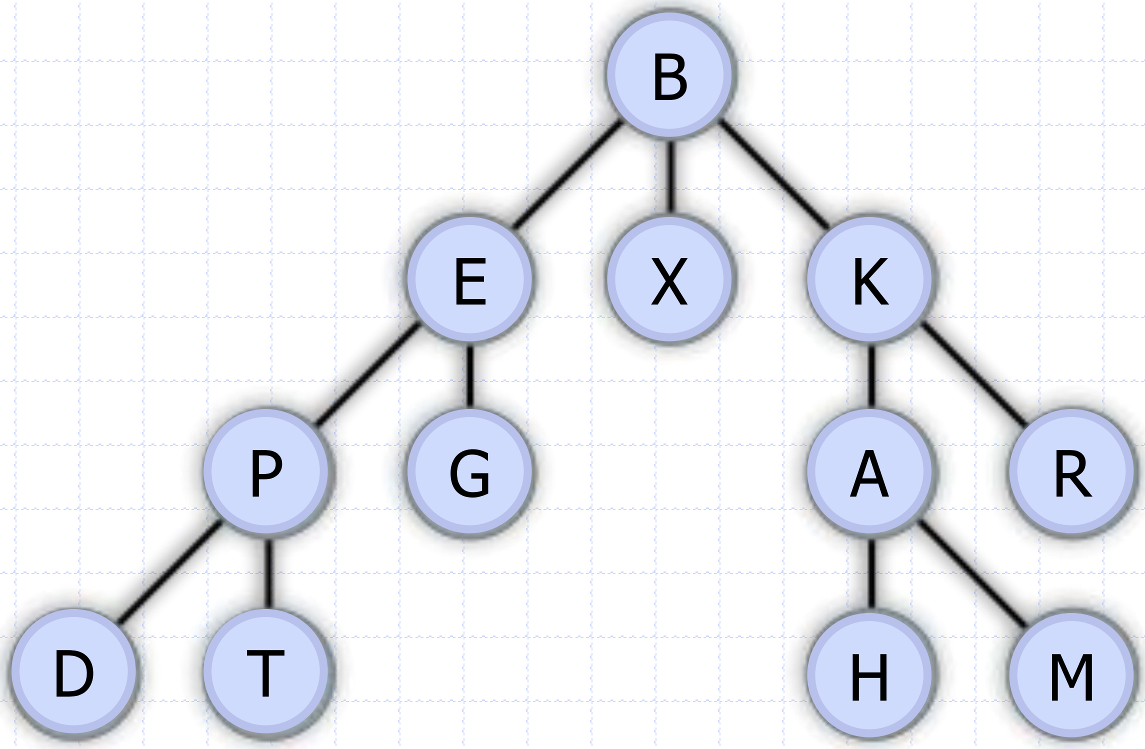
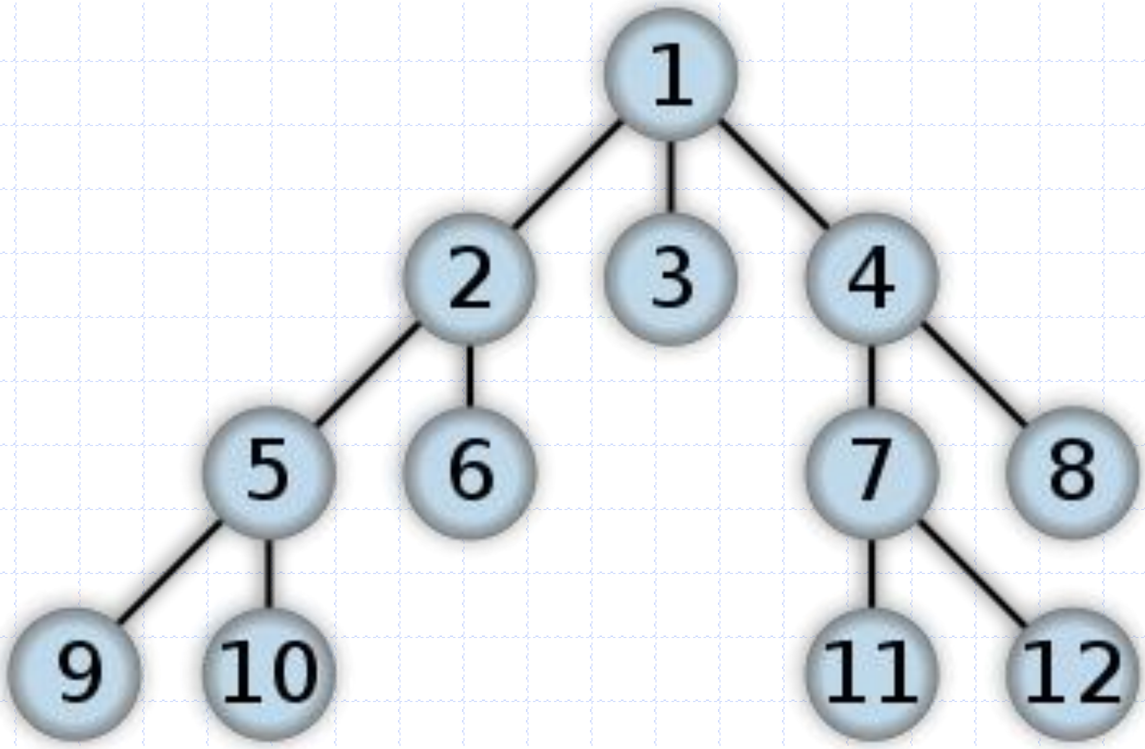
Graph Traversal

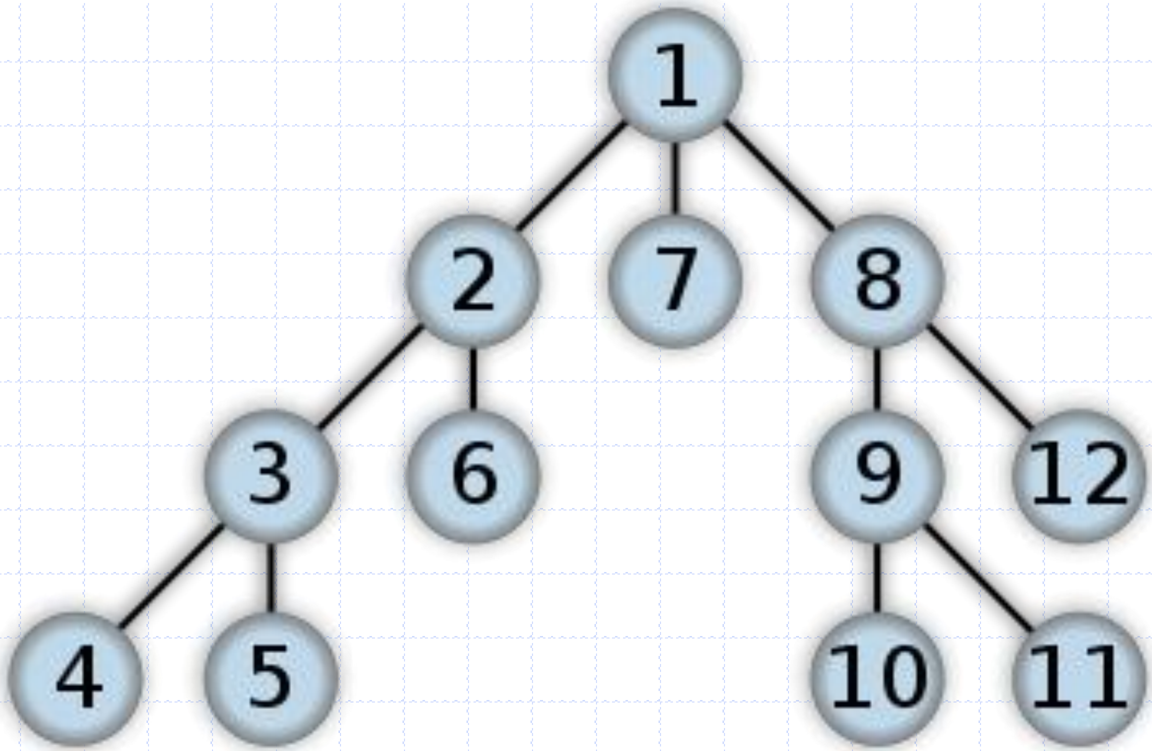
Can you name all the nodes?



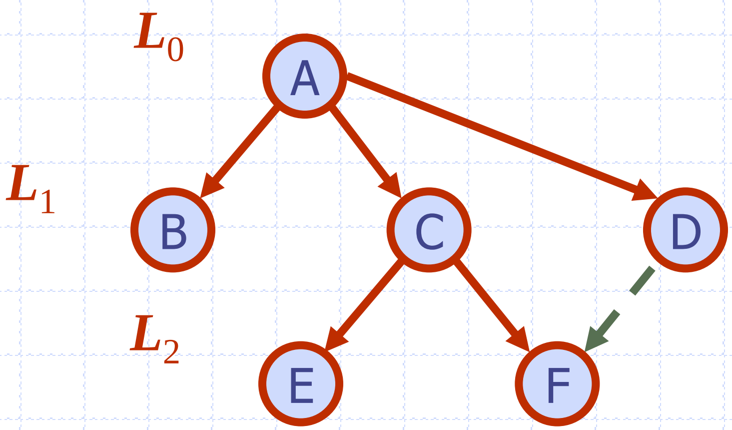
Breadth-First Search



Depth-First Search

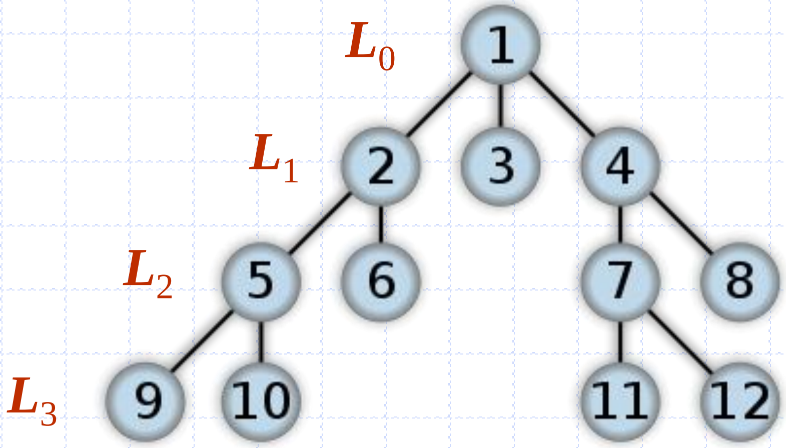


Breadth-First Search

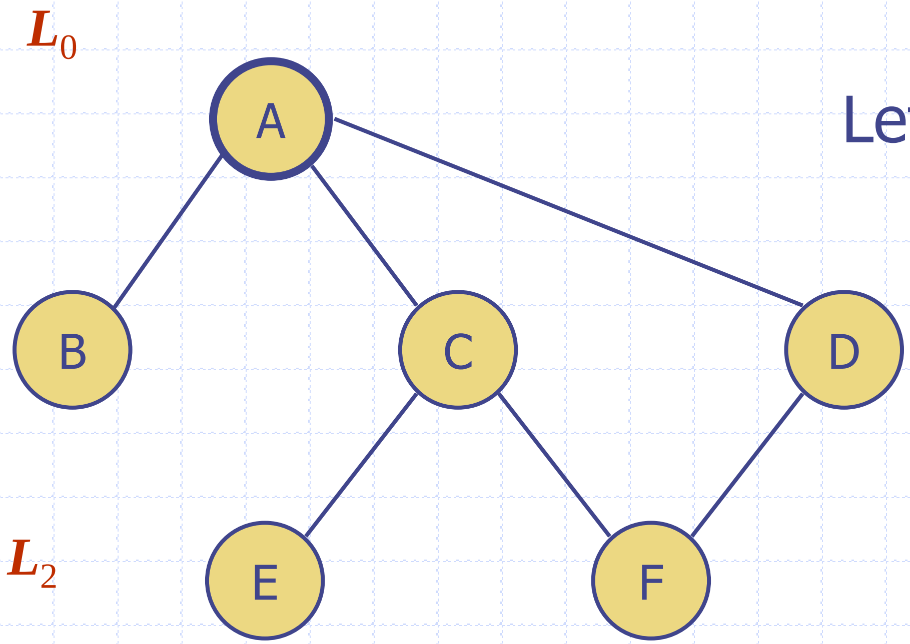


Breadth-First Search

- Breadth-first search (BFS) is a general technique for traversing a graph.
- A BFS traversal of a graph returns the nodes of the graph level by level.

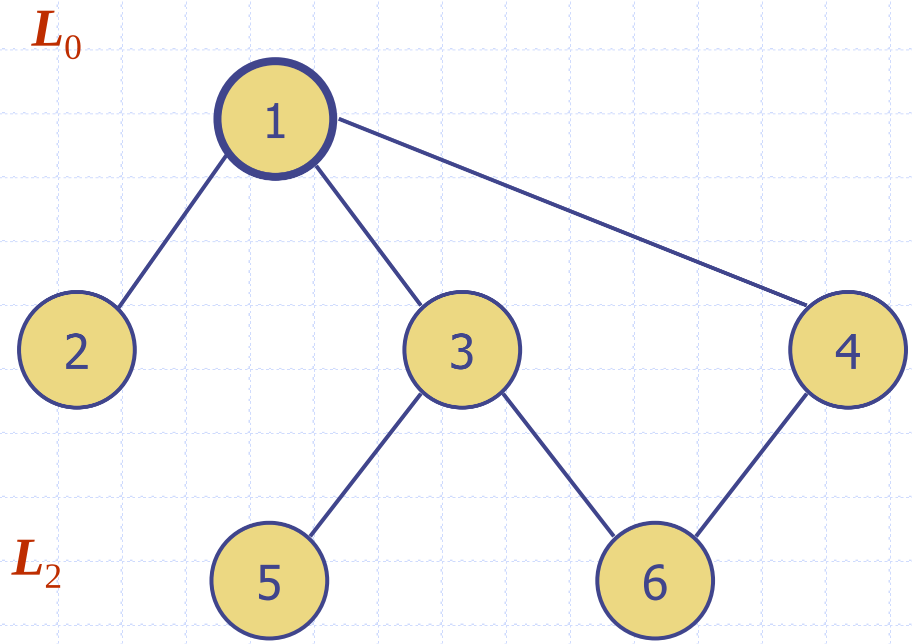


What is a BFS traversal?

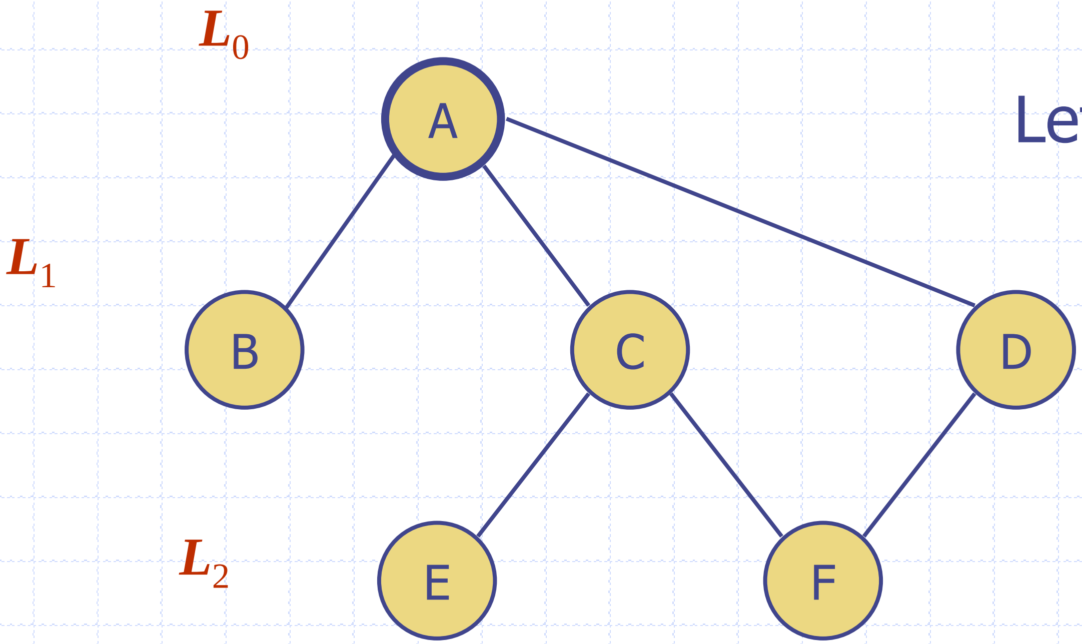


Let's start at A

What is a BFS traversal?



What is a BFS traversal?

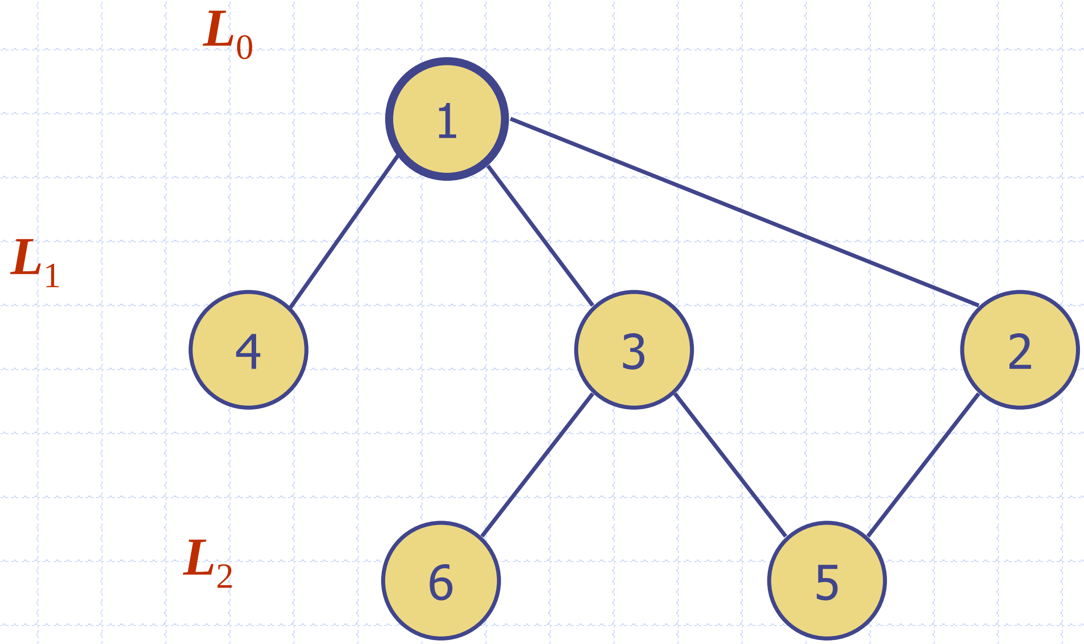


Let's start at A

A B C D E F

Is that the only BFS?

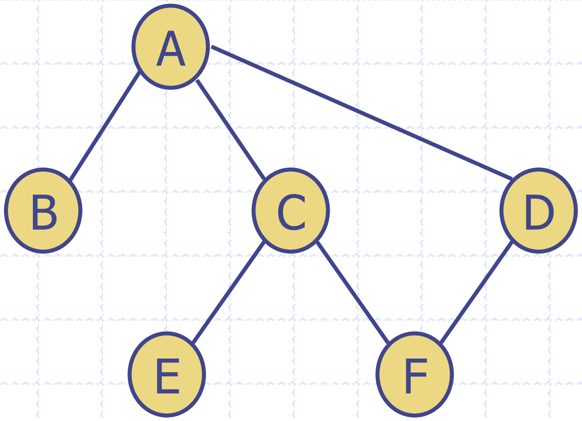
What is a BFS traversal?



Computer Algorithm

- A computer does not see the BIG PICTURE of a graph. It has a list of nodes in a graph and a list of edges, and it knows which nodes are connected by which edge.
- How would a computer perform BFS?

Adjacency Matrix



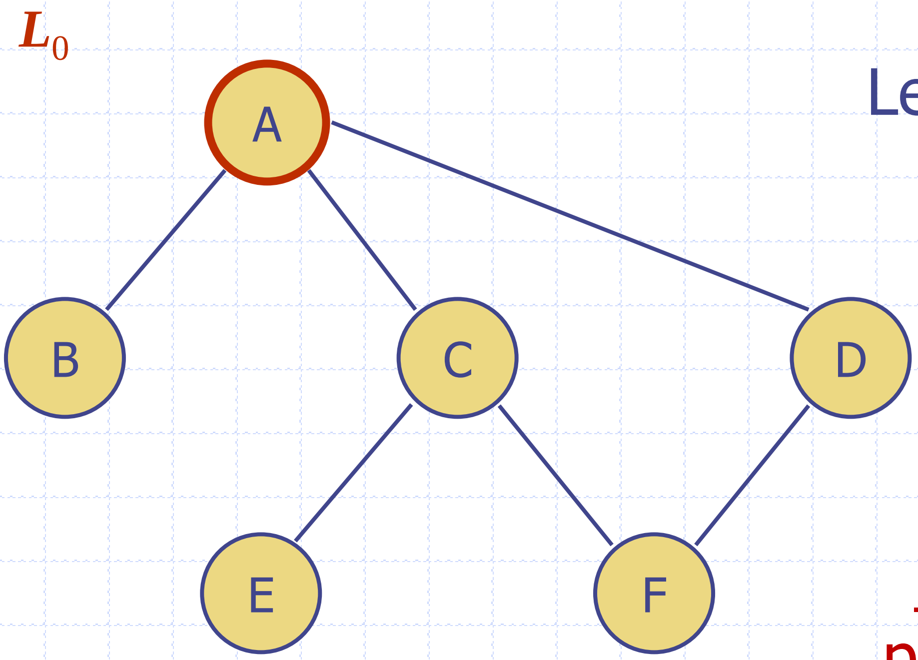
	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	0	0
C	1	0	0	0	1	1
D	1	0	0	0	0	1
E	0	0	1	0	0	0
F	0	0	1	1	0	0

Queue

- A queue is a line.
- If you're the first to get in a bus line, you're the first to get on the bus.

First In, First Out

Example



Let's start at A

Enqueue means to put something at the end of the line.

Enqueue the root A.
Queue: A

Here’s Our Algorithm

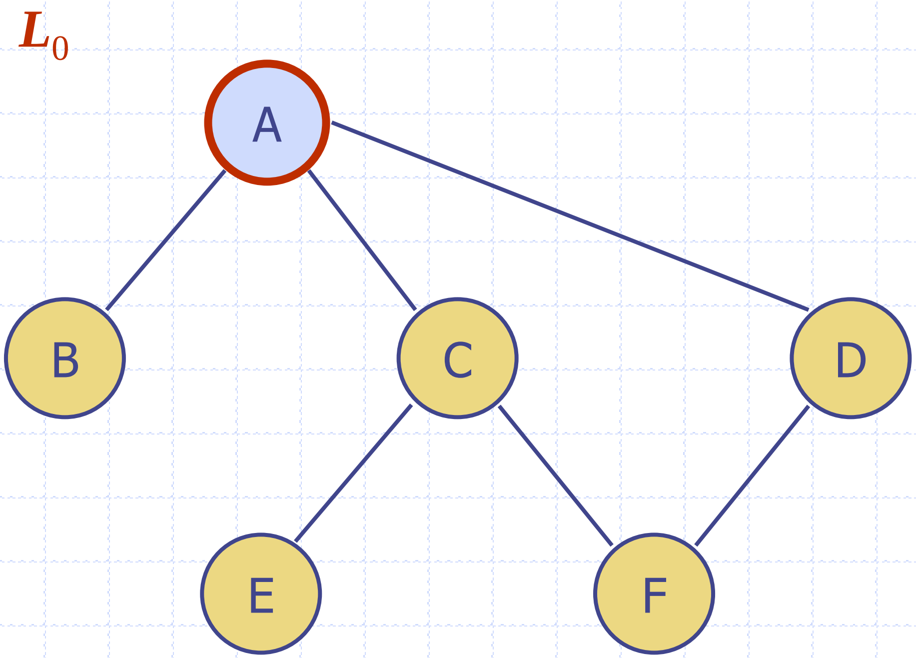
- Each time, we’re going to take out one node (we’ll call it node X) from the queue. We call the process dequeue.
- Then, we’re going to add to the queue all of the new neighbors of node X. We call that process enqueue.

First Step:

DEQUEUE!

(take a node out of the queue)

Example



A

Queue: A
Dequeue A → Queue:

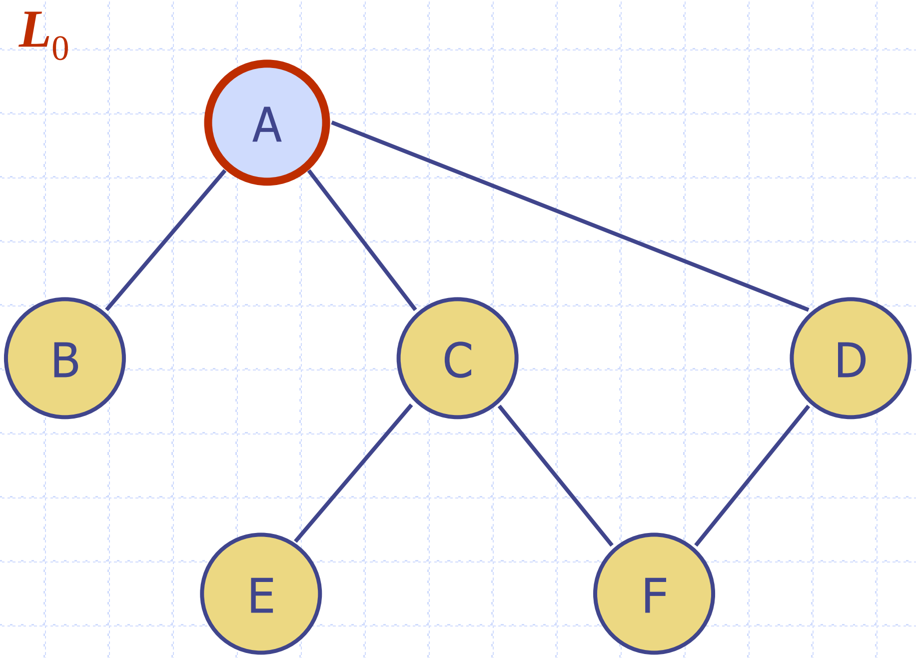
Second Step:

ENQUEUE THE

NEW NEIGHBORS!

(put the nodes into the queue)

Example

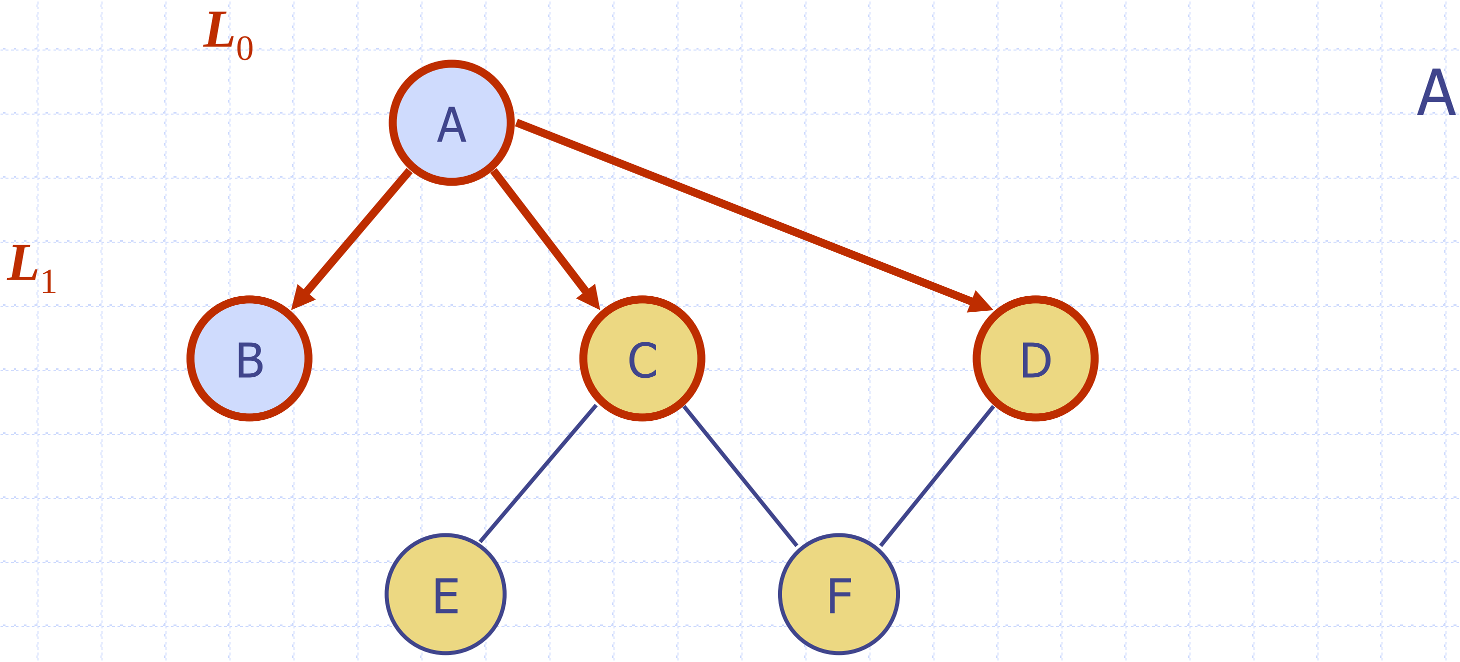


A

What are the new neighbors of A?

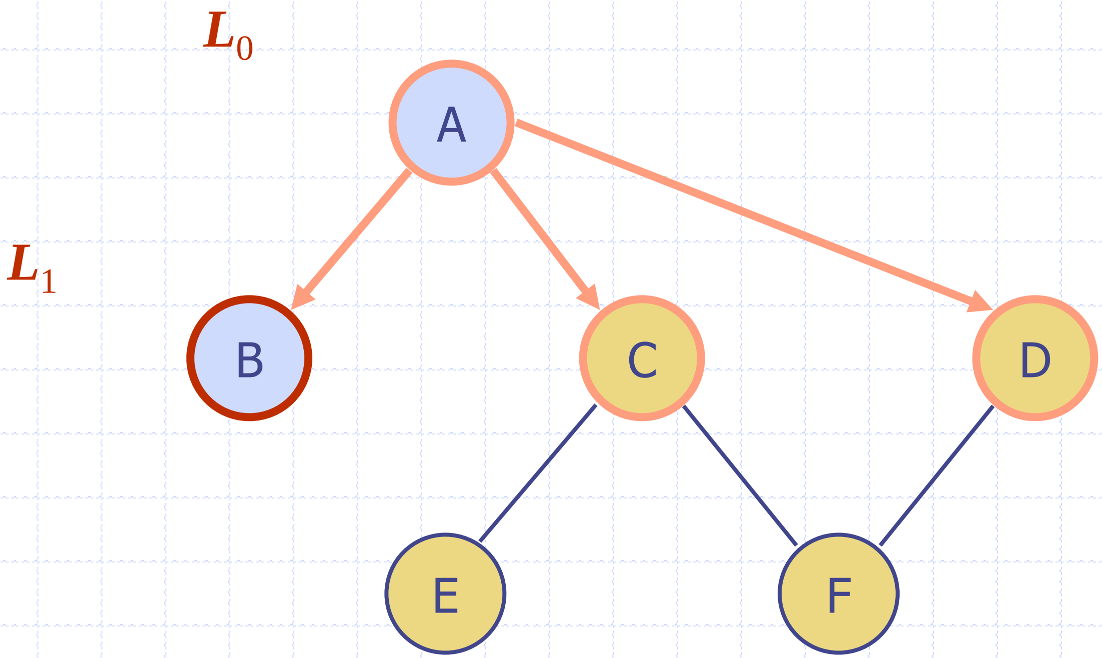
Queue:

Example



Enqueue new neighbors of root A.
Queue: B C D

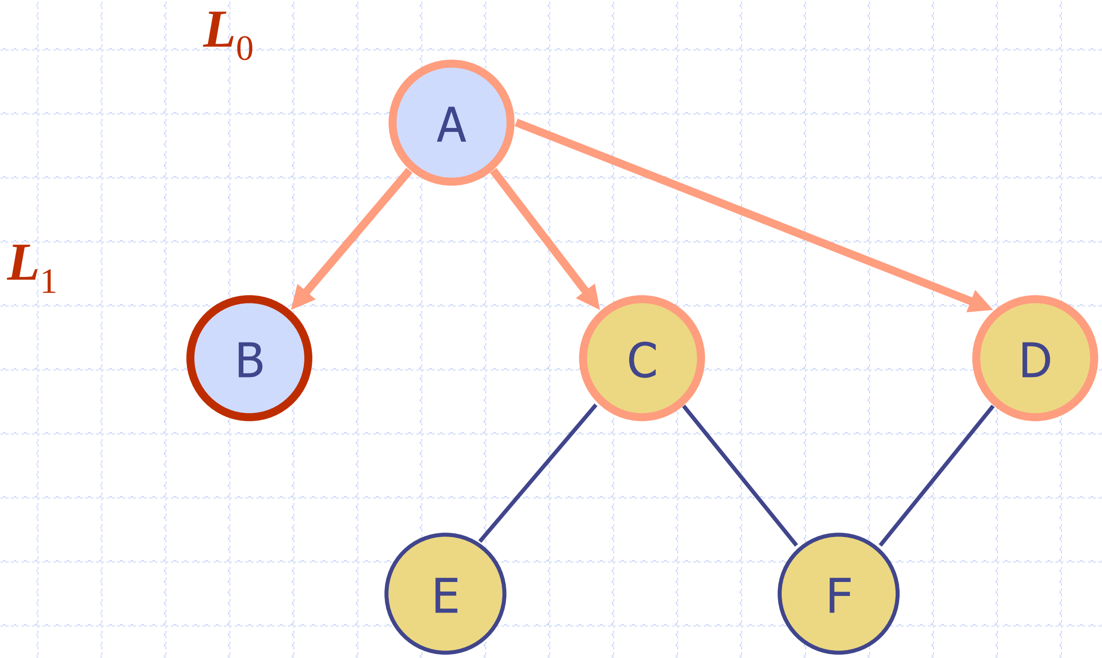
Example



A
B

Queue: B C D
Dequeue B → Queue: C D

Example

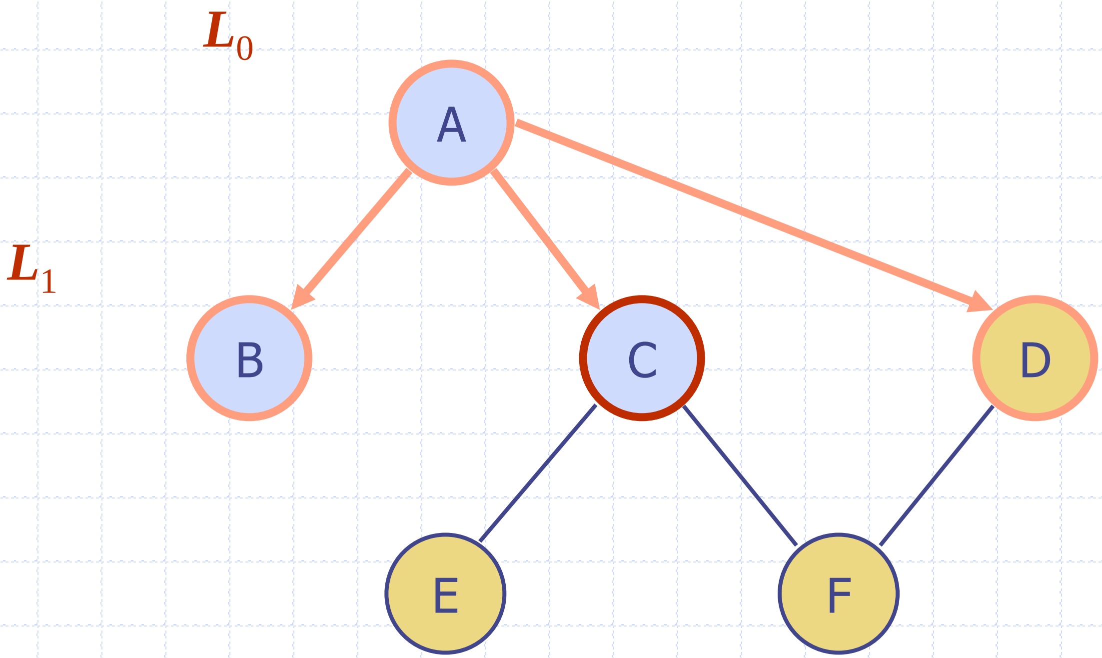


A
B

There are none!

Enqueue new neighbors of B.
Queue: C D

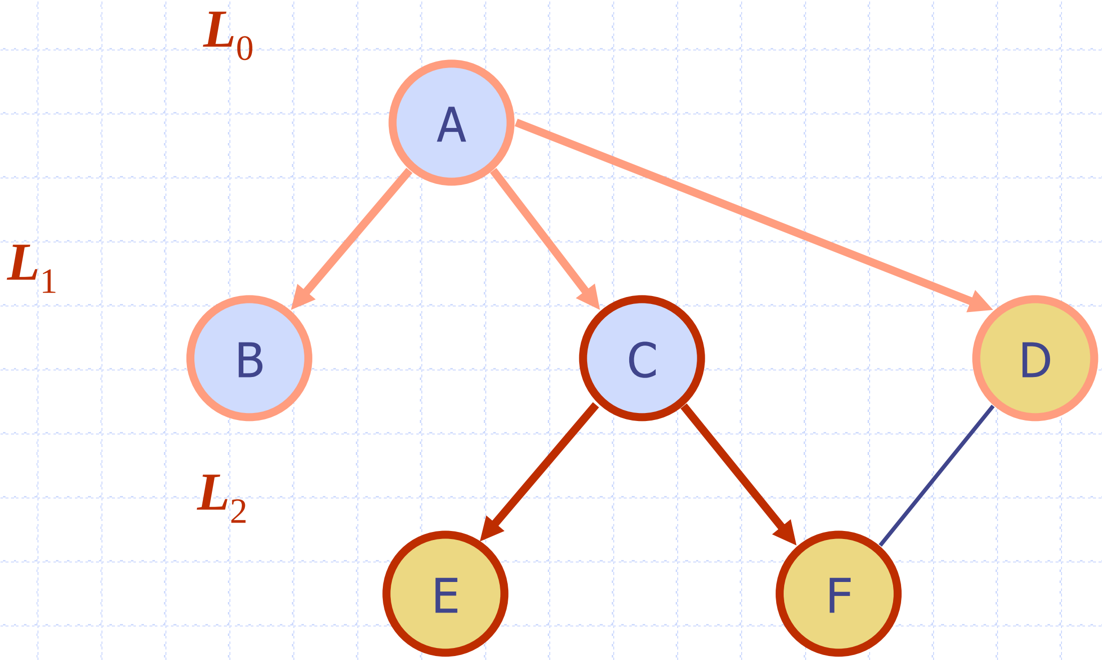
Example



A
B
C

Queue: C D
Dequeue C → Queue: D

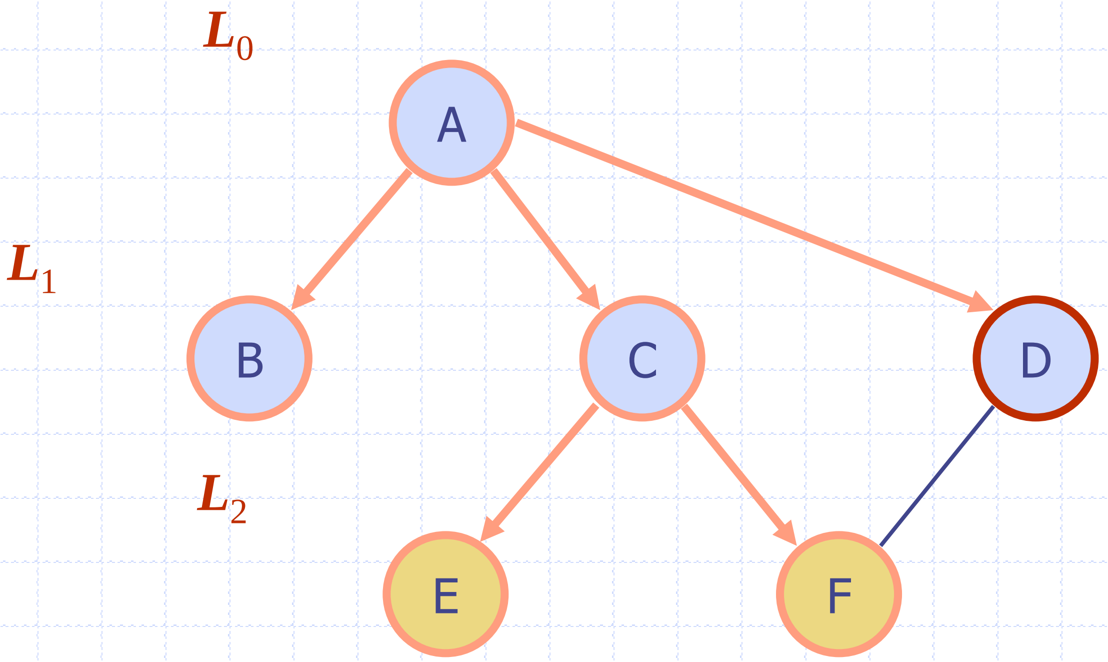
Example



A
B
C

Enqueue new neighbors of C.
Queue: D E F

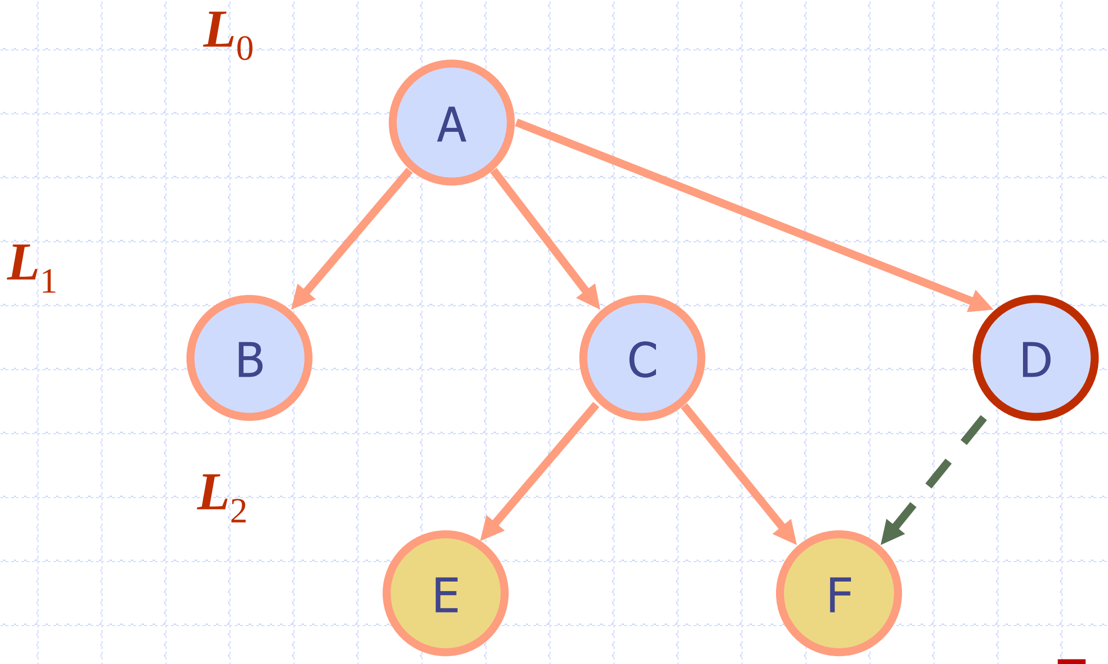
Example



A
B
C
D

Queue: D E F
Dequeue D → Queue: E F

Example

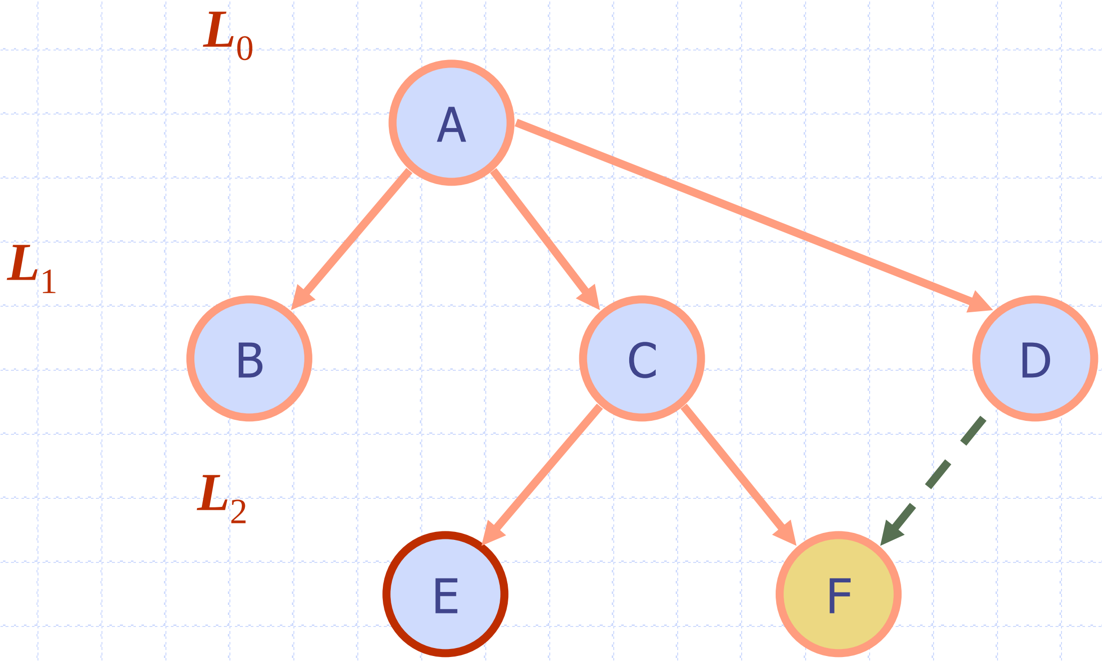


A
B
C
D

F is already visited!

Enqueue new neighbors of D.
Queue: E F

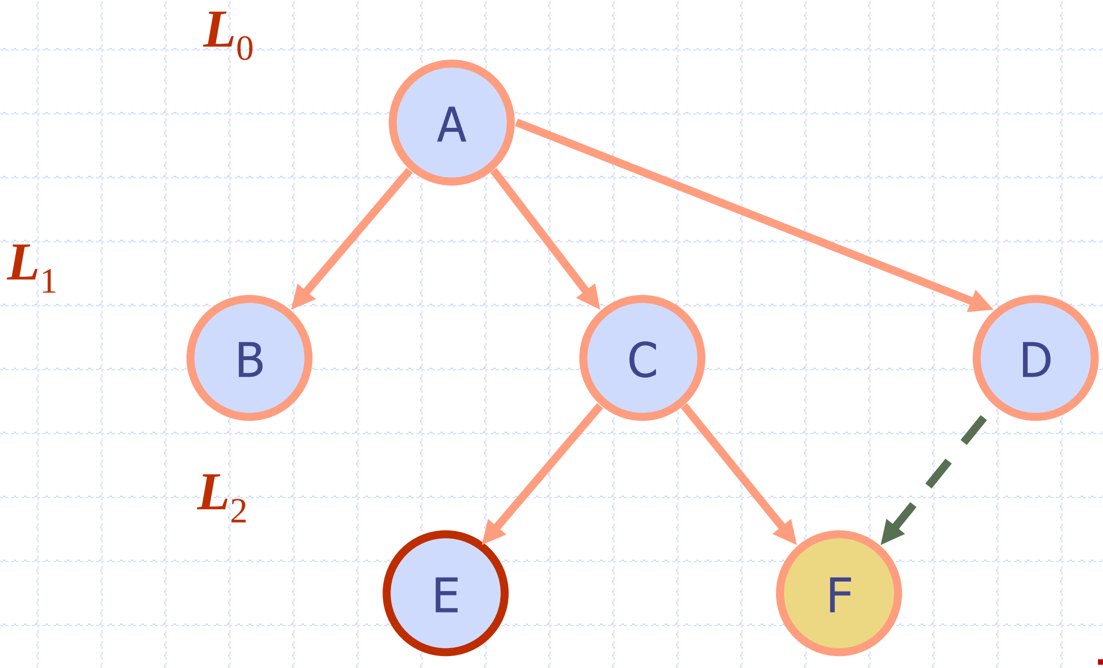
Example



A
B
C
D
E

Queue: E F
Dequeue E → Queue: F

Example

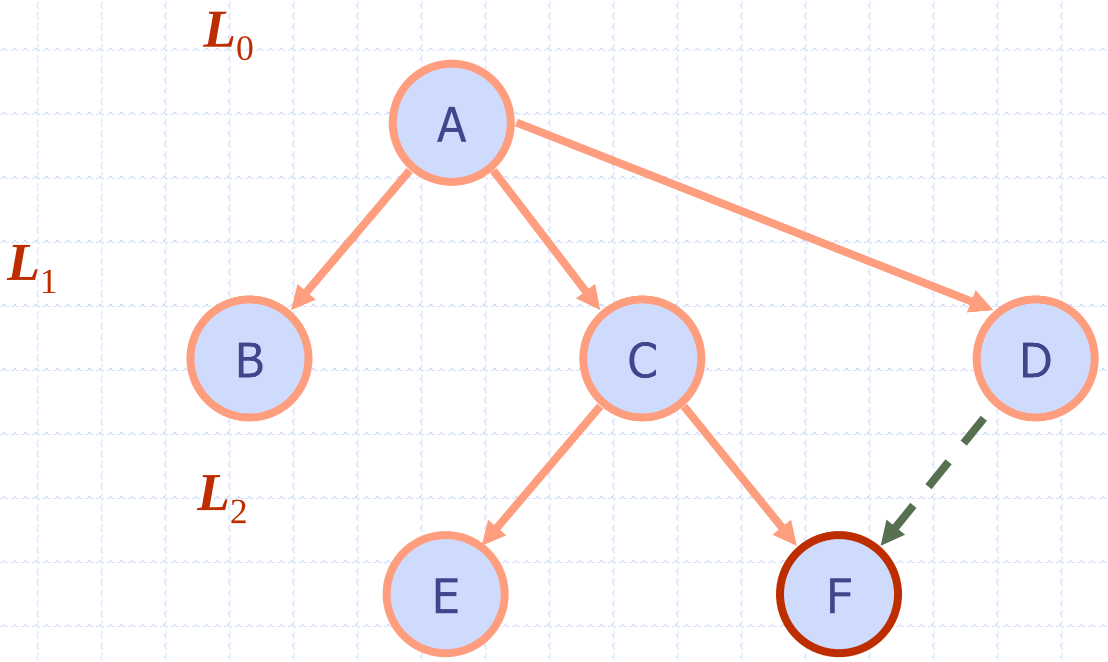


A
B
C
D
E

There are none!

Enqueue new neighbors of E.
Queue: F

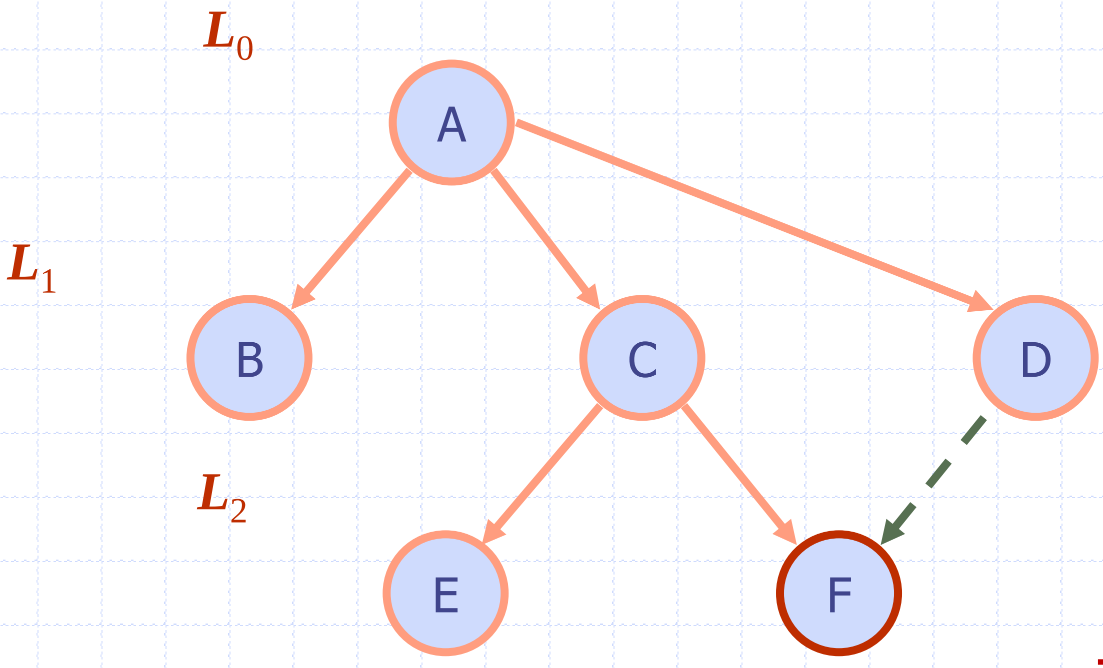
Example



A
B
C
D
E
F

Queue: F
Dequeue F → Queue:

Example

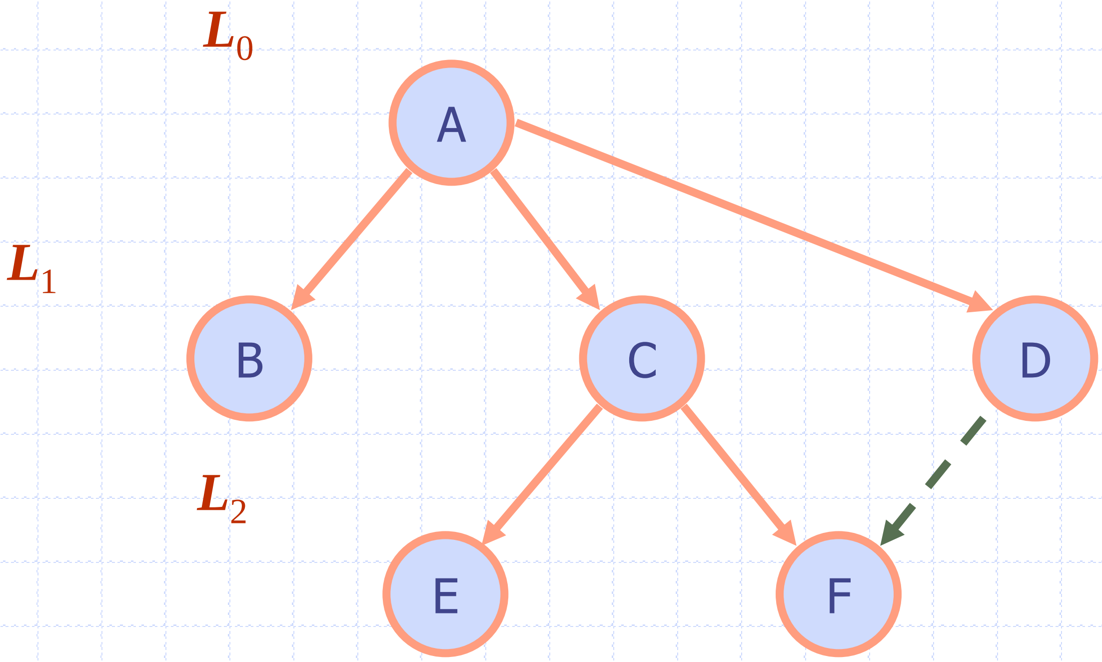


A
B
C
D
E
F

There are none!

Enqueue new neighbors of F.
Queue:

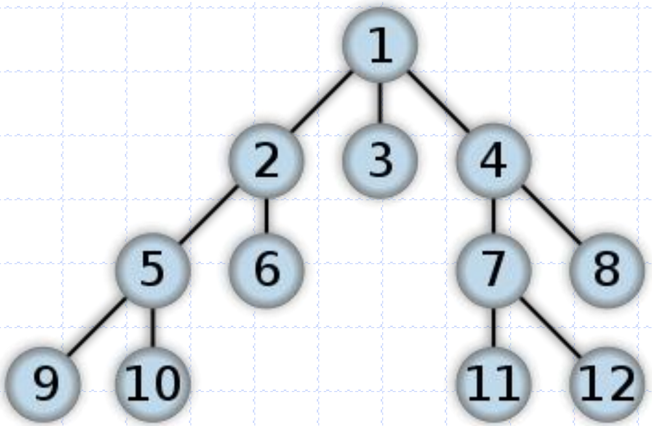
Example



A
B
C
D
E
F

Queue: The next step would be to dequeue.
But since the queue is empty, we're done!

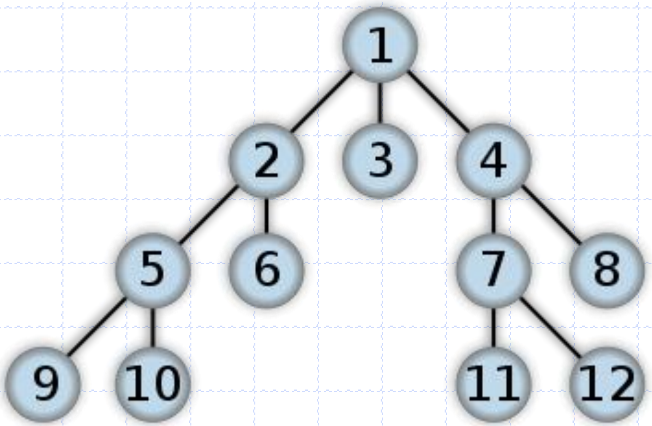
Now you try it!



Enqueue root 1.

Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:
Dequeue ____. Queue:

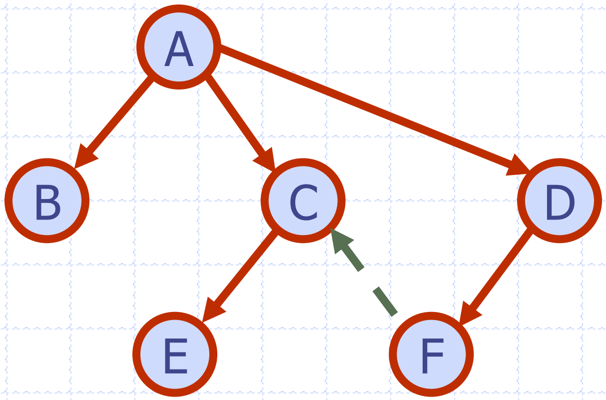
Now you try it!



Enqueue root 1.

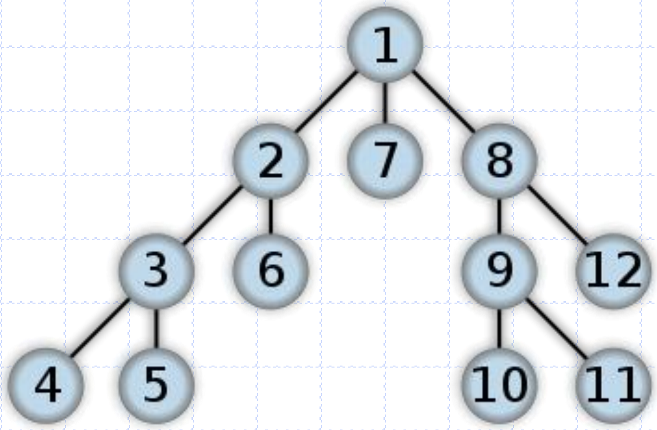
Dequeue __1_. Queue: 2 3 4
Dequeue __2_. Queue: 3 4 5 6
Dequeue __3_. Queue: 4 5 6
Dequeue __4_. Queue: 5 6 7 8
Dequeue __5_. Queue: 6 7 8 9 10
Dequeue __6_. Queue: 7 8 9 10
Dequeue __7_. Queue: 8 9 10 11 12
Dequeue __8_. Queue: 9 10 11 12
Dequeue __9_. Queue: 10 11 12
Dequeue _10_. Queue: 11 12
Dequeue _11_. Queue: 12
Dequeue _12_. Queue:

Depth-First Search

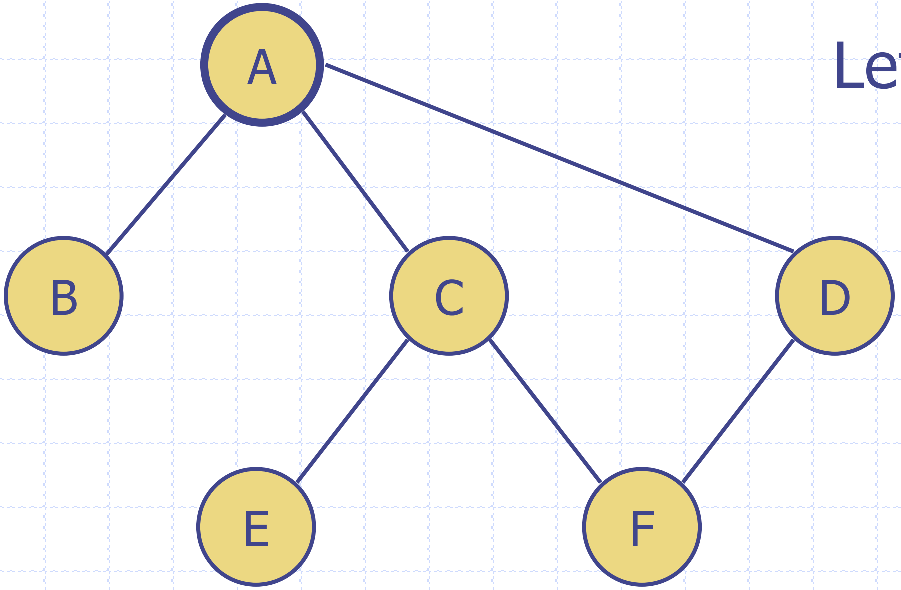


Depth-First Search

- Depth-first search (DFS) is a another technique for traversing a graph.
- A DFS traversal of a graph returns the nodes of the graph by traveling deep through one path until hitting a dead end and then retracing the steps.

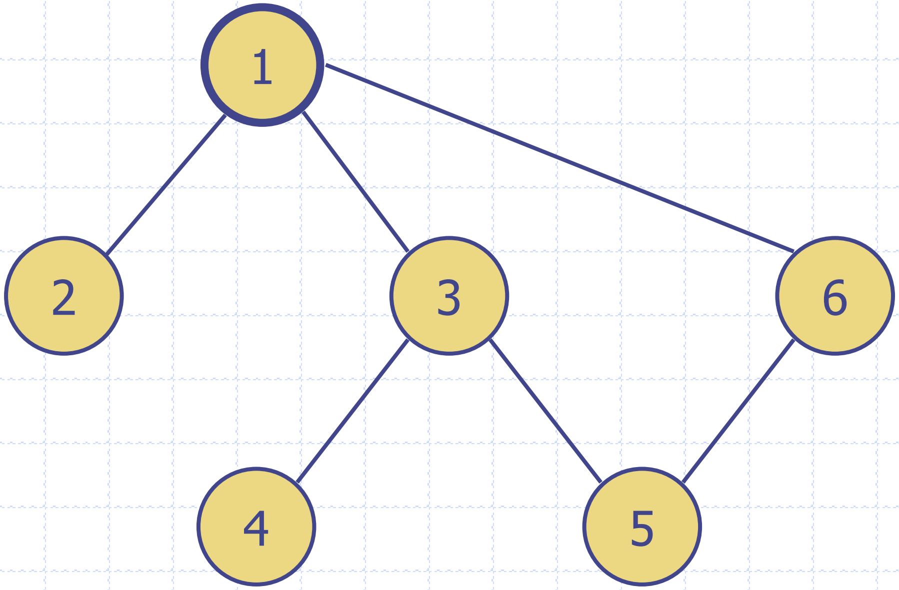


What is a DFS traversal?

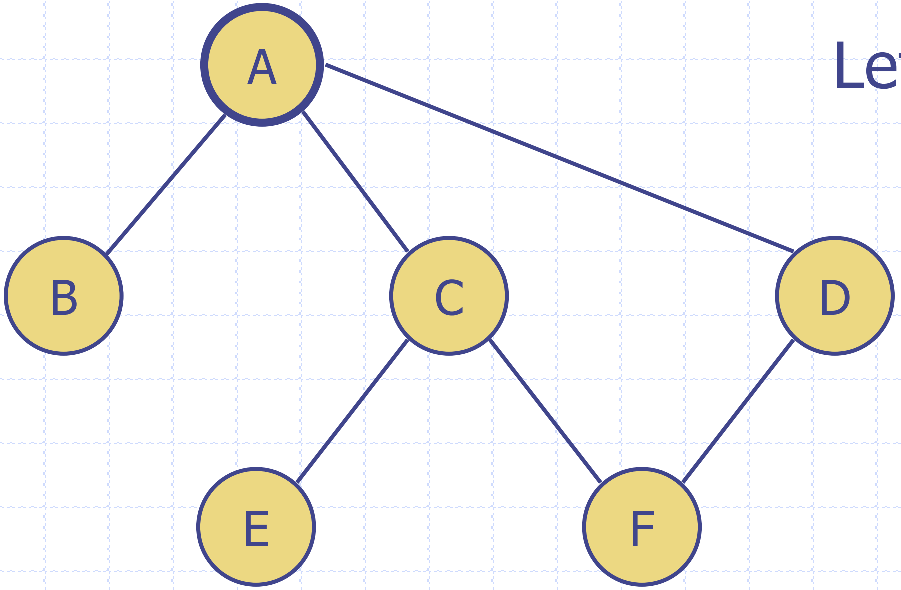


Let's start at A

What is a DFS traversal?



What is a DFS traversal?

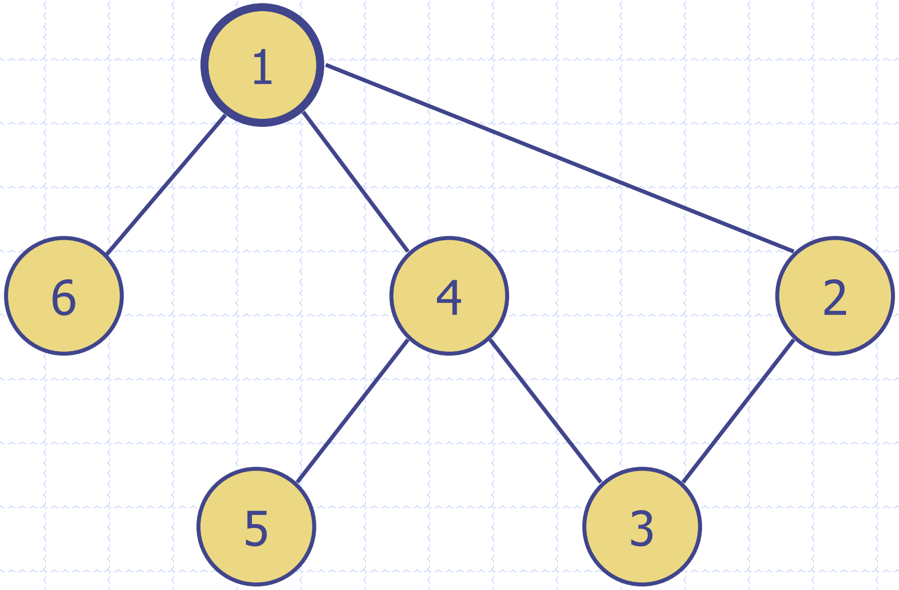


Let's start at A

A B C E F D

Is that the only DFS?

What is a DFS traversal?



Computer Algorithm

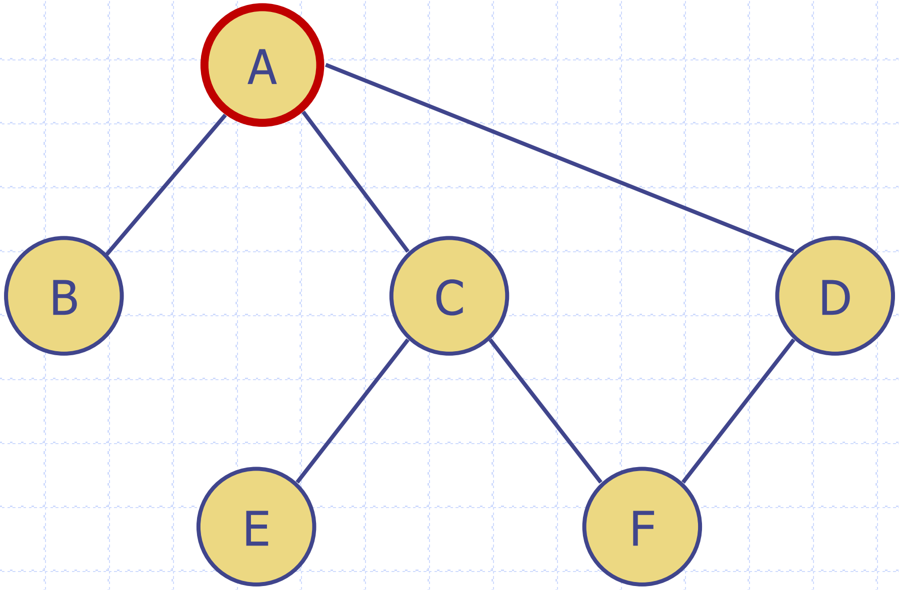
- Let’s look at how a computer would perform DFS!

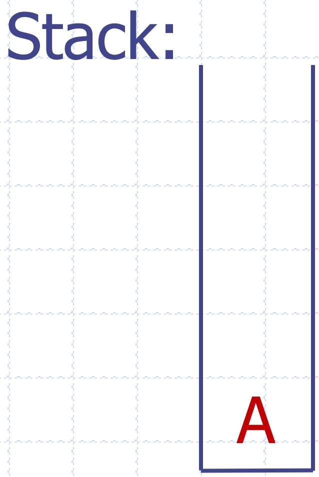
Stack

- A stack is a pile.
- If you put books in a pile, the last book will be on the top, and it will be the first one to be retrieved.

Last In, First Out

Example





Let's start at A!

Push the root A onto the stack.

Our Algorithm is the SAME!

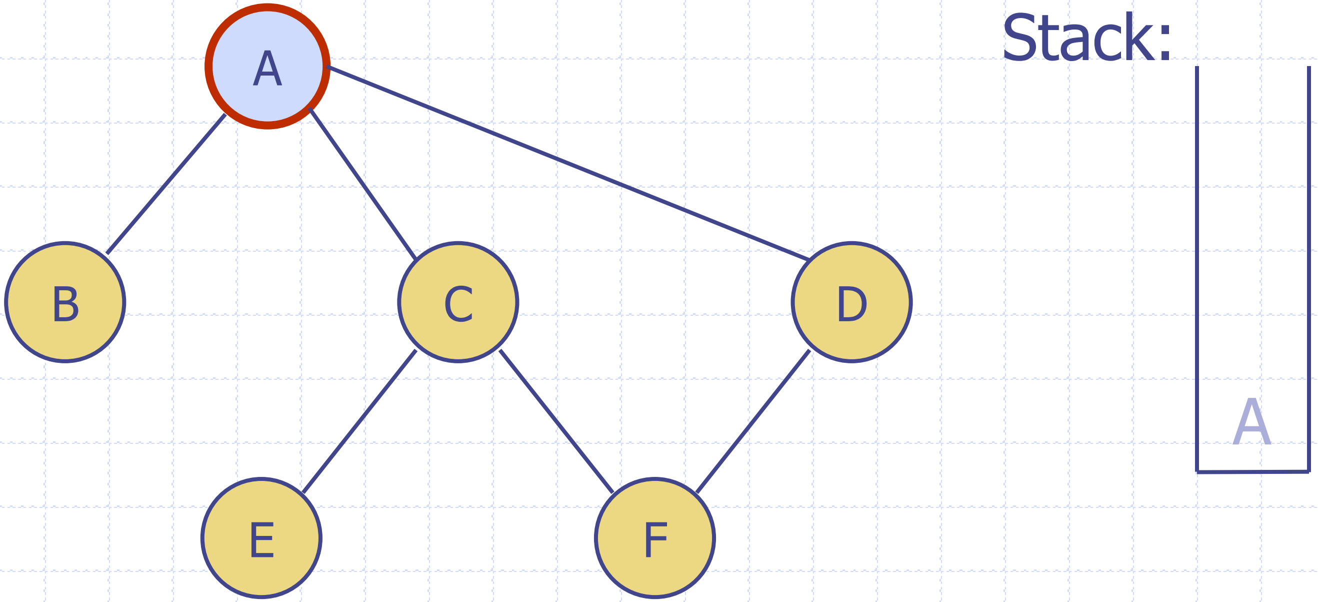
- *Except, this time, we're using a stack!*
- Each time, we're going pop a node (we'll call it node X) off the stack.
- Then, we're going to push all of the new neighbors of node X onto the stack.

First Step:

POP A NODE

OFF THE STACK!

Example



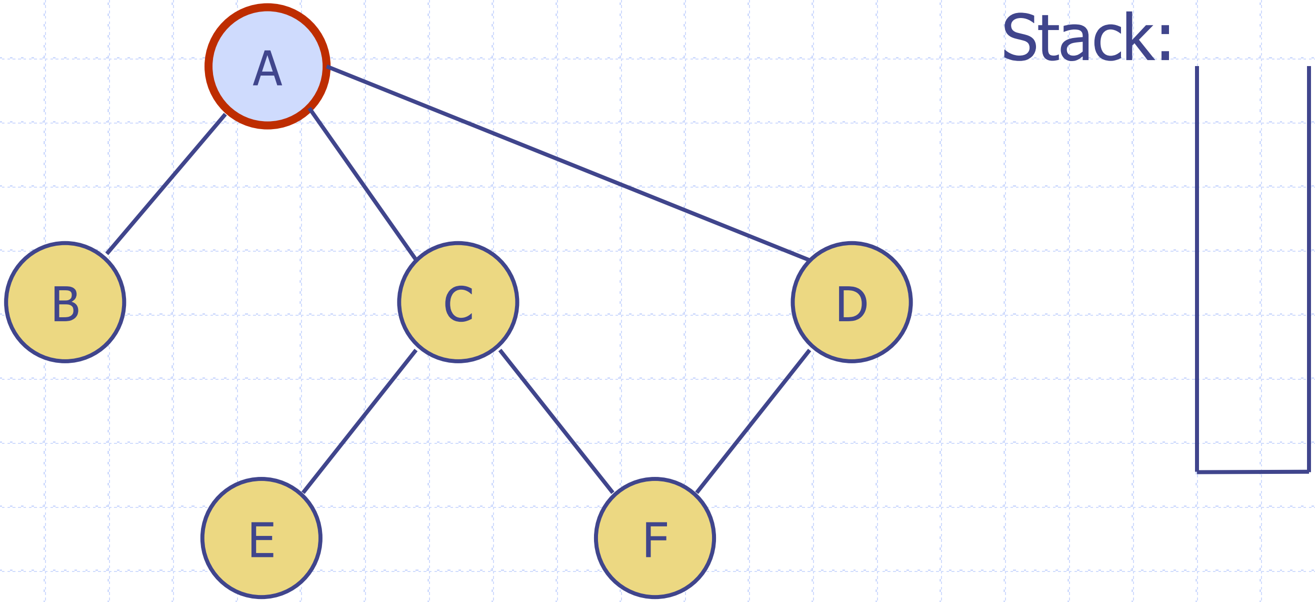
Pop a node off the stack.

A

Second Step:

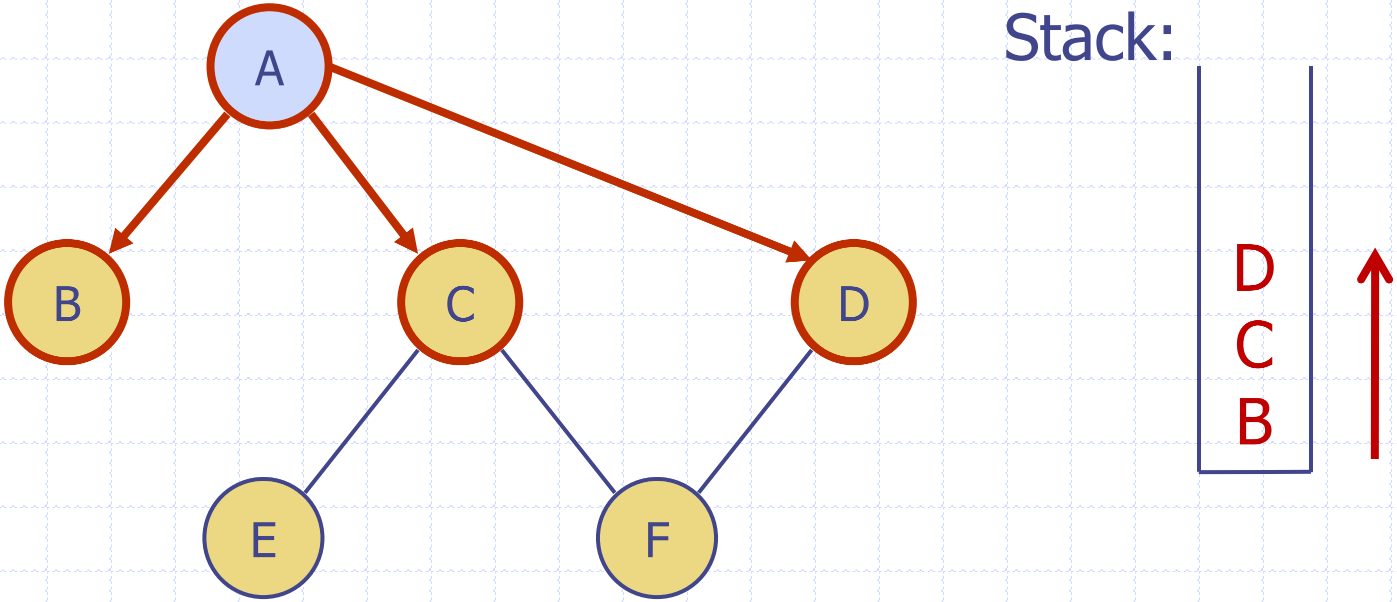
PUSH THE NEW NEIGHBORS ONTO THE STACK!

Example



What are the new neighbors of A?

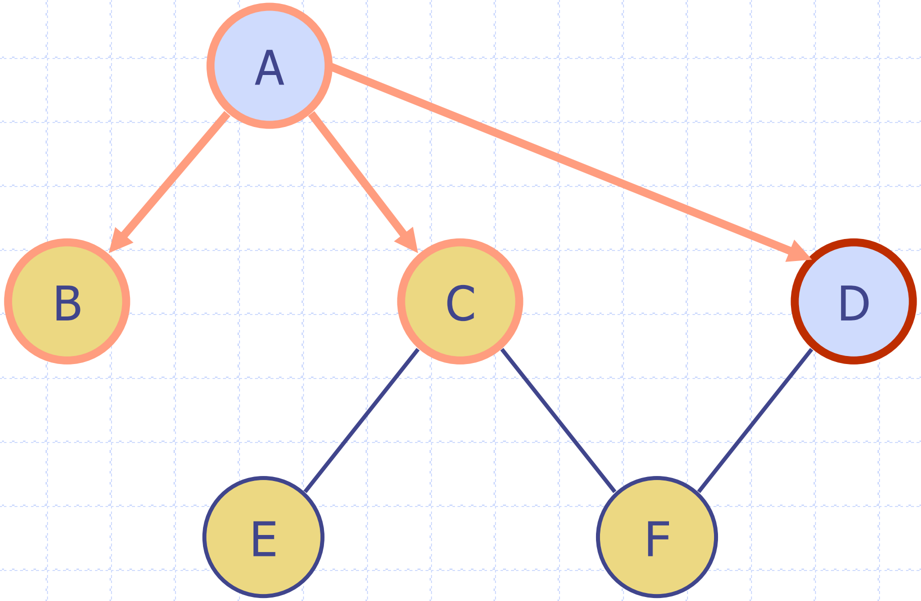
Example

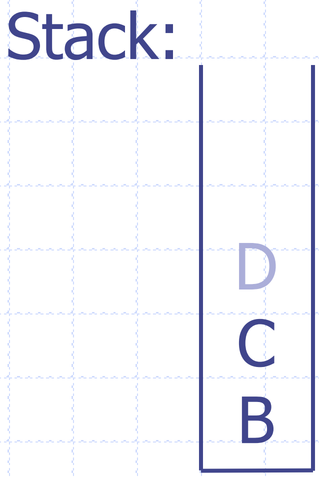


Push the new neighbors of root A onto the stack.

A

Example

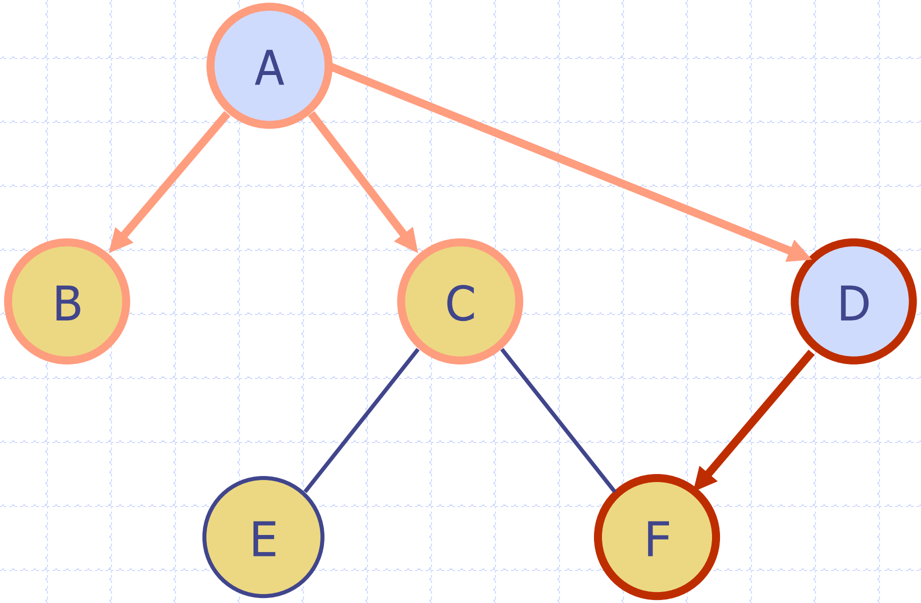




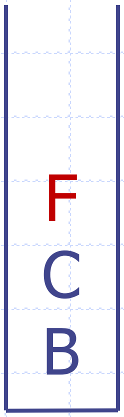
Pop a node off the stack.

A D

Example



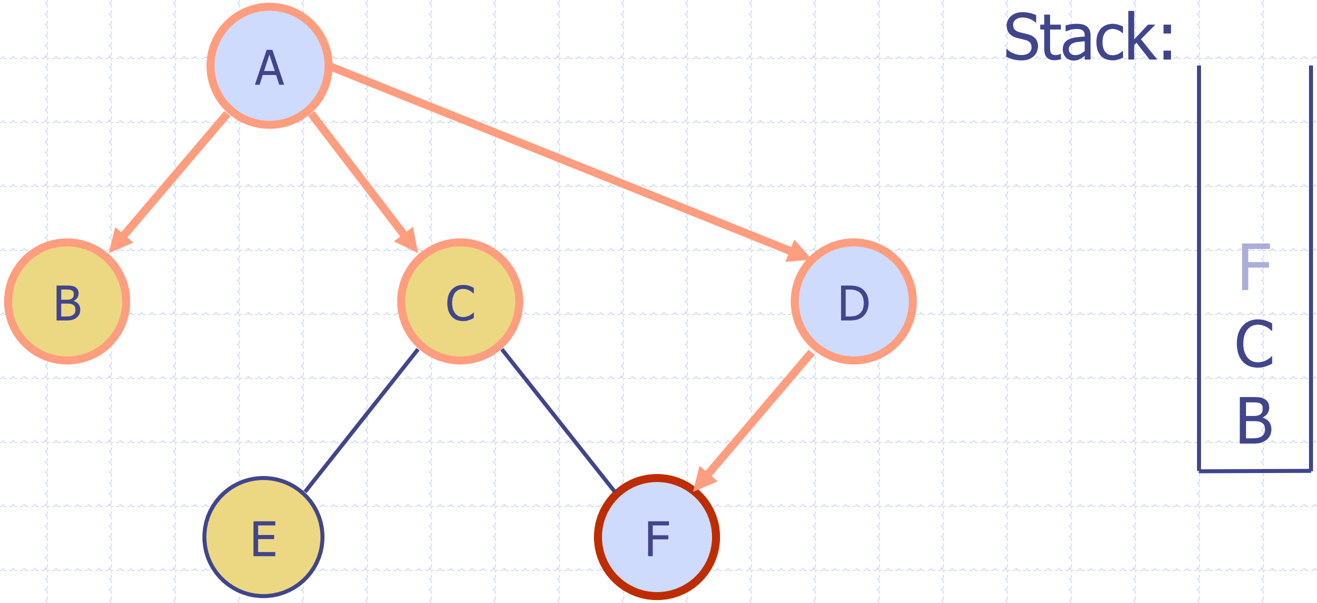
Stack:



Push the new neighbors of D onto the stack.

A D

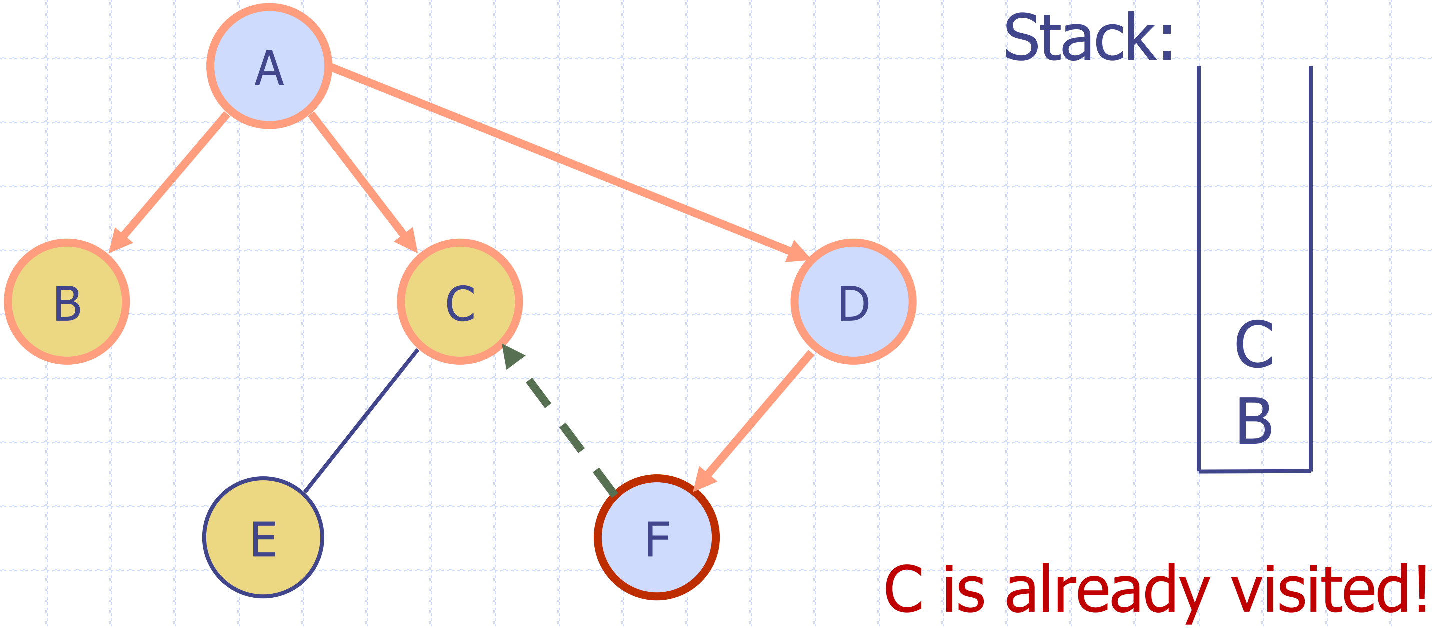
Example



Pop a node off the stack.

A D F

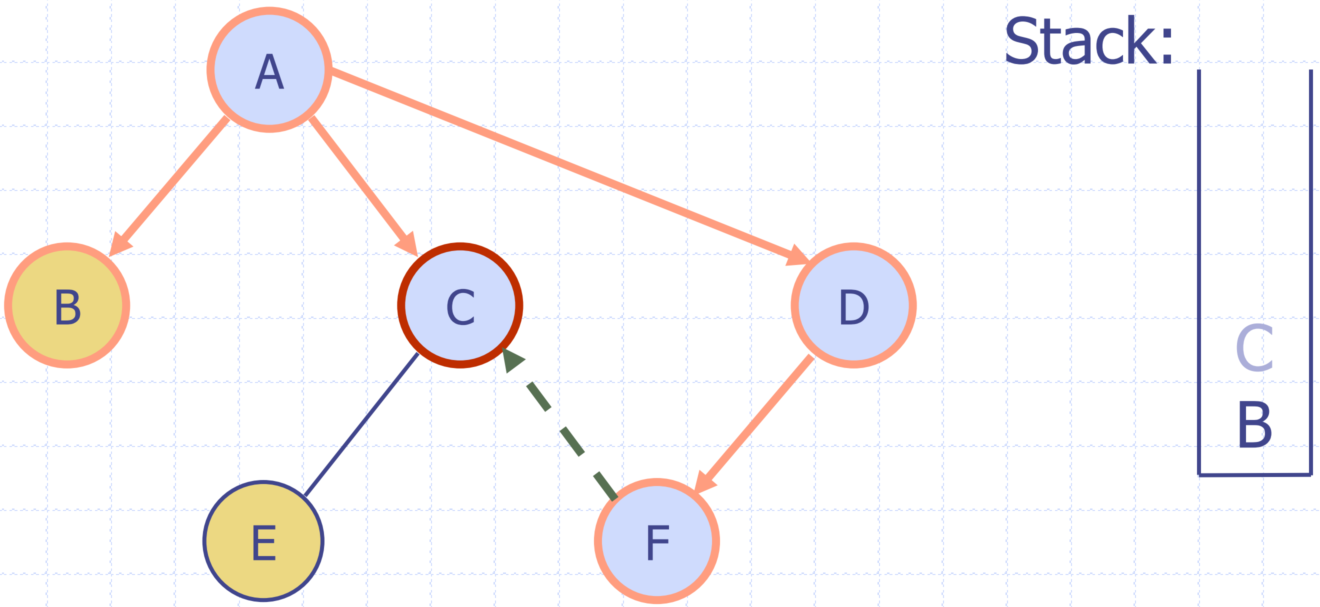
Example



Push the new neighbors of F onto the stack.

A D F

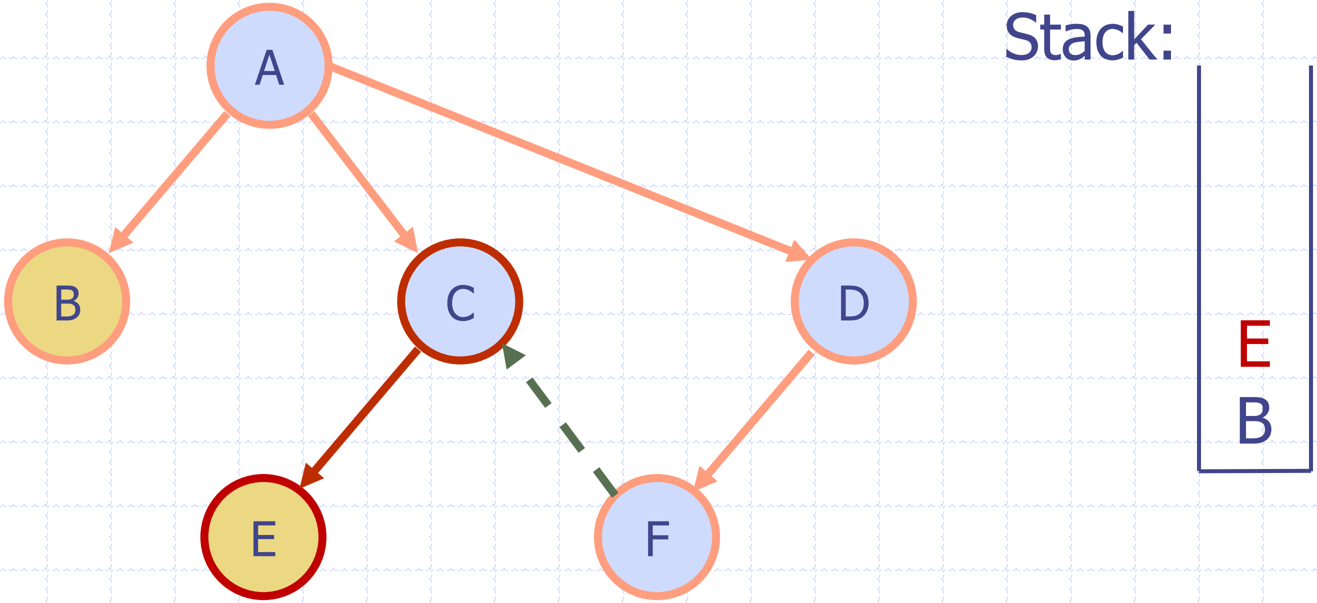
Example



Pop a node off the stack.

A D F C

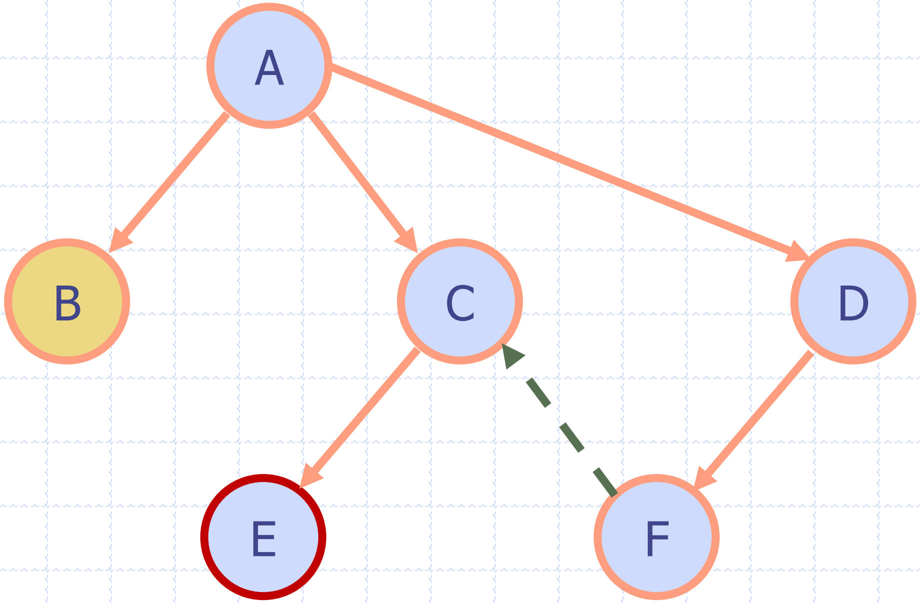
Example



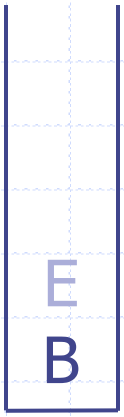
Push the new neighbors of C onto the stack.

A D F C

Example



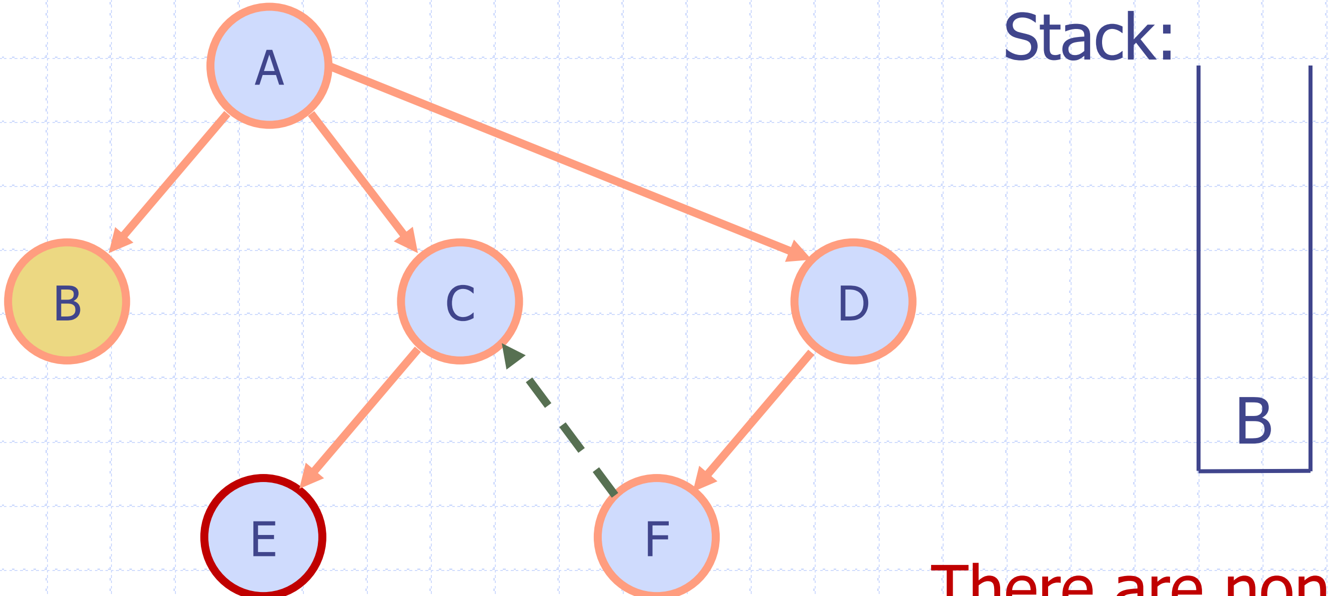
Stack:



Pop a node off the stack.

A D F C E

Example

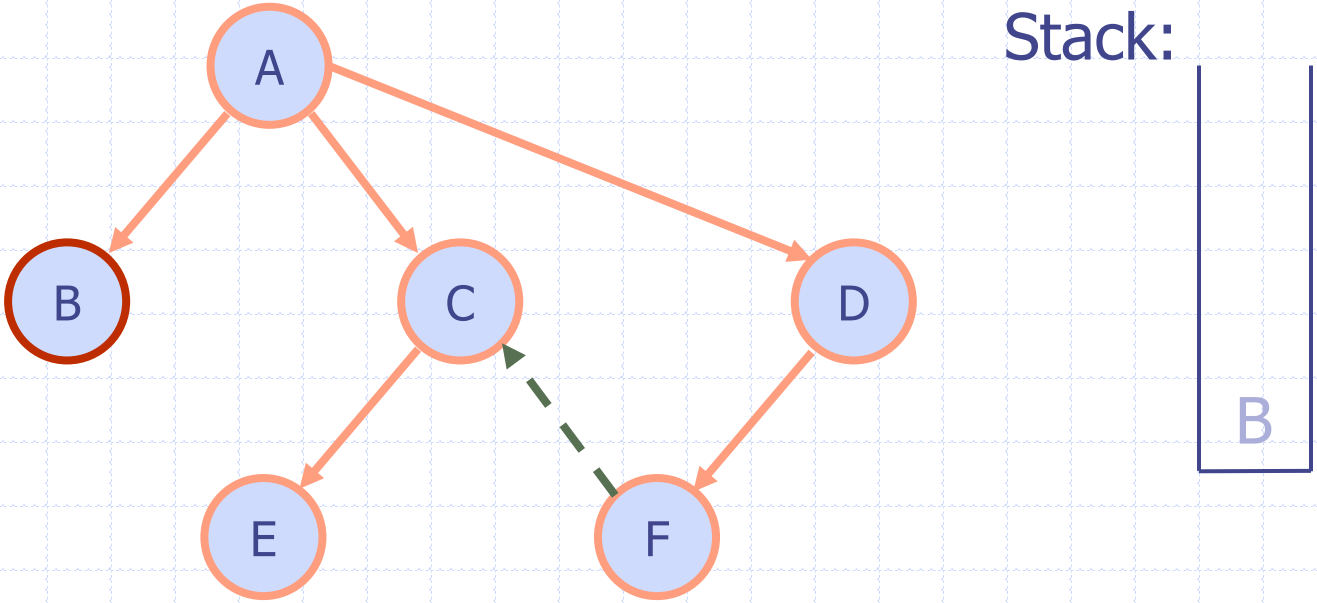


There are none!

Push the new neighbors of E onto the stack.

A D F C E

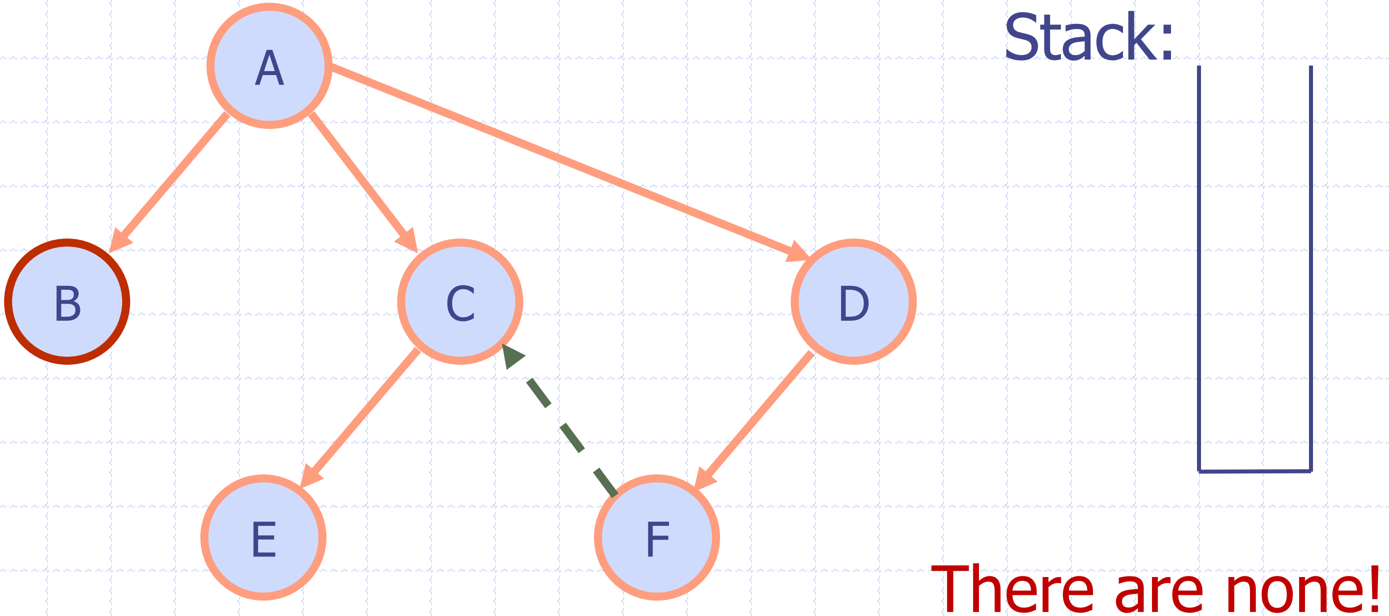
Example



Pop a node off the stack.

A D F C E B

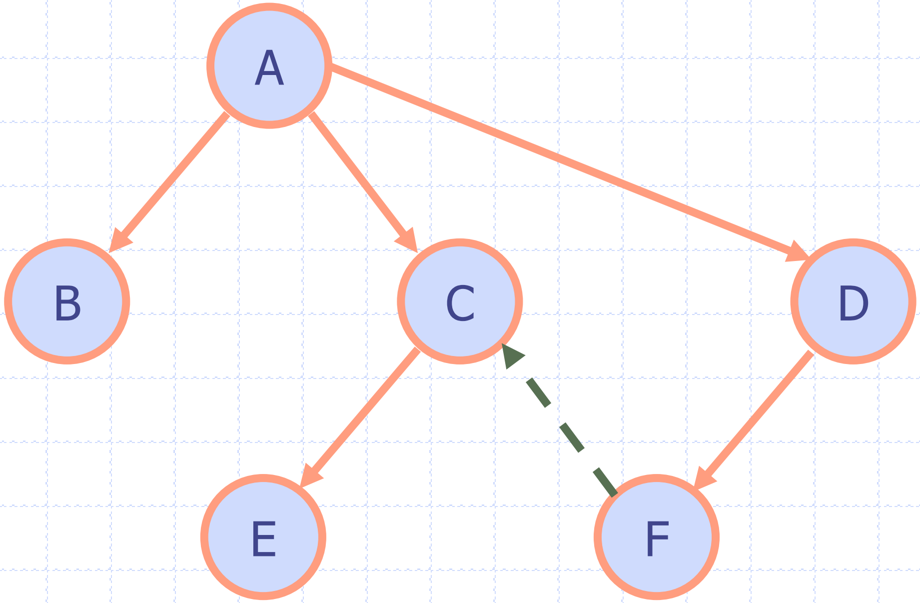
Example



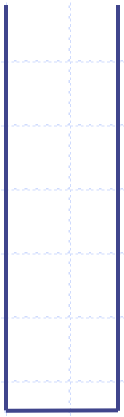
Push the new neighbors of B onto the stack.

A D F C E B

Example



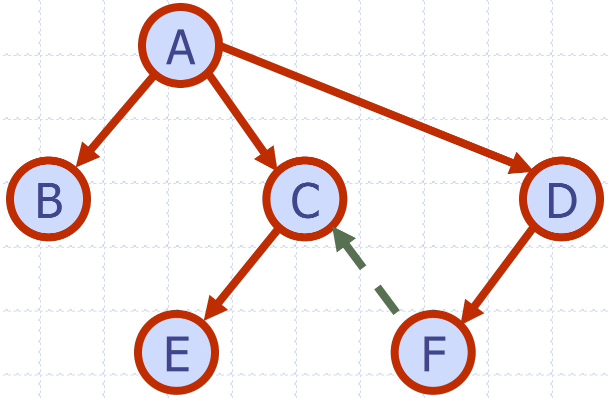
Stack:



The next step would be to pop a node off the stack.

A D F C E B   But since the stack is empty, we're done!

Example



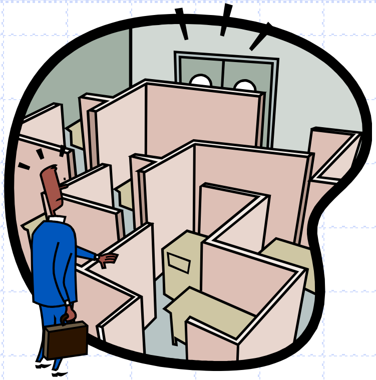
Did you find it strange that although we pushed in the neighbors from left to right (e.g. B, C, D), the order of the search that we got back starts on the right?

A D F C E B

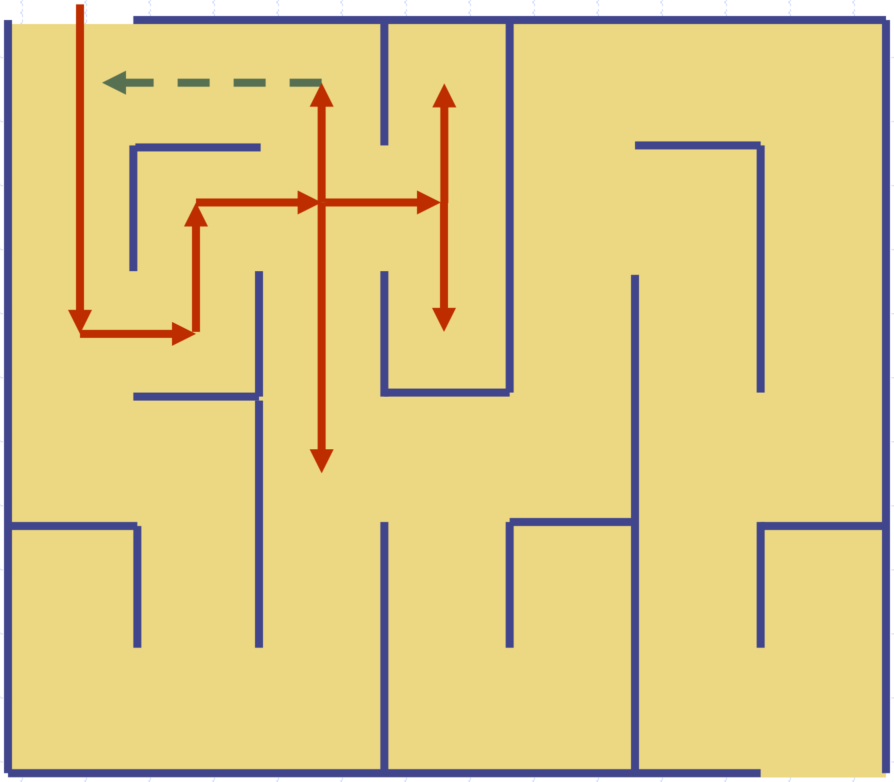
That's because we used a stack! Last in first out, remember?

If we want to search to end up A B C E F D, all we have to do is to push the neighbors into the stack from right to left (e.g. D, C, B).

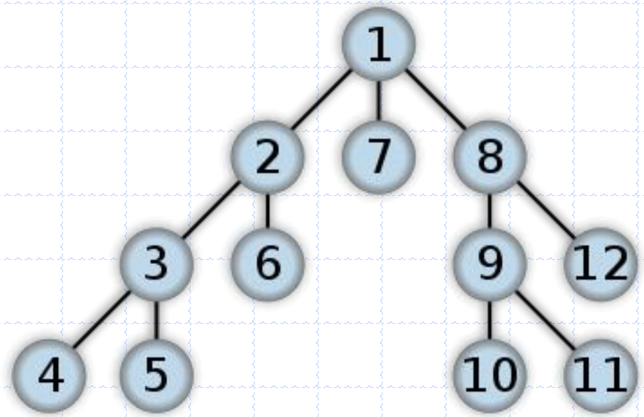
DFS and Maze Traversal



- The DFS algorithm is similar to a classic strategy for exploring a maze.
  - We go as far as possible on one path until we reach a dead end.
  - Then, we retrace our steps and go somewhere we haven't visited before until we reach a dead end.



Now you try it!



Stack 1

Stack

Pop off ____.

Stack

Pop off ____.

Stack

Pop off ____.

Stack

Pop off ____.

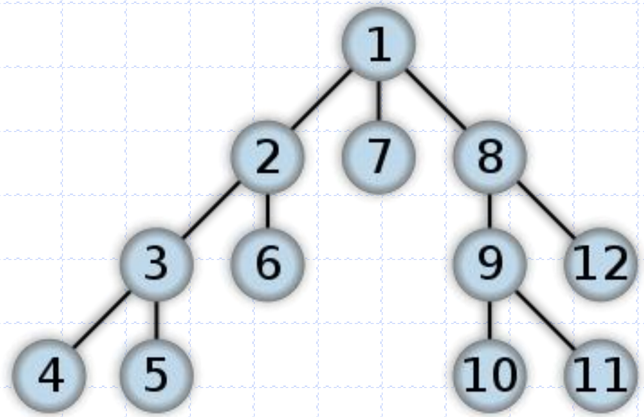
Stack

Pop off ____.

Stack

Pop off ____.

Now you try it!



Stack

← *From previous slide!*

Stack

Pop off ____.

Stack

Pop off ____.

Stack

Pop off ____.

Stack

Pop off ____.

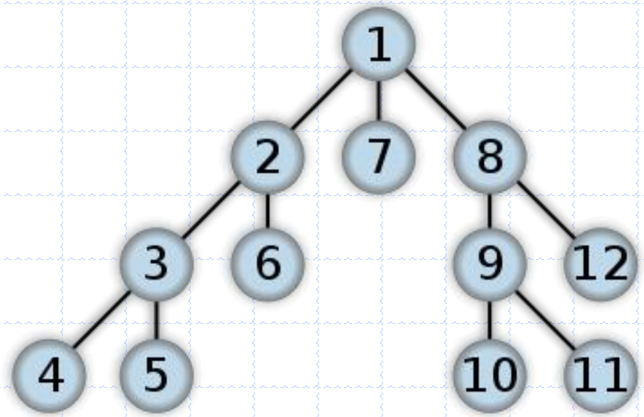
Stack

Pop off ____.

Stack

Pop off ____.

Now you try it!



Stack
1

Stack
2
7
8

Pop off __1__.

Stack
3
6
7
8

Pop off __2__.

Stack
4
5
6
7
8

Pop off __3__.

Stack
5
6
7
8

Pop off __4__.

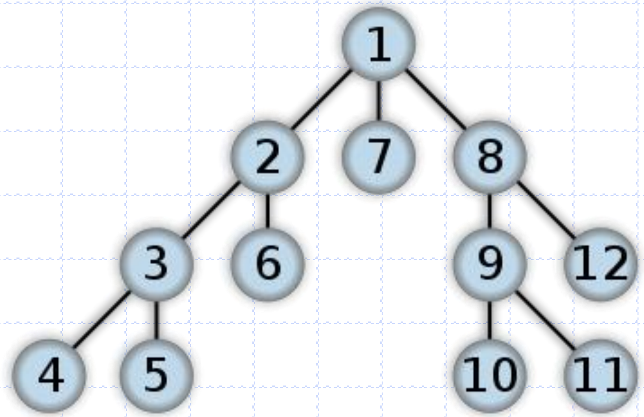
Stack
6
7
8

Pop off __5__.

Stack
7
8

Pop off __6__.

Now you try it!



Stack

7
8

From previous slide!

Stack | 8

Pop off __7__.

Stack | 9 12

Pop off __8__.

Stack | 10 11 12

Pop off __9__.

Stack | 11 12

Pop off __10__.

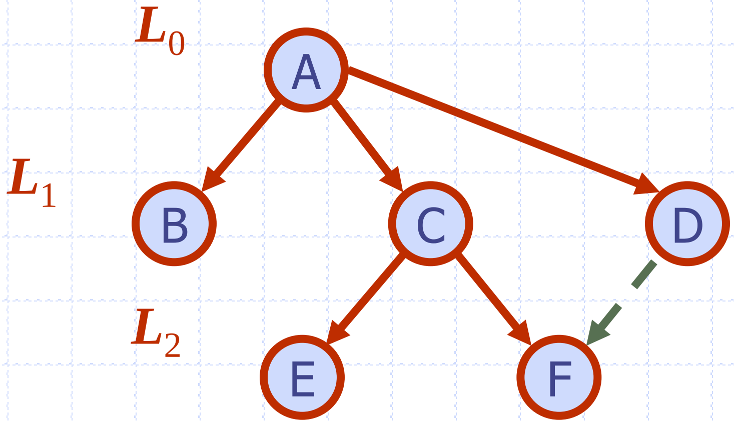
Stack | 12

Pop off __11__.

Stack |

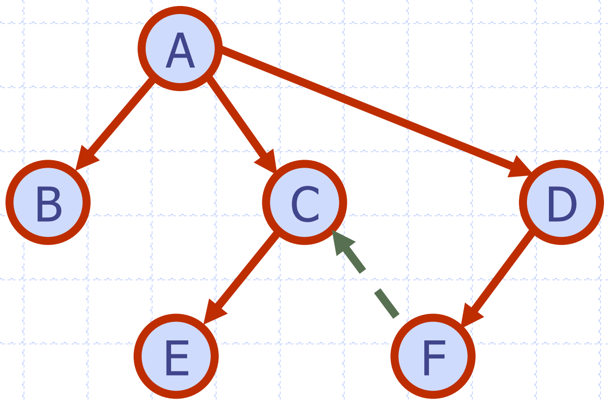
Pop off __12__.

BFS vs. DFS



BFS

A B C D E F



DFS

A D F C E B

Applications

- BFS
  - Finding the shortest path
  - Testing for bipartiteness
  - Bipartite graphs are useful for modeling matching problems
- DFS
  - Maze Problem

Bipartite Graphs