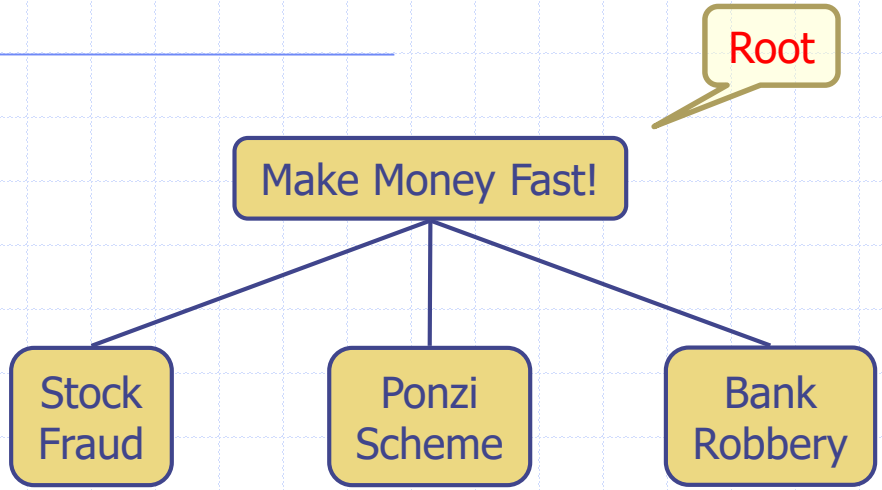
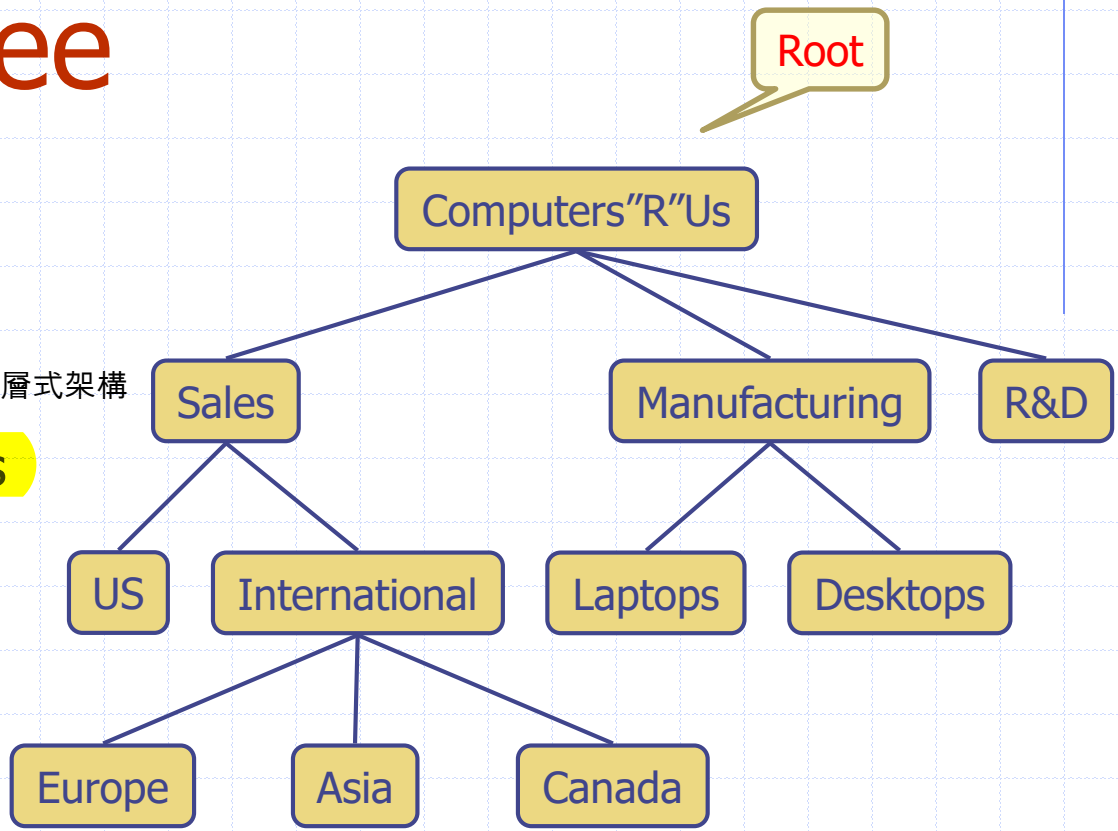
General Trees

Root



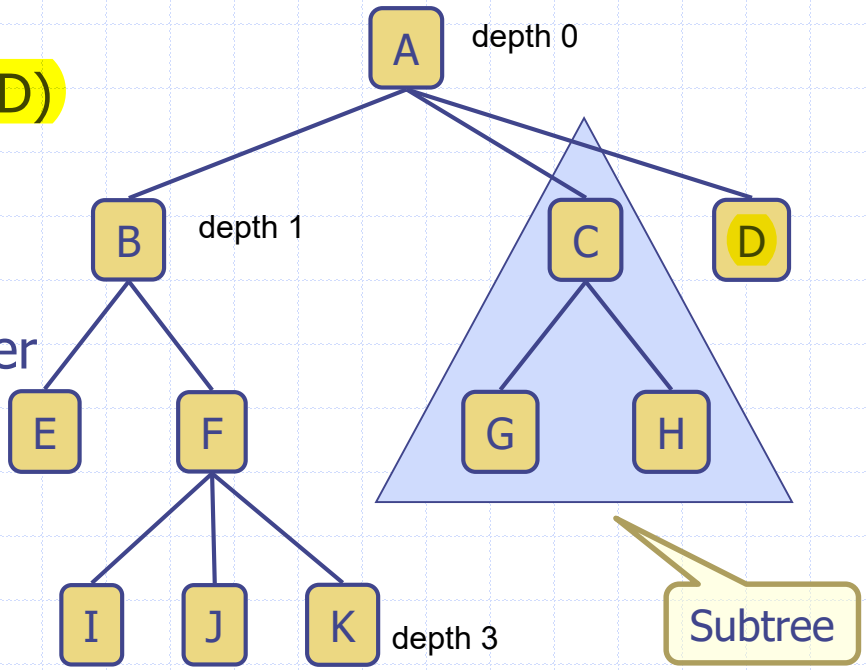
What is a Tree

- In CS, a tree is an abstract model of a hierarchical structure 階層式架構
- A tree consists of nodes with a parent-child relation
- Applications:
  - Organization charts
  - File systems
  - Function invocation in programming



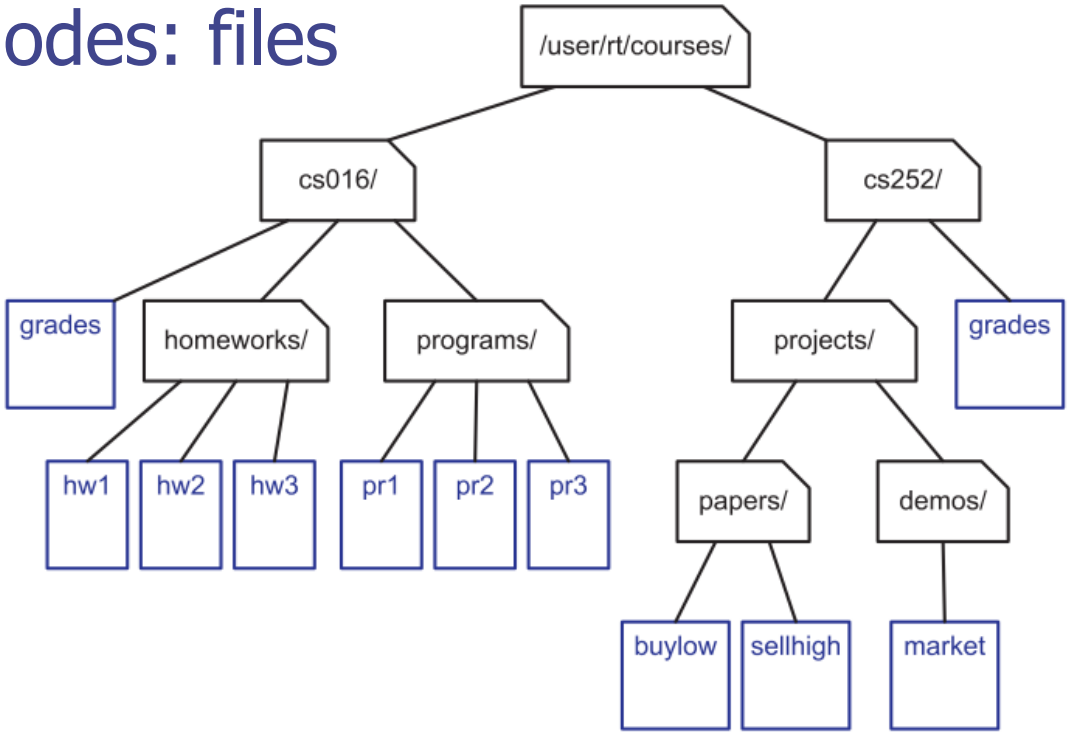
Tree Terminologies

- Root: node without parent (A)
- Internal node: node with at least one child (A, B, C, F)
- External node (a.k.a. leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Depth (or level) of a node: number of ancestors
- Height of a tree: maximum depth of any node
- Descendant of a node: child, grandchild, grand-grandchild, etc.
- Subtree: tree consisting of a node and its descendants



Examples of Trees

- File system
  - Internal nodes: directories (folders)
  - External nodes: files



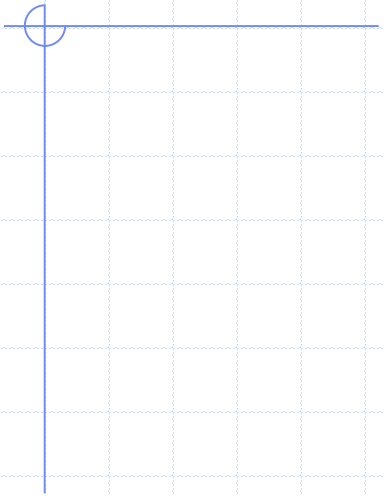
Tree ADT

- We use positions to access nodes
- Generic methods:
  - integer size()
  - boolean empty()
- Accessor methods:
  - position root()
  - list<position> positions()
- Position-based methods:
  - position p.parent()
  - list<position> p.children()

position = pointer

list = linked list or vector

- Query methods:
  - boolean p.isRoot()
  - boolean p.isExternal()
- Additional methods may be defined for specific applications



root first

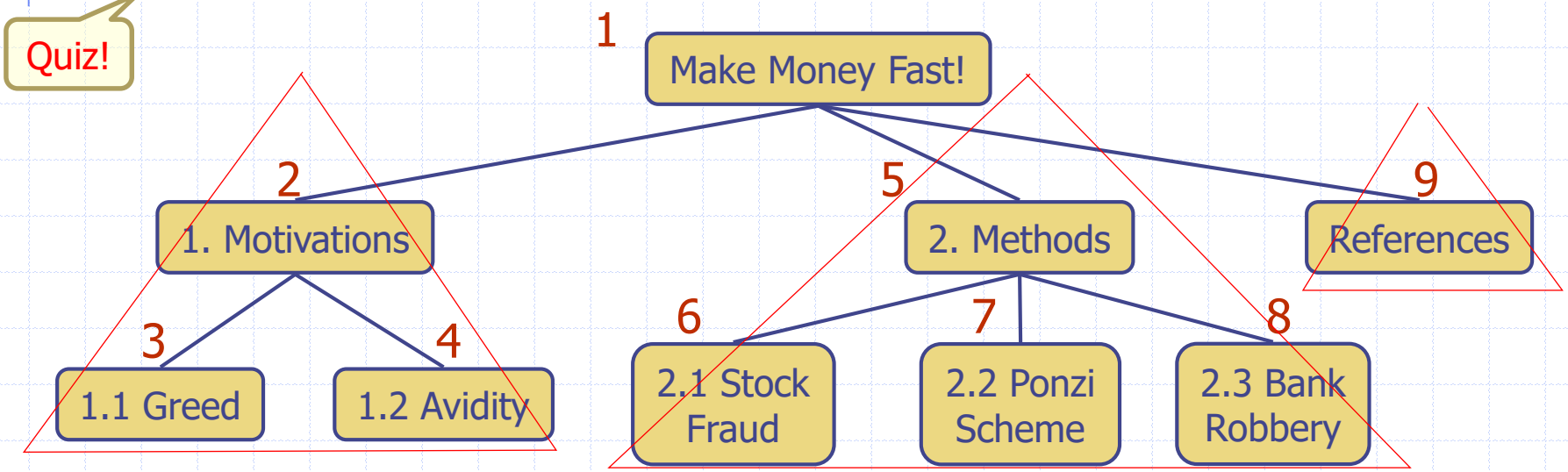
Preorder Traversal

root

- A traversal visits the nodes of a tree in a systematic manner
- Preorder traversal: a node is visited before its descendants
- Application: print a structured document

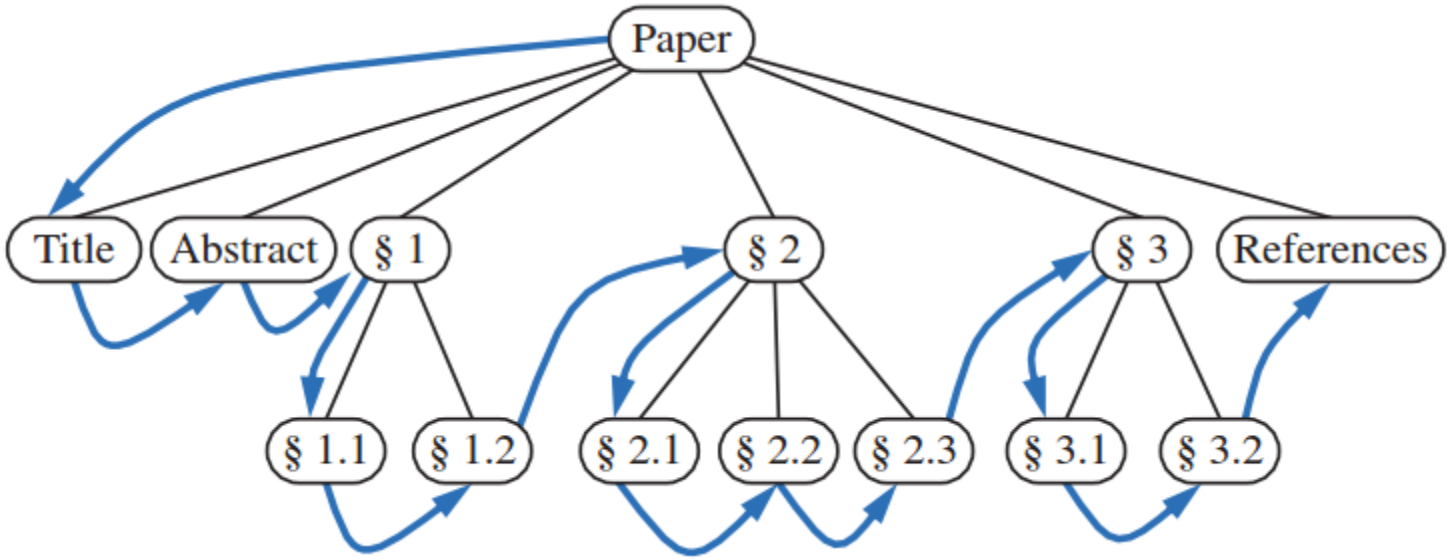
Algorithm ***preOrder(v)***
  visit(v)
  for each child ***w*** of ***v***
    preOrder ***(w)***

Quiz!



Preorder Traversal: Example

- How to print this document?

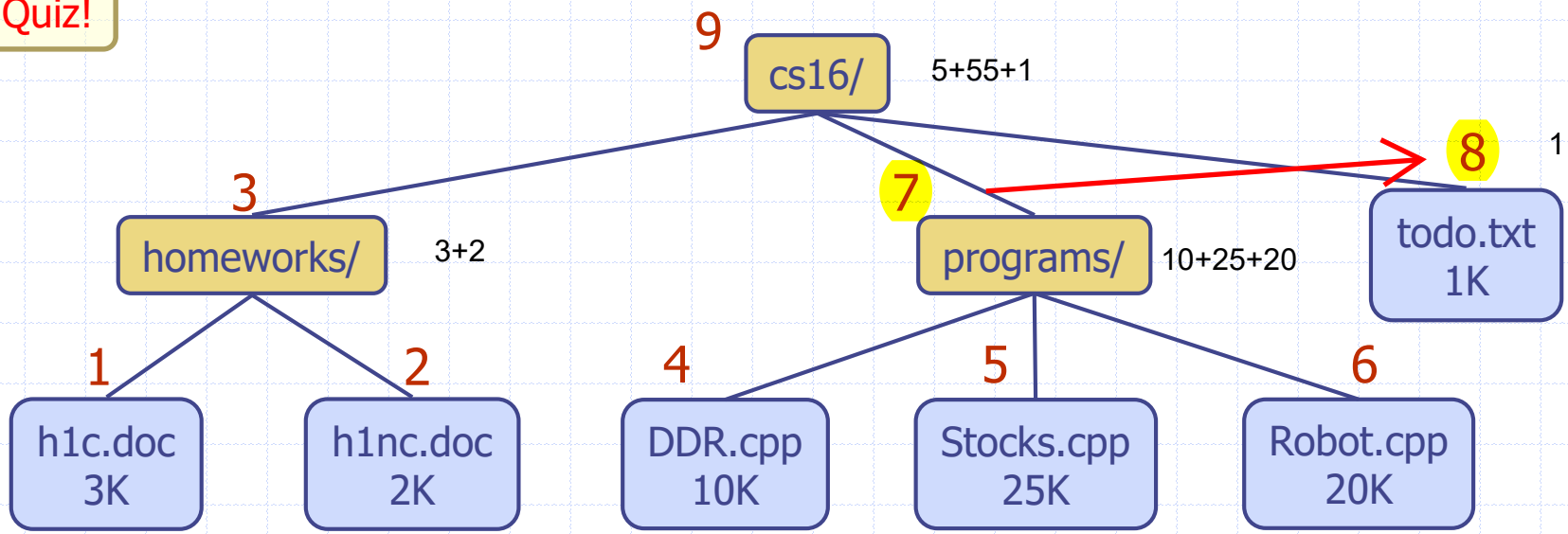


Postorder Traversal

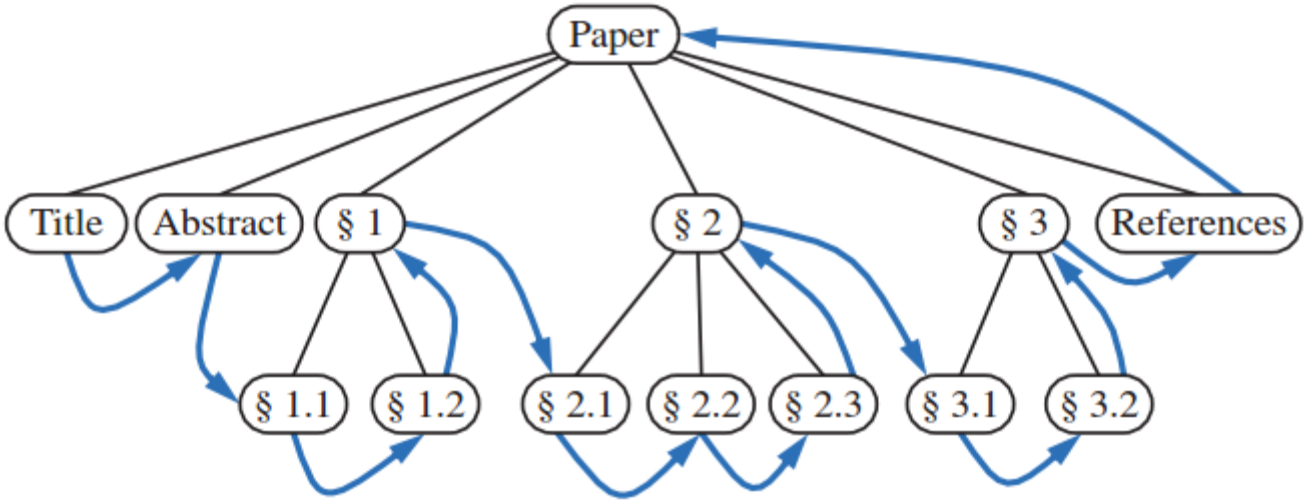
- Postorder traversal: a node is visited after its descendants
- Application: compute space used by files in a directory and its subdirectories

Algorithm *postOrder*(*v*)
  for each child *w* of *v*
    postOrder (*w*)
  visit(*v*)

Quiz!

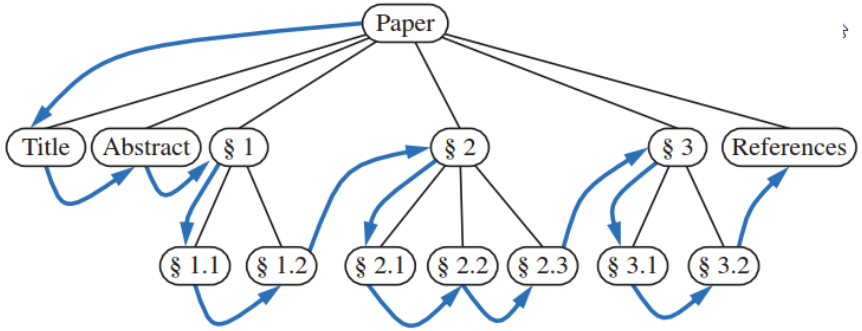


Postorder Traversal: Example

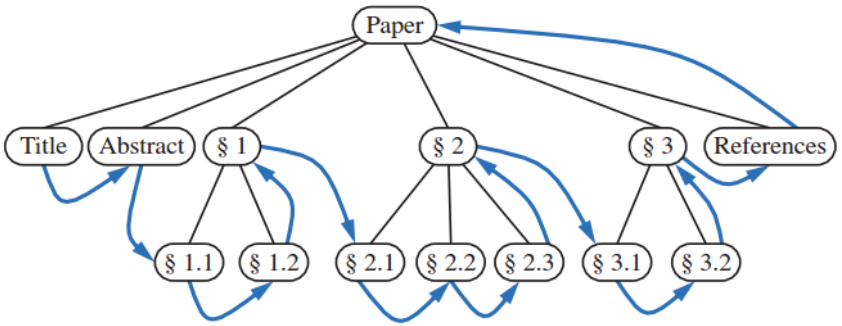


Comparison

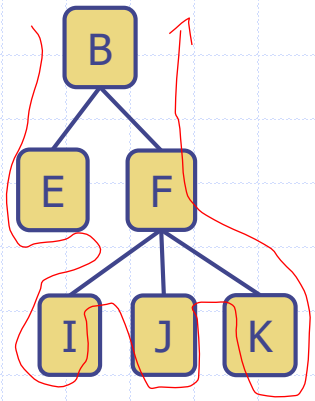
- Preorder



- Postorder



- Euler Tour Traversal
  - Preorder: Visit a node at the first sight
  - Postorder: Visit a node at the last sight



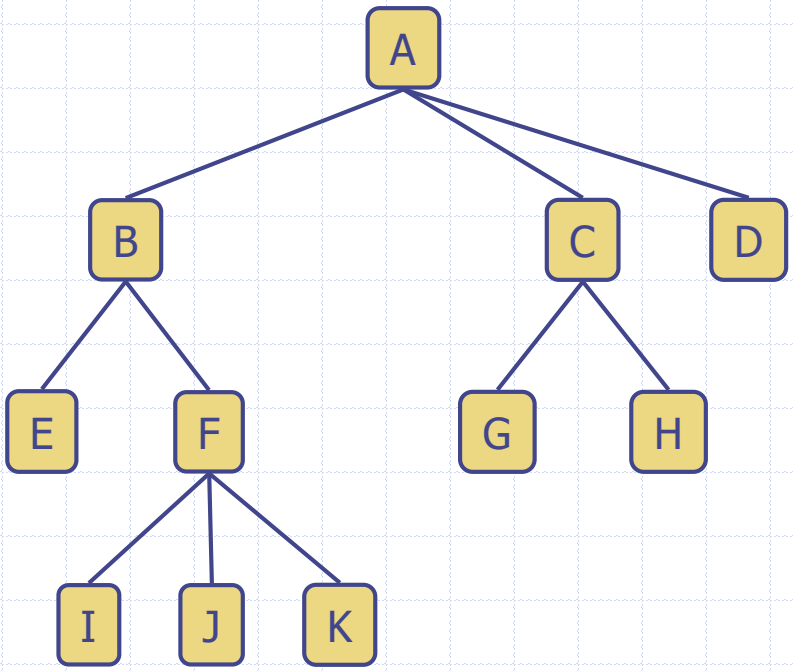
Iterative Implementation of Preorder Traversal

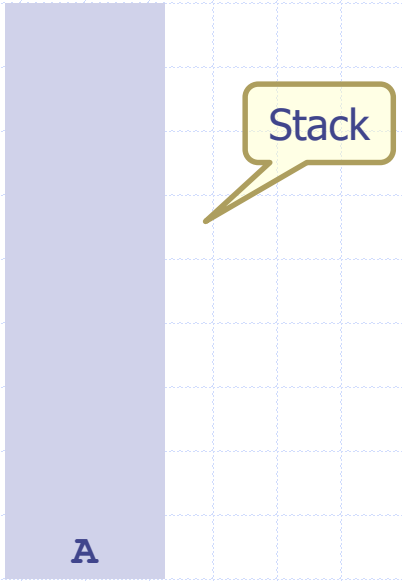
- Steps

See the animation in the subsequent slides!

1. Push the root to the stack
2. Pop the stack and visit it
3. Push the children in a reverse order (via the use of another stack)
4. Repeat 2) and 3) until the stack is empty

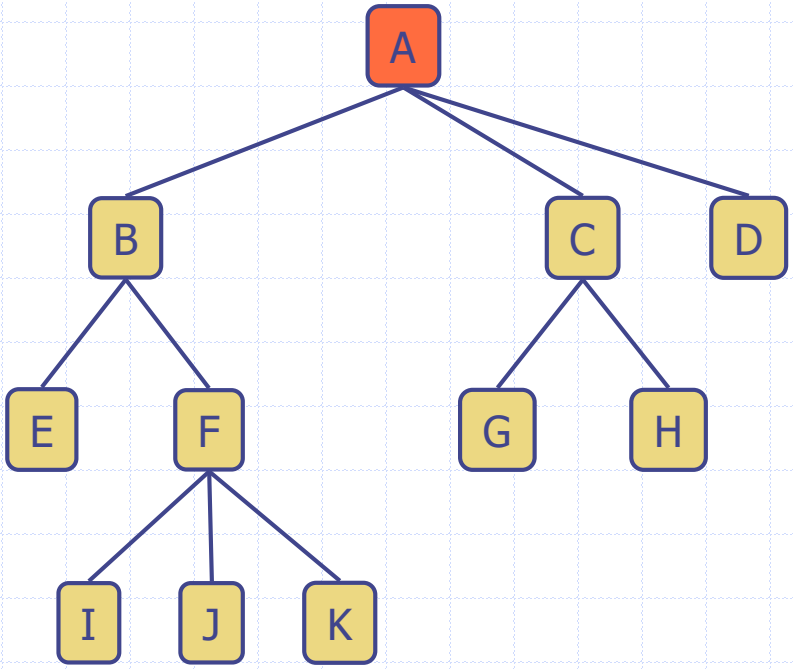
Iterative Implementation of Preorder Traversal: Demo

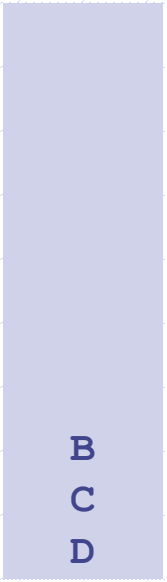




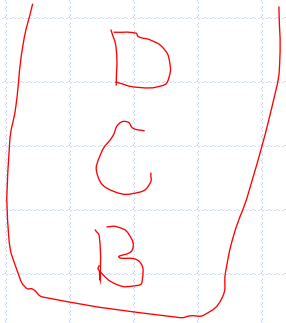
Preorder:

Iterative Implementation of Preorder Traversal: Demo



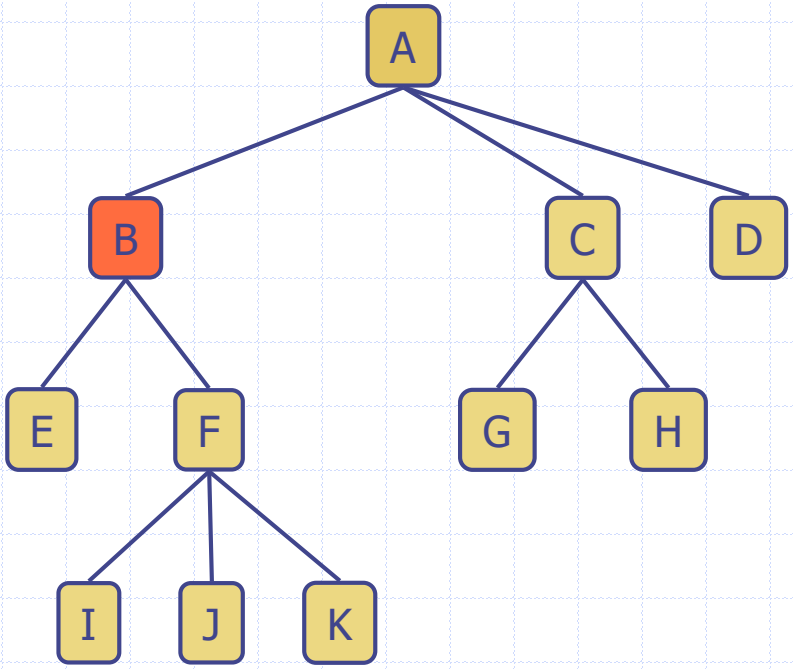


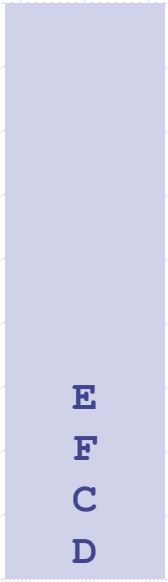
By using another stack



Preorder: A

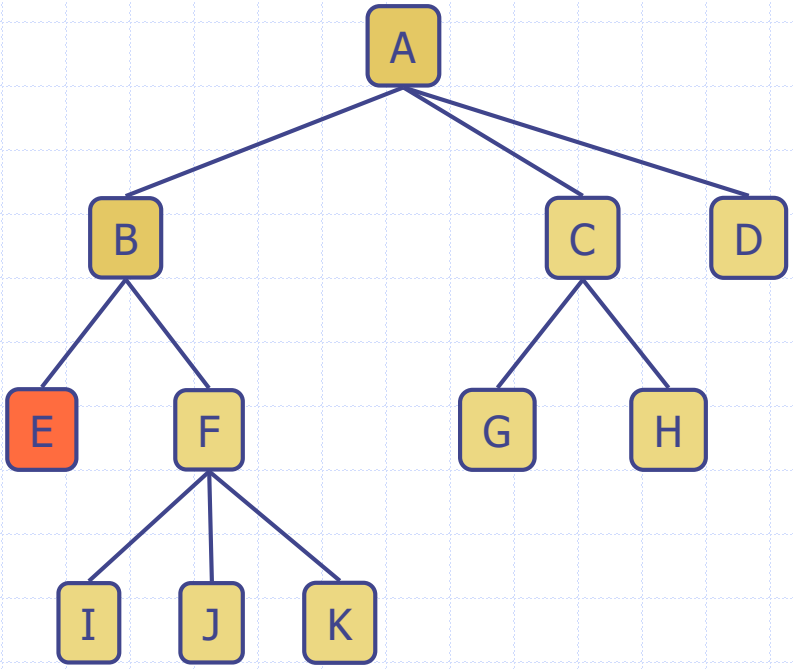
Iterative Implementation of Preorder Traversal: Demo





Preorder: A B

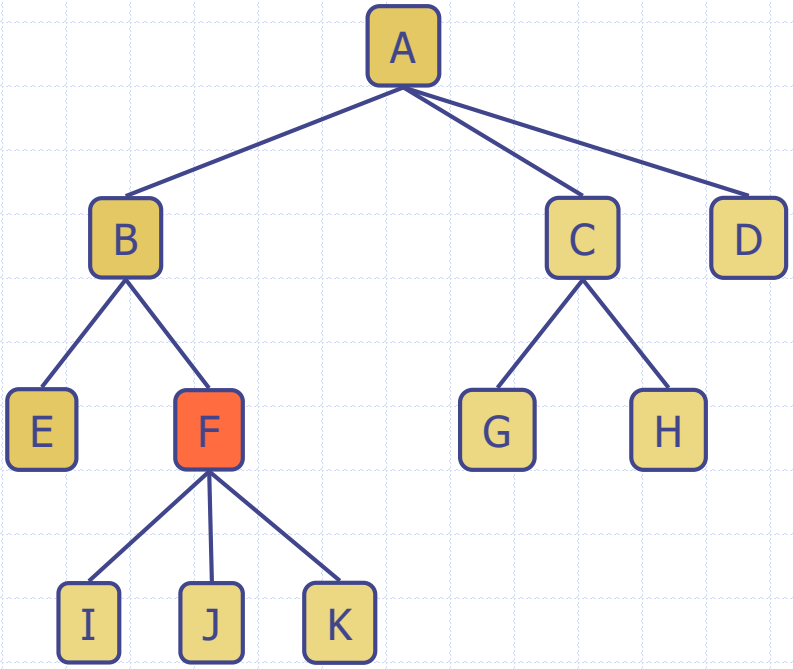
Iterative Implementation of Preorder Traversal: Demo

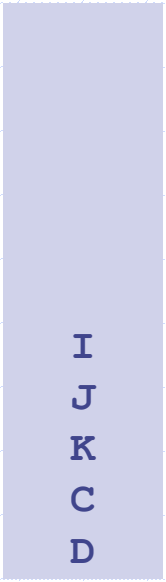


F
C
D

Preorder: A B E

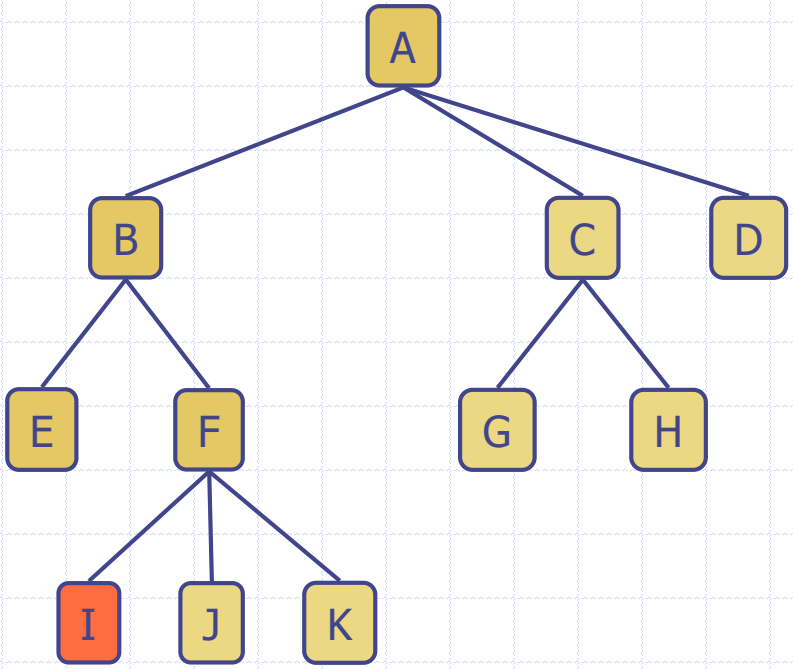
Iterative Implementation of Preorder Traversal: Demo





Preorder: A B E F

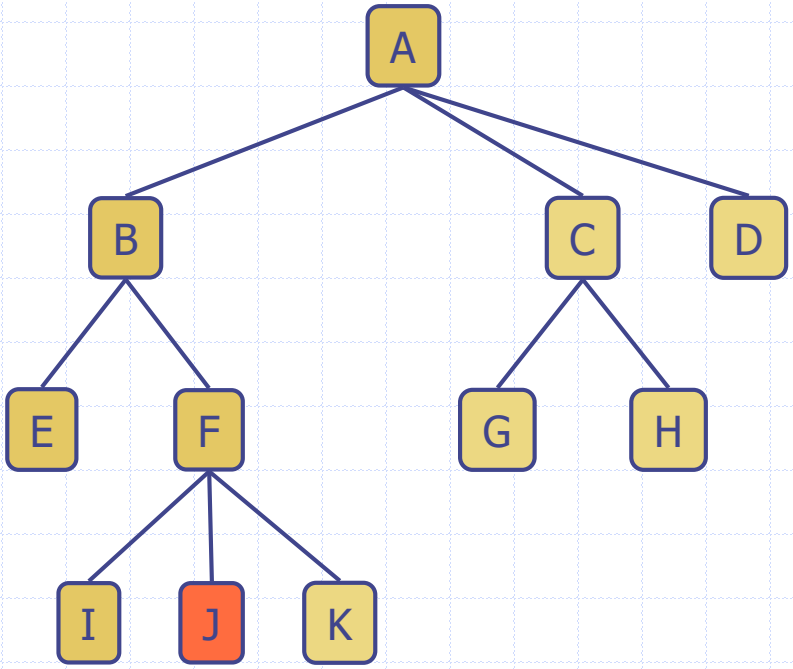
Iterative Implementation of Preorder Traversal: Demo



J
K
C
D

Preorder: A B E F I

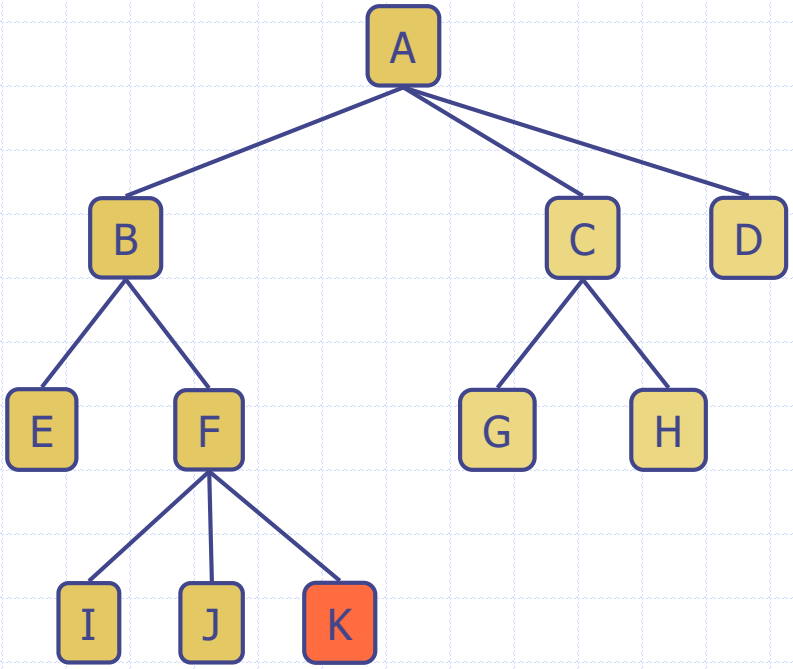
Iterative Implementation of Preorder Traversal: Demo

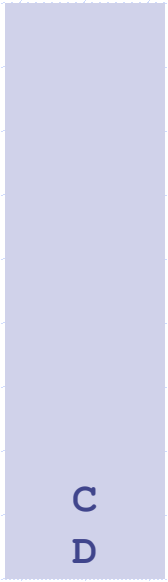


K
C
D

Preorder: A B E F I J

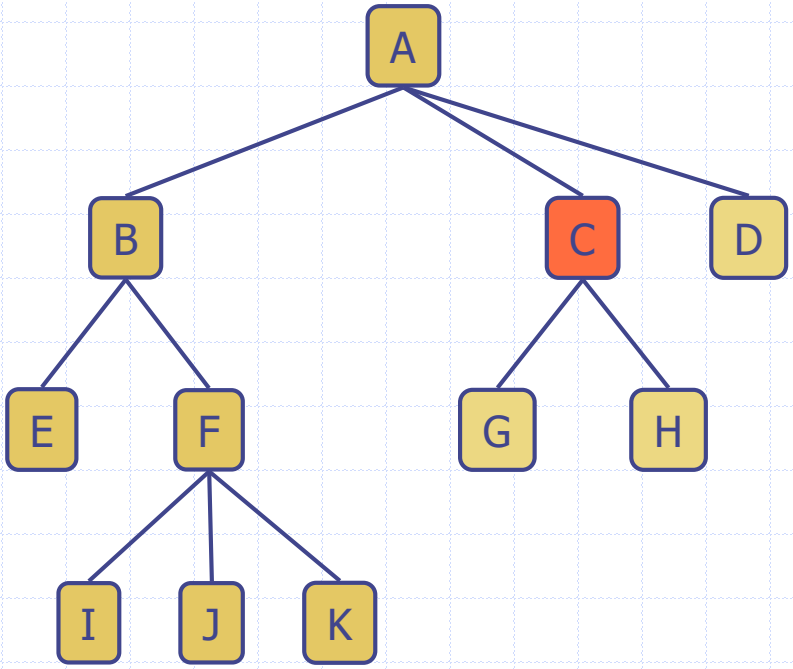
Iterative Implementation of Preorder Traversal: Demo

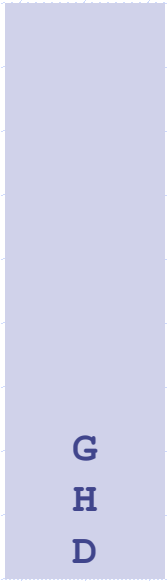




Preorder: A B E F I J k

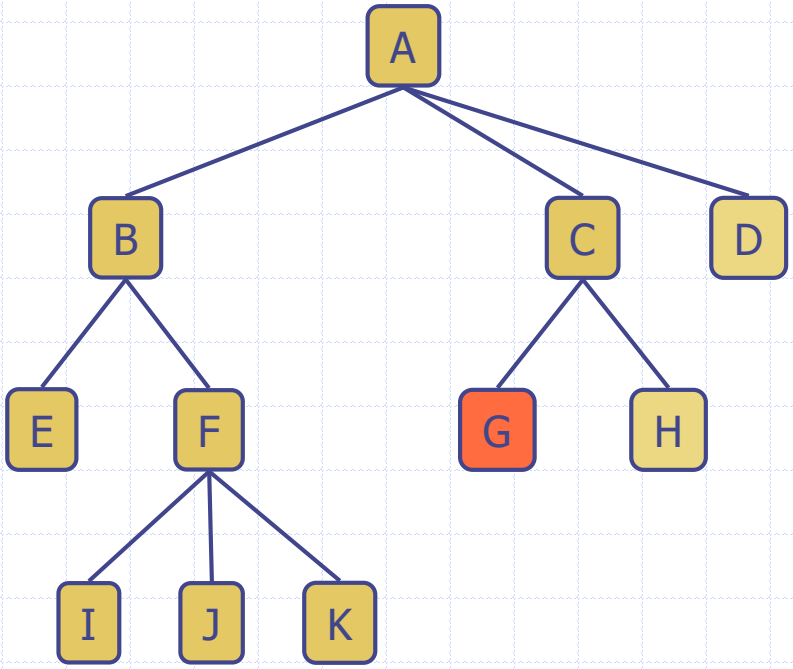
Iterative Implementation of Preorder Traversal: Demo

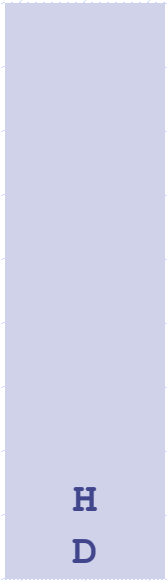




Preorder: A B E F I J k C

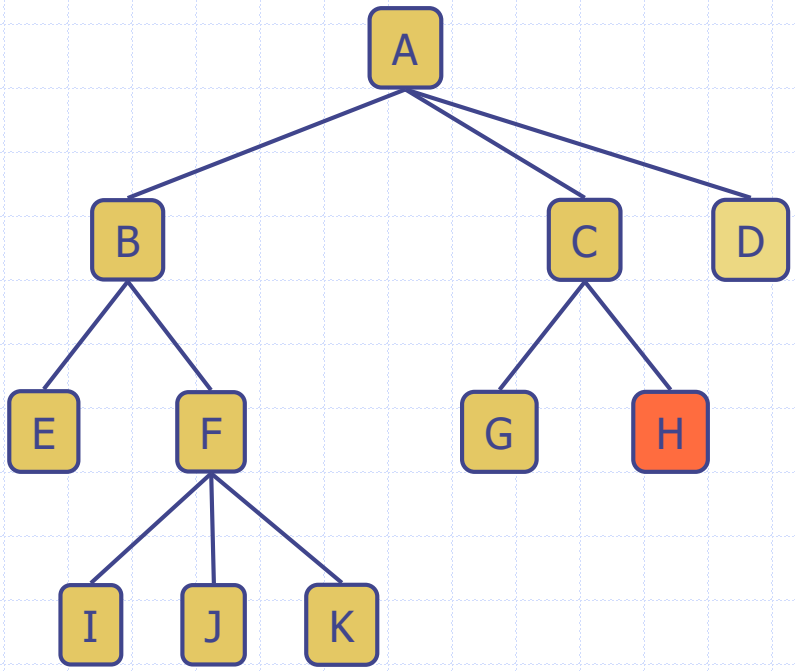
Iterative Implementation of Preorder Traversal: Demo

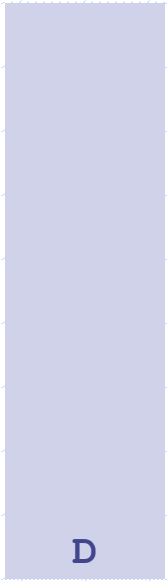




Preorder: A B E F I J k C G

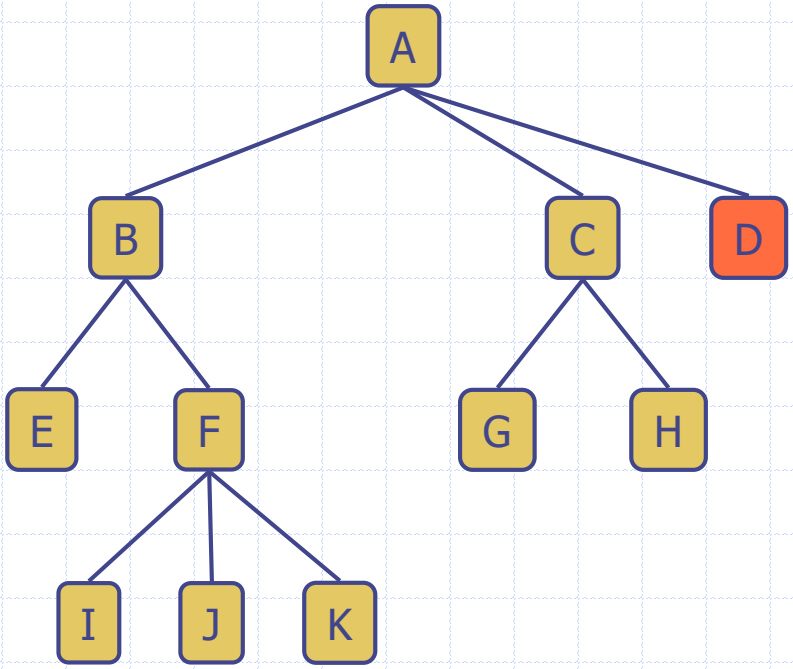
Iterative Implementation of Preorder Traversal: Demo





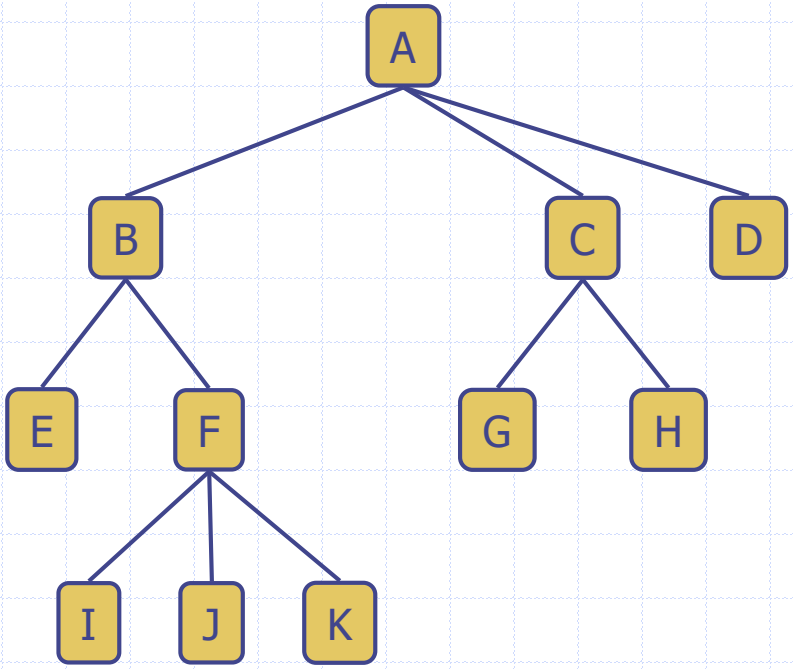
Preorder: A B E F I J k C G H

Iterative Implementation of Preorder Traversal: Demo



Preorder: A B E F I J k C G H D

Iterative Implementation of Preorder Traversal: Demo



Done!

Preorder: A B E F I J k C G H D

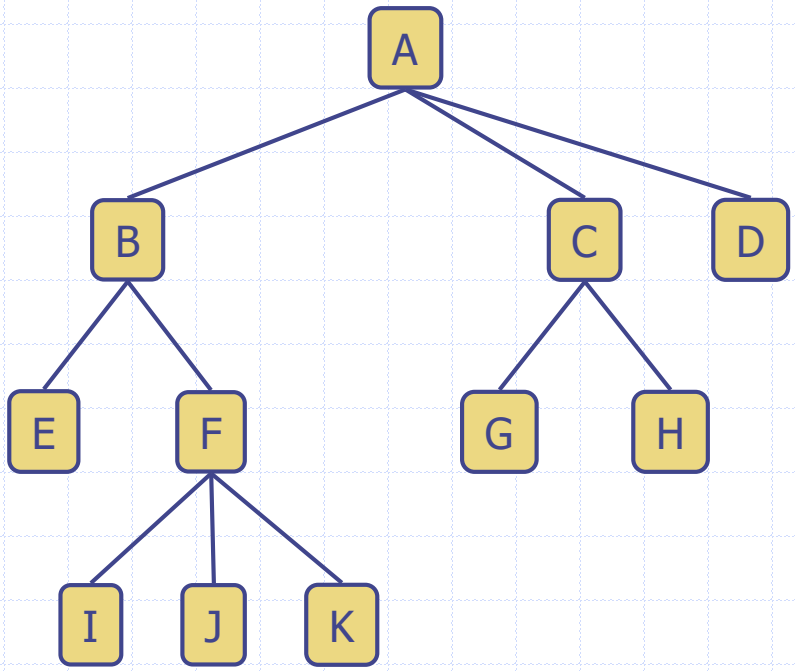
Iterative Implementation of Postorder Traversal

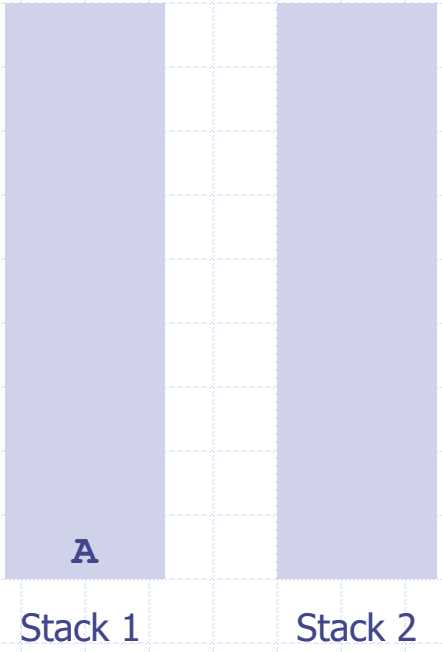
- Steps

See the animation in the subsequent slides!

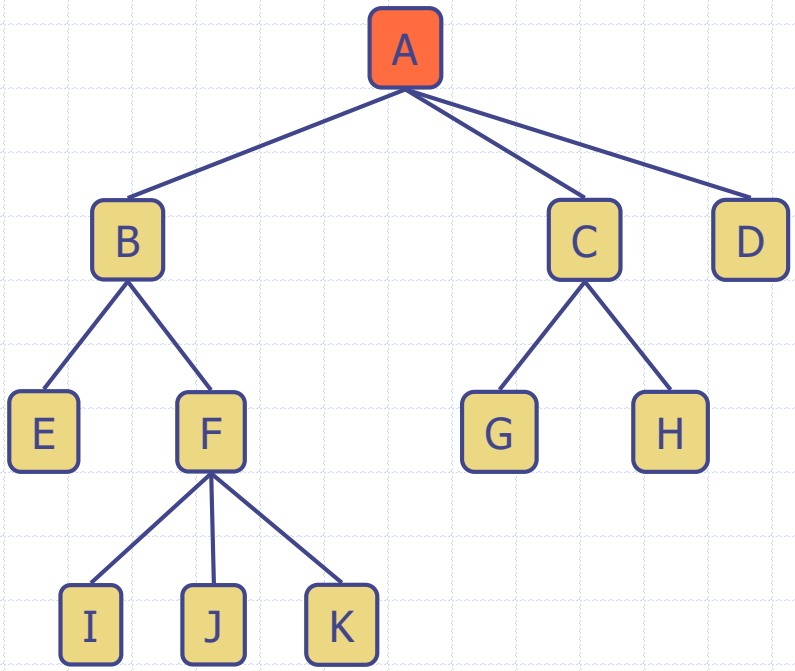
1. Push the root node to stack1 .
2. Pop a node from stack 1, and push it to stack 2.
3. Then push its children sequentially to stack 1.
4. Repeat step 2) and 3) until stack 1 is empty.
5. Pop all nodes from stack 2 to obtain the traversal in postorder.

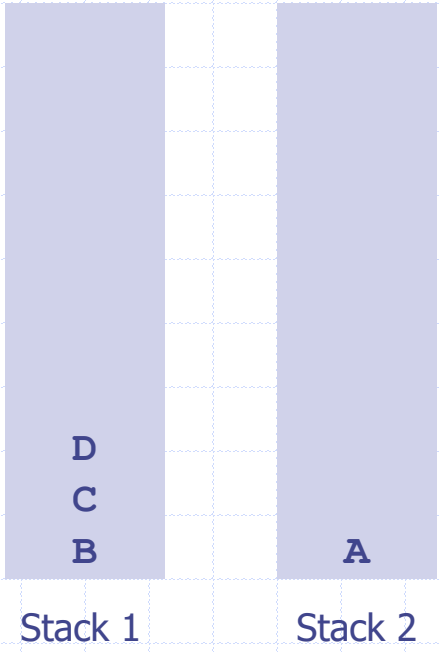
Iterative Implementation of Postorder Traversal: Demo



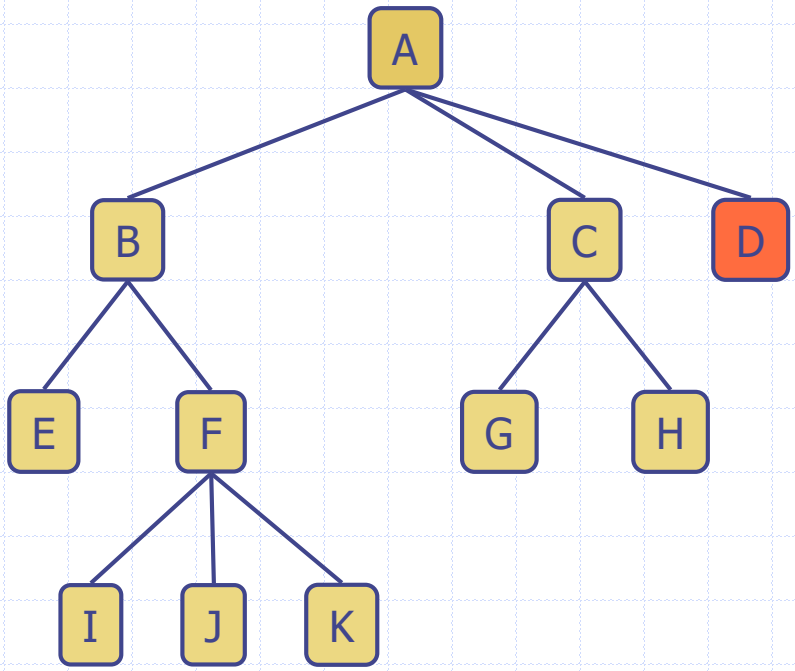


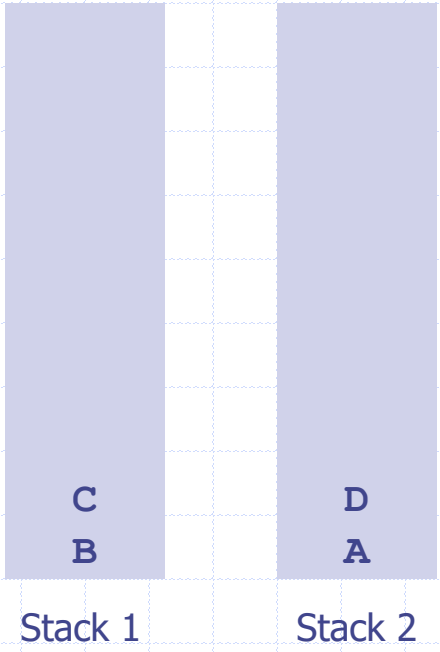
Iterative Implementation of Postorder Traversal: Demo



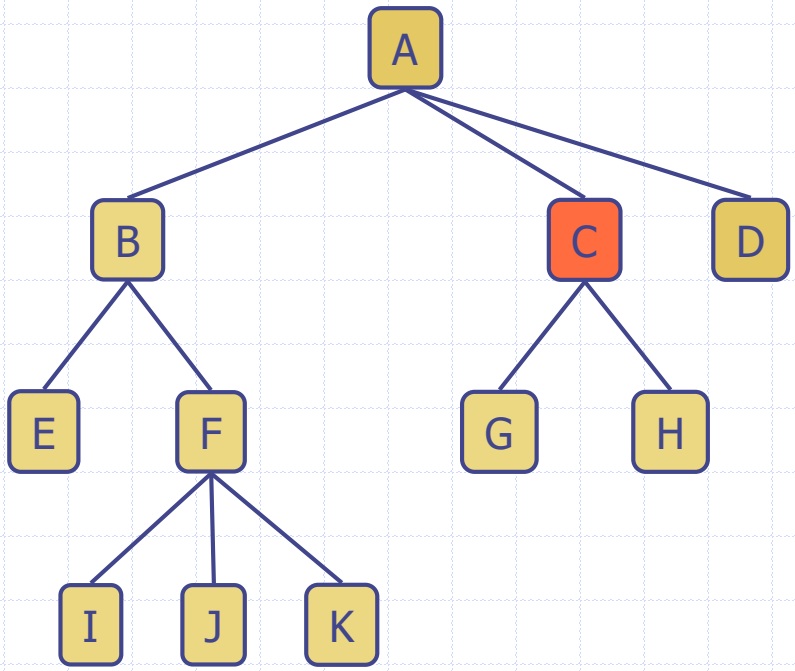


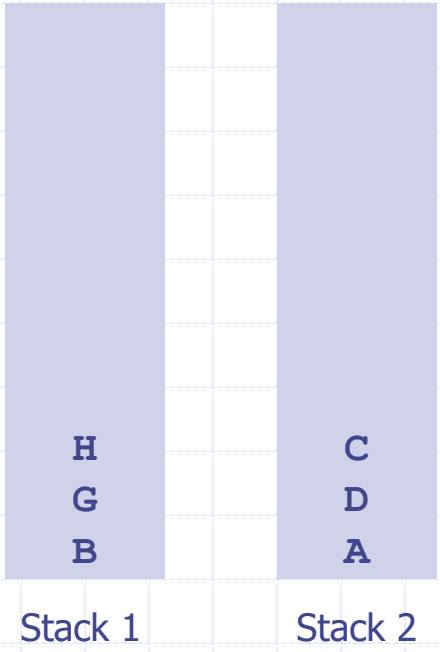
Iterative Implementation of Postorder Traversal: Demo



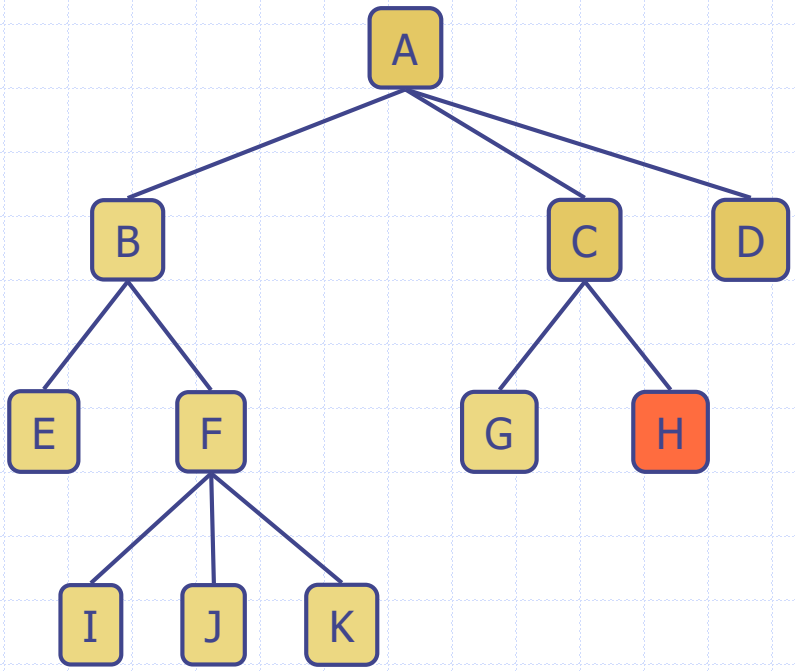


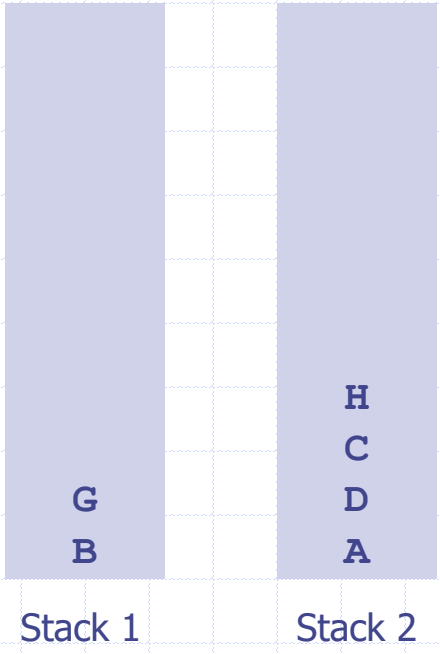
Iterative Implementation of Postorder Traversal: Demo



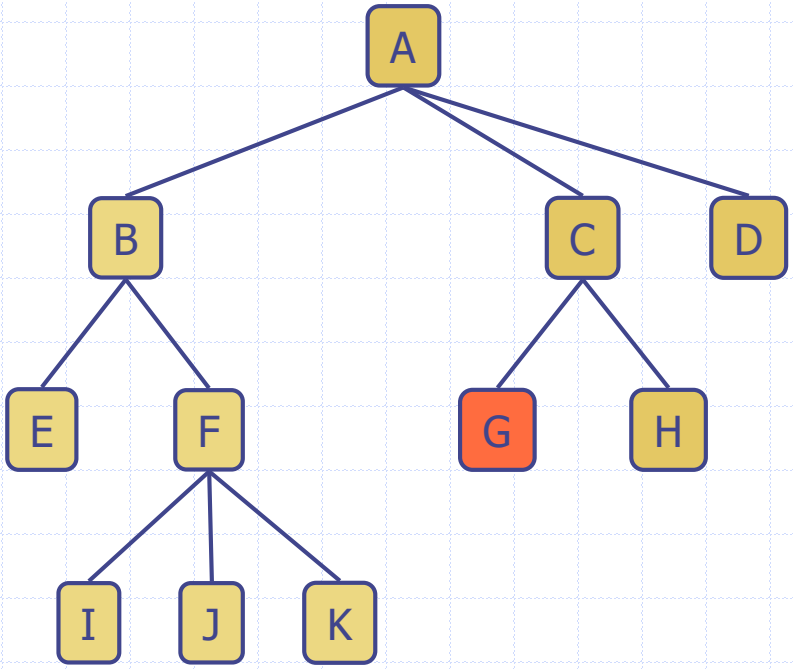


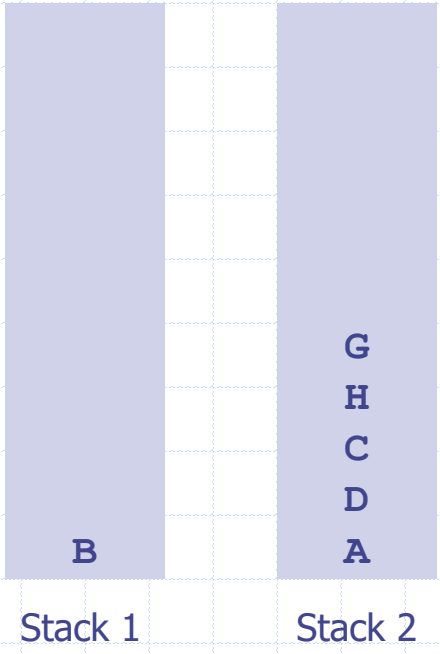
Iterative Implementation of Postorder Traversal: Demo



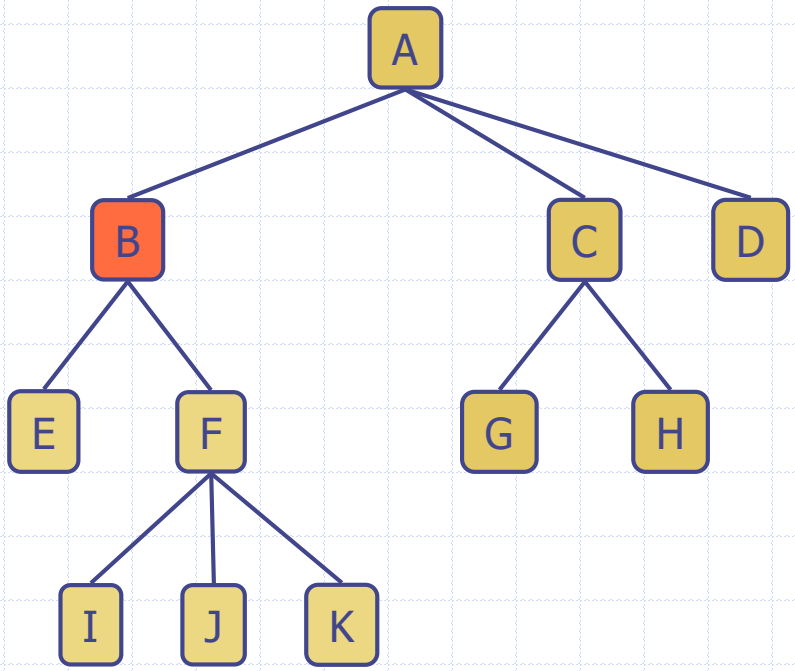


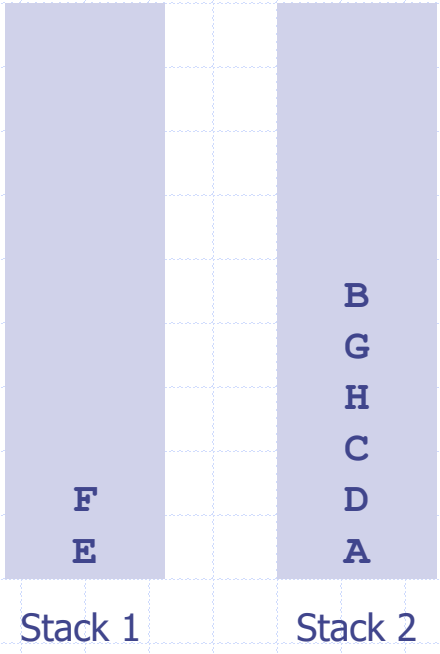
Iterative Implementation of Postorder Traversal: Demo



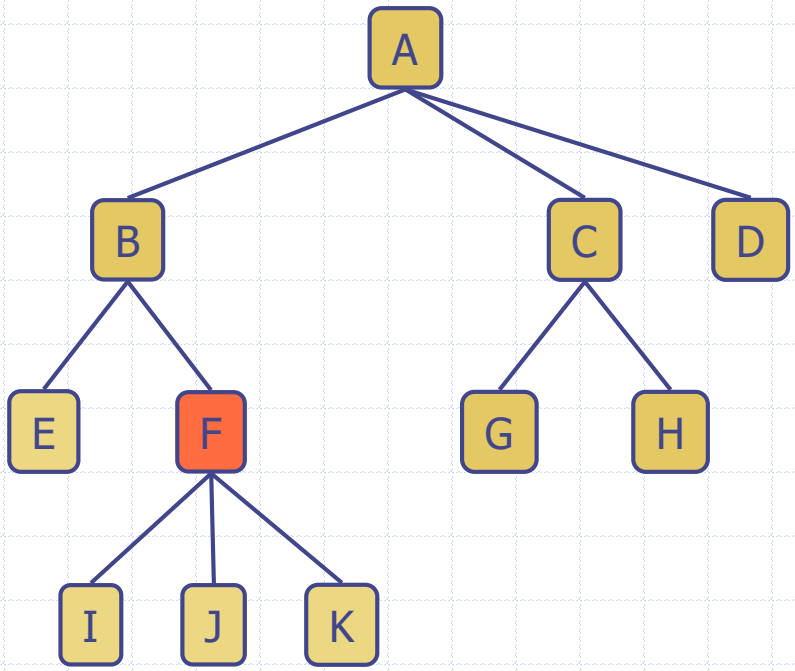


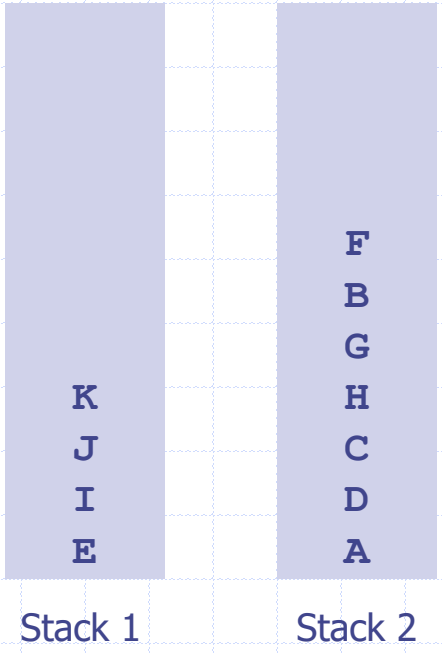
Iterative Implementation of Postorder Traversal: Demo



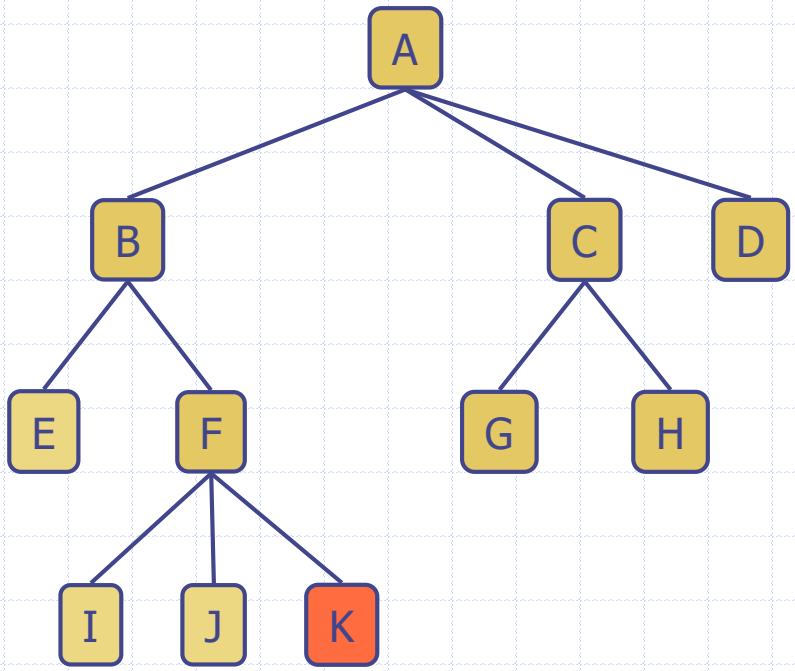


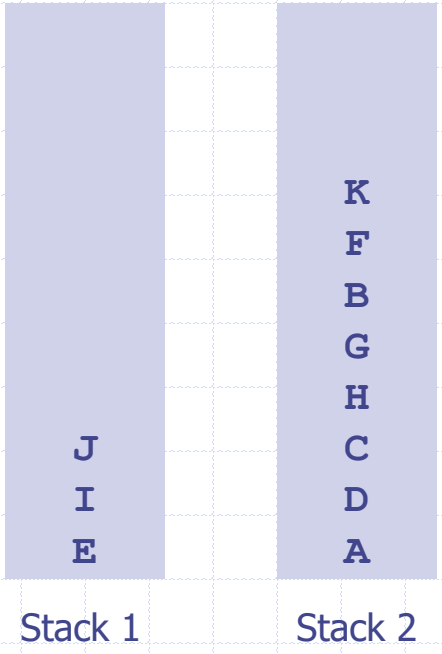
Iterative Implementation of Postorder Traversal: Demo



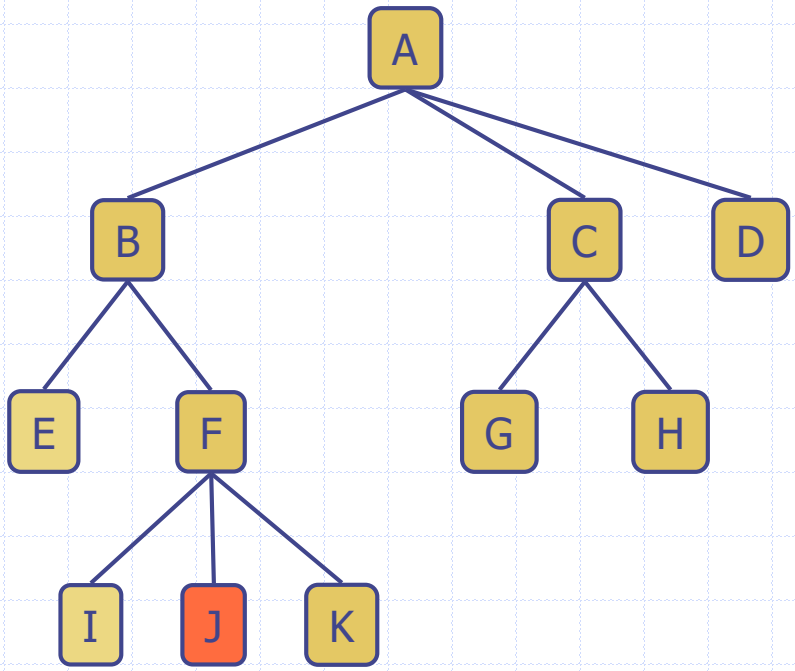


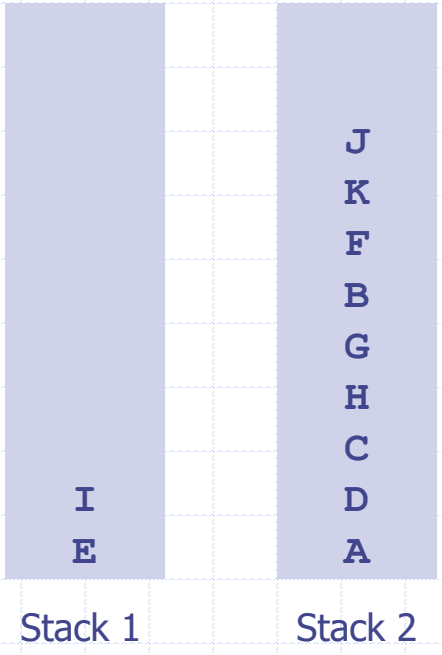
Iterative Implementation of Postorder Traversal: Demo



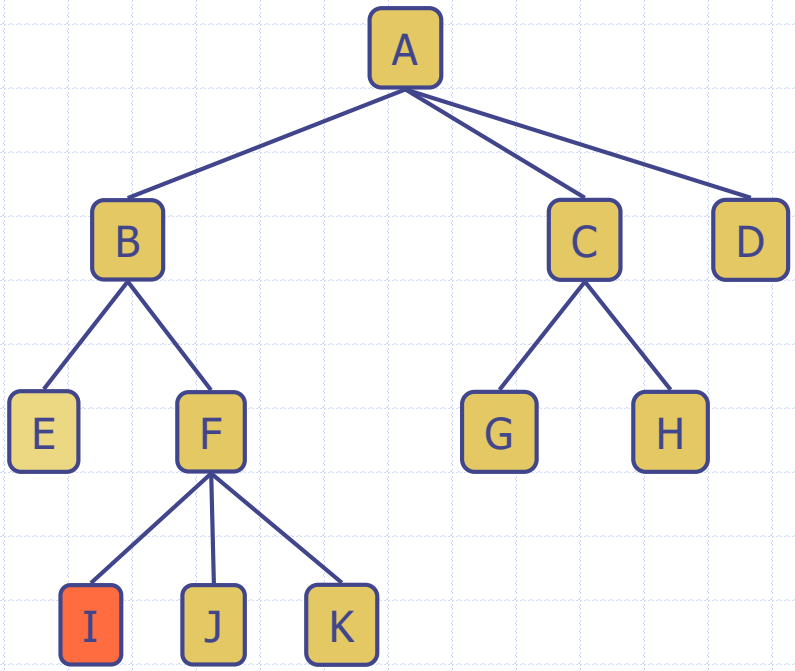


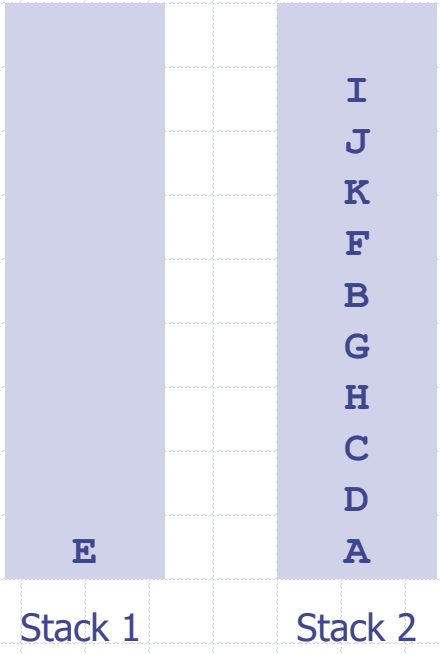
Iterative Implementation of Postorder Traversal: Demo



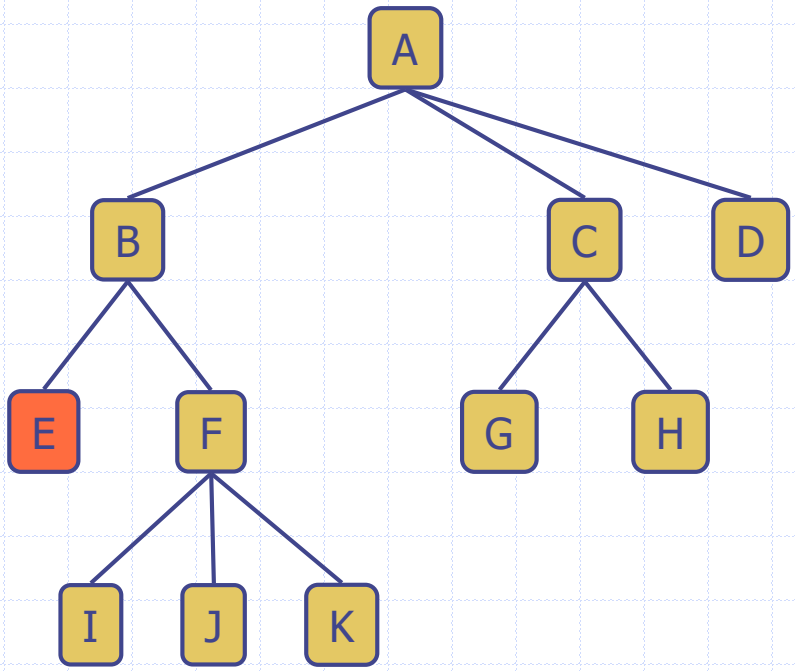


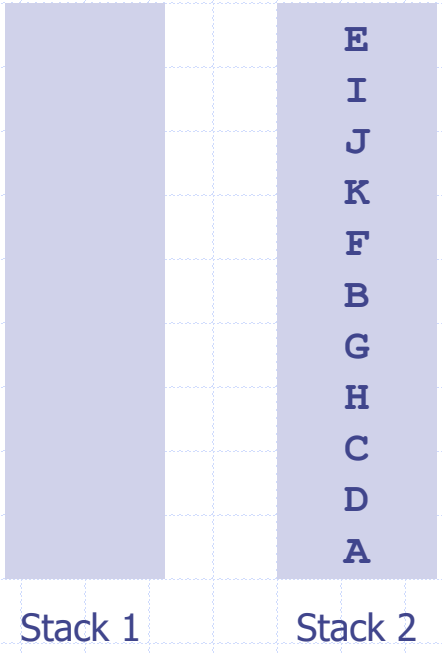
Iterative Implementation of Postorder Traversal: Demo



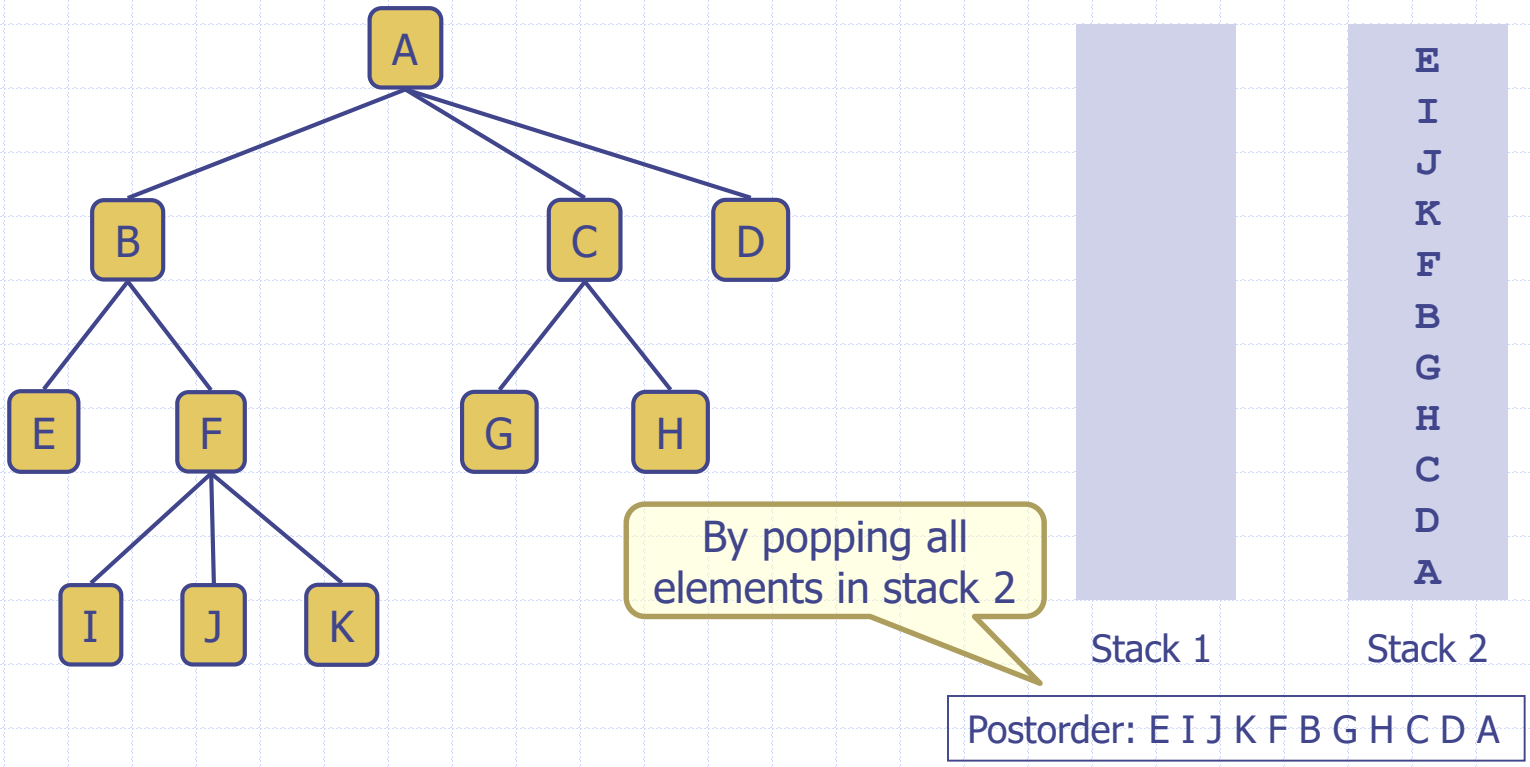


Iterative Implementation of Postorder Traversal: Demo



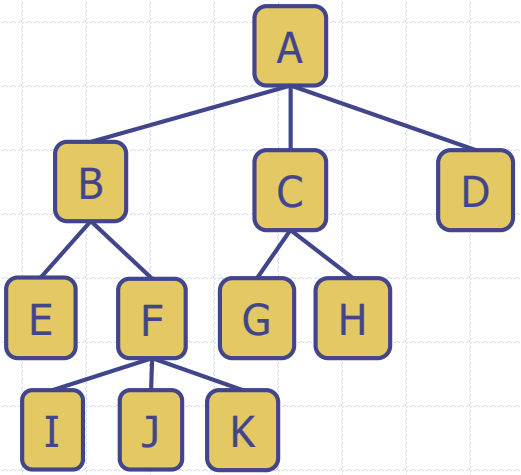


Iterative Implementation of Postorder Traversal: Demo



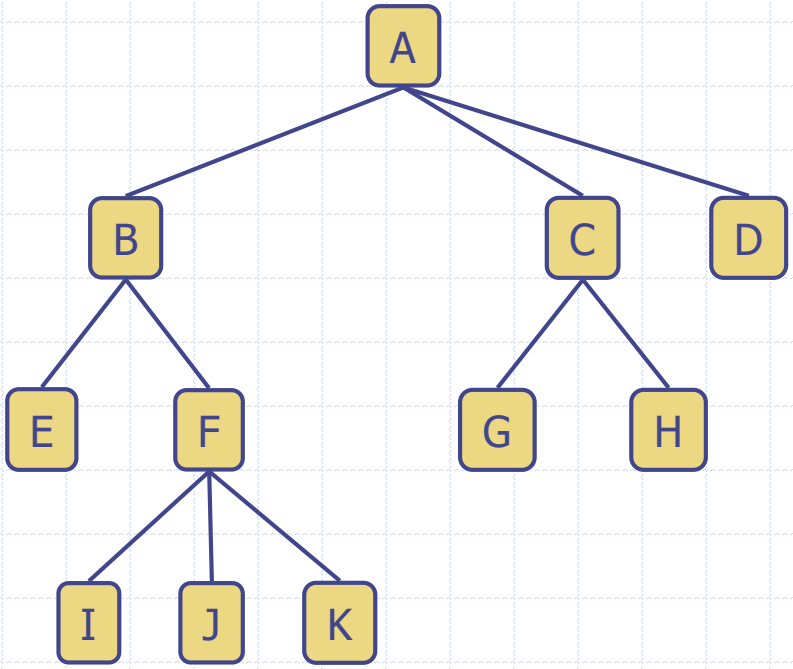
Other Traversal

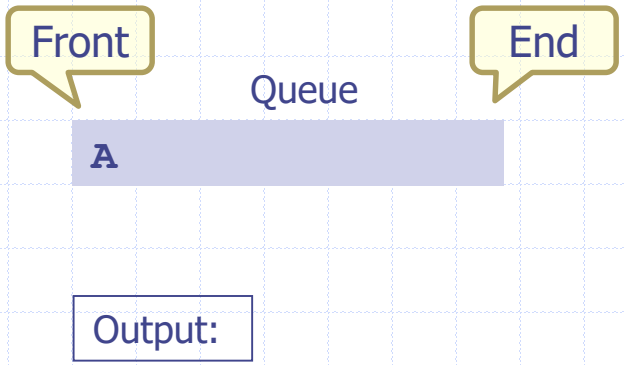
- Breadth-first traversal (aka level-order traversal)
  - Idea: Visit all the nodes at depth *d* before visiting the nodes at depth *d+1*
  - Implementation: Using a queue



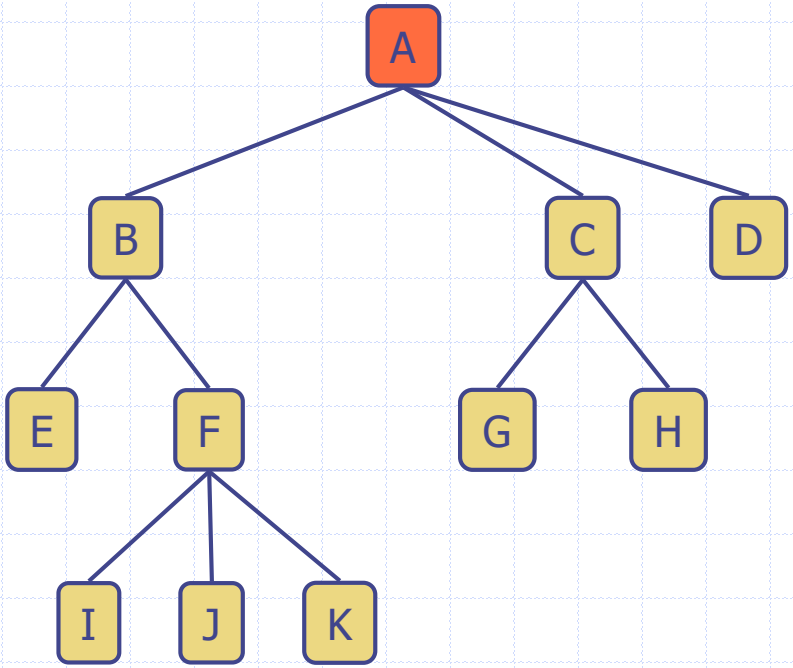
level-order traversal:
A B C D E F G H I J K

Breadth-first Traversal: Demo





Breadth-first Traversal: Demo

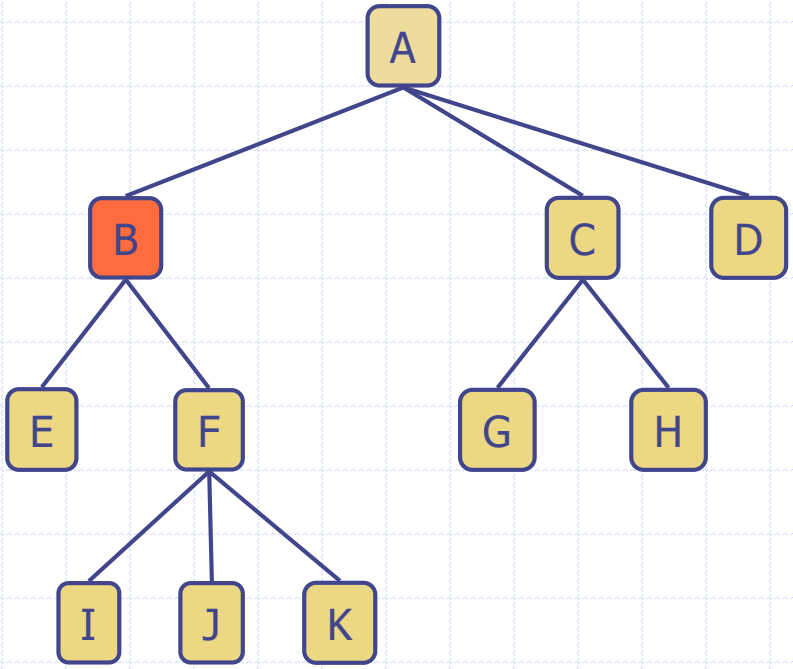


Queue

BCD

Output: A

Breadth-first Traversal: Demo

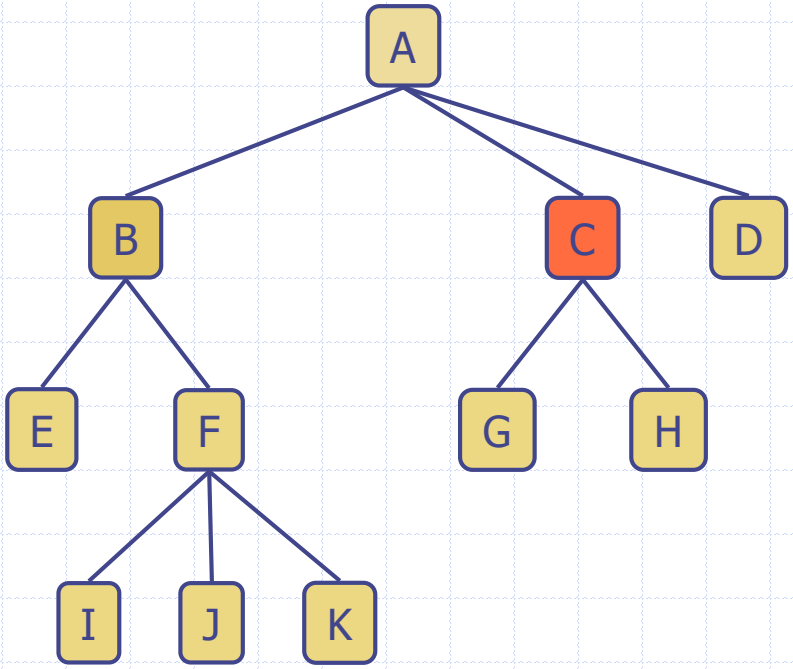


Queue

CDEF

Output: A~~B~~

Breadth-first Traversal: Demo

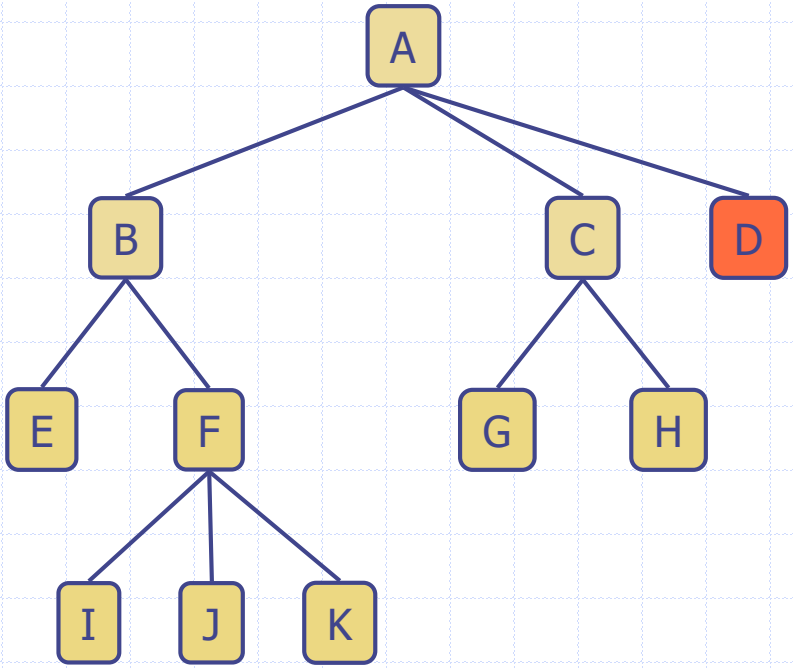


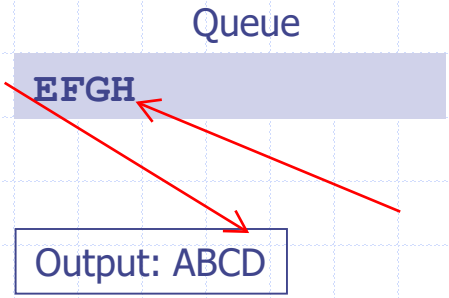
Queue

DEFGH

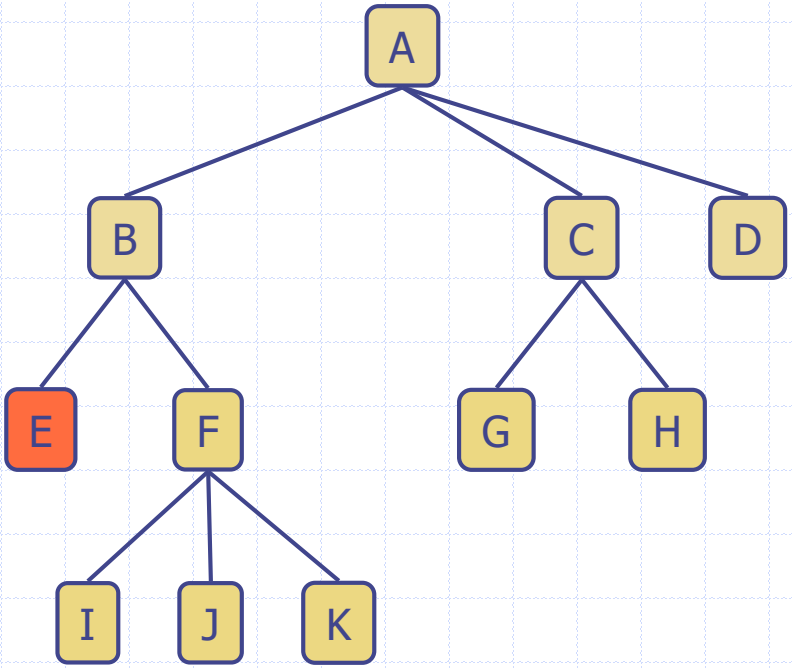
Output: ABC

Breadth-first Traversal: Demo





Breadth-first Traversal: Demo

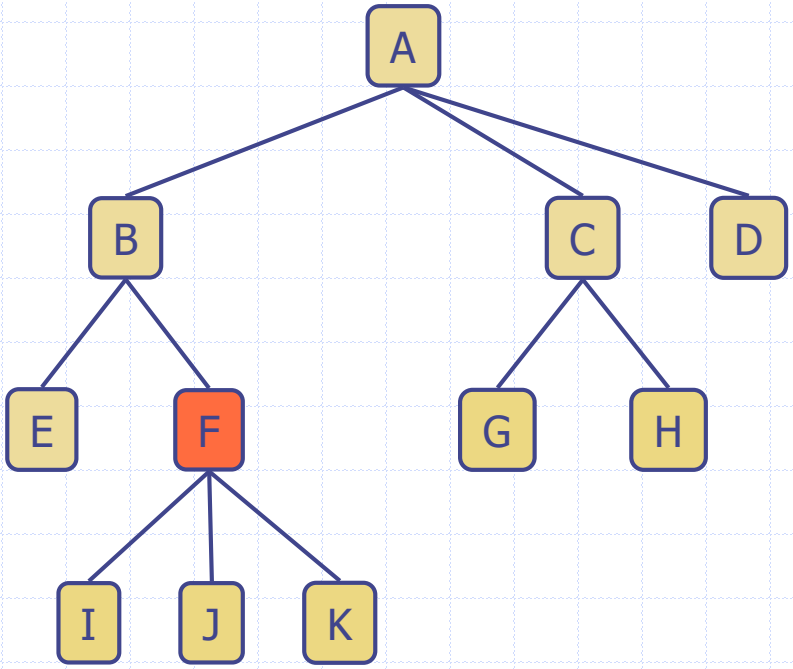


Queue

FGH

Output: ABCDE

Breadth-first Traversal: Demo

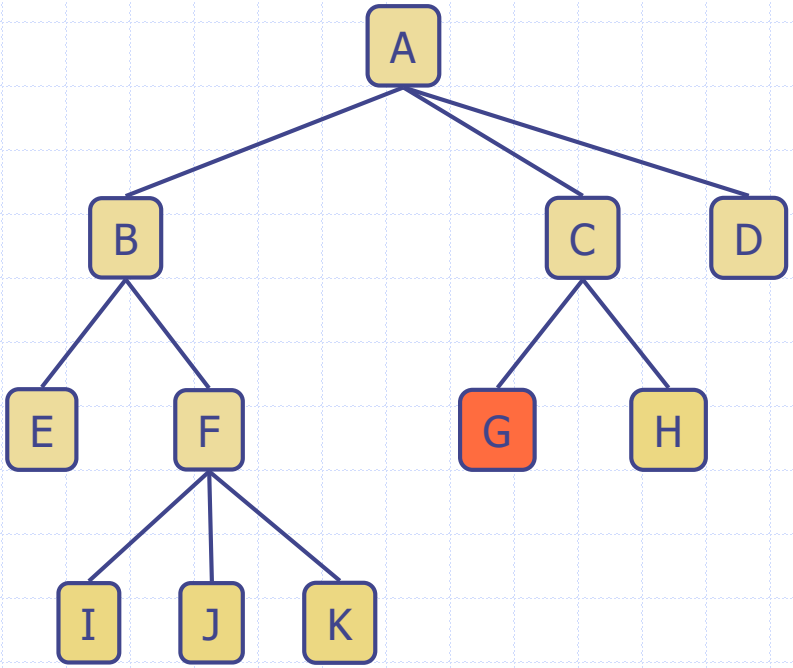


Queue

GHIJK

Output: ABCDEF

Breadth-first Traversal: Demo

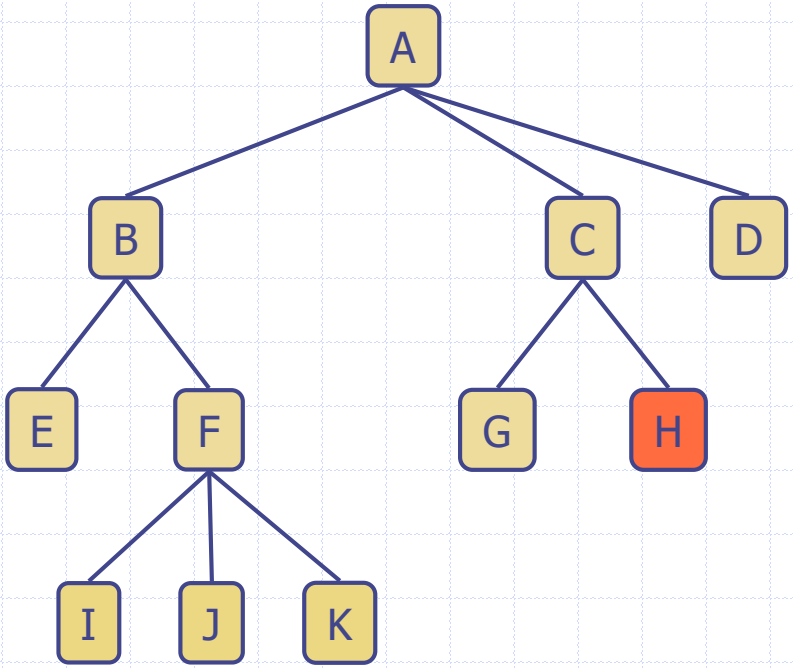


Queue

HIJK

Output: ABCDEFG

Breadth-first Traversal: Demo

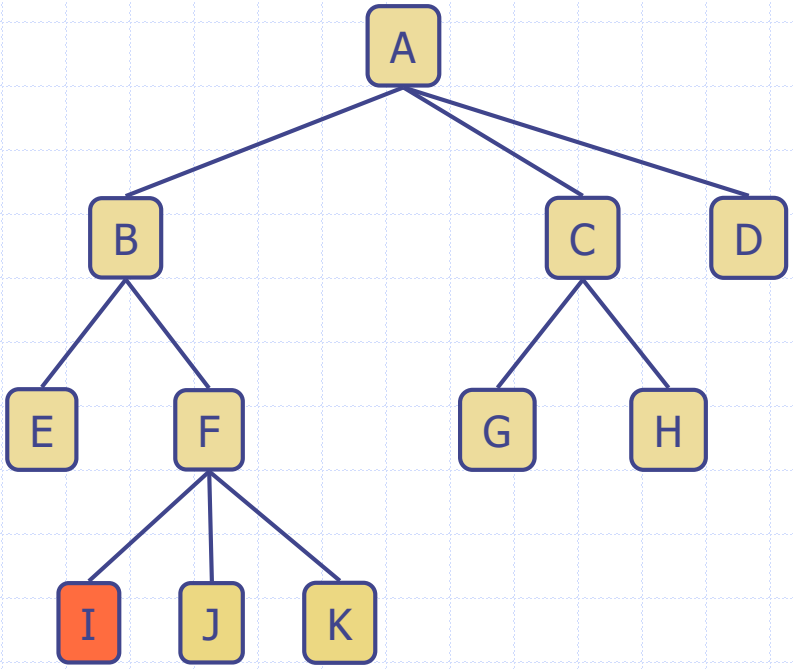


Queue

IJK

Output: ABCDEFGH

Breadth-first Traversal: Demo

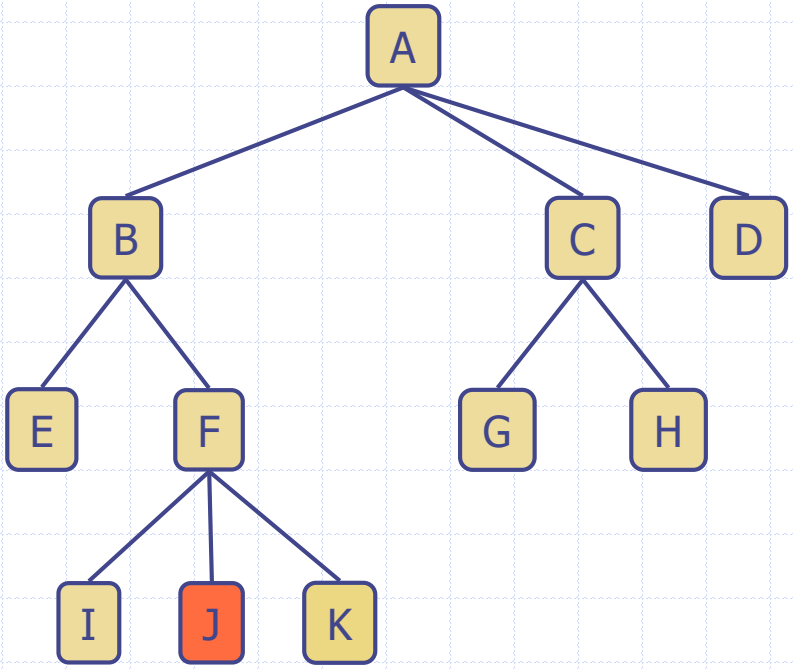


Queue

JK

Output: ABCDEFGHI

Breadth-first Traversal: Demo

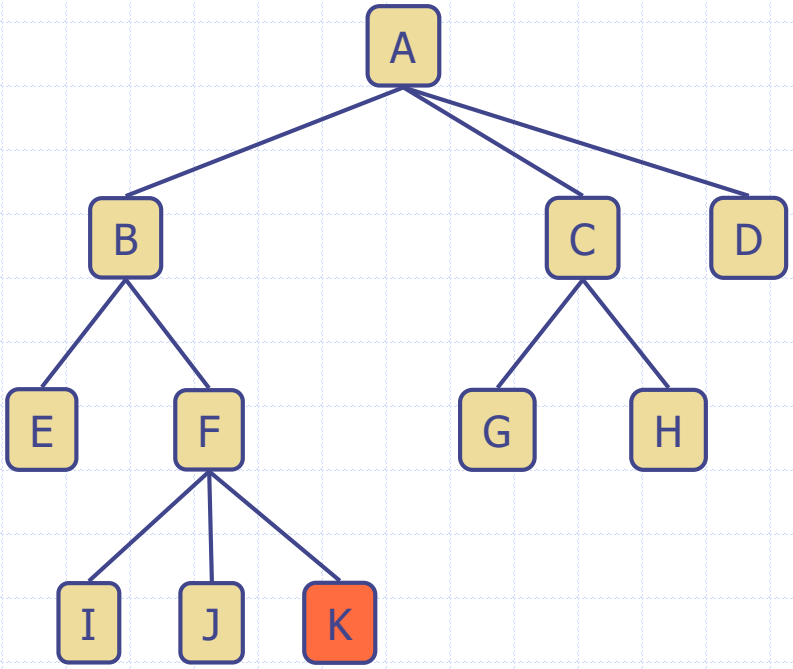


Queue

K

Output: ABCDEFGHIJ

Breadth-first Traversal: Demo



Queue

Output: ABCDEFGHIJK

Breadth-first traversal!

Quiz

- A general tree is shown next
  - Preorder?
  - Postorder?
  - Level order?