COL106
Data Structures and Algorithms

Subodh Sharma and Rahul Garg

Late Submission Policy for Assignments

- Late submissions cases
  - Missing their deadline by a small margin (e.g., 30 minutes)
  - Medical condition
  - Family emergency

Late Submission Policy

- All students get 7 days buffer for the semester
- Applicable assignment 3 onwards
- Late submission will automatically consume buffer days
- No penalty if buffer days are available
- 20% per day penalty if buffer days have been exhausted
- Will be automatically applied to late submissions A3 onwards
  - First buffer days then penalty days

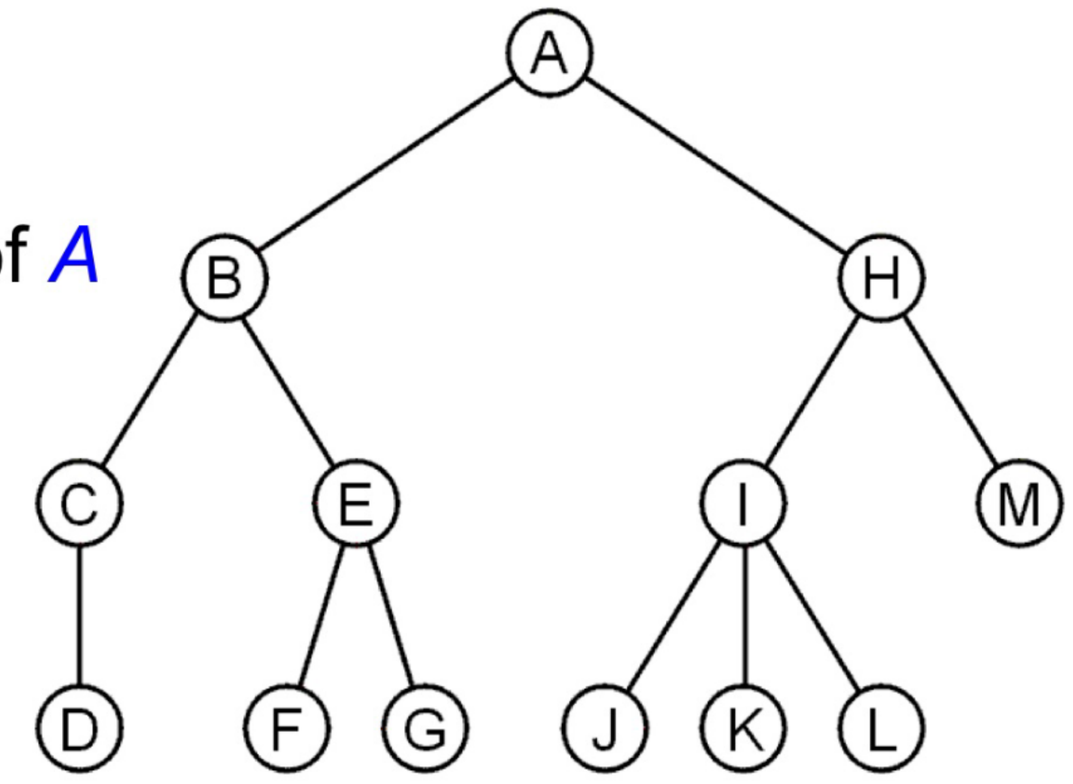
Late Submission Policy: Medical Issues

- Obtain a medical certificate clearly stating that your condition and the doctor's recommendation
- Submit your IIT-Delhi medical certificate to your lab TA
- Do not sent emails to the instructors
- Family medical issues
  - Relaxation give only in the case of immediate family members
  - Extreme situations only (death or emergency hospitalization)
  - Produce suitable documents (discharge reports from the hospital) to your TA

Tree Traversals

Recall Trees

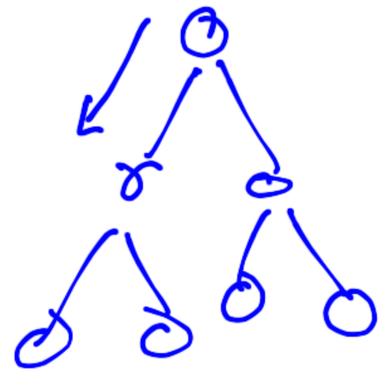
- Collection of nodes with parent-child relationship
- *A* is the *root* node
- *B* is *parent* of C & E
- *B* is *ancestor* of C, E, D, F, G
- D, C and G are *descendants* of *A*
- *C* is the *sibling* of E
- *I* and *M* are the *children* of H
- *D, F, G, J, K, L, M* are *leaves*



Tree Traversals

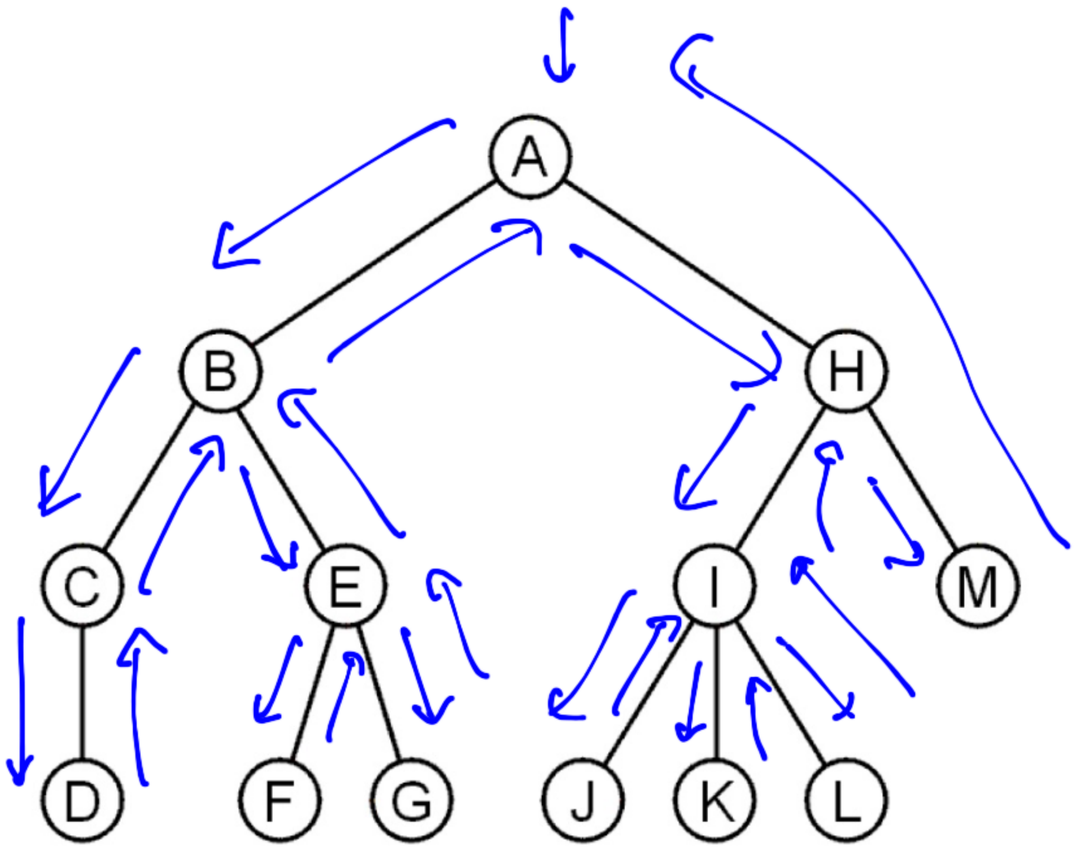
- Visiting all the items of a tree exactly once
  - Say to print or add the values stored in the nodes
- Traversal is simple for linear data structures such as arrays, linked lists, queues, stacks
- How to traverse a tree?

Tree Traversals



- Pre-order traversal: Processes a node before processing its children
- Post-order traversal: Processes a node after processing its children
- In-order traversal: Process left subtree followed by processing the node and then the right sub-tree
- Breath-first traversal: Processes all the nodes at a level from left to right beginning at the root

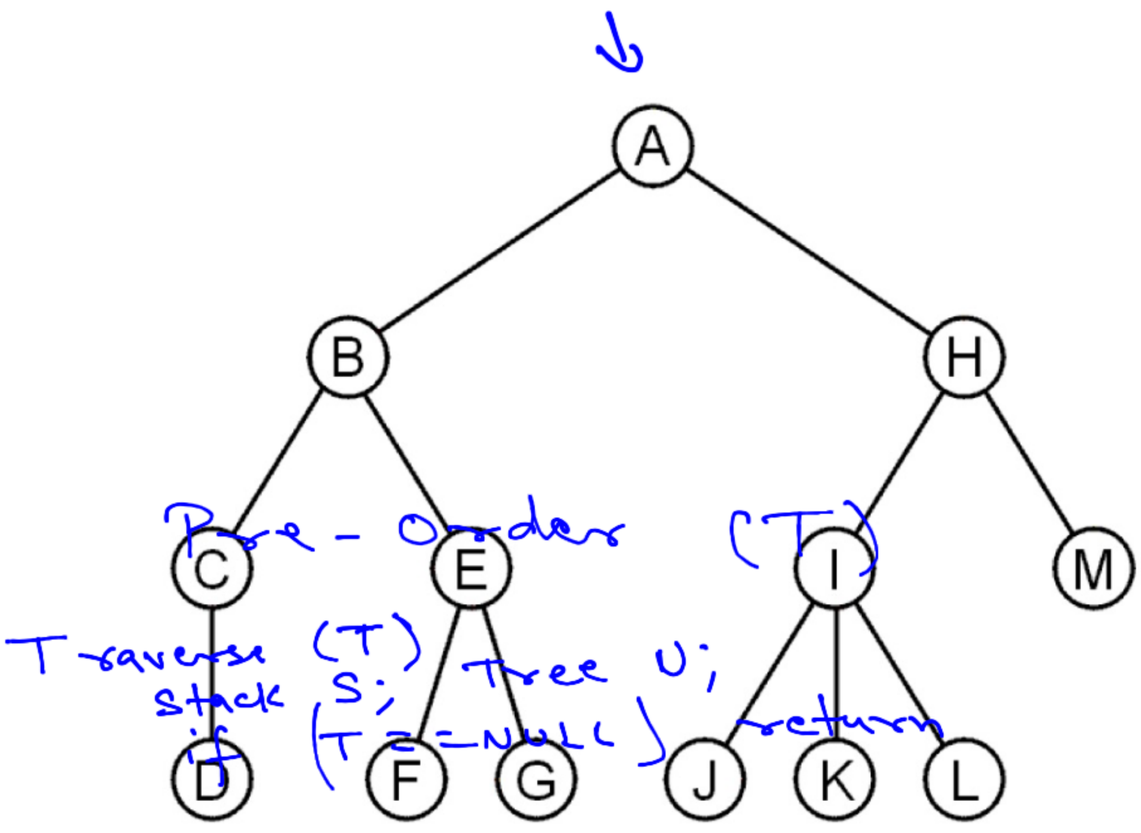
Pre-order Traversal



- First visit the node
- Then visit its children recursively
- Order of visiting the nodes

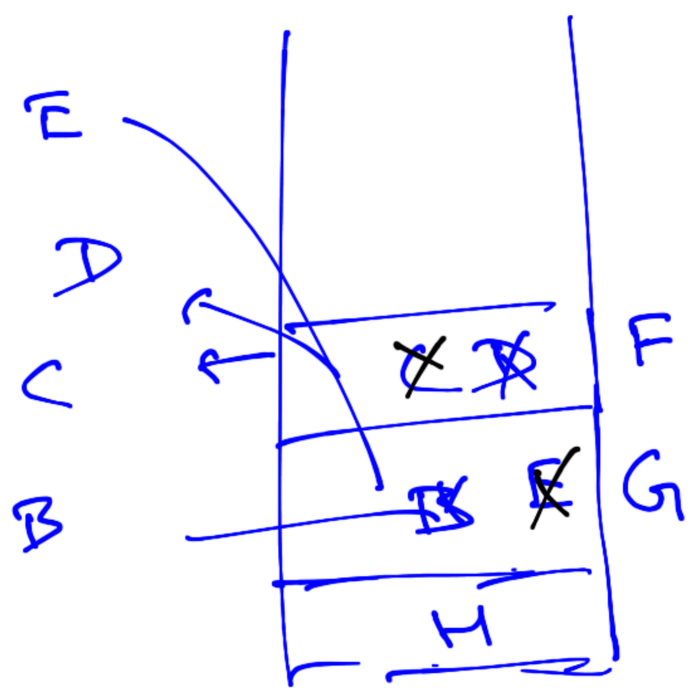
A, B, C, D, E, F, G, H, I, J, K, L, M

Pre-order Traversal (Iterative)



- Order of visiting the nodes

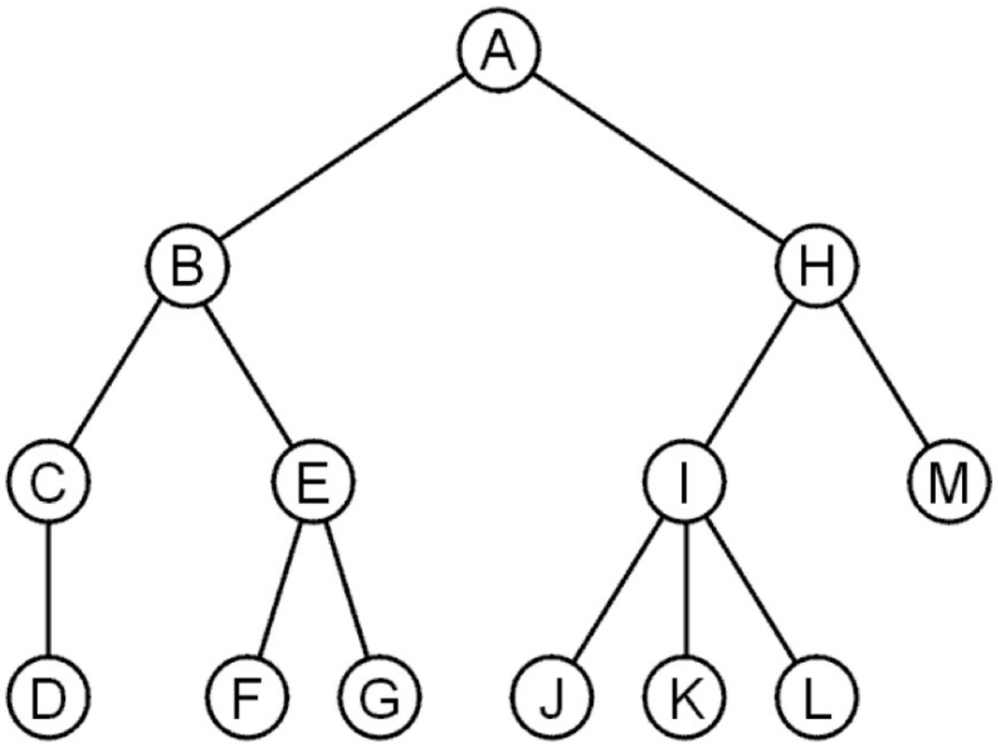
A, B, C, D, E, F, G, H, I, J, K, L, M




```
push (T)
while Not empty(S)          -(1)
      U = pop (S)           -(2)
      visit (U)             -(3)
if (U.right != NULL)
    --> push (U.right)      -(4)
if  --> push (U.left)       -(5)
end while
```

Output : A, B, C, D, E

Pre-order Traversal



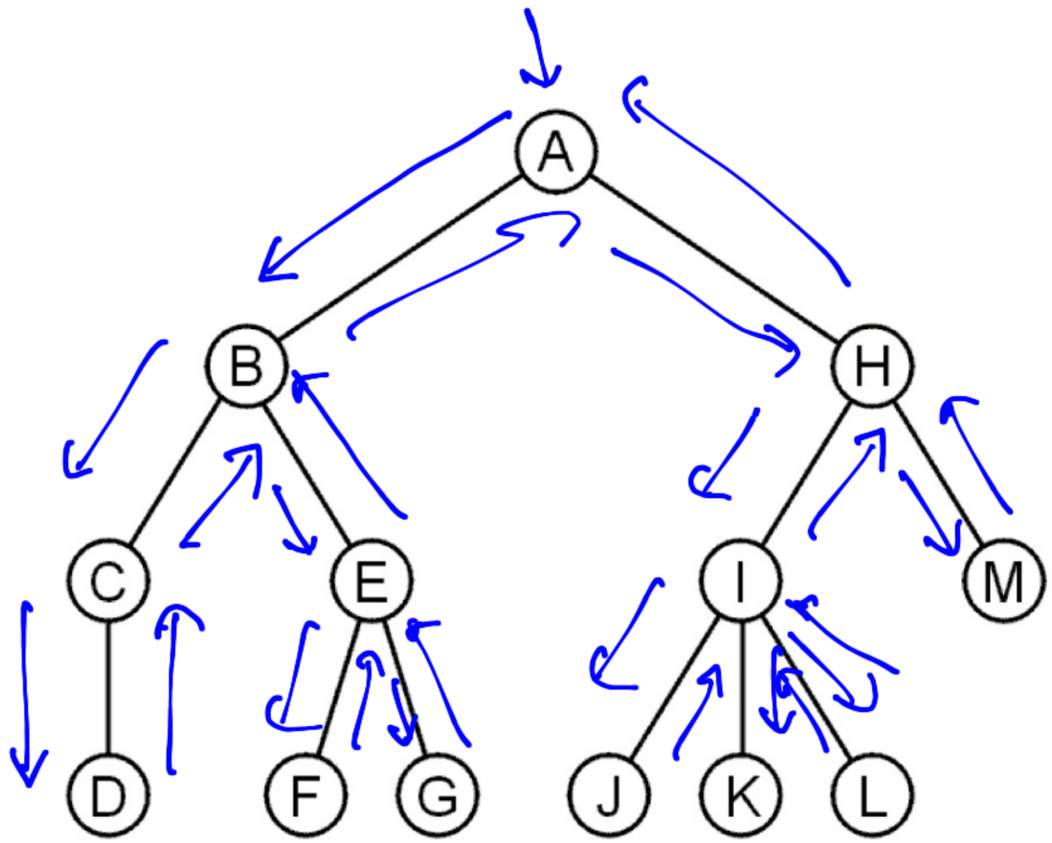
- Order of visiting the nodes

A, B, C, D, E, F, G, H, I, J, K, L, M

Pre-order Traversal

```
preorder(v)
        if (v == null) then return
        else visit(v)
        preorder(v.left)
        preorder(v.right)
```

Post-order Traversal



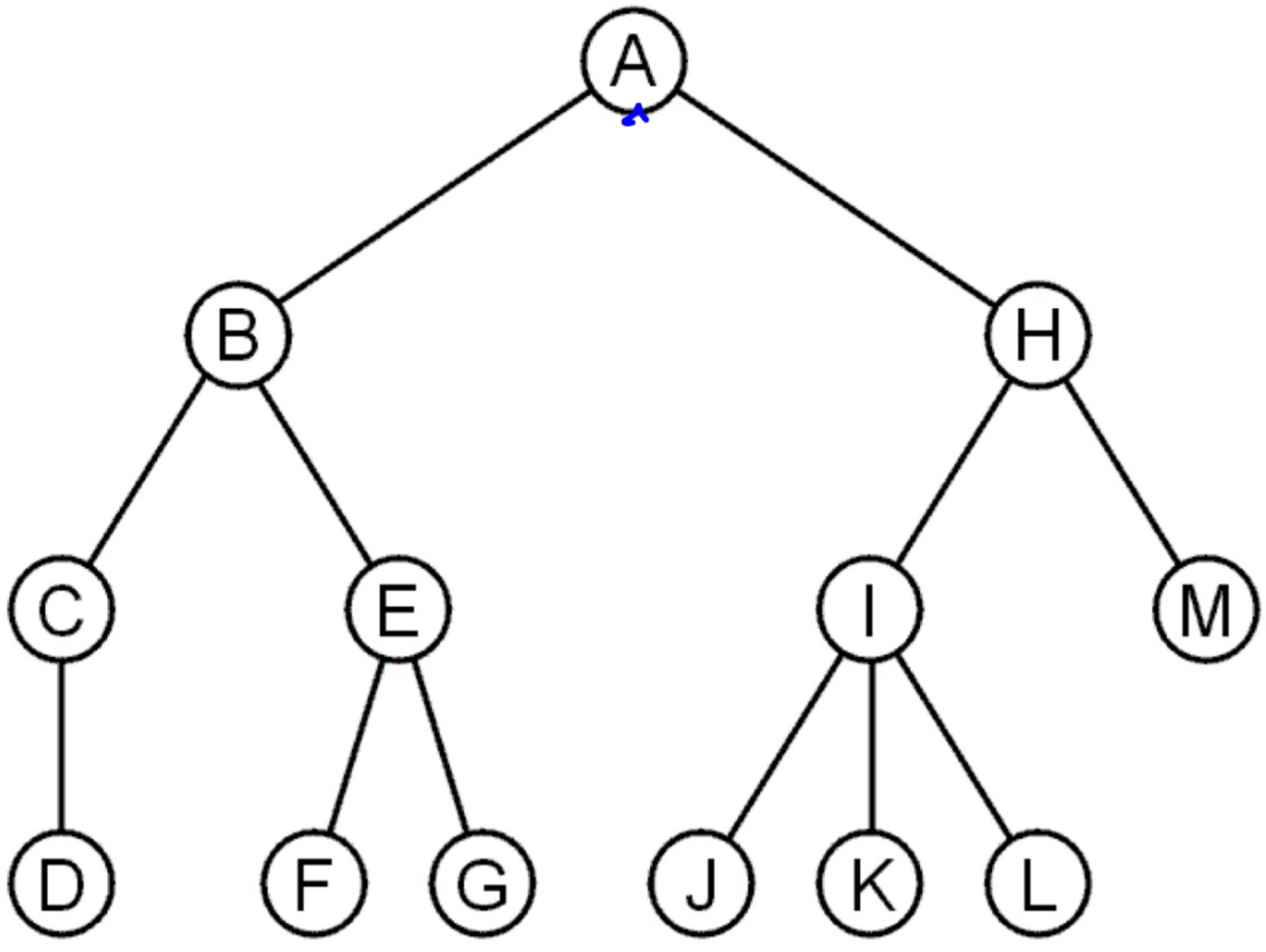
- First visit the children recursively
- Then visit the node

D, C, F, G, E, B, J, K, L, I, M, H, A

Post-order Traversal

```
postorder(v)
        if (v == null) then return
        else
            postorder(v.left)
            postorder(v.right)
    visit(v)
```

Post-order Traversal



Pre-order

A, B, C, D, E, F, G, H, I, J, K, L, M

Post-order

D, C, F, G, E, B, J, K, L, I, M, H, A

In-Order Traversal

- First visit left subtree recursively
- Then visit the node
- Then visit the right subtree recursively

In-order Traversal

```
Inorder(v)
        if (v == null) then return
        else
            Inorder(v.left)
            visit(v)
            Inorder(v.right)
```

In-order Traversal (Binary Trees Only)

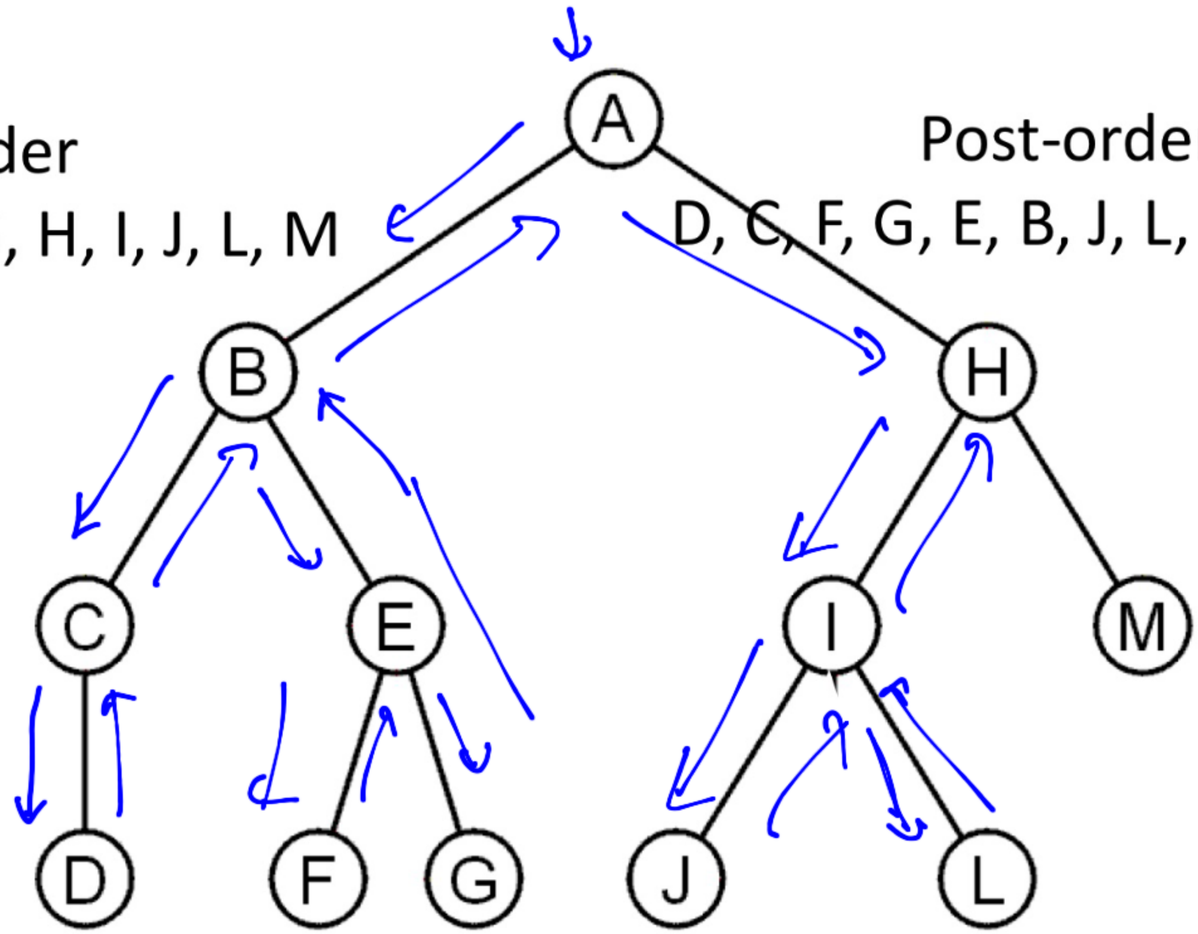
Pre-order

A, B, C, D, E, F, G, H, I, J, L, M

Post-order

D, C, F, G, E, B, J, L, I, M, H, A

D, C, B, F, E, G, A, J, I, L, H, M



In-order

D, C, B, F, E, G, A, J, I, L, H, M

In-order Traversal (Binary Trees Only)

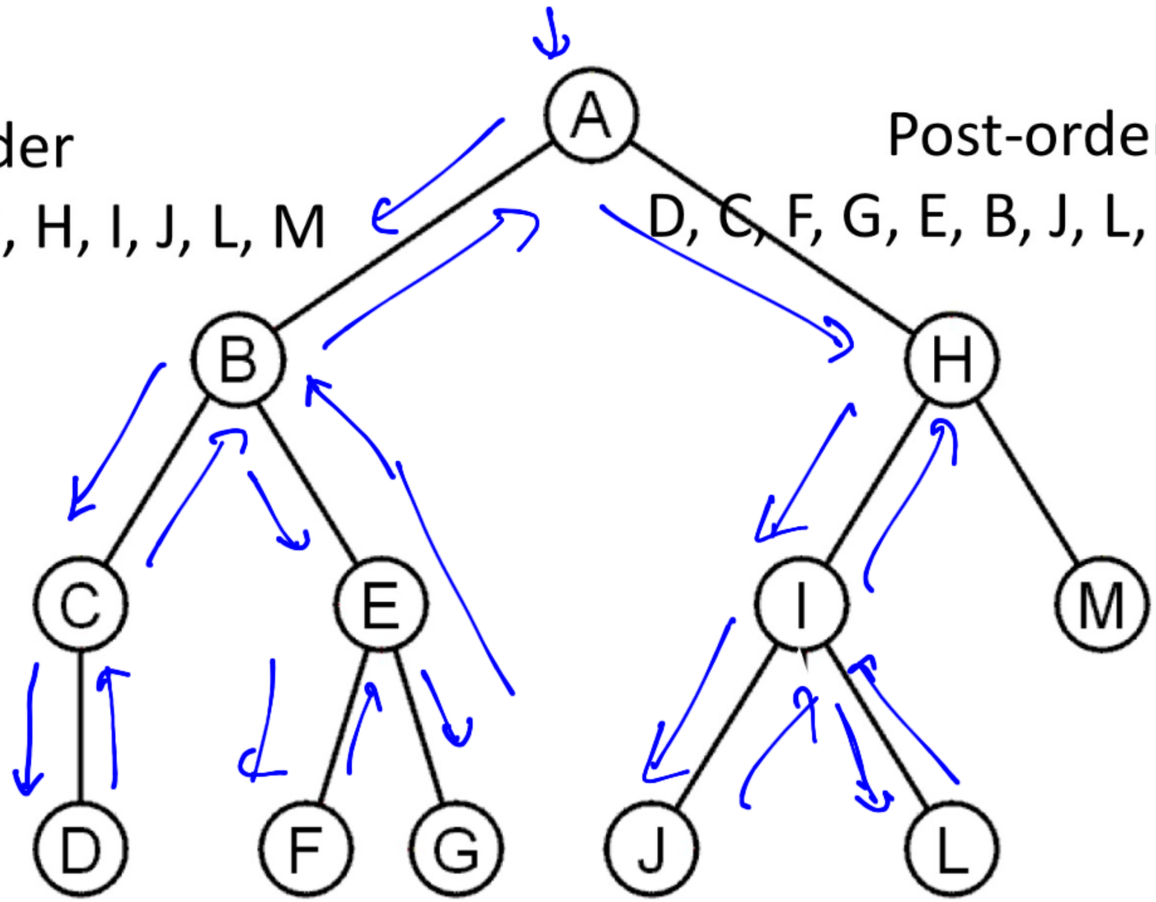
Pre-order

A, B, C, D, E, F, G, H, I, J, L, M

Post-order

D, C, F, G, E, B, J, L, I, M, H, A

D, C, B, F, E, G, A, J, I, L, H, M

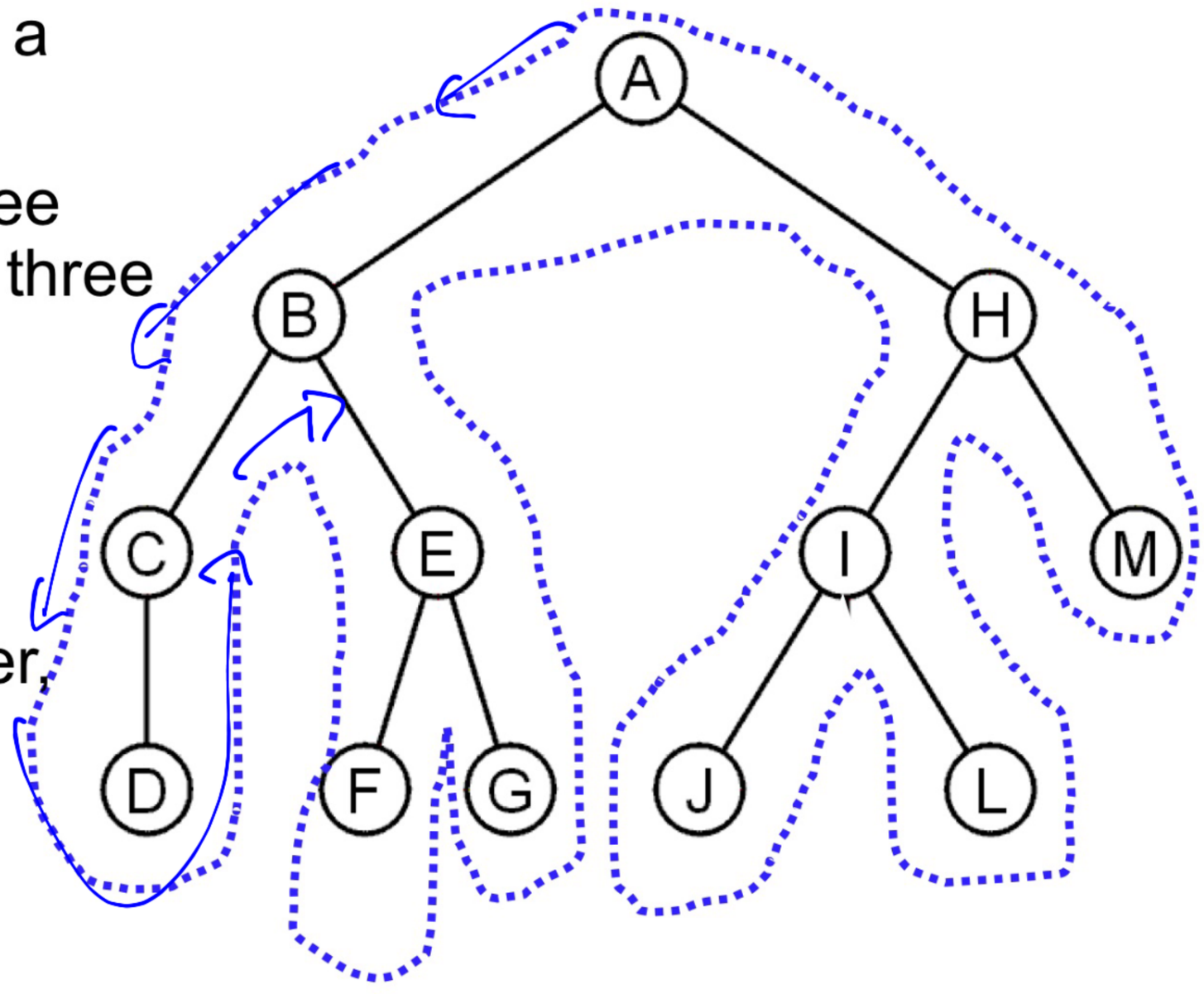


In-order

D, C, B, F, E, G, A, J, I, L, H, M

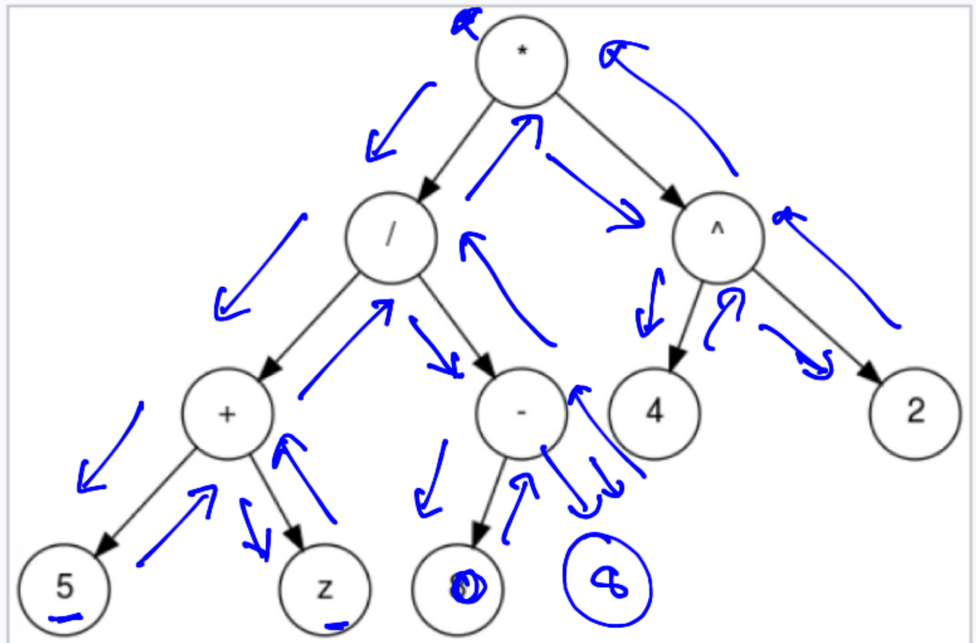
Euler Tour Traversal

- Generic traversal of a binary tree
- “walk around” the tree and visit each node three times
  - on the left
  - from below
  - on the right
- The preorder, inorder, and postorder traversals special cases



Example: Printing an Arithmetic Expression

- An Euler's walk of the tree
- Print '(' before visiting the subtree
- Visit the left sub-tree
- Print value of the node
- Visit the right sub-tree
- Print ')' after visiting the right sub-tree



Binary algebraic expression tree equivalent to ((5 + z) / -8) * (4 ^ 2)

(((5 + z) / (0 - 8)) * (4 ^ 2))

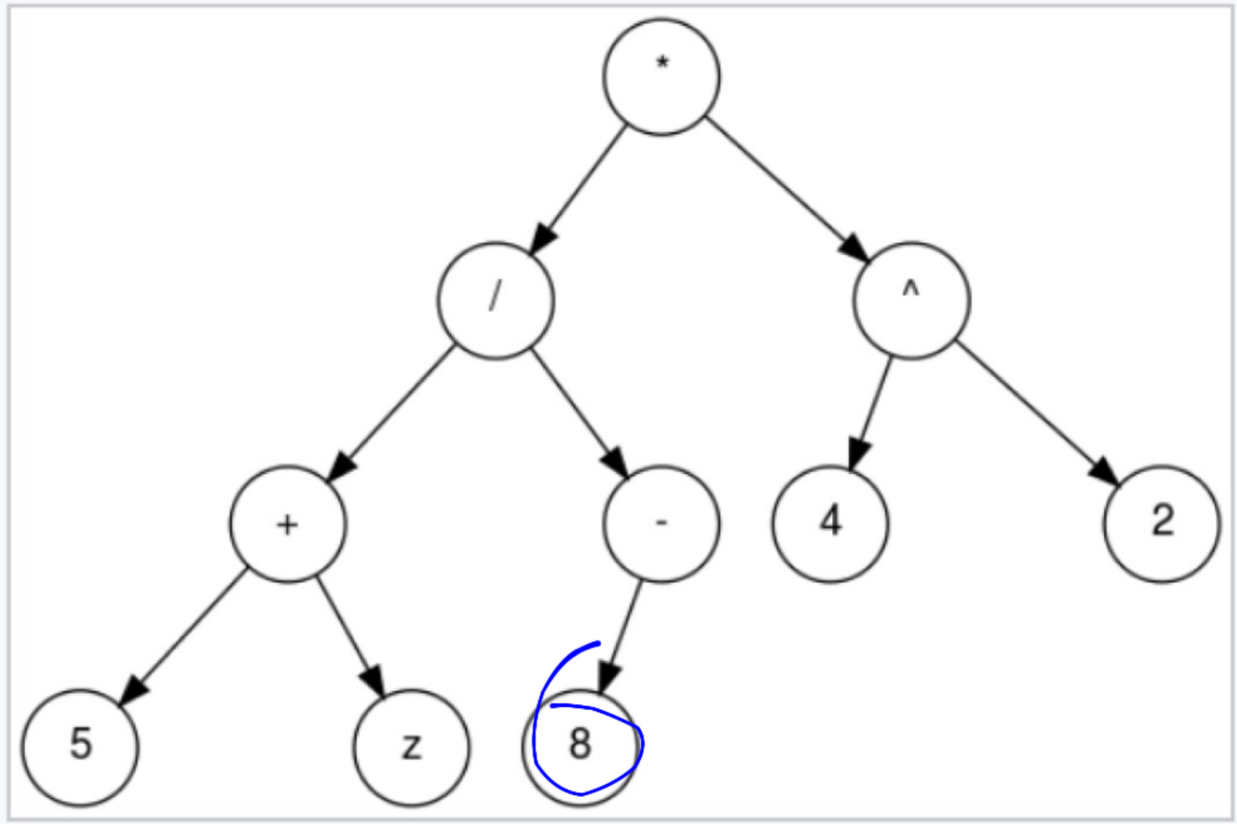
Image source: Wikipedia

Expression Tree Traversal

```
Traverse (T)
    if (T == NULL) return
    if (T->left == Null) && (T->Right == Null)
        Print T.value;
        return
    else
        Print "("
        Traverse (T-> left)
        print T.value
        Traverse (T-> right)
        print ")"
        return
```

Example: Printing an Arithmetic Expression

- An Euler's walk of the tree
- Print '(' before visiting the subtree
  - Visit the left sub-tree
  - Print value of the node
  - Visit the right sub-tree
- Print ')' after visiting the right sub-tree



Binary algebraic expression tree equivalent to ((5 + z) / -8) * (4 ^ 2)

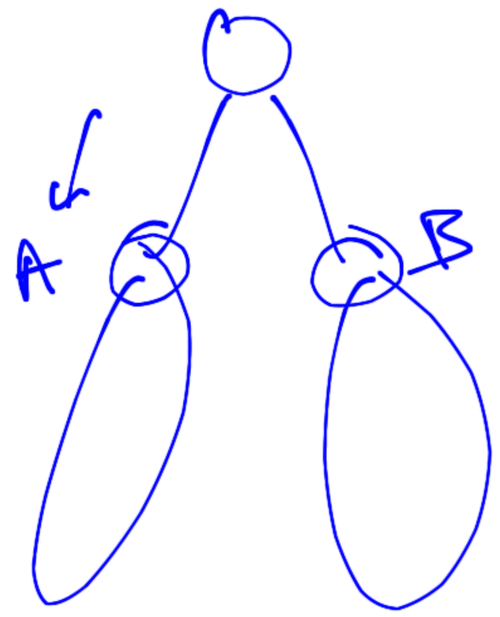
((((5)+(z))/(-8)))*((4)^(2))))

Image source: Wikipedia

Pre-order (T)

```
Traverse (T)  Tree U;
    stack S;
    if (T==NULL) return
    push (T)
    while Not empty(S)
        U = pop (S)
        visit (U)
        push (U.right)
        push (U.left)
    end while
```

```
Traverse (T)
    Visit (T)
    Traverse (T->left)
    Traverse (T->Right)
```



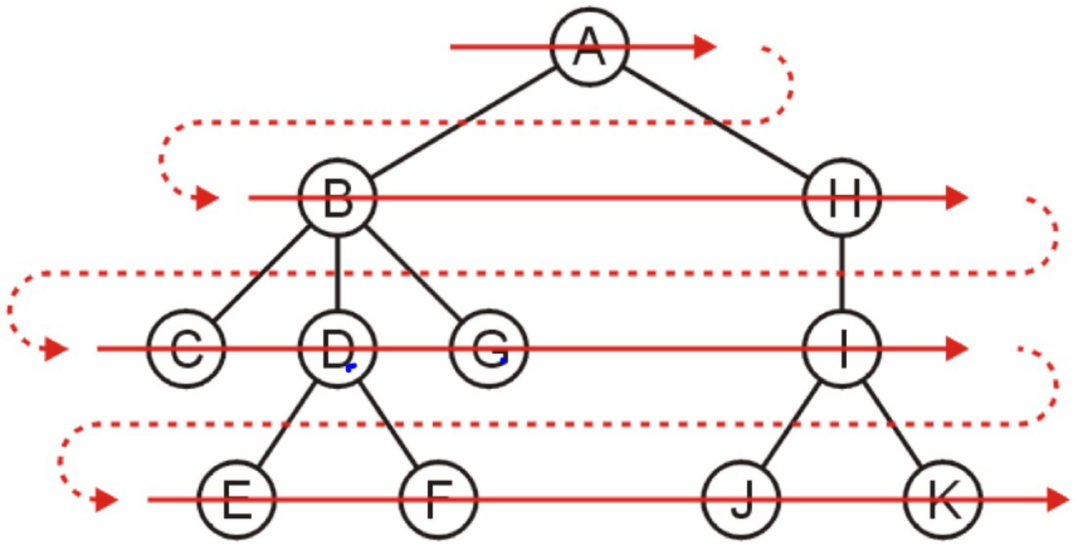
Traversing Iteratively: Pre-order

```
Pre-order(v)
        push(v)
        while (stack not empty) do
            v = pop()
            if (v == NULL) break
            visit(v)
            push(v.right)
            push(v.left)
end while
```

Homework

- Write algorithm for in-order and post-order traversals using stack
- Given a fully bracketed expression, write an algorithm to create its expression tree

Breath First Traversal



- Visits all the nodes at the same depth in sequence
- Starts at the root node and goes to higher depths
- Can be implemented using a queue
  - Takes $\Theta(n)$ time
  - Space is also $\Theta(n)$
  - Order: A B H C D G I E F J K

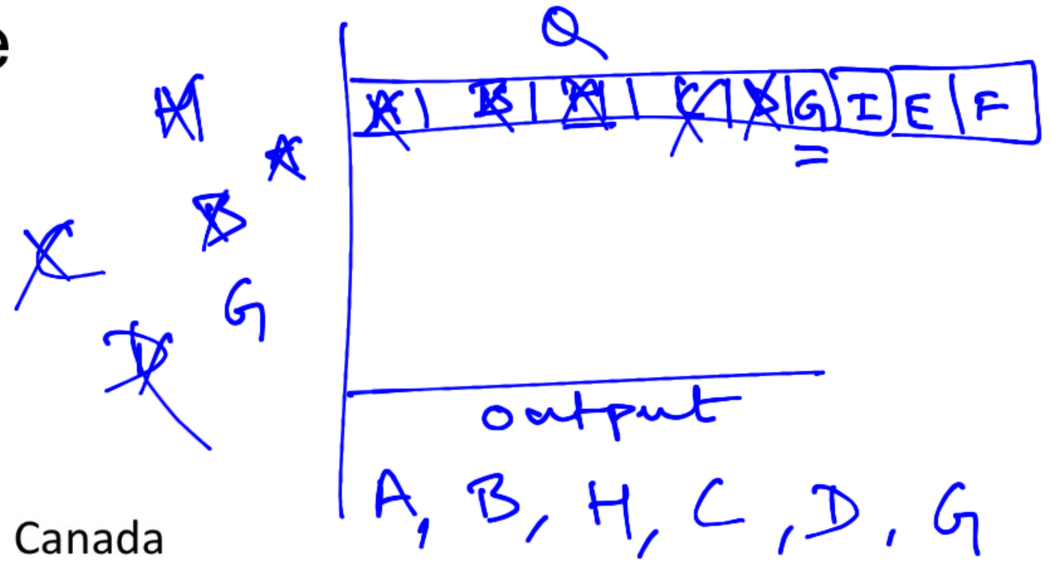


Figure courtesy: Douglas Wilhelm Harder, University of Waterloo, Canada

Breath First Traversal

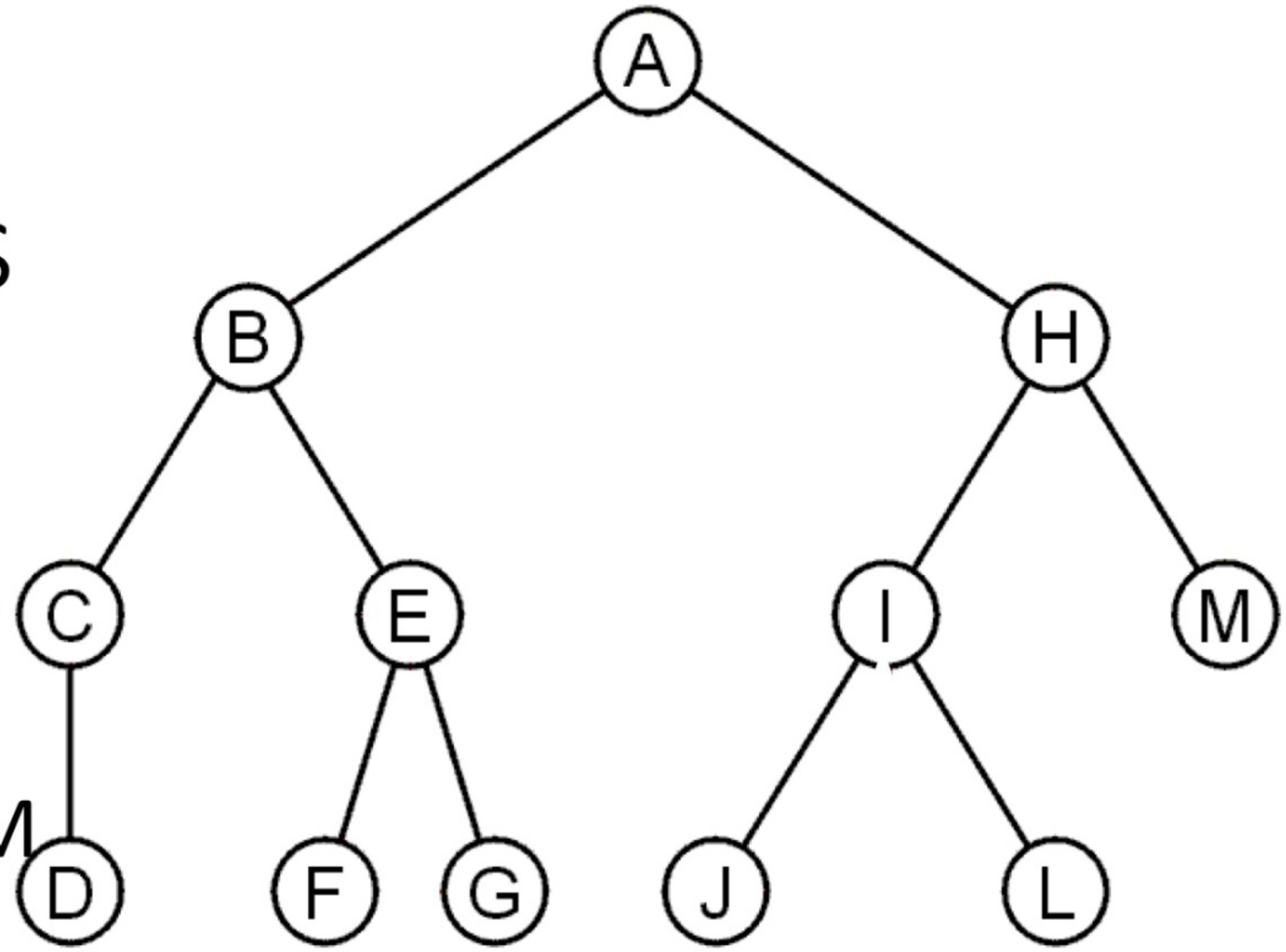
```
void BreathFirst(BinaryTree *t) {
     Queue Q;
     Q.enqueue(t);
     while (not empty Q) {
          t = Q.Dequeue();
          visit(t);
          if (t.left != 0) Q.enqueue(t.left);
          if (t.right != 0) Q.enqueue(t.right);
     }
}
```


I CAN TRAVERSE TREE.
BFS or DFS ?
imgflip.com

Counting Trees

- How many trees with N-nodes have a given a pre-order traversal?

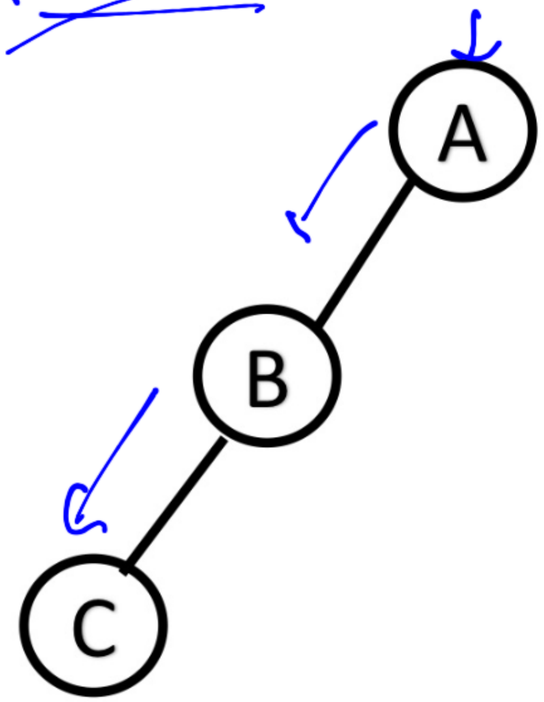
A, B, C, D, E, F, G, H, I, J, L, M

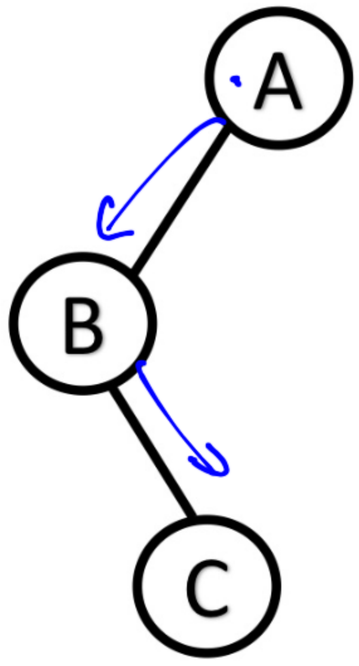


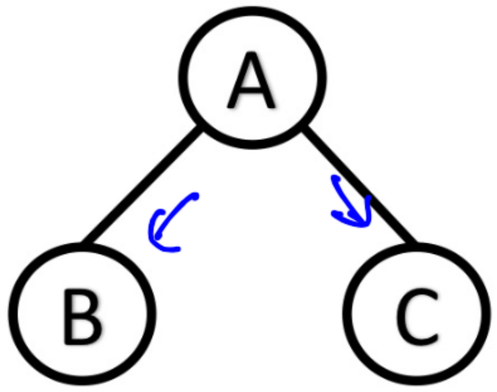
Counting Trees

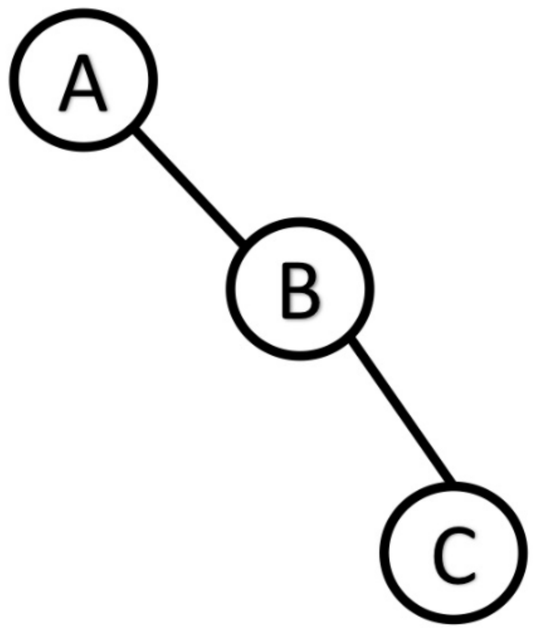
- How many trees with N-nodes have a given a pre-order traversal?

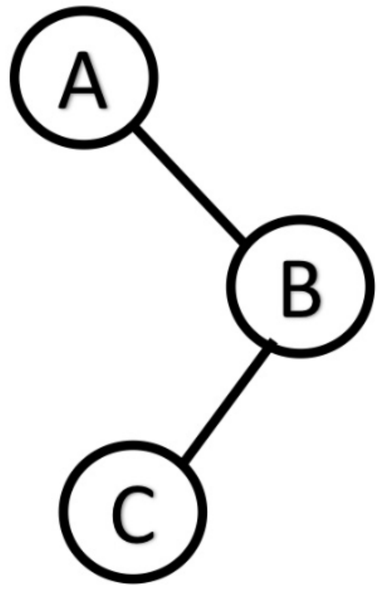
A, B, C





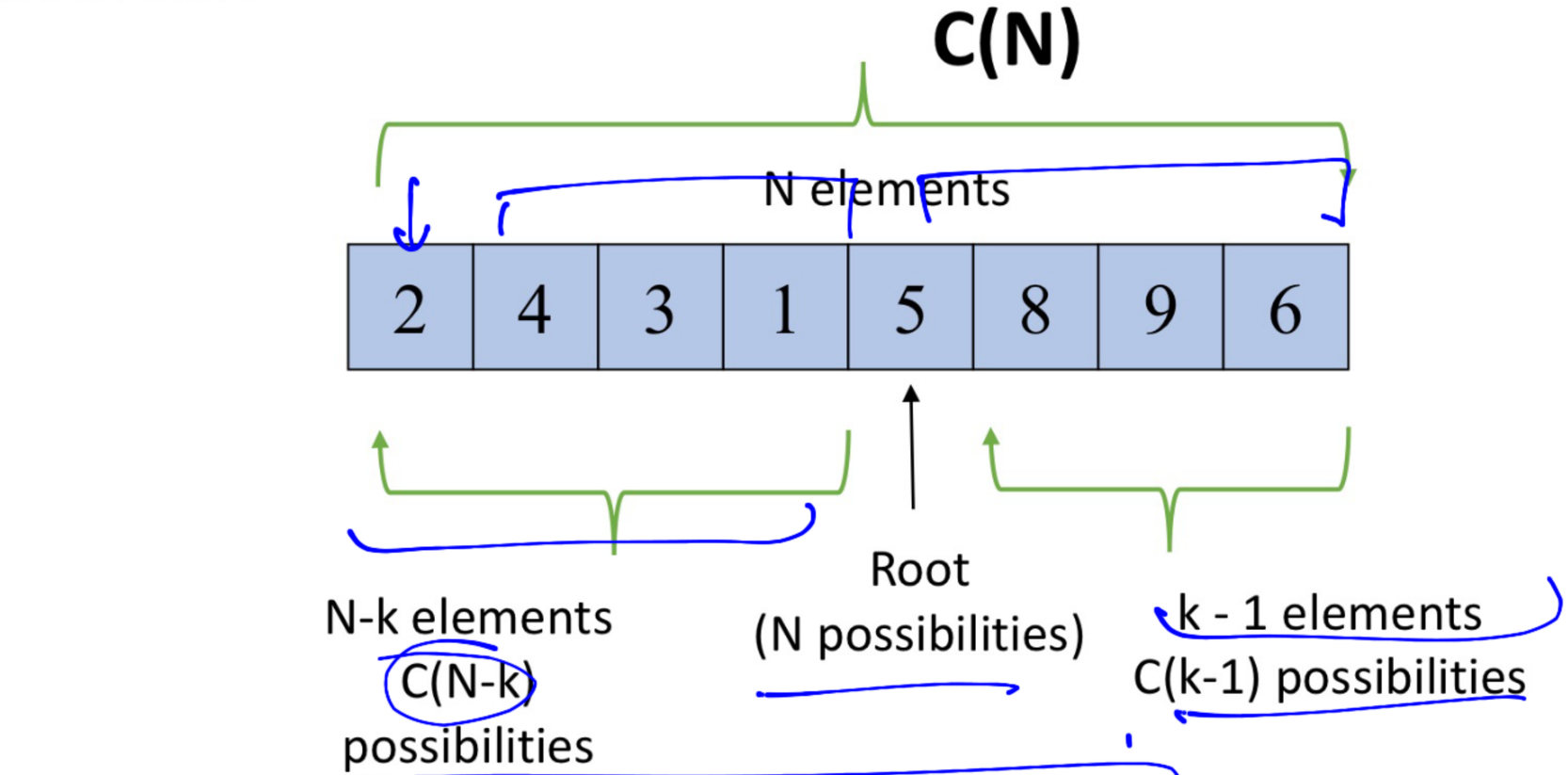






Counting Trees

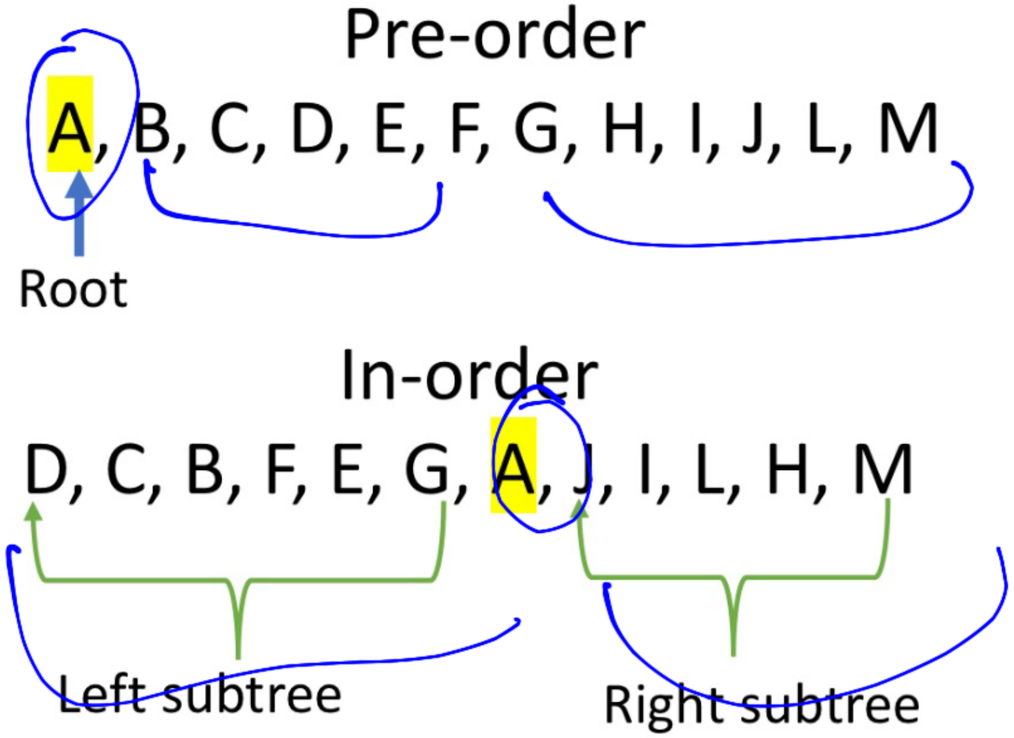
- How many trees with N-nodes have a given a ~~pre-order~~ in-order traversal?



$$C(N) = \sum_{k=0}^{N-1} C(N-k)\, C(k-1)$$

Reconstructing Tree

- How many trees with a given pre-order and in-order traversal can you construct?



Homework

How many trees with a give post-order and in-order traversal can you construct?

Given a post-order and pre-order traversal of a binary tree, can you uniquely reconstruct the tree? Prove if you can, give counter example if you can’t.

Answer the above question if every internal node had exactly two children.

Thank You