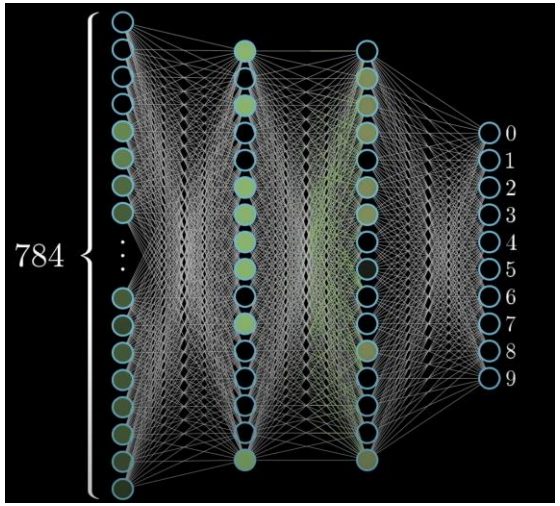
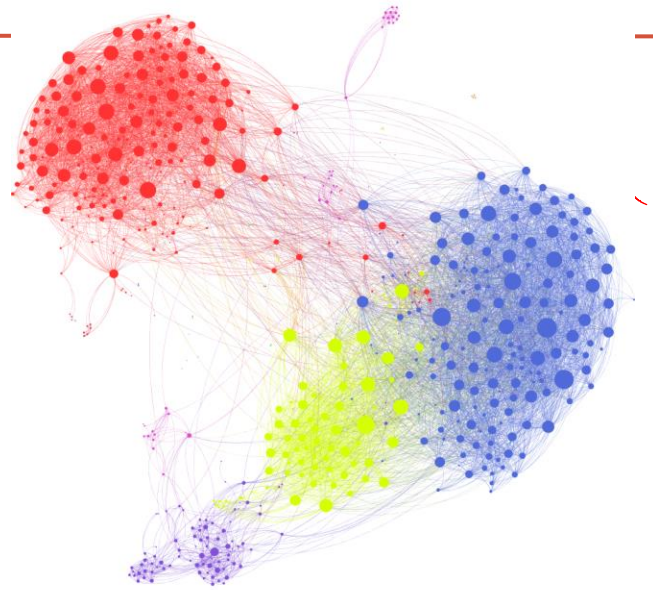
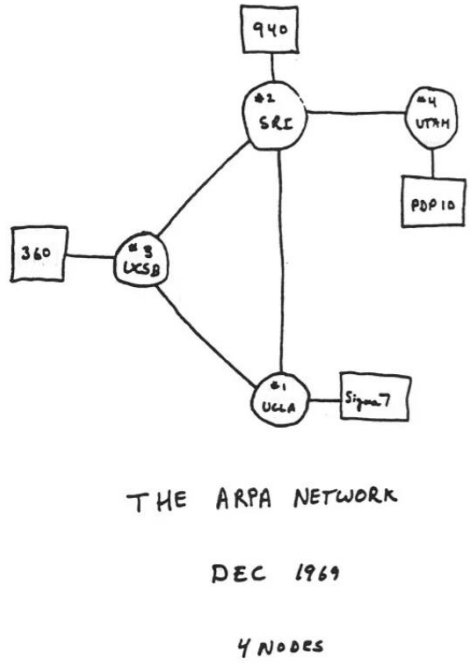
Handout: https://bit.ly/GraphSearch-BFS-DFS

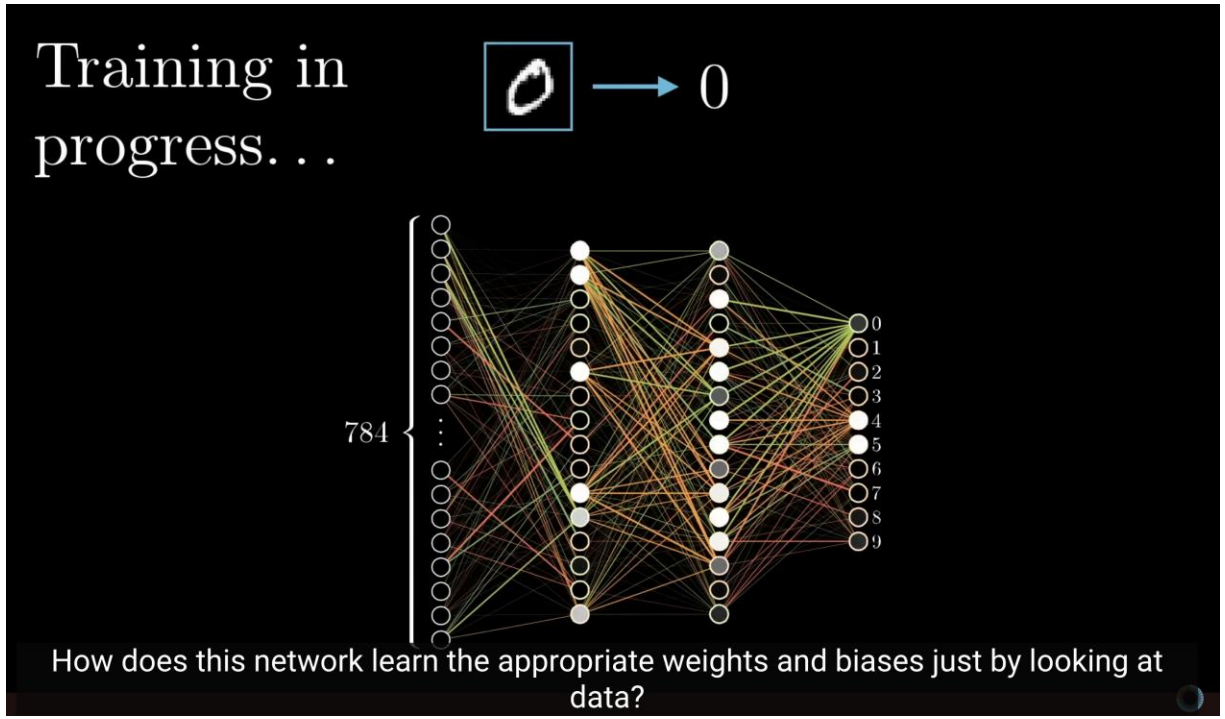
GRAPH SEARCH – BFS & DFS

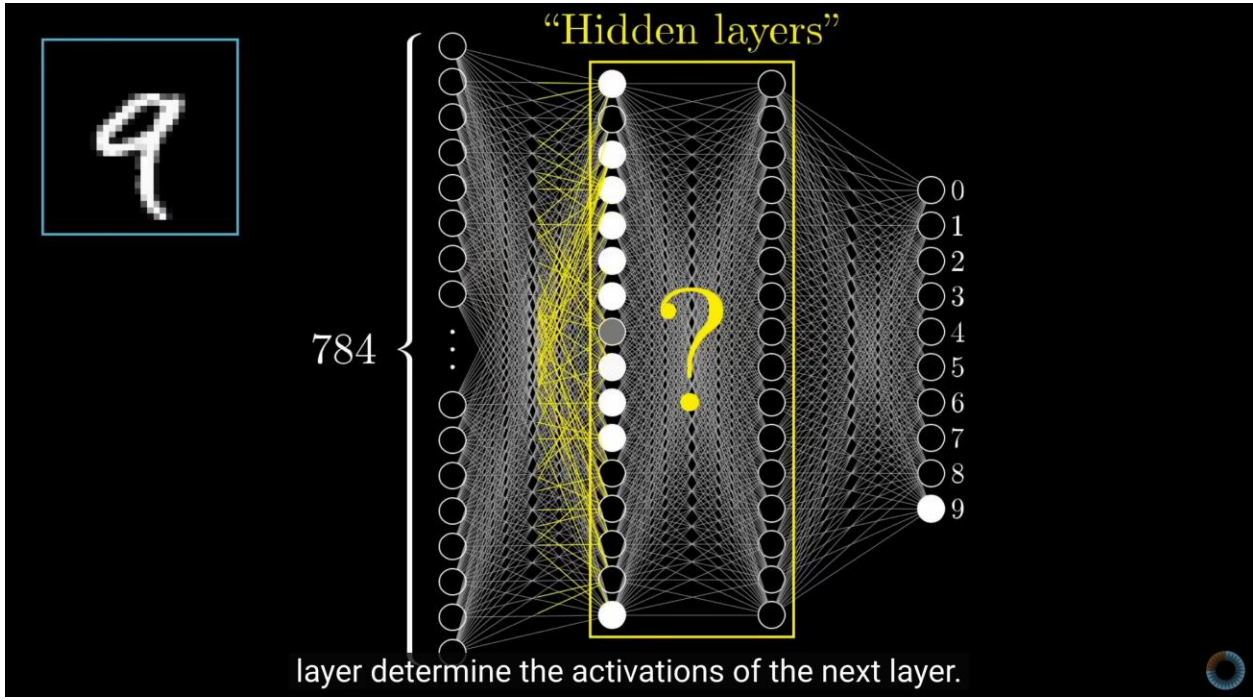






How does information flow in a Neural Network ?





Training

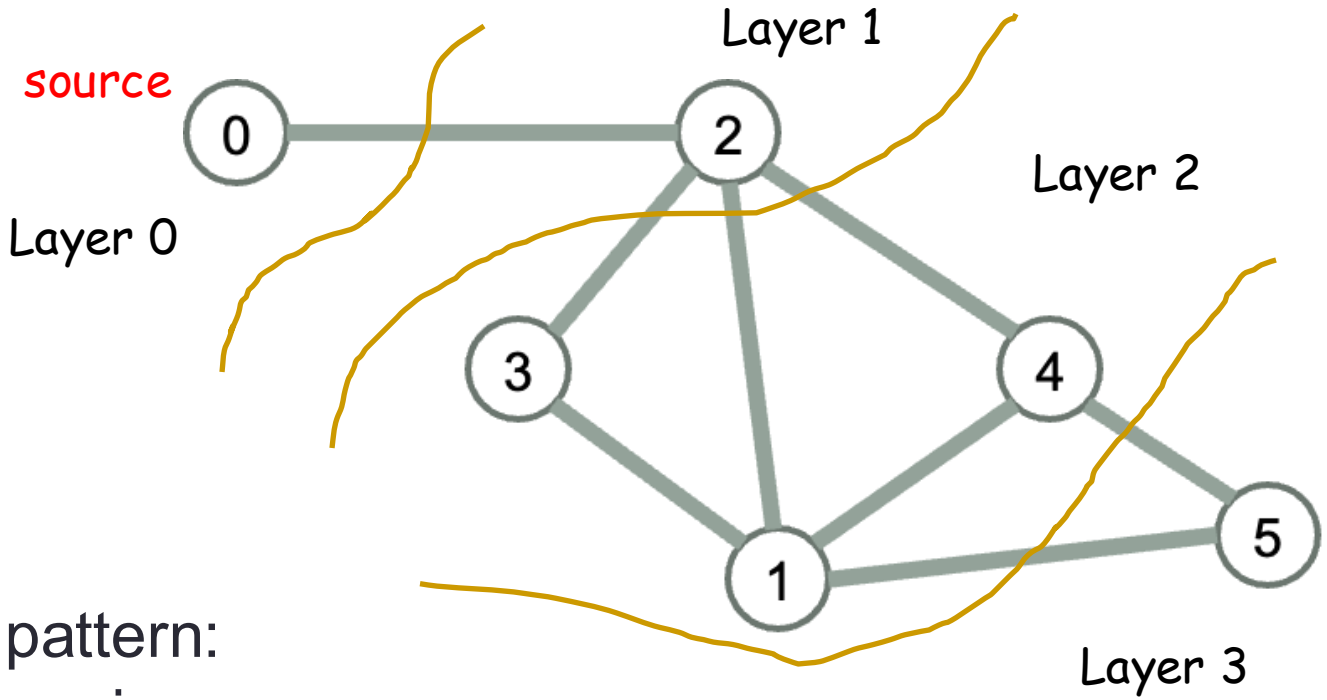
Learn network parameters (all the weights and biases)

Evaluation/Prediction

Activations in one layer determine activations in the next layer

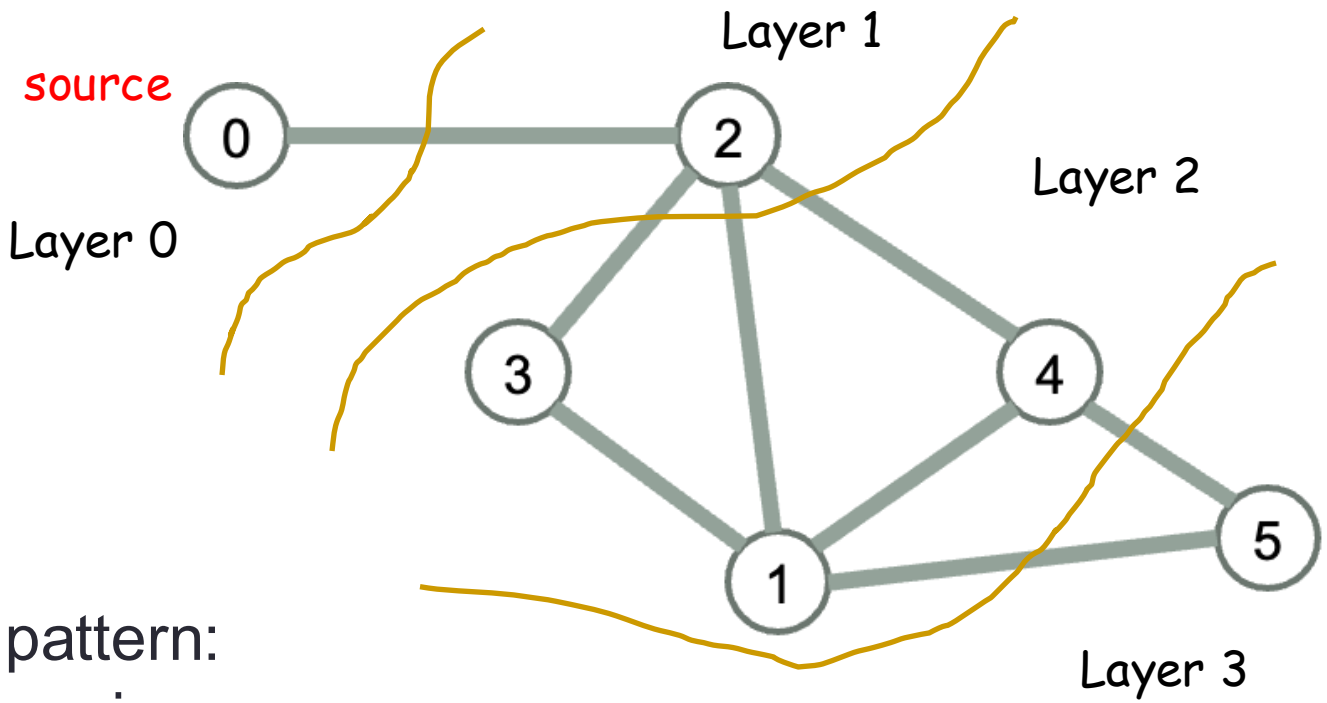
Credits: 3Blue1Brown

Breadth First Traversal: Sketch of Algorithm



Explore the graph in a wave (layered) pattern: explore all the vertices reachable from a given vertex before exploring their neighbors.

Breadth First Traversal: Sketch of Algorithm



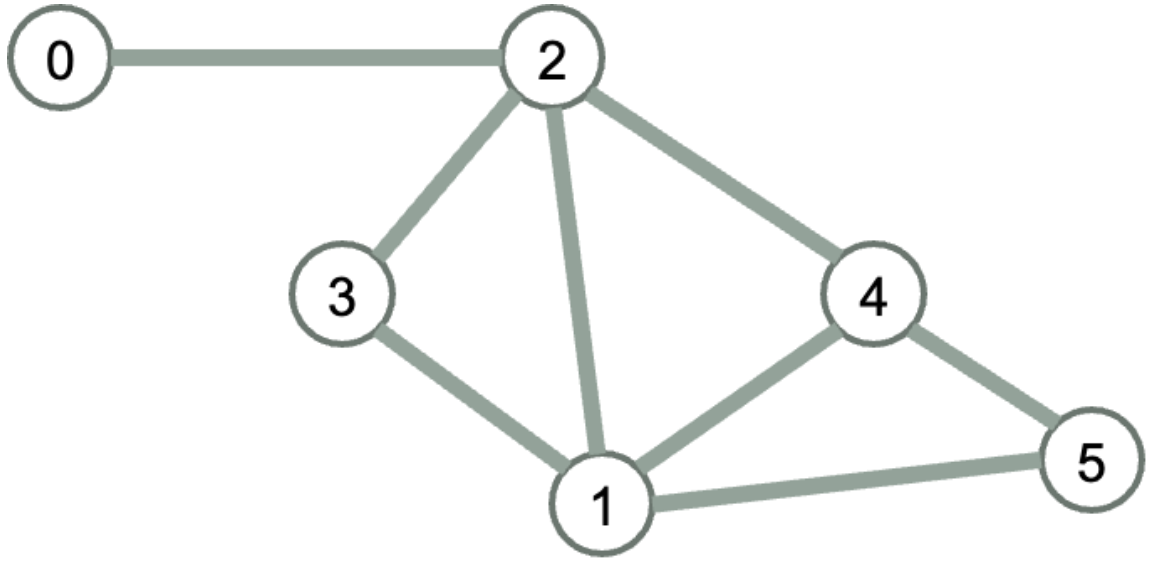
Explore the graph in a wave (layered) pattern: explore all the vertices reachable from a given vertex before exploring their neighbors.

- In general, a search algorithm would explore (or “visit”) from a source vertex
  - all the vertices reachable ,
  - never exploring out from the same vertex twice
- How does the Breadth First Search/Traversal algorithm ensure this?

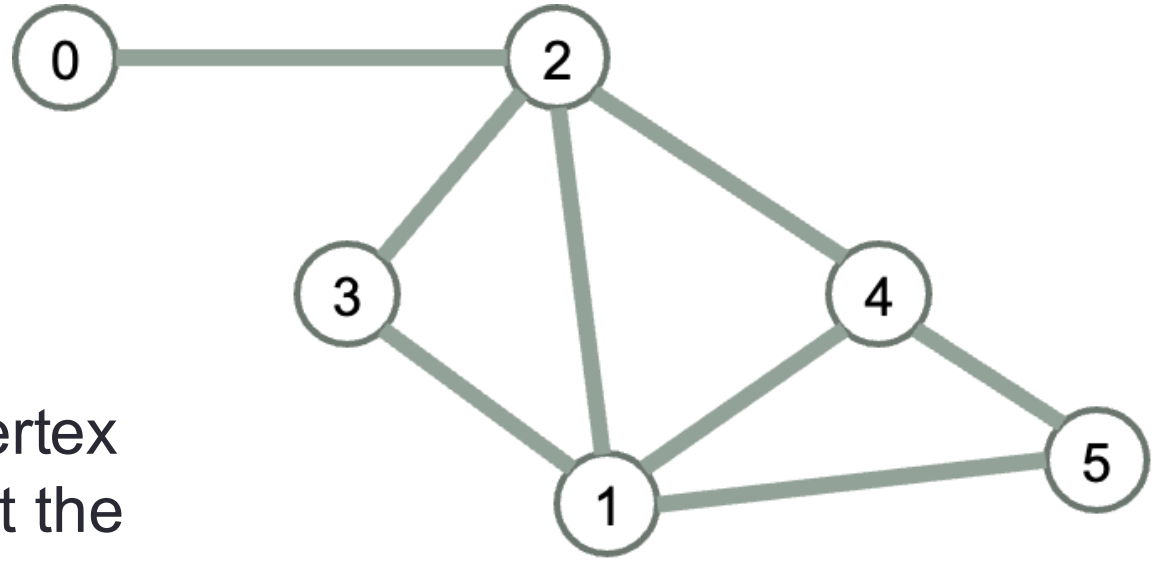
Breadth First Algorithm

Algo exploreBFS (Graph G, vertex s):

- Mark all the vertices as “not visited”
- Mark ***s*** as visited
- push ***s*** into a queue
- while the queue is not empty:
  - pop the vertex ***u*** from the front of the queue
  - for each of ***u***’s neighbor (***v***)
    - If ***v*** has not yet been visited:
      - Mark ***v*** as visited
      - Push ***v*** in the queue



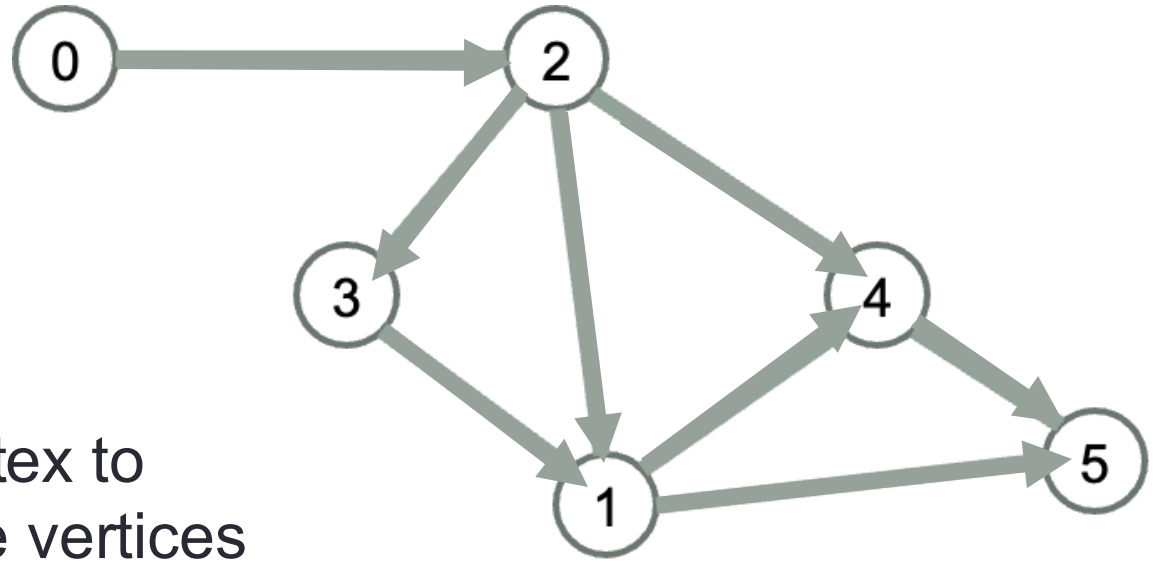
Trace BFS for the example graph



Assume BFS chooses the lower number vertex to explore first, in what order does BFS visit the nodes in this graph starting at **source vertex 3**.

A. 0, 1, 2, 3, 4, 5
B. 0, 1, 3, 2, 4, 5
C. 3, 2, 0, 1, 4, 5
D. 3, 1, 2, 0, 4, 5
E. Something else

Trace BFS (different source vertex)

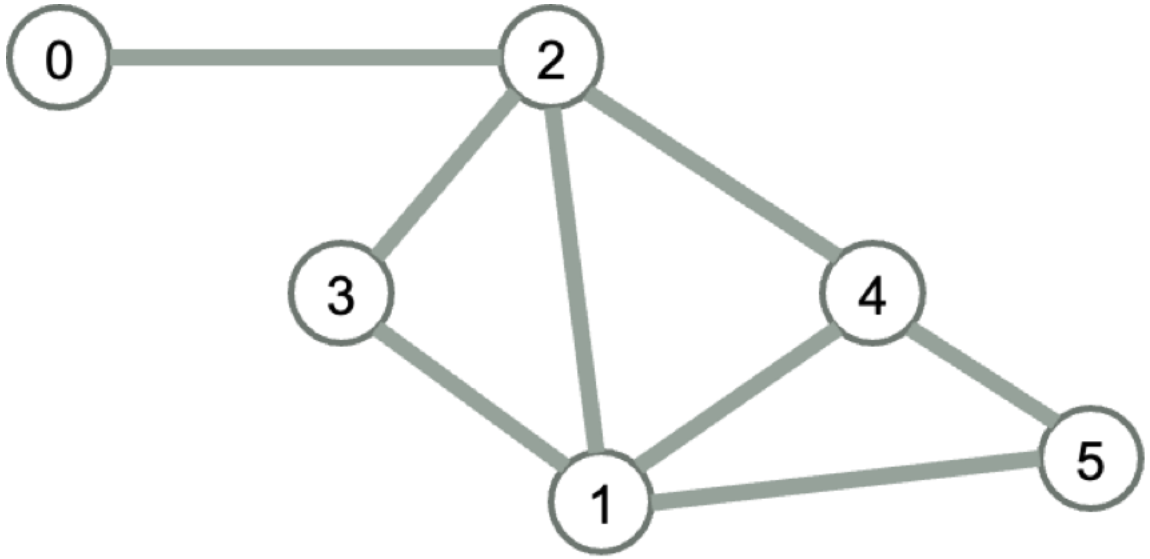


What if edges were directed as shown?
Assume BFS chooses the lower number vertex to explore first, in what order does BFS visit the vertices in this graph starting at **source vertex 3**.

A. 0, 1, 2, 3, 4, 5
B. 0, 1, 3, 2, 4, 5
C. 3, 2, 0, 1, 4, 5
D. 3, 1, 2, 0, 4, 5
E. Something else

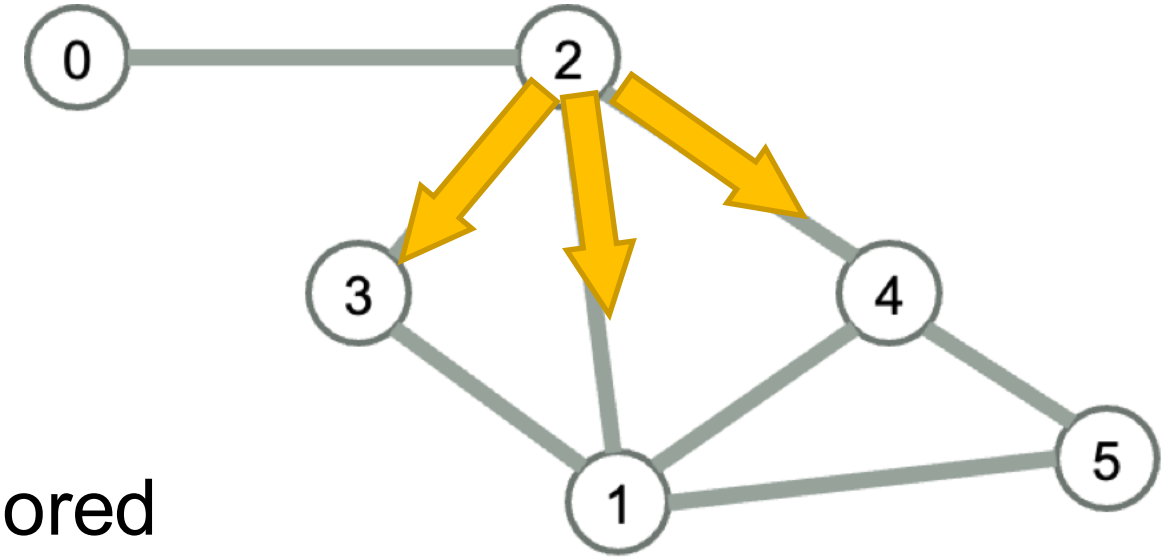
Graph search: general approach

```
Keep track of all areas discovered

While there is an unexplored path, follow path
```



Systematize the Search with DFS

Depth-First Search explores a graph by following one branch as far as it can go before backtracking. It uses a stack (explicit or via recursion) to remember where to return.



Need to keep track of:

– Which vertices discovered

– Which edges have yet to be explored

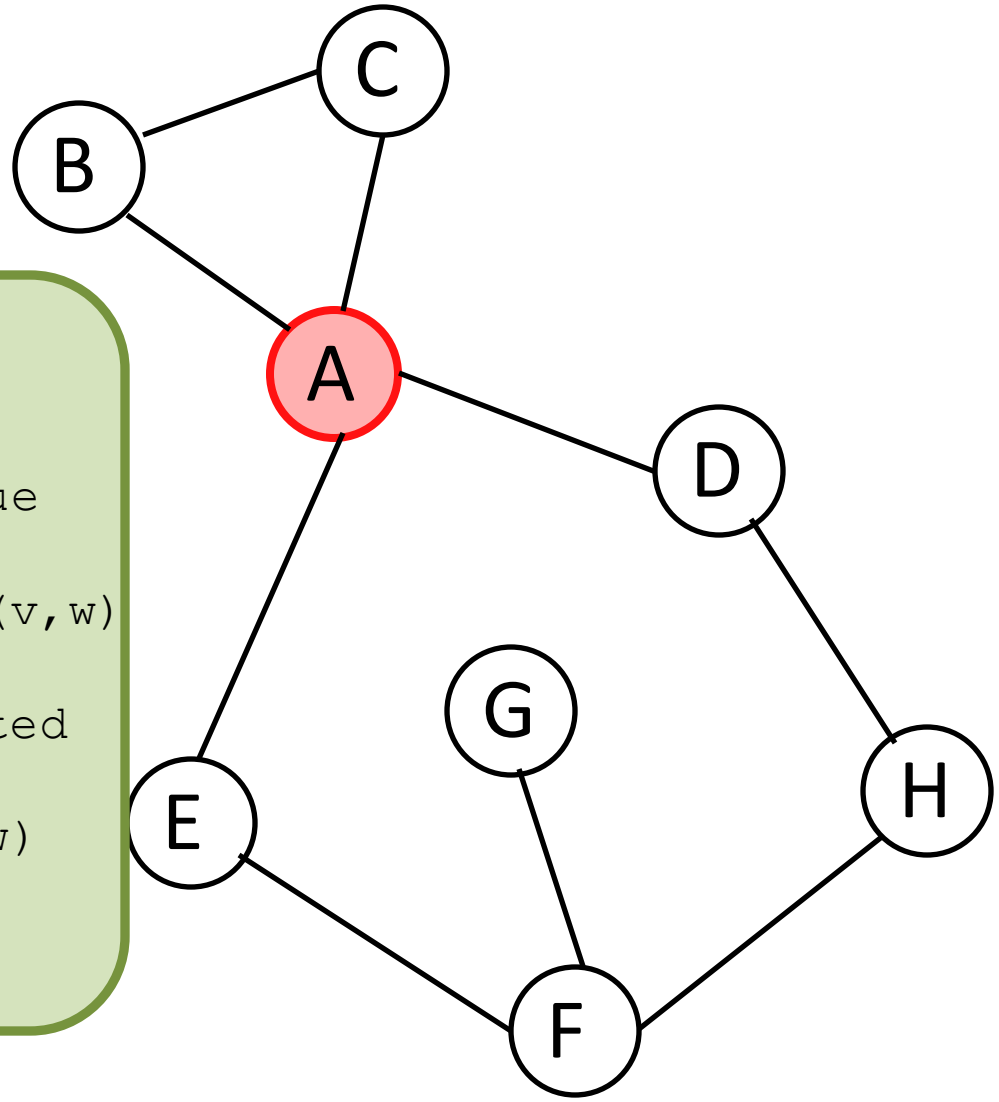
Explore – Depth First

```
exploreDFS(v)
  v.visited ← true
  For each edge (v,w)
    If not w.visited
      exploreDFS(w)
```

https://visualgo.net/en/dfsbfs

Explore (Depth First): Example

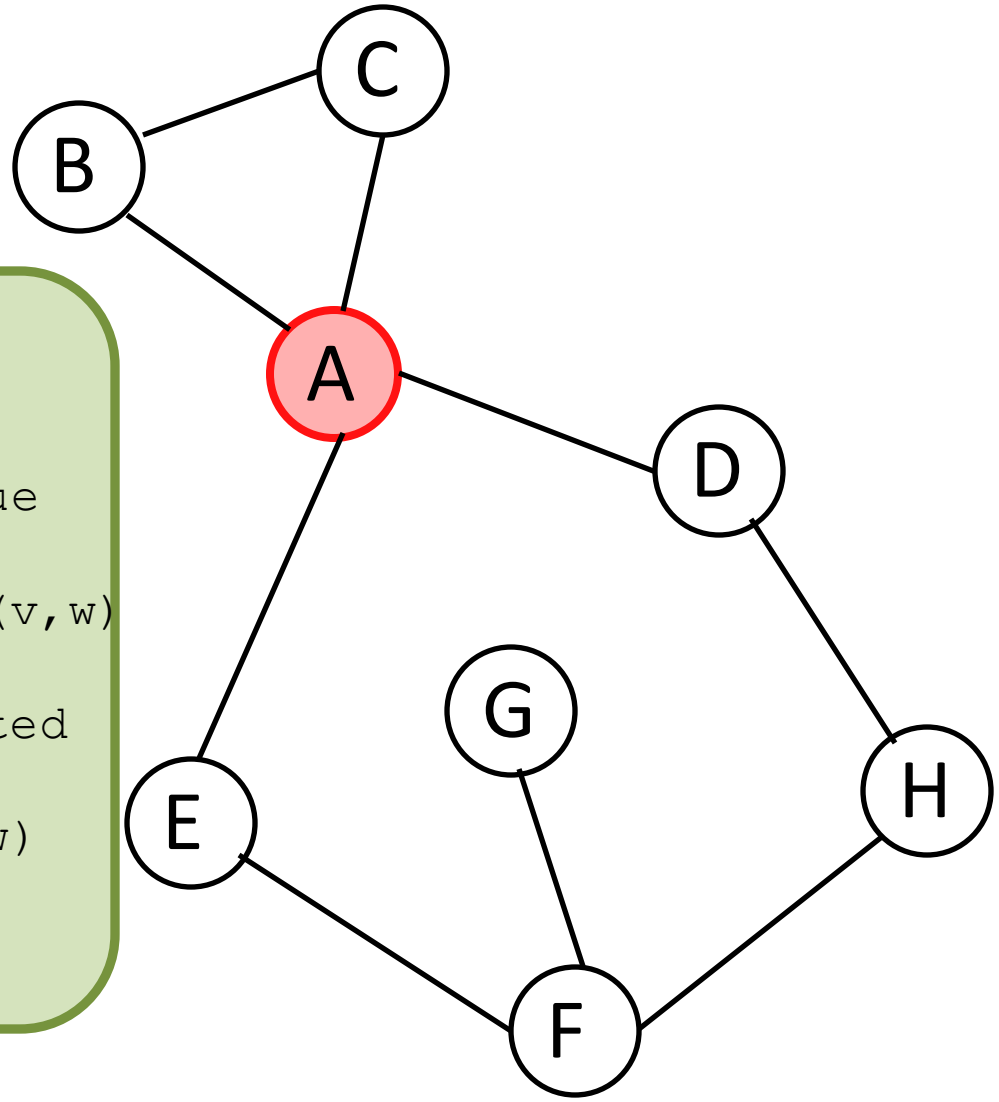
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```



explore(A)

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```



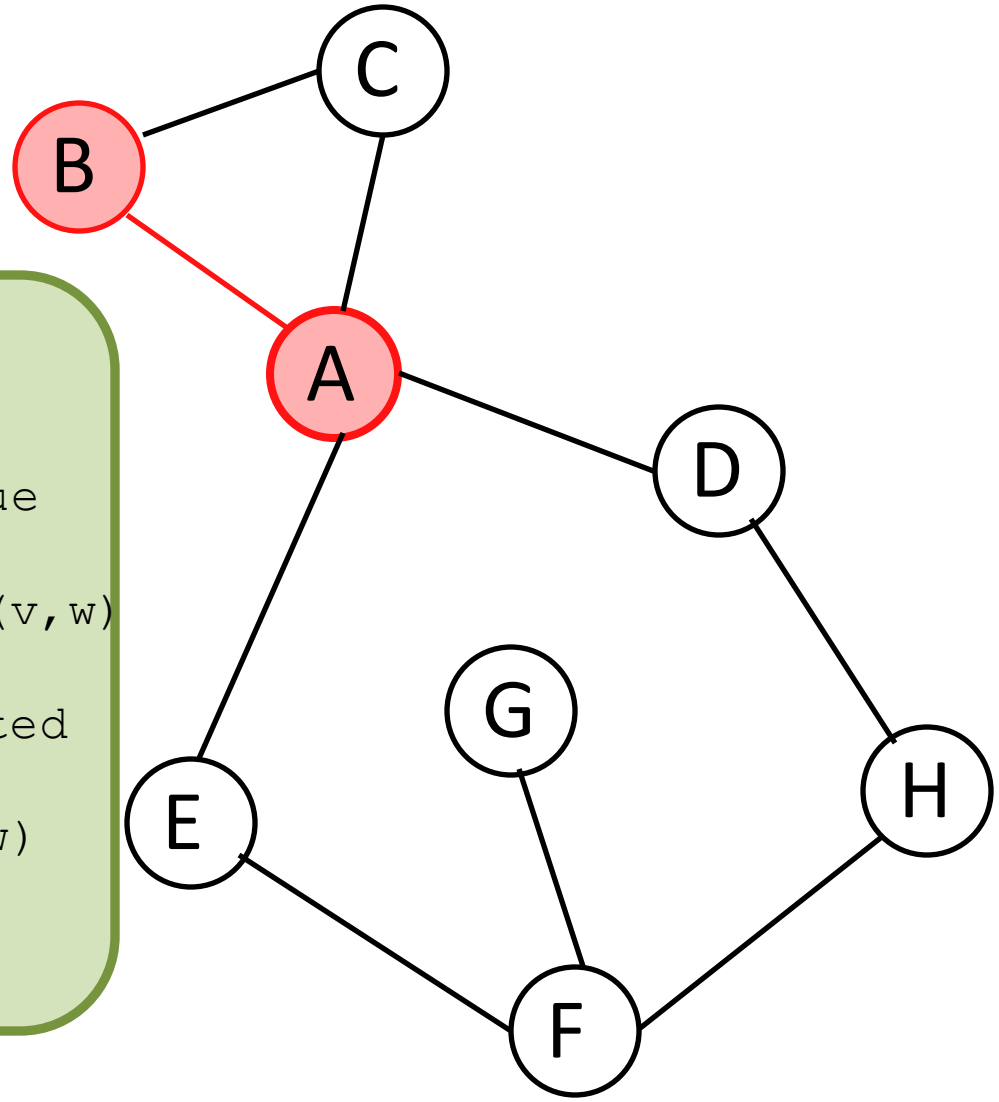
explore(A)
  explore(B)

explore(C)
explore(D)

explore(E)

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```



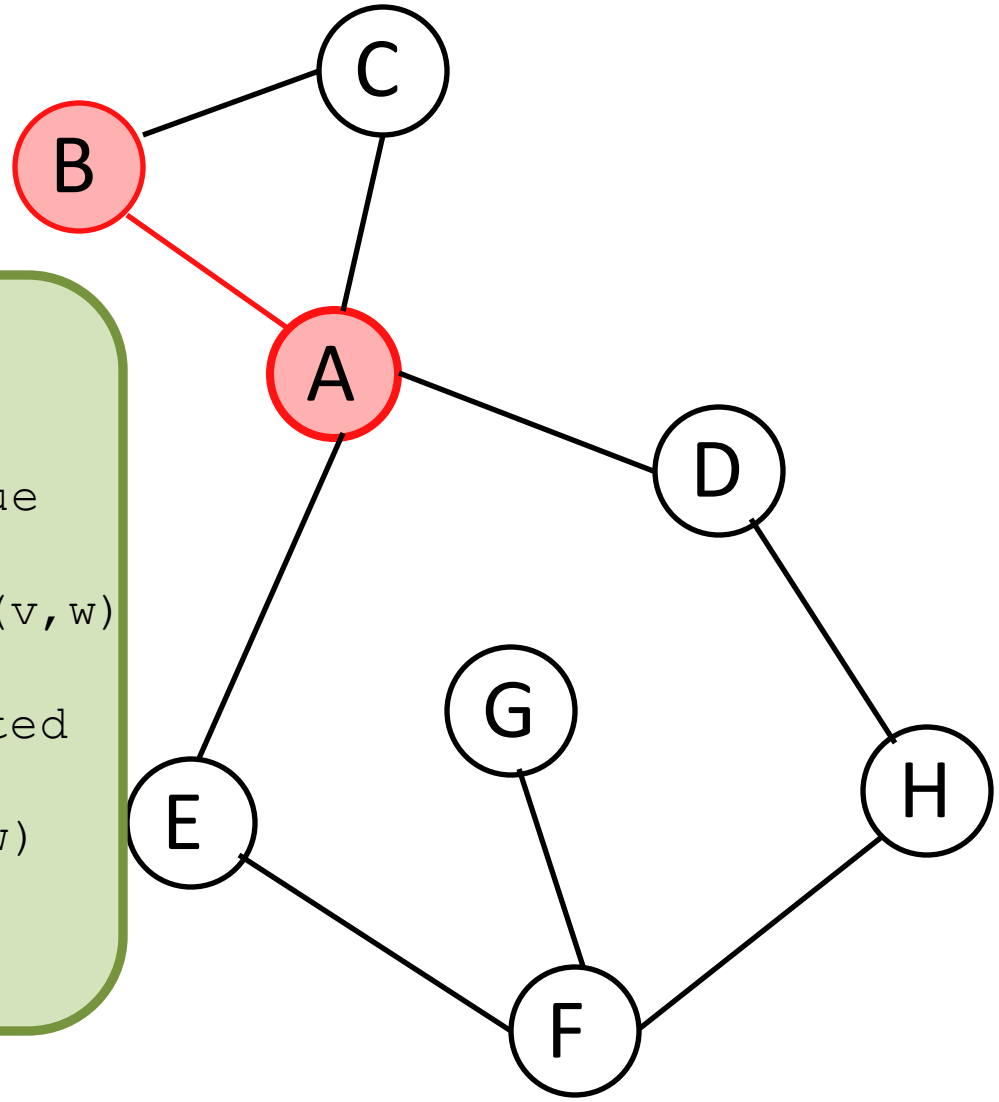
explore(A)
explore(B)

explore(C)
explore(D)

explore(E)

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```

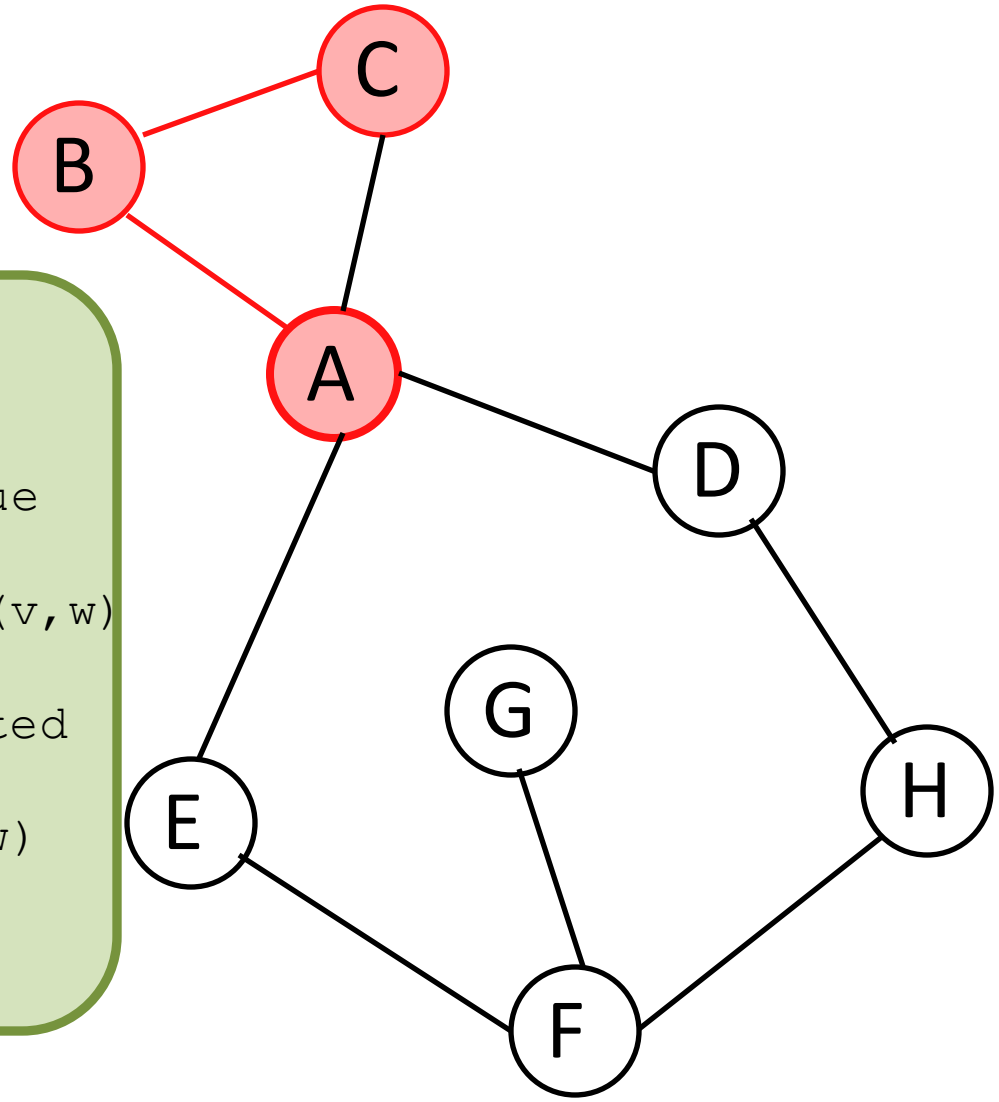



```
explore(A)
  explore(B)
    explore(A)
    explore(C)

  explore(C)
  explore(D)

  explore(E)
```

Based on slides by Professor Daniel Kane, UCSD

Explore (Depth First): Example

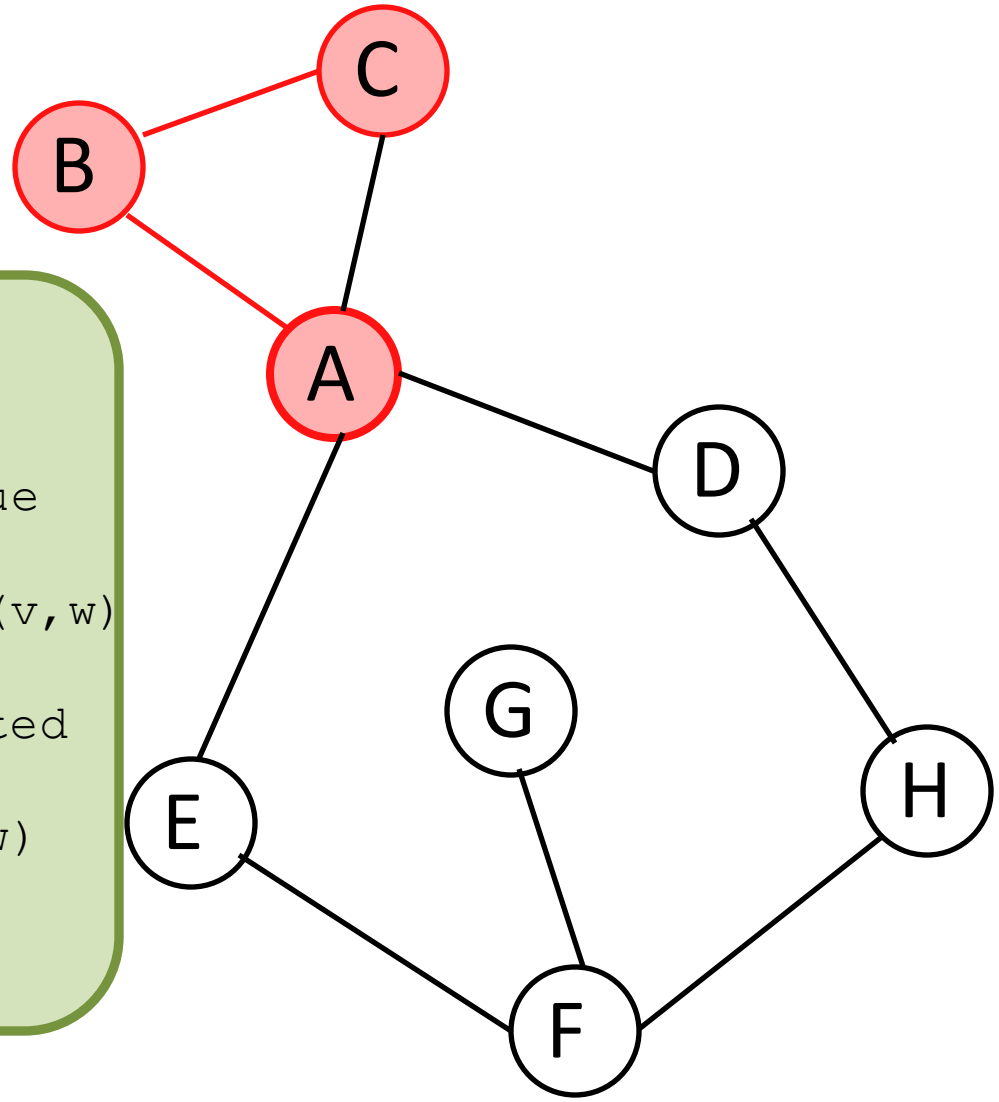
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)

explore(C)
explore(D)

explore(E)
```

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```



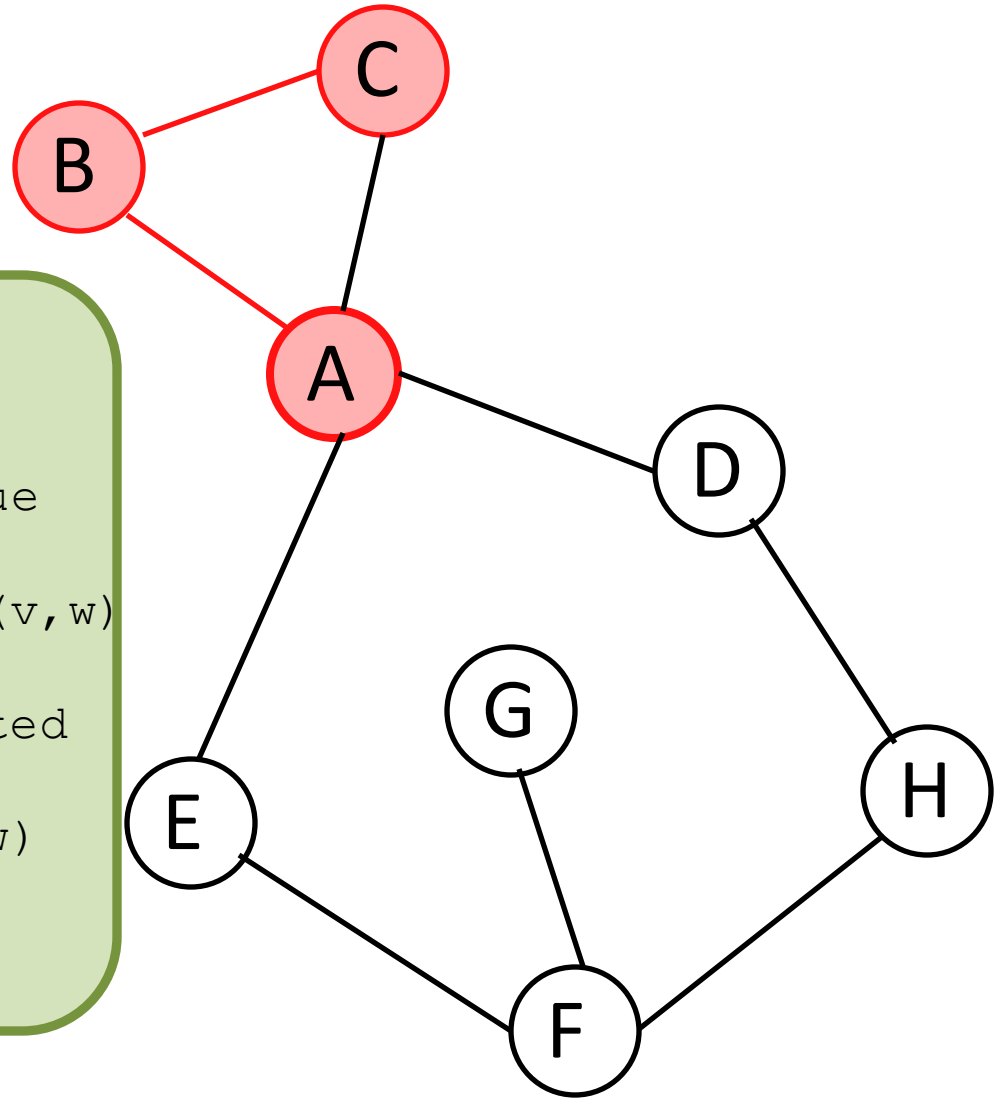

```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
```

```
explore(E)
```

Based on slides by Professor Daniel Kane, UCSD

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```

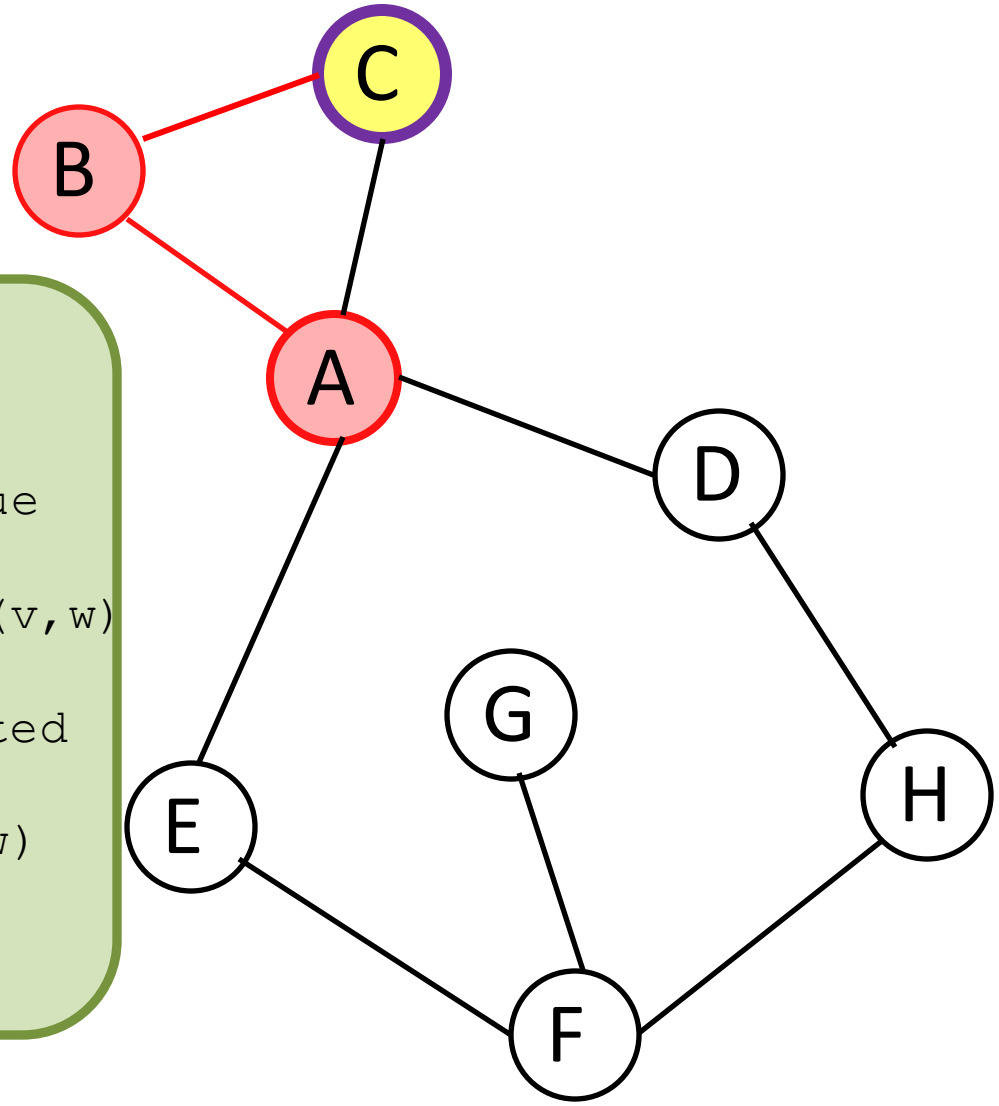



```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
```

```
explore(E)
```

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
```

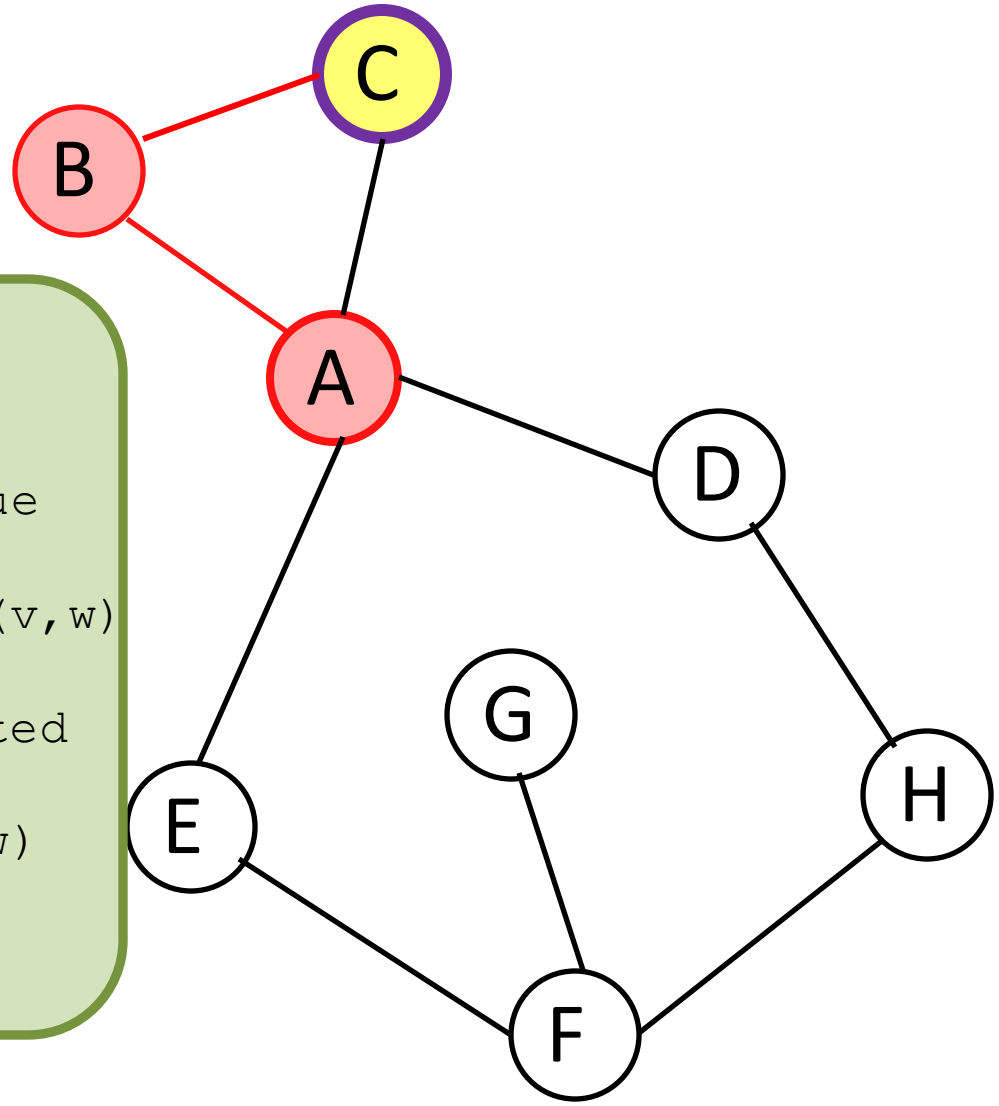
Dead end!

What should happen next?

```
explore(E)
```

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```



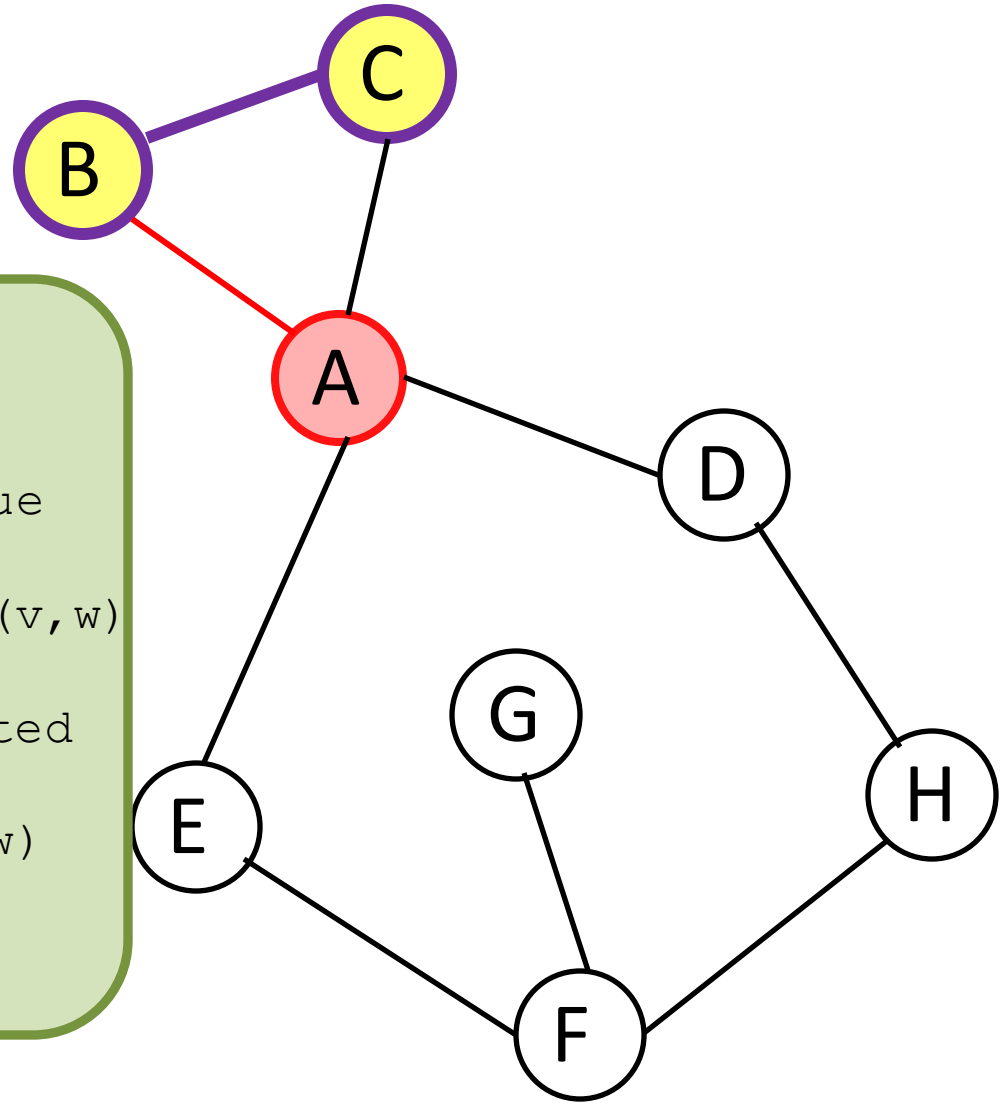

```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
```

Dead end!
What should happen next?

Go back to vertex B
Explore any unexplored neighbors of B

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
```

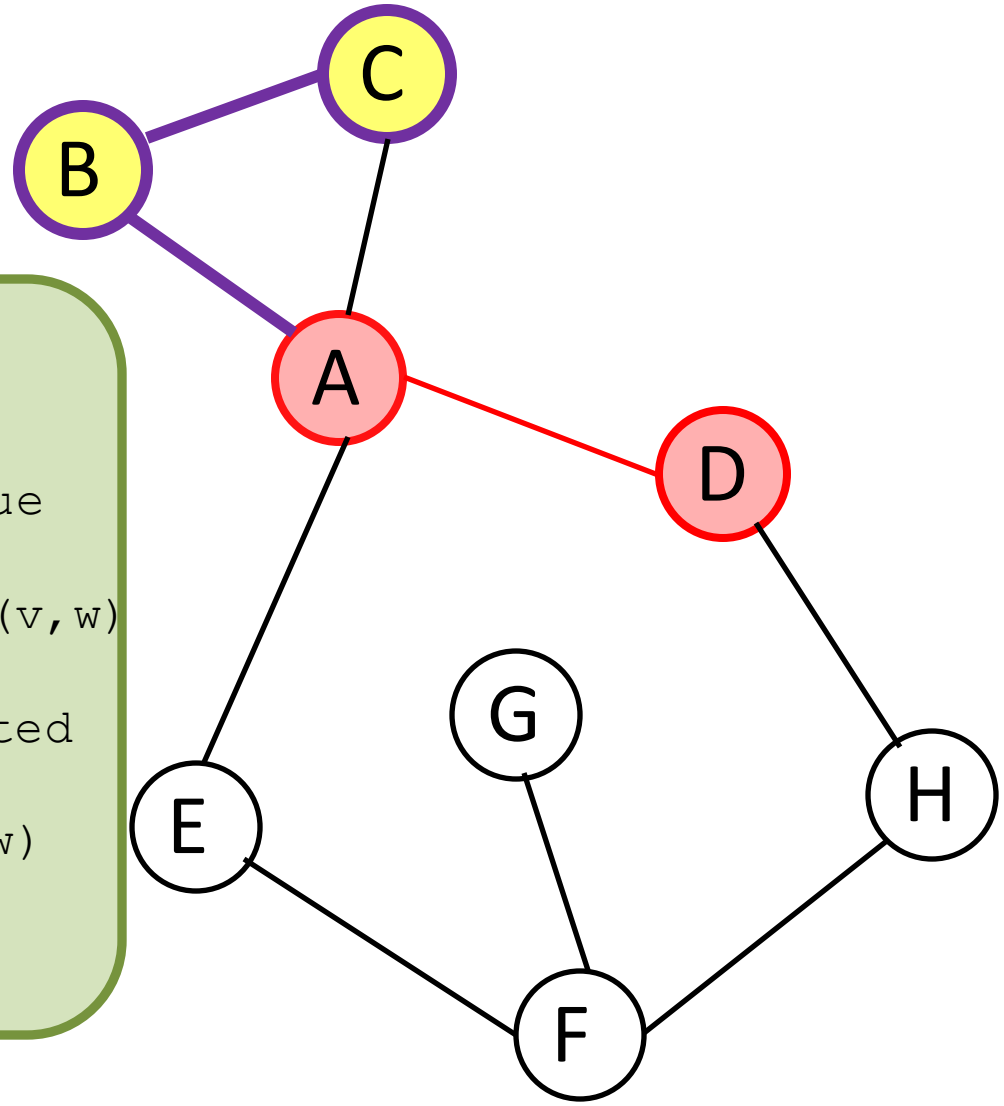
Backtracking!

Go back to A
Explore unexplored neighbors of A

```
  explore(E)
```

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```

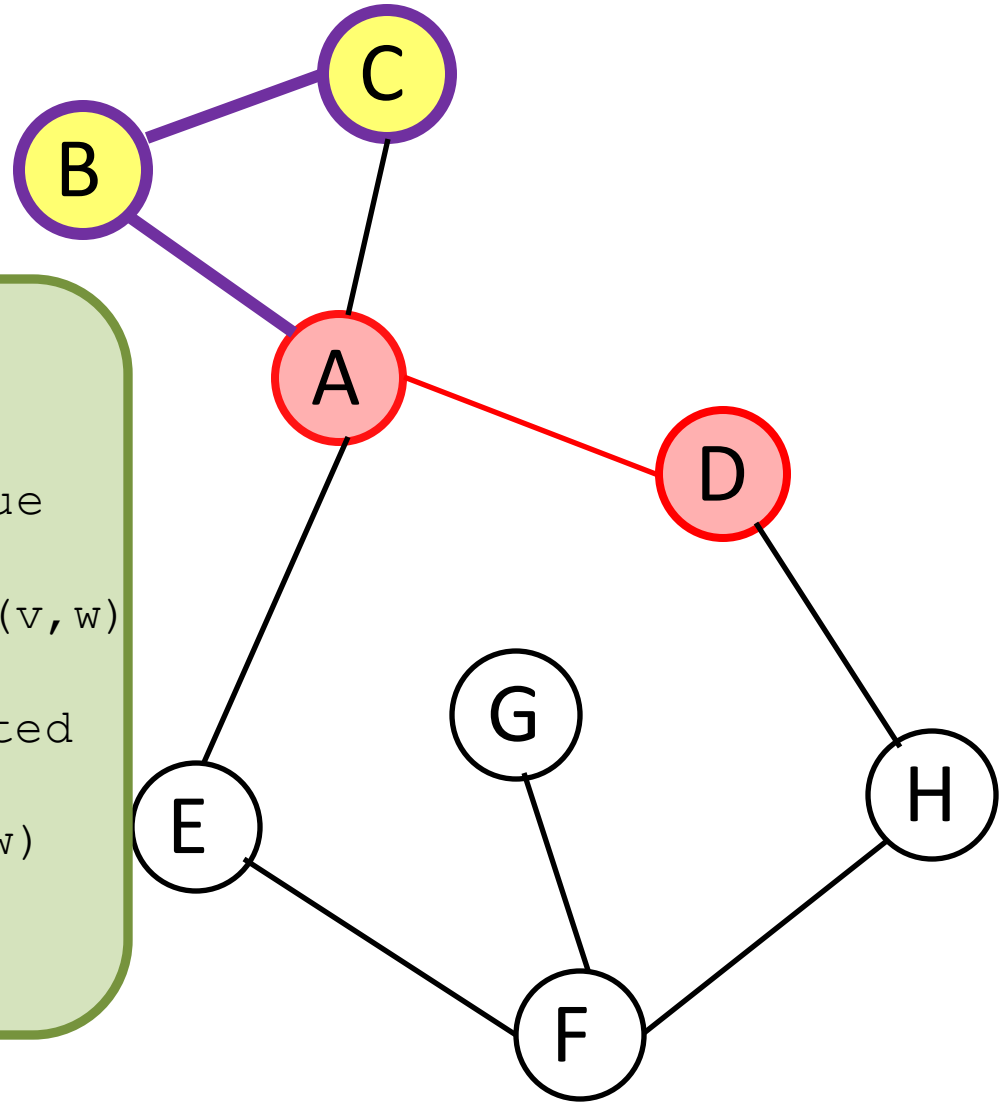



```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
```

```
explore(E)
```

Explore (Depth First): Example

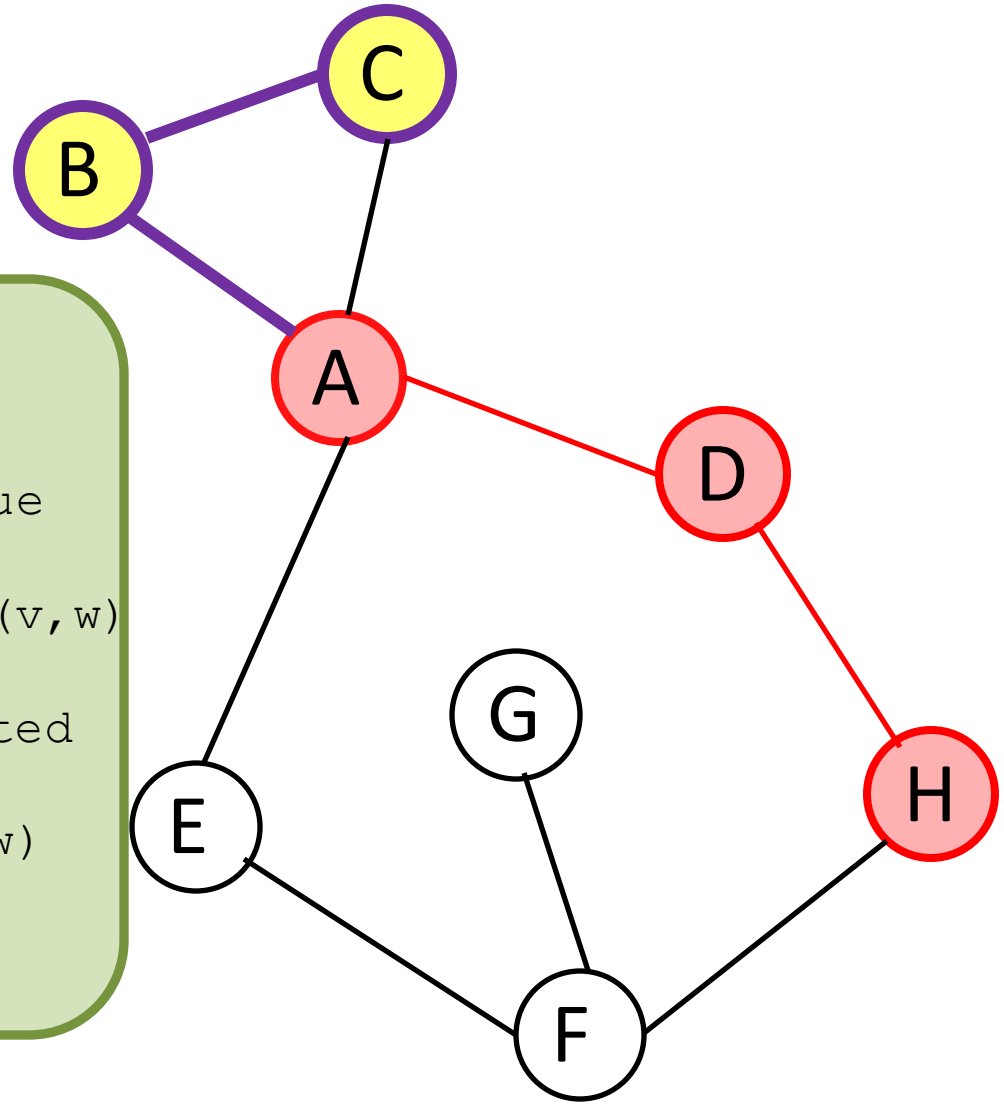
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
```

```
explore(E)
```

Explore (Depth First): Example



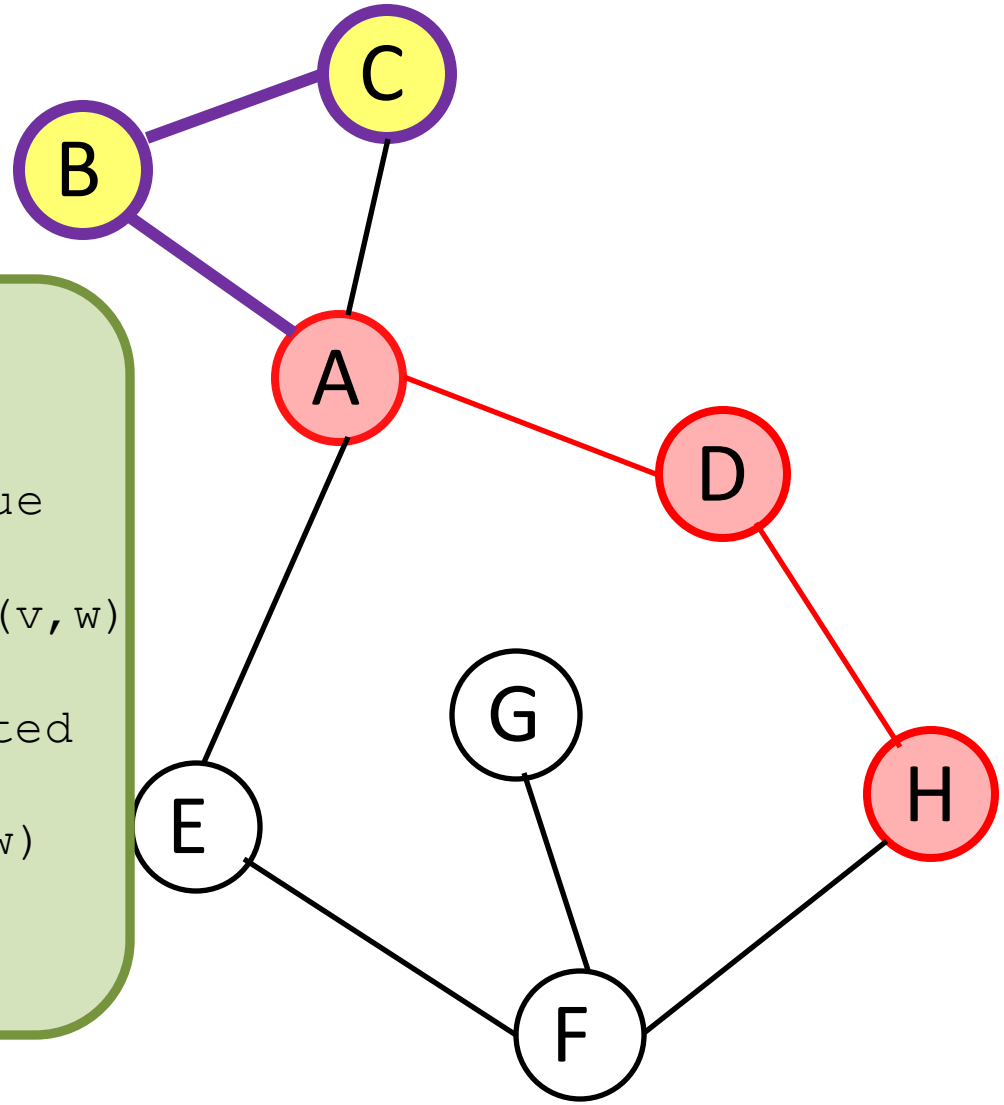

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```

```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
```

```
explore(E)
```

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```

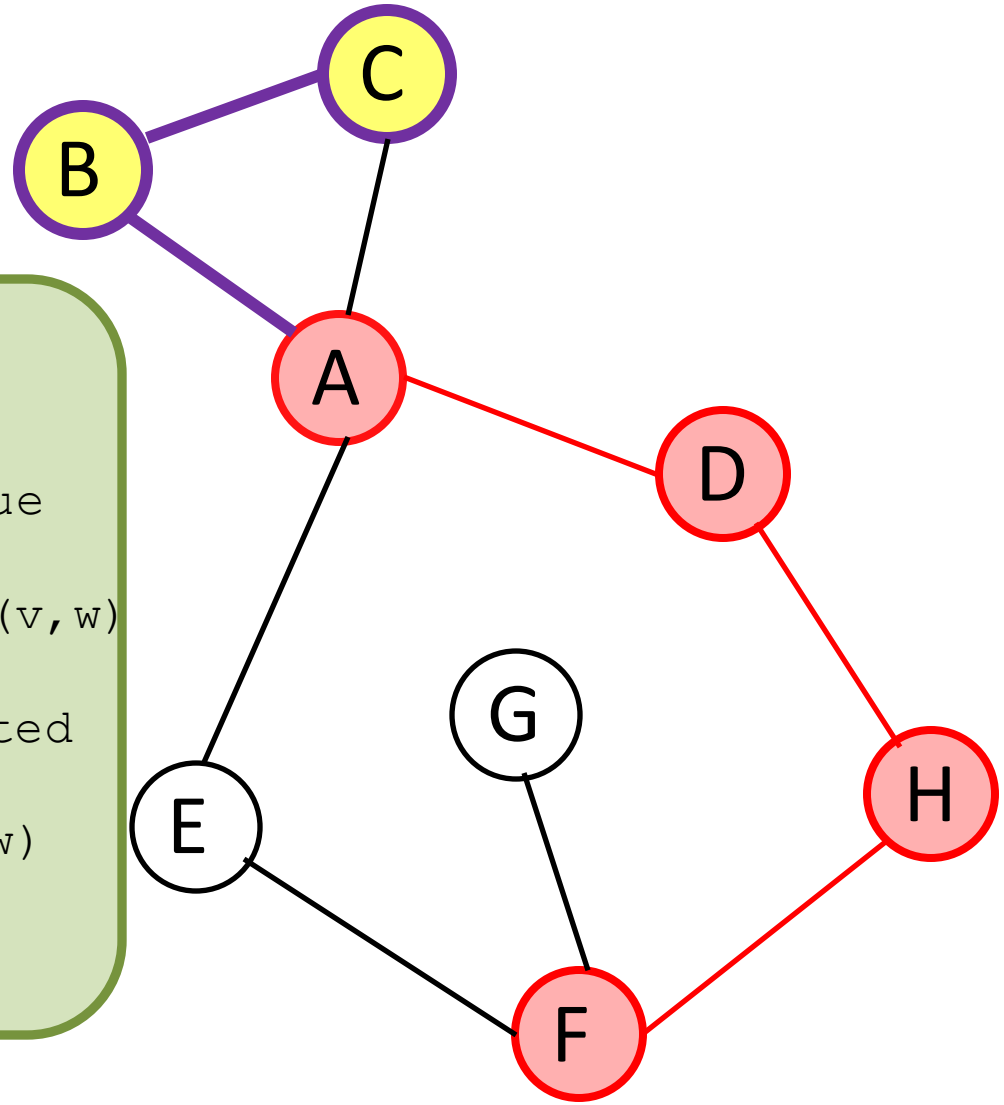



```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
```

```
explore(E)
```

Explore (Depth First): Example

```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```

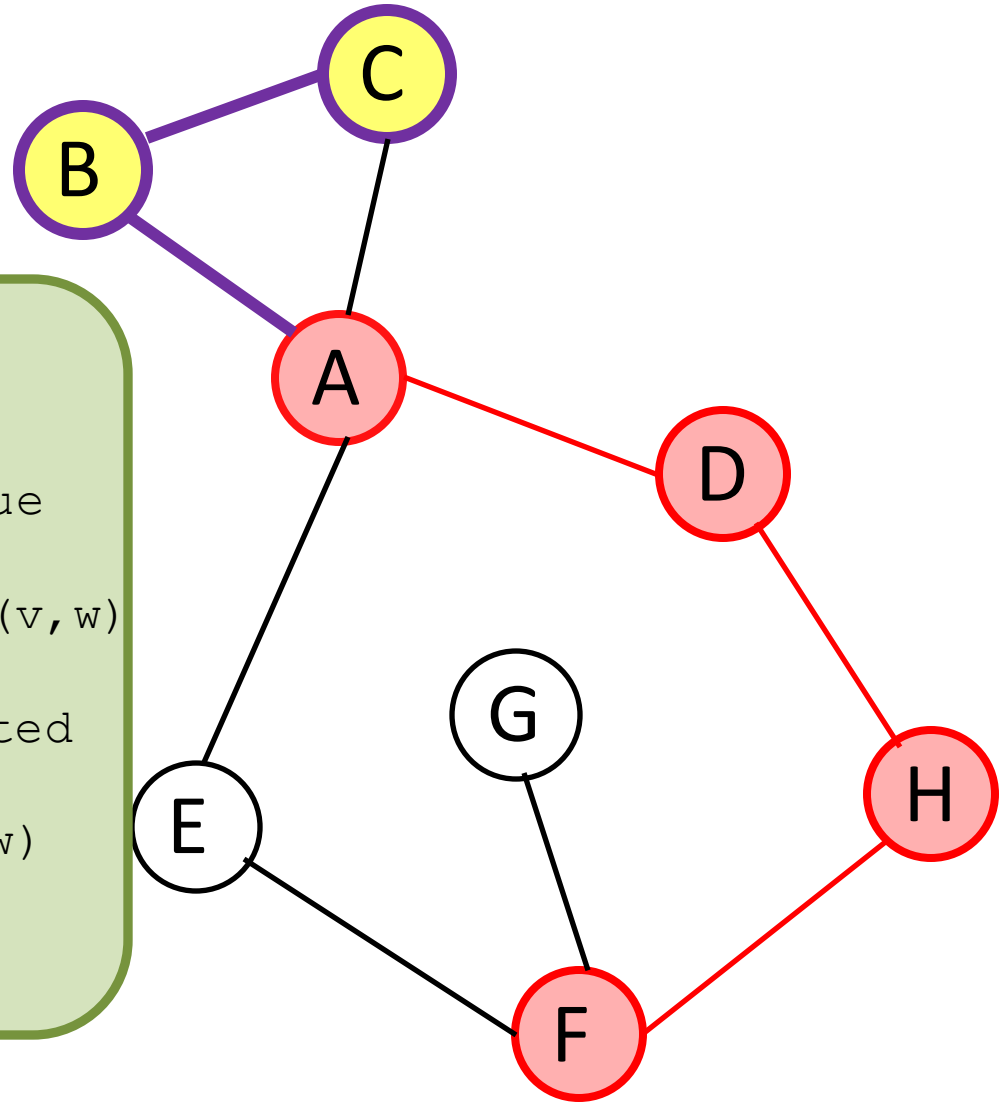



```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
```

explore(E)

Explore (Depth First): Example

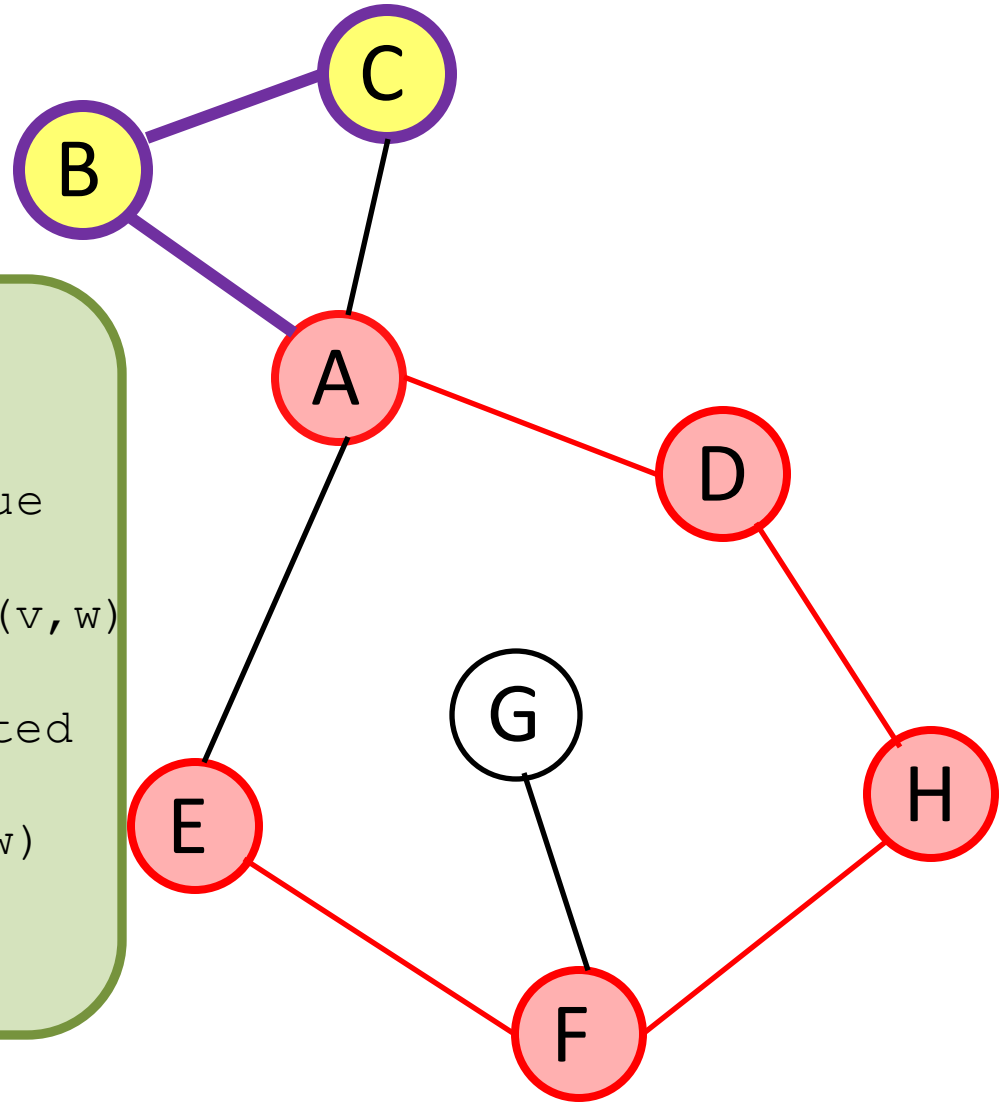
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)


        explore(G)

        explore(H)
explore(E)
```

Explore (Depth First): Example

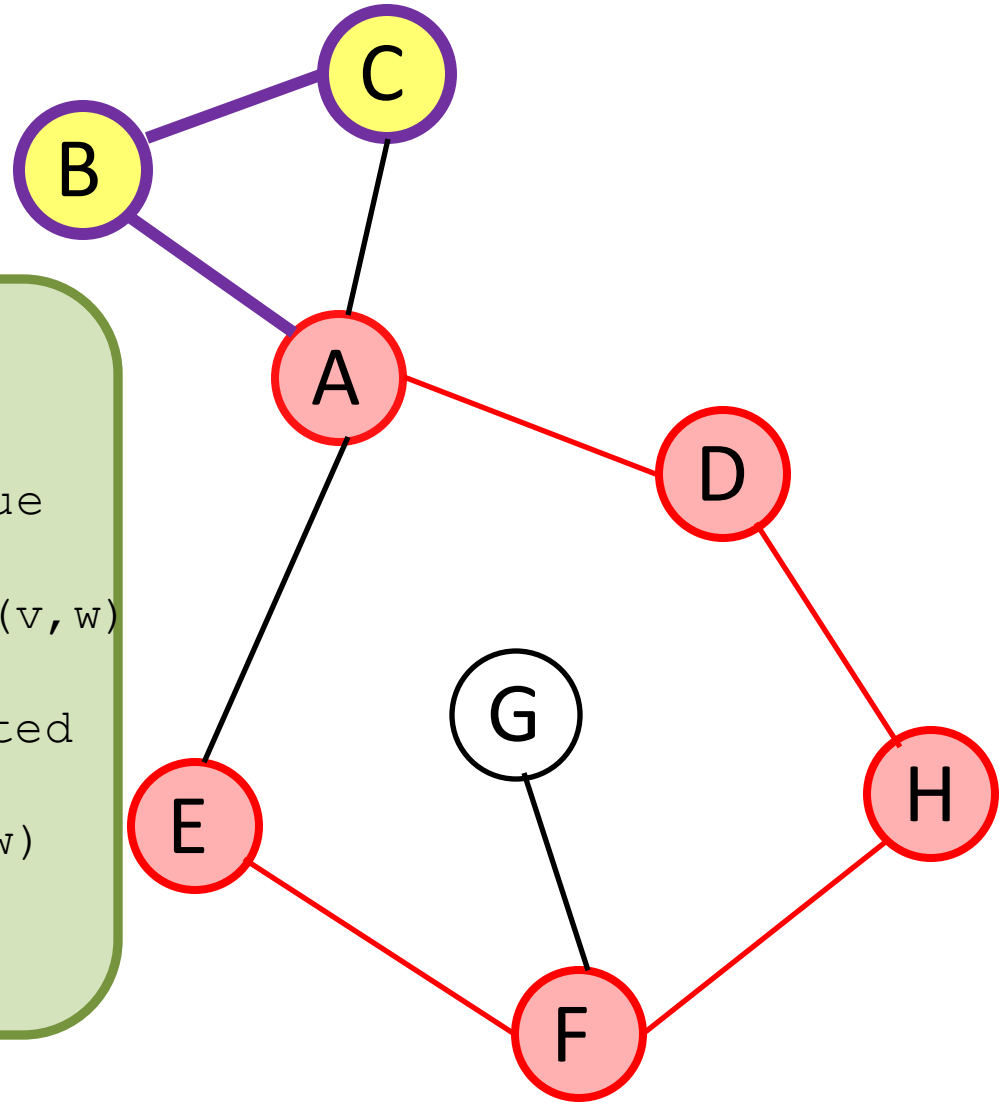
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)


          explore(G)


          explore(H)
explore(E)
```

Explore (Depth First): Example

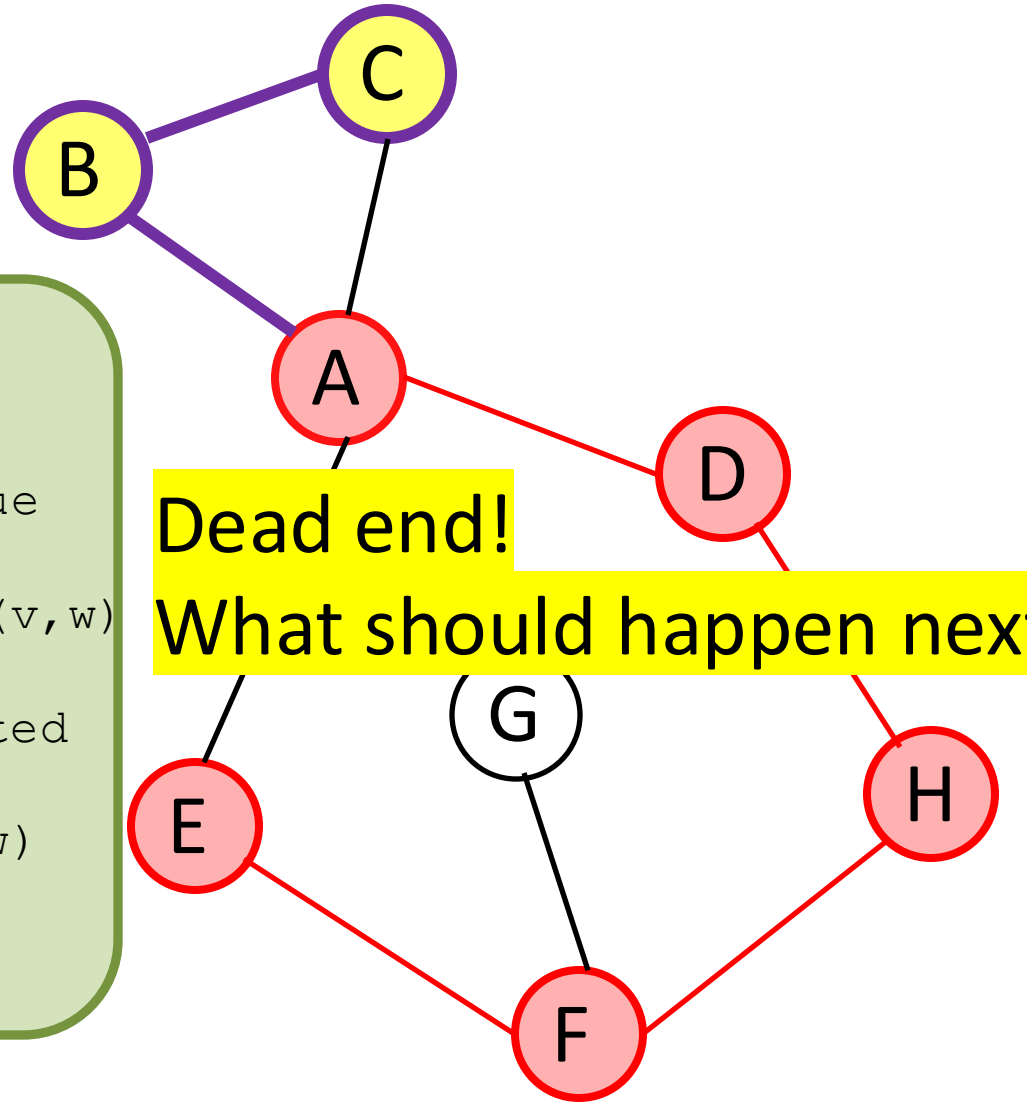
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)

        explore(H)
explore(E)
```

Explore (Depth First): Example

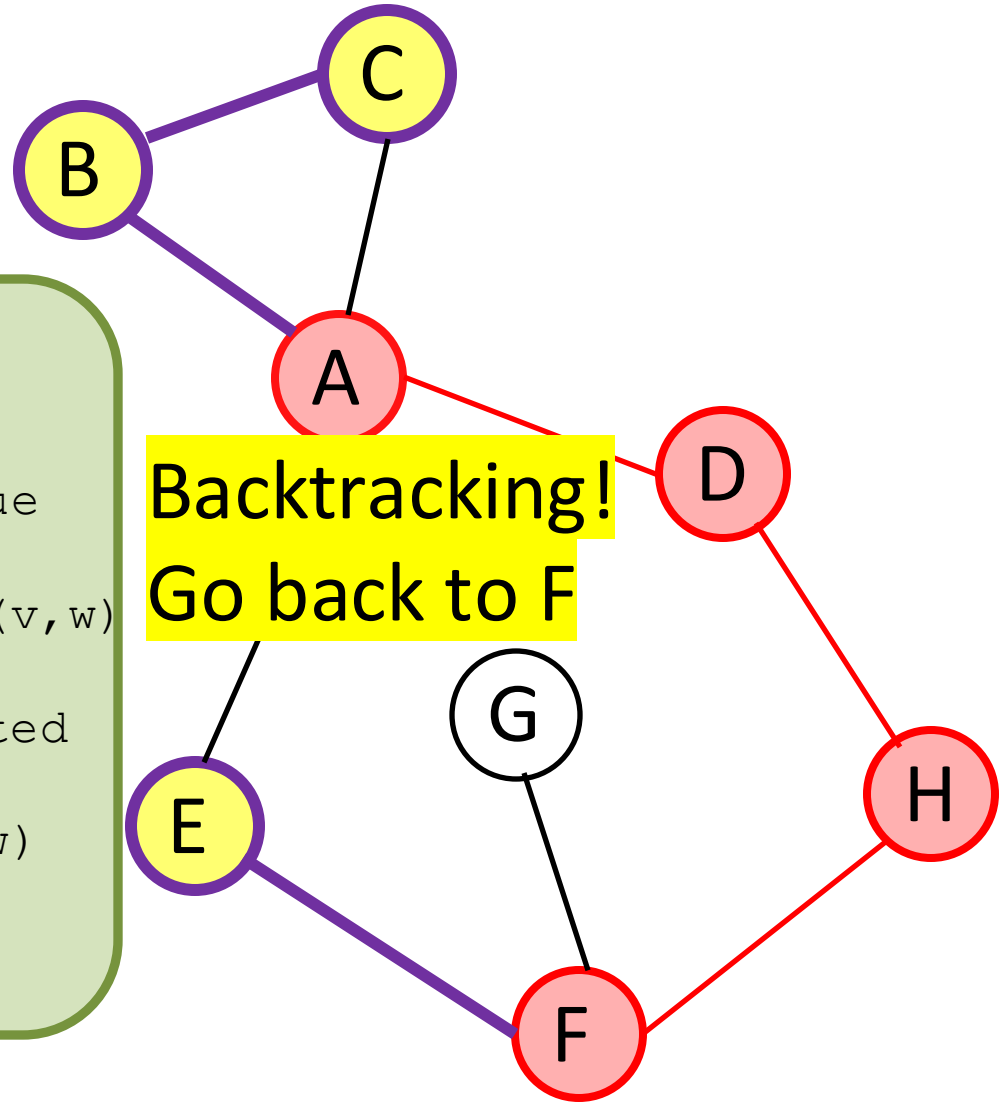
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)

        explore(H)
explore(E)
```

Explore (Depth First): Example

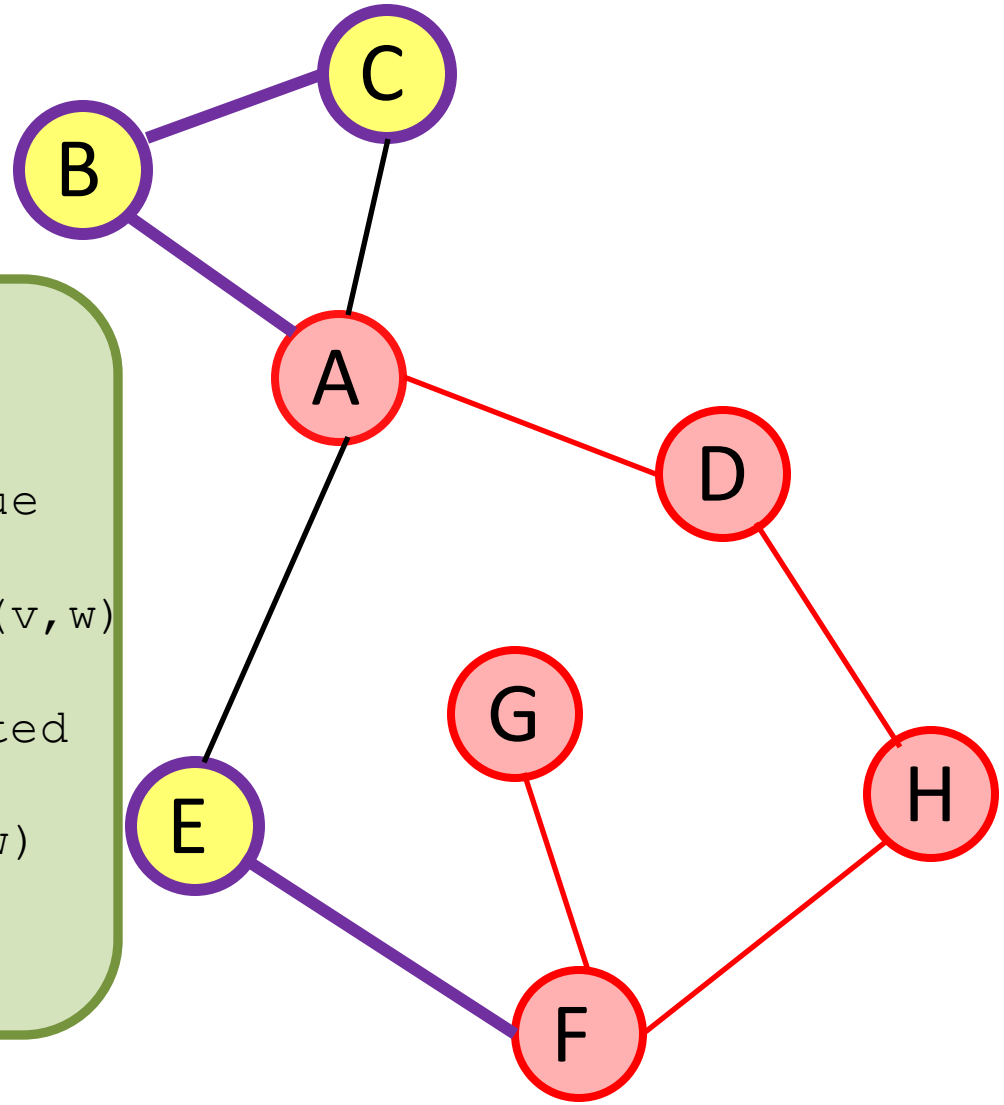
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)

        explore(H)
  explore(E)
```

Explore (Depth First): Example

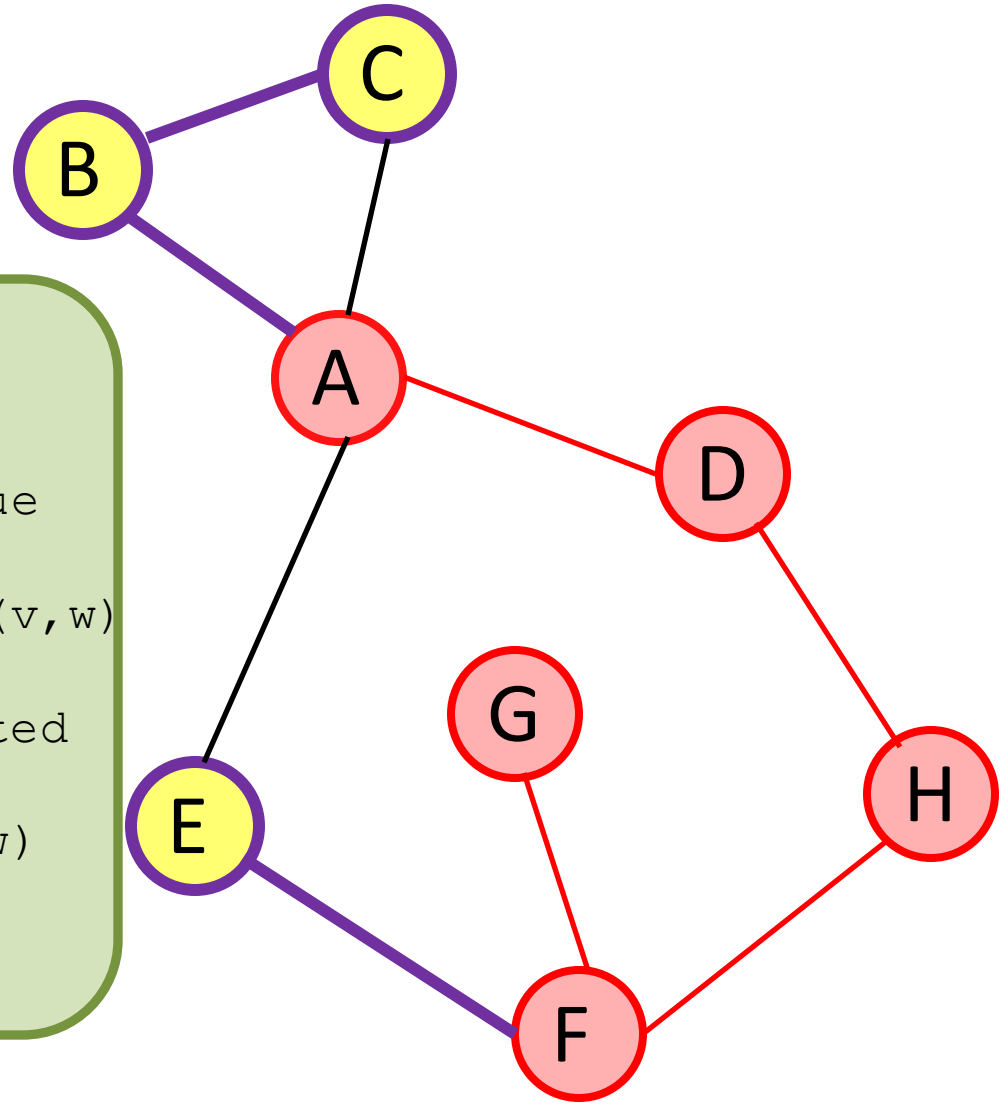
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)

        explore(H)
explore(E)
```

Explore (Depth First): Example

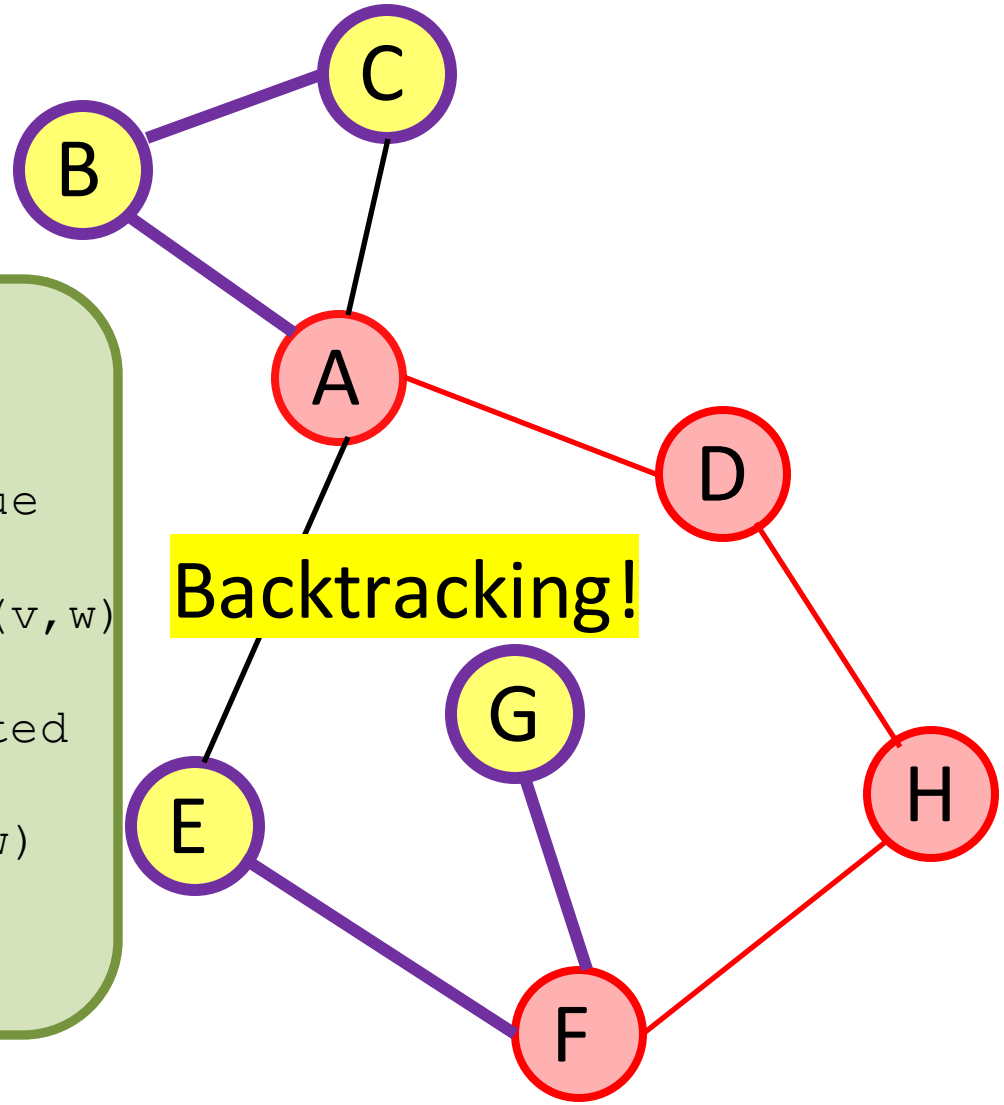
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)
          explore(F)
          explore(H)
explore(E)
```

Explore (Depth First): Example

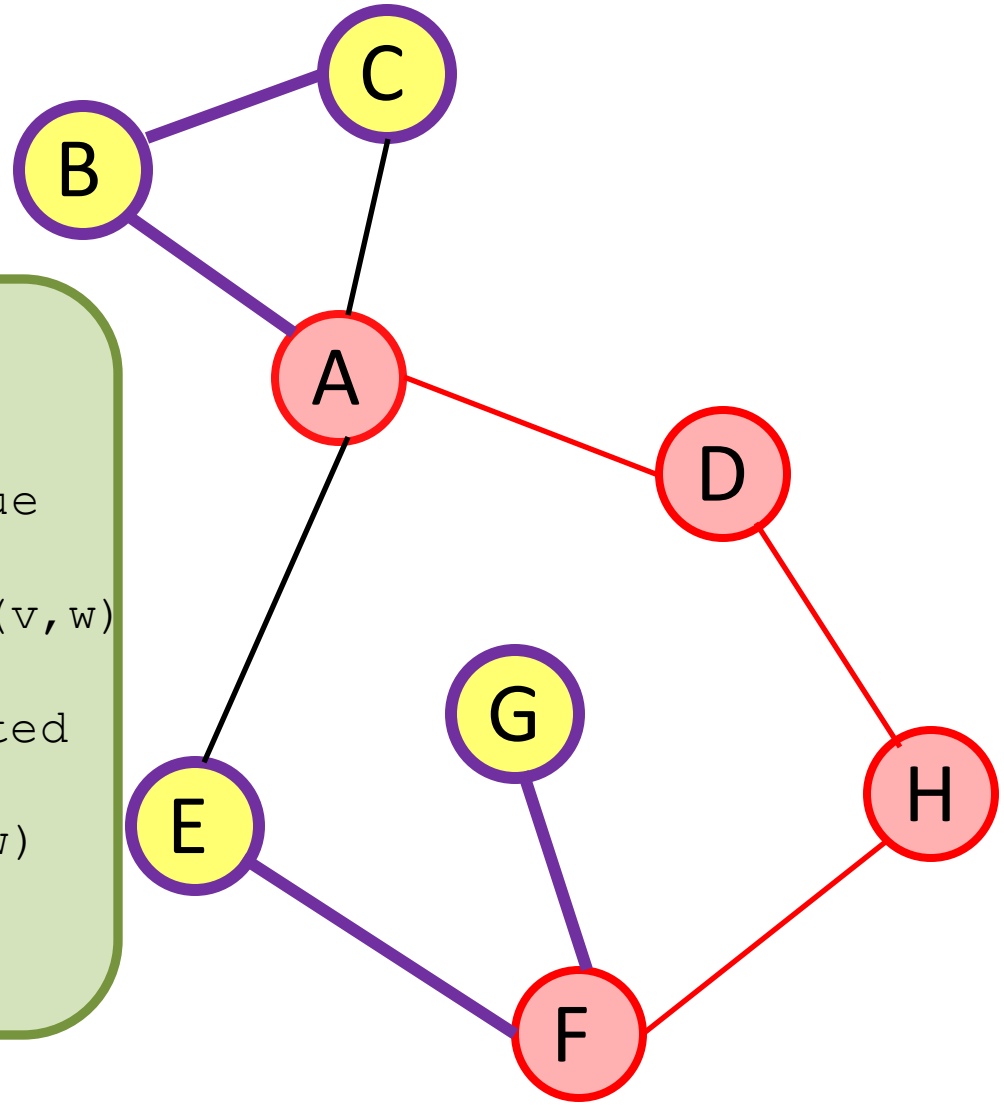
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)
          explore(F)
        explore(H)
explore(E)
```

Explore (Depth First): Example

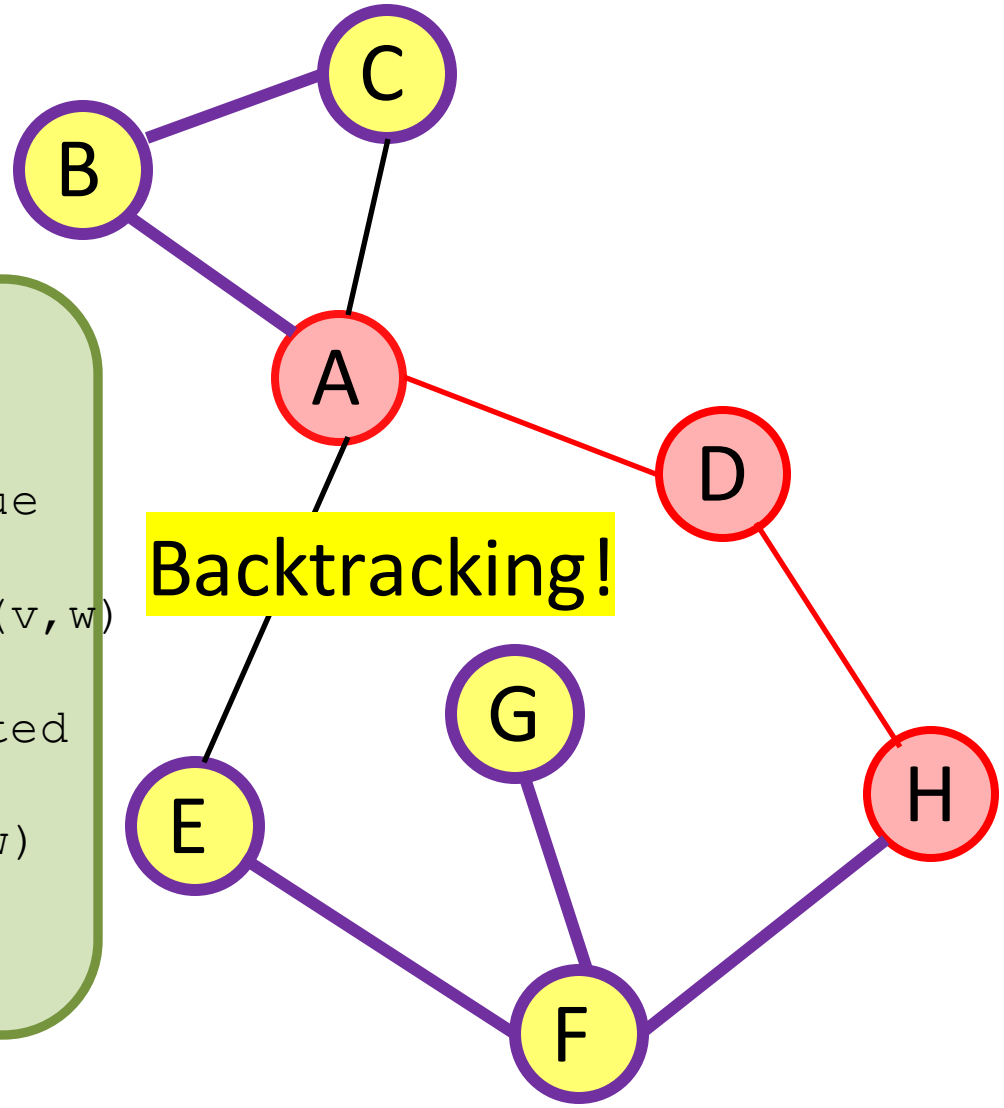
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)
          explore(F)
        explore(H)
explore(E)
```

Explore (Depth First): Example

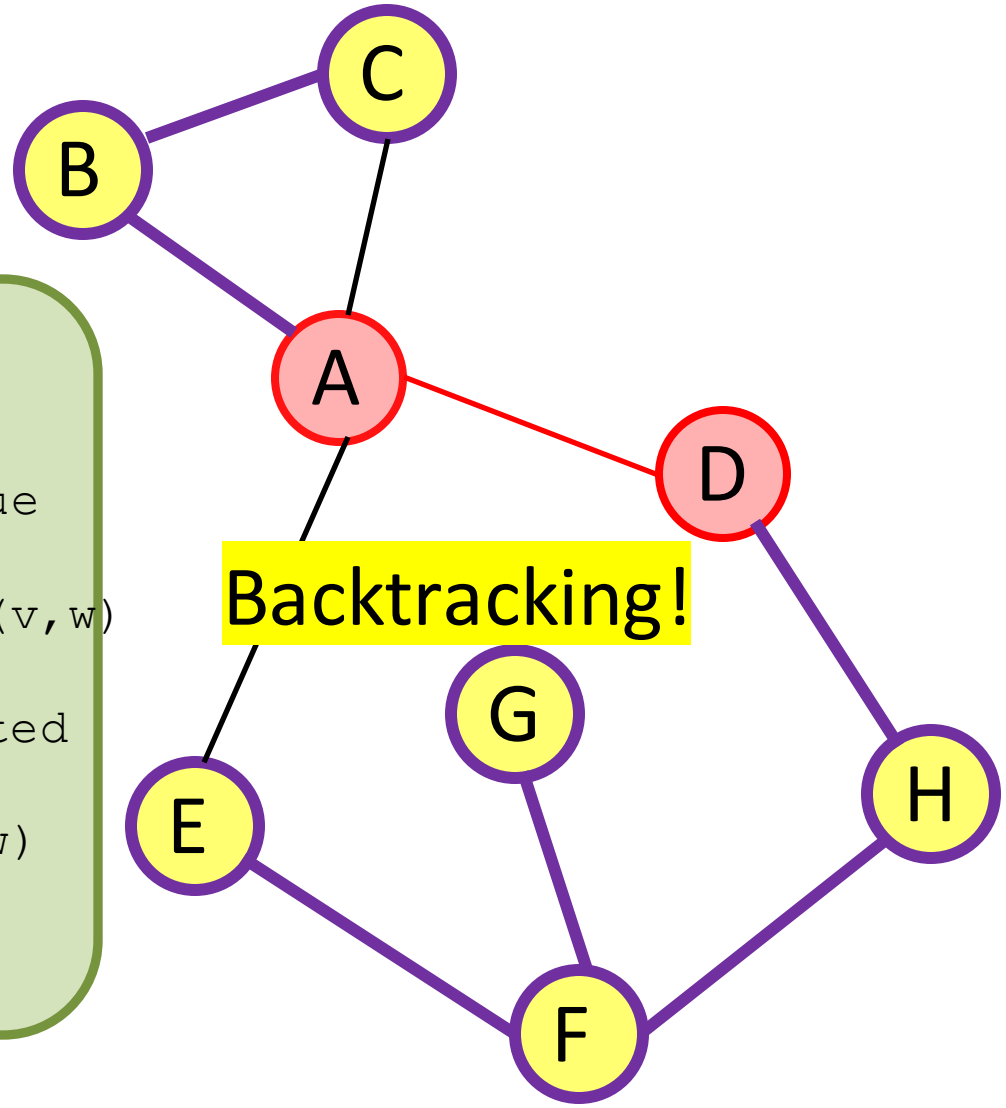
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)
          explore(F)
        explore(H)
explore(E)
```

Explore (Depth First): Example

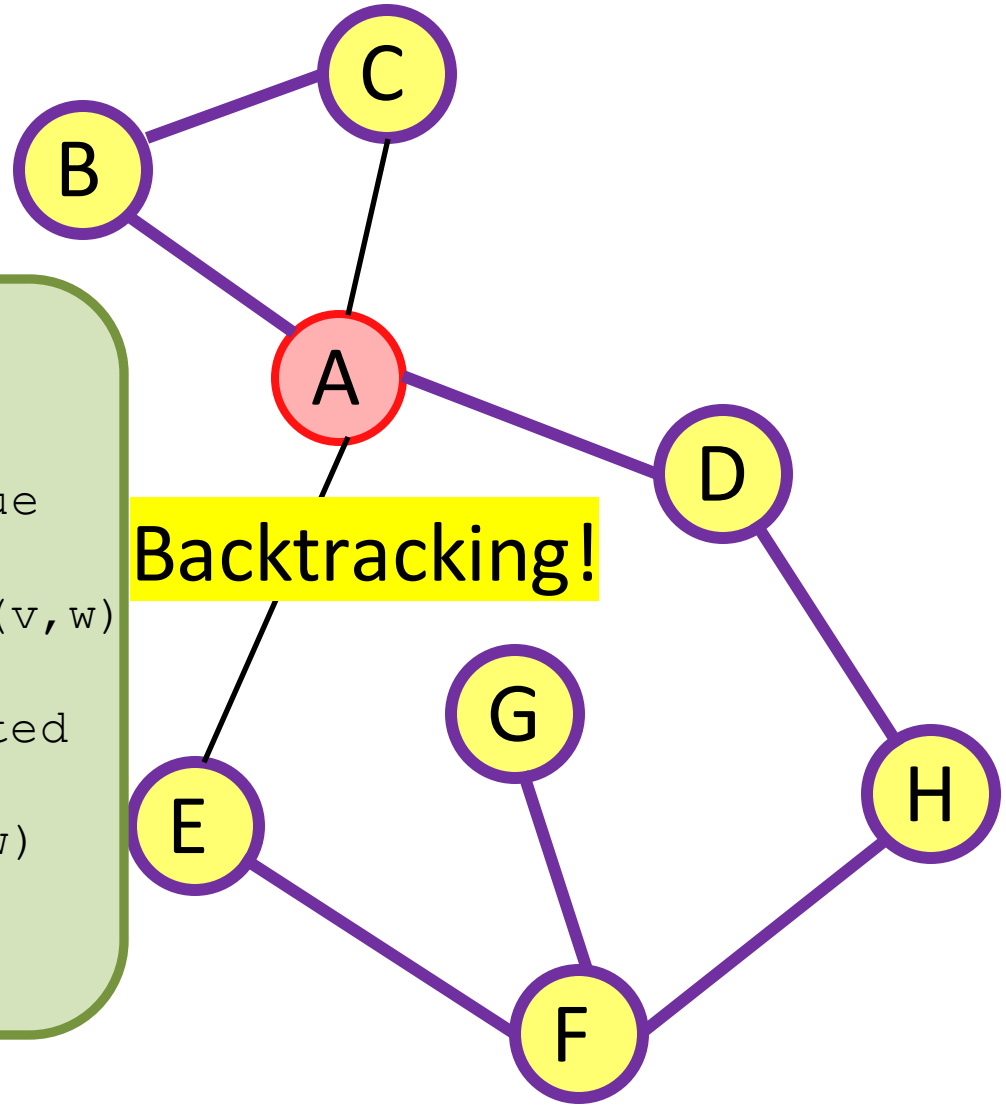
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)
          explore(F)
        explore(H)
explore(E)
```

Explore (Depth First): Example

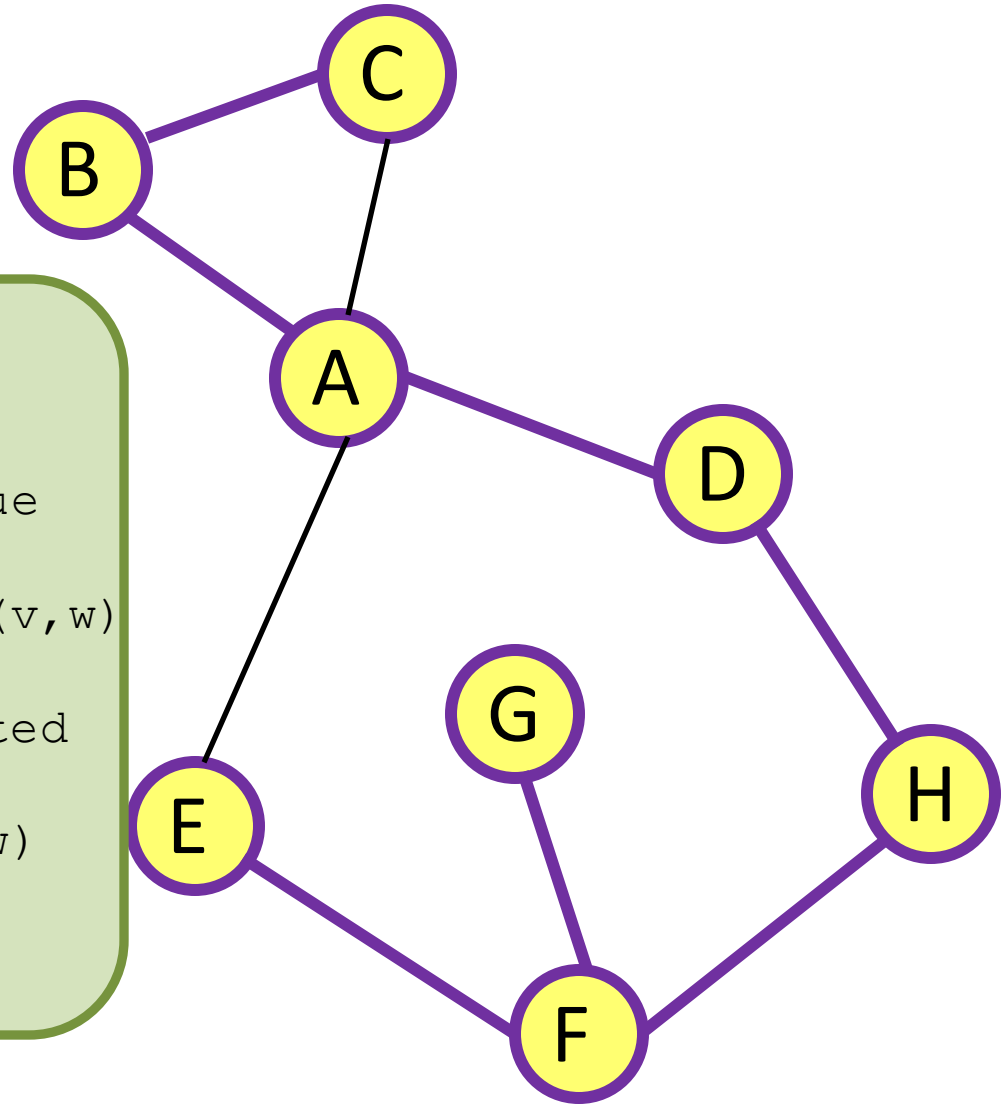
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)
          explore(F)
        explore(H)
  explore(E)
```

Explore (Depth First): Example

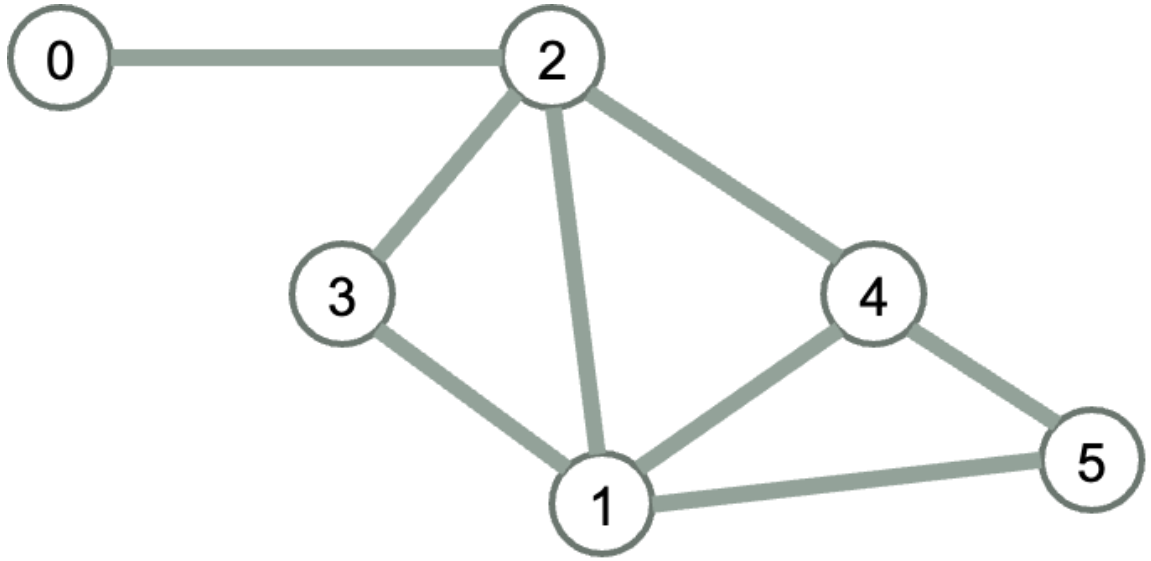
```
exploreDFS(v)

v.visited ← true

For each edge (v,w)

 If not w.visited

   exploreDFS(w)
```




```
explore(A)
  explore(B)
    explore(A)
    explore(C)
      explore(A)
      explore(B)
  explore(C)
  explore(D)
    explore(A)
    explore(H)
      explore(D)
      explore(F)
        explore(E)
          explore(A)
          explore(F)
        explore(G)
          explore(F)
        explore(H)
  explore(E)
```

Explore (Depth First)

Search as far down a single path as possible, backtrack as needed



Assuming `exploreDFS` chooses the lower number node to explore first,
in what order does `exploreDFS` visit the nodes in this graph starting at source 0?

A. 0, 2, 0, 1, 3, 4, 5
B. 0, 2, 3, 4, 1, 5
C. 0, 2, 1, 3, 4, 5
D. Something else

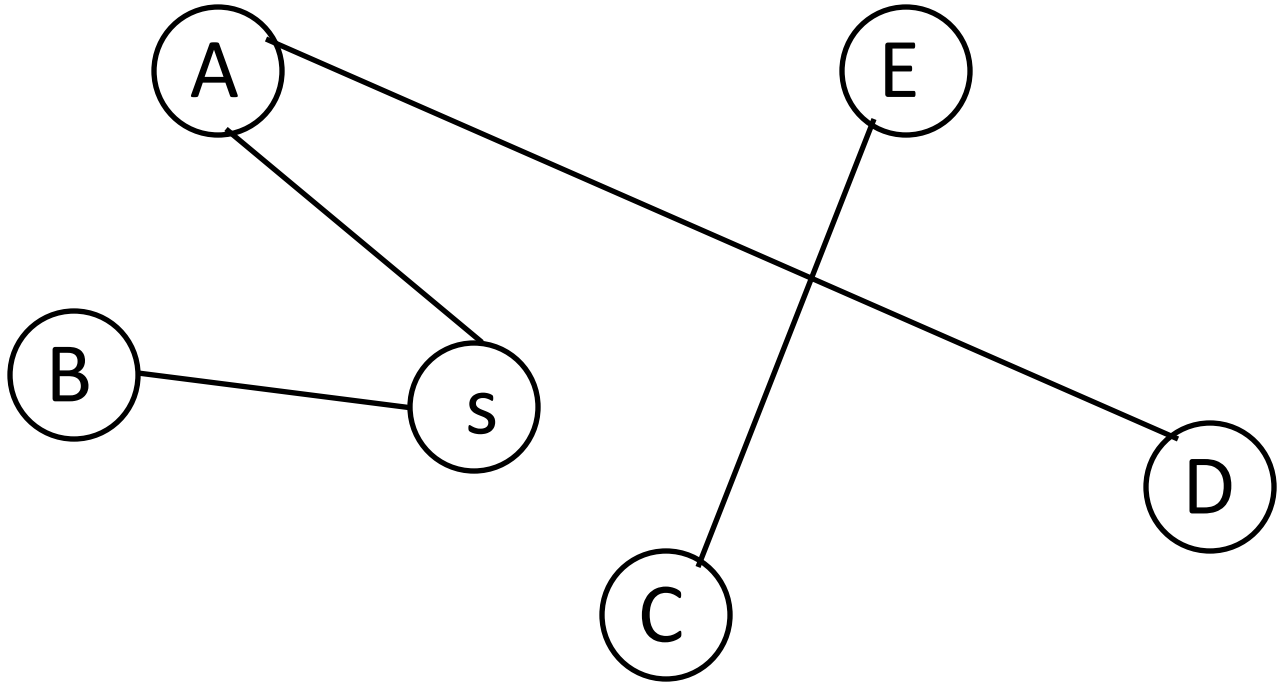
Question: exploreDFS

Which vertices does exploreDFS(s) mark as visited?

A. All the vertices

B. All vertices except C & E

C. None of the above



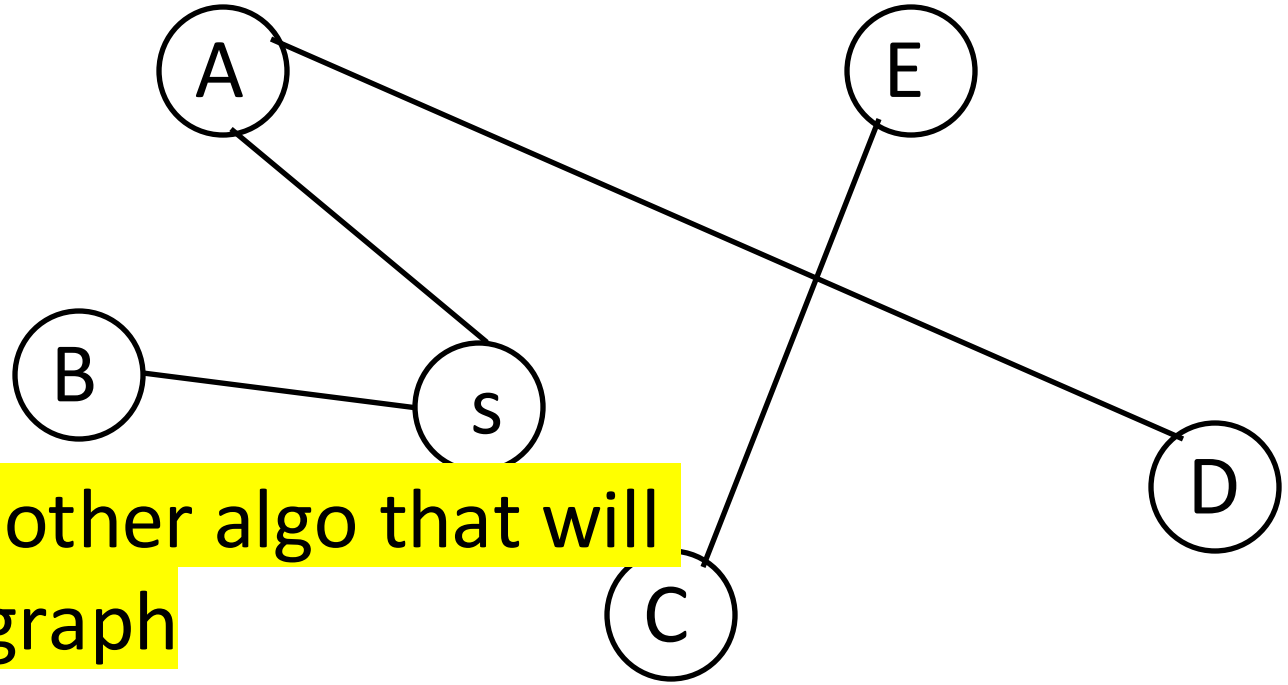
Question: exploreDFS

Which vertices does exploreDFS(s) mark as visited?

A. All the vertices

B. All vertices except C & E

C. None of the above



Use exploreDFS to write another algo that will visit all the vertices in this graph

Depth First Search

`exploreDFS` only finds the part of the graph reachable from a single vertex. If you want to discover the entire graph, you may need to run it multiple times.

```
DepthFirstSearch(G)

  Mark all v ∈ G as unvisited

  For v ∈ G

    If not v.visited, exploreDFS(v)
```

There are n rooms labeled from 0 to n - 1 and all the rooms are locked except for room 0. Your goal is to visit all the rooms. However, you cannot enter a locked room without having its key.

When you visit a room, you may find a set of distinct keys in it. Each key has a number on it, denoting which room it unlocks, and you can take all of them with you to unlock the other rooms.

Given an array rooms where rooms[i] is the set of keys that you can obtain if you visited room i, return true if you can visit all the rooms, or false otherwise.

Input: rooms = [[1],[2, 3],[1],[]]
Output: ?

https://leetcode.com/problems/keys-and-rooms/description/

Input: rooms = [[1],[2, 3],[1],[]]
Output: true
Explanation:
We visit room 0 and pick up key 1.
We then visit room 1 and pick up keys 2 and 3.
We then visit room 2 and pick up key 1.
We then visit room 3.
Since we were able to visit every room, we return true.

Cast as a graph exploration problem

https://leetcode.com/problems/keys-and-rooms/description/

Before next lecture…

Complete the preclass activities from last lecture if you haven't done so already.

- Review pa03 tutorial: https://ucsb-cs24.github.io/s25/pa/pa03-tutorial/
- Watch intro video on NN (3Blue1Brown) : https://youtu.be/aircAruvnKk?feature=share

Next lecture preclass activities:

- Watch videos from statQuest:
  - Neural Network Basics (great for understanding the prediction algorithm): https://youtu.be/CqOfi41LfDw?si=8waS2U01uMWcpH2i
  - Back Propagation (great for understanding the contribute algorithm): https://youtu.be/IN2XmBhILt4?si=bnDft-3T4DQ2iO9X
- Finish the PA03 "check your understanding assignment" on Gradscope.

Acknowledgements

Slides on Depth First Search and animation from Prof. Daniel Kane at UC San Diego