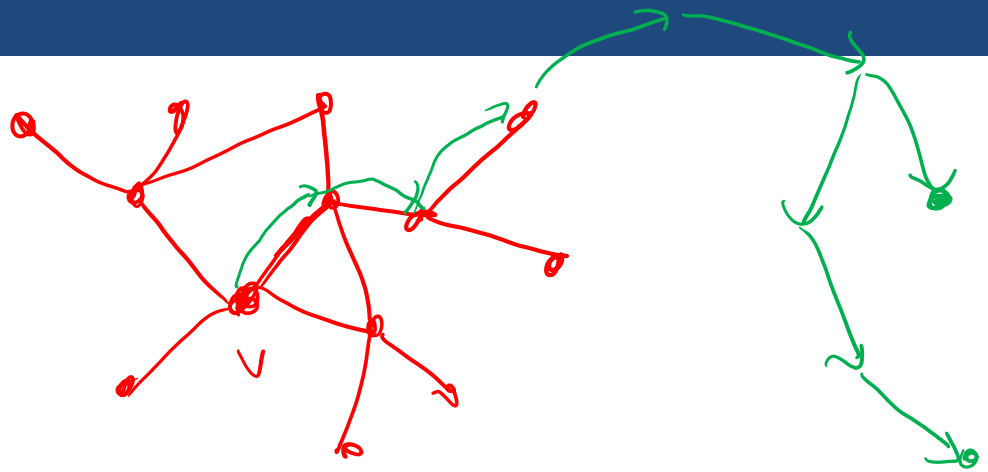
Announcements

Announcements

- HW1 grades are done
  - Note: the TAs are new and forgot to leave feedback!
  - They are going to try to go back and add some
  - In the meantime, check the posted solutions on Blackboard
  - If you can't figure out why you lost points, e-mail the TAs and ask
- HW2 is due ~~today~~ March 4
  - Any questions on HW2?
- HW3 has been posted – March 11



Graphs

Depth First Search

N = # nodes
M = # edges
adjacency matrix

```
procedure explore(G, v)
Input:   G = (V, E) is a graph; v ∈ V
Output:  visited(u) is set to true for
         all nodes u reachable from v

previsit(v)
visited(v) = true
for each edge (v, u) ∈ E:
  if not visited(u): explore(u)
postvisit(v)
```

called N times, O(N)

O(N)

Ignore for now

O(M)

O(M)

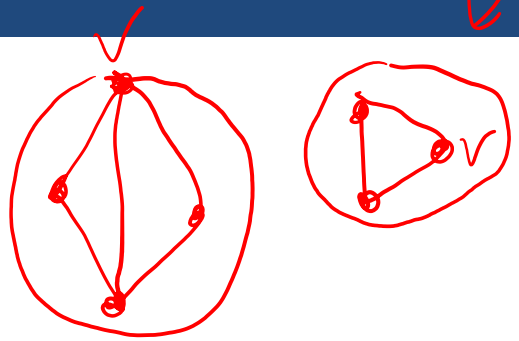
O(N)

O(N)

What is the running time of Depth First Search? O(N+M)

Depth First Search

nodes are indexed 1 ... N



DFS(G):

~~pick a~~

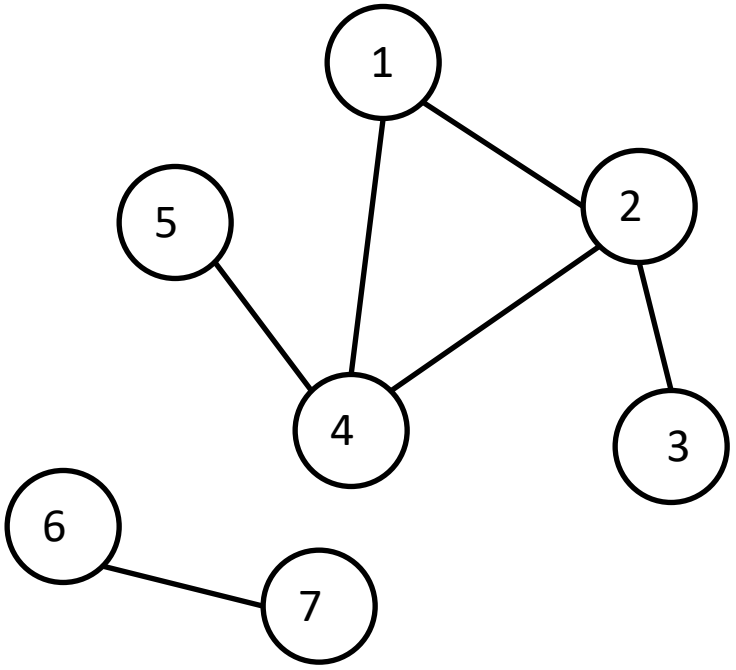
initialize visited = False

while there are ~~still~~ unvisited nodes:

pick an unvisited node v

explore(v)

Depth First Search: Example with Stacks

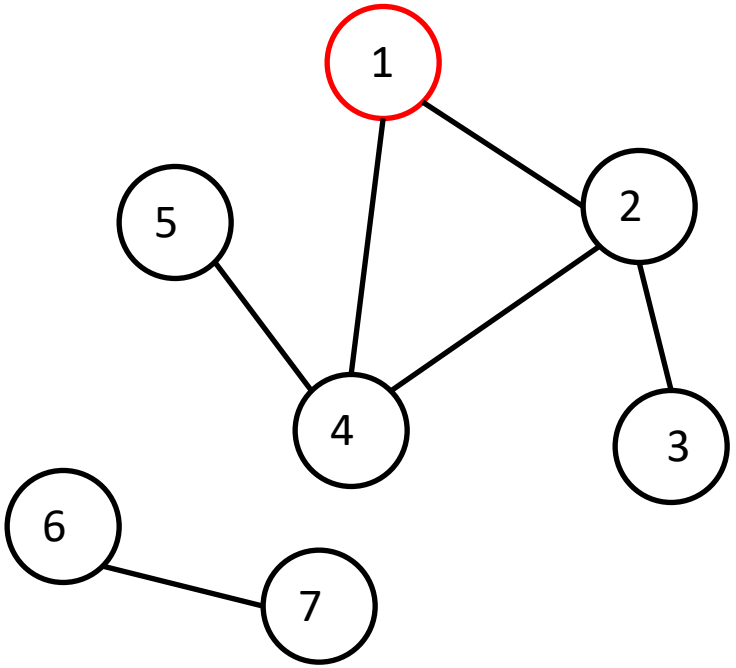


v = 1

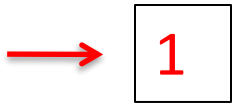
Start by putting 1 on the stack

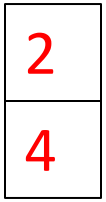
1

Depth First Search: Example with Stacks

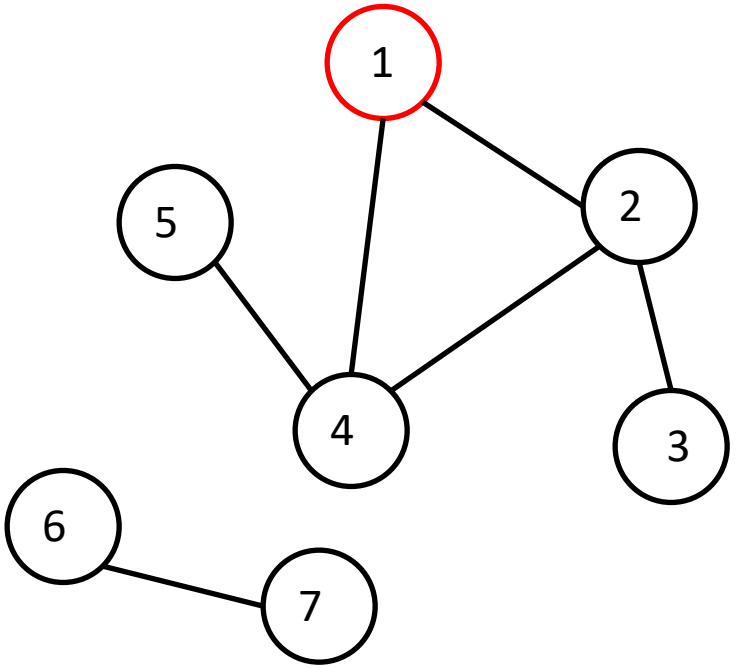


Which node is first on the stack? Pop it from the stack, mark it as explored, put its unexplored neighbors on the stack.





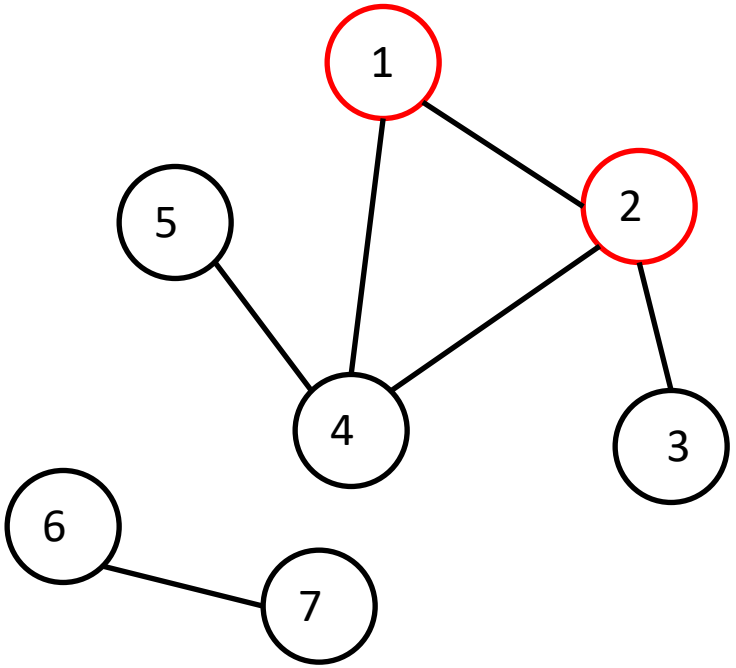
Depth First Search: Example with Stacks



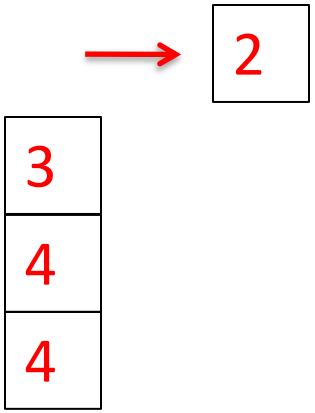
Which node is first on the stack? Pop it from the stack, mark it as explored, put its unexplored neighbors on the stack.

2
4

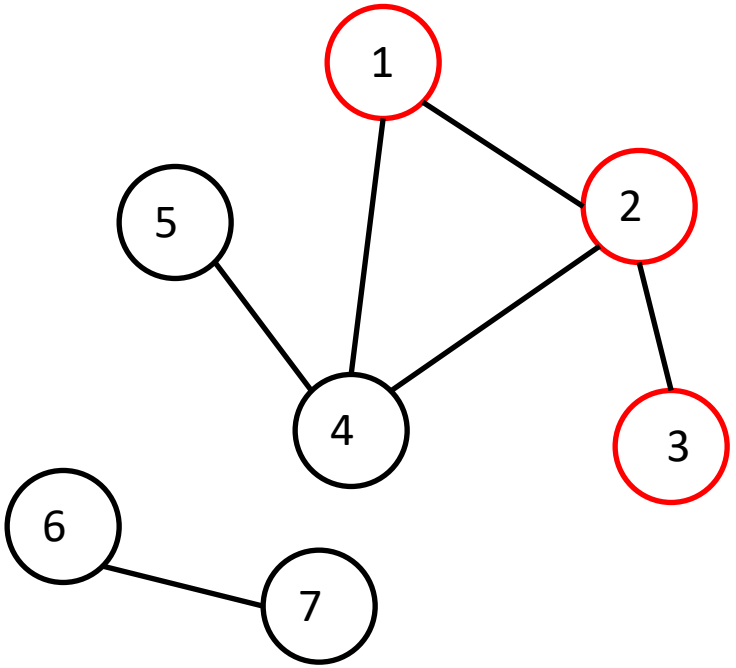
Depth First Search: Example with Stacks



Which node is first on the stack? Pop it from the stack, mark it as explored, put its unexplored neighbors on the stack.

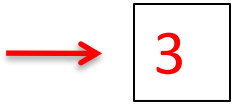


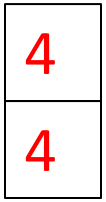
Depth First Search: Example with Stacks



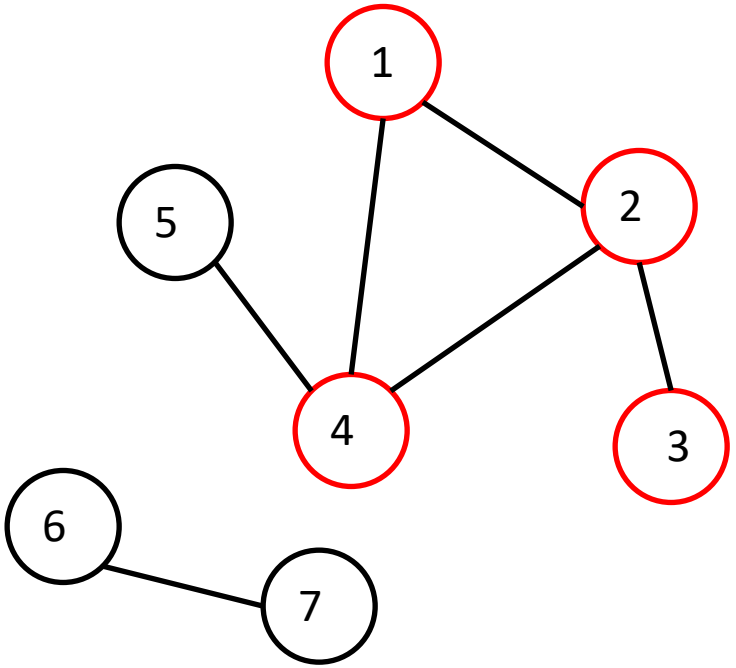
Which node is first on the stack? Pop it from the stack, mark it as explored, put its unexplored neighbors on the stack.

No unexplored neighbors!

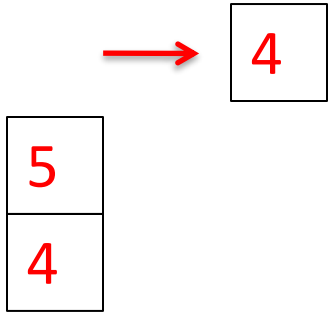




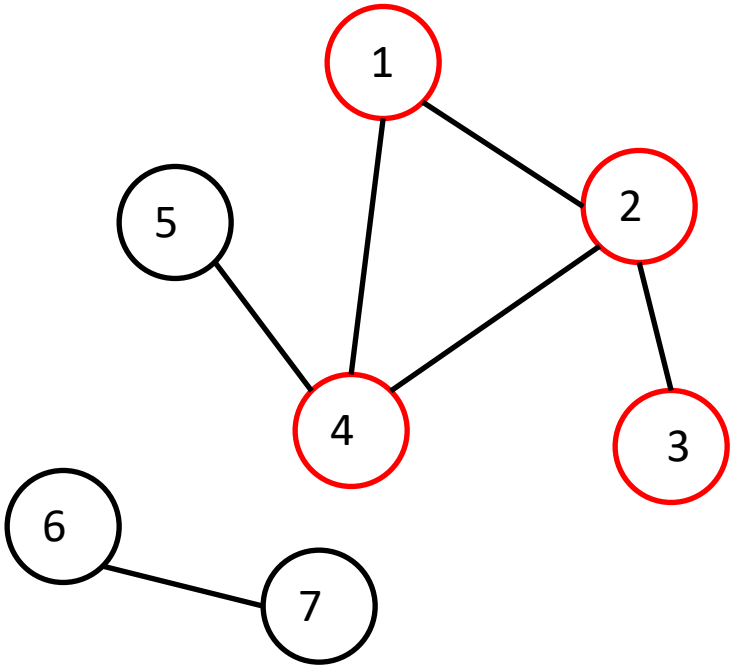
Depth First Search: Example with Stacks



Which node is first on the stack? Pop it from the stack, mark it as explored, put its unexplored neighbors on the stack.

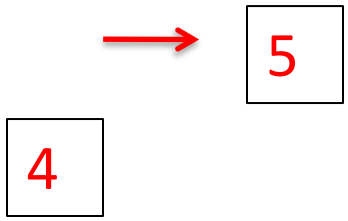


Depth First Search: Example with Stacks

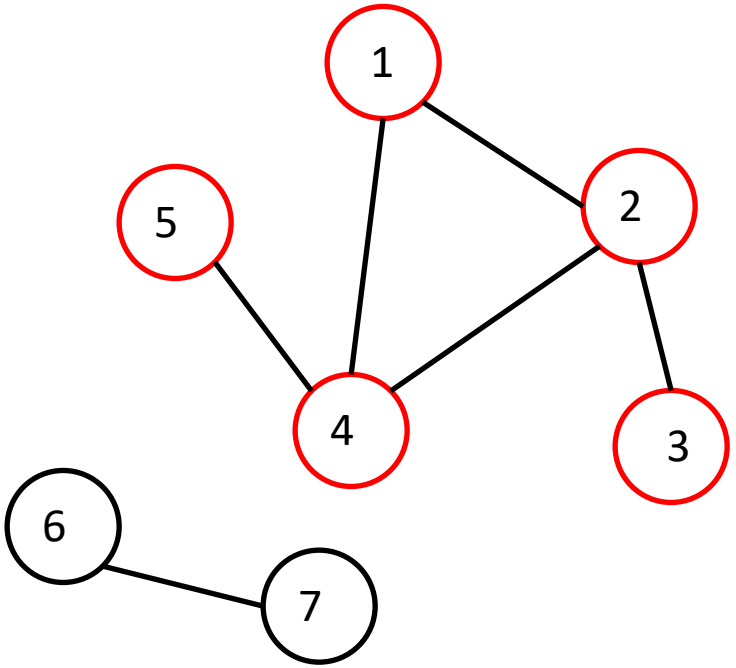


Which node is first on the stack? Pop it from the stack, mark it as explored, put its unexplored neighbors on the stack.

No unexplored neighbors!

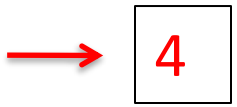


Depth First Search: Example with Stacks



Which node is first on the stack? 4 is, but it has already been explored!

Done!



Timekeeping

```
procedure previsit(v)
pre[v] = clock
clock = clock + 1

procedure postvisit(v)
post[v] = clock
clock = clock + 1
```

pre[*u*] to post[*u*] is the time *u* is on the stack

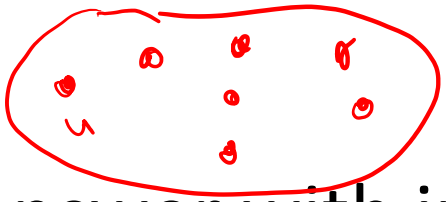
For all nodes *u*, *v*, either [pre[*u*], post[*u*]] is completely within [pre[*v*], post[*v*]], or the other way around, or there is no overlap. Why?

In-class Exercise

explore(u)

Suppose we start at node *u*. Are we guaranteed to find all nodes reachable from *u*?

IF v is reach. from u, there is a path from u→v



Let p_v be the length of shortest path from $u \to v$

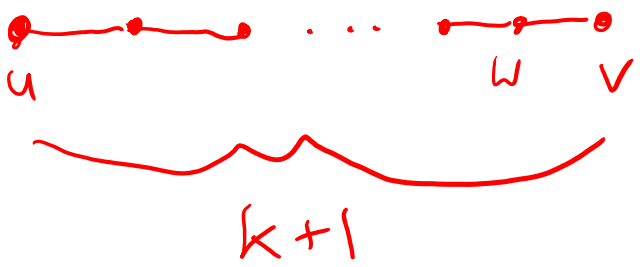
Prove your answer with induction!

Base case: $p_v = 0$. The only node within 0 edges of u is u itself. explore(u) marks u as visited. ✓

I.H.: Assume all v with $p_v \leq k$ are marked. as visited.

I.S.: We need to show that if a node v has $p_v = k+1$, v gets marked.

In-class Exercise



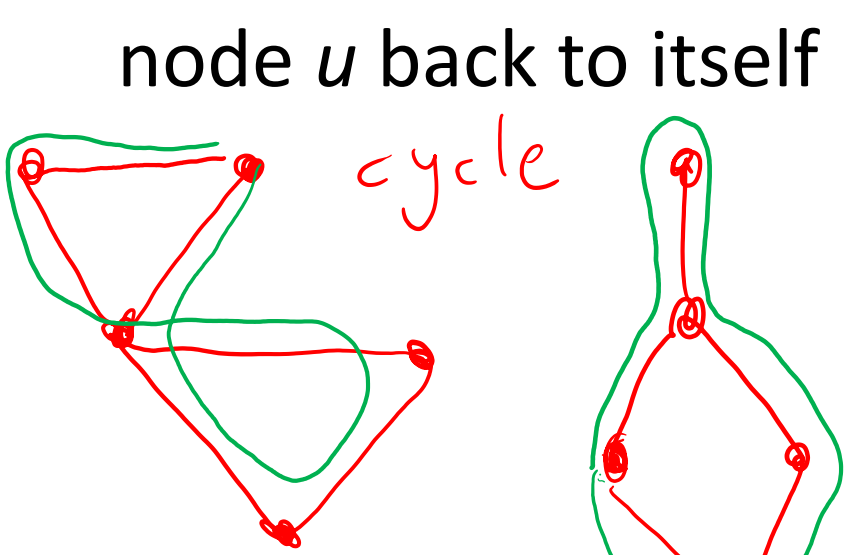
$p_v = k+1$

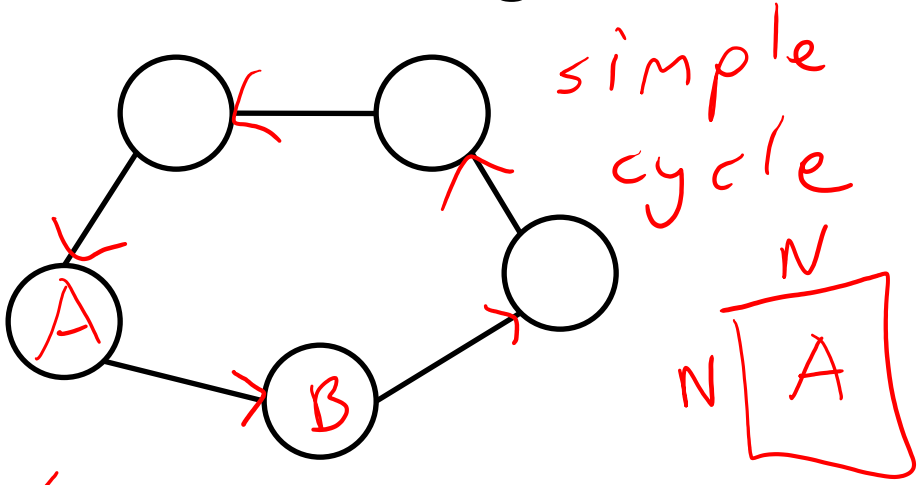
$p_w = k$

Let w be the node right before v on path.
$p_w = k$. If $p_w < k$, then $p_v < k+1$. If $p_w > k$,
then we could use the above path to find
a shorter path. Because $p_w = k$, it get marked
(by I.H.). Only time a node is marked is
when we call explore on it. As part of explore(w),
we call explore(v), unless v is already marked.
Either way, it gets marked. □

More Terminology

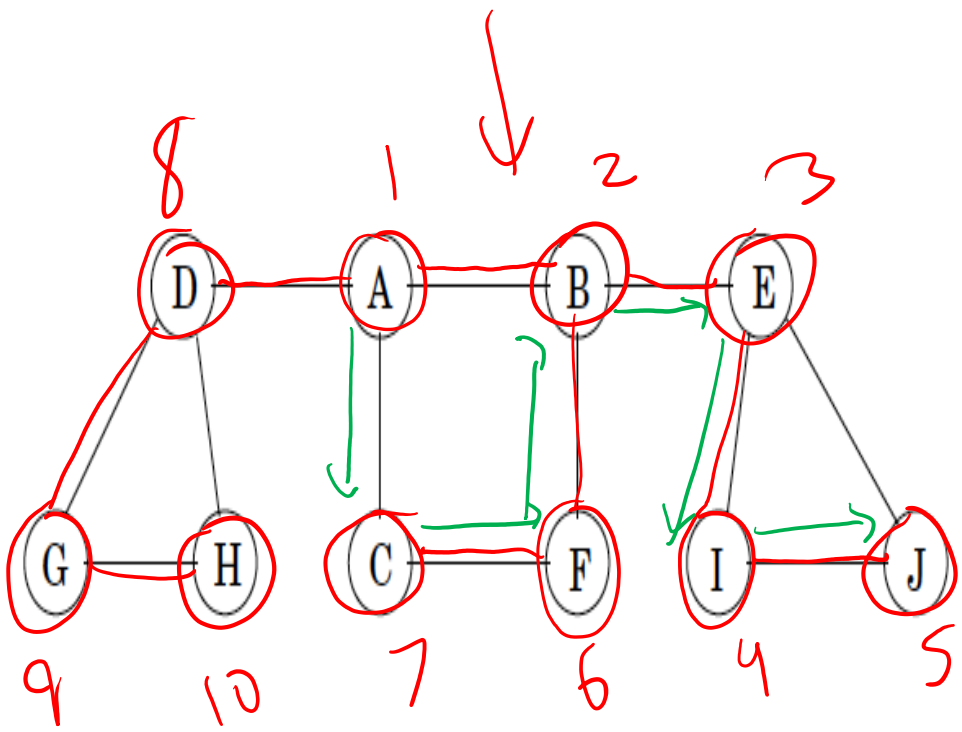
- A cycle is a path with at least two edges from a node *u* back to itself

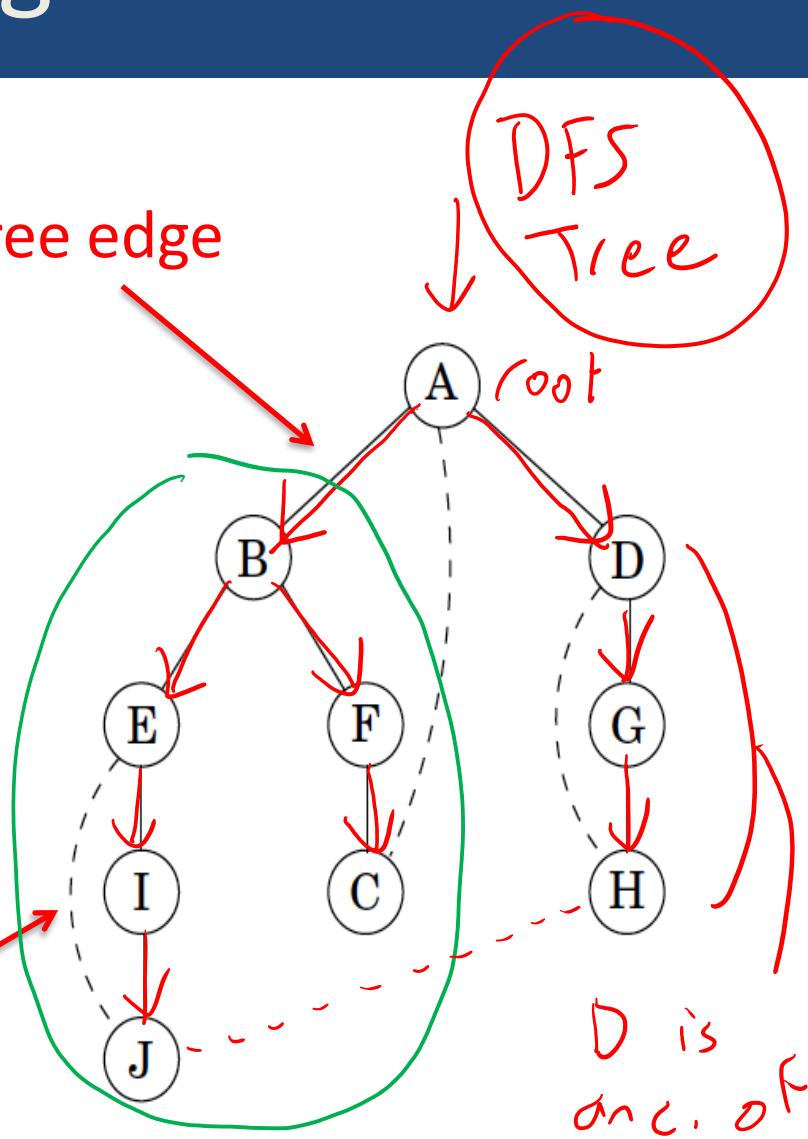




- A tree is a graph without cycles

Tree Edges vs. Back Edges



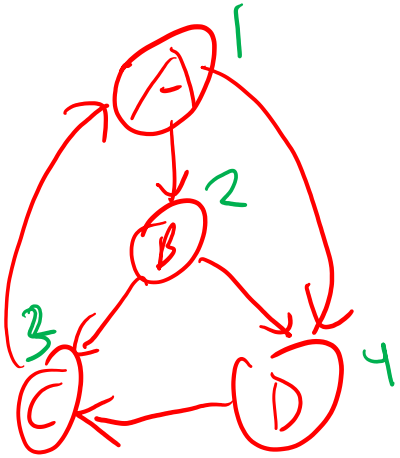


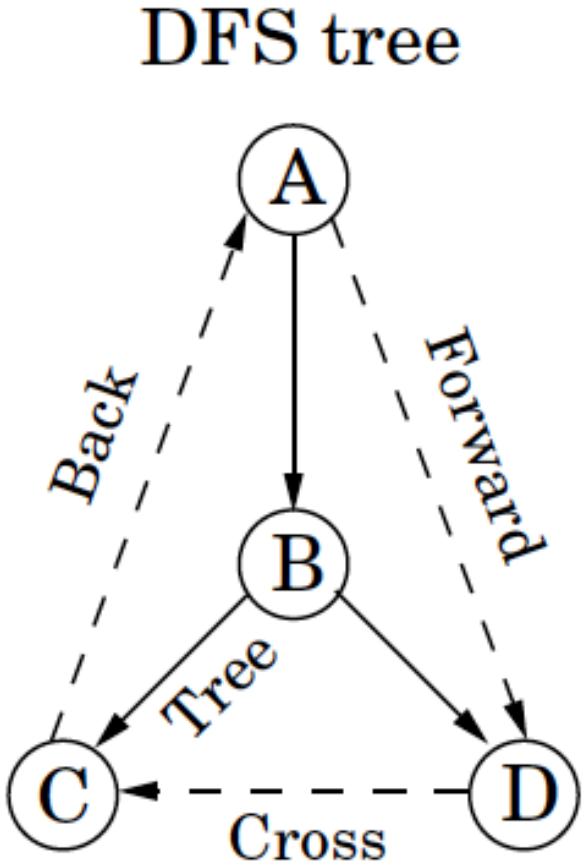
DFS on Directed Graphs

- We can run DFS on directed graphs, making sure to only follow edges in their correct direction
- The starting node is the root of the DFS tree
- *u* is an ancestor of *v* if there is a path from *u* to *v* in the DFS tree. *v* is a descendant of *u*.
- *u* is the parent of *v* if there is a directed edge from *u* to *v* in the DFS tree. *v* is the child of *u*.

DFS on Directed Graphs

- We can run DFS on directed graphs, making sure to only follow edges in their correct direction





A is the root

B is a child of A

B is a parent of C

A is an ancestor of D

In-Class Exercise

1. How many parents can a node have in the DFS tree?
2. How many children can a node have in the DFS tree?
3. How many ancestors can a node have in the DFS tree?
4. How many descendants can a node have in the DFS tree?

Give examples for your answers!