More Binary Trees

- Recursive Program to find sibling of node, given the tree and the node key

- Write a program to count the number of leaf nodes in a binary tree.

BST

- Write a program to insert a new node to Binary Search Tree (BST)
- Main() {
- insertBST(root, val) // root = newnode{
- }
- insertBST(node, val) {
- while (true) {
- if (node.val < val) {
- if (node.right == null) { insert new node……node.right = nd.; return }
- node = node.right;
- }
- if (node.val > val) {
- if (node.left == null) { insert ……node.left = nd; return }
- node = node.left;
- }

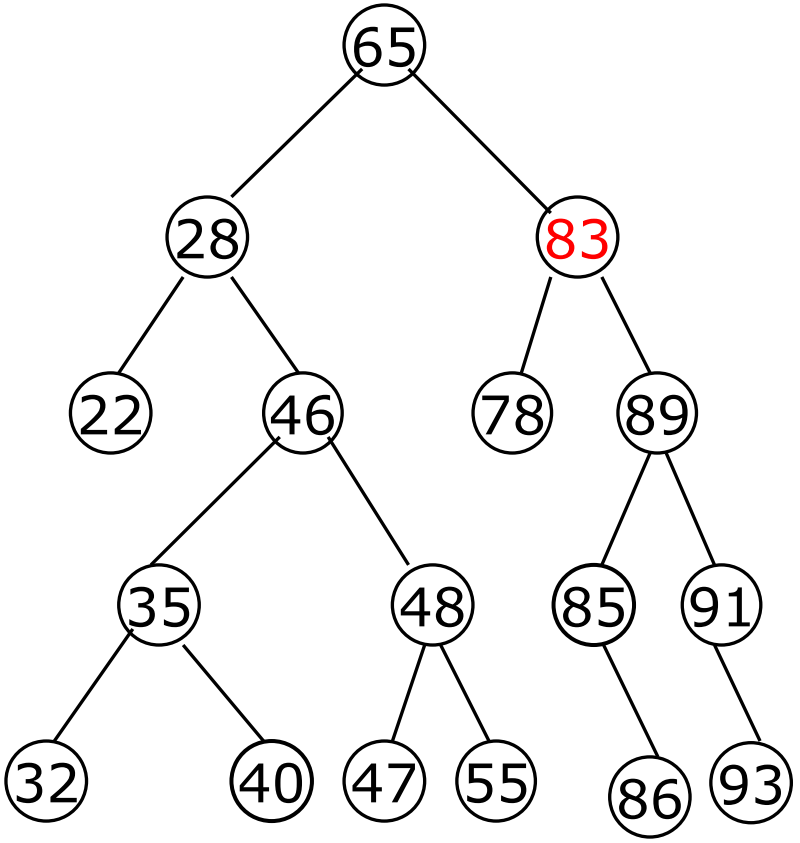
BST

- Write a program to insert a new node to Binary Search Tree (BST)
- Main() {
- rinsertBST(root, val){
- }
- rinsertBST(node, val) {
- if (node.val < val) {
- if (node.right == null) { insert …….; return; }
- else { return rinsertBST(node.right, val); }
- }
- if (node.val > val) {
- if (node.left == null) { insert ……; return; }
- else { return rinsertBST(node.left, val); }
- }

More Binary Trees

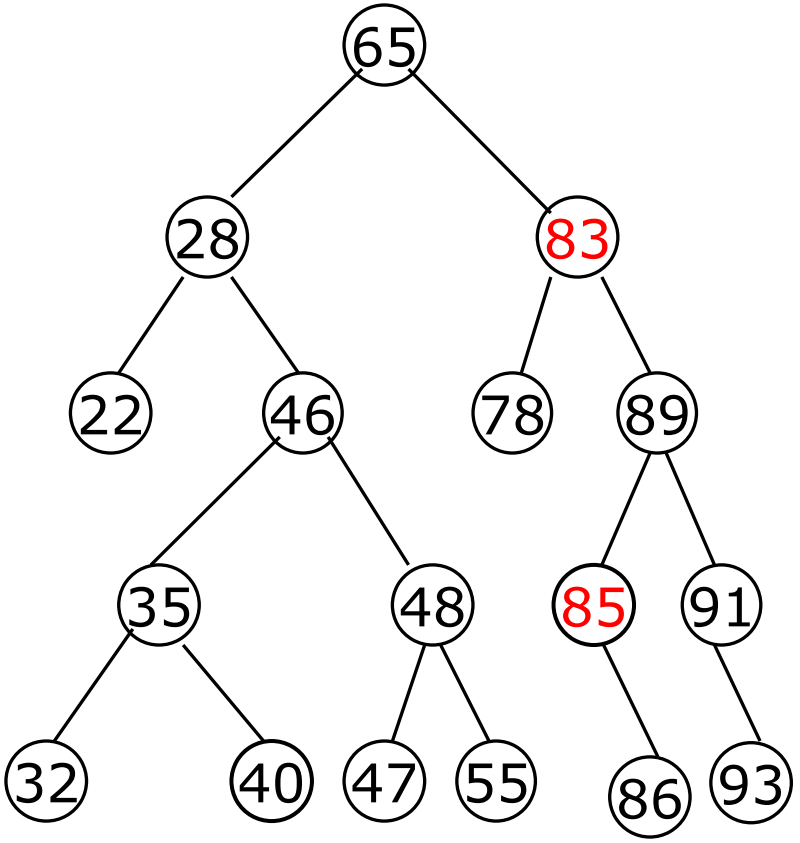
- Program to delete node from BST

- Delete 83



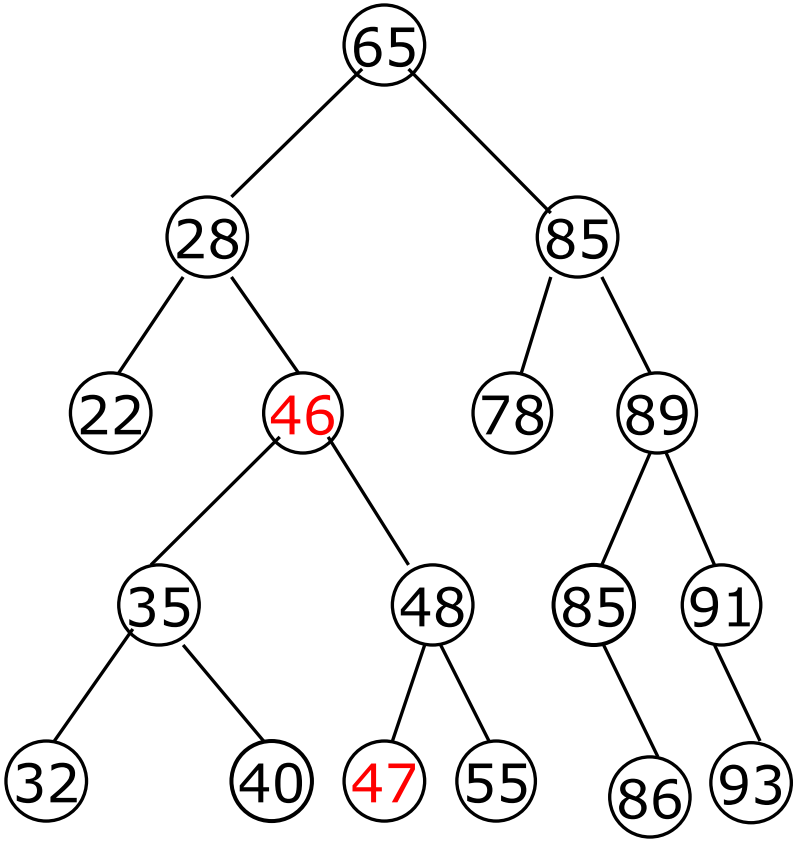
More Binary Trees

- Program to delete node from BST
- Delete 83



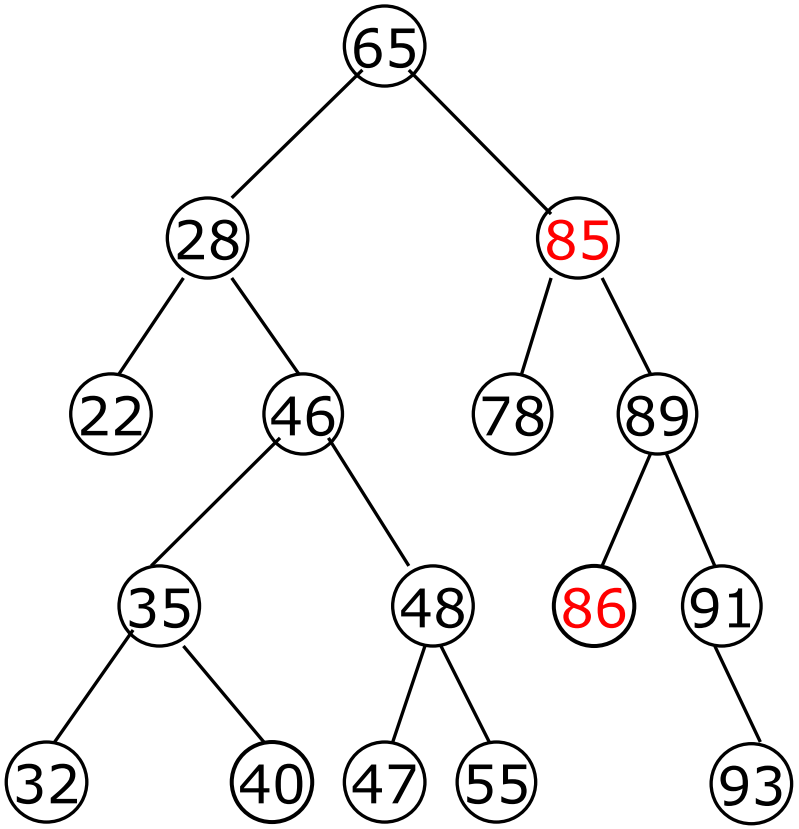
More Binary Trees

- Program to delete node from BST
- Delete 83



More Binary Trees

- Program to delete node from BST
- Delete 83



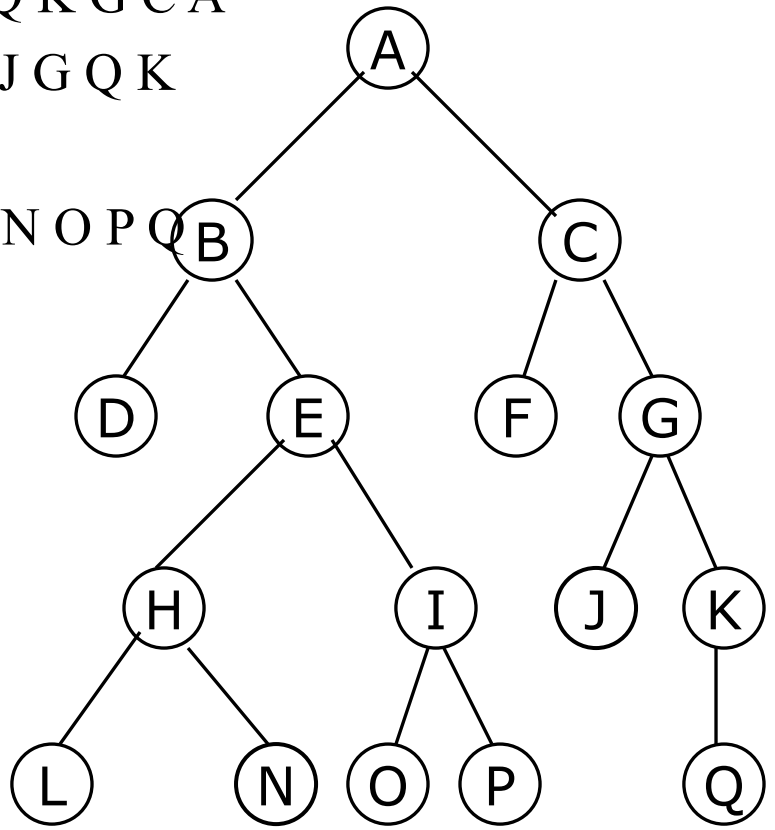
More Binary Trees

- DFS – Depth First Search
  - Visit Node, left subtree, right subtree
- BFS – Breadth First Search
  - Visit each node at the same level, then go down a level

More Binary Trees

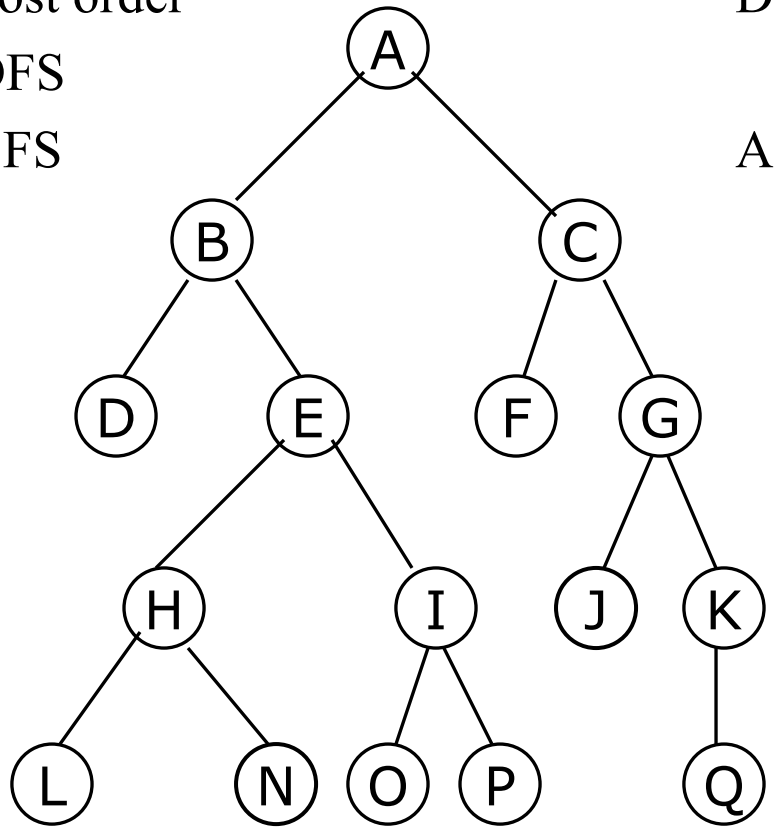
- Pre Order A B D E H L N I O P C F G J K Q
- Post order D L N H O P I E B F J Q K G C A
- Inorder D B L H N E O I P A F C J G Q K

- BFS A B C D E F G H I J K L N O P Q



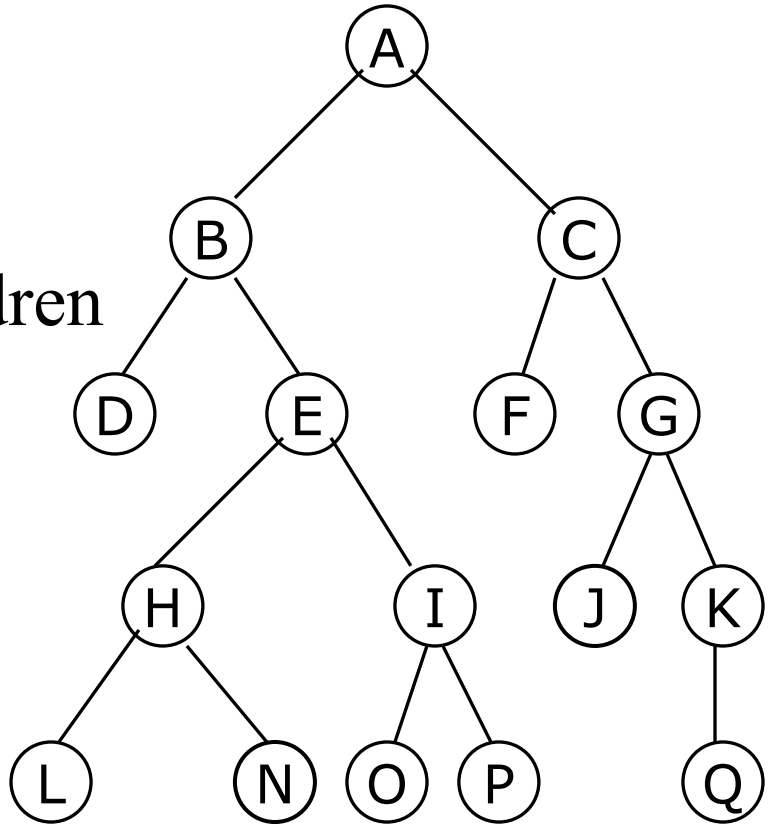
More Binary Trees

- Pre Order Traversal — A B D E H L N I O P C F G J K Q
- Post order — D L N H O P I E B F J Q K G C A
- DFS
- BFS — A B C D E F G H I J K L N O P Q



More Binary Trees

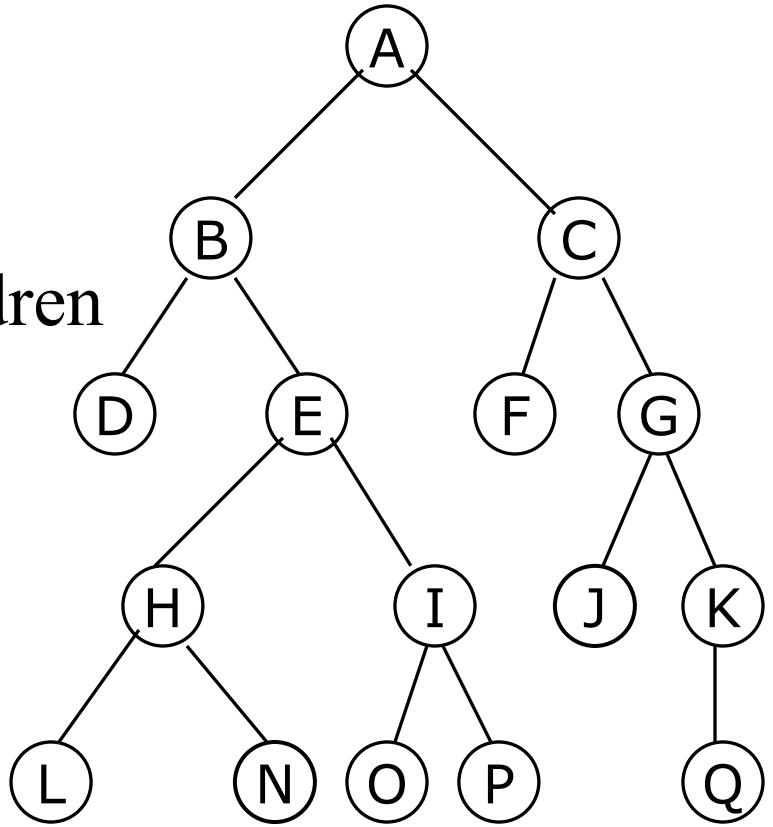
- BFS ; not recursive ; why ?
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while



More Binary Trees

- BFS Visited = {}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

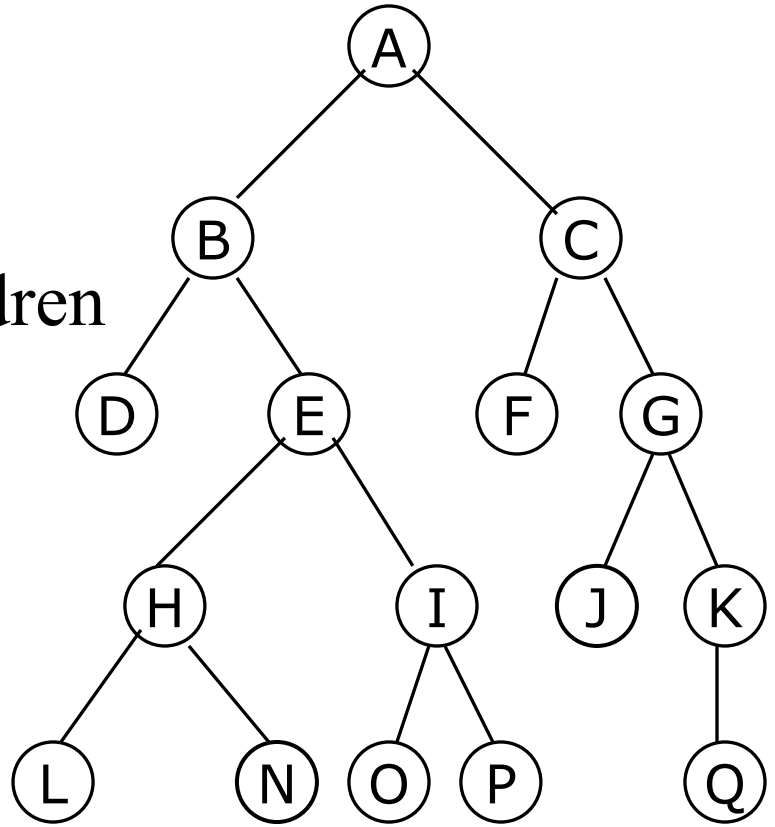
Q = {}



More Binary Trees

- BFS Visited = {}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

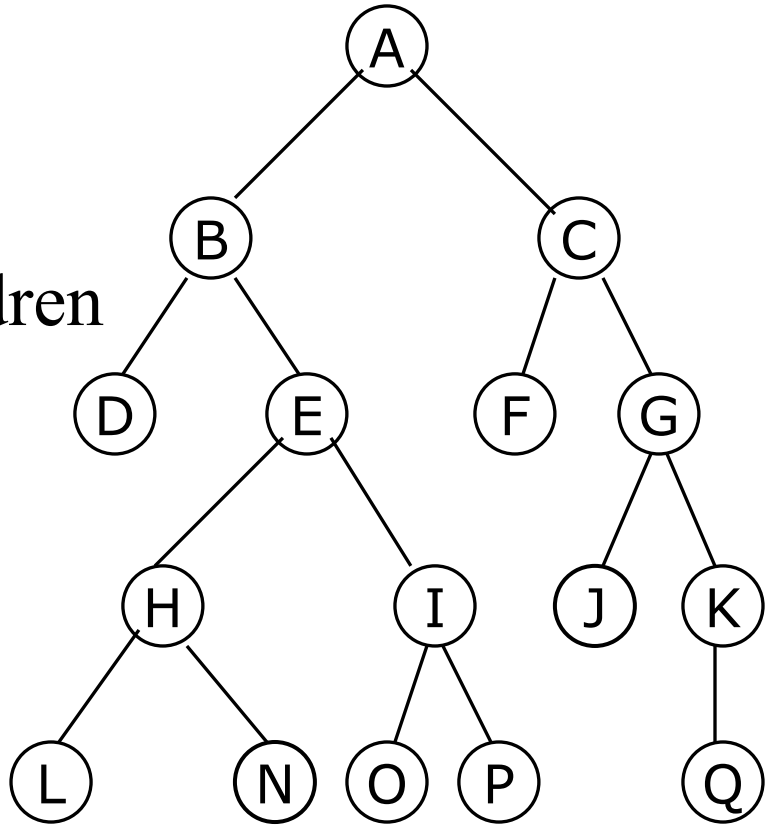
Q = {A}



More Binary Trees

- BFS Visited = {A}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

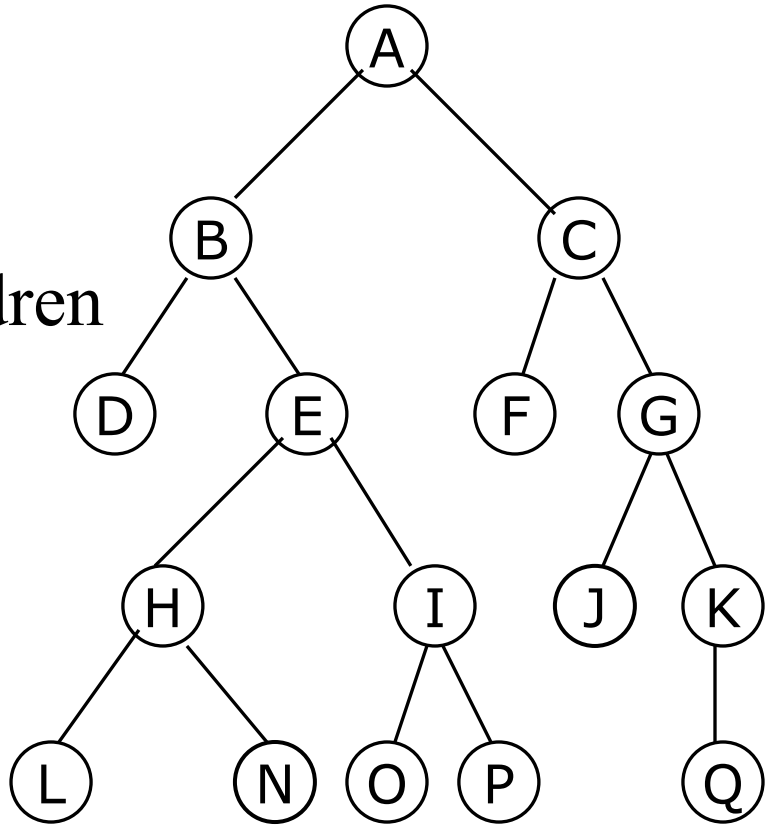
Q = {}



More Binary Trees

- BFS Visited = {A}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

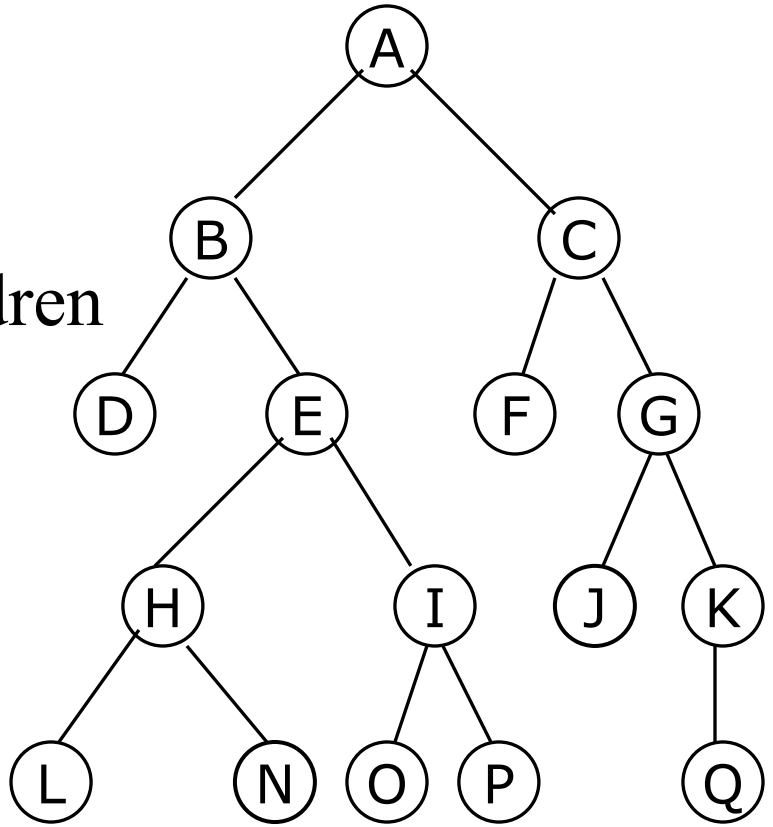
Q = {B}



More Binary Trees

- BFS Visited = {A}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

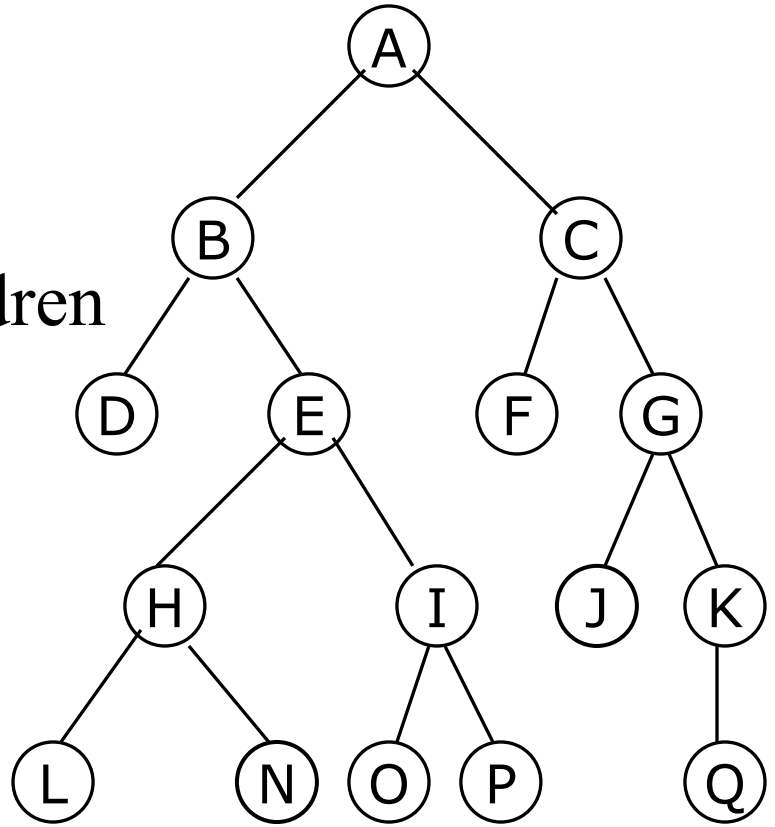
Q = {C, B}



More Binary Trees

- BFS Visited = {A,B}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

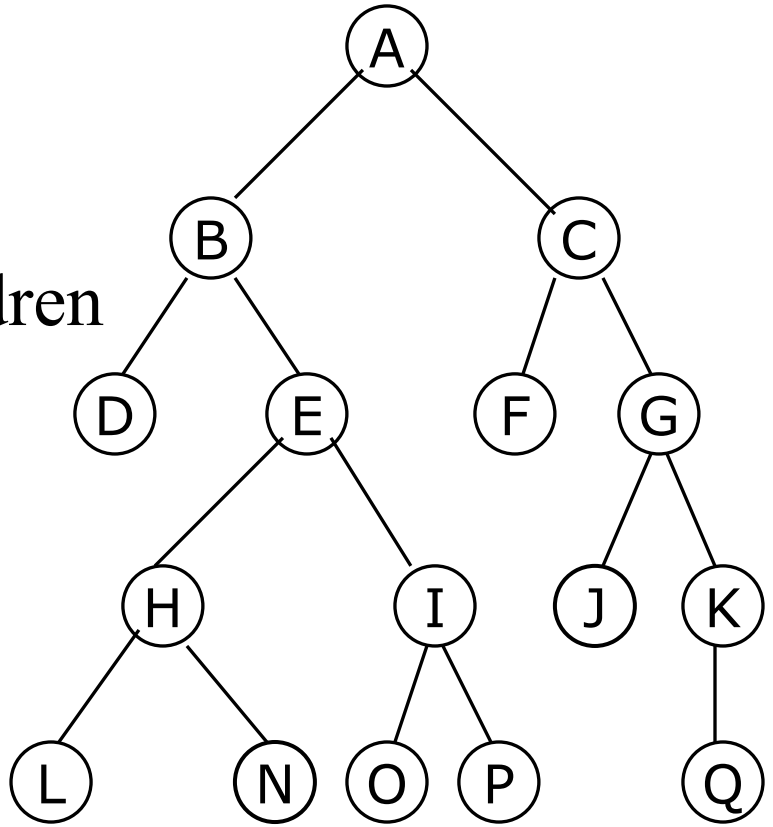
Q = {C}



More Binary Trees

- BFS Visited = {A,B}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

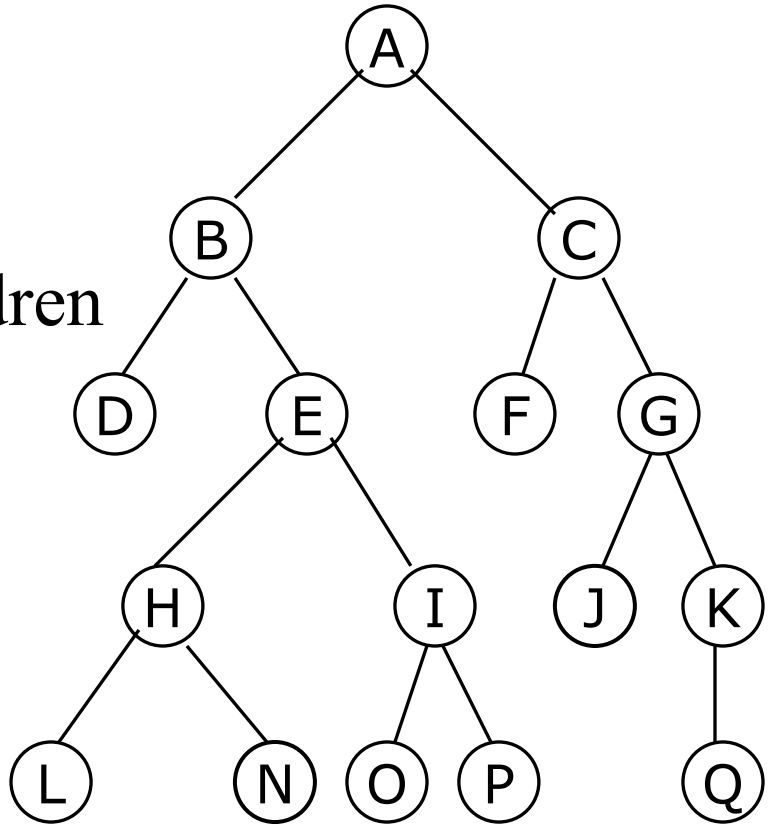
Q = {D,C}



More Binary Trees

- BFS Visited = {A,B}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

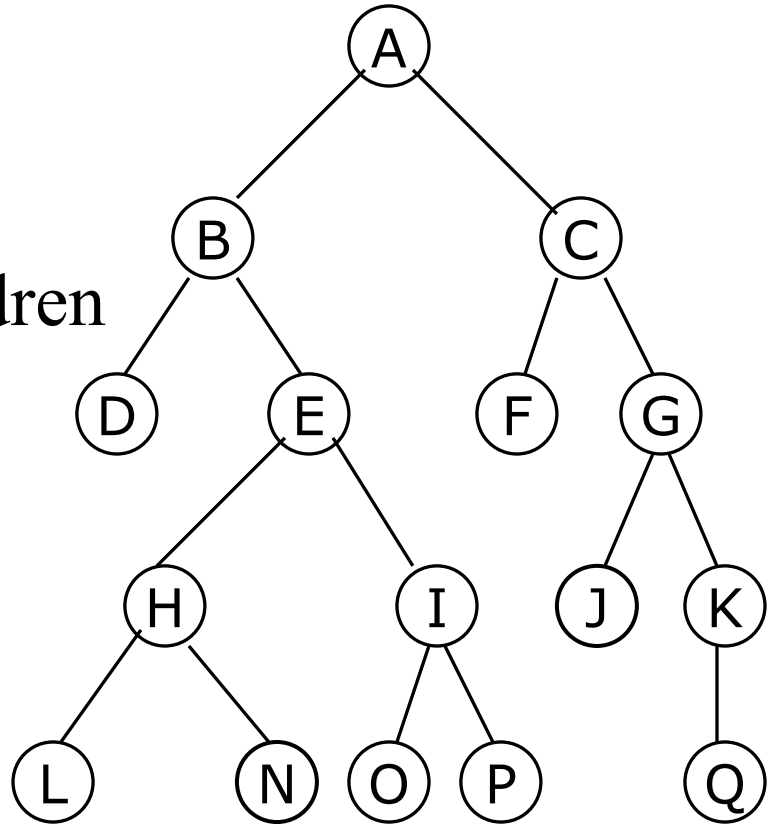
Q = {E,D,C}



More Binary Trees

- BFS Visited = {A,B,C}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

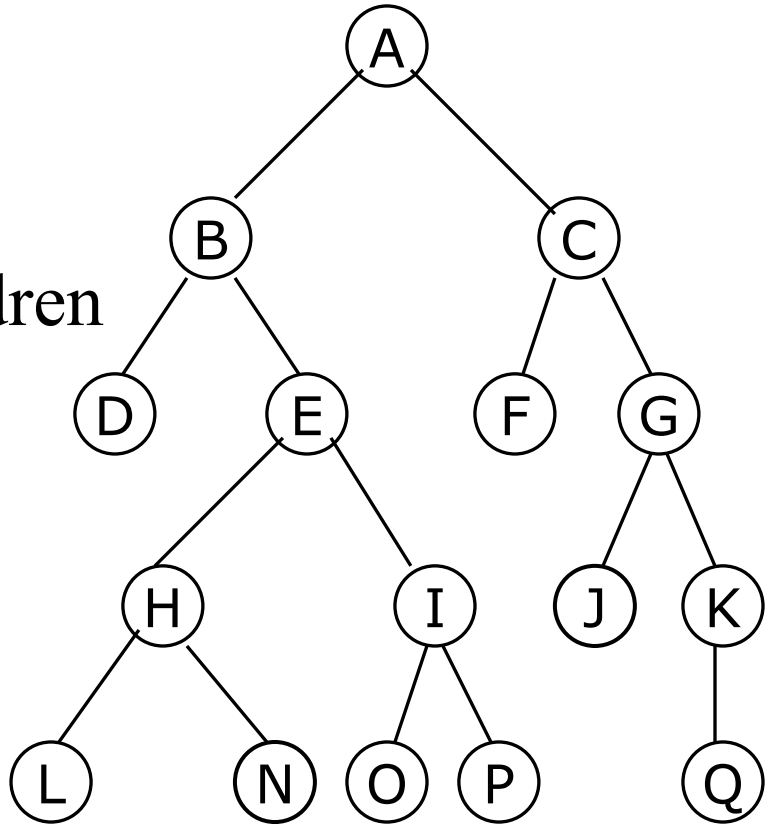
Q = {E,D}



More Binary Trees

- BFS Visited = {A,B,C}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

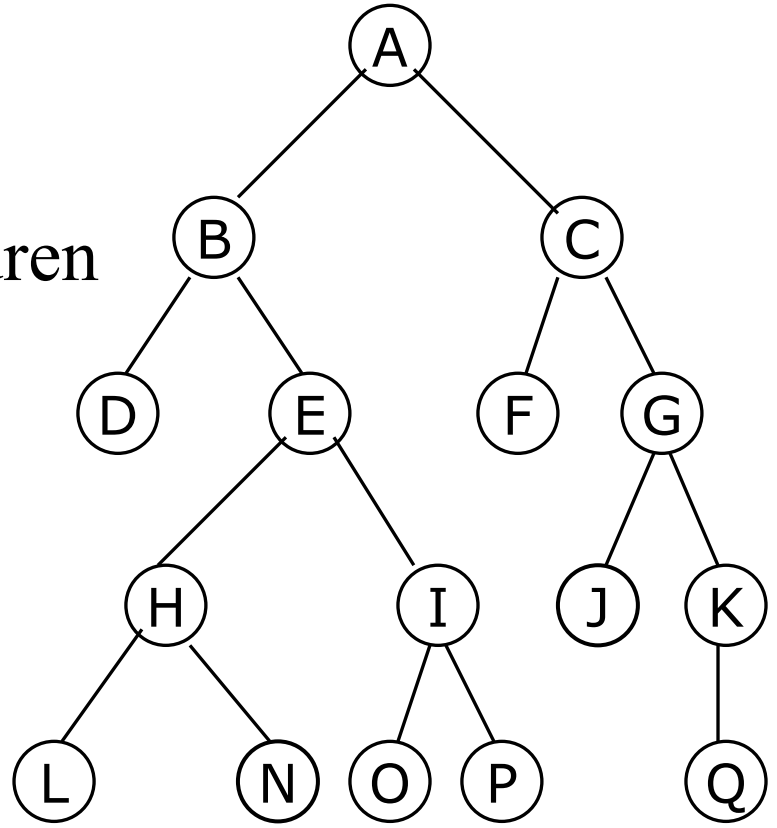
Q = {F,E,D}



More Binary Trees

- BFS Visited = {A,B,C}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

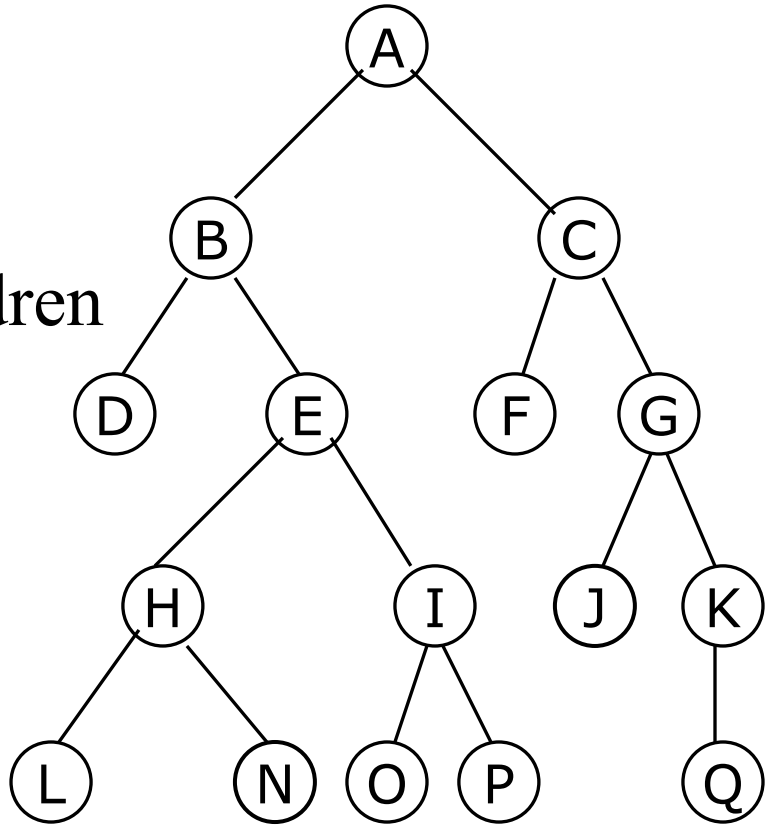
Q = {G,F,E,D}



More Binary Trees

- BFS Visited = {A,B,C,D}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

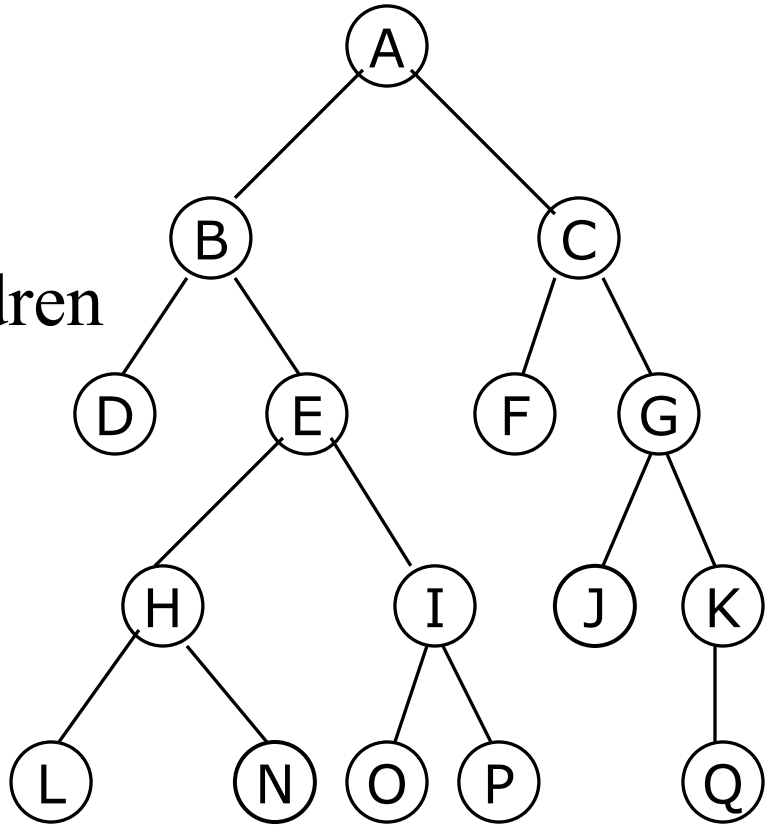
Q = {G,F,E}



More Binary Trees

- BFS Visited = {A,B,C,D,E}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

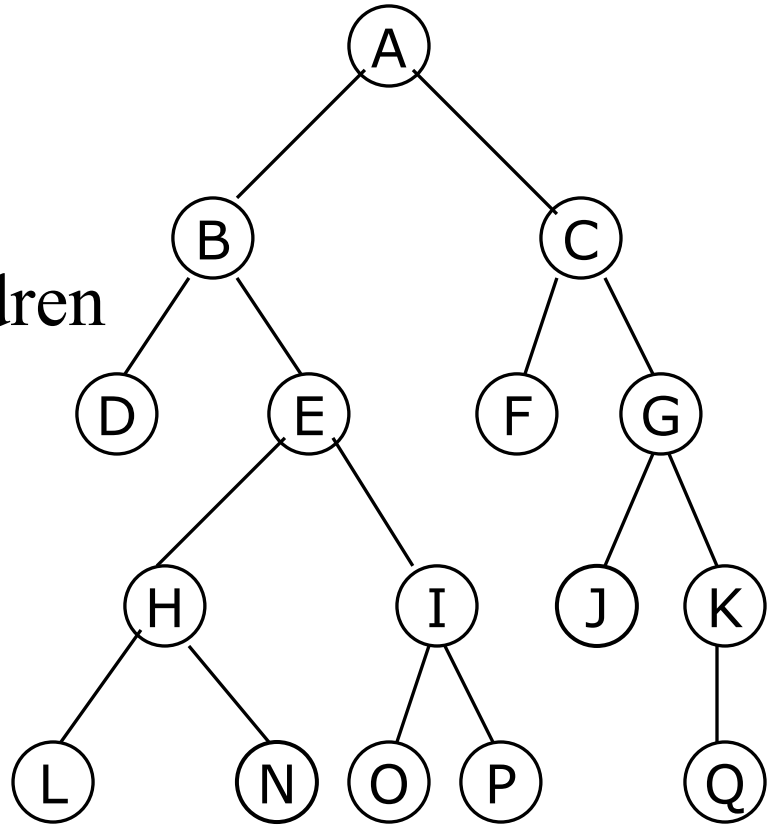
Q = {G,F}



More Binary Trees

- BFS Visited = {A,B,C,D,E}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

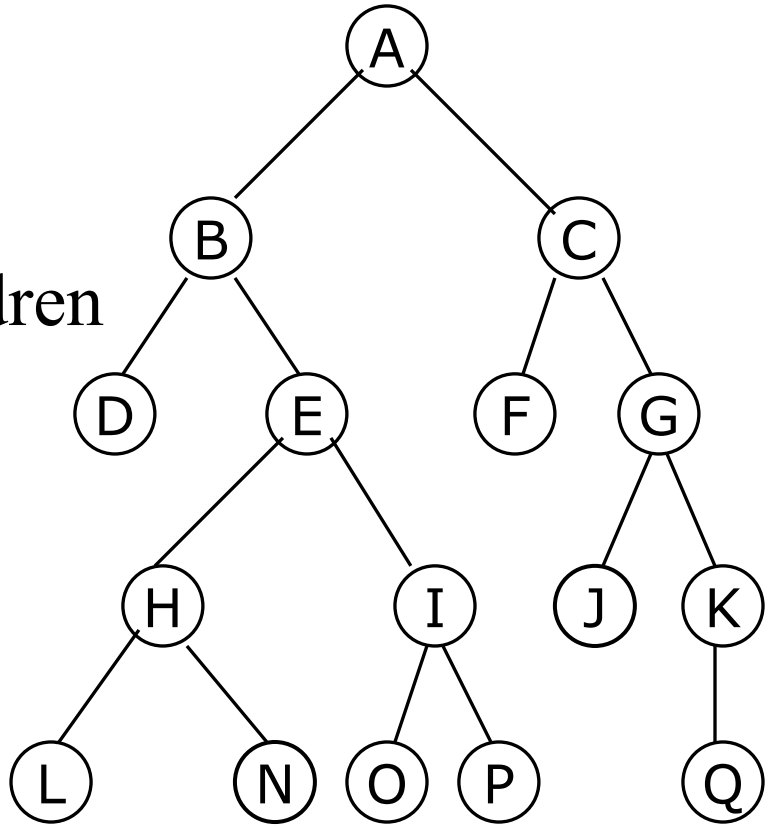
Q = {I,H,G,F}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

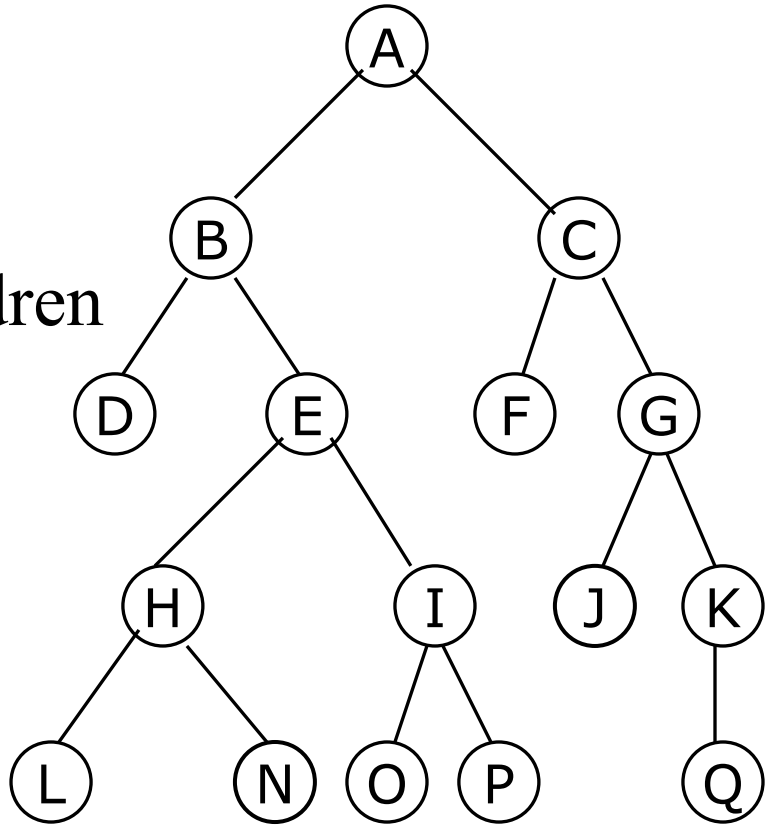
Q = {I,H,G}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

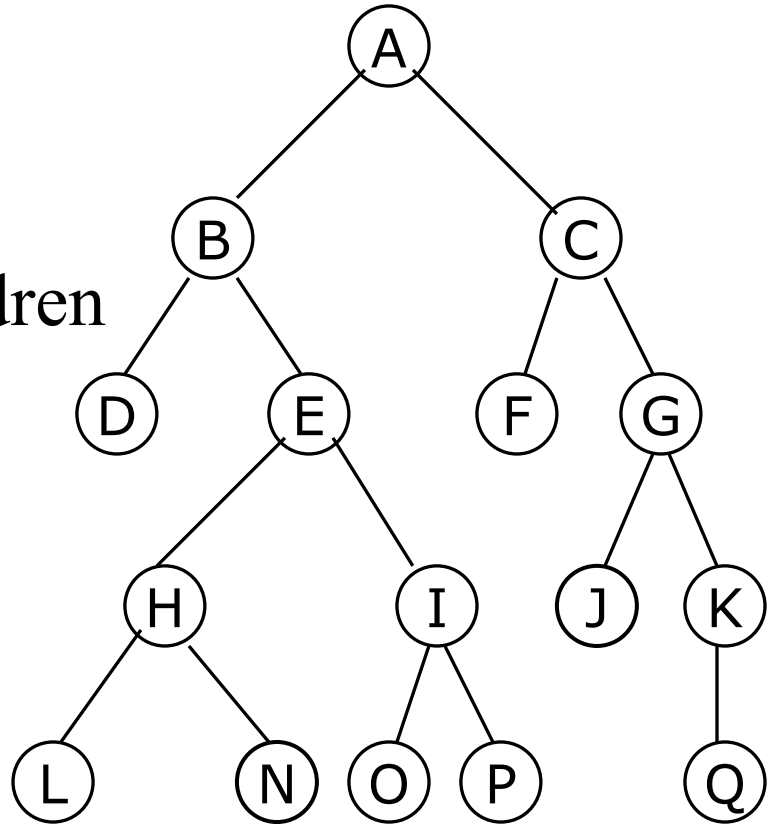
Q = {I,H}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

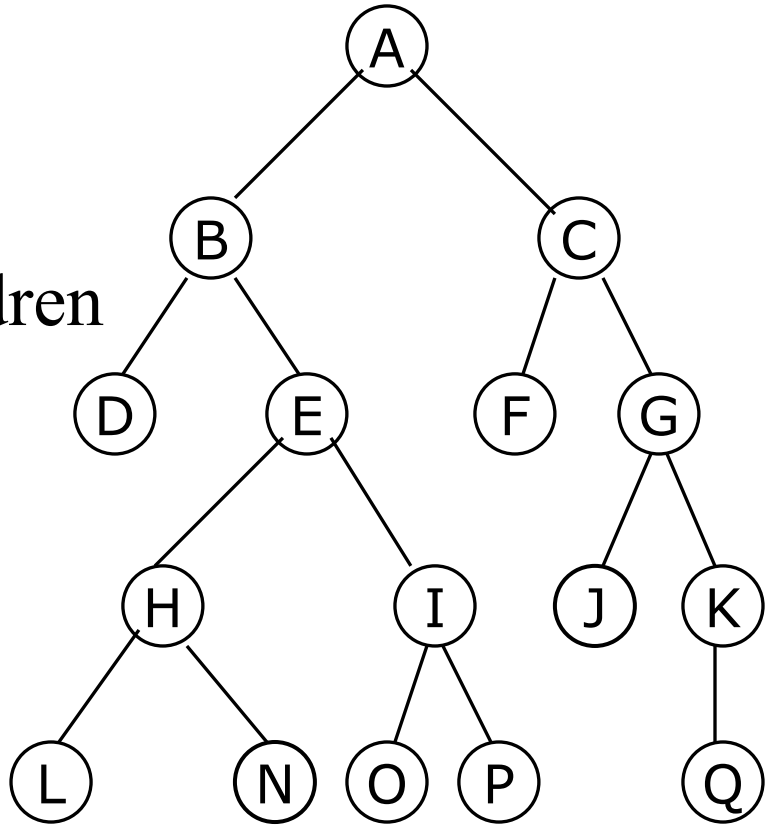
Q = {J,I,H}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

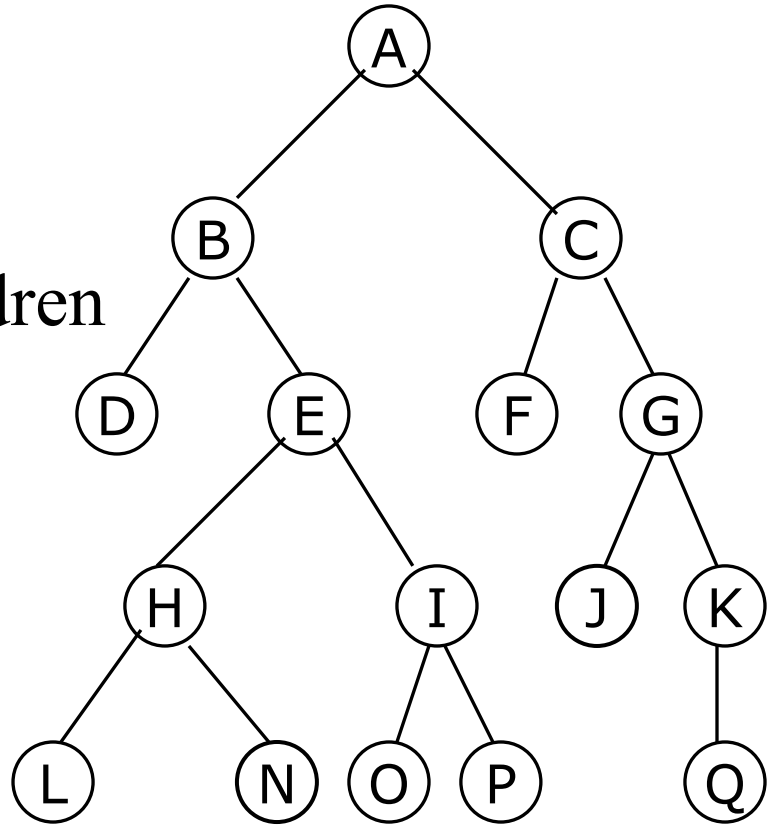
Q = {K,J,I,H}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

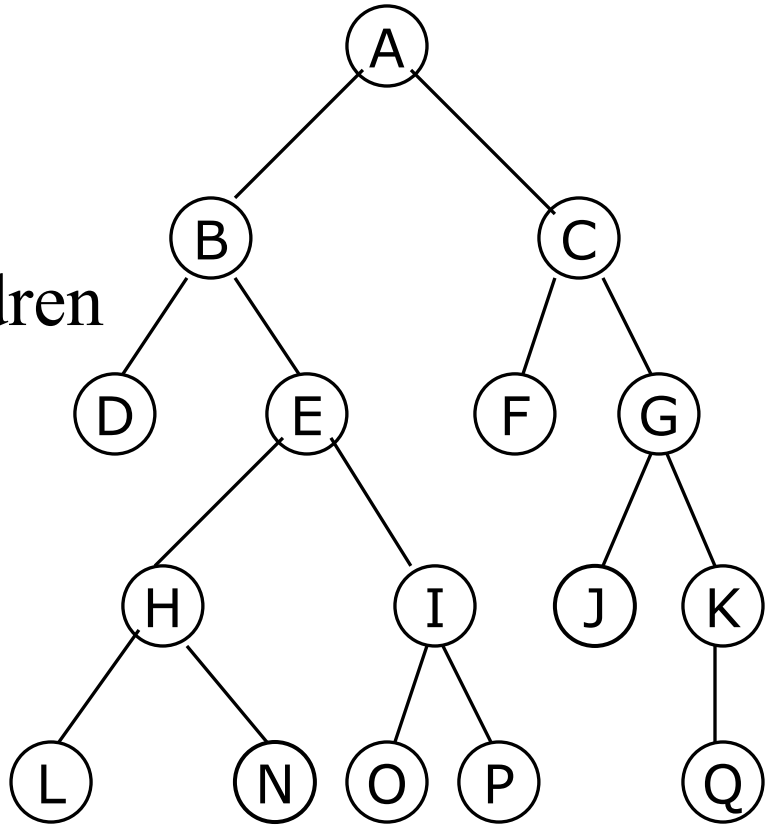
Q = {K,J,I}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

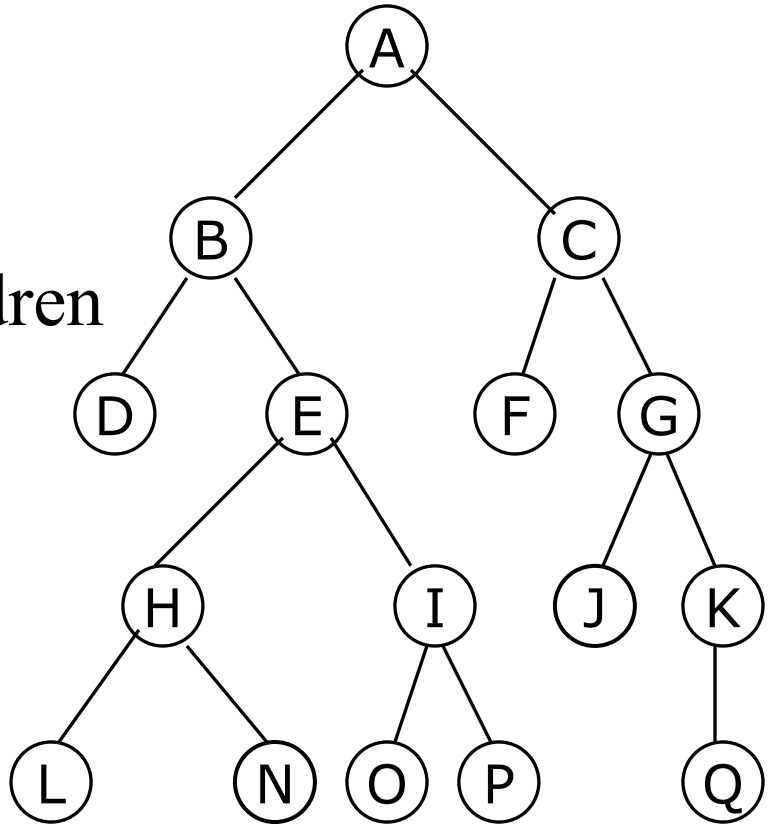
Q = {L,K,J,I}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

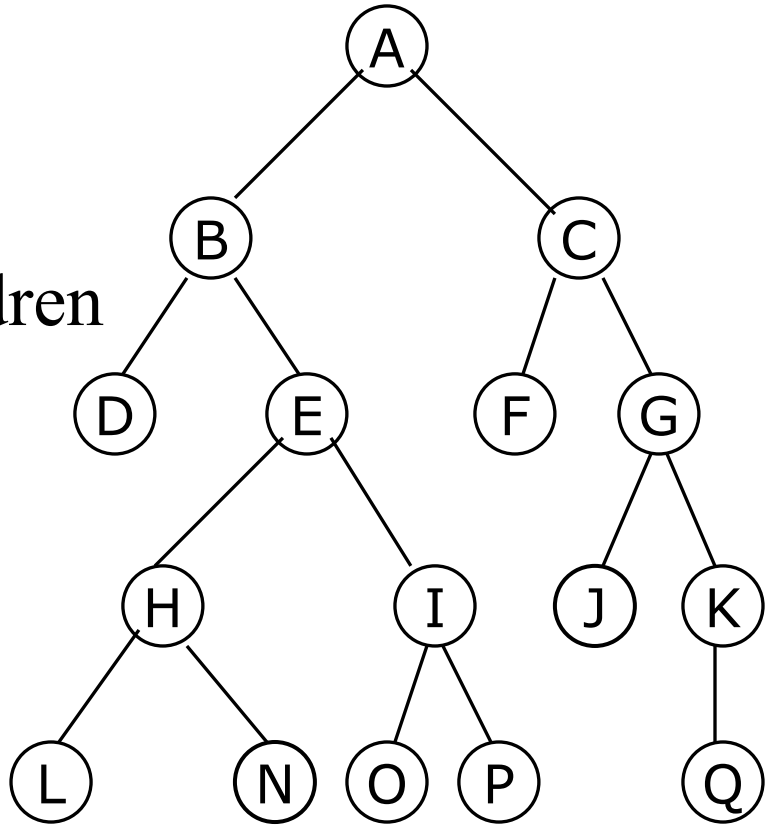
Q = {N,L,K,J,I}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H,I}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

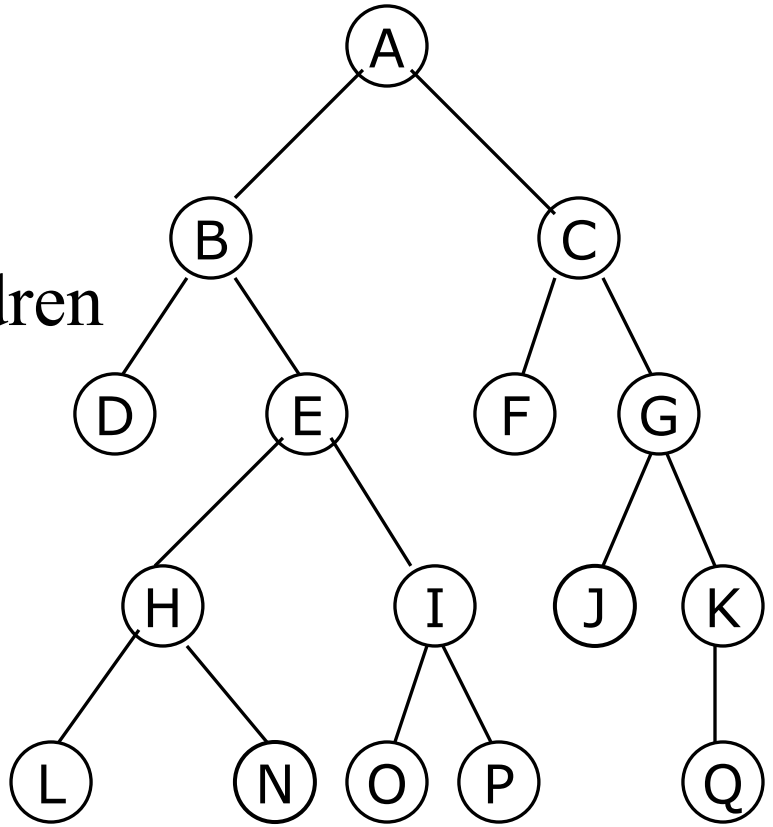
Q = {N,L,K,J}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H,I}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

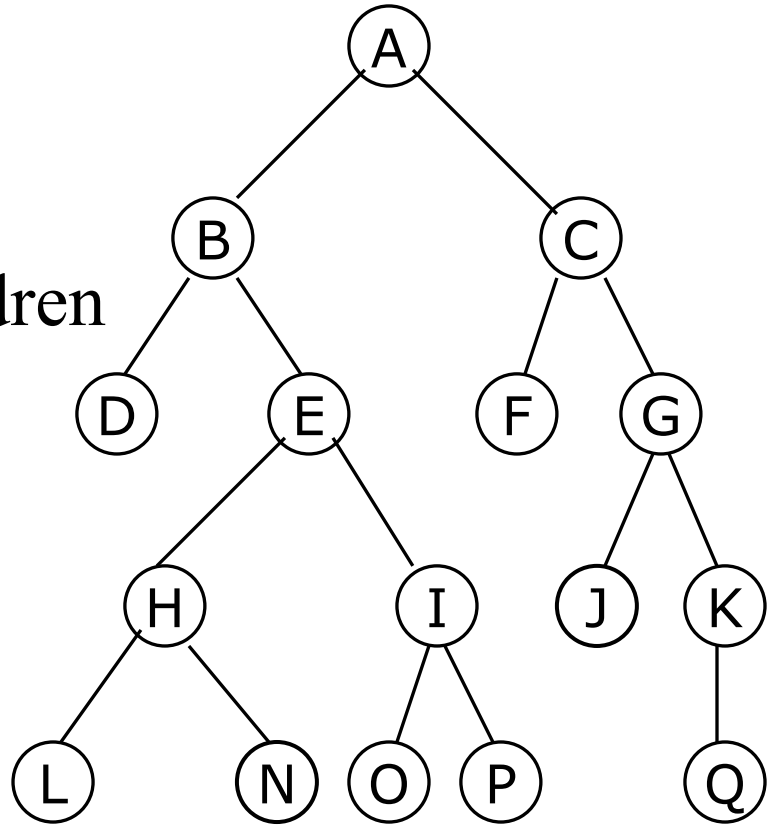
Q = {O,N,L,K,J}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H,I}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

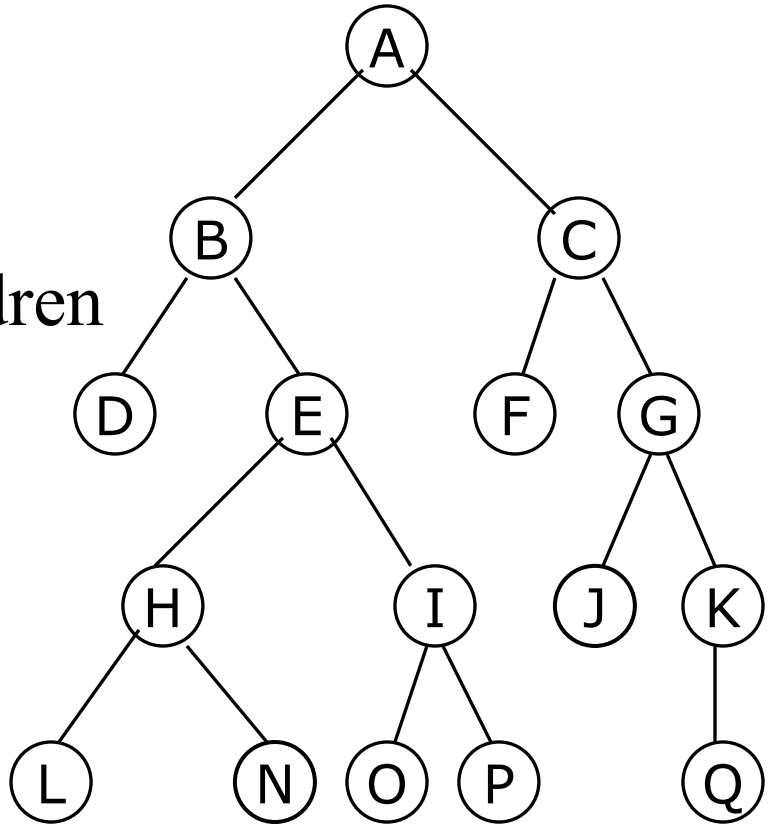
Q = {P,O,N,L,K,J}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H,I,J}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

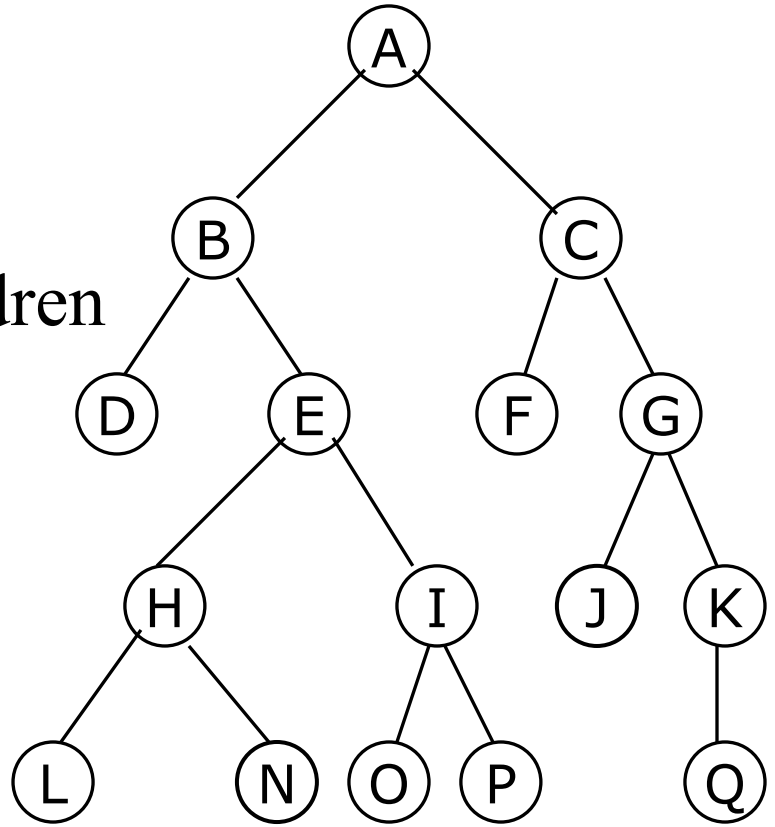
Q = {P,O,N,L,K}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H,I,J,K}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

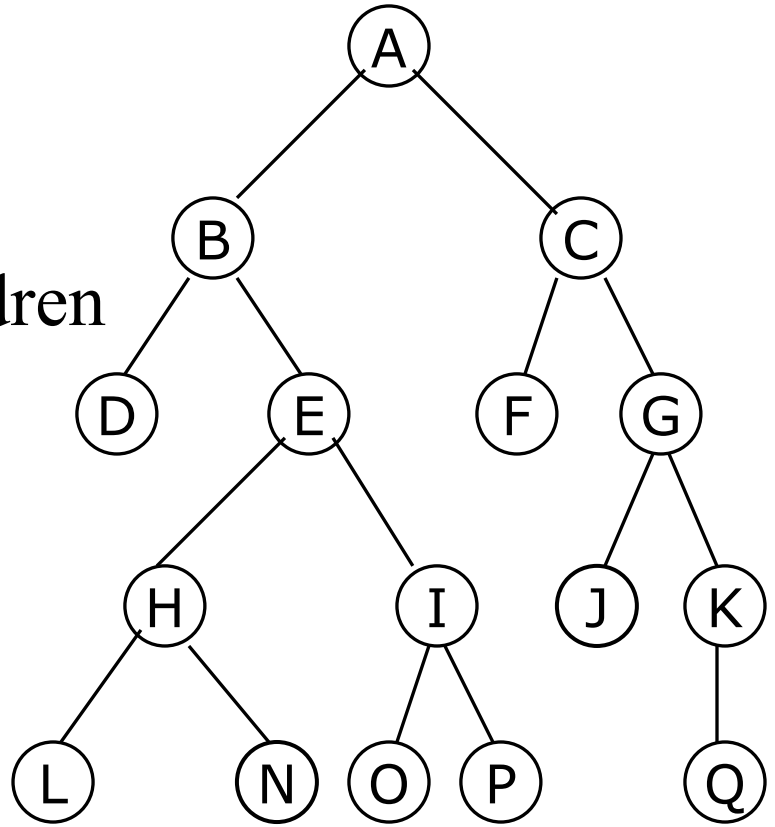
Q = {P,O,N,L}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H,I,J,K}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

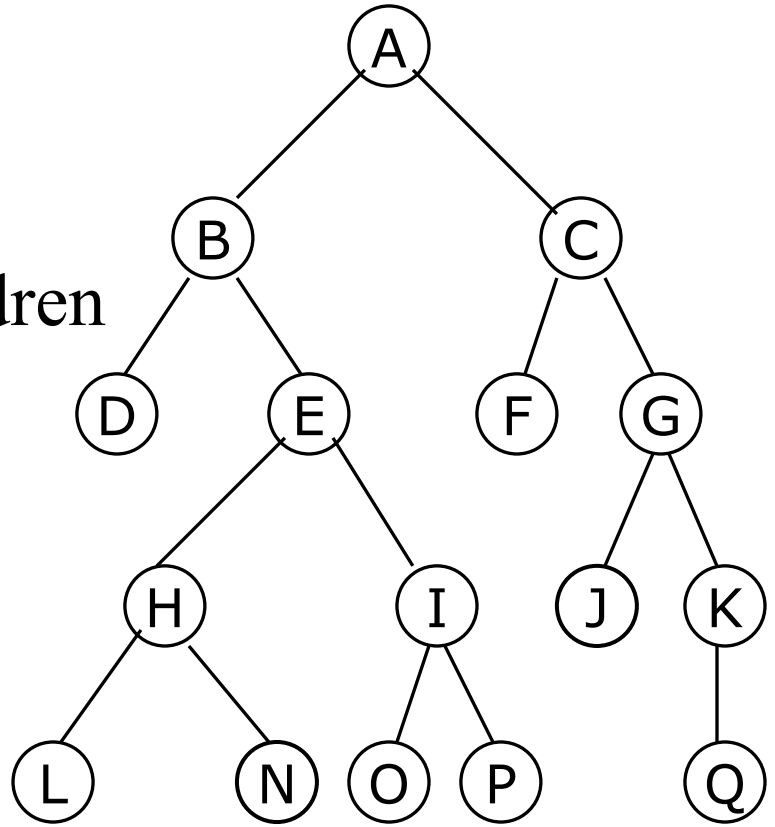
Q = {Q,P,O,N,L}



More Binary Trees

- BFS Visited = {A,B,C,D,E,F,G,H,I,J,K,L}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

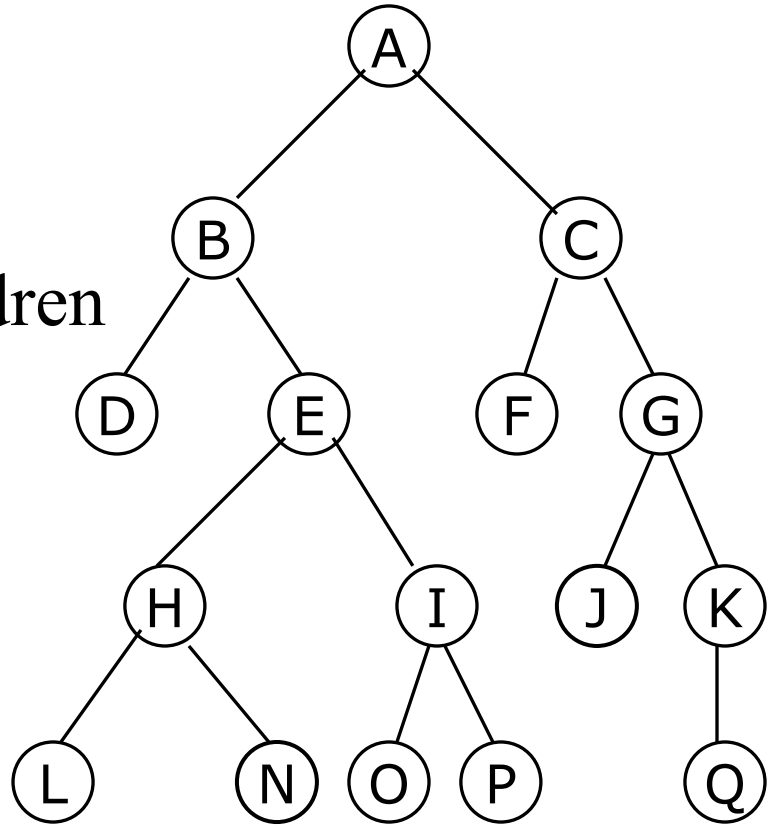
Q = {Q,P,O,N}



More Binary Trees

- BFS; Visited = {A,B,C,D,E,F,G,H,I,J,K,L,N}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

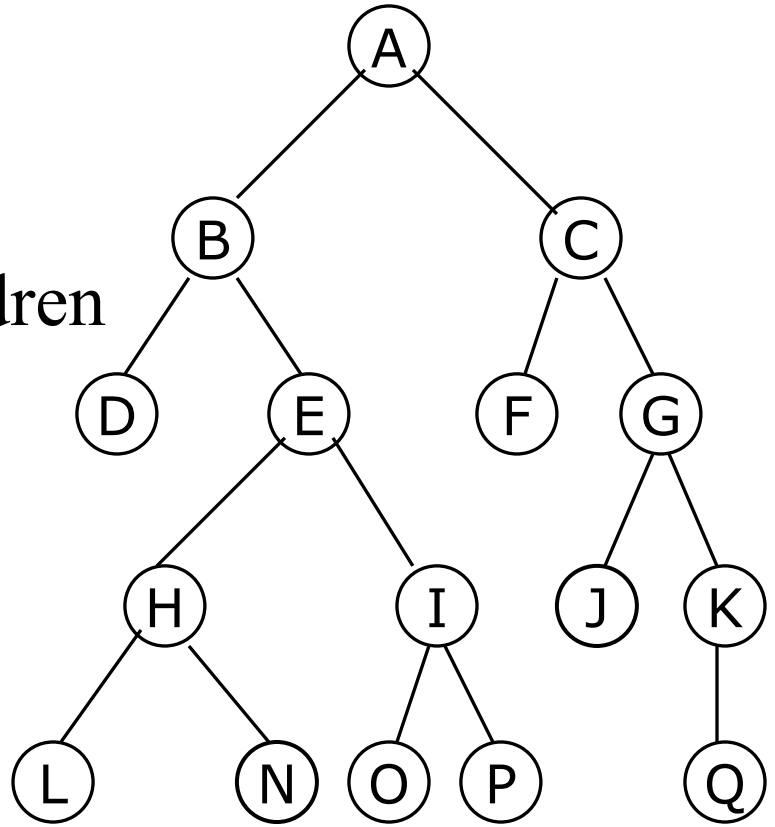
Q = {Q,P,O}



More Binary Trees

- BFS; Visited = {A,B,C,D,E,F,G,H,I,J,K,L,N,O}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

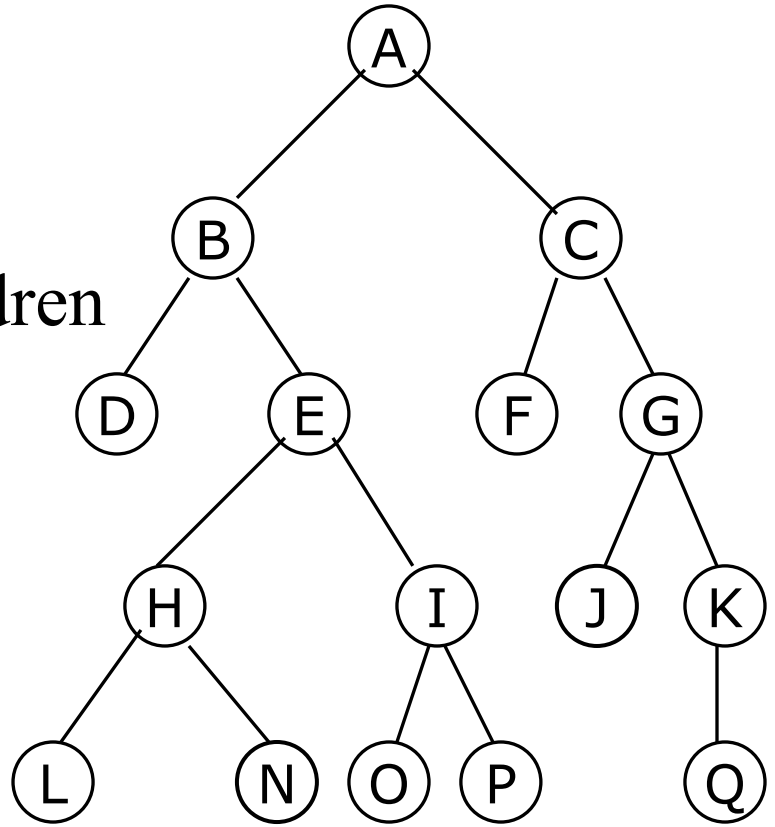
Q = {Q,P}



More Binary Trees

- BFS; Visited = {A,B,C,D,E,F,G,H,I,J,K,L,N,O,P}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

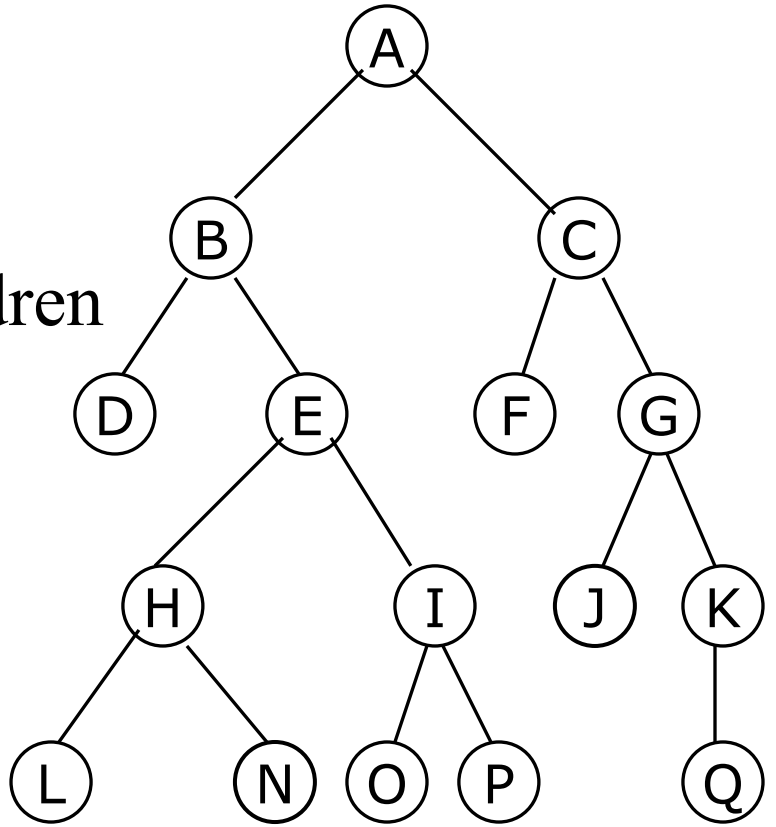
Q = {Q}



More Binary Trees

- BFS;Visit = {A,B,C,D,E,F,G,H,I,J,K,L,N,O,P,Q}
  - Enqueue root
  - While Q not empty
    - Dequeue
    - Enqueue all children
  - End while

Q = {}

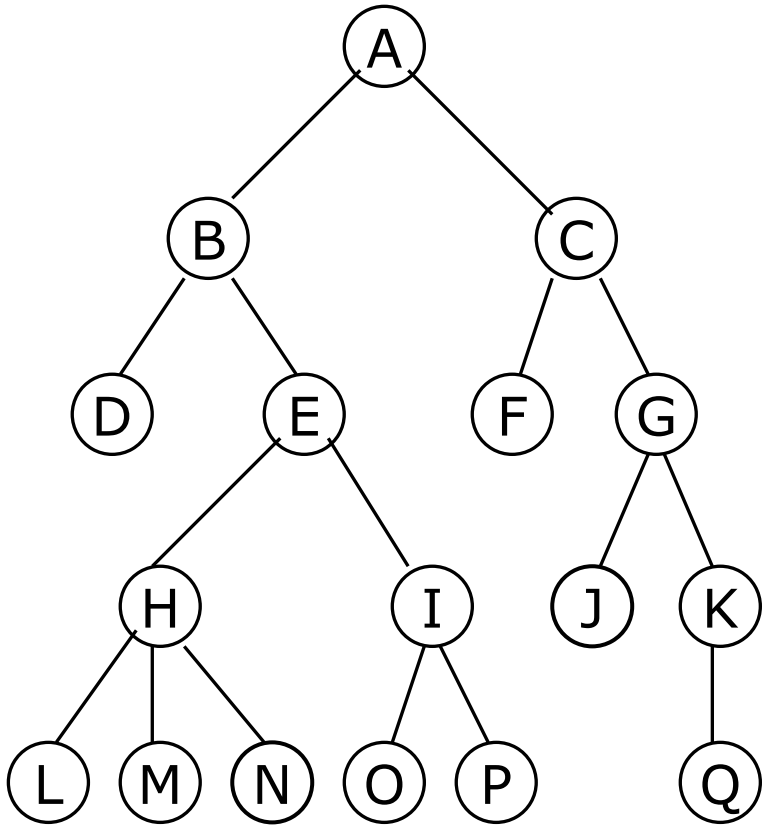


Trees

- Tree Traversal
  - Pre Order
  - Post Order
  - inorder
  - DFS
  - BFS

Trees

- Tree Traversal
  - Pre Order
  - Post Order
  - DFS
  - BFS



Representing Trees

- Linked list Based representation

- Node {
  - int Data;
  - Node *leftChild;
  - Node *rightSibling;
  - Node *parent;
  - }

regular trees

- Inserting a node .. How to specify
- Program to insert a node
- Program to find a key ?

regular trees

- Tree traversal – modify the binary tree traversal.