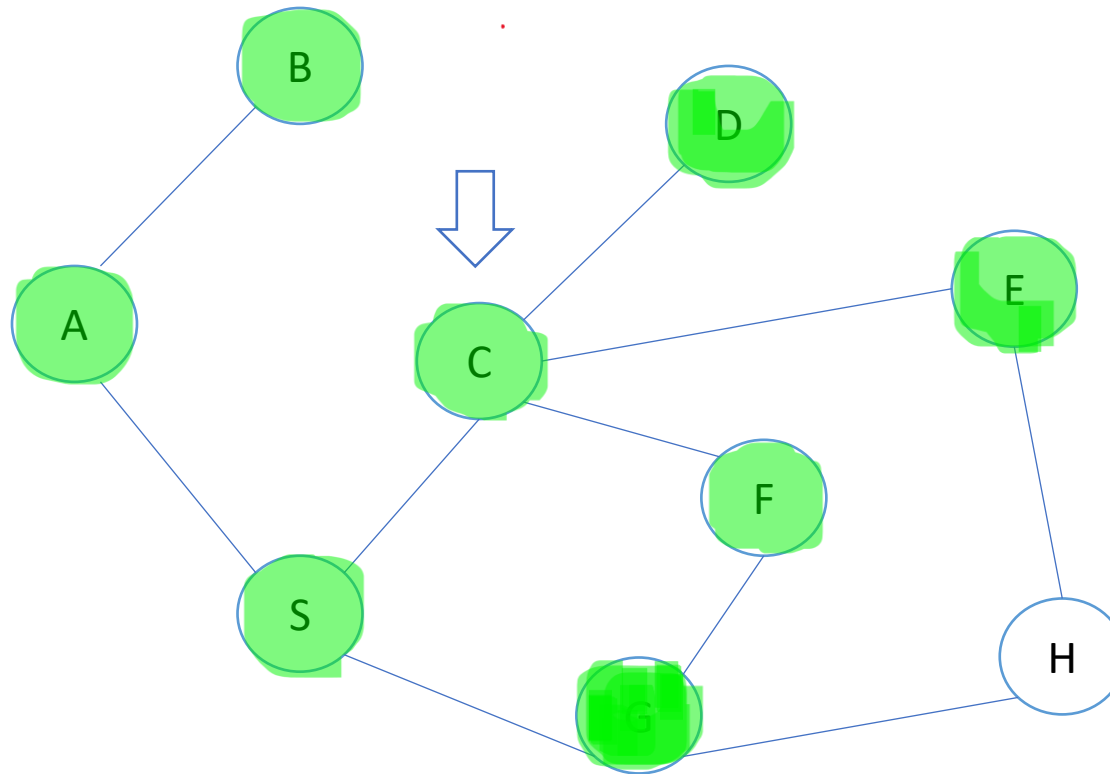
Output: A, B, S, C, G

Visited node

Keeping track of
all visited nodes



Breadth First Search (BFS)



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

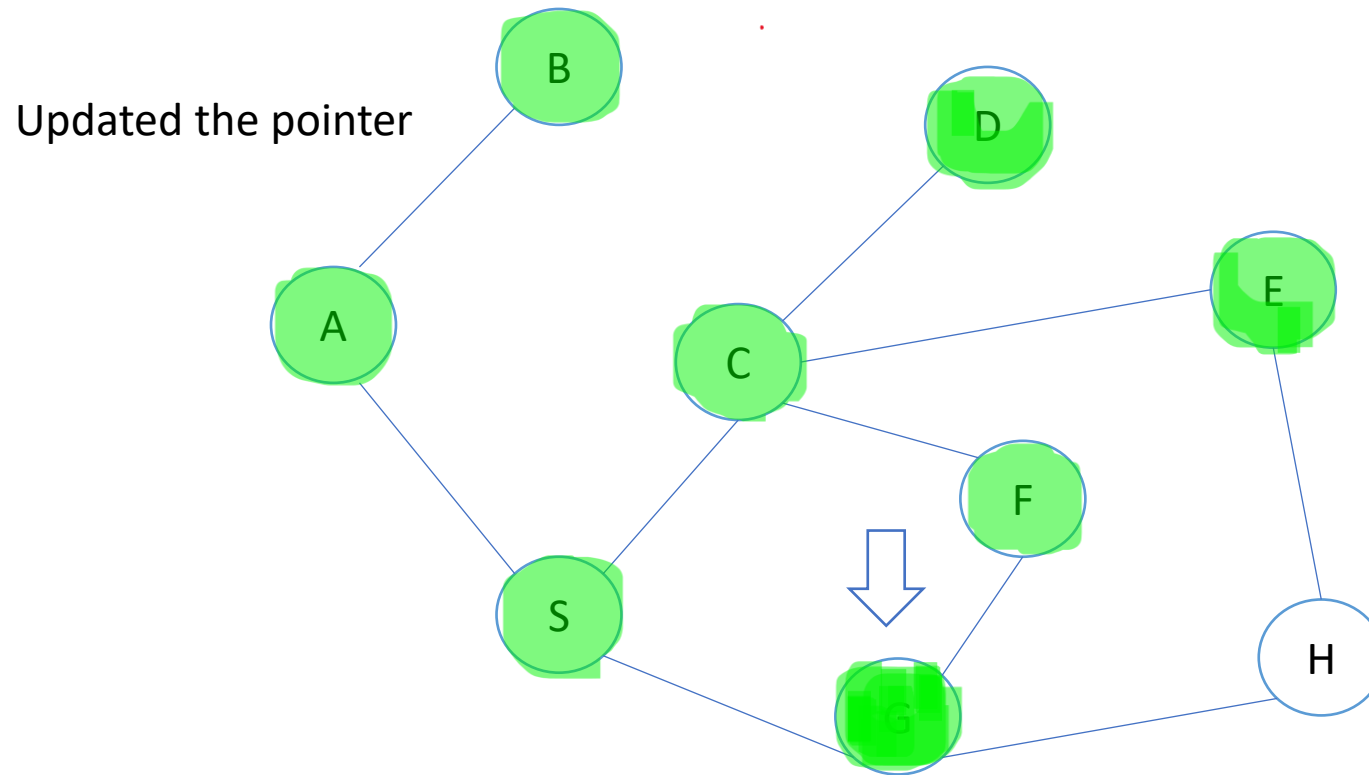
Visited node

Keeping track of
all visited nodes

G
D
E
F

Queue

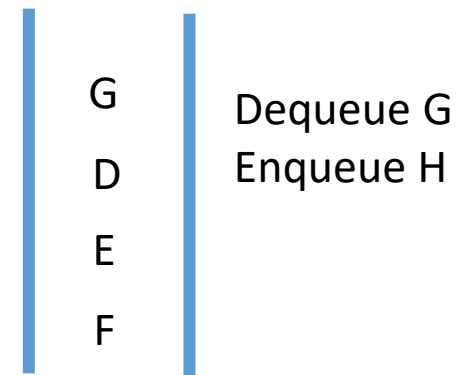
Breadth First Search (BFS)



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

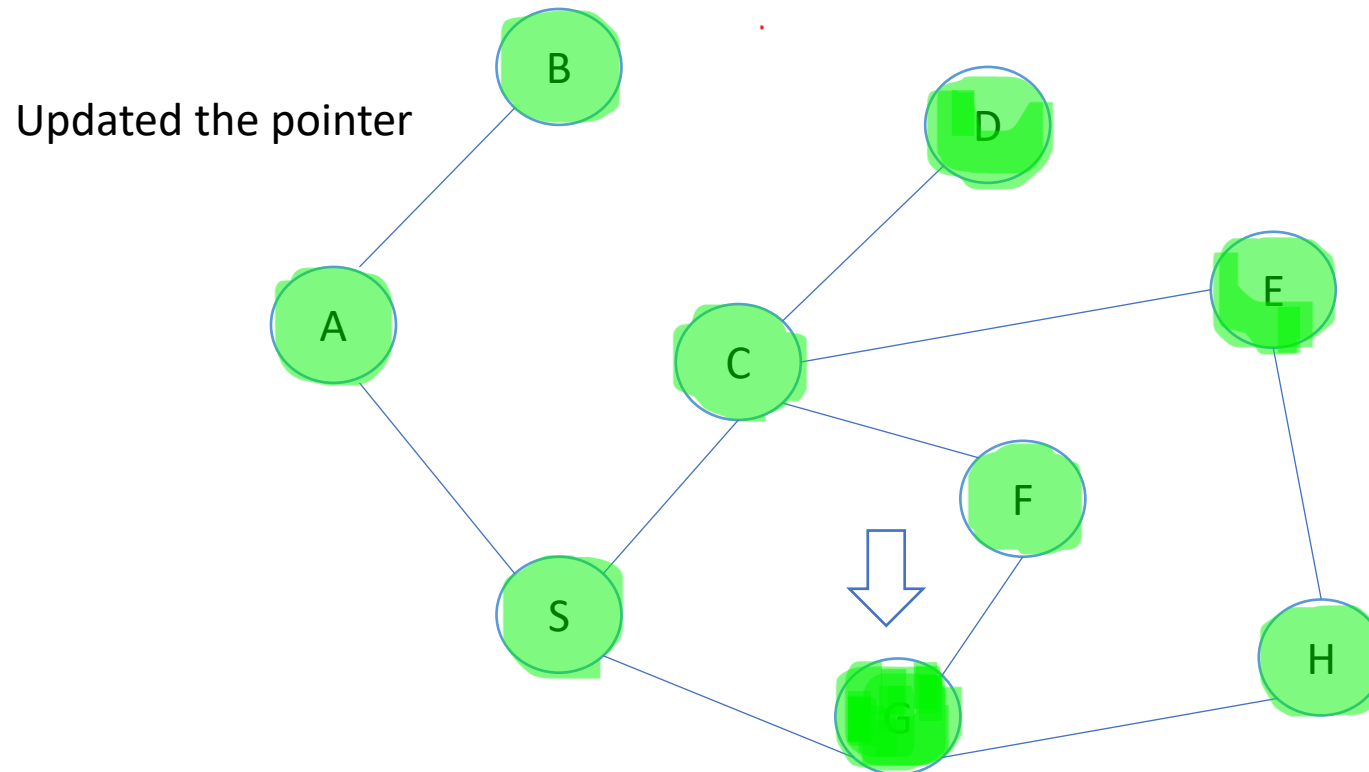
Visited node

Keeping track of
all visited nodes



Queue

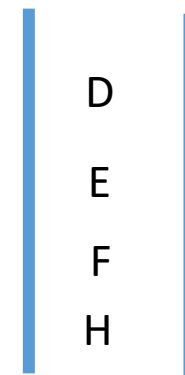
Breadth First Search (BFS)



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

Visited node

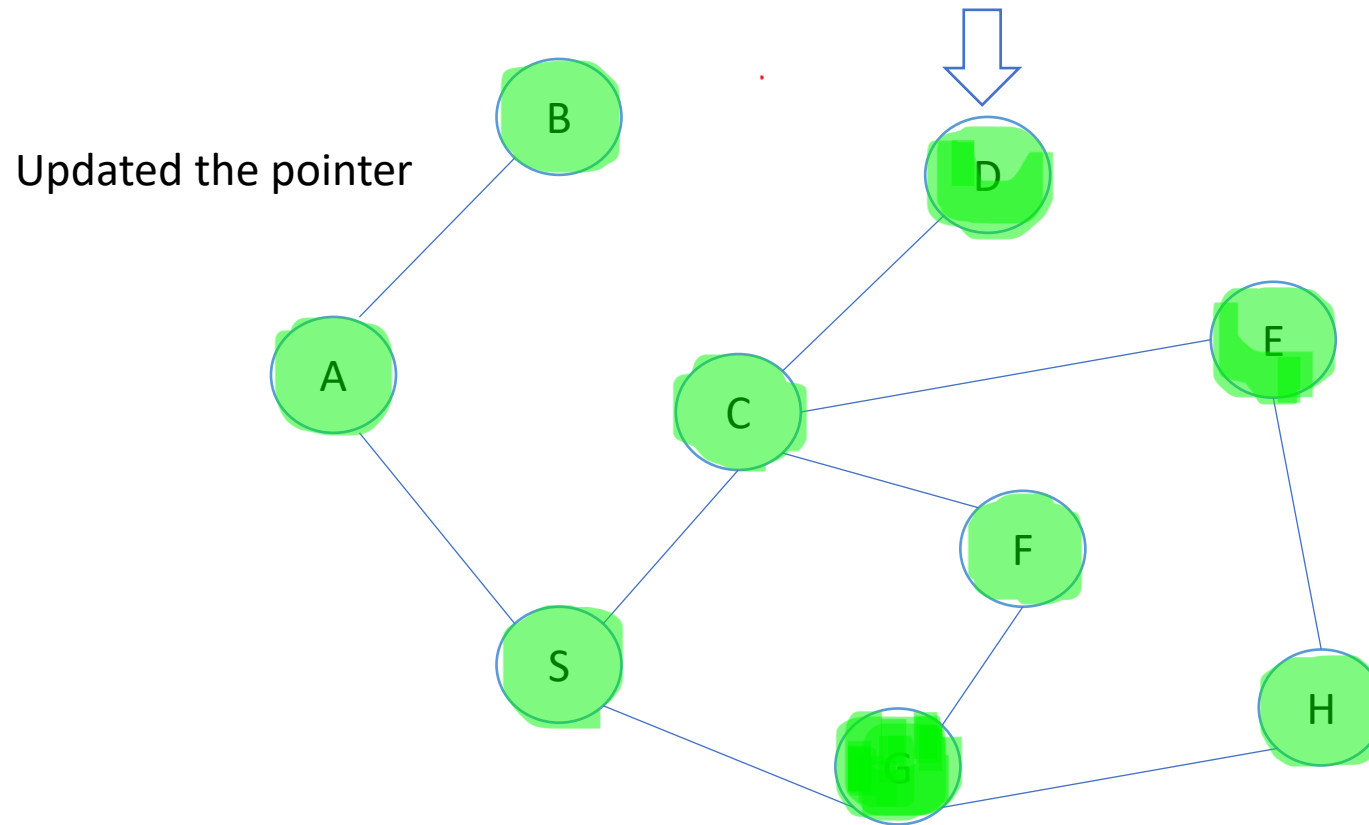
Keeping track of
all visited nodes



Dequeue D
And others

Queue

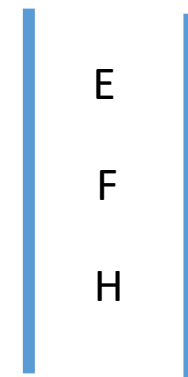
Breadth First Search (BFS)



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

 Visited node

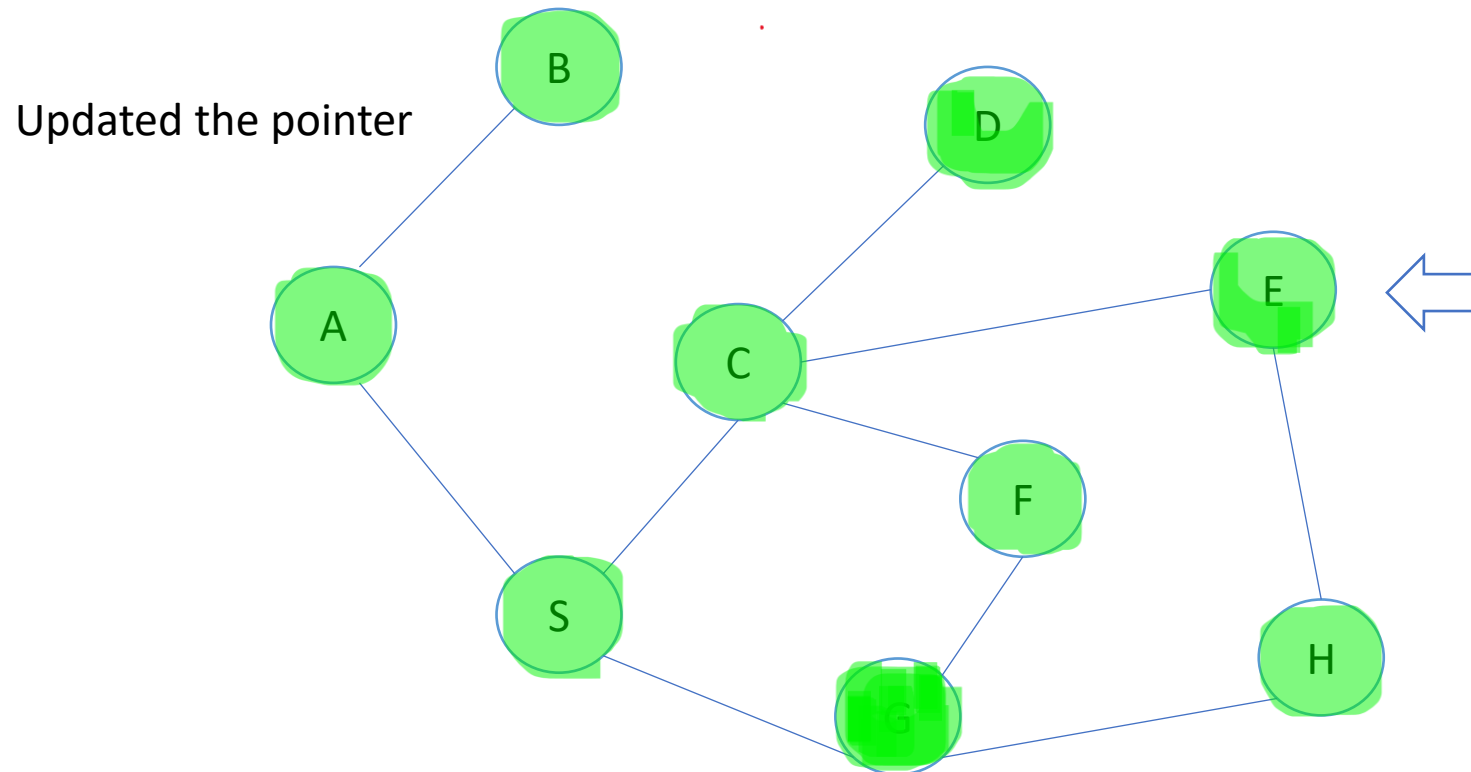
Keeping track of
all visited nodes



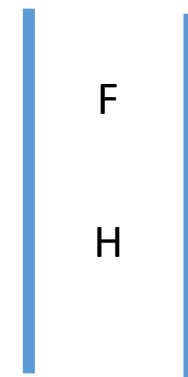
Queue

Breadth First Search (BFS)

 Visited node



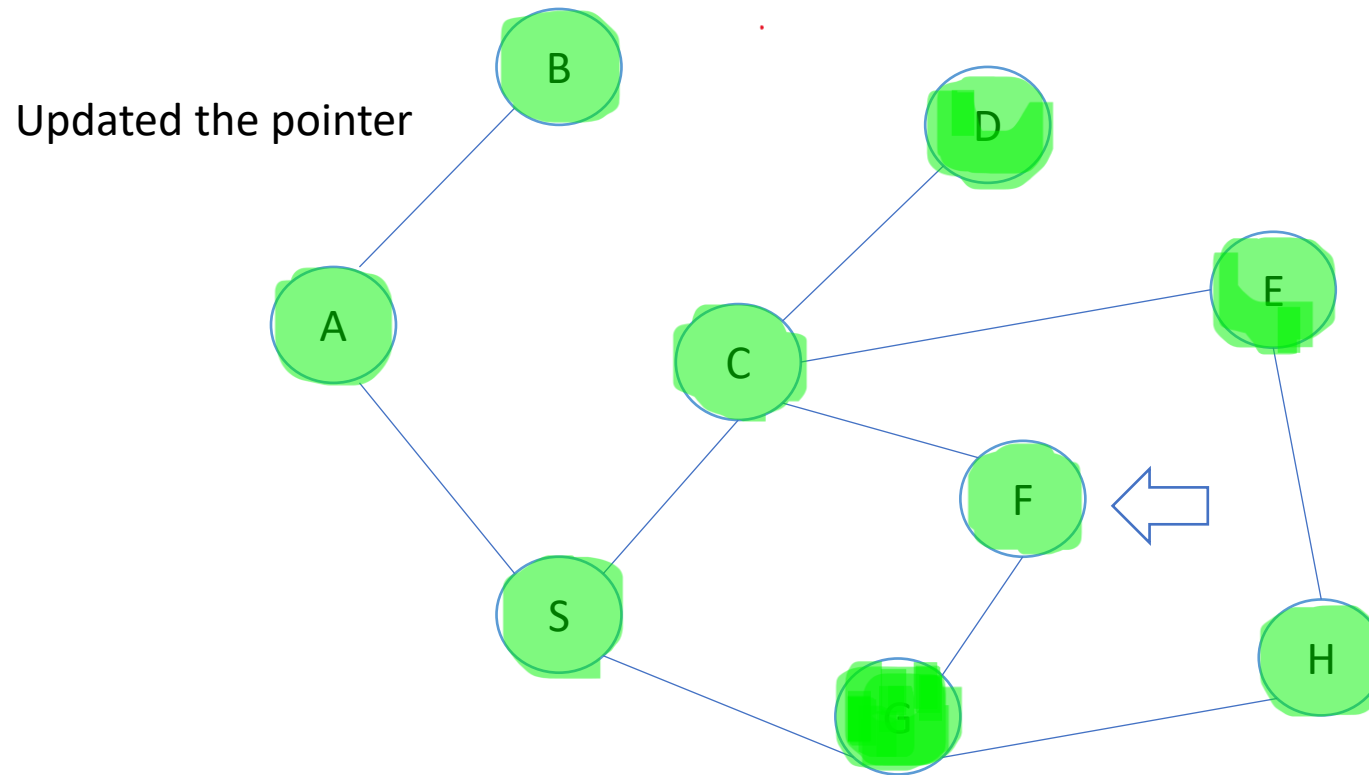
Keeping track of
all visited nodes



Queue

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

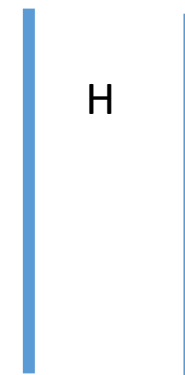
Breadth First Search (BFS)



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

Visited node

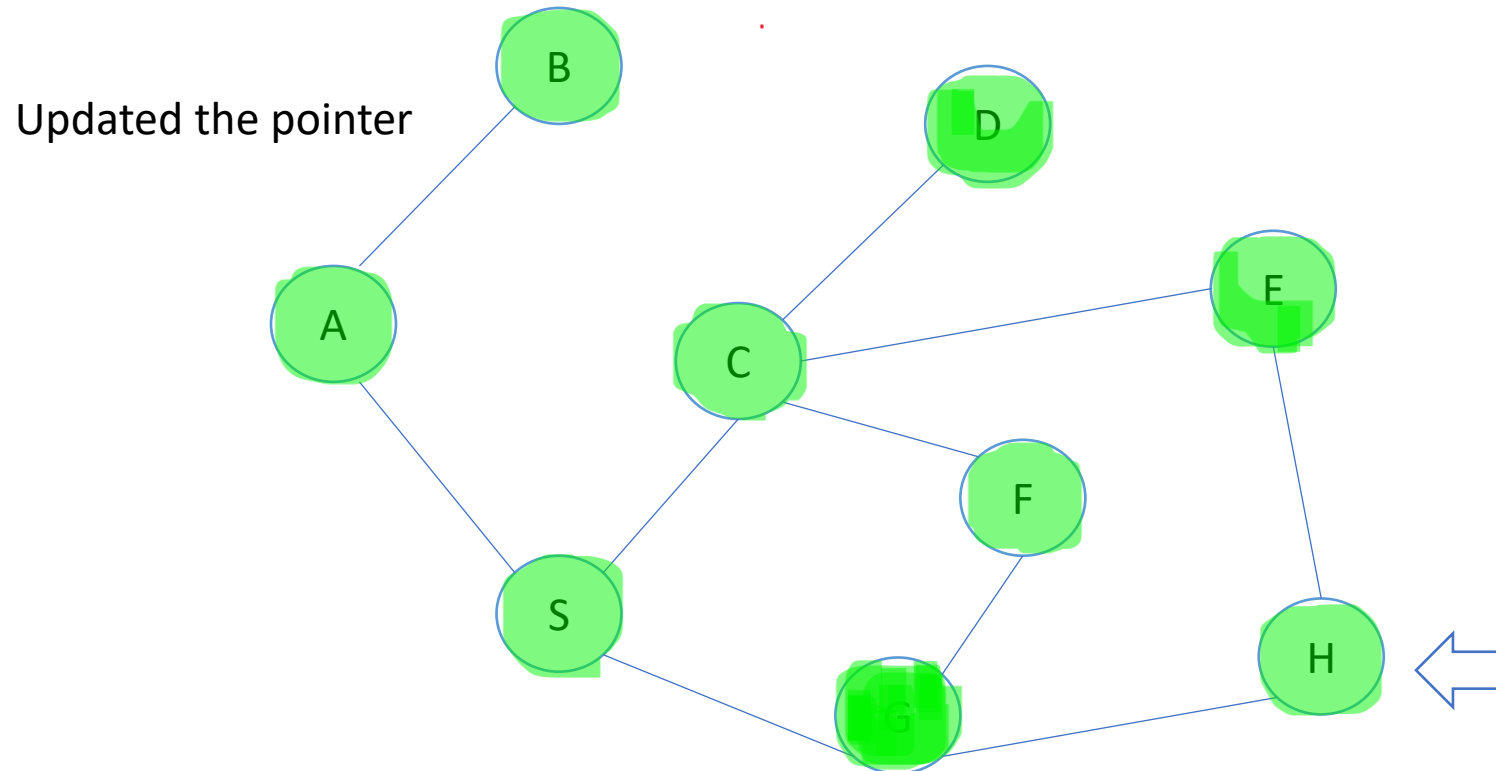
Keeping track of
all visited nodes



Queue

Breadth First Search (BFS)

 Visited node



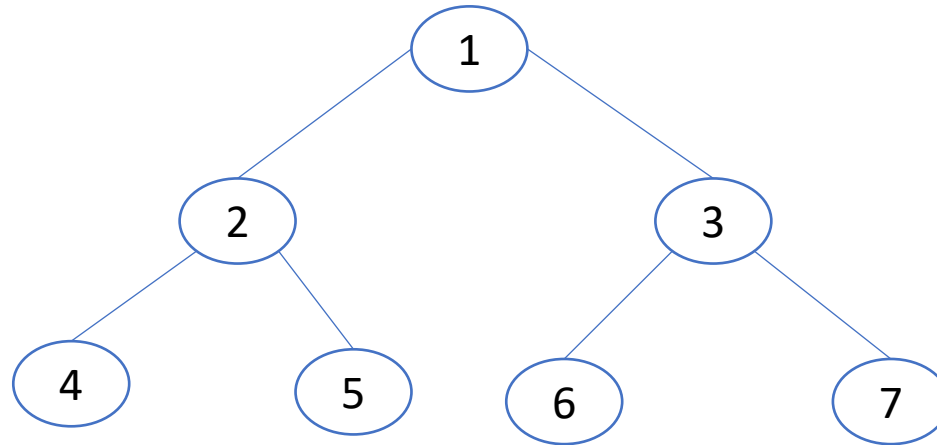
Output: A, B, S, C, G, D, E, F, H

Keeping track of
all visited nodes



Queue

DFS & BFS in Tree



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

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