JamCoders: Week 4 Day 1B

Graph Traversals: BFS Review

Welcome! We'll start at 3:13PM!

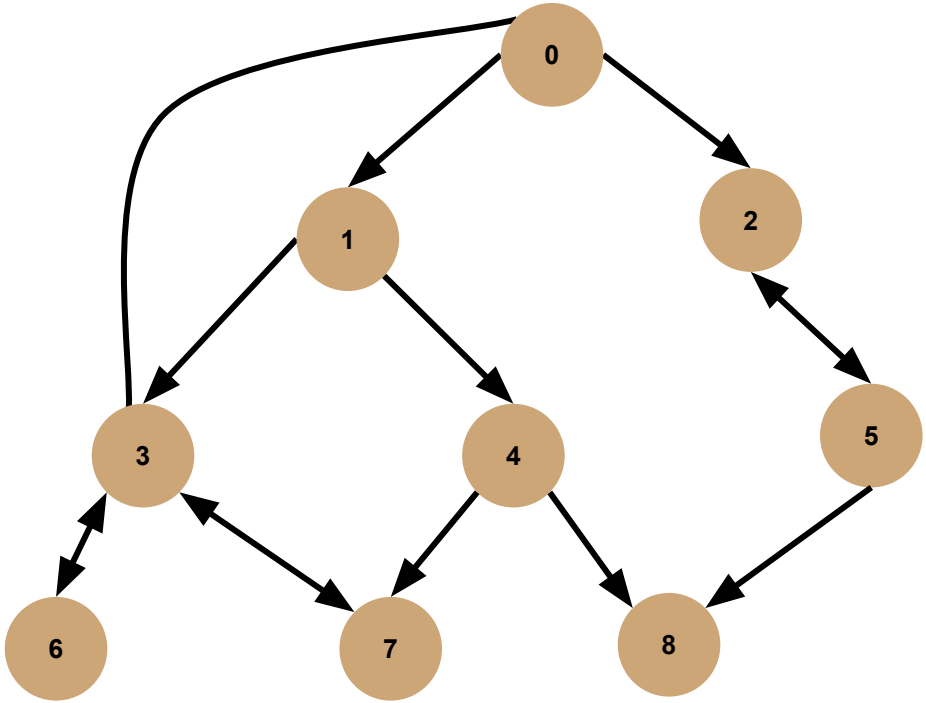
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: - - - - - - - - -

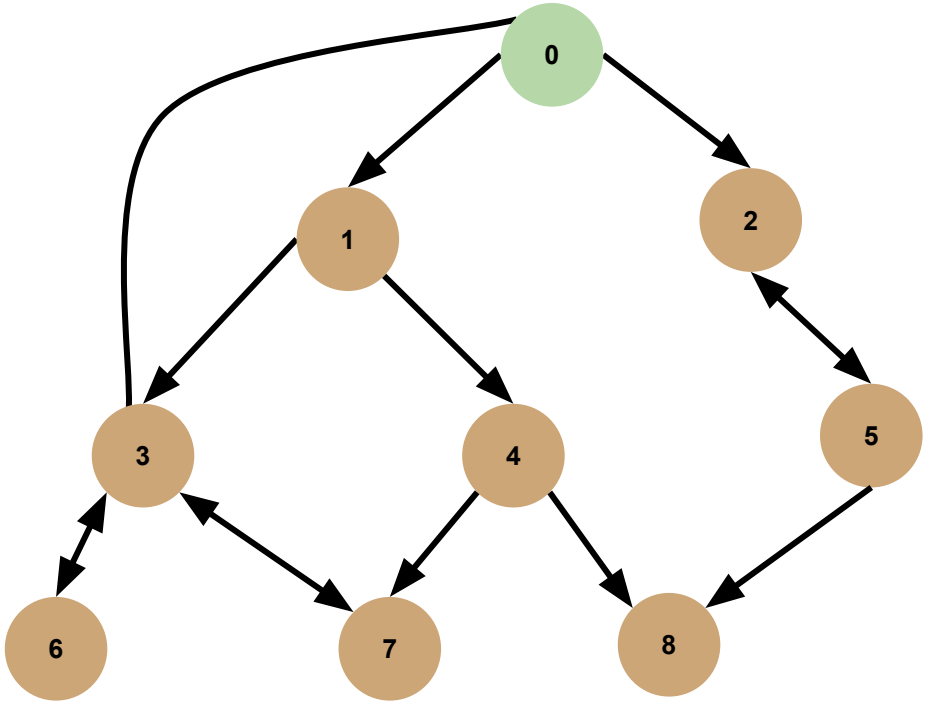
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 - - - - - - - -

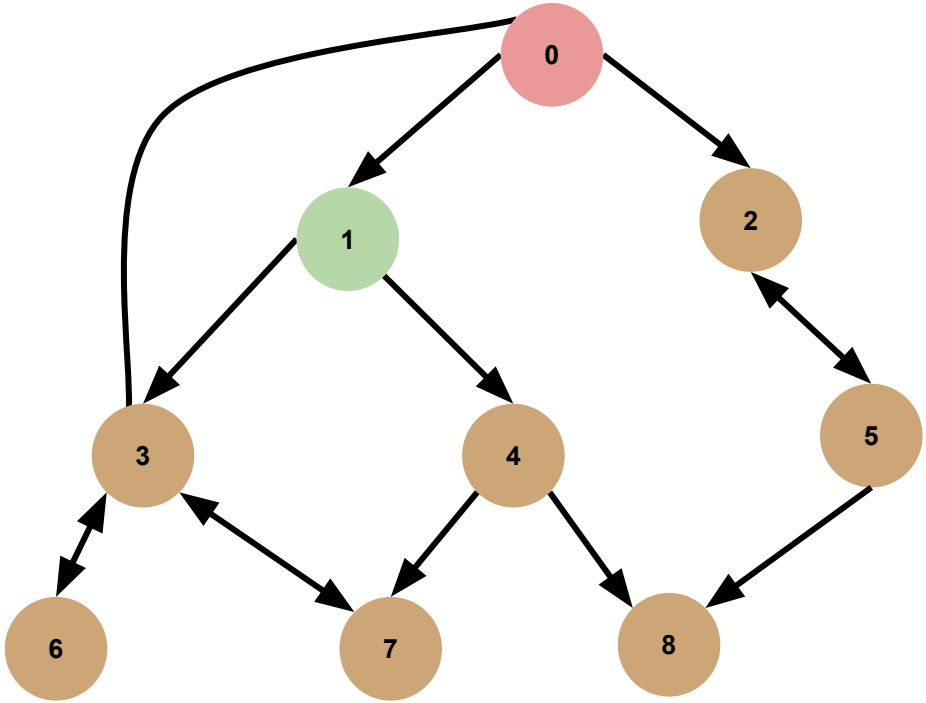
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 - - - - - - -

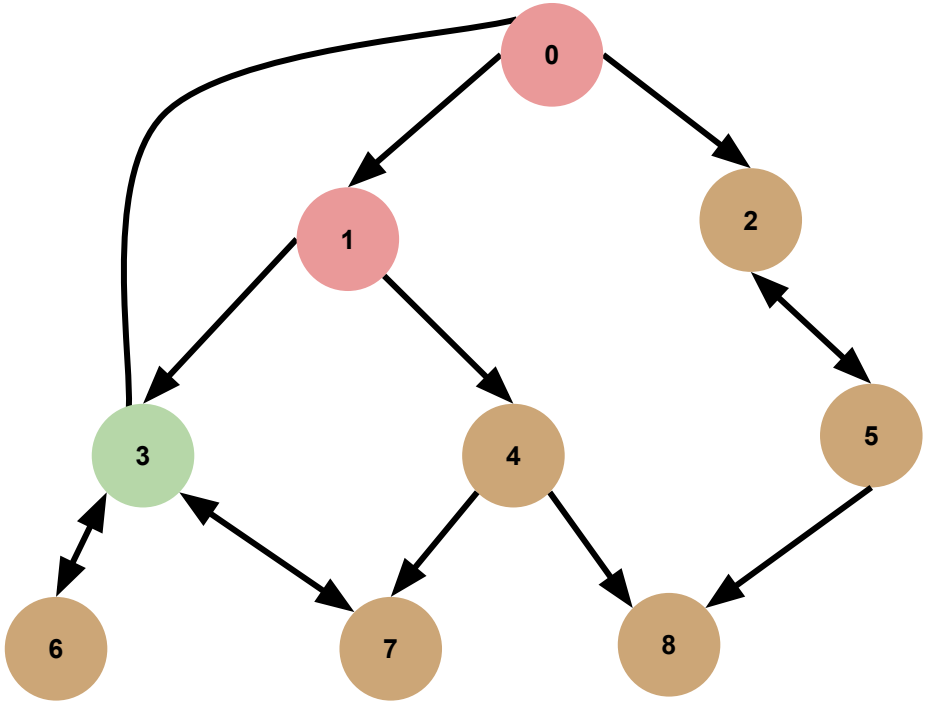
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 - - - - - -

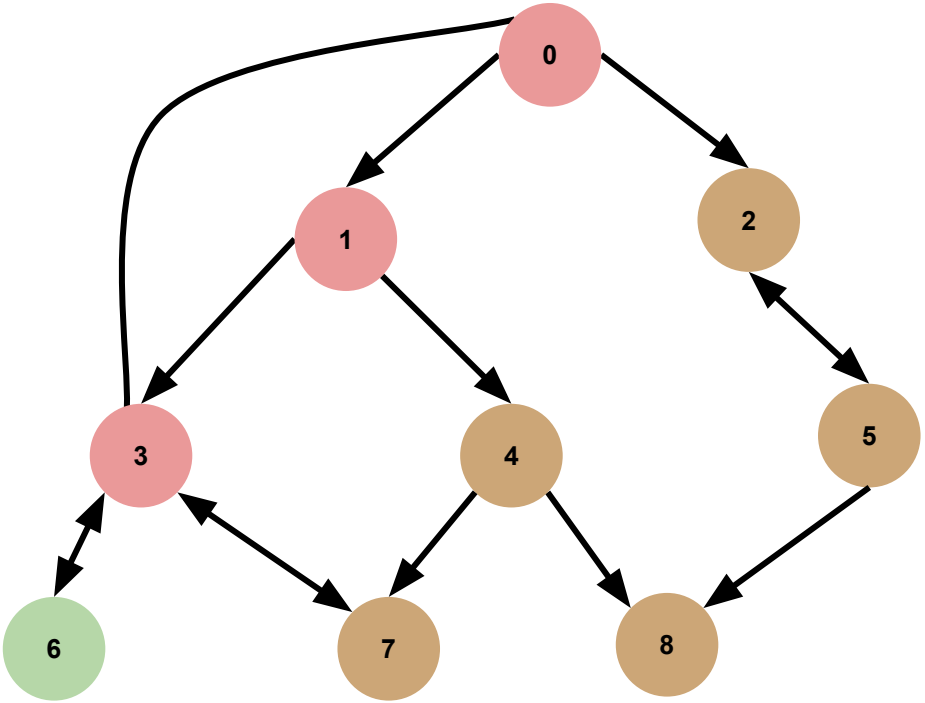
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 - - - - -

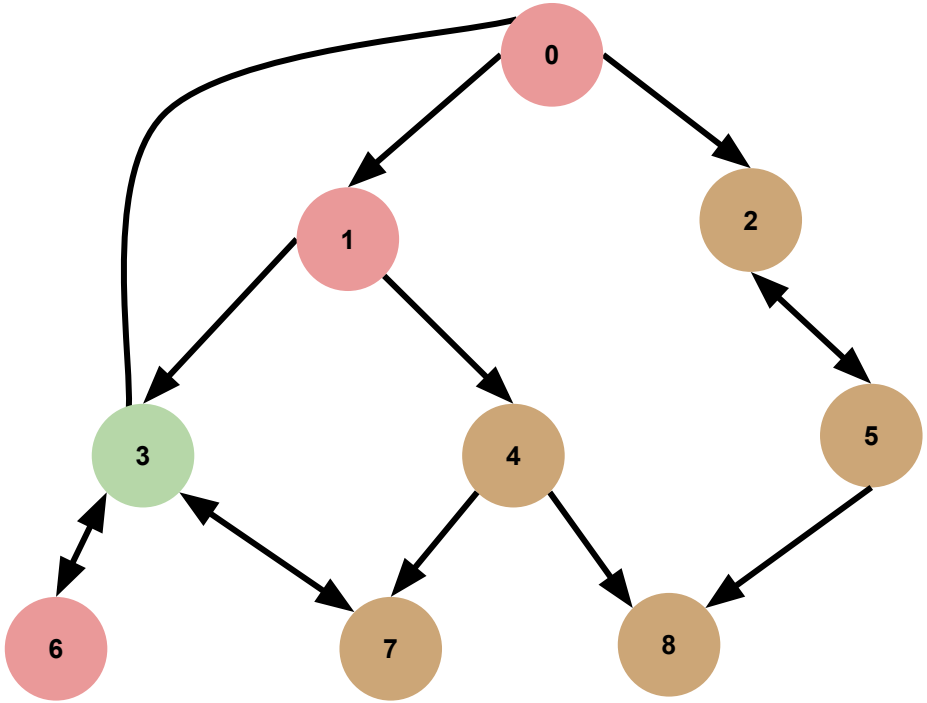
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 - - - - -

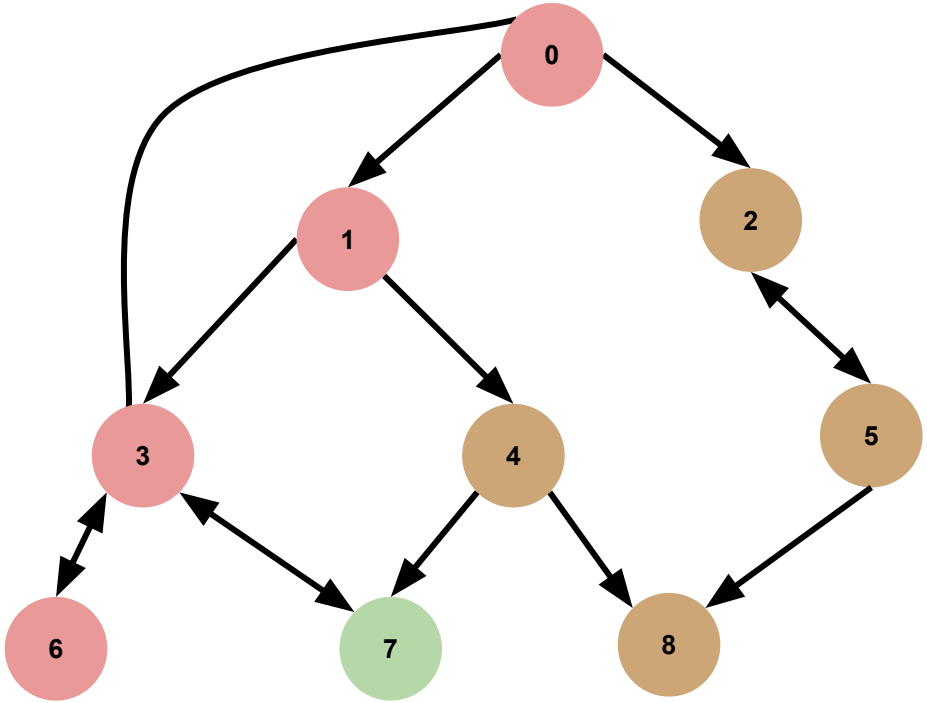
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 - - - -

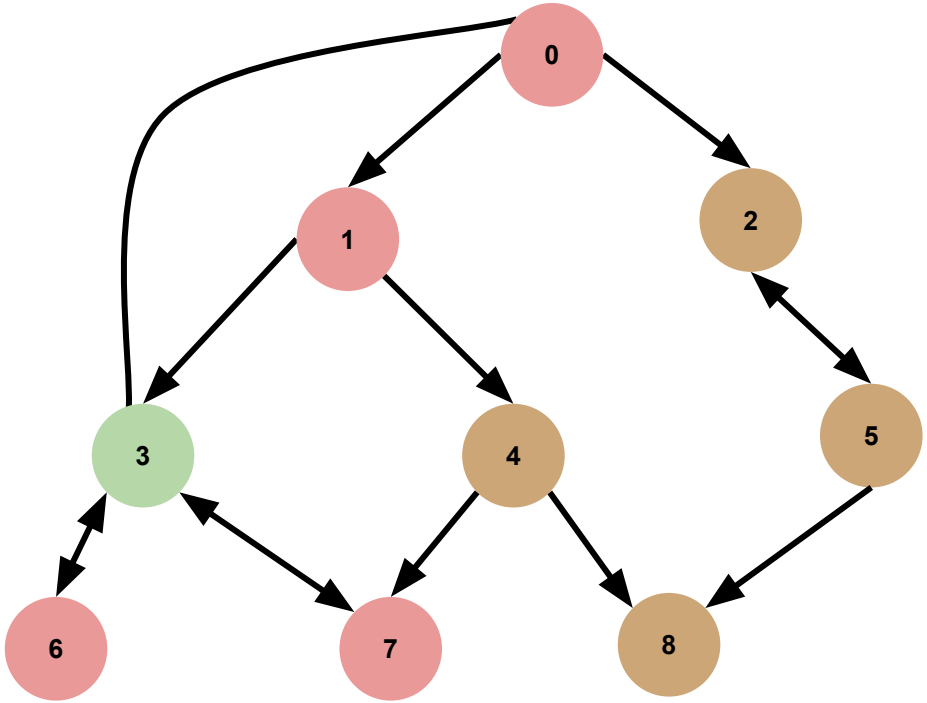
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 - - - -

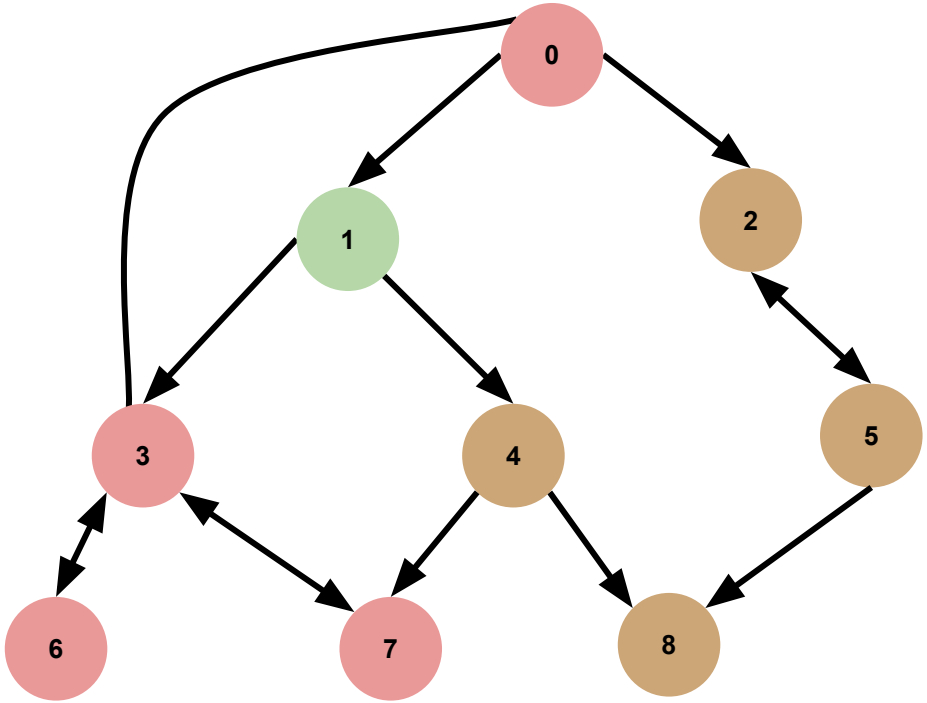
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 - - - -

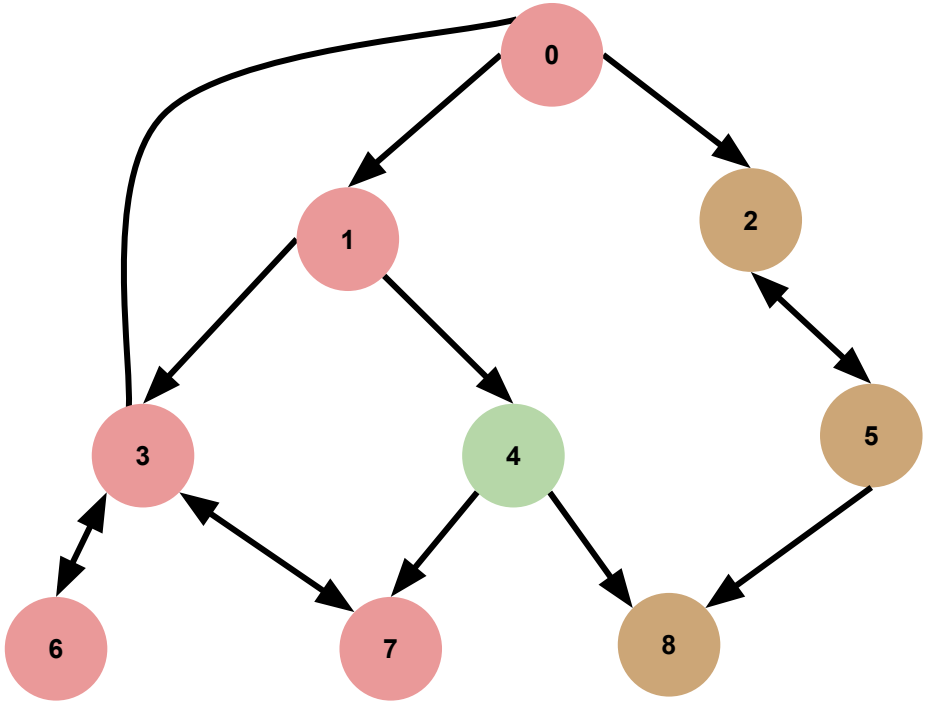
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 - - -

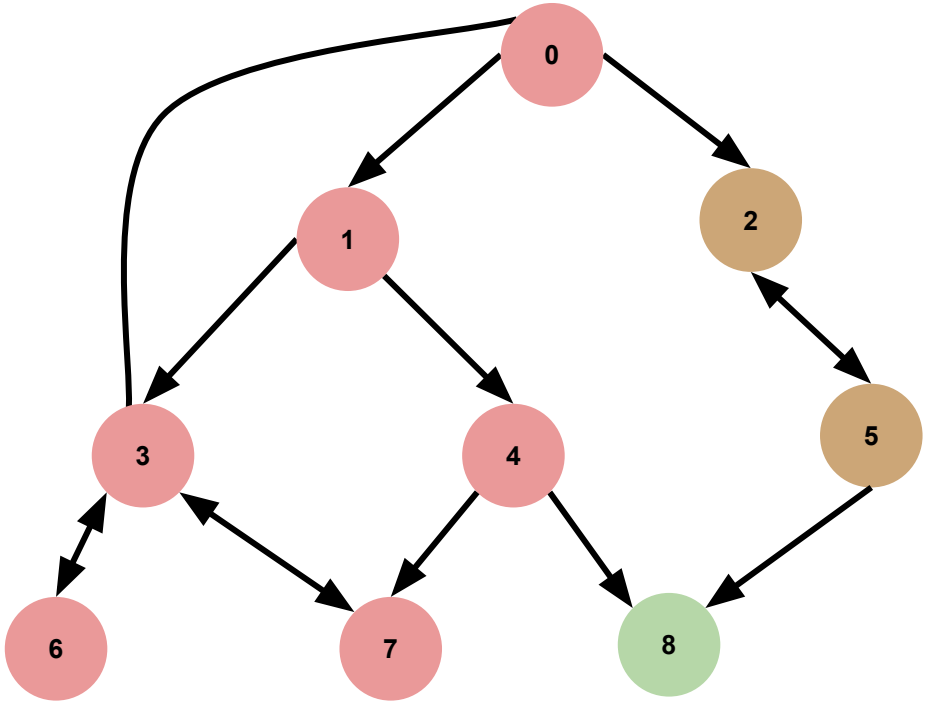
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 - -

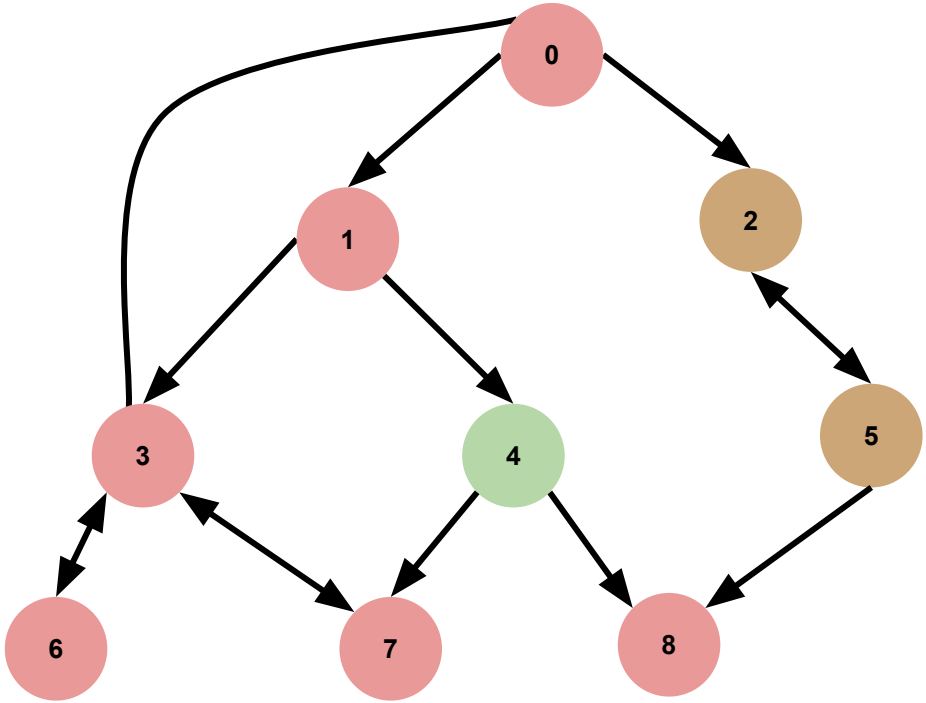
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 - -

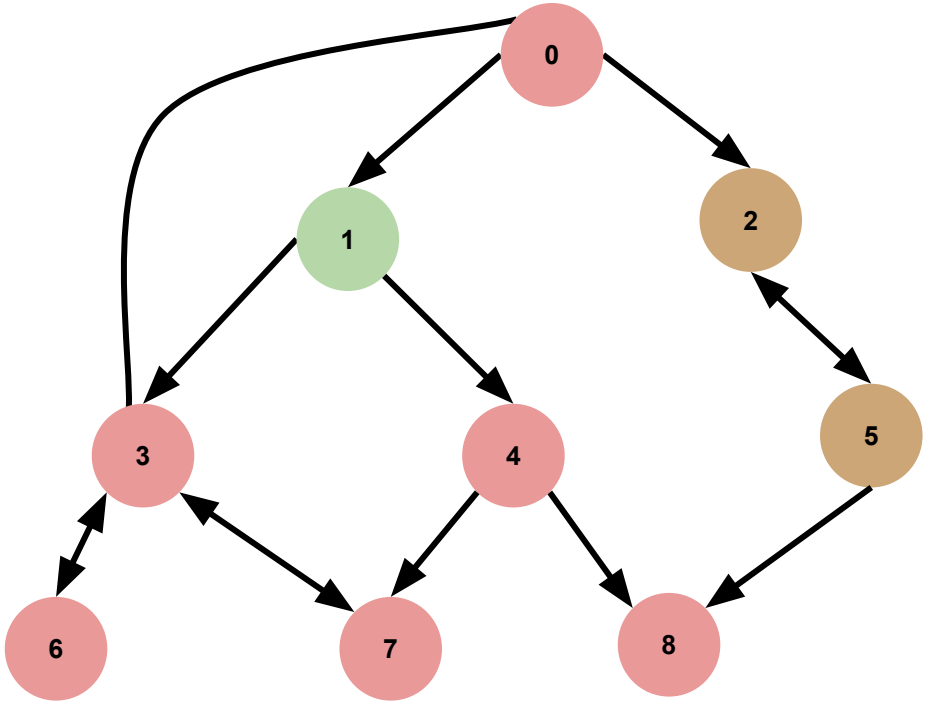
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 - -

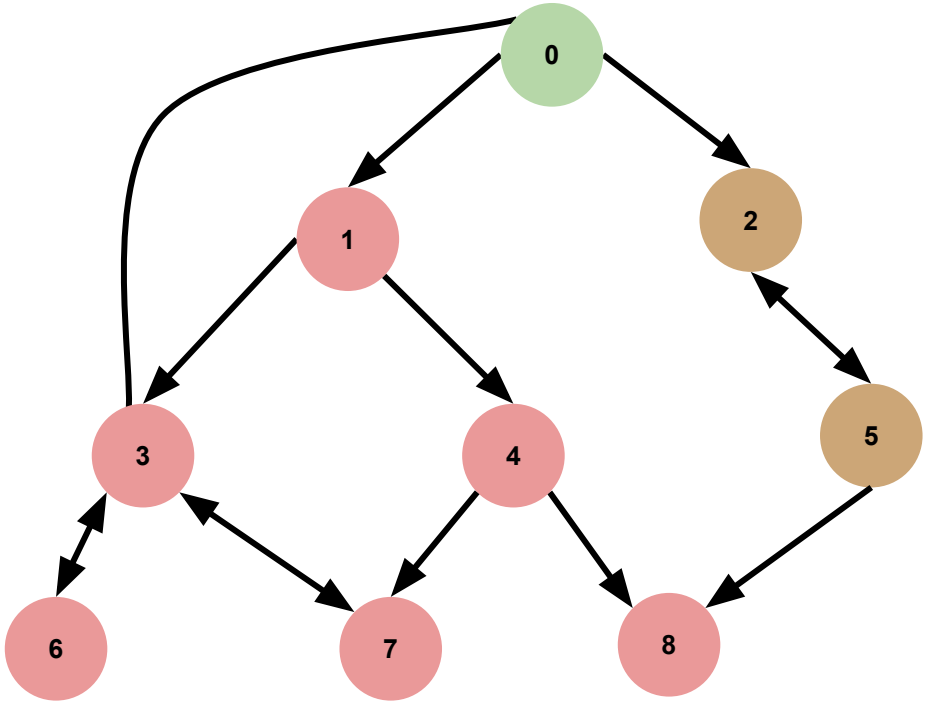
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 - -

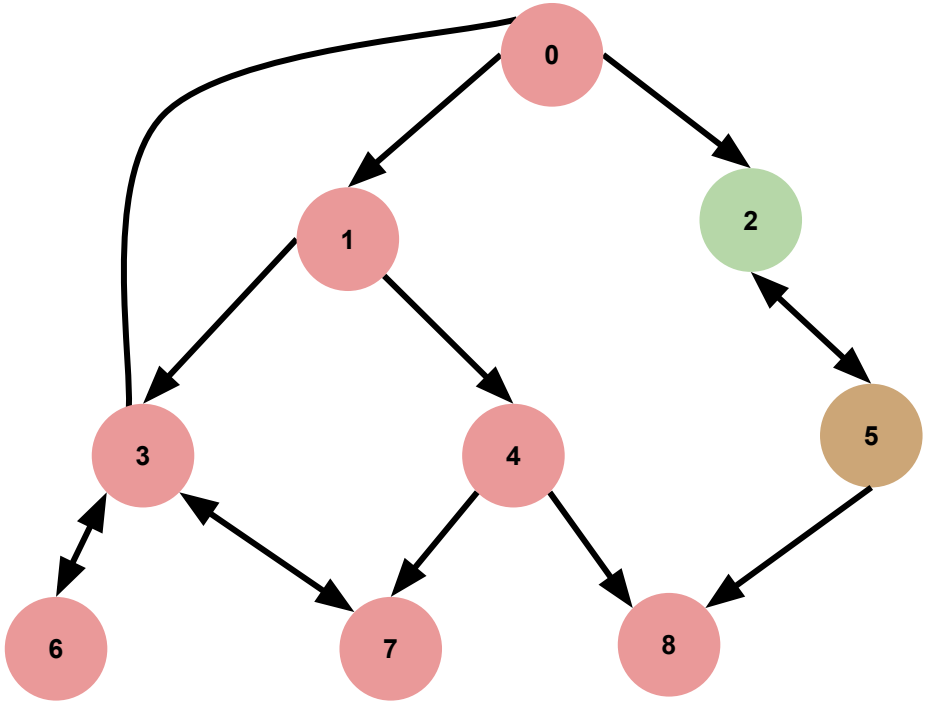
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 2 -

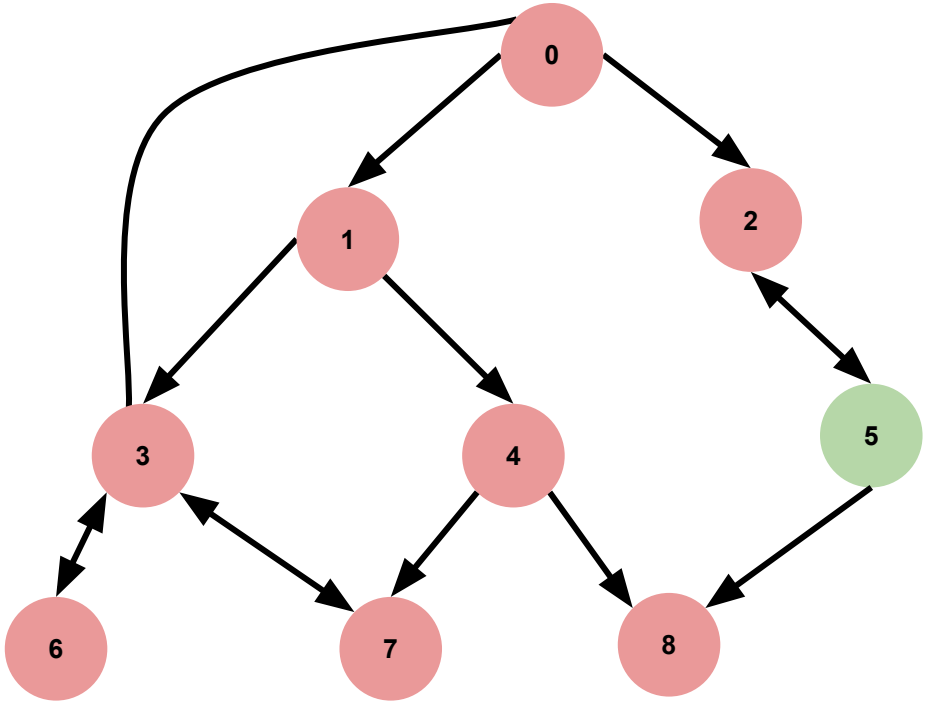
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 2 5

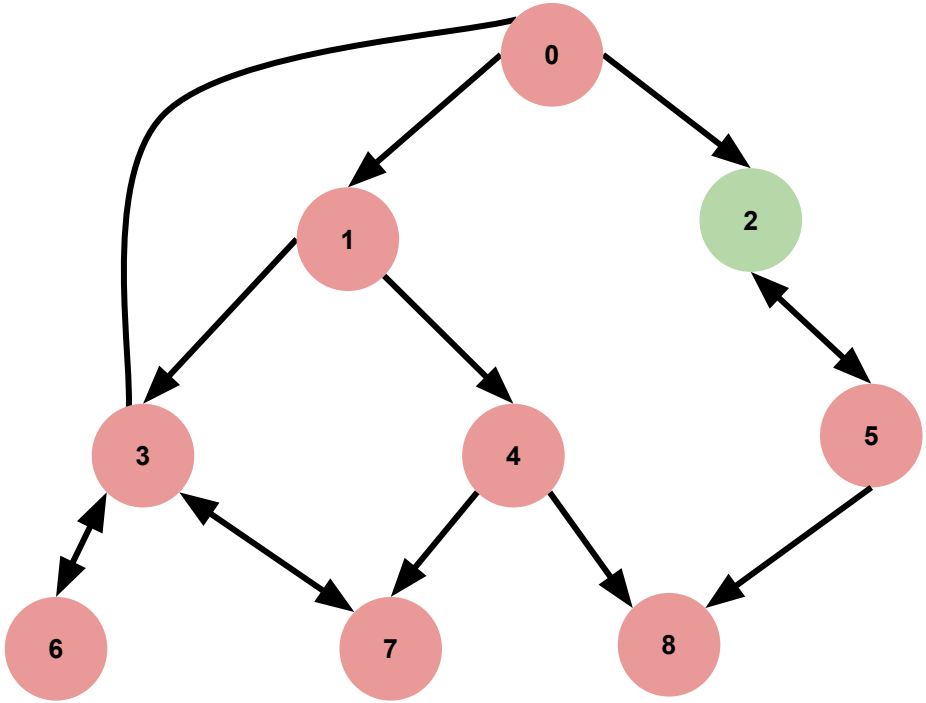
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 2 5

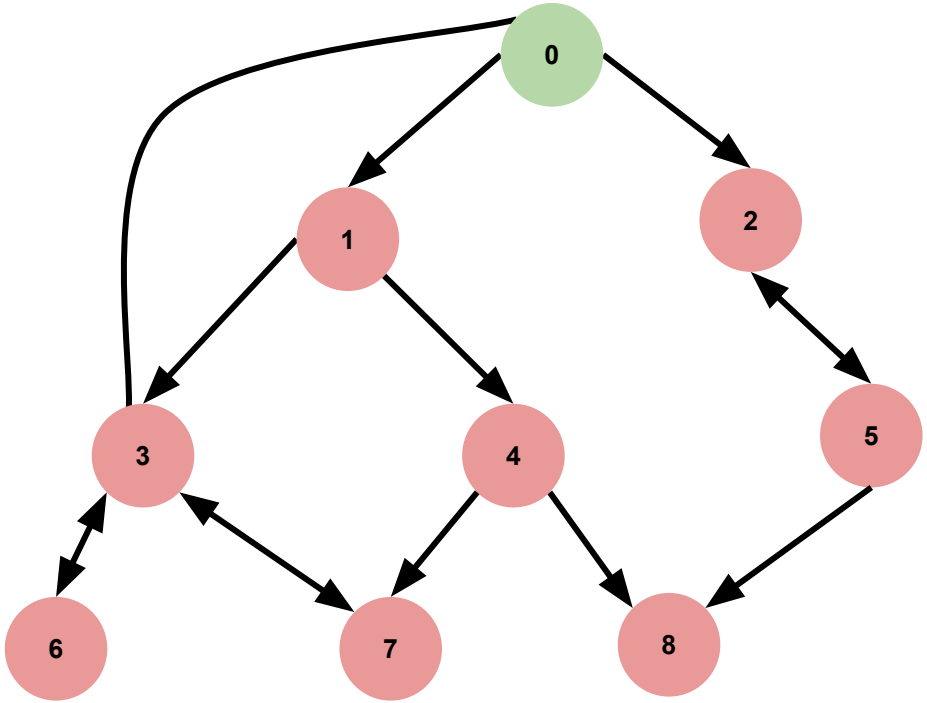
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 2 5

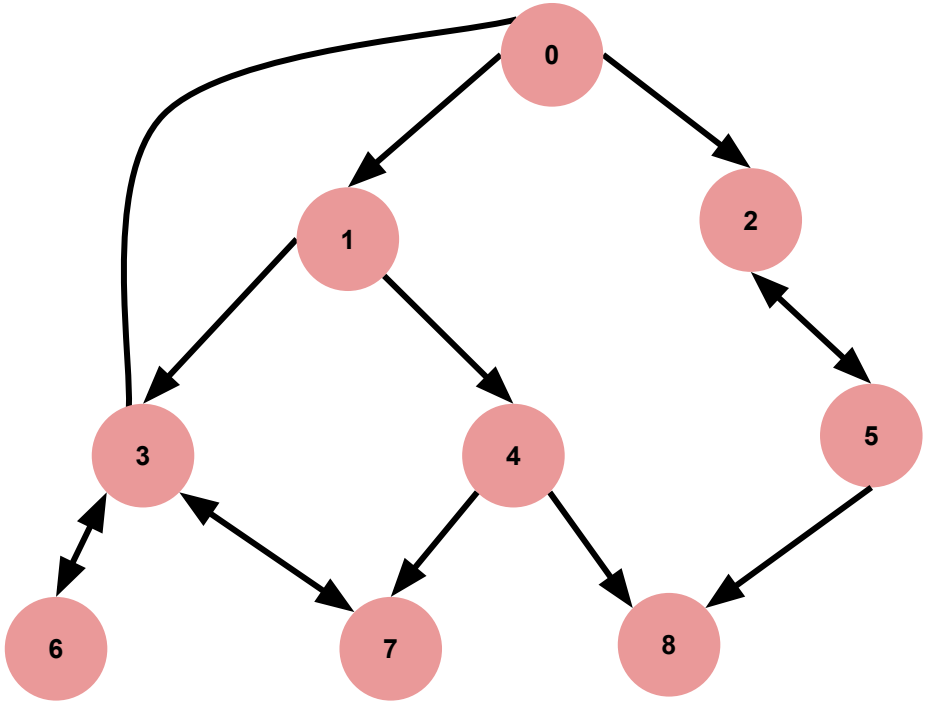
DFS

Depth-First Search is a way to explore connected nodes in a graph.

DFS goes deep before wide: fully explore one path before trying any others.

Analogy: In a cave with lots of tunnels.

Explore the first tunnel to its *completion* before moving on to the next.



Order of DFS: 0 1 3 6 7 4 8 2 5

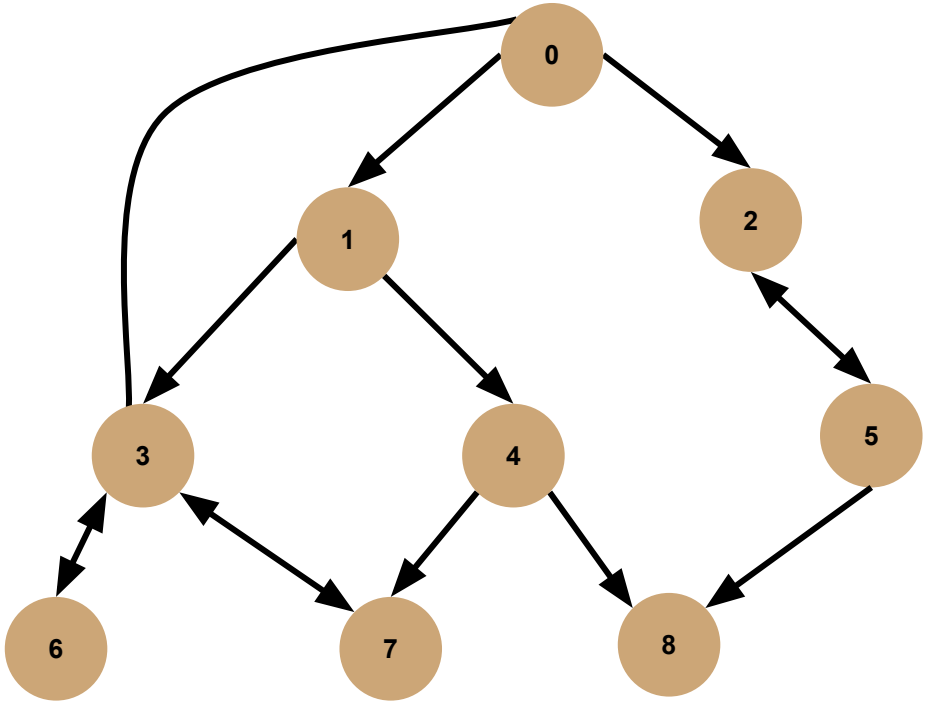
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: - - - - - - - - -

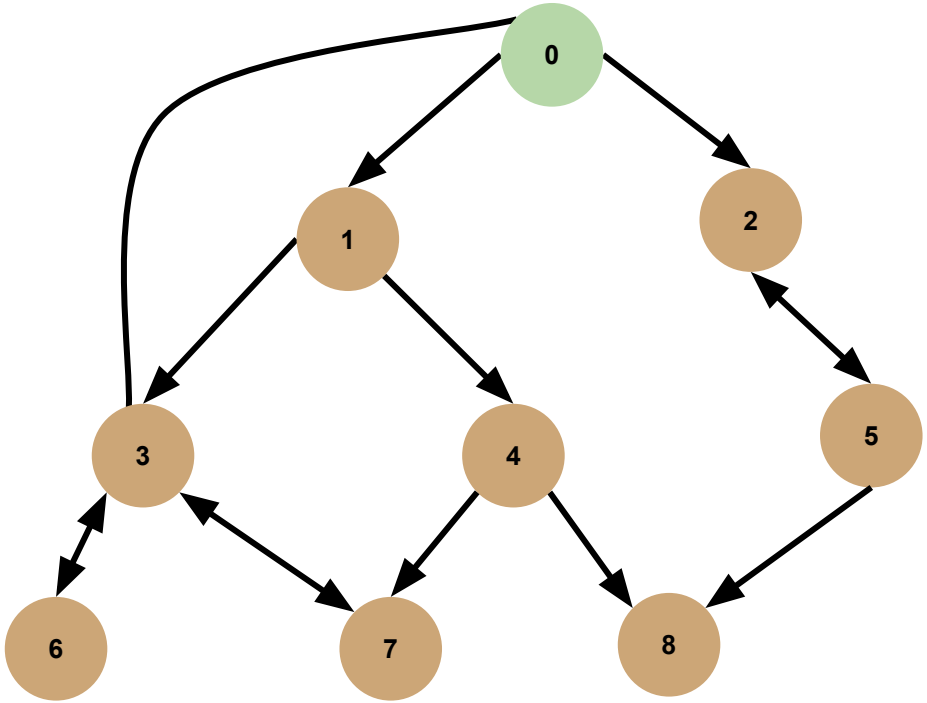
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 - - - - - - - -

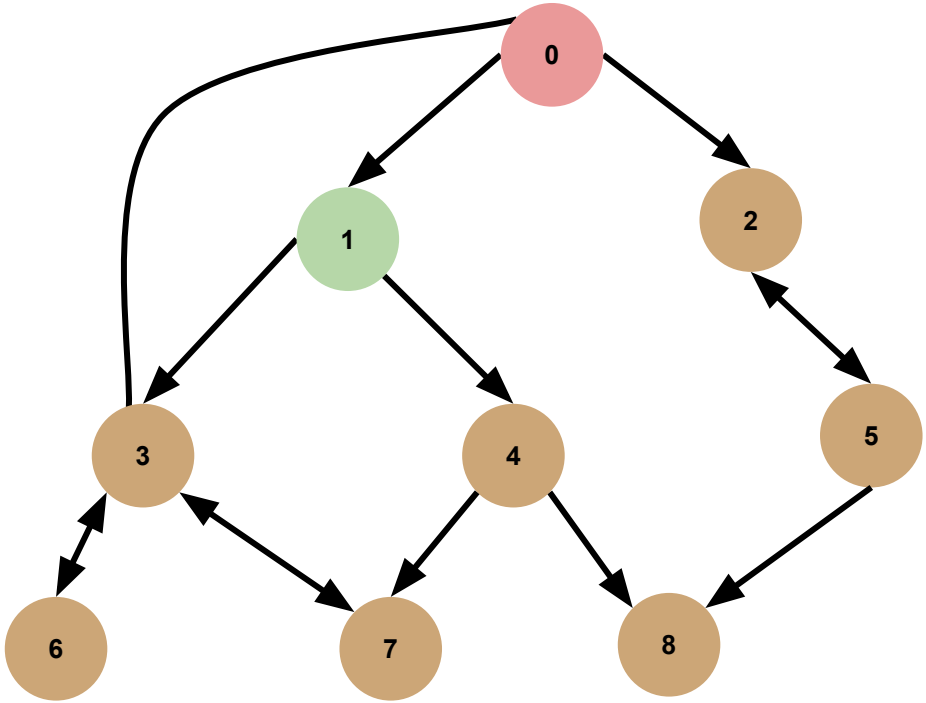
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 - - - - - - -

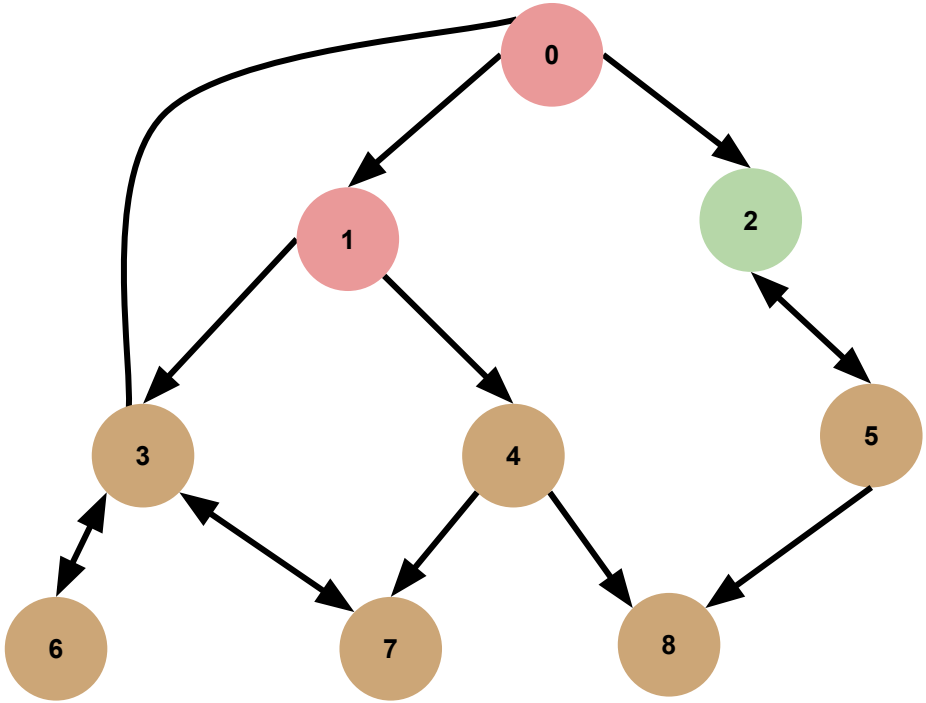
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 - - - - - -

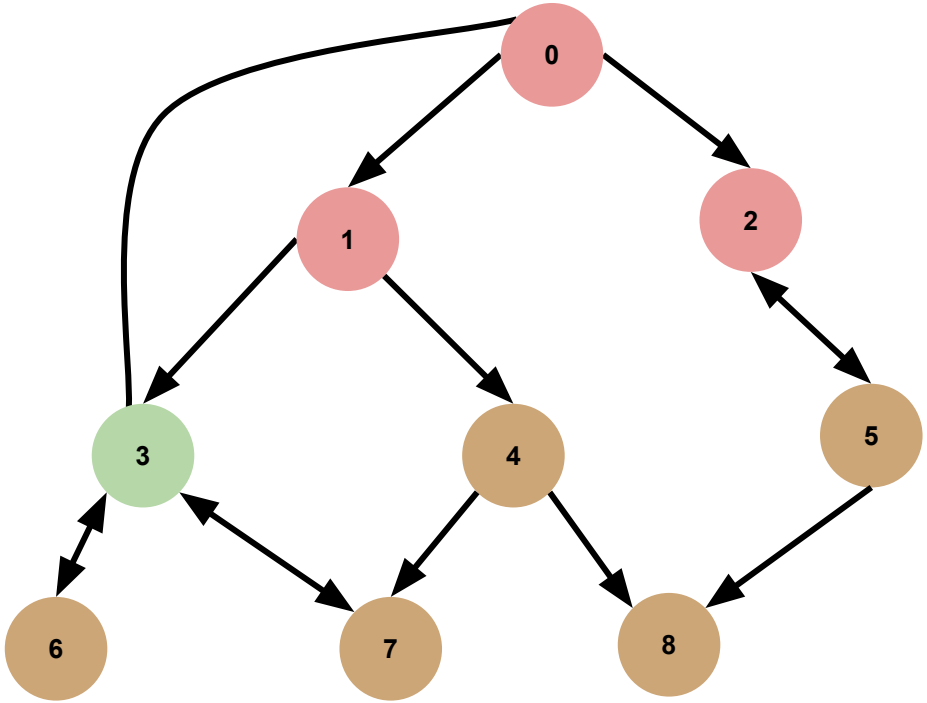
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 3 - - - - -

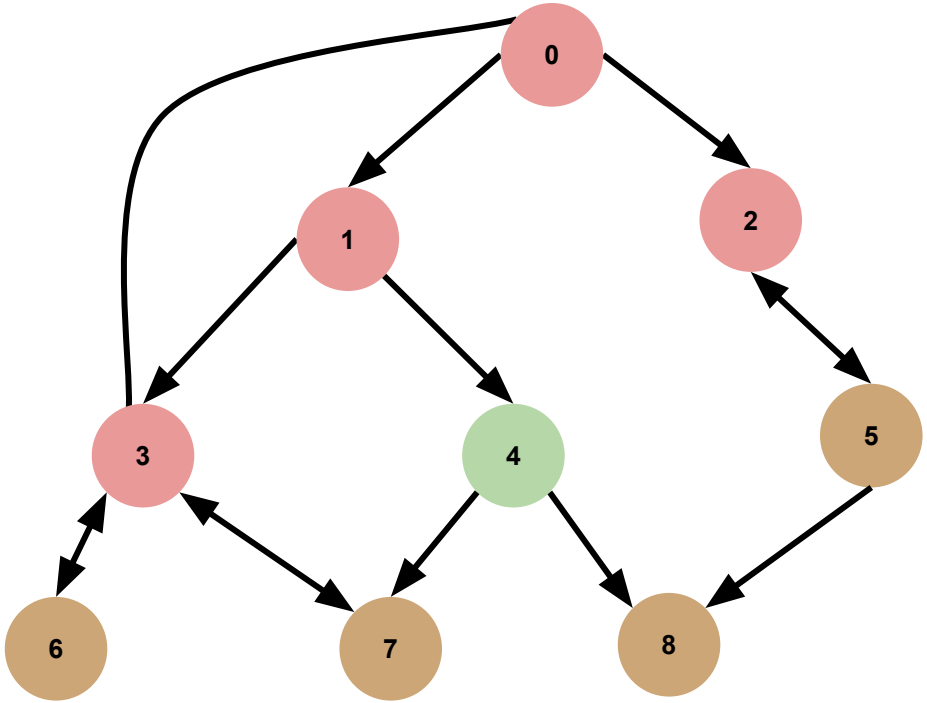
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 3 4 - - - -

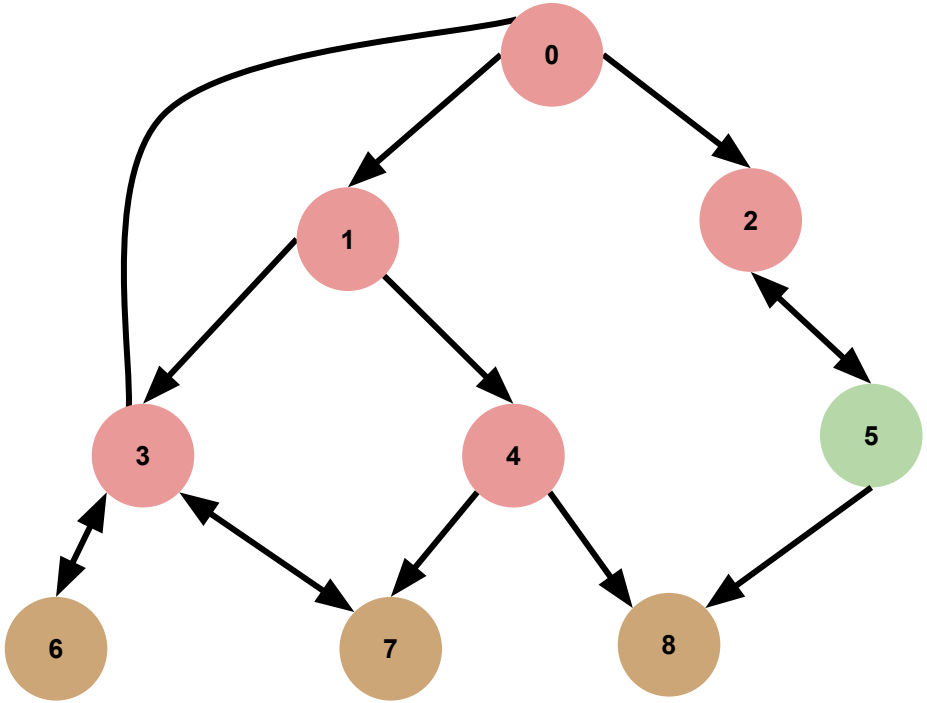
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 3 4 5 - - -

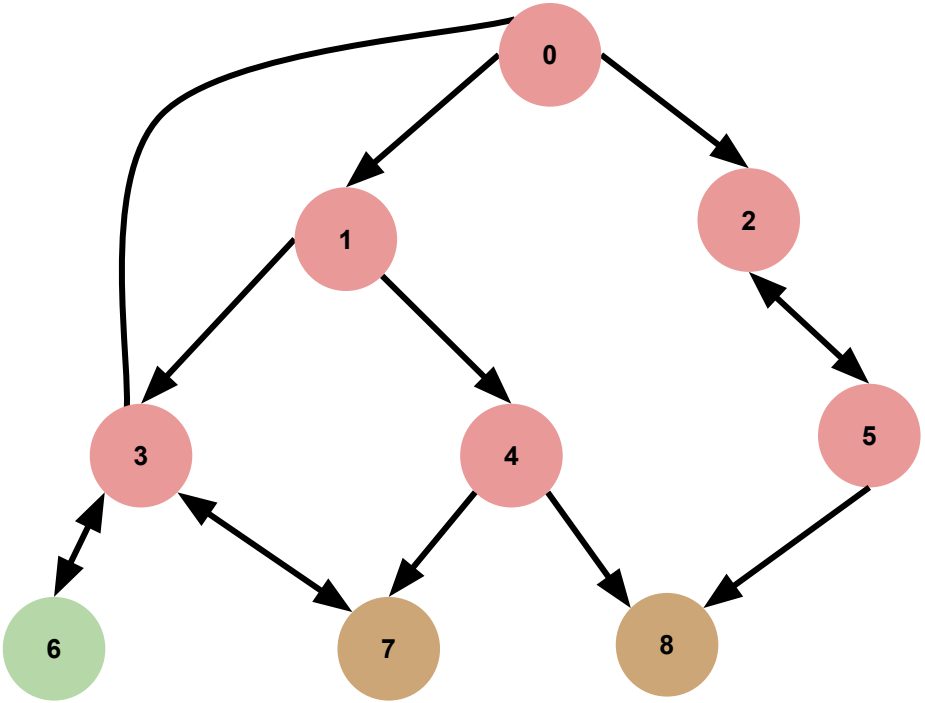
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 3 4 5 6 - -

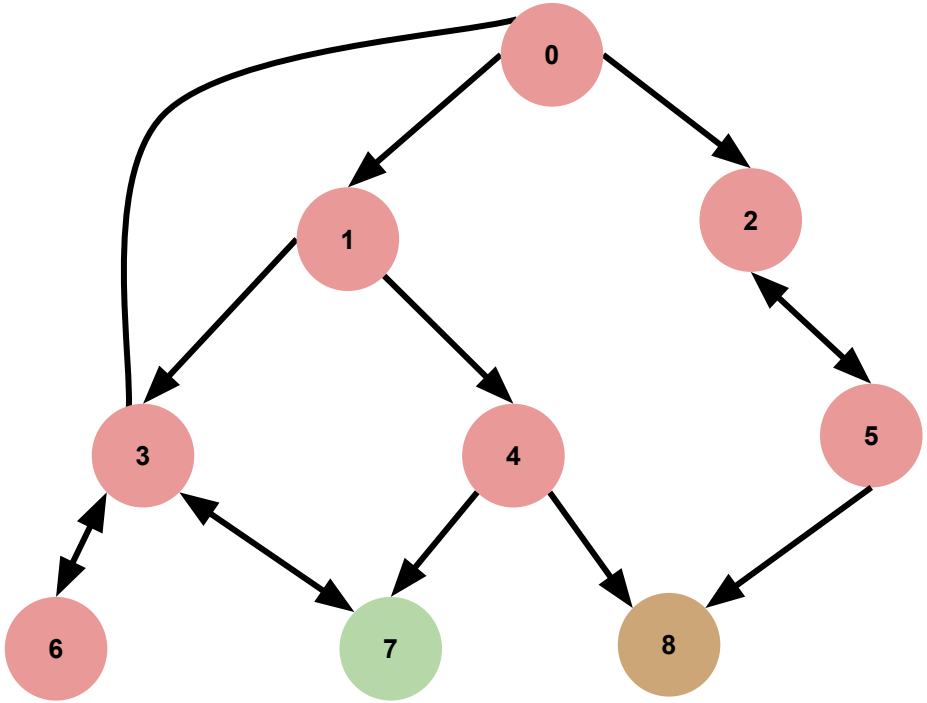
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 3 4 5 6 7 -

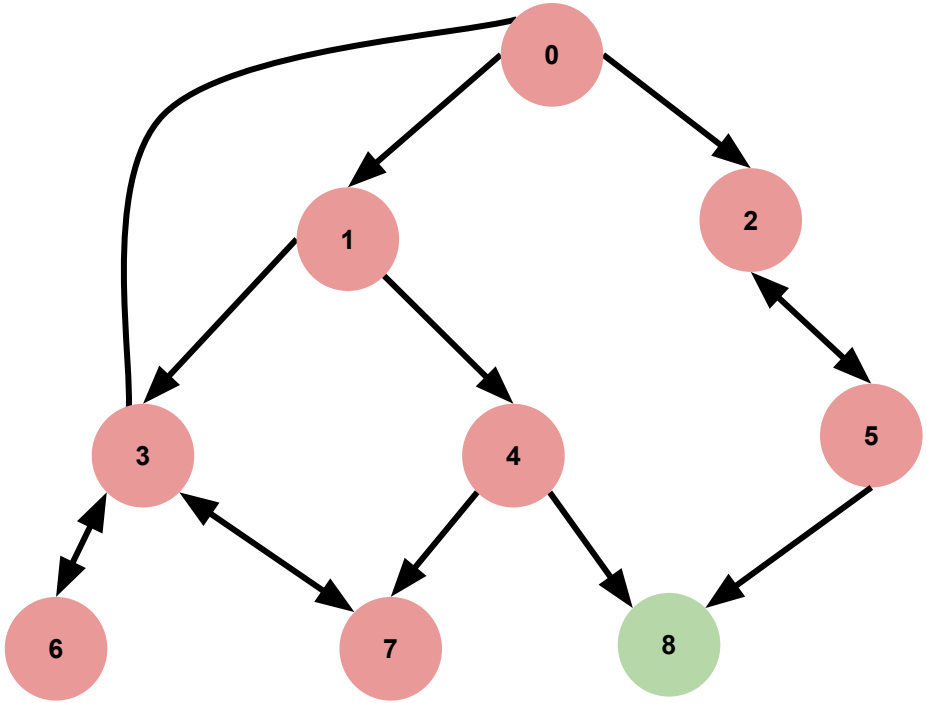
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 3 4 5 6 7 8

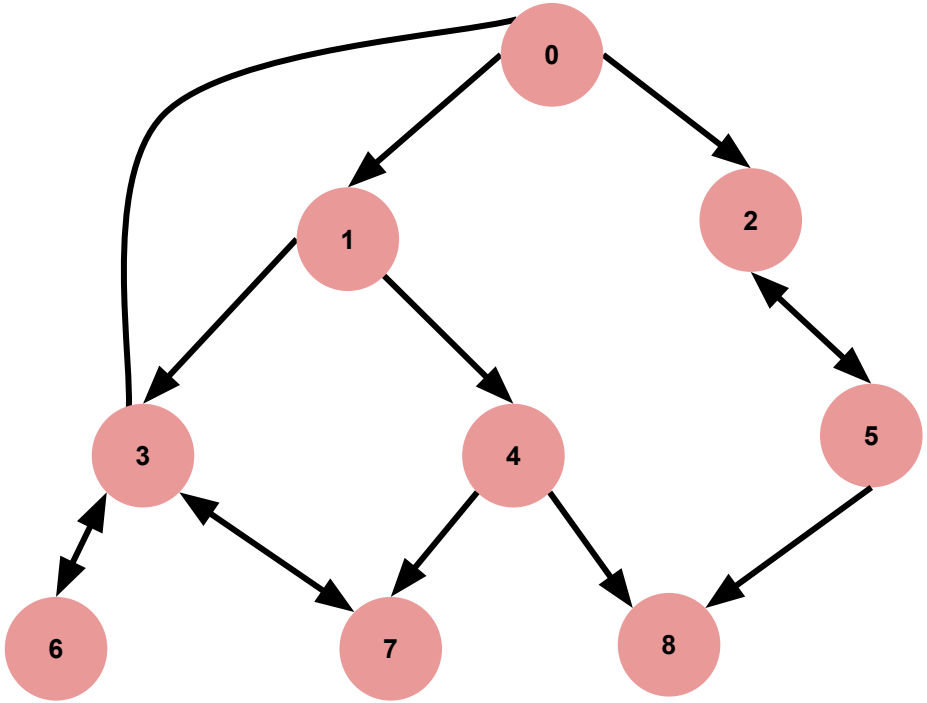
BFS

Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.



Order of BFS: 0 1 2 3 4 5 6 7 8

BFS

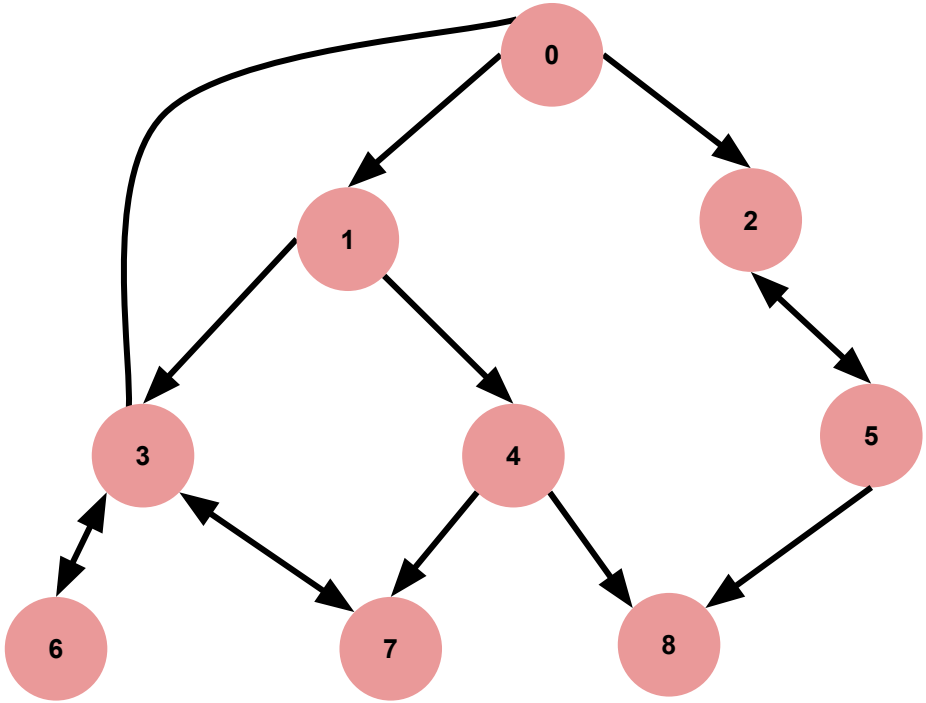
Breadth-First Search is another way to explore connected nodes in a graph.

BFS goes wide before deep: explore closest nodes first.

Analogy: In the center of a crowd.

Greet everyone standing in the first row. Then, green everyone in the second row, and so on.

Fun Fact: If edges are **unweighted**, then BFS can find the ***shortest path*** from the starting node to all nodes!



Order of BFS: 0 1 2 3 4 5 6 7 8

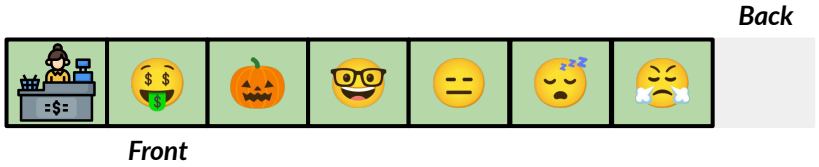
Queue

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**



Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Back

Front

Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Back

Front

Queue Example

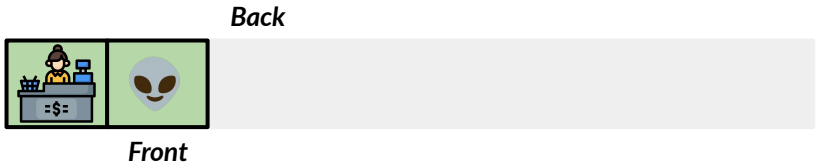
BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()



Queue Example

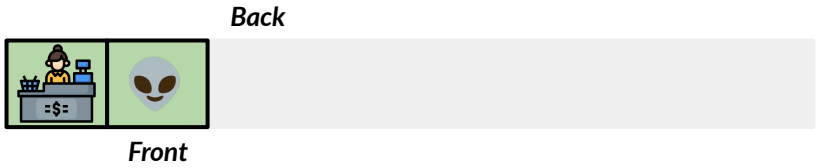
BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()



Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Queue Example

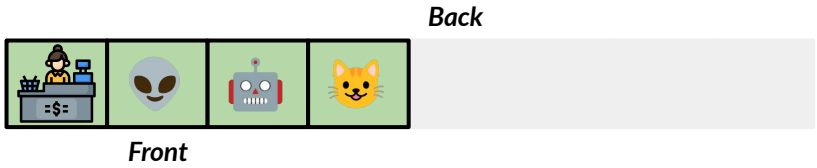
BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()



Queue Example

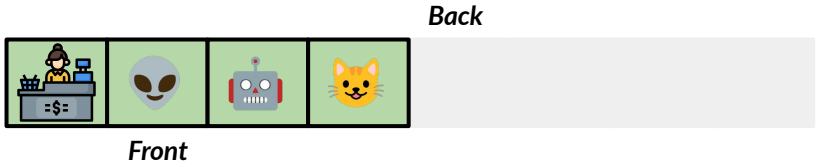
BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()



Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Queue Example

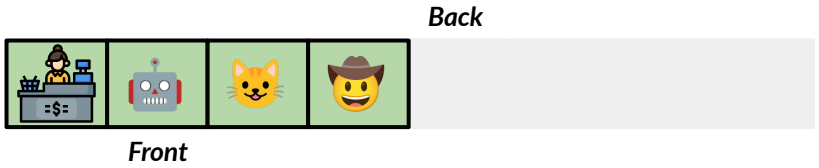
BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()



Queue Example

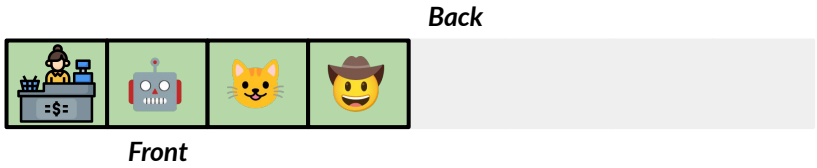
BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()



Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Queue Example

BFS utilizes a **queue**.

A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Queue Example

BFS utilizes a **queue**.

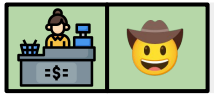
A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

Back



Front

Queue Example

BFS utilizes a **queue**.

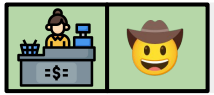
A **queue** is essentially a normal line at the grocery store.

Two main operations:

1. Enter the line from the back: **(Enqueue).**
2. Exit the line at the front: **(Dequeue).**

enqueue(👽)
enqueue(🤖)
enqueue(😺)
dequeue()
enqueue(🤠)
dequeue()
dequeue()

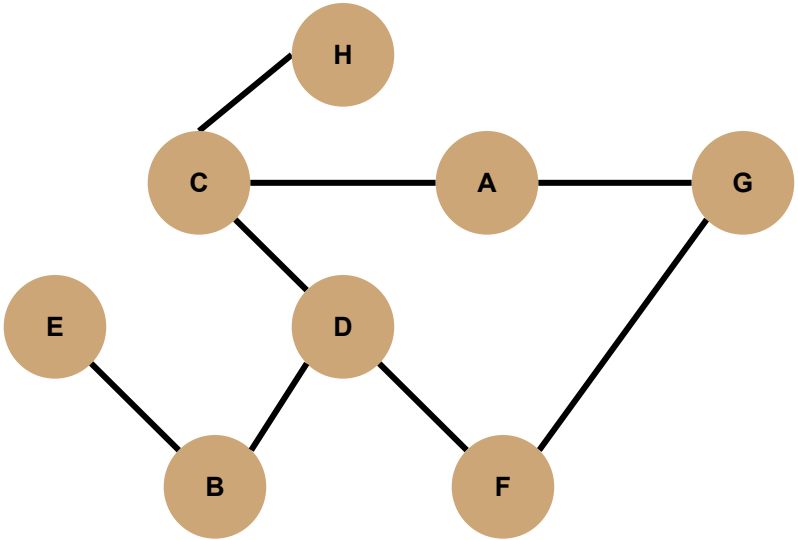
Back



Front

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS:

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

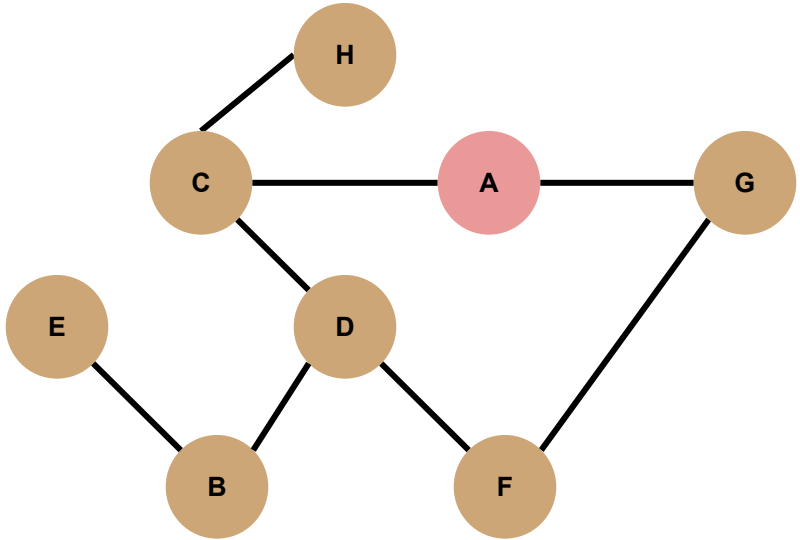
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	F
B	F
C	F
D	F
E	F
F	F
G	F
H	F

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS:

Initialize Queue with Starting Vertex & Mark it

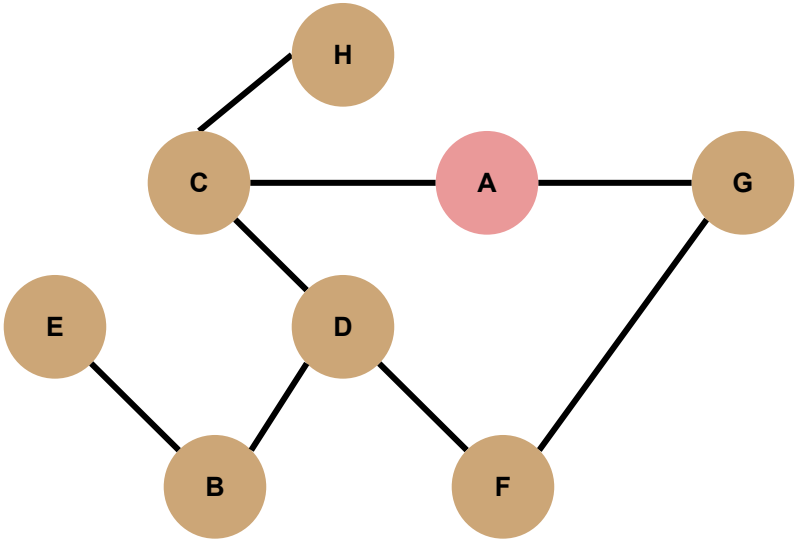
1. ***While Queue is not empty:***
   - ***Dequeue vertex v***
2. ***For every unmarked neighbor n:***
   - ***Mark neighbor***
   - ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	F
D	F
E	F
F	F
G	F
H	F

Queue: [A]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS:

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

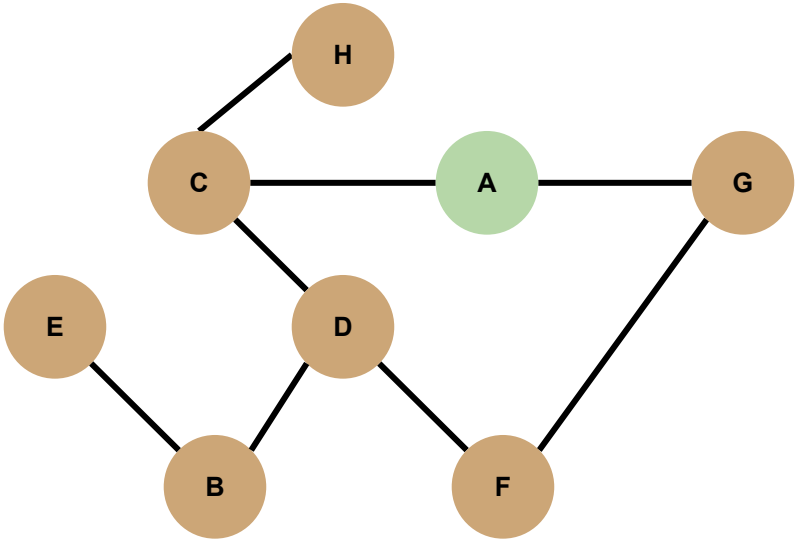
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	F
D	F
E	F
F	F
G	F
H	F

Queue: [A]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

- ***Mark neighbor***
- ***Enqueue n to Queue***

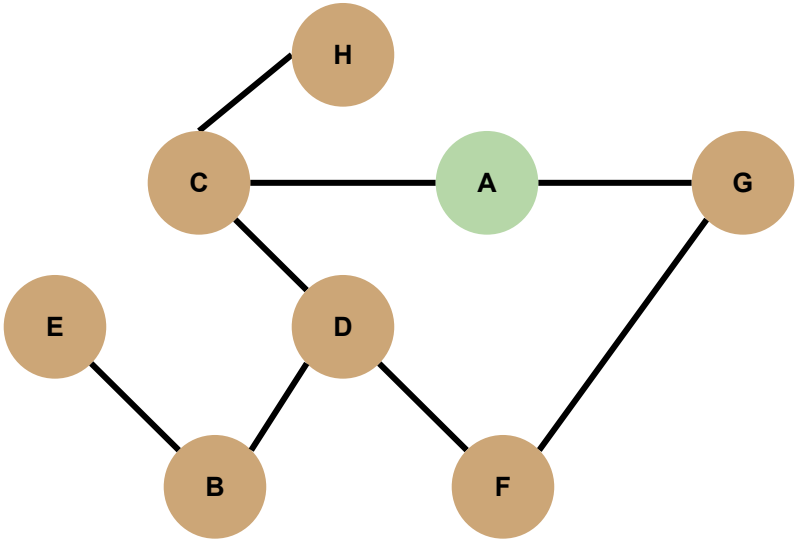
Vertex	marked[]
A	T
B	F
C	F
D	F
E	F
F	F
G	F
H	F

Order of BFS: A

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

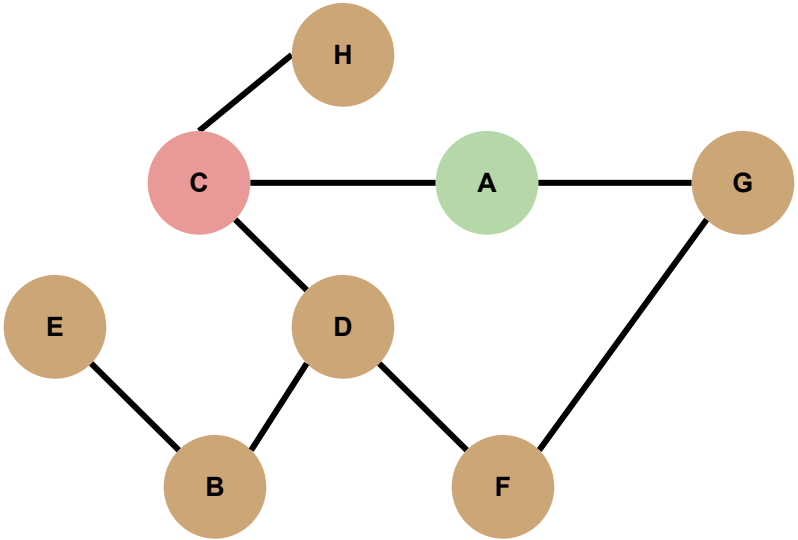
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	F
D	F
E	F
F	F
G	F
H	F

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

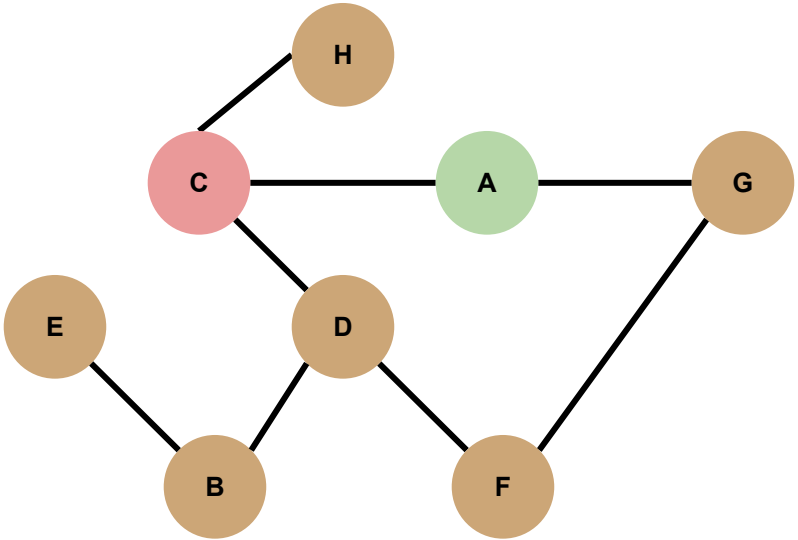
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	F
H	F

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

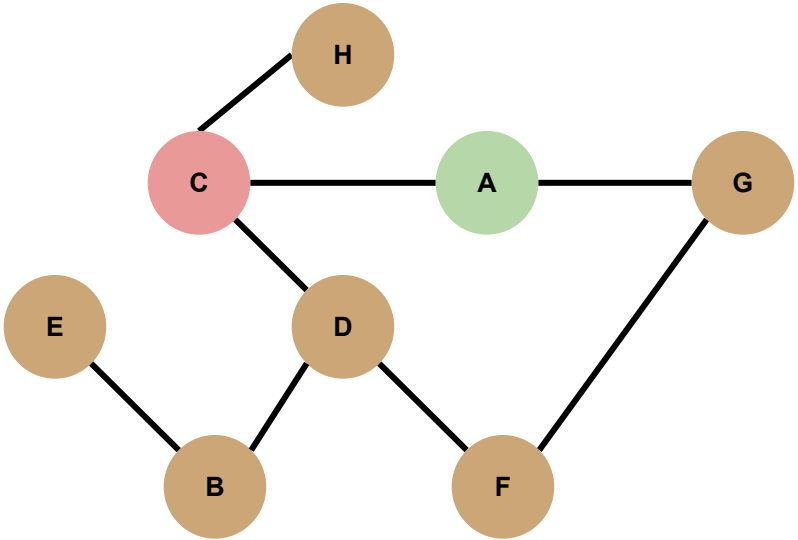
- *Mark neighbor*
- *Enqueue n to Queue*

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	F
H	F

Queue: [C]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

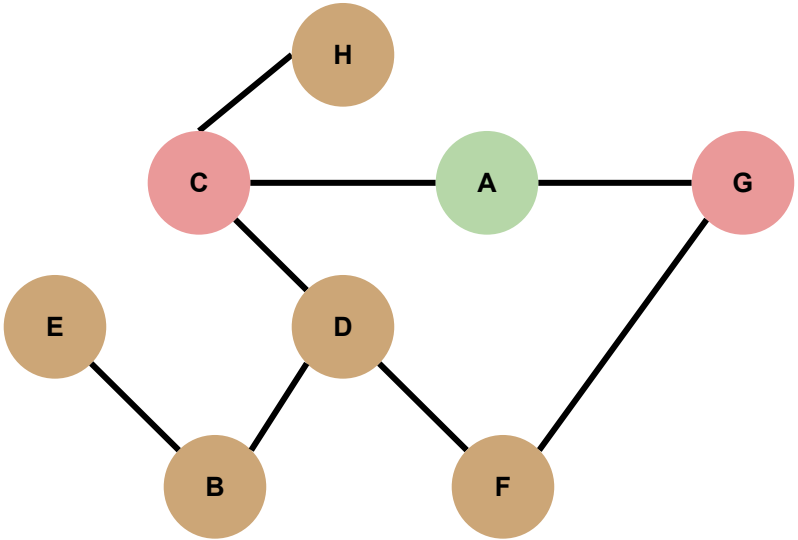
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	F
H	F

Queue: [C]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it
1. While Queue is not empty:
- ***Dequeue vertex v***

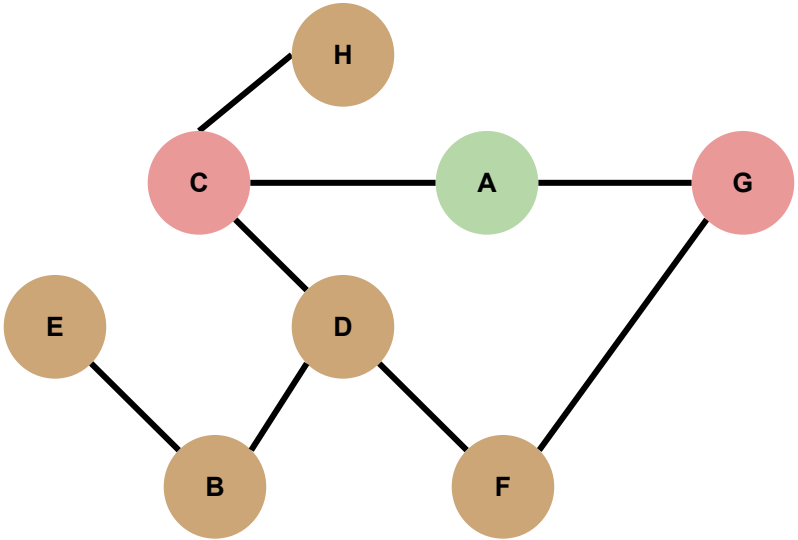
2. For every unmarked neighbor n:
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	T
H	F

Queue: [C]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

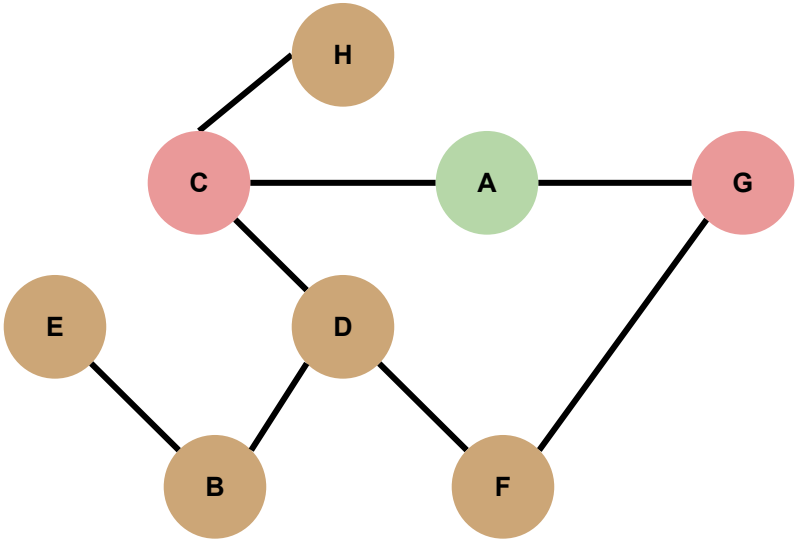
- *Mark neighbor*
- *Enqueue n to Queue*

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	T
H	F

Queue: [C, G]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

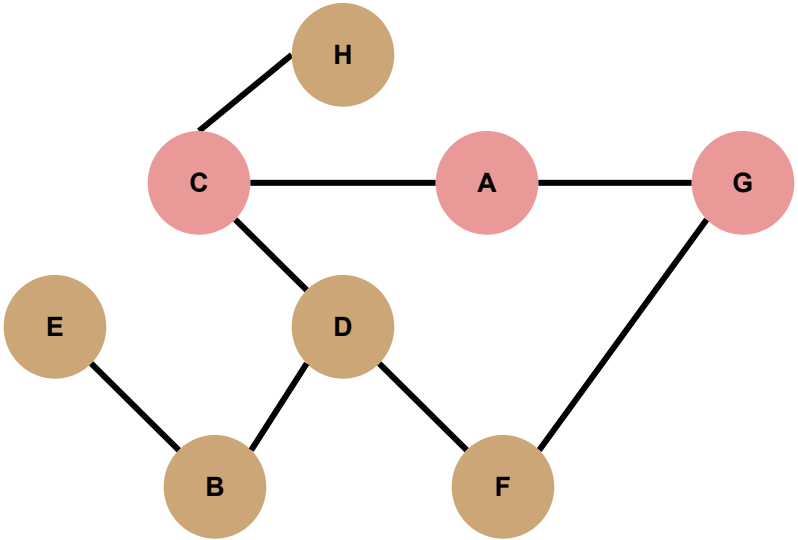
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	T
H	F

Queue: [C, G]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

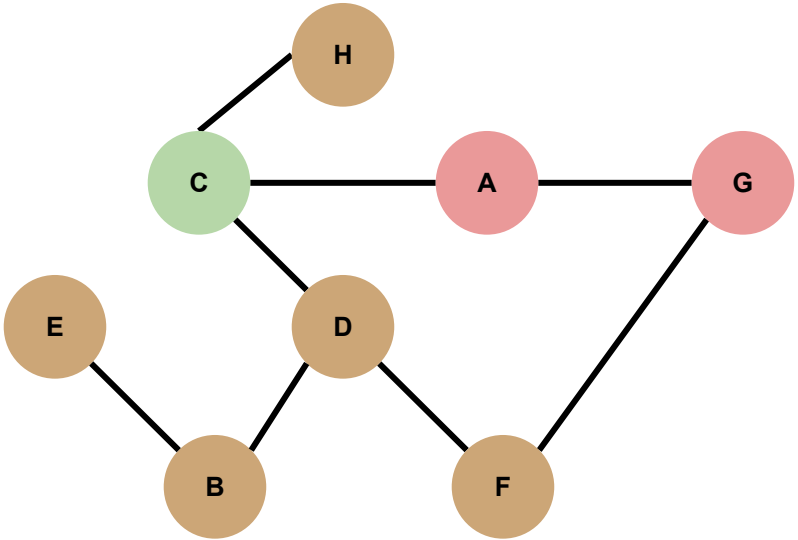
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	T
H	F

Queue: [C, G]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

- ***Mark neighbor***
- ***Enqueue n to Queue***

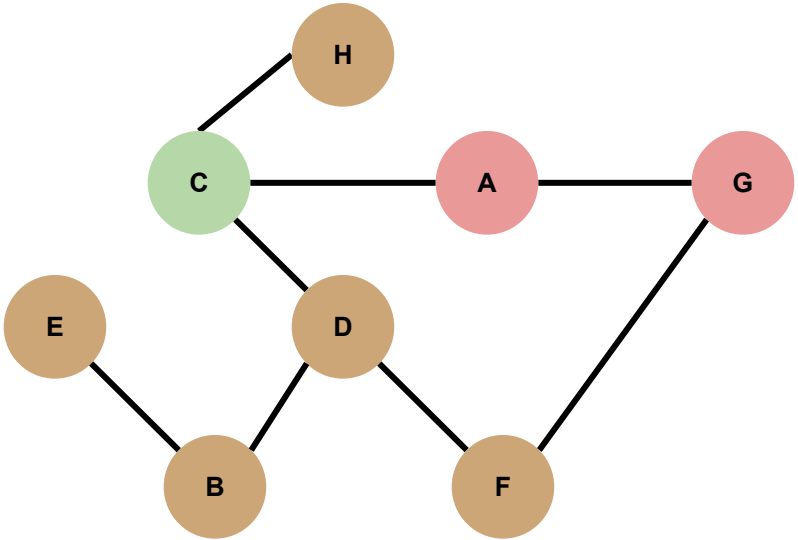
Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	T
H	F

Order of BFS: A, C

Queue: [G]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it
1. While Queue is not empty:
- ***Dequeue vertex v***

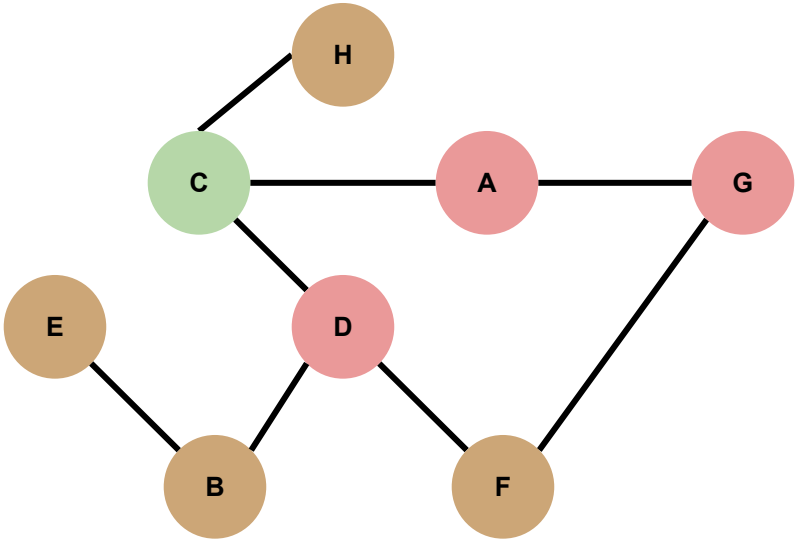
2. For every unmarked neighbor n:
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	F
E	F
F	F
G	T
H	F

Queue: [G]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it
1. While Queue is not empty:
- ***Dequeue vertex v***

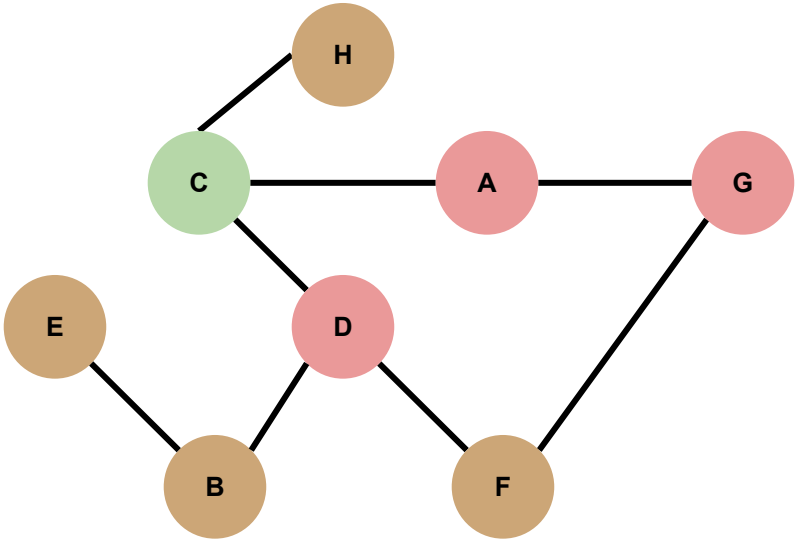
2. For every unmarked neighbor n:
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	F

Queue: [G]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

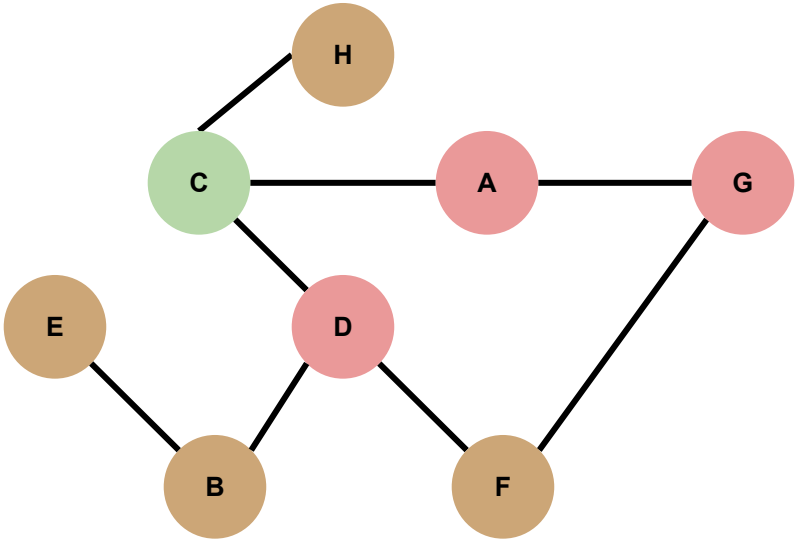
- *Mark neighbor*
- *Enqueue n to Queue*

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	F

Queue: [G, D]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

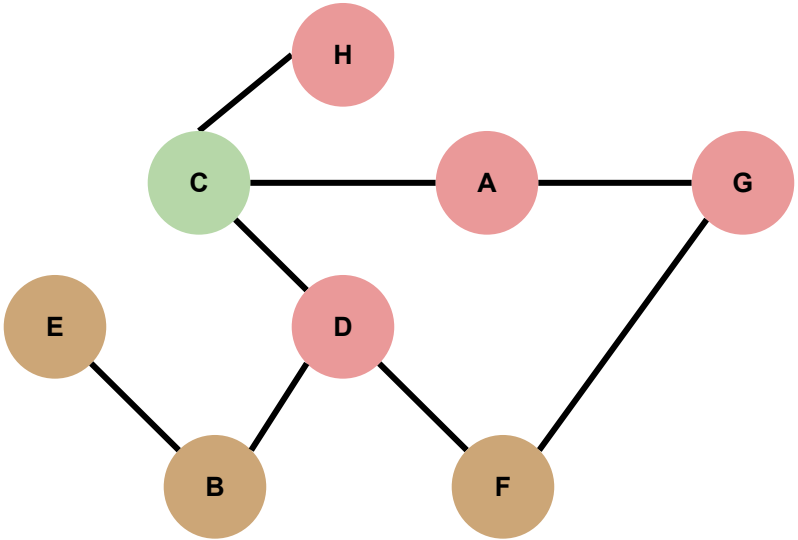
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	F

Queue: [G, D]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

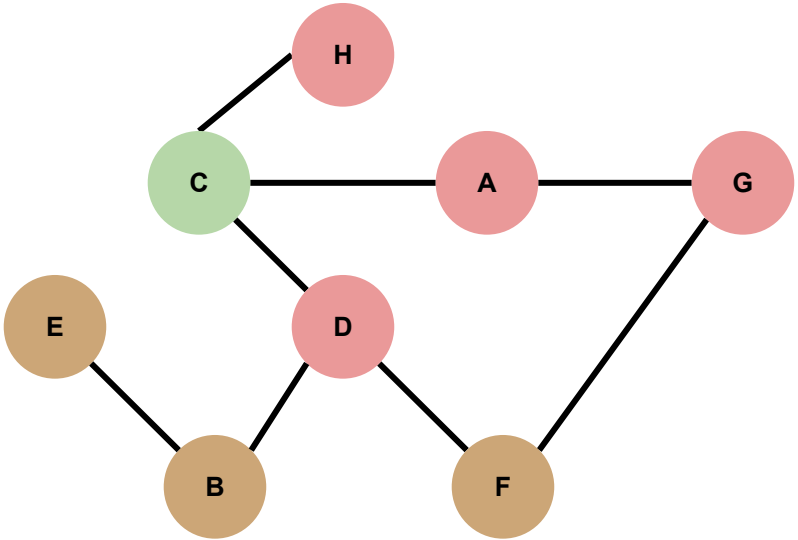
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	T

Queue: [G, D]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

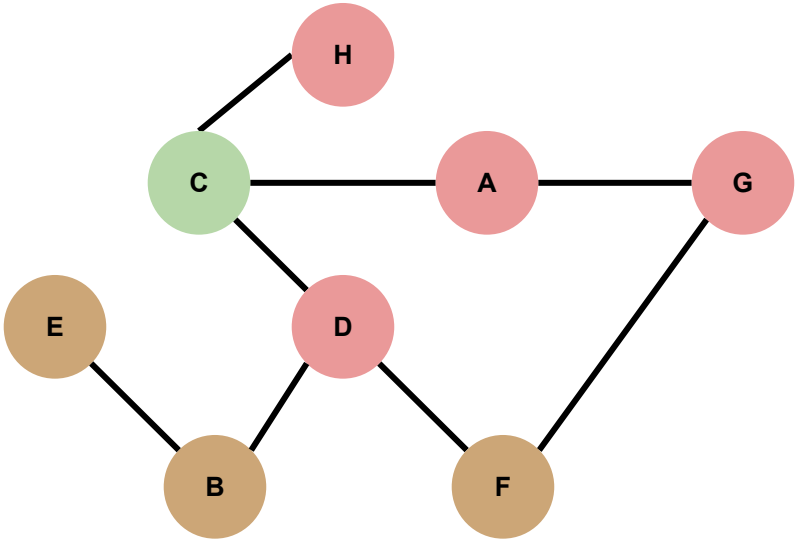
- *Mark neighbor*
- *Enqueue n to Queue*

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	T

Queue: [G, D, H]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

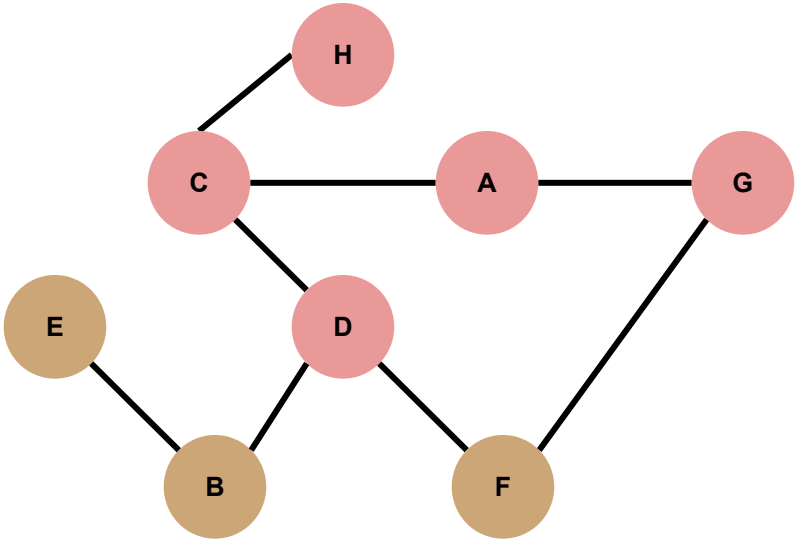
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	T

Queue: [G, D, H]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

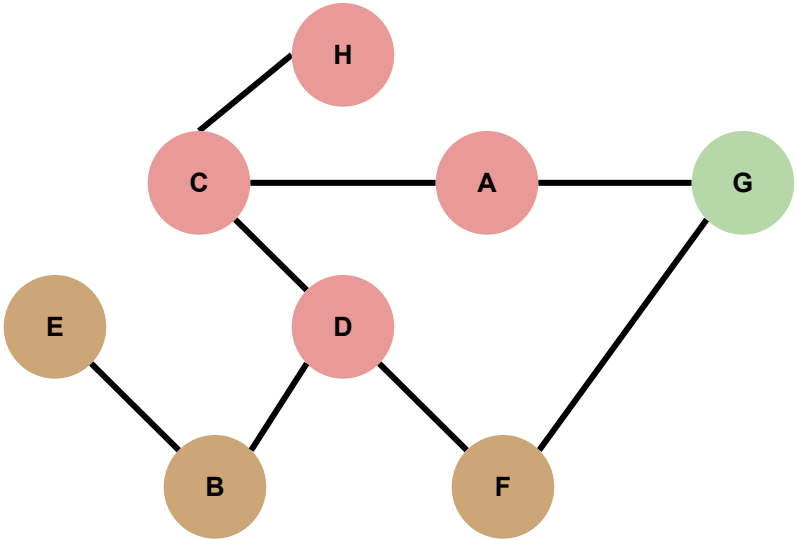
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	T

Queue: [G, D, H]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

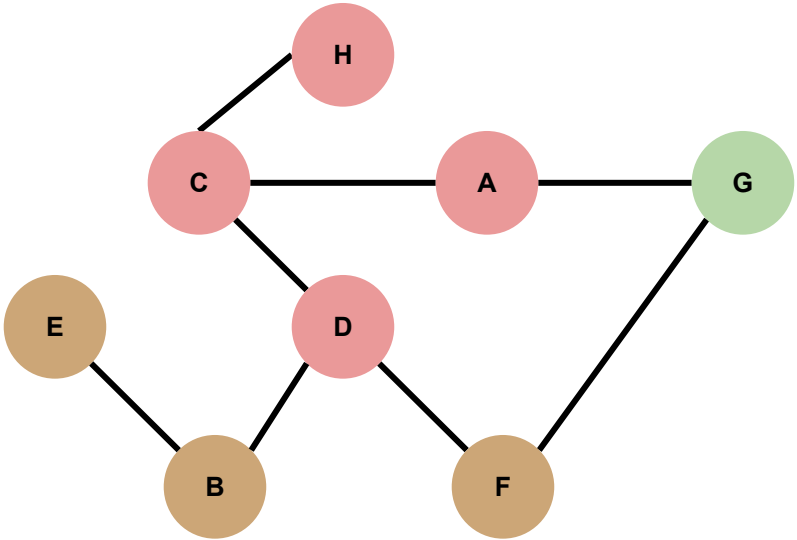
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	T

Queue: [D, H]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

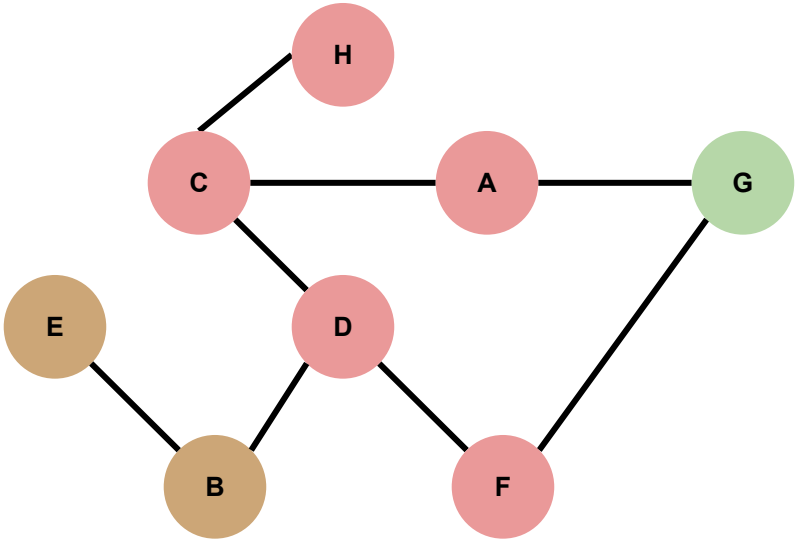
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	F
G	T
H	T

Queue: [D, H]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

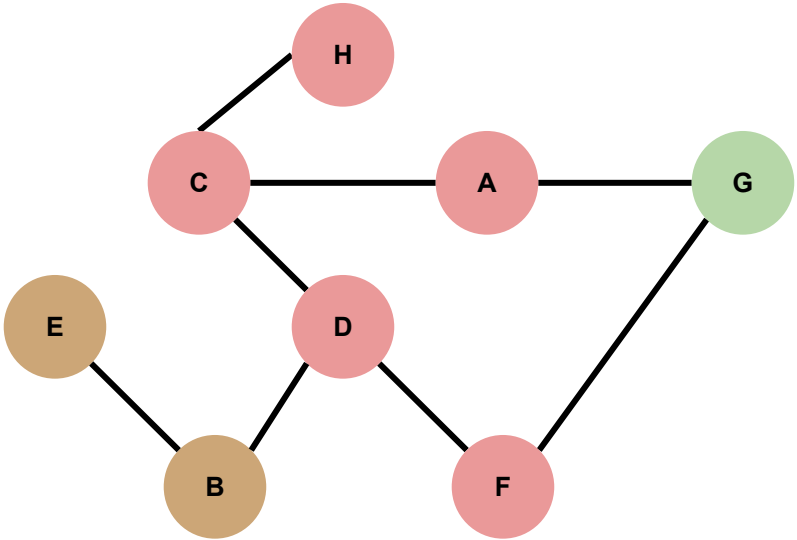
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [D, H]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

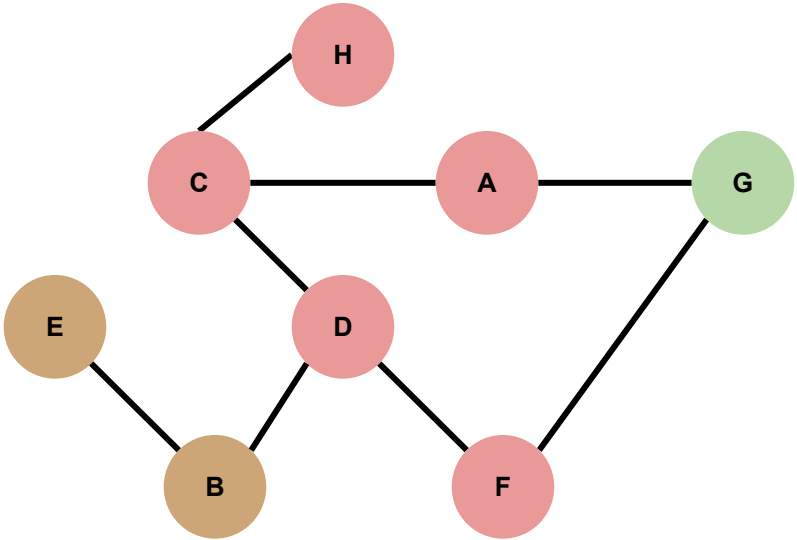
- *Mark neighbor*
- *Enqueue n to Queue*

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [D, H, F]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

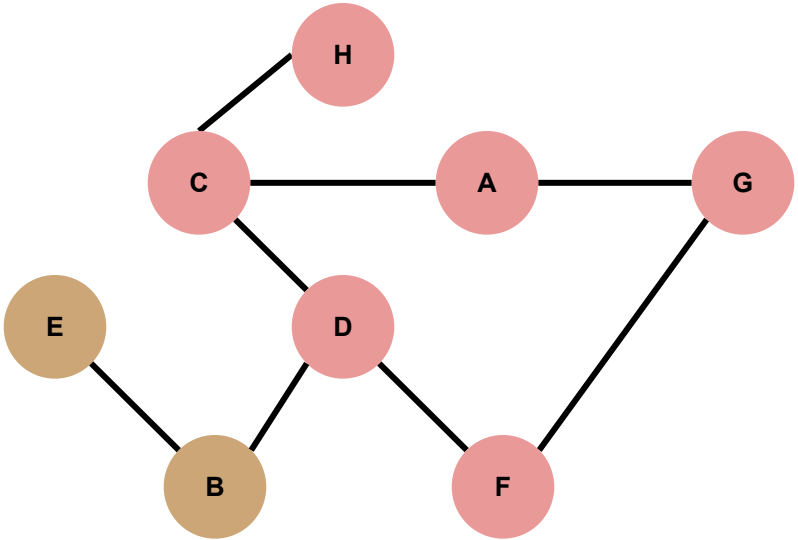
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [D, H, F]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [D, H, F]

BFS

Which vertex gets visited next? What does the queue look like after visiting?

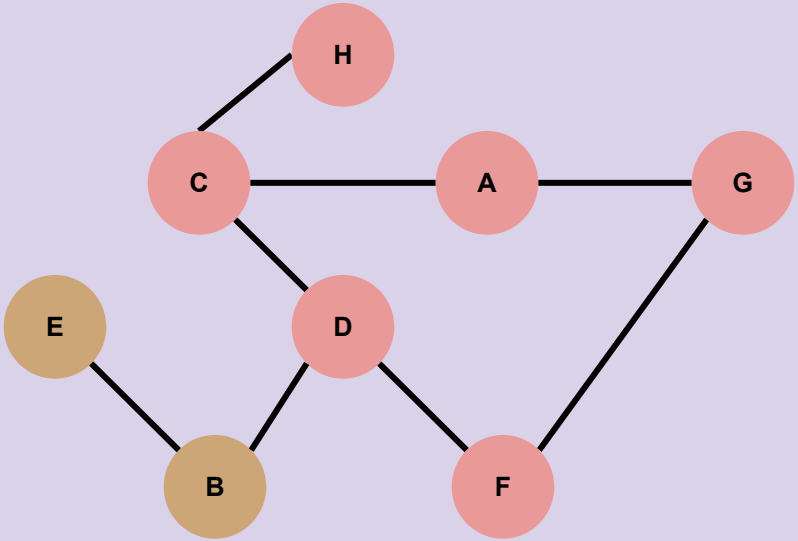
Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

- *Mark neighbor*
- *Enqueue n to Queue*



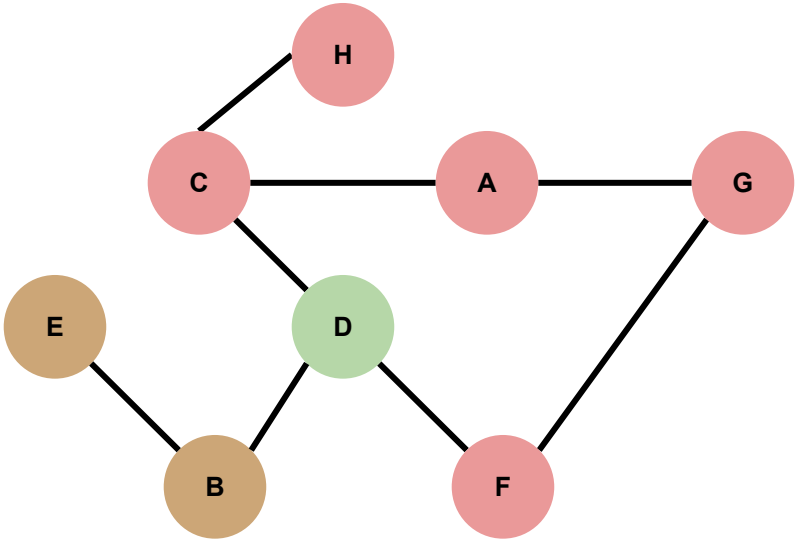
Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	T
G	T
H	T

Order of BFS: A, C, G

Queue: [D, H, F]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

- ***Mark neighbor***
- ***Enqueue n to Queue***

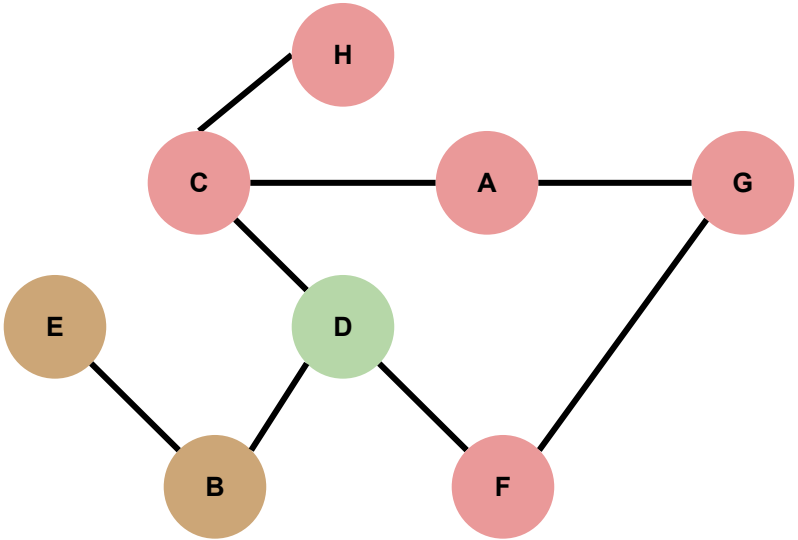
Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	T
G	T
H	T

Order of BFS: A, C, G, D

Queue: [H, F]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

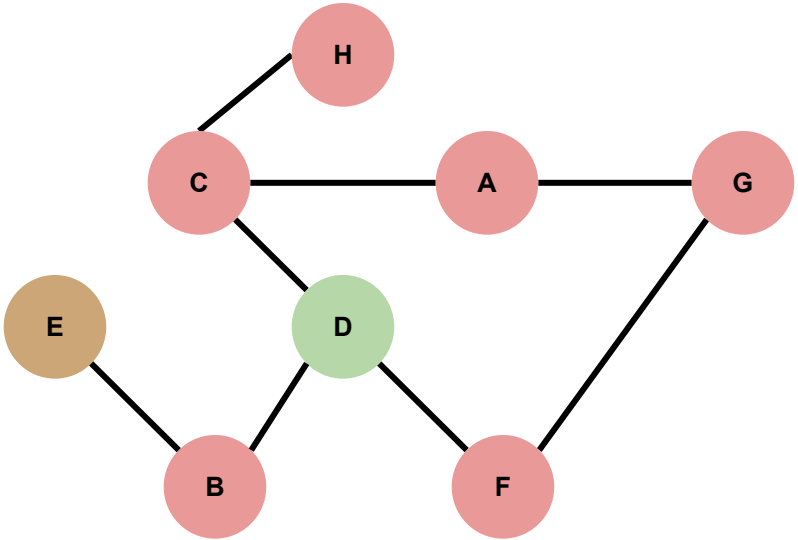
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	F
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [H, F]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

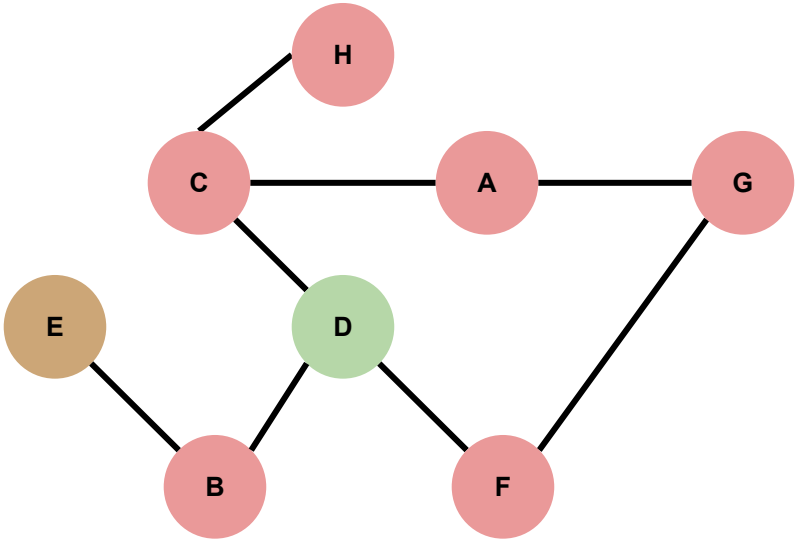
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [H, F]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

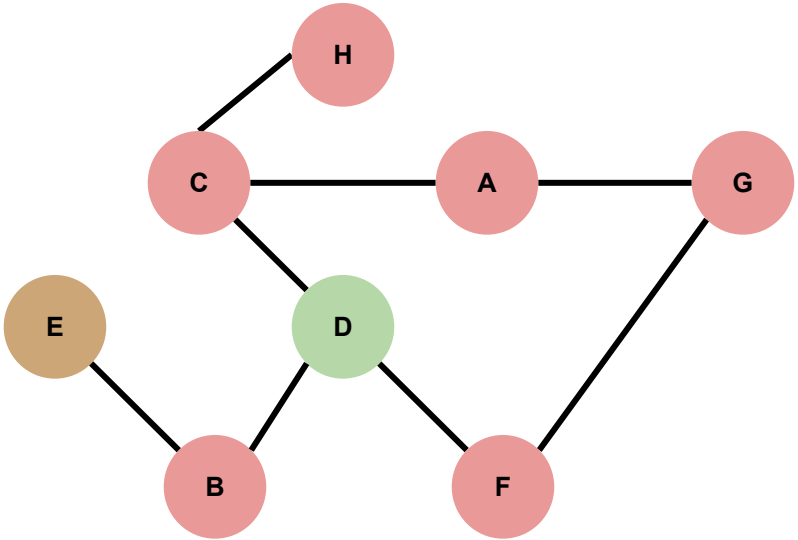
- *Mark neighbor*
- *Enqueue n to Queue*

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [H, F, B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

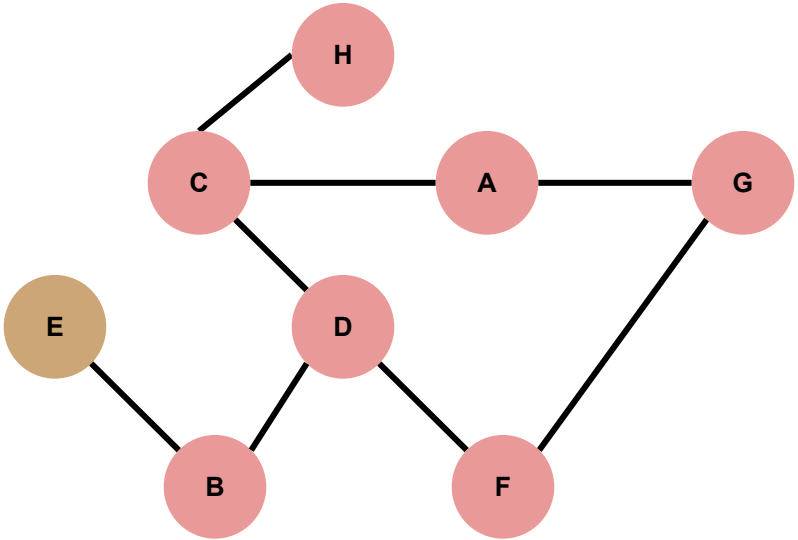
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [H, F, B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

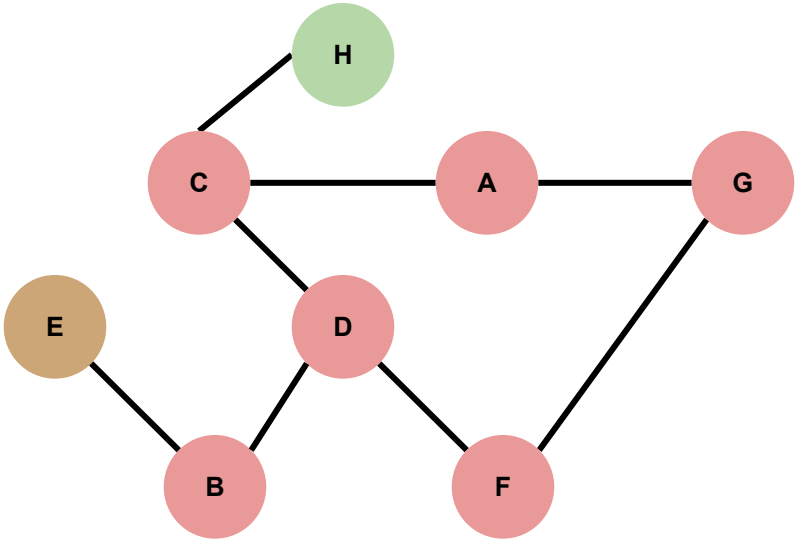
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [H, F, B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

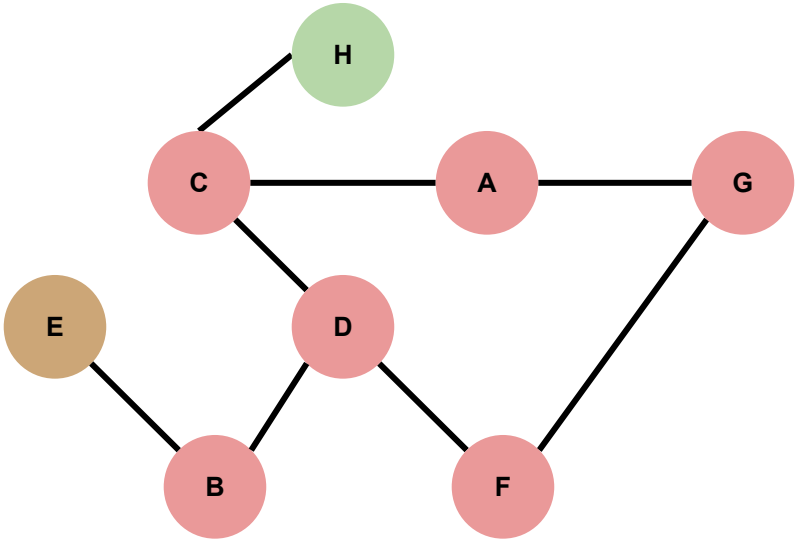
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [F, B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

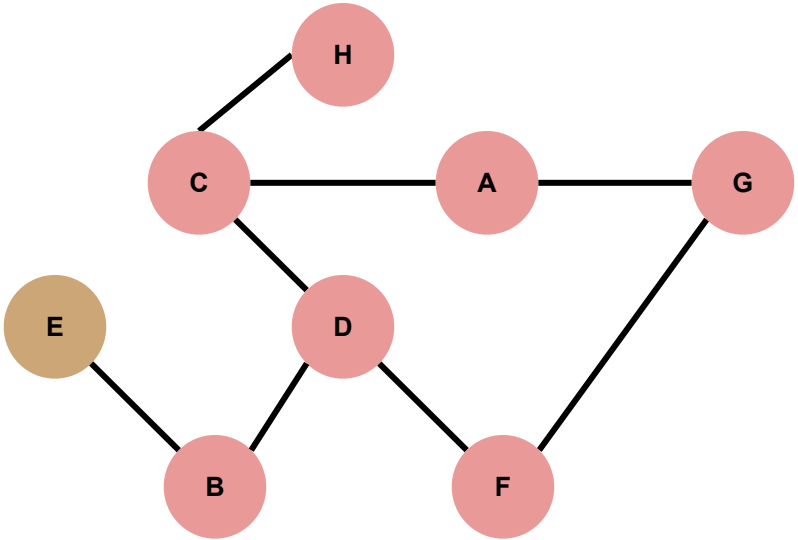
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [F, B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

- ***Mark neighbor***
- ***Enqueue n to Queue***

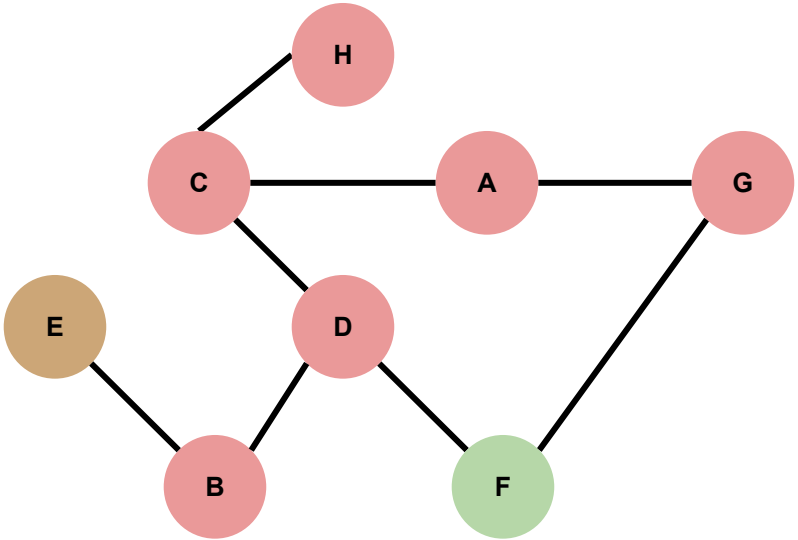
Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Order of BFS: A, C, G, D, H

Queue: [F, B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

- ***Mark neighbor***
- ***Enqueue n to Queue***

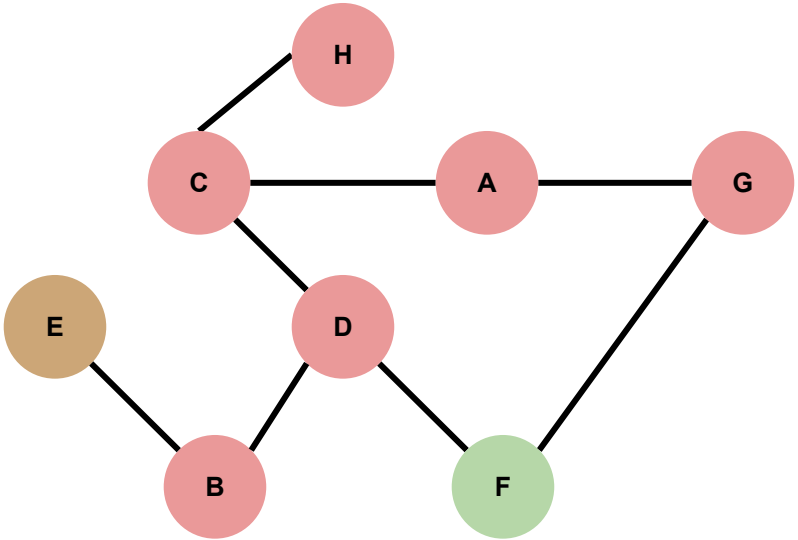
Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Order of BFS: A, C, G, D, H, F

Queue: [B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

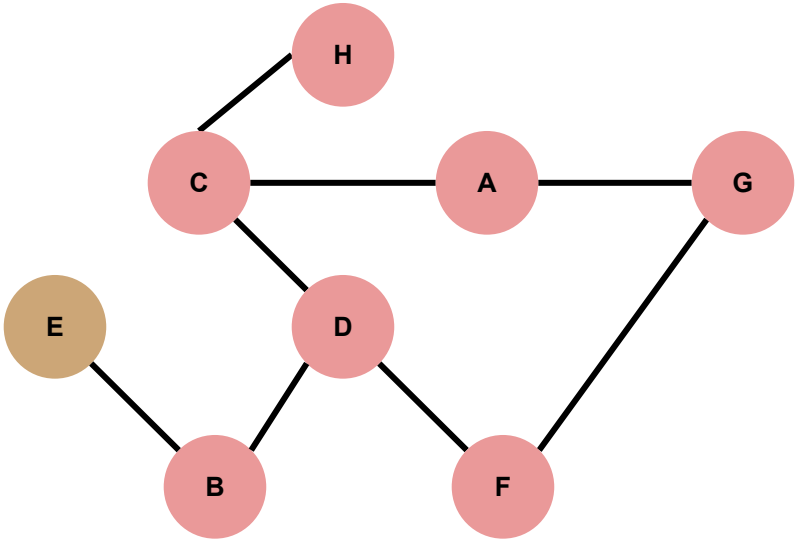
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

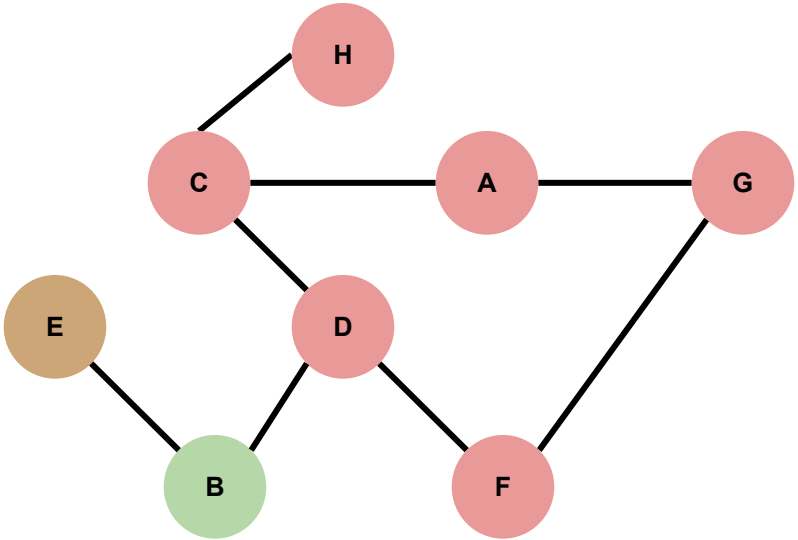
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: [B]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

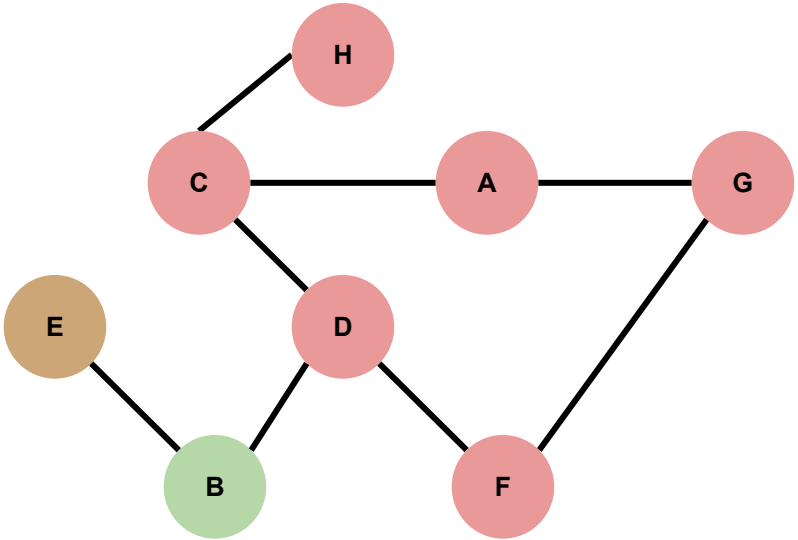
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B

Initialize Queue with Starting Vertex & Mark it
1. While Queue is not empty:
- ***Dequeue vertex v***

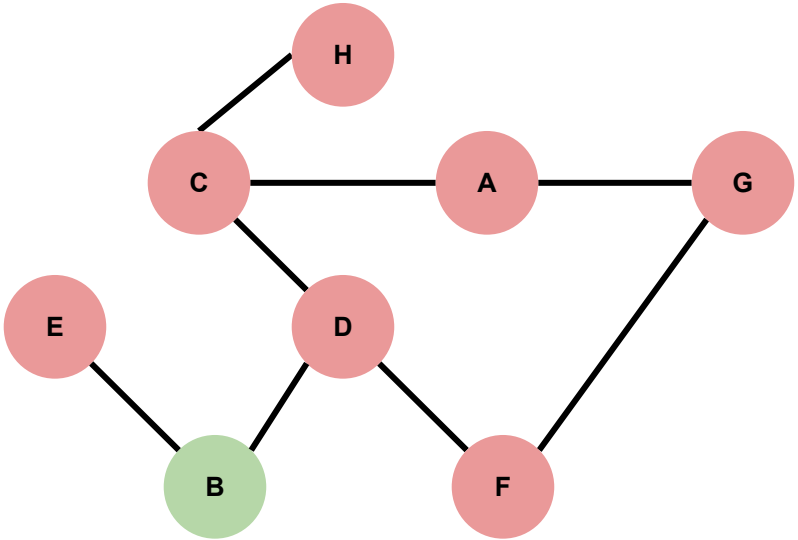
2. For every unmarked neighbor n:
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	F
F	T
G	T
H	T

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

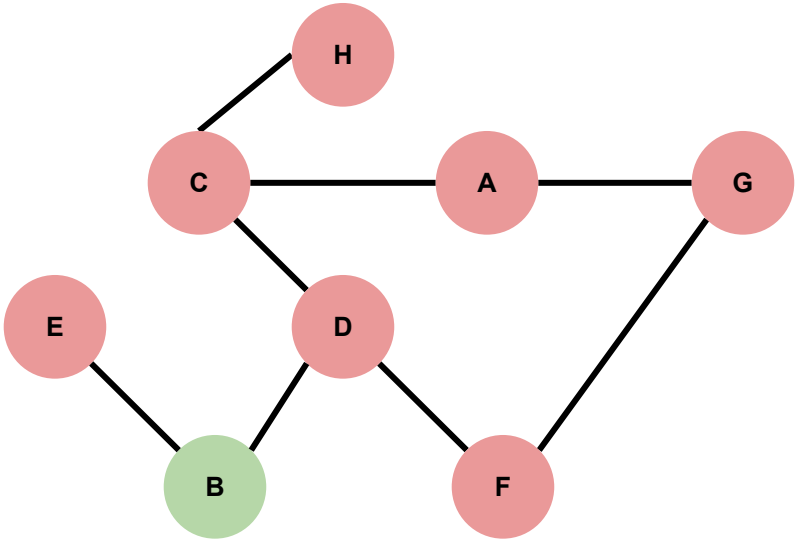
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- *Dequeue vertex v*

2. For every unmarked neighbor n:

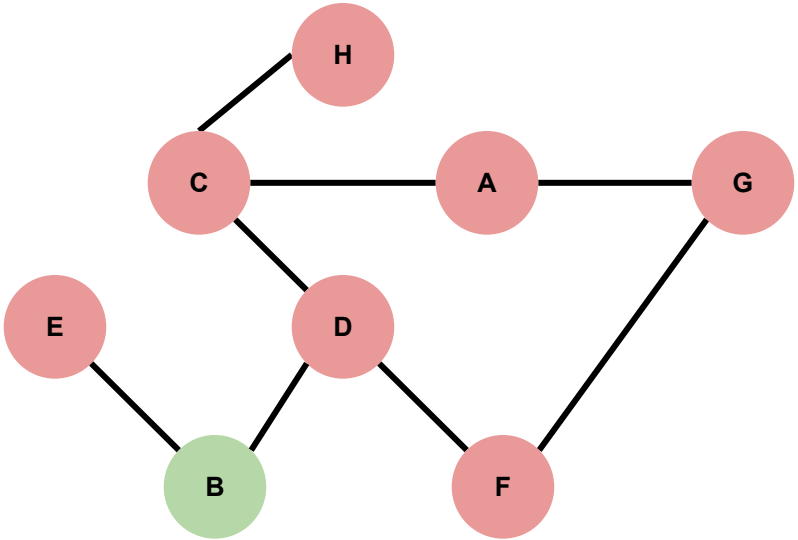
- *Mark neighbor*
- *Enqueue n to Queue*

Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Queue: [E]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

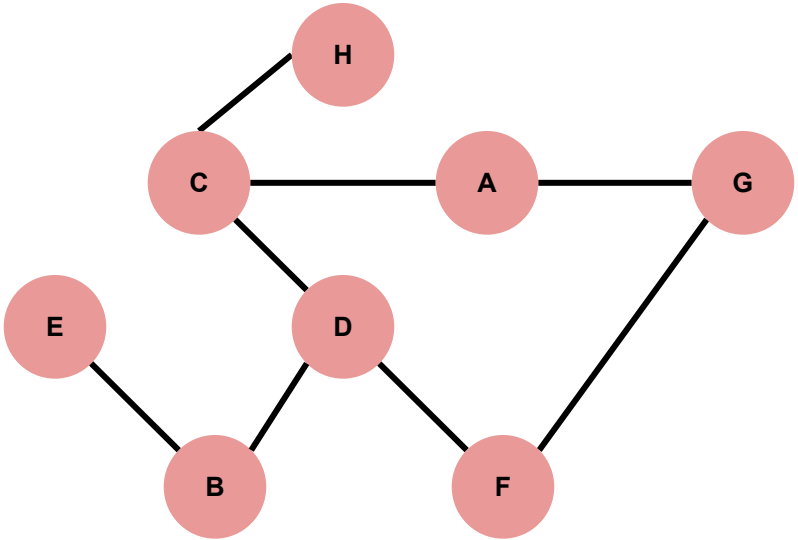
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Queue: [E]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

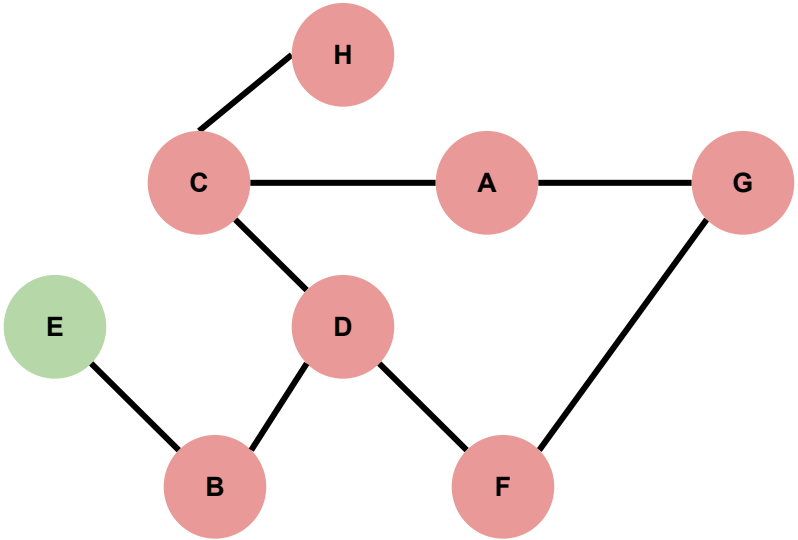
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Queue: [E]

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

- ***Mark neighbor***
- ***Enqueue n to Queue***

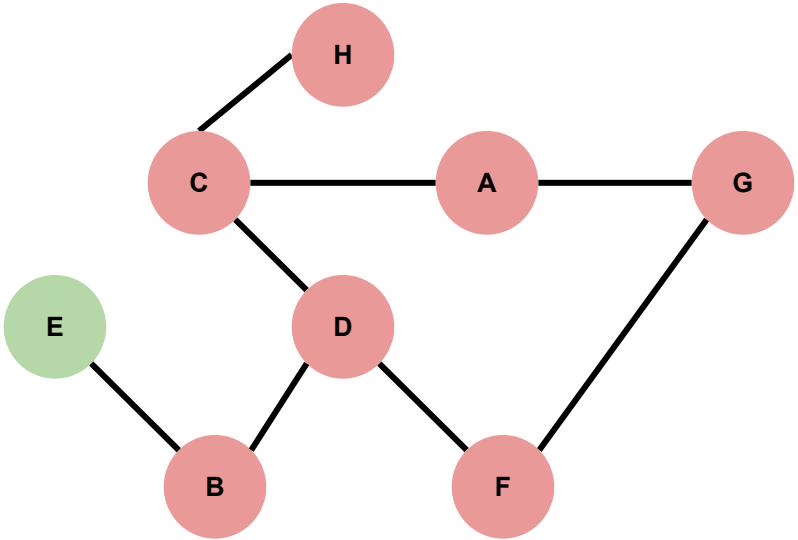
Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Order of BFS: A, C, G, D, H, F, B, E

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B, E

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

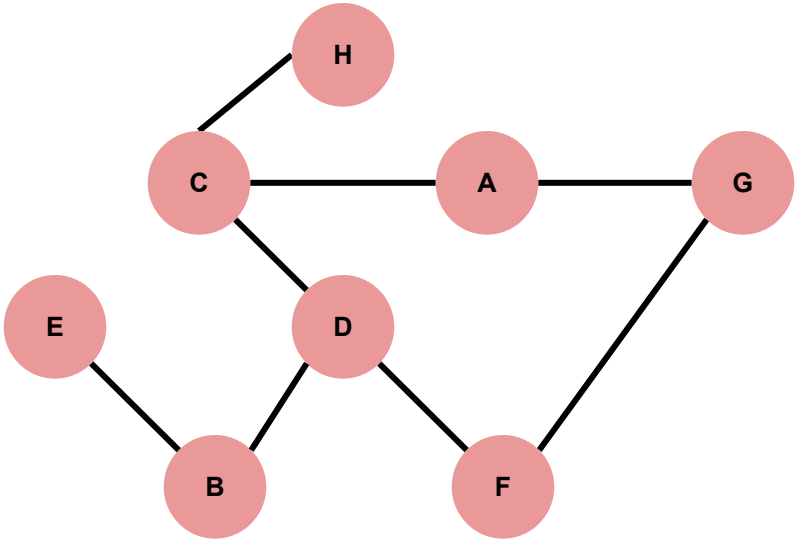
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B, E

Initialize Queue with Starting Vertex & Mark it

1. While Queue is not empty:

- ***Dequeue vertex v***

2. For every unmarked neighbor n:

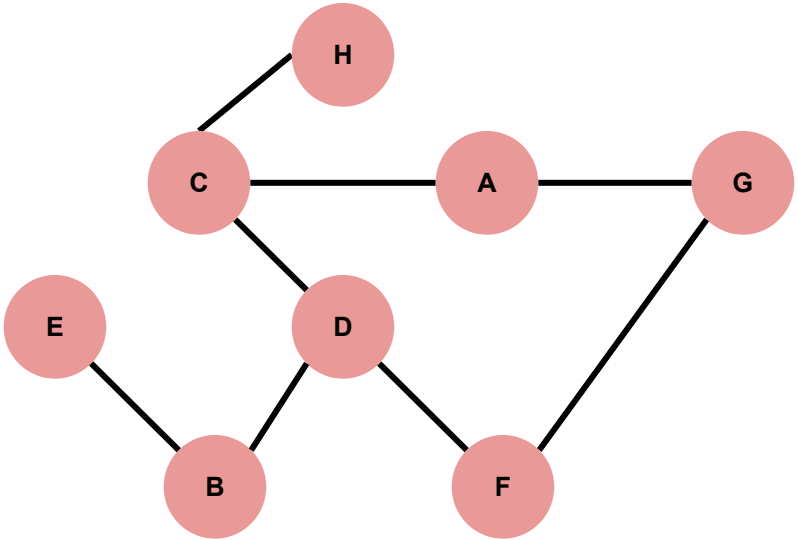
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Queue: []

BFS

*Starting from A, write the order in which vertices are visited using **BFS**.*



Order of BFS: A, C, G, D, H, F, B, E

Initialize Queue with Starting Vertex & Mark it
1. While Queue is not empty:
- ***Dequeue vertex v***

2. For every unmarked neighbor n:
- ***Mark neighbor***
- ***Enqueue n to Queue***

Vertex	marked[]
A	T
B	T
C	T
D	T
E	T
F	T
G	T
H	T

Queue: []