Revised: 7/31/2024

Graphs II

Michael C. Hackett

Assistant Professor, Computer Science

COMMUNITY COLLEGE OF PHILADELPHIA

Lecture Topics

- Graph Traversals
  - Depth-First
  - Breadth-First
- Finding distance with breadth-first

Graph Traversal

- Graphs can be traversed using depth-first or breadth-first traversals.

- Similar to traversing a tree, but we can begin at any node in the graph.

- Depth-first traversal uses a *stack*
- Breadth-first traversal uses a *queue*

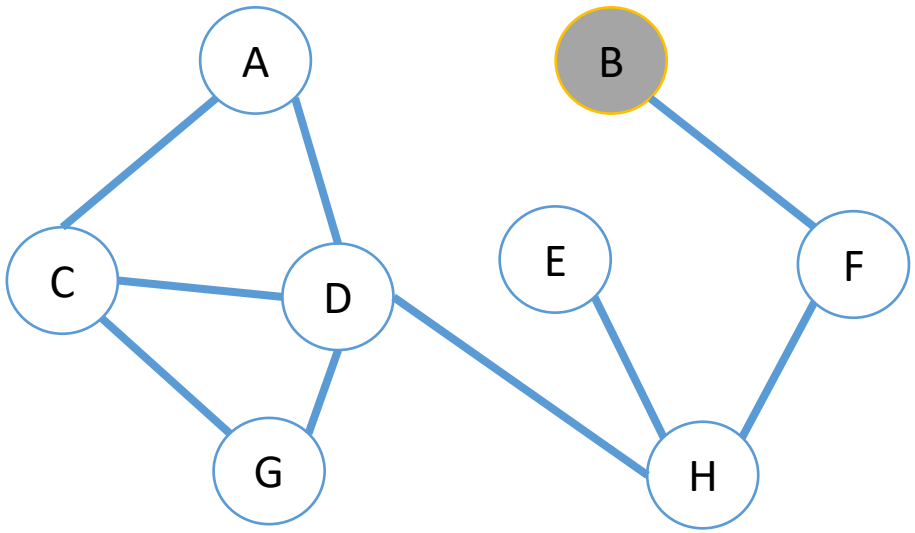
Depth-First Traversal

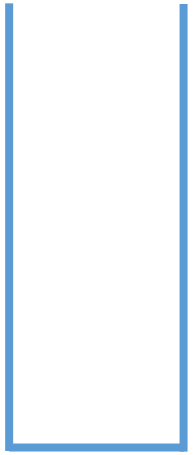
- Depth-first traversals work by coloring each node
  - White: Node has not been seen previously
  - Gray: Node has been seen previously
  - Black: Node has been visited and we are done with it

- This prevents getting stuck in a cycle, if one existed
  - The traversal is the same for bi-directional graphs or digraphs

Depth-First Traversal

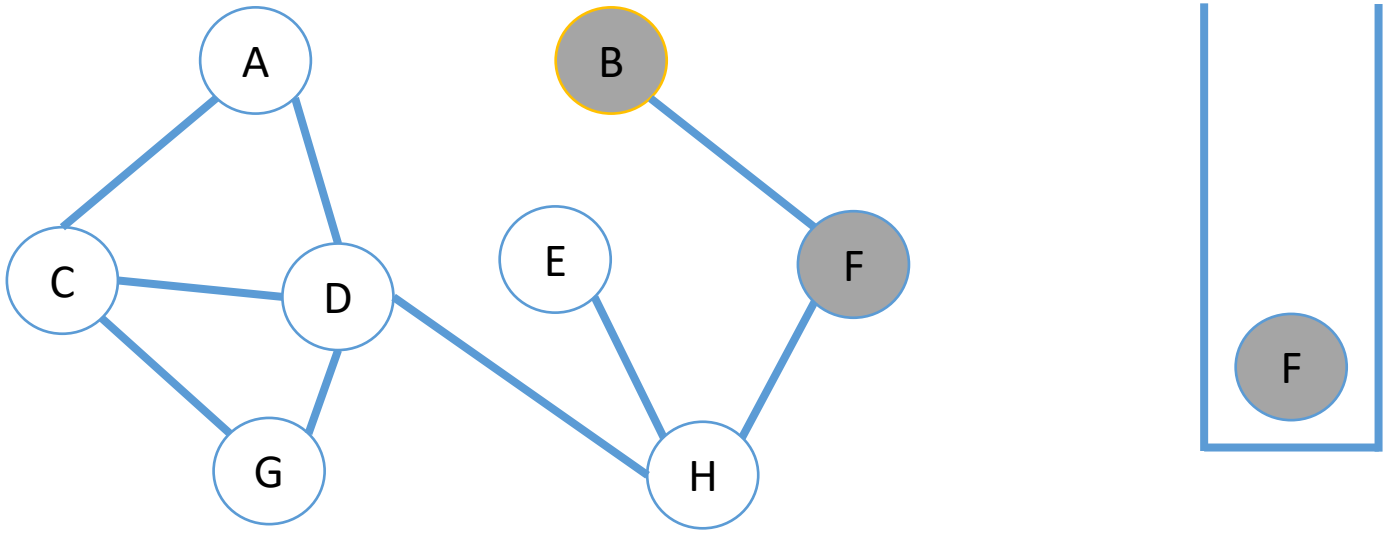
- Starting at B
- Order visited: N/A





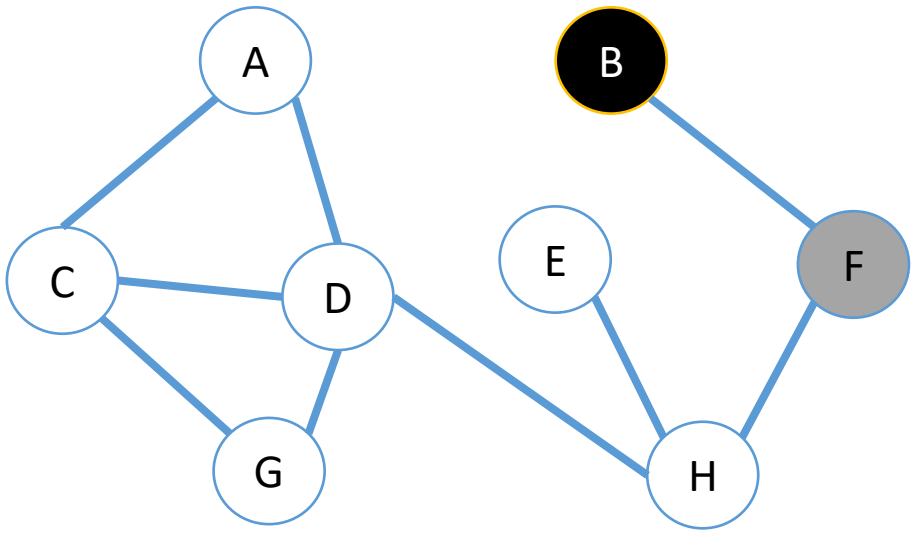
Depth-First Traversal

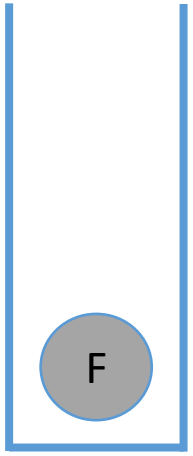
- Adds its adjacent white node(s) to the stack (turning the white node(s) gray)
- Order visited: N/A



Depth-First Traversal

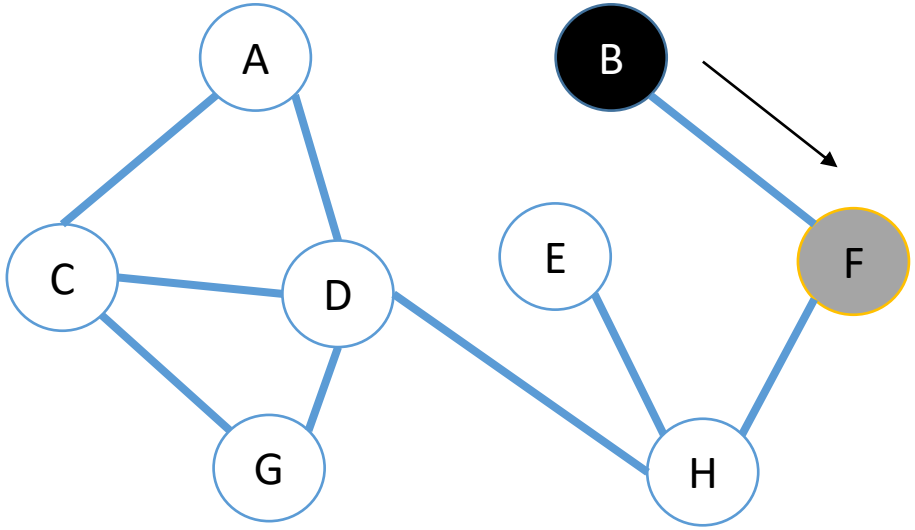
- B was visited
  - Changed to black
- Order visited: B

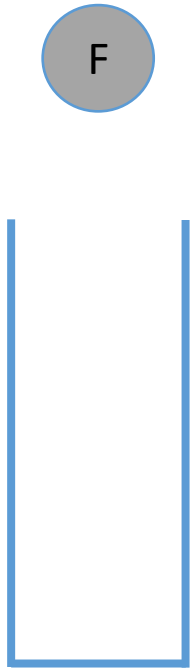




Depth-First Traversal

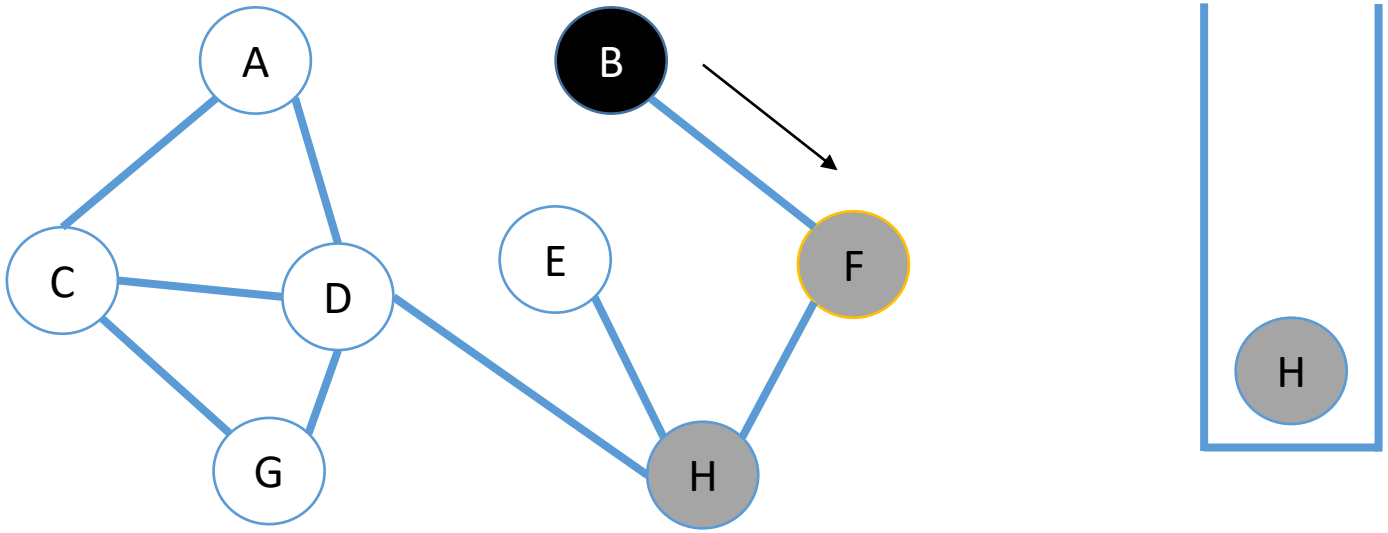
- Next node popped from stack
- Order visited: B





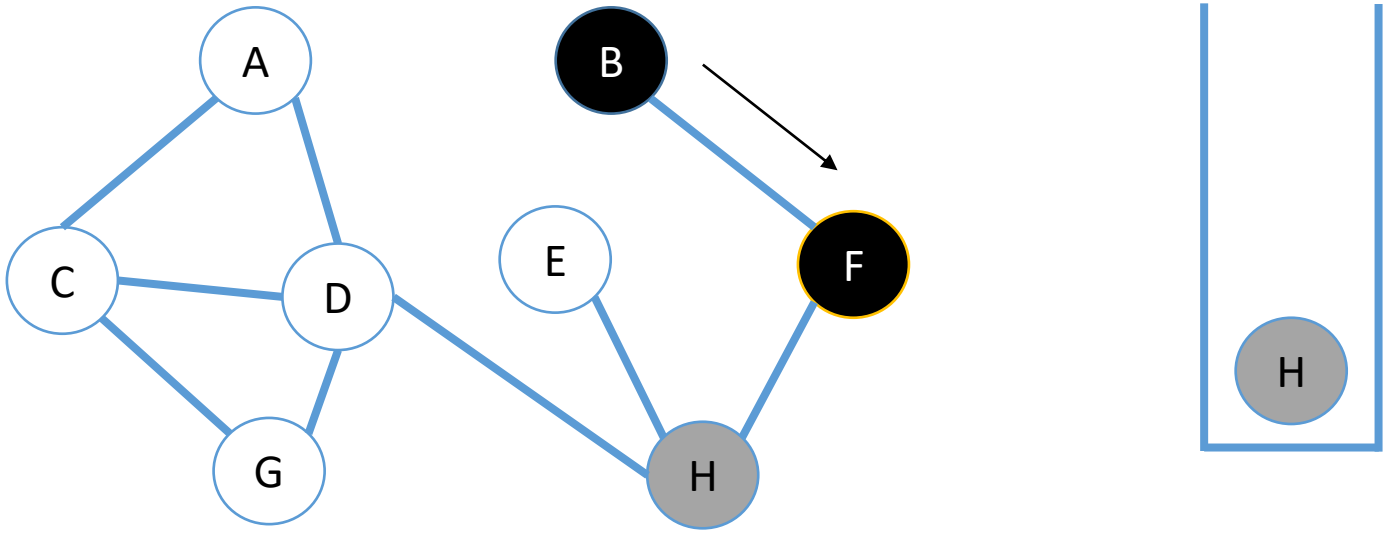
Depth-First Traversal

- Adds its adjacent white nodes to the stack (turning them gray)
- Order visited: B



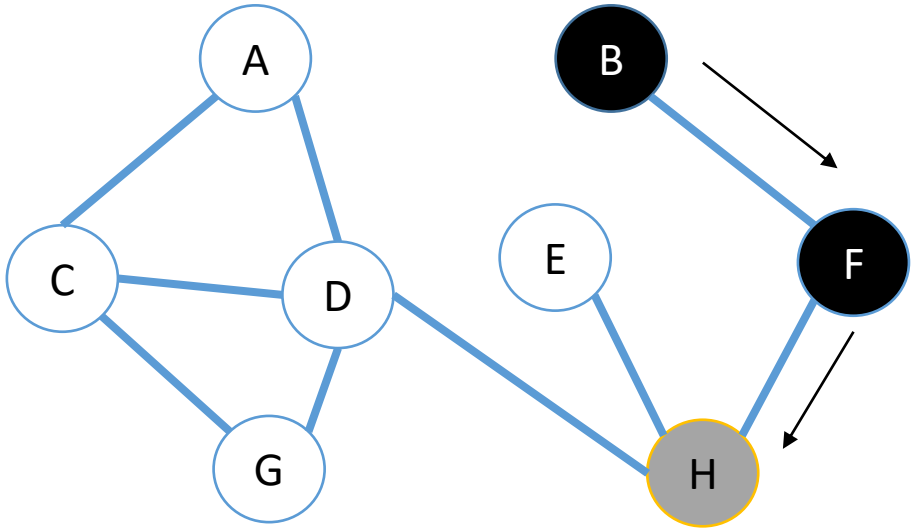
Depth-First Traversal

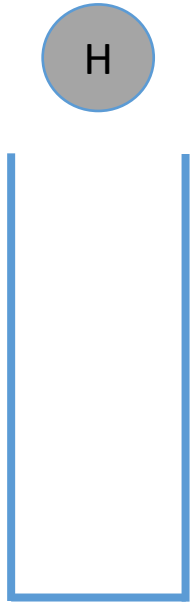
- F was visited
- Order visited: B, F



Depth-First Traversal

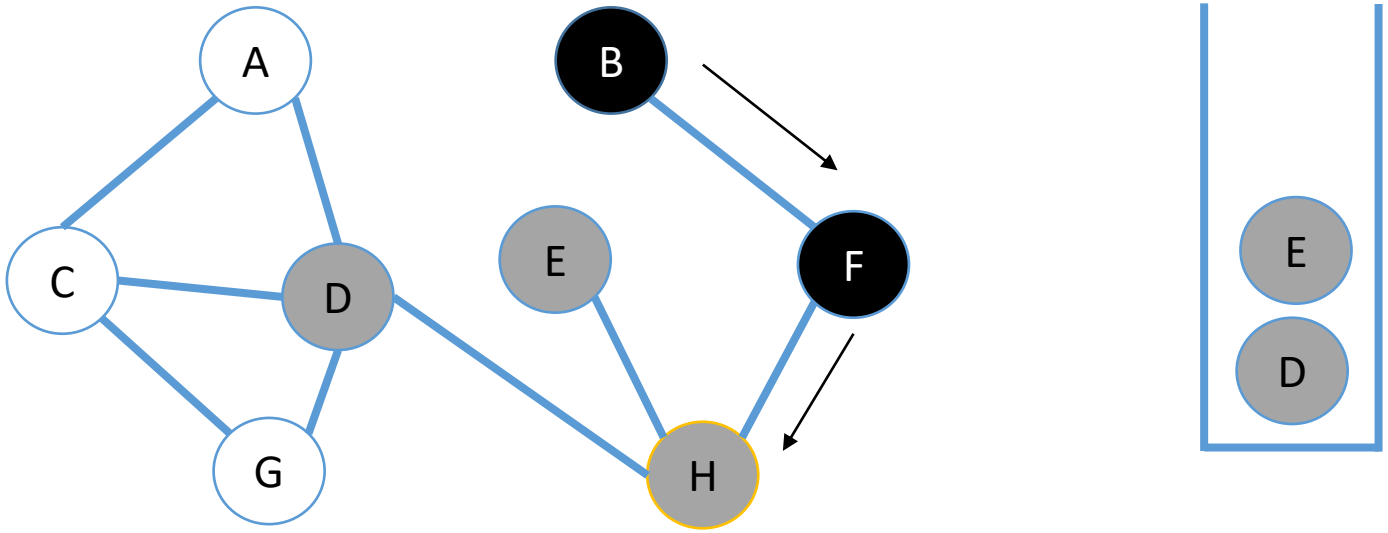
- Next node popped from stack
- Order visited: B, F





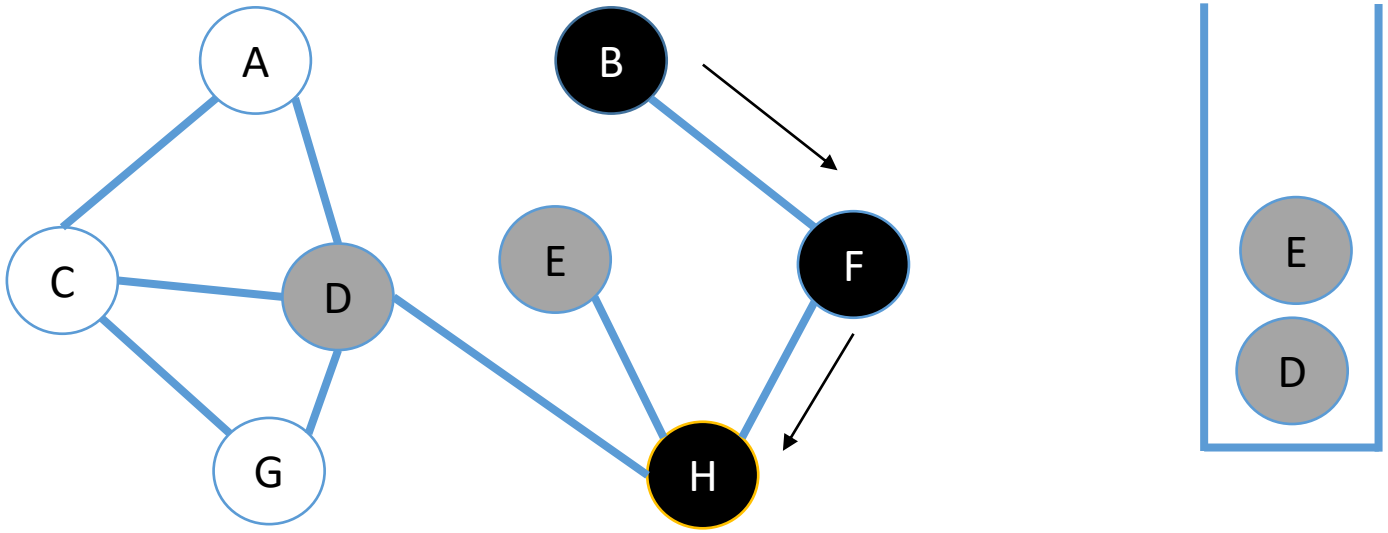
Depth-First Traversal

- Adds its adjacent white nodes to the stack
- Order visited: B, F



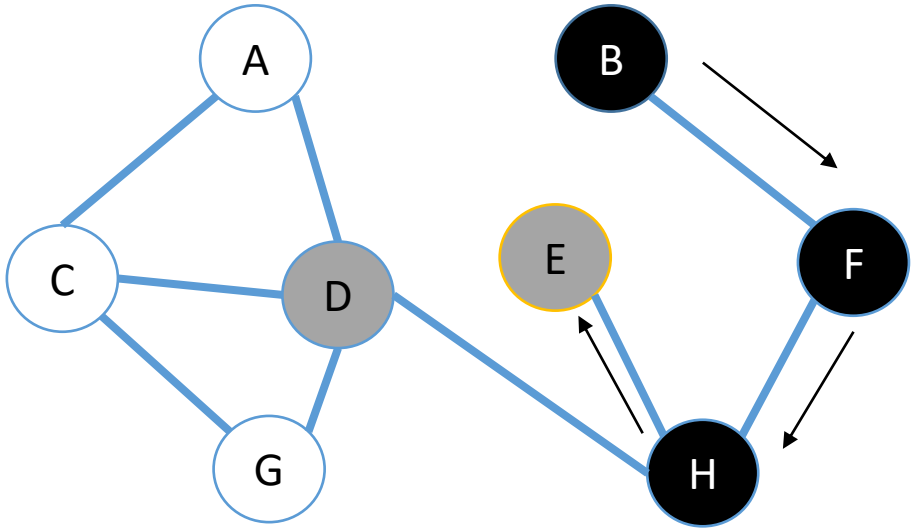
Depth-First Traversal

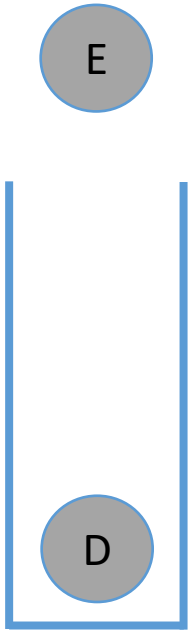
- H was visited
- Order visited: B, F, H



Depth-First Traversal

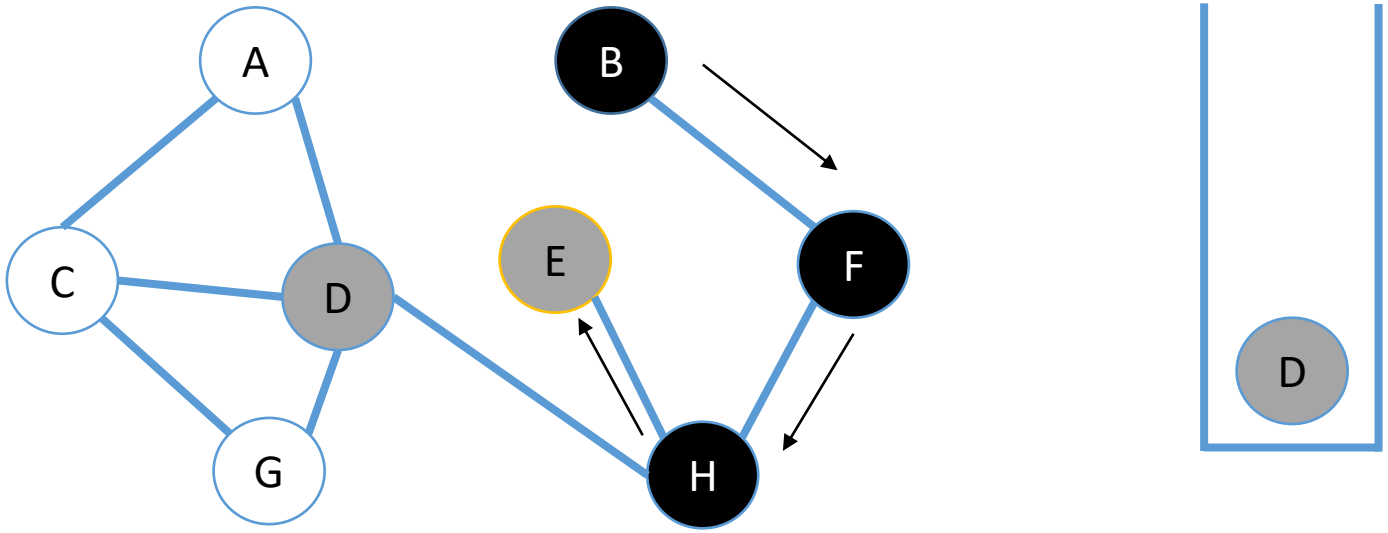
- Next node popped from stack
- Order visited: B, F, H





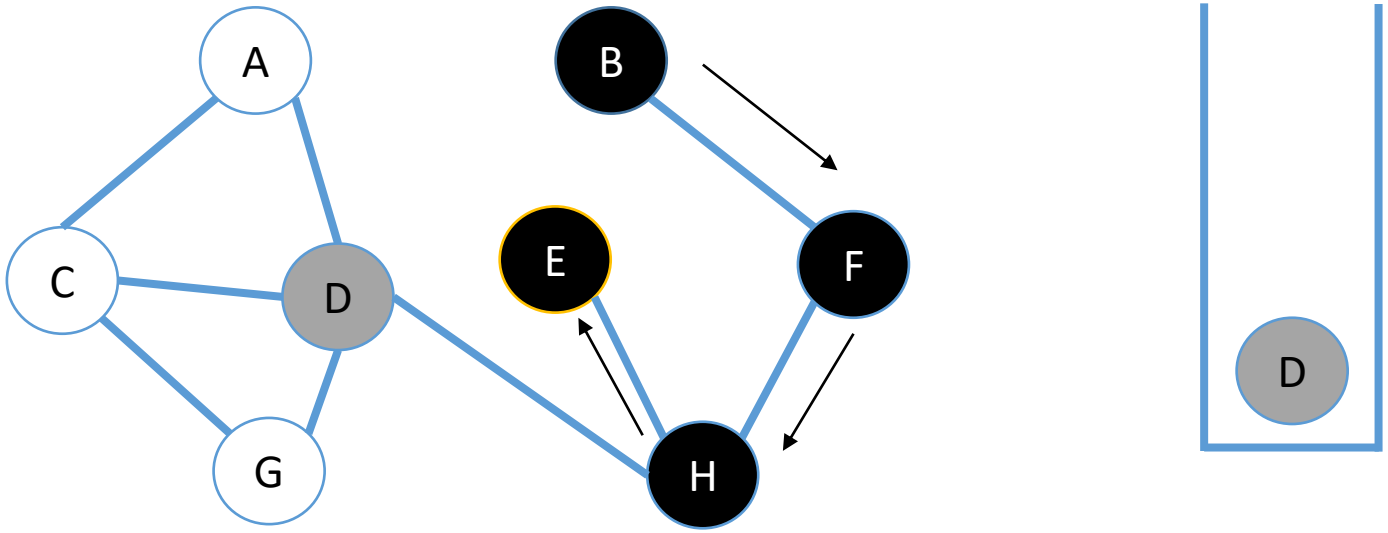
Depth-First Traversal

- Add its adjacent white nodes to the stack (there are none)
- Order visited: B, F, H



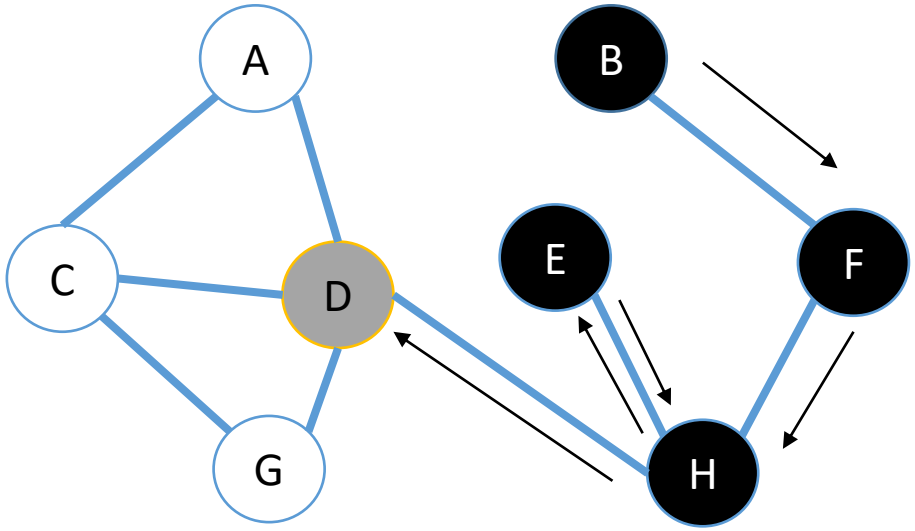
Depth-First Traversal

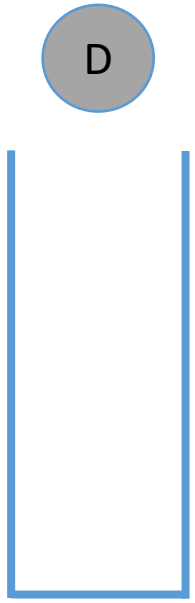
- E was visited
- Order visited: B, F, H, E



Depth-First Traversal

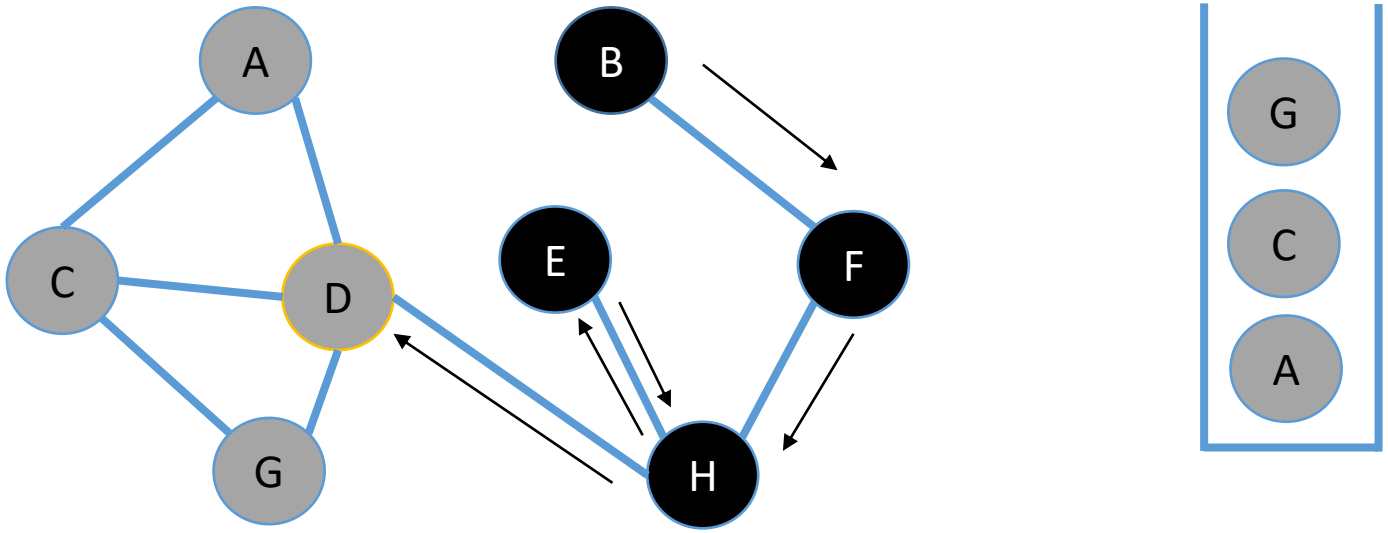
- Next node popped from stack
- Order visited: B, F, H, E





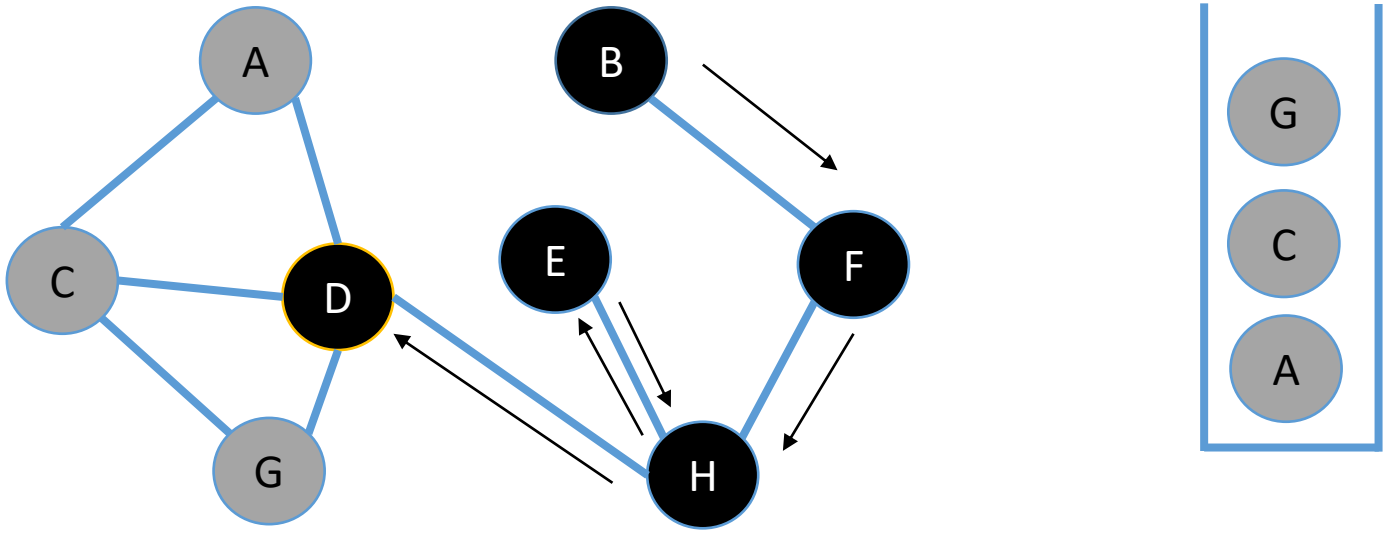
Depth-First Traversal

- Add its adjacent white nodes to the stack
- Order visited: B, F, H, E



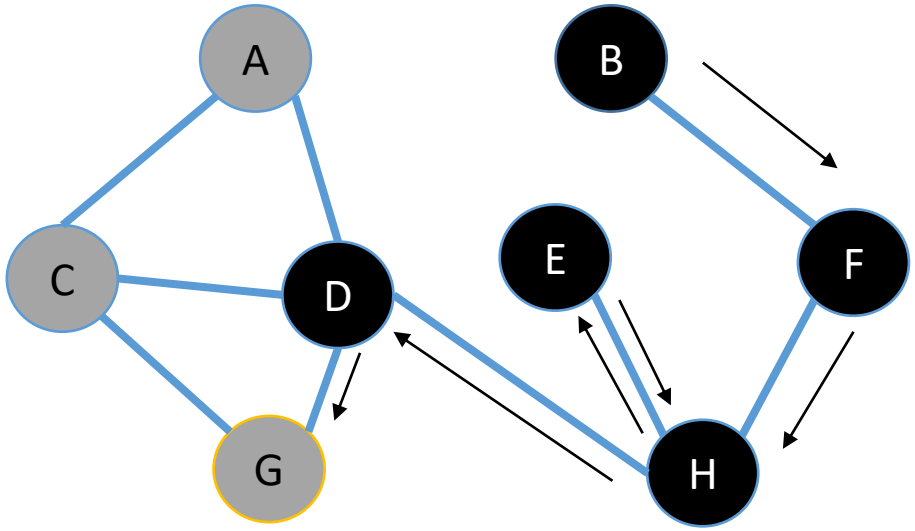
Depth-First Traversal

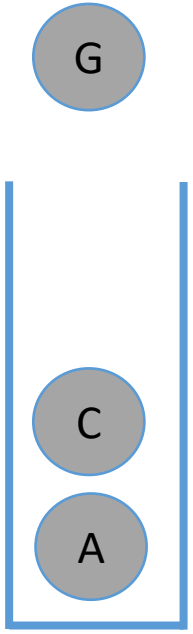
- D was visited
- Order visited: B, F, H, E, D



Depth-First Traversal

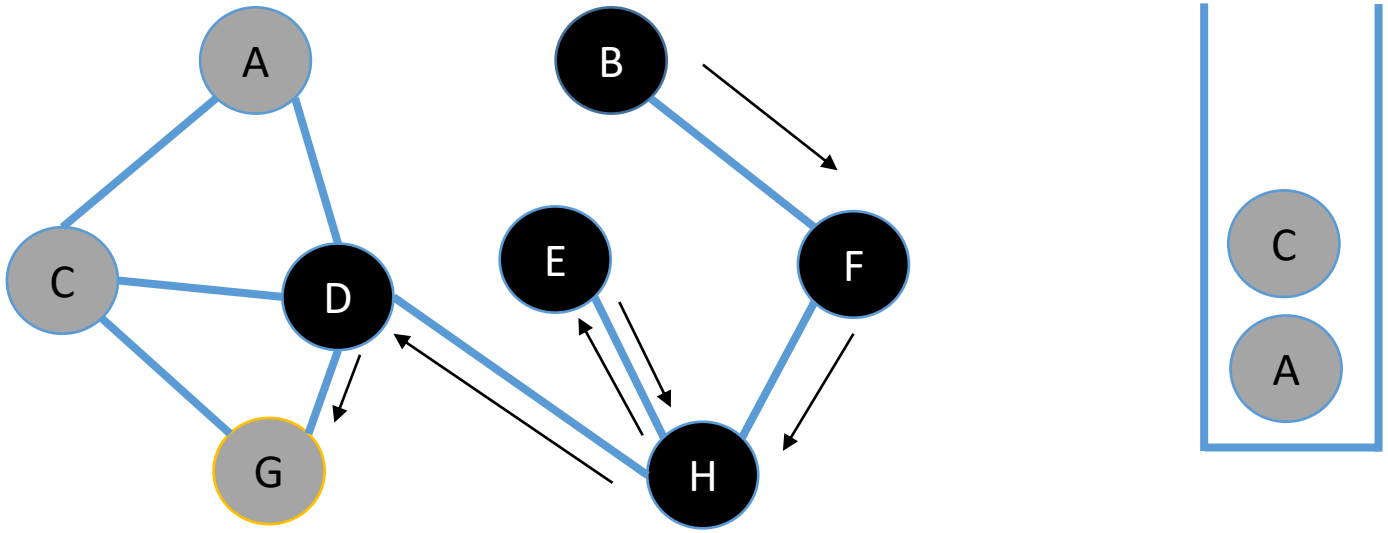
- Next node popped from the stack
- Order visited: B, F, H, E, D





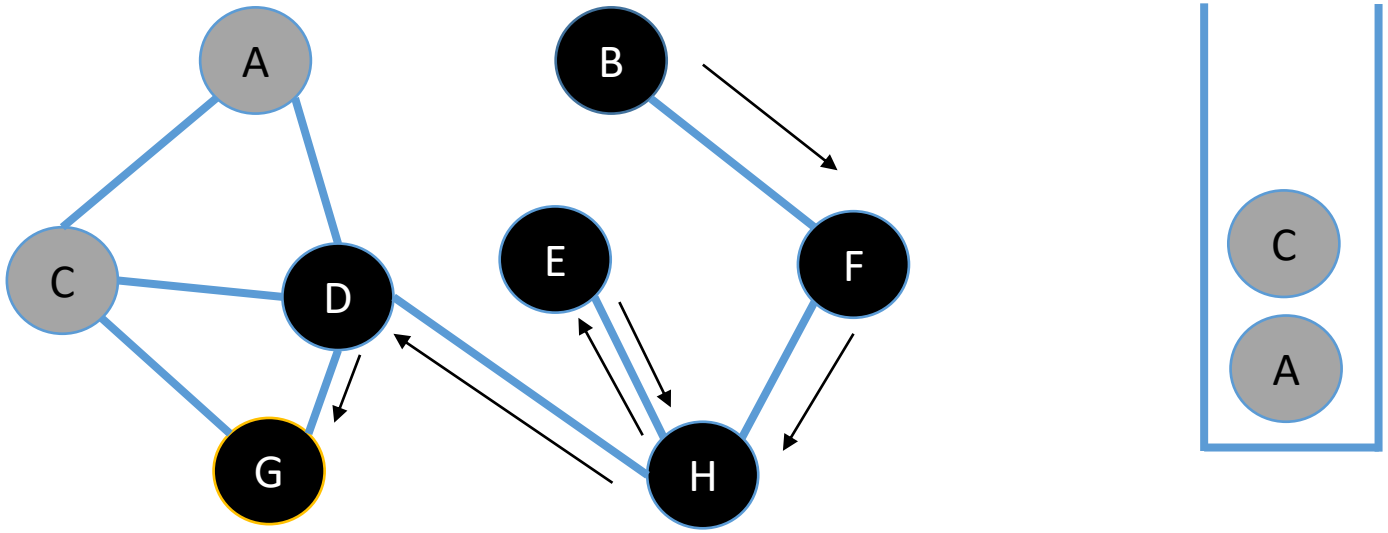
Depth-First Traversal

- Add its adjacent white nodes to the stack (there are none)
- Order visited: B, F, H, E, D



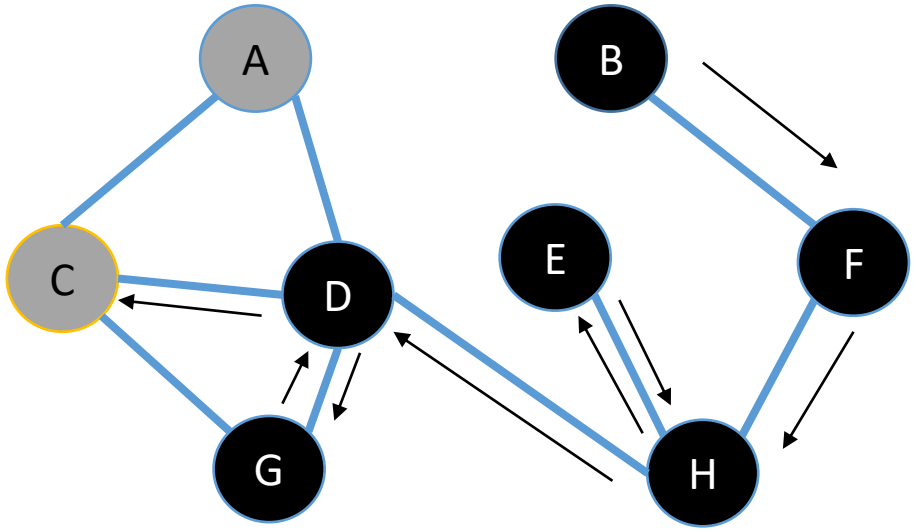
Depth-First Traversal

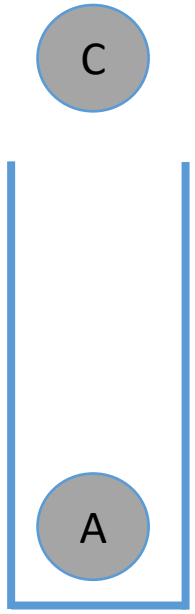
- G has been visited
- Order visited: B, F, H, E, D, G



Depth-First Traversal

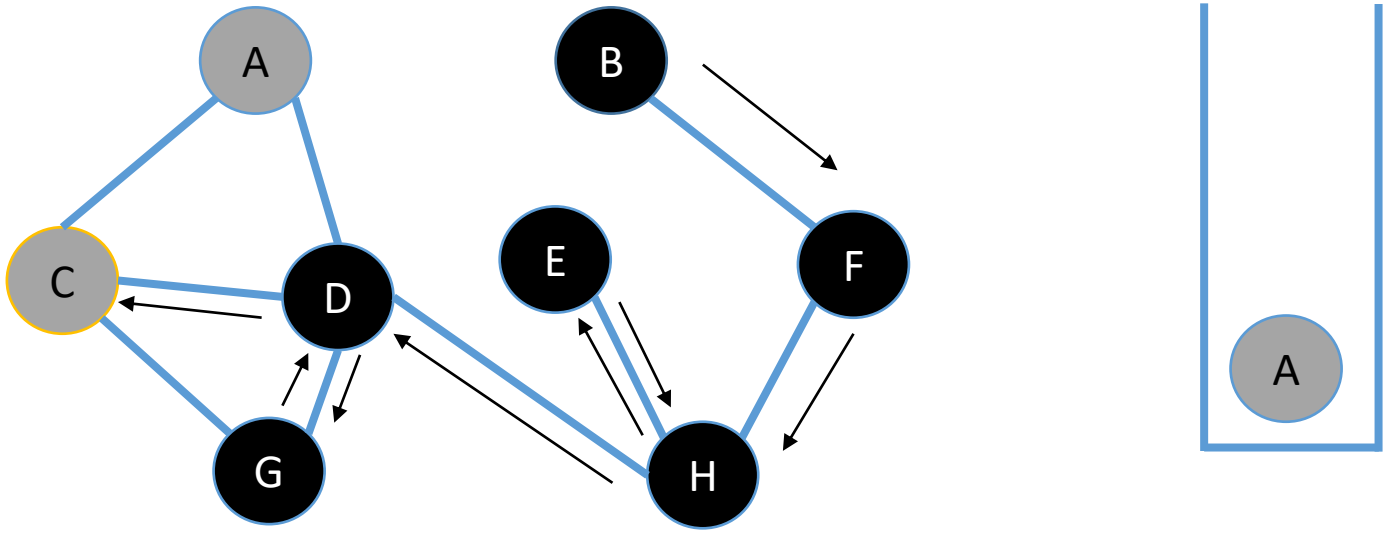
- Next node popped from the stack
- Order visited: B, F, H, E, D, G





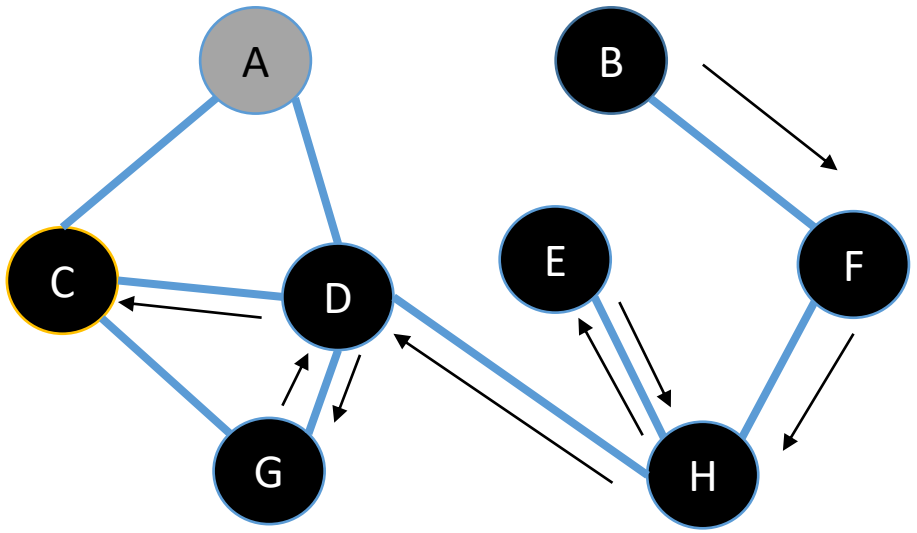
Depth-First Traversal

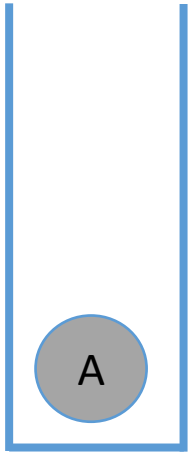
- Add its adjacent white nodes to the stack (there are none)
- Order visited: B, F, H, E, D, G



Depth-First Traversal

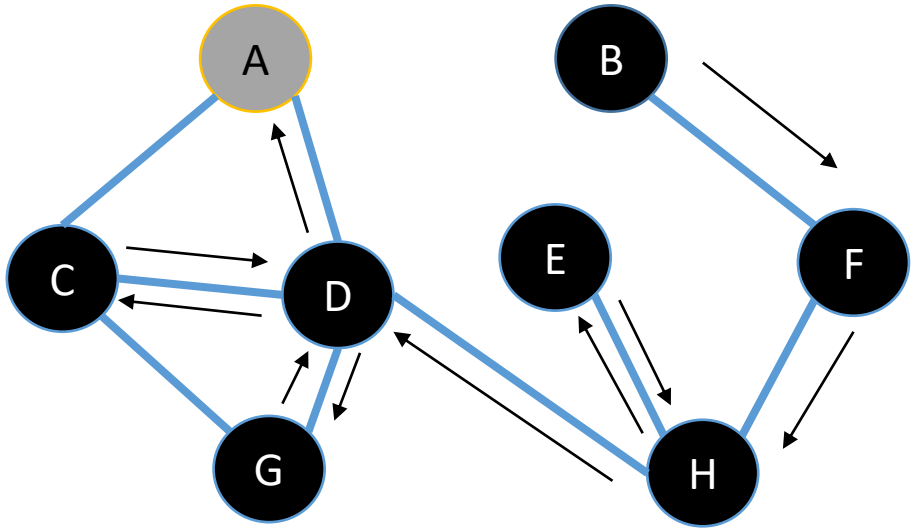
- C has been visited
- Order visited: B, F, H, E, D, G, C

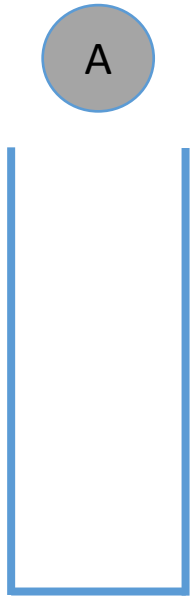




Depth-First Traversal

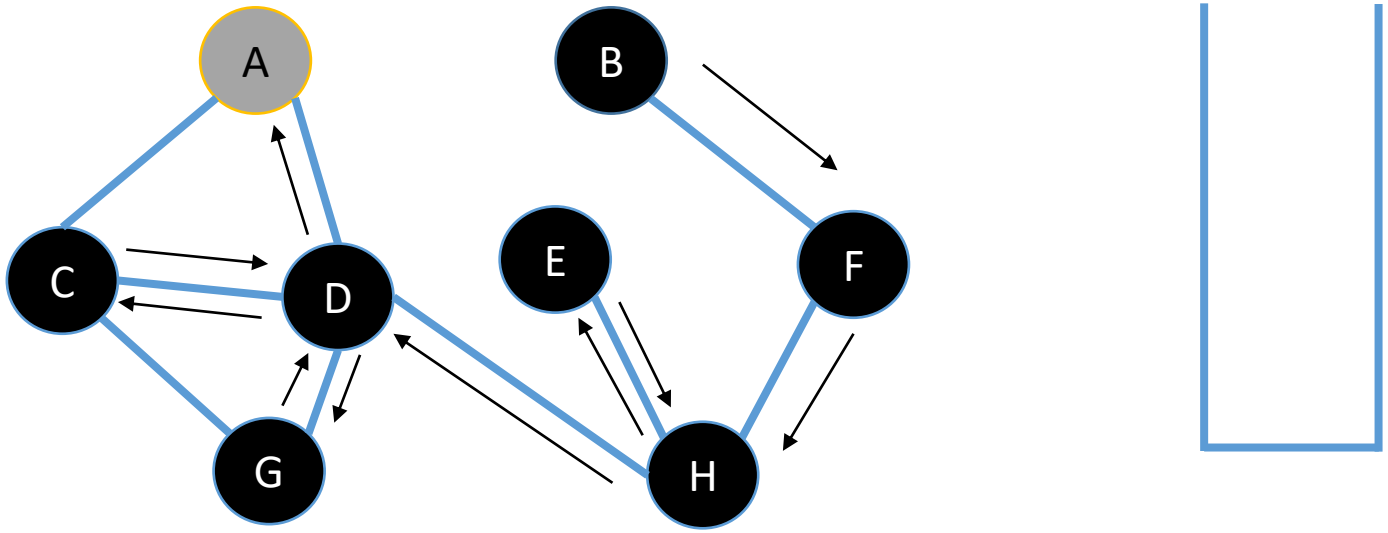
- Next node is popped from the stack
- Order visited: B, F, H, E, D, G, C





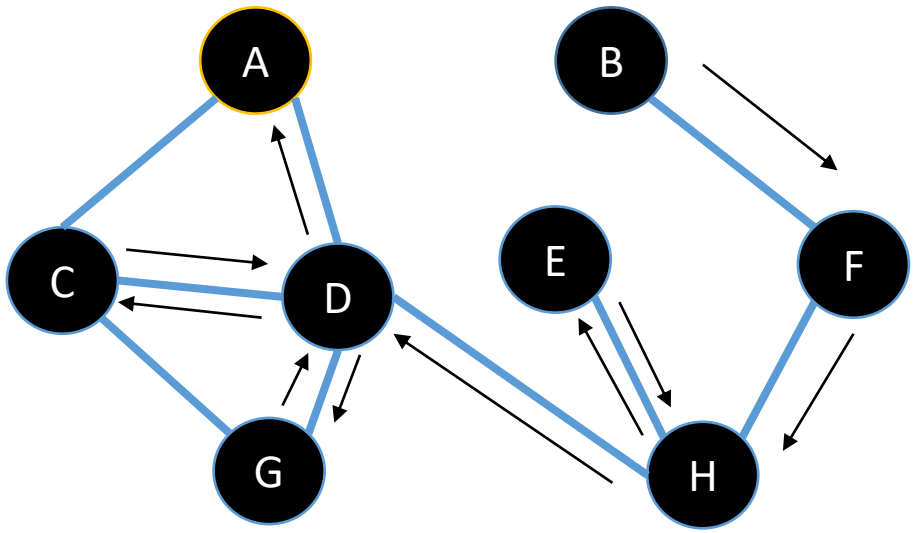
Depth-First Traversal

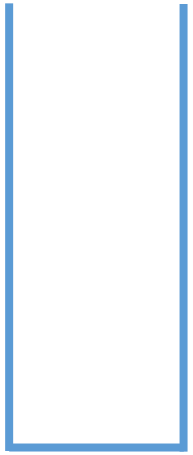
- Add its adjacent white nodes to the stack (there are none)
- Order visited: B, F, H, E, D, G, C



Depth-First Traversal

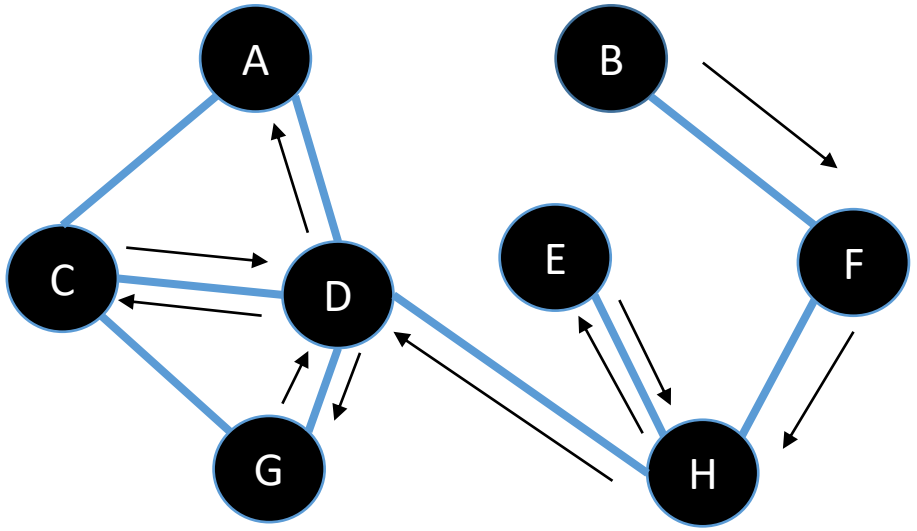
- A has been visited
- Order visited: B, F, H, E, D, G, C, A

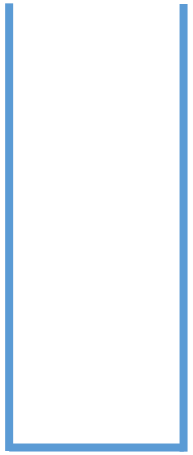




Depth-First Traversal

- Next node is popped from the stack (there are none)
  - Traversal is complete when the stack is empty
- Order visited: **B, F, H, E, D, G, C, A**



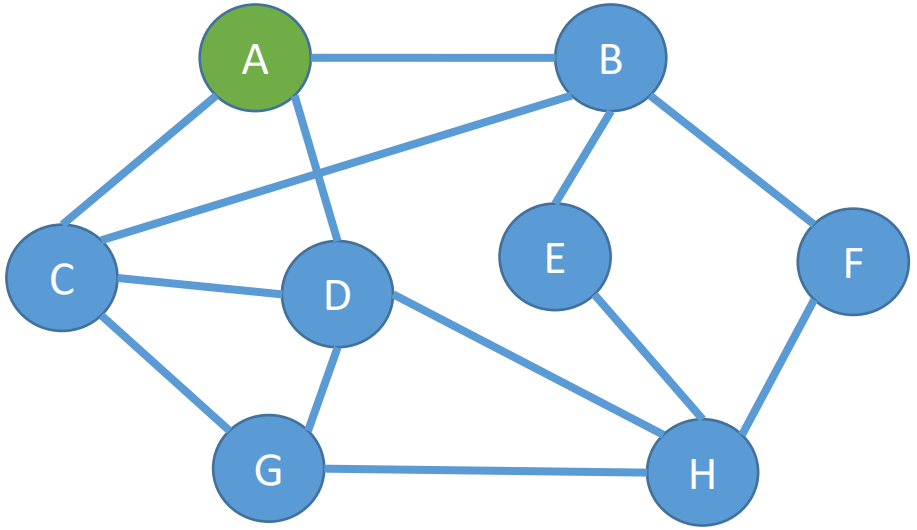


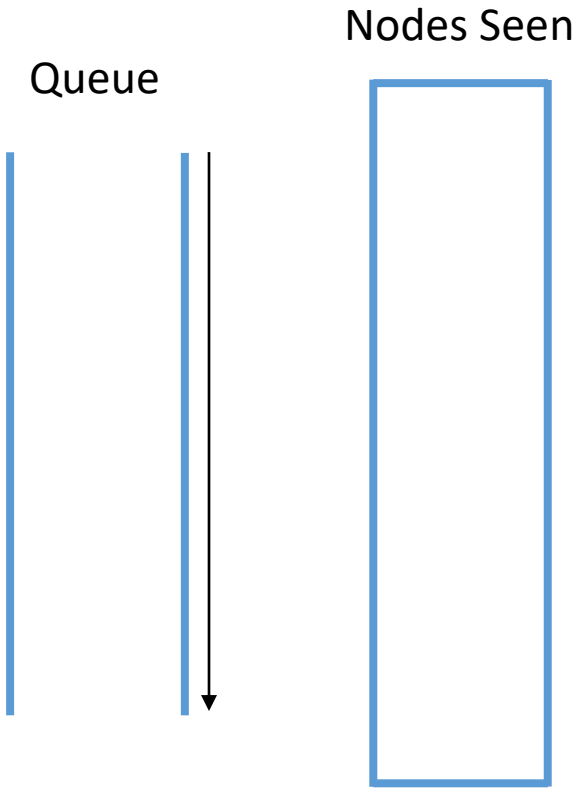
Breadth-First Traversal

- Breadth-first traversal of a graph is similar to the breadth-first traversal of a tree
- Uses a queue to manage the next nodes to visit
- Maintains an array of “seen” nodes to prevent getting stuck in a cycle, if one existed
  - The traversal process is the same for bi-directional graphs or digraphs

Breadth-First Traversal

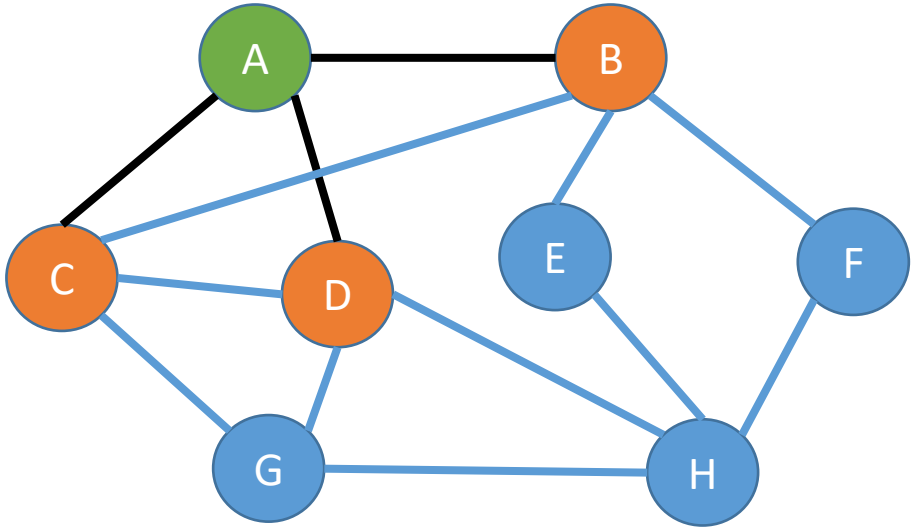
- Starting at A
- Order visited: N/A

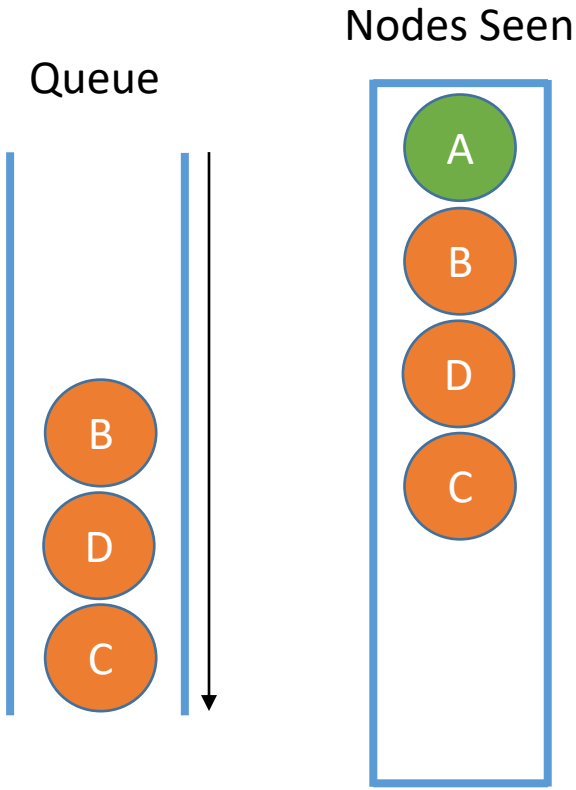




Breadth-First Traversal

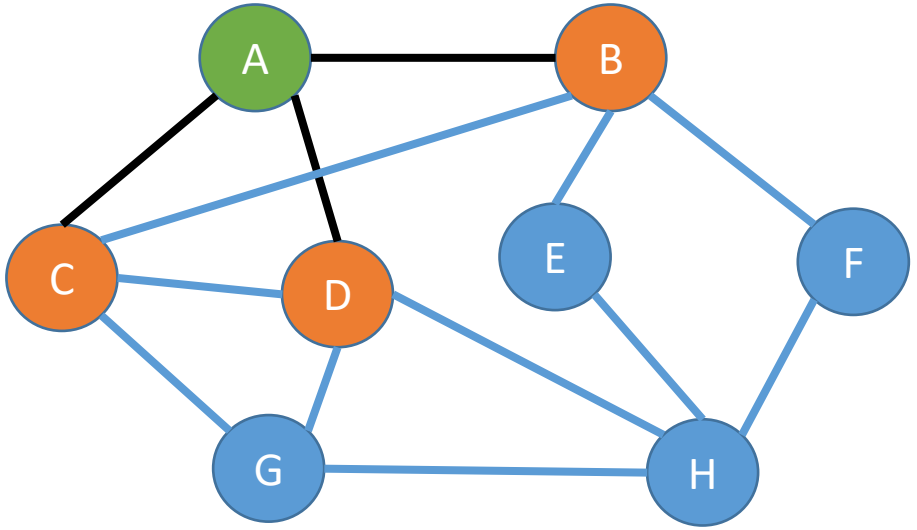
- A's unseen neighbors added to queue
  - A and unseen neighbors are added to nodes seen
- Order visited: A

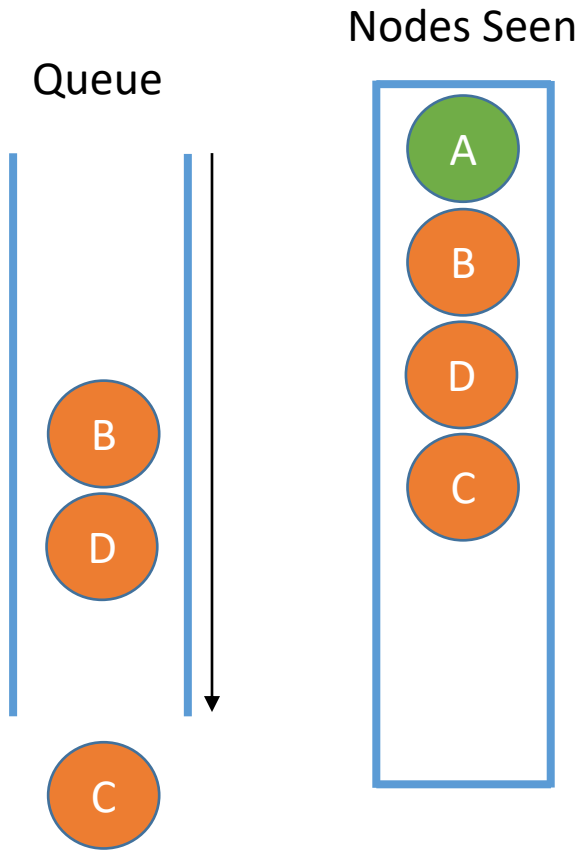




Breadth-First Traversal

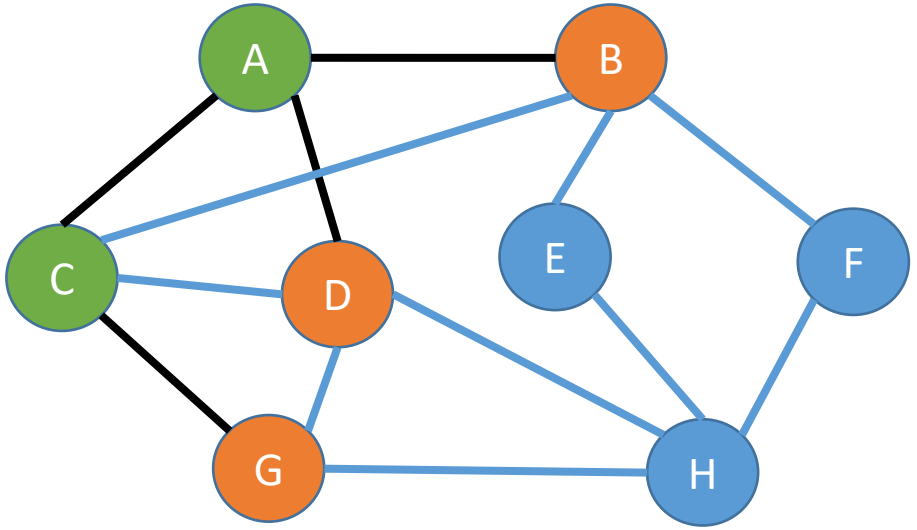
- C popped from the queue
- Order visited: A

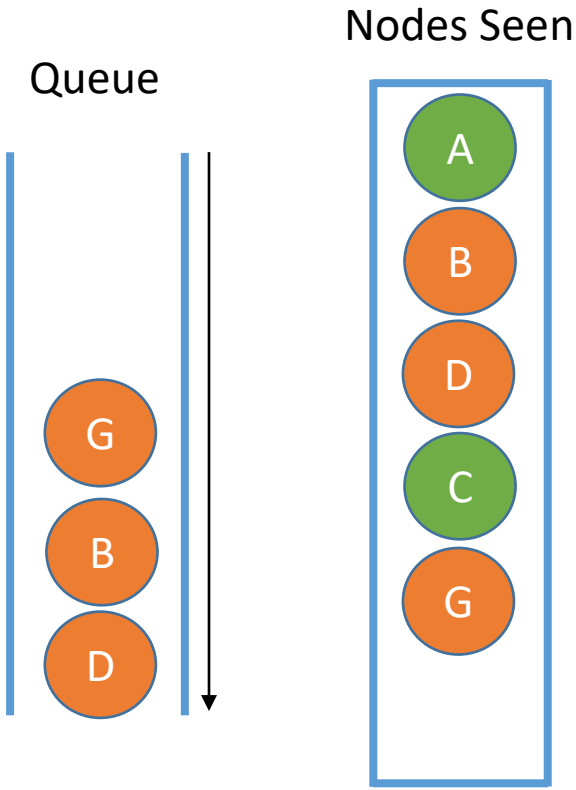




Breadth-First Traversal

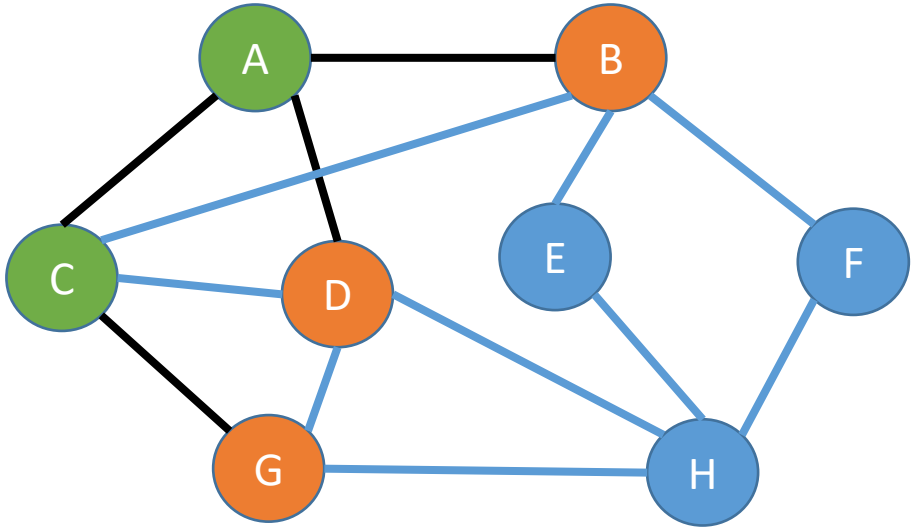
- C's unseen neighbors added to queue
  - Unseen neighbors are added to nodes seen
- Order visited: A, C

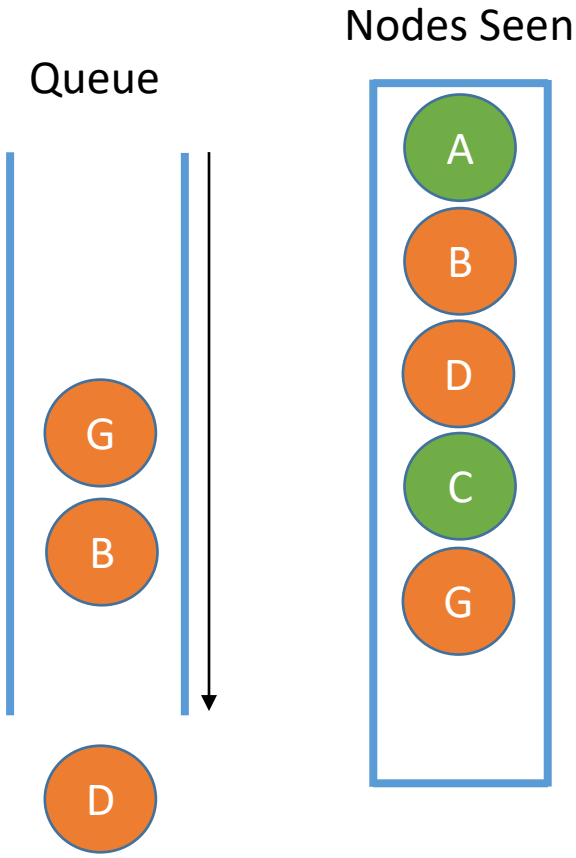




Breadth-First Traversal

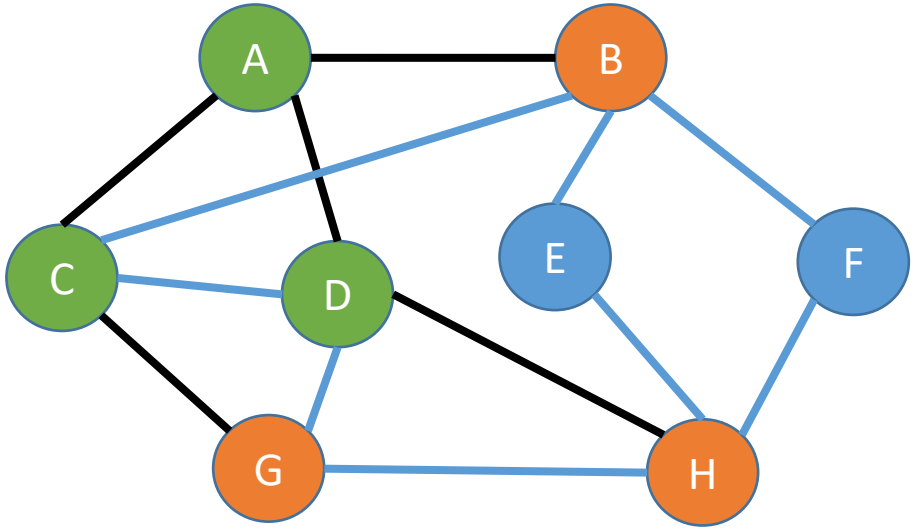
- D popped from the queue
- Order visited: A, C

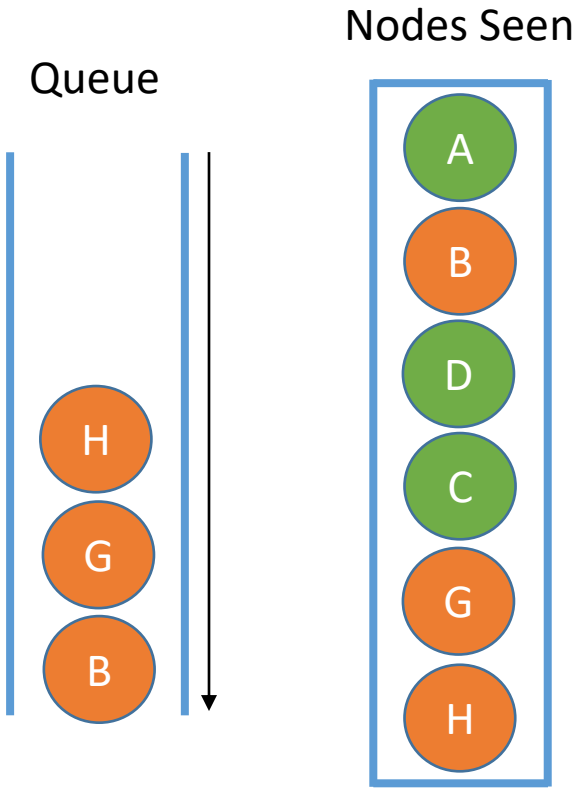




Breadth-First Traversal

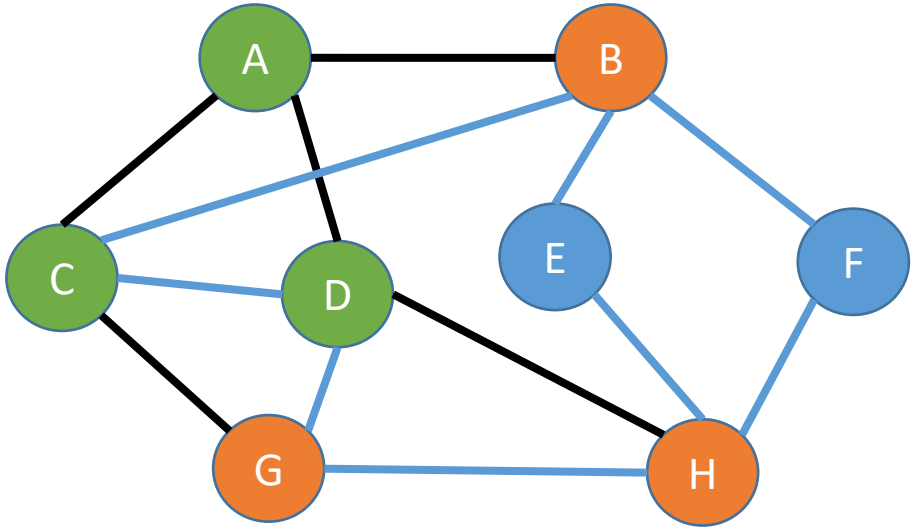
- D's unseen neighbors added to queue
  - Unseen neighbors are added to nodes seen
- Order visited: A, C, D

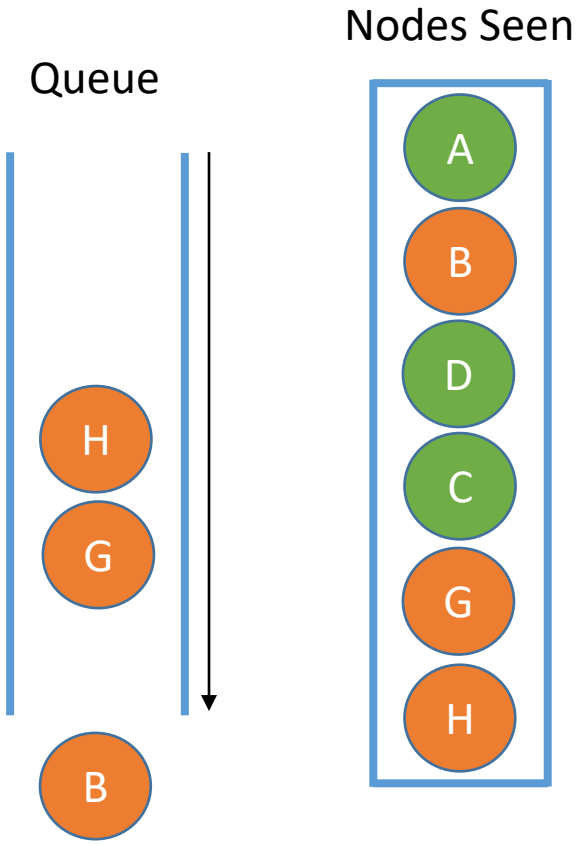




Breadth-First Traversal

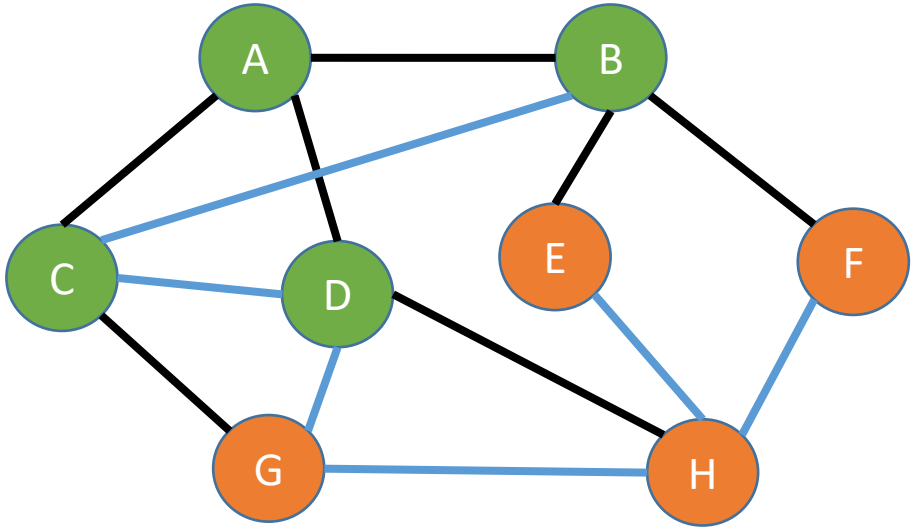
- B popped from the queue
- Order visited: A, C, D



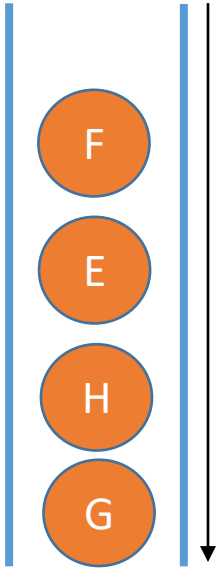


Breadth-First Traversal

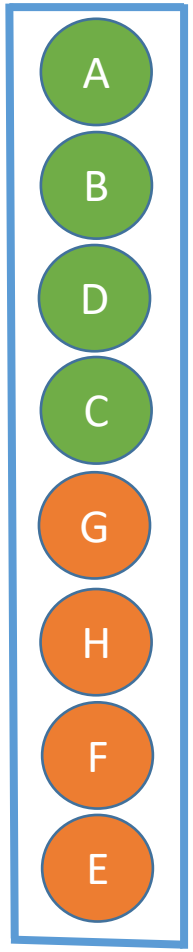
- B's unseen neighbors added to queue
  - Unseen neighbors are added to nodes seen
- Order visited: A, C, D, B



Queue

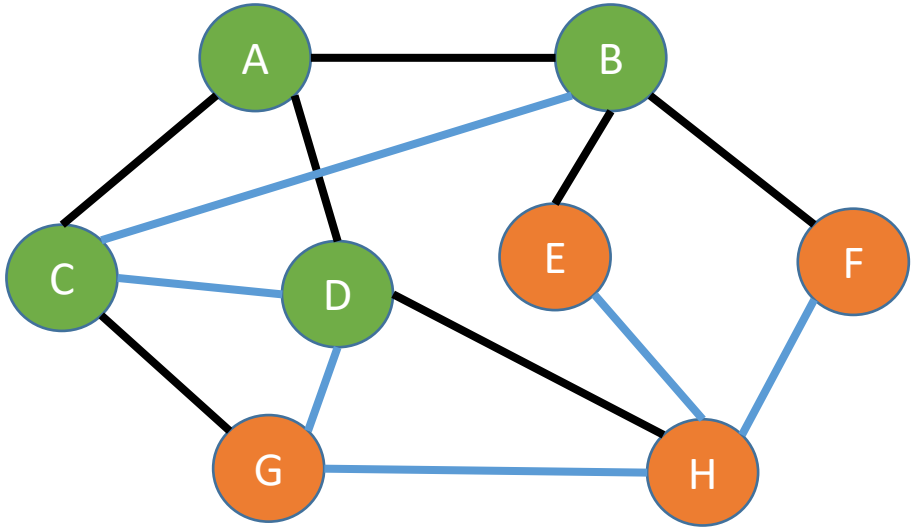


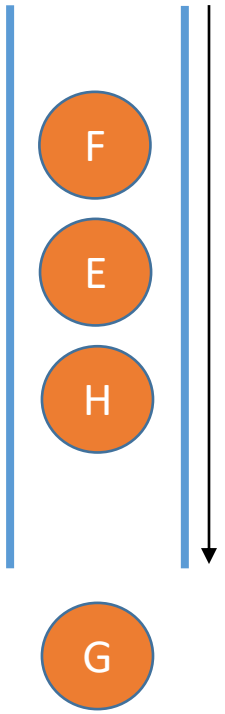
Nodes Seen

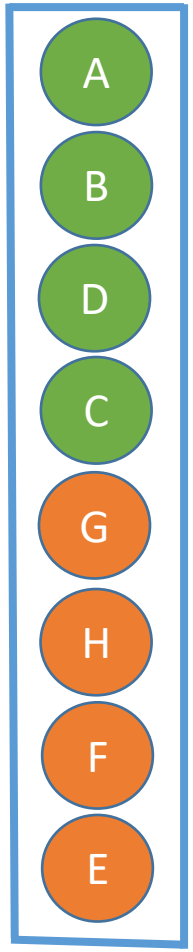


Breadth-First Traversal

- G popped from the queue
- Order visited: A, C, D, B

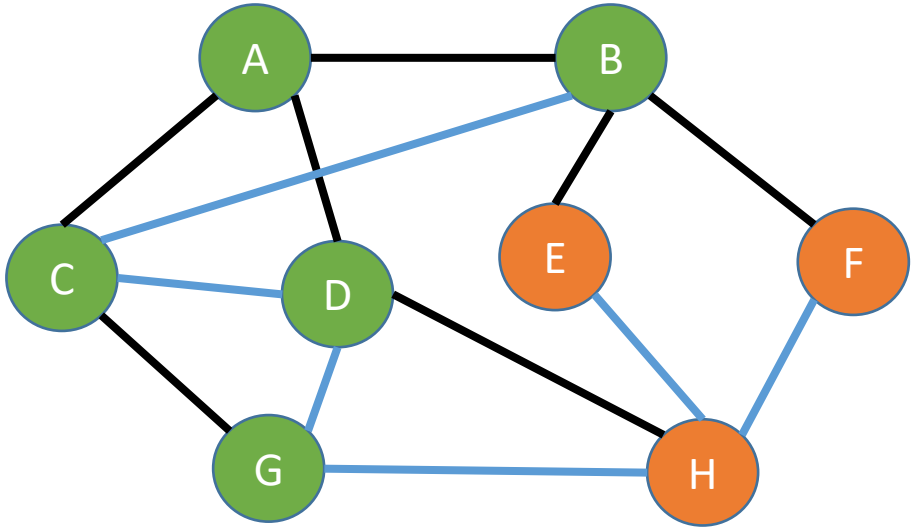




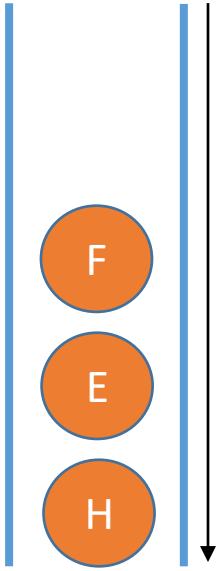


Breadth-First Traversal

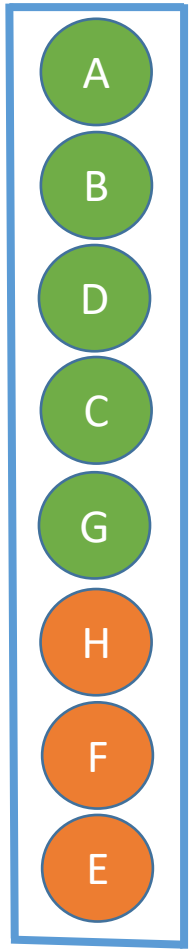
- G's unseen neighbors added to queue (none)
  - Unseen neighbors are added to nodes seen
- Order visited: A, C, D, B, G



Queue

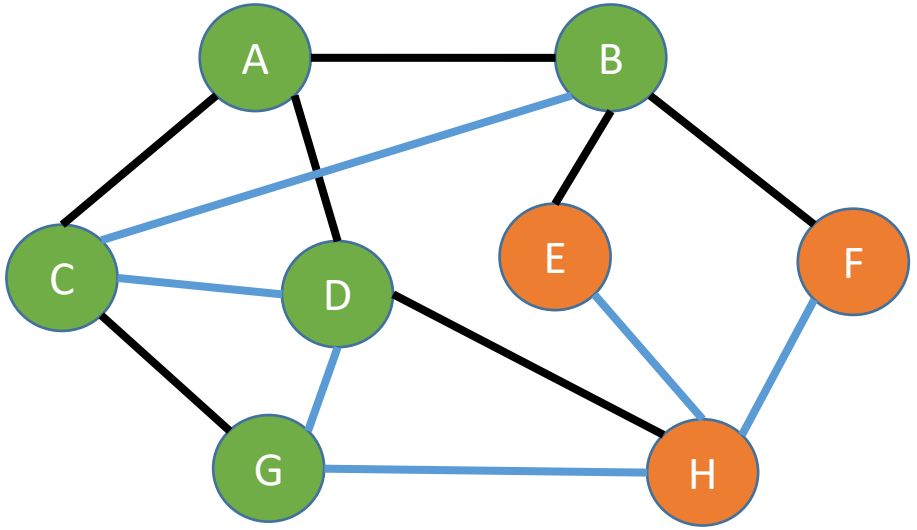


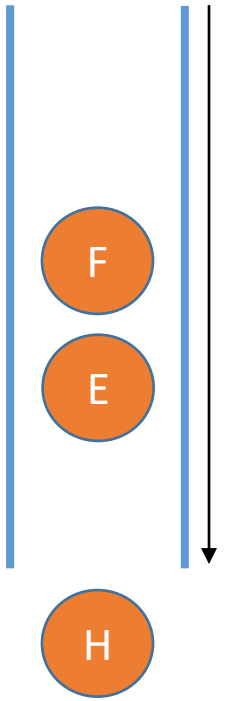
Nodes Seen

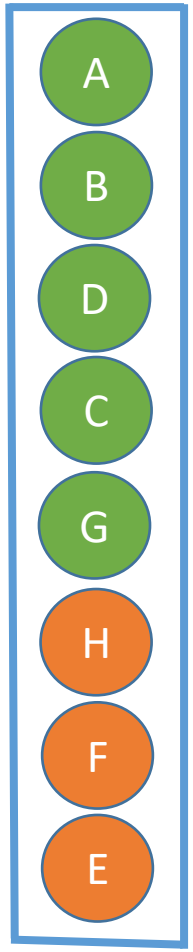


Breadth-First Traversal

- H popped from the queue
- Order visited: A, C, D, B, G

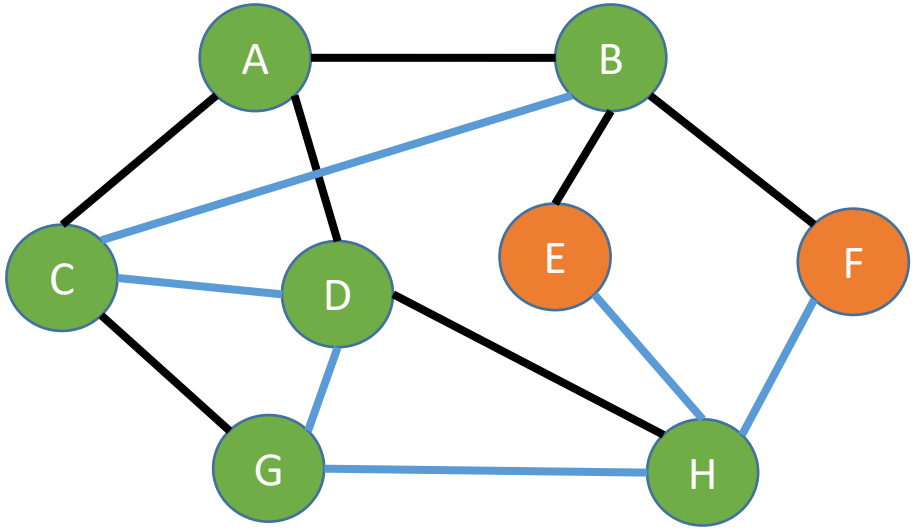




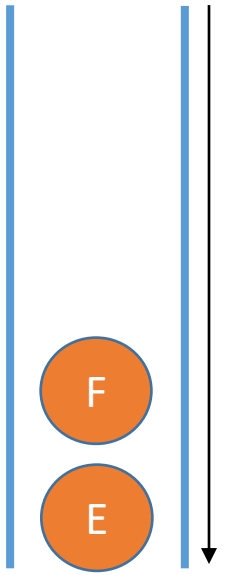


Breadth-First Traversal

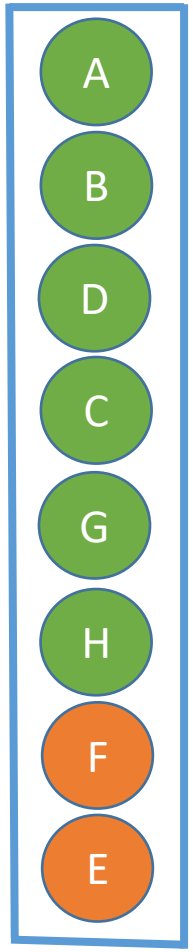
- H's unseen neighbors added to queue (none)
  - Unseen neighbors are added to nodes seen
- Order visited: A, C, D, B, G, H



Queue

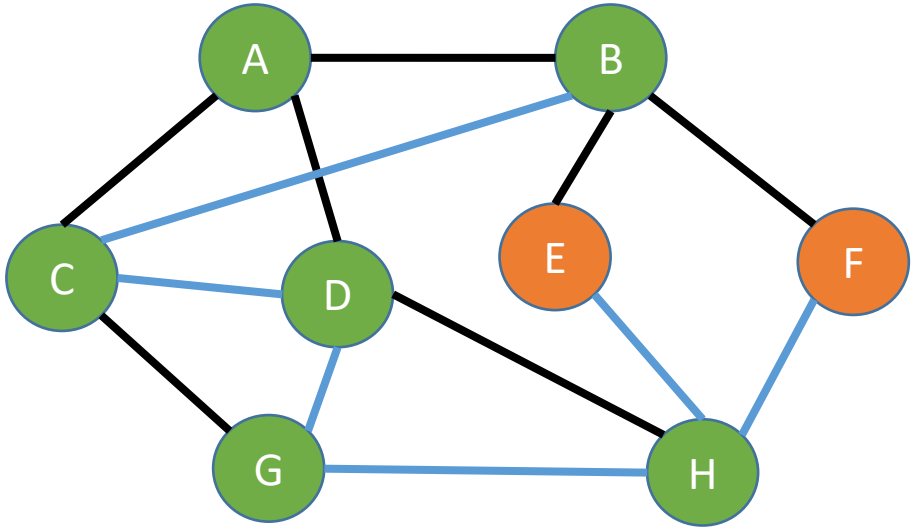


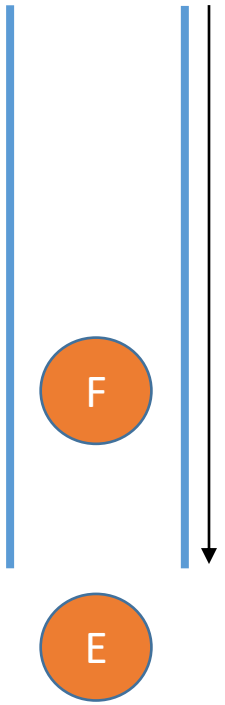
Nodes Seen

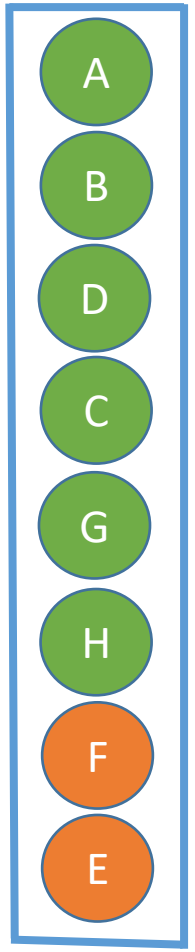


Breadth-First Traversal

- E popped from the queue
- Order visited: A, C, D, B, G, H

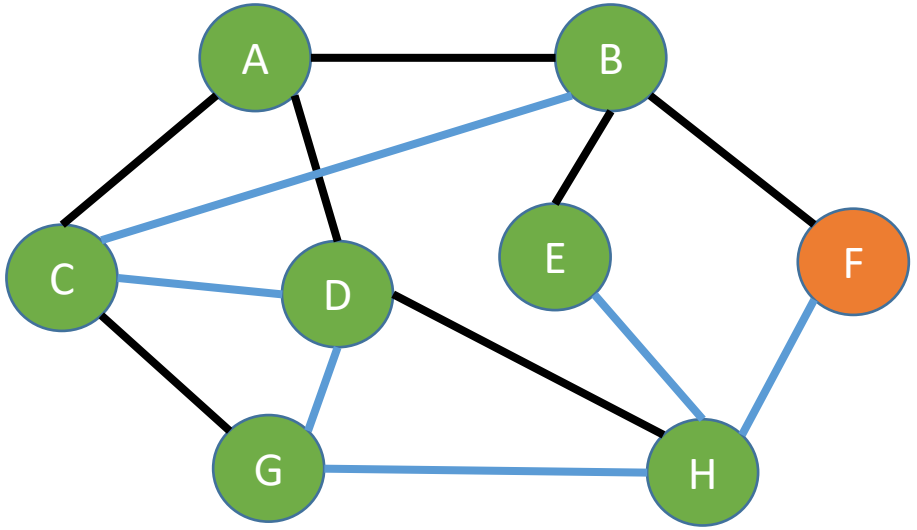




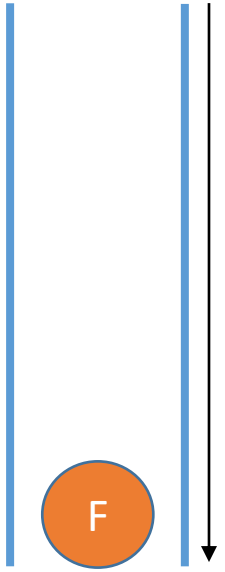


Breadth-First Traversal

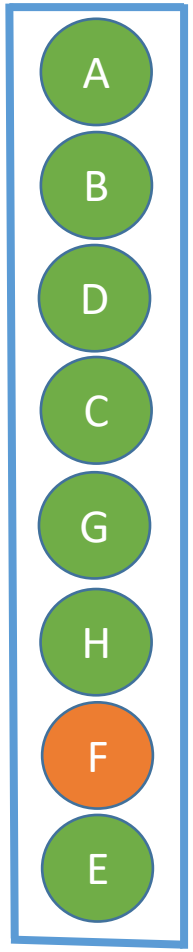
- E's unseen neighbors added to queue (none)
  - Unseen neighbors are added to nodes seen
- Order visited: A, C, D, B, G, H, E



Queue

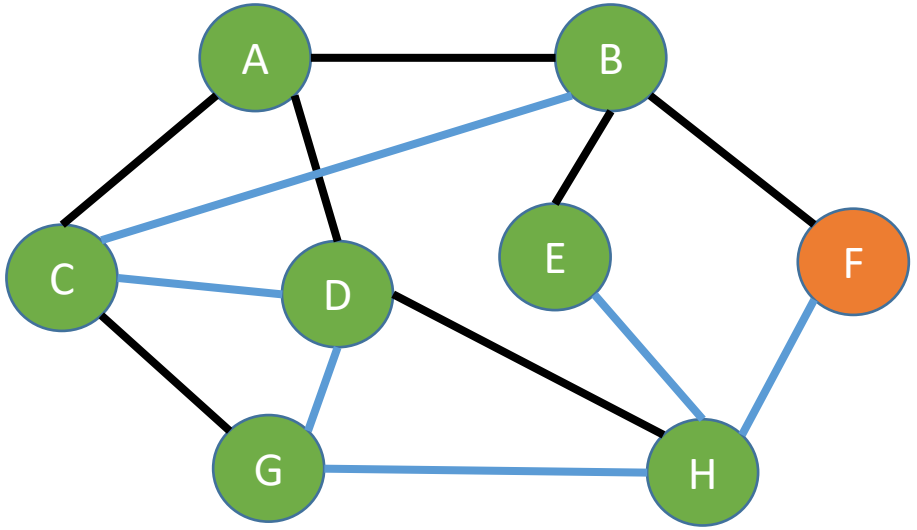


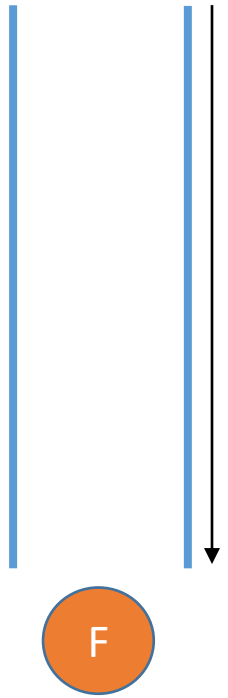
Nodes Seen

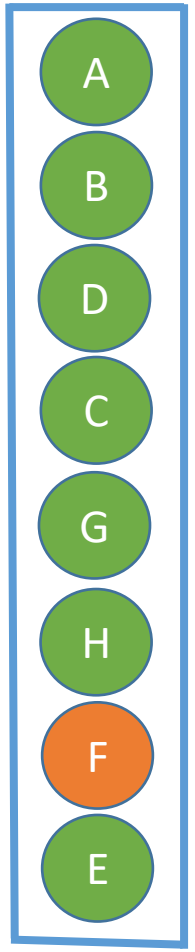


Breadth-First Traversal

- F popped from queue
- Order visited: A, C, D, B, G, H, E

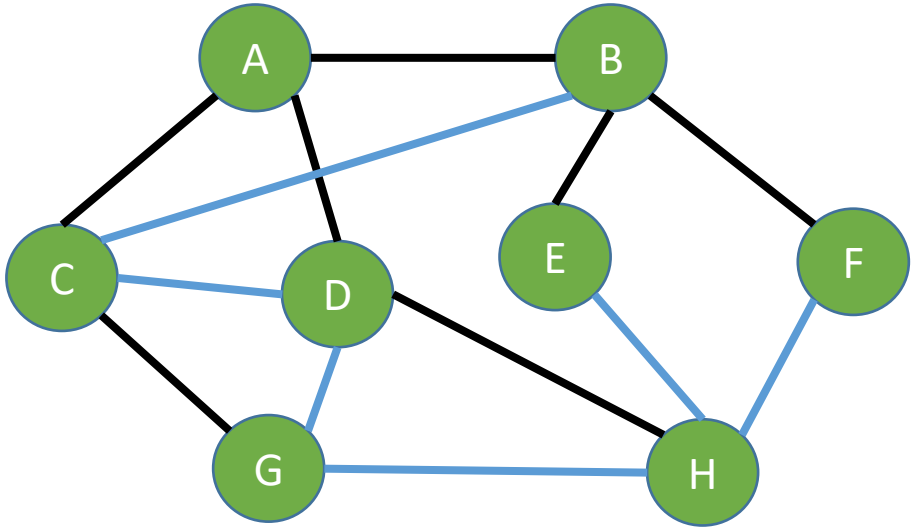




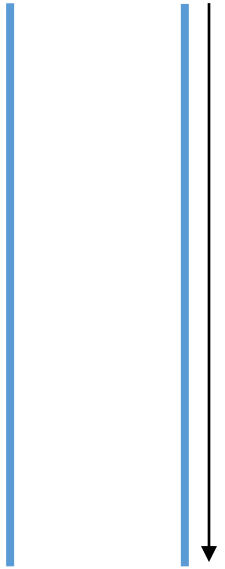


Breadth-First Traversal

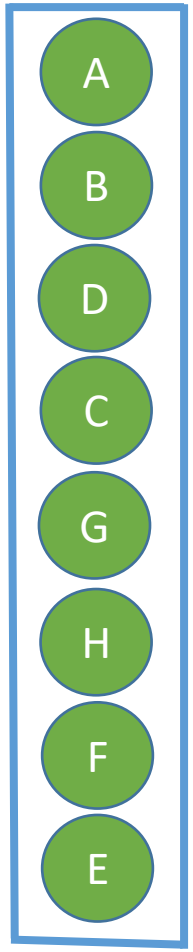
- F's unseen neighbors added to queue (none)
  - Unseen neighbors are added to nodes seen
- Order visited: A, C, D, B, G, H, E, F



Queue

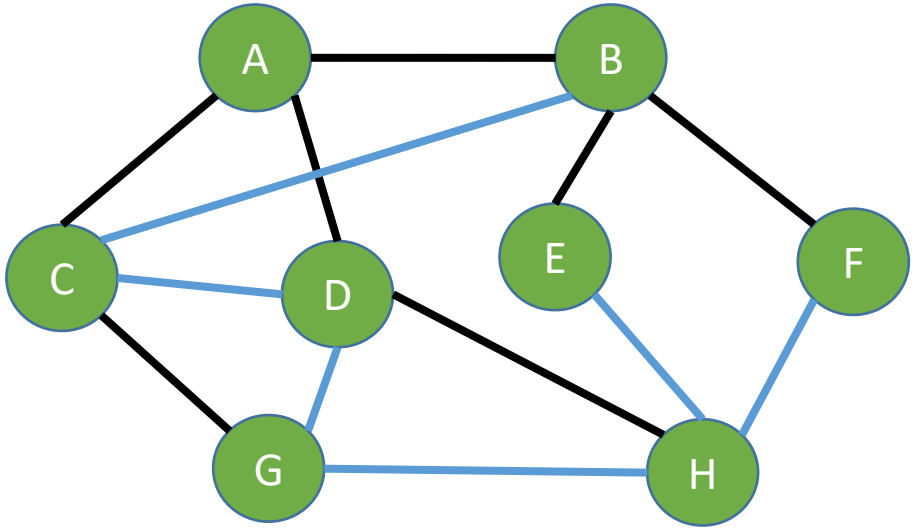


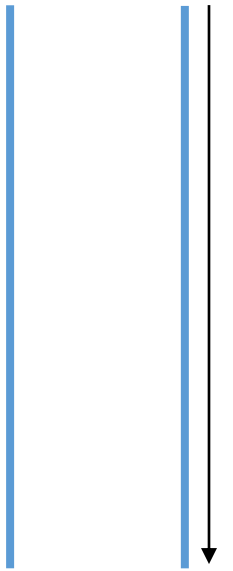
Nodes Seen



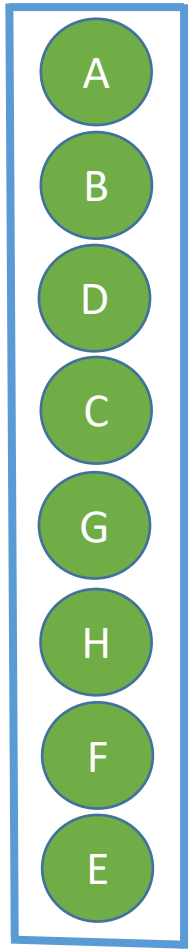
Breadth-First Traversal

- Queue is empty
  - Traversal is complete
- Order visited: **A, C, D, B, G, H, E, F**



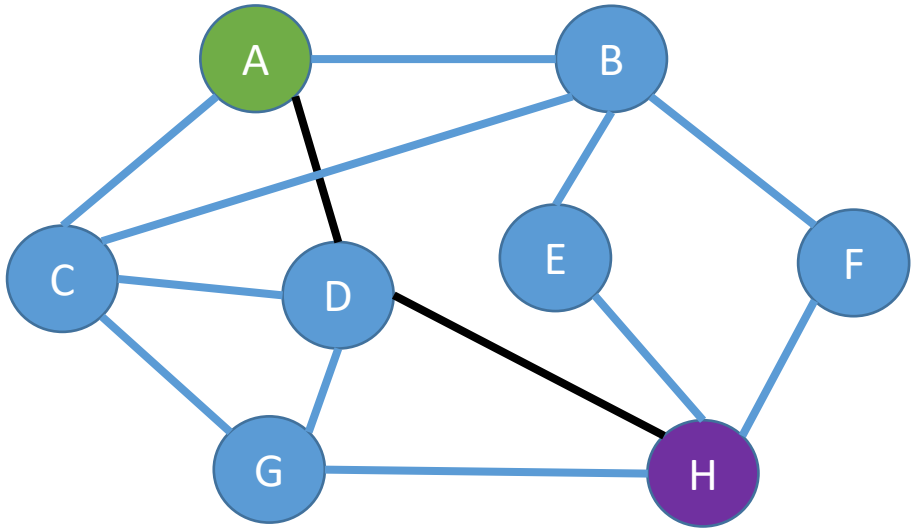


Nodes Seen



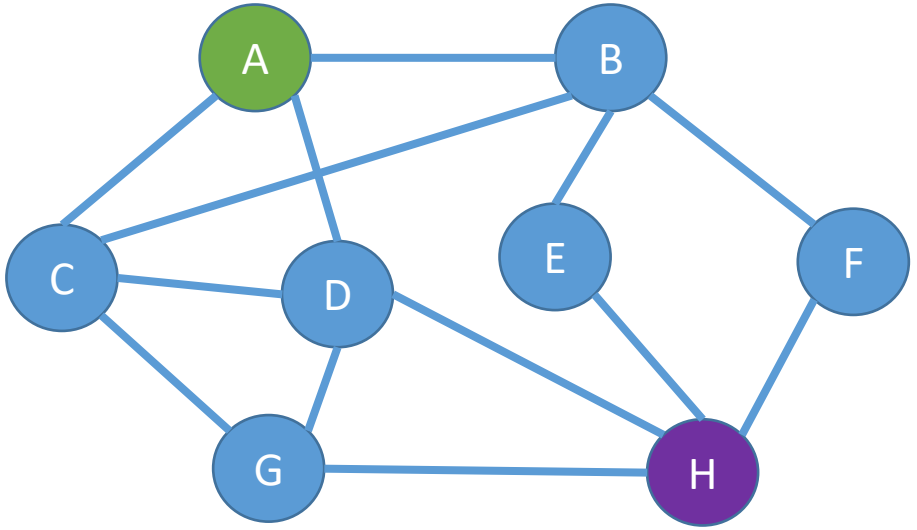
Finding Distance with Breadth-First

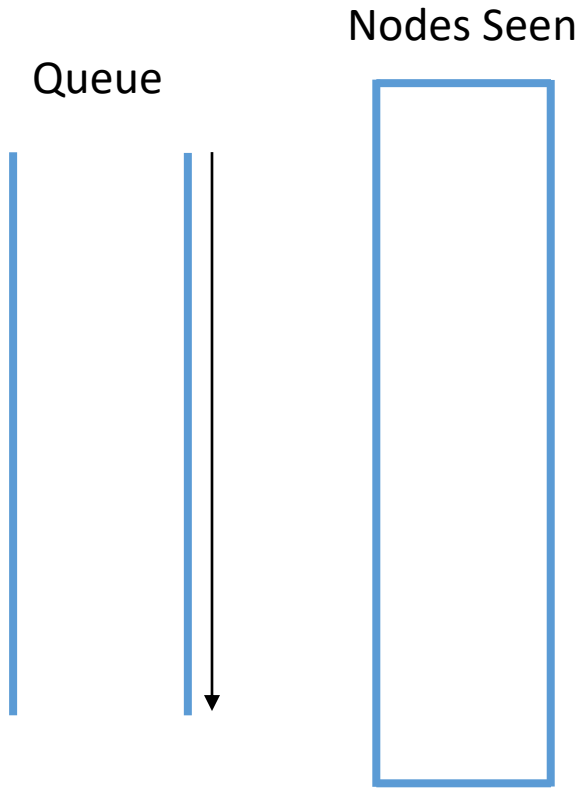
- Based on the last example, its easy to see a breadth-first traversal can be used to find the distance (shortest path) between two nodes
- As an example, we'll use a breadth-first traversal to find the distance between A and H



Finding Distance with Breadth-First

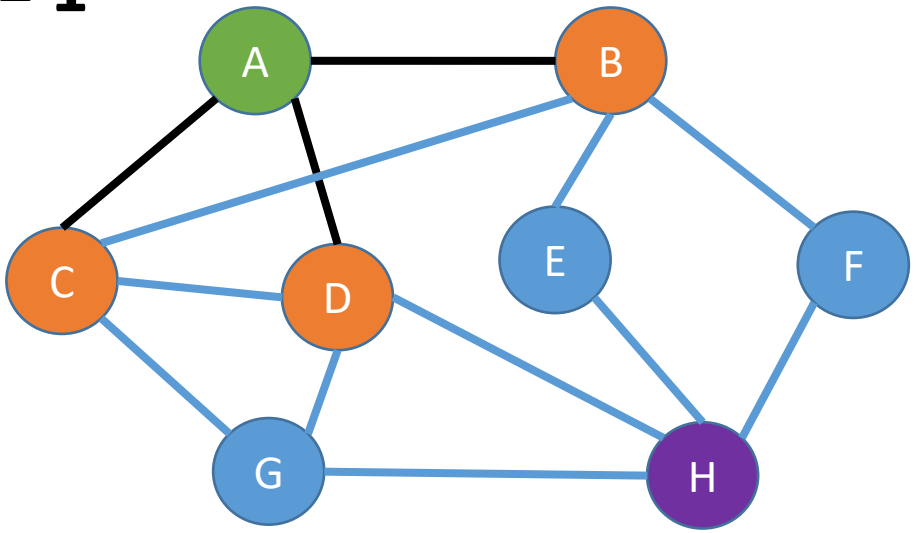
- Starting at A
- Order visited: N/A

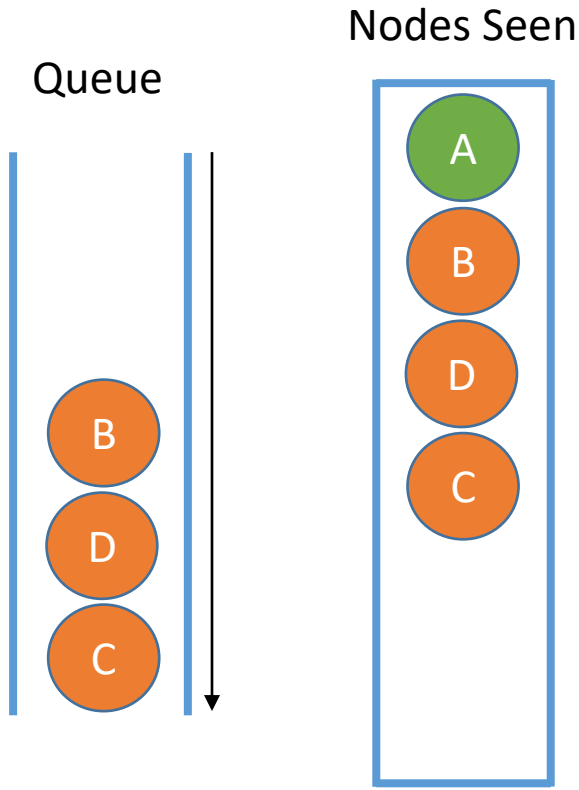




Finding Distance with Breadth-First

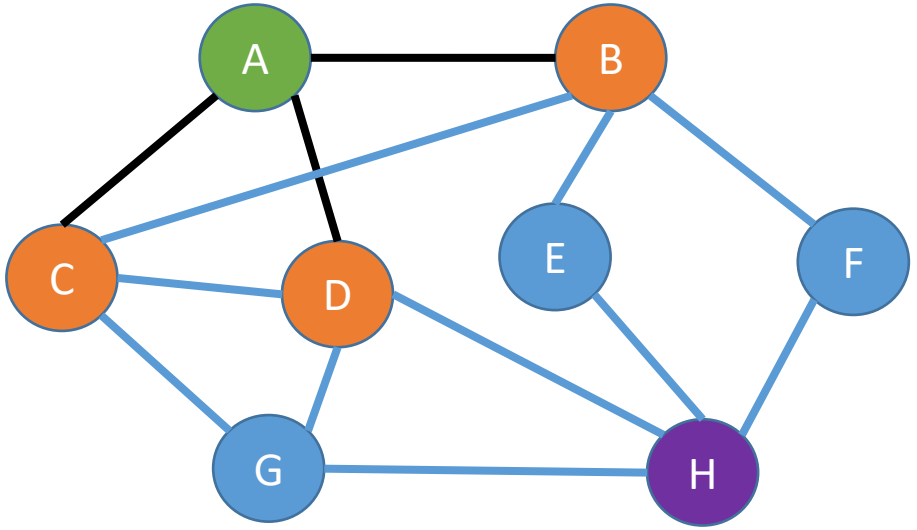
- A's unseen neighbors added to queue
  - A and unseen neighbors are added to nodes seen
- Order visited: A
- **DEEPEST LEVEL = 1**

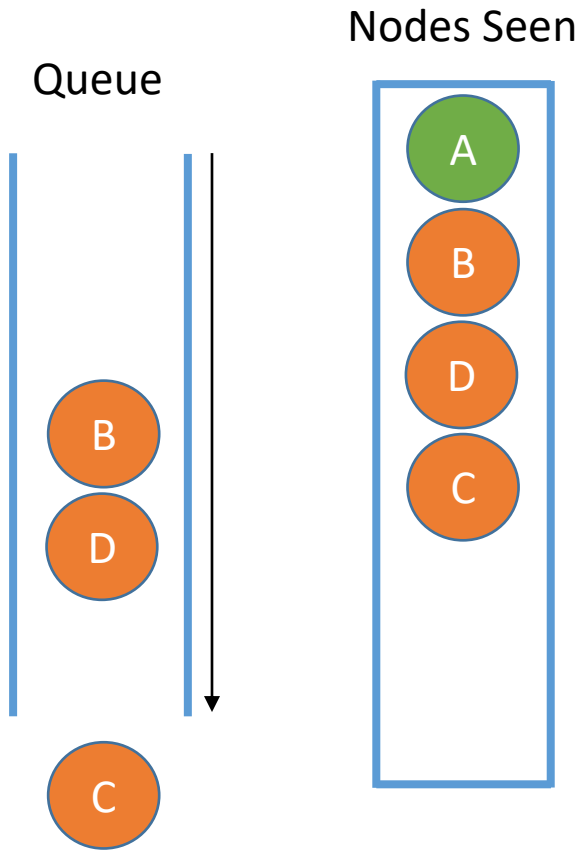




Finding Distance with Breadth-First

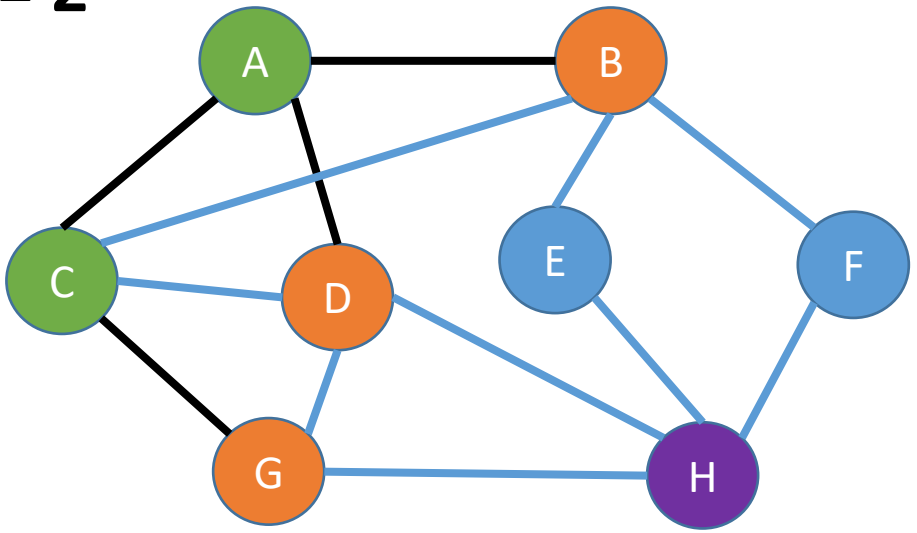
- C popped from the queue
- Order visited: A
- **DEEPEST LEVEL = 1**

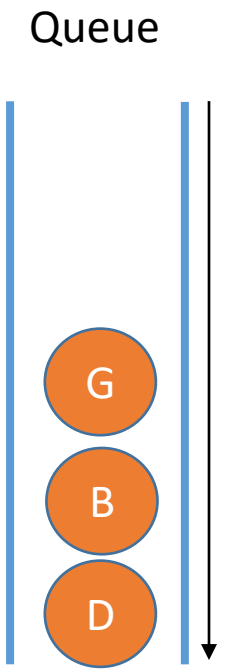


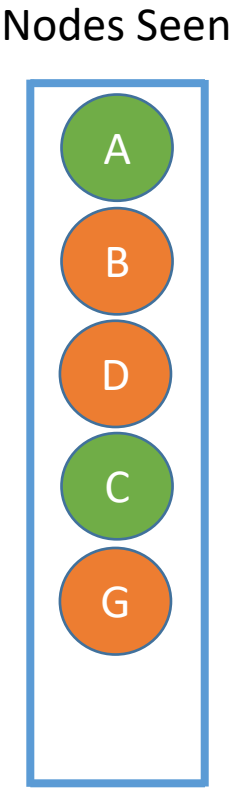


Finding Distance with Breadth-First

- C's unseen neighbors added to queue
  - Unseen neighbors are added to nodes seen
- Order visited: A, C
- **DEEPEST LEVEL = 2**

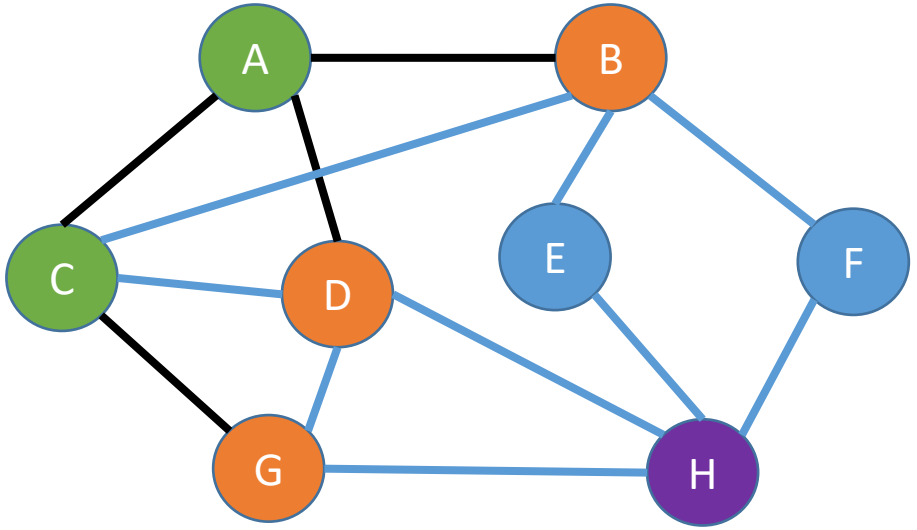


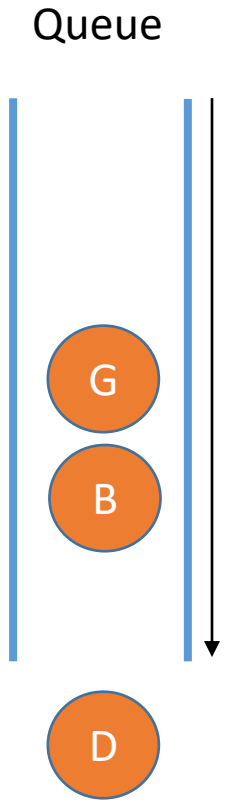


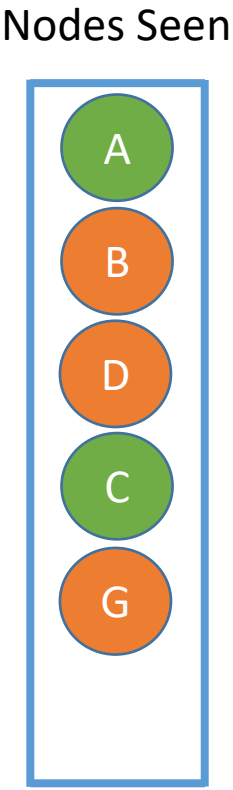


Finding Distance with Breadth-First

- D popped from the queue
- Order visited: A, C
- **DEEPEST LEVEL = 2**







Finding Distance with Breadth-First

- D's unseen neighbors added to queue
  - H was reached
  - Traversal stops
- **DEEPEST LEVEL = 2**
  - **DISTANCE = 2**