# Data Structures (15B11CI311)

Odd Semester 2020


3rd Semester , Computer Science and Engineering

Jaypee Institute Of Information Technology (JIIT), Noida

# Lecture: 25

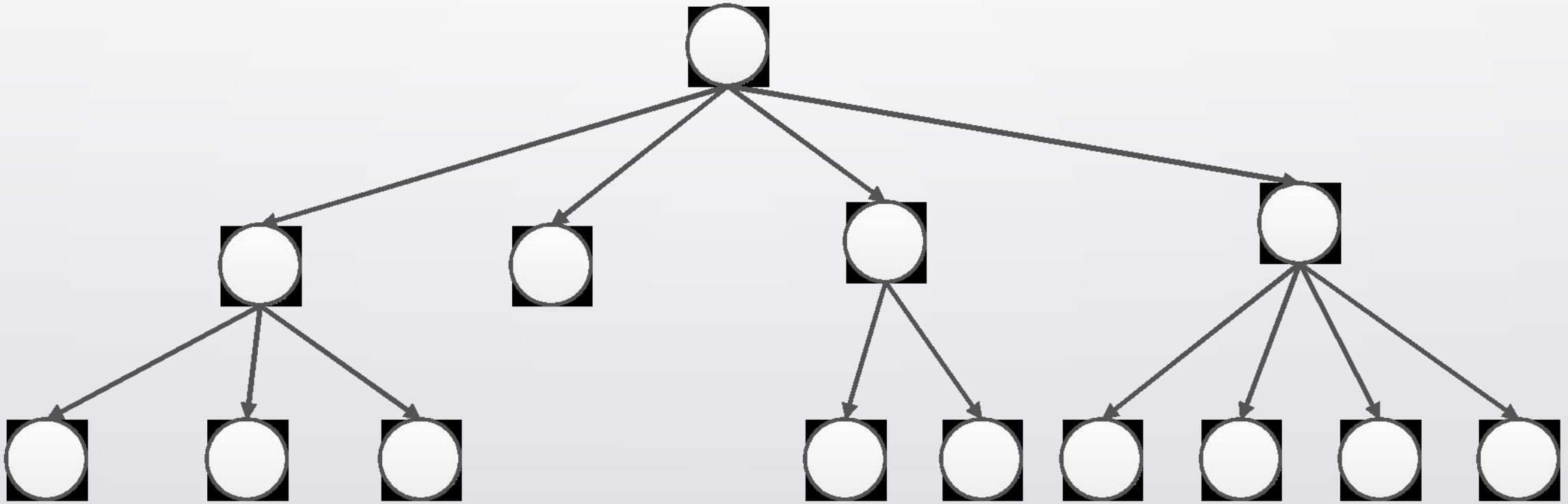
Topics to be covered:

- Introduction to k-ary tree
- Operations on k-ary Tree

# k-ary Tree

- A tree in which every node has at most K children is known as an K-ary tree
- If K=2 then it will be a binary tree
- If K=3 then it will be a ternary trees, and so on
- The examples with k>2: 2-3 tree, B-tree
- All the children of a node in a K-ary tree are sequentially ordered, i.e. left-to-right

# Example of k-ary Tree with k=4



# Properties of k-ary Tree

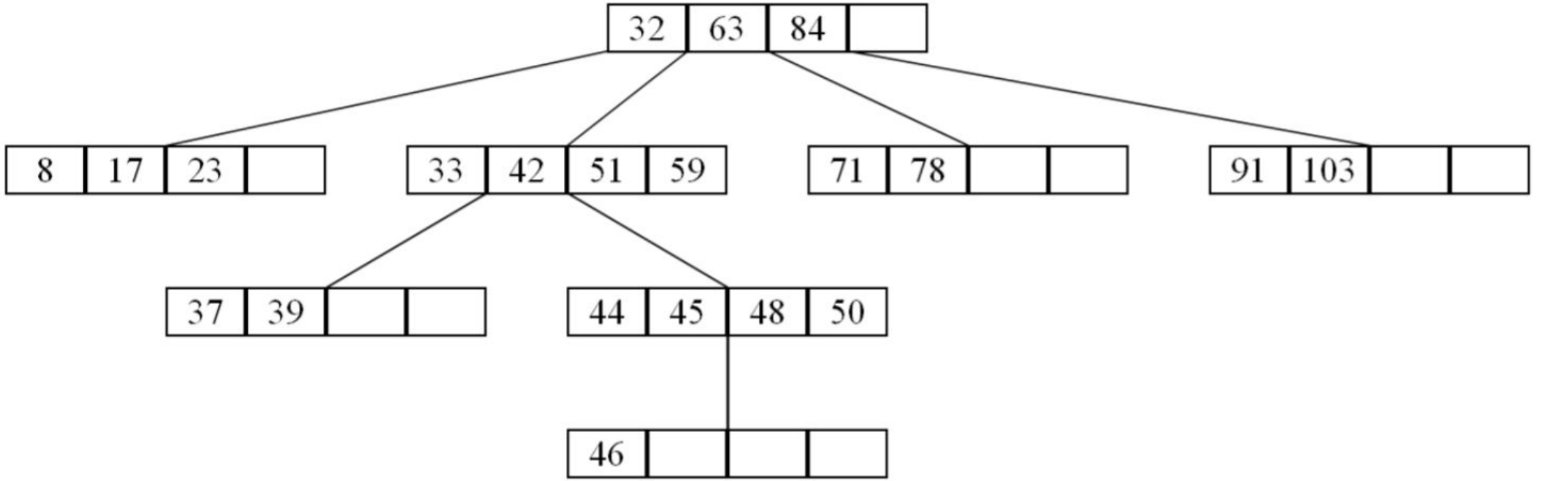
- The maximum depth of a K-ary tree, having N nodes: N
- The minimum depth of a K-ary tree, having N leaves: $\log_k (N+1)$

# k-ary Search Tree

- An K-ary search tree is a K-ary tree in which
  - A set of K-1 key values is stored at each node
  - All key values in a sub-tree Sn, lie between the key values Vn and Vn+1 of the parent node.
- A binary search tree is an example of an K-ary search tree, where K is 2.

http://www2.cs.uregina.ca/~beattieb/CS340/notes/09%20-%20Non-Binary%20Trees.pdf

# Example of k-ary Search Tree with k=5

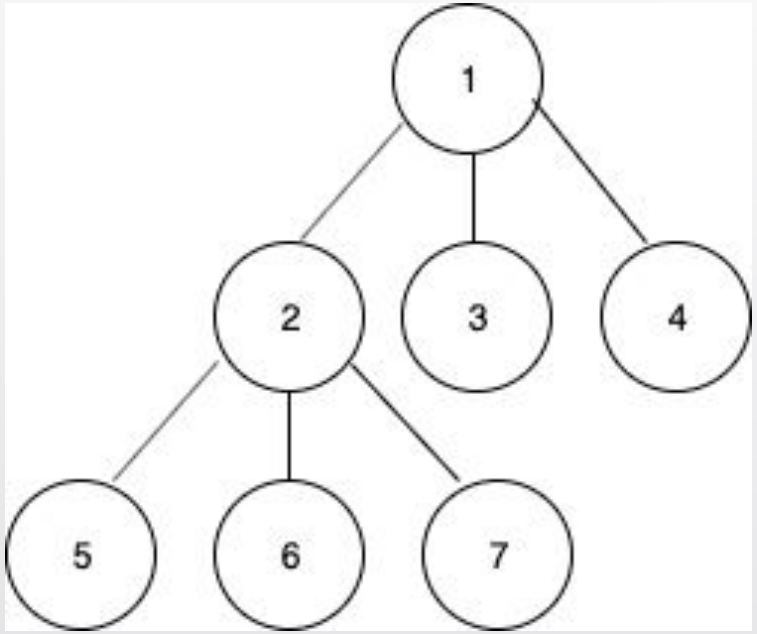



http://www2.cs.uregina.ca/~beattieb/CS340/notes/09%20-%20Non-Binary%20Trees.pdf

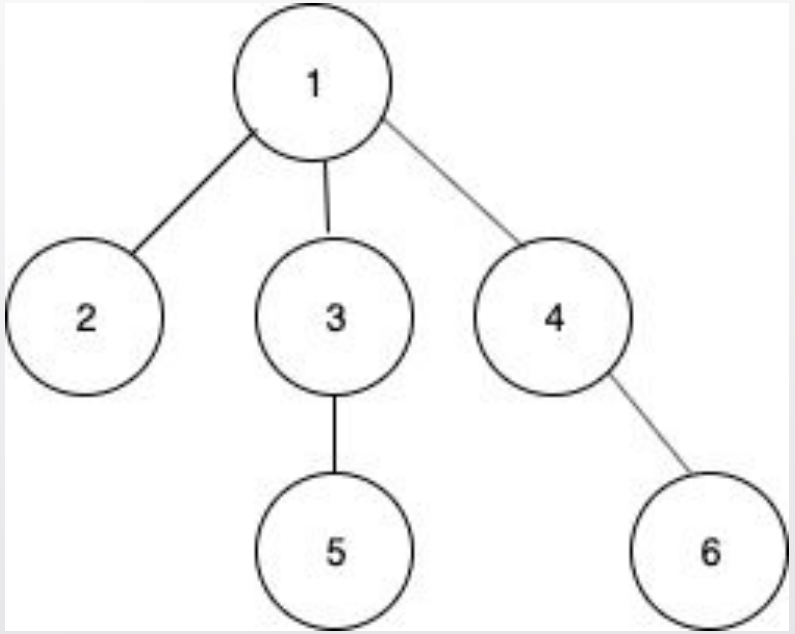
# Traversal in k-ary Tree

- Traversing a k-ary tree is very similar to binary tree traversal.
- Pre-order traversal:  visit parent, left subtree and then right subtree
- Post-order traversal: left subtree, right subtree, and then visit parent
- In-order traversal: left subtree, visit parent and then right subtree
- However, as there are more than two children per node for m > 2, left and right subtrees are to be defined
- One possible approach is to divide the list of children nodes into two groups:
  - First (m-1) can be considered as part of left subtree and $m^{th}$ node as right subtree or vice-versa or
  - The first m/2 nodes as the left subtree and rest m/2 nodes as the right subtree

# Inorder traversal of k-ary Tree (k=3)




***Output:*** *5 6 2 7 3 1 4*




***Output:*** *2 5 3 1 4 6*

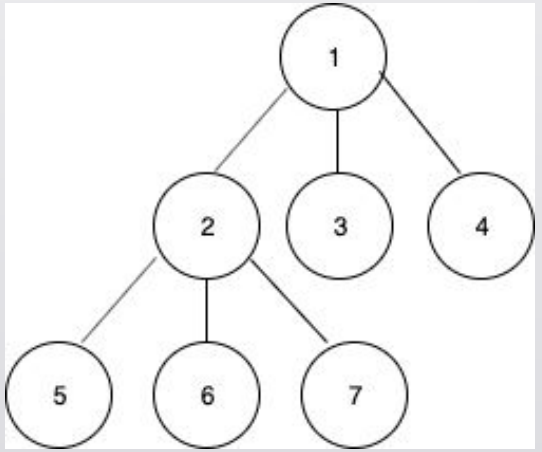
**Approach:** First (m-1) can be considered as part of left subtree and $m^{th}$ node as right subtree

Source: https://www.geeksforgeeks.org/inorder-traversal-of-an-n-ary-tree/

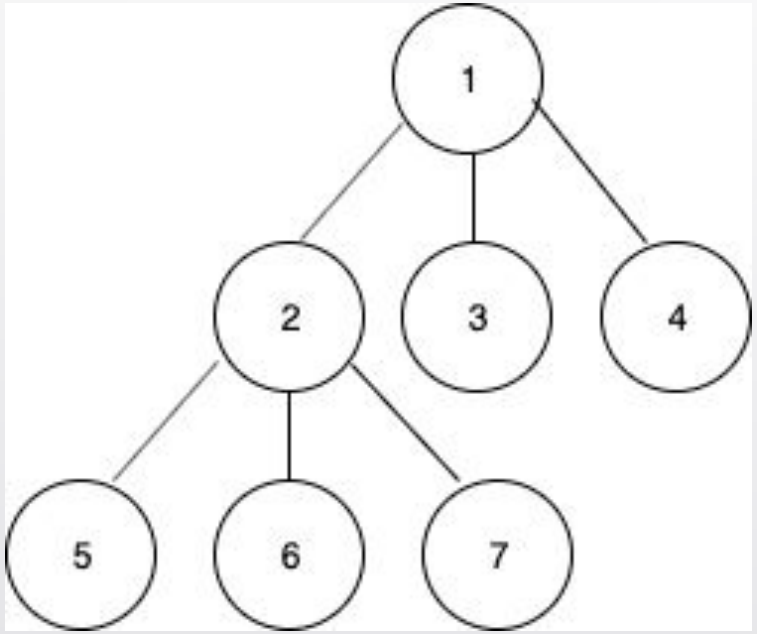
# Inorder Traversal of k-ary tree (k=3)

```
#include<iostream>
using namespace std;

class Node {
   public:
     int data;
     Node *children[3];
     Node(int val)
     {
       data = val;
     }
};

void inorder(Node *node) {
        if (node == NULL)
                return;
        for (int i = 0; i < 2; i++)
                inorder(node->children[i]);
        cout<< node->data << " ";
        inorder(node->children[2]);
}
```

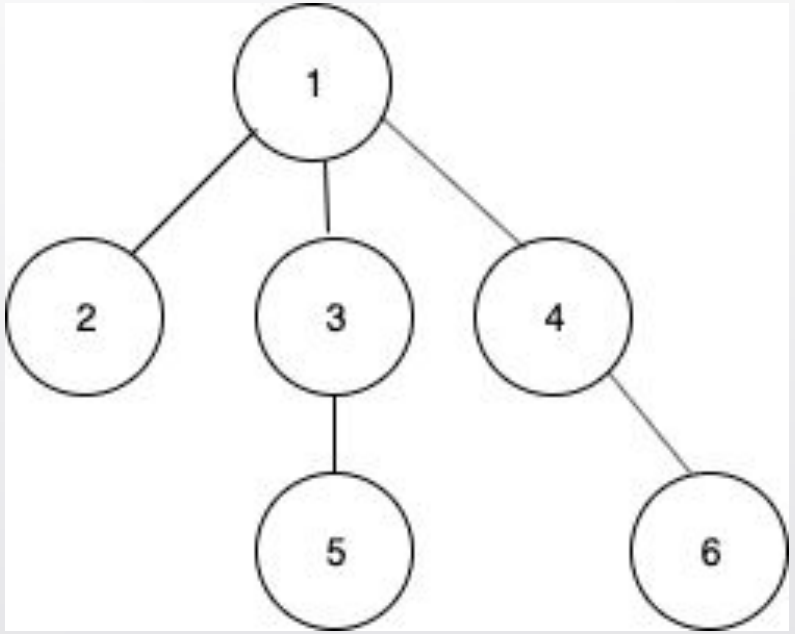
```
int main() {
        Node* root = new Node(1);
        root->children[0] = new Node(2);
        root->children[1] = new Node(3);
        root->children[2] = new Node(4);
        root->children[0]->children[0] =   new Node(5);
        root->children[0]->children[1] = new Node(6);
        root->children[0]->children[2] = new Node(7);
        inorder(root);
        return 0;
}
```



# Preorder traversal of k-ary Tree (k=3)




***Output:*** *1 2 5 6 7 3 4*

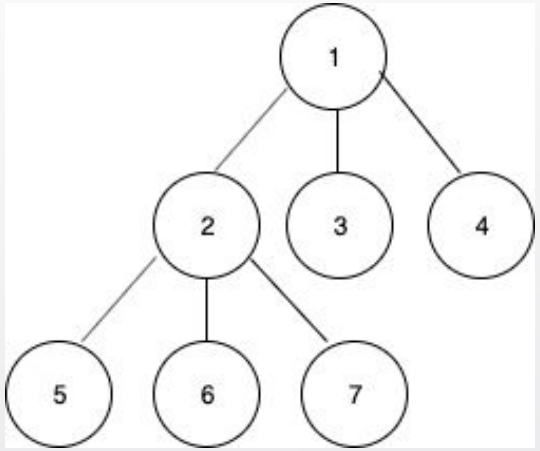



***Output:*** *1 2 3 5 4 6*

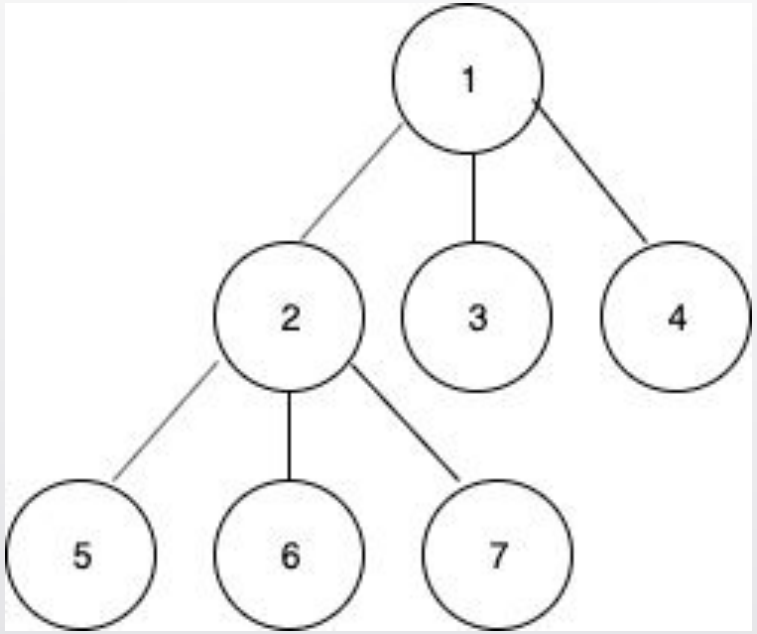
**Approach:** First (m-1) can be considered as part of left subtree and $m^{th}$ node as right subtree

# Preorder Traversal of k-ary tree (k=3)

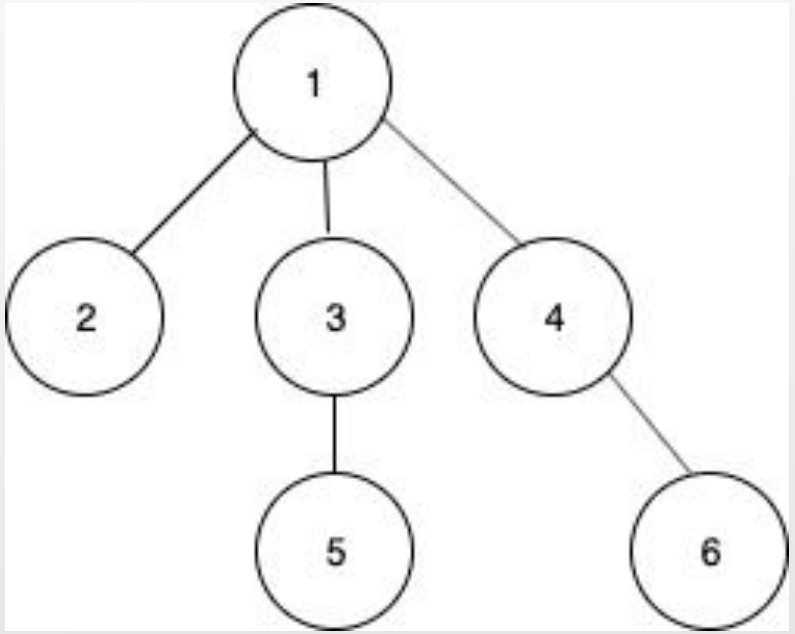
```
void preorder(Node *node)
{
    if (node == NULL)
        return;
    cout<< node->data << " ";
    for (int i = 0; i < 2; i++)
        preorder(node->children[i]);
    preorder(node->children[2]);
}
```



# Postorder traversal of k-ary Tree (k=3)




***Output:*** *5 6 7 2 3 4 1*

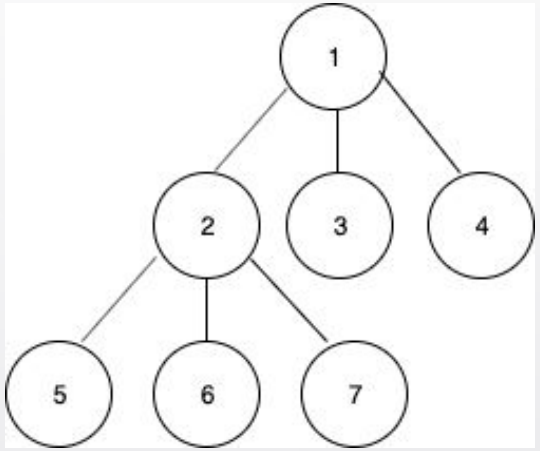



***Output:*** *2 5 3 6 4 1*

**Approach:** First (m-1) can be considered as part of left subtree and $m^{th}$ node as right subtree

# Postorder Traversal of k-ary tree (k=3)

```
void postorder(Node *node)
{
    if (node == NULL)
        return;
    for (int i = 0; i < 2; i++)
        postorder(node->children[i]);
    postorder(node->children[2]);
    cout<< node->data << " ";
}
```



# References


- *Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009) [1990]*
- *Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill. ISBN 0-262-03384-4. 1320 pp.*
- Adam Drozdek, Data Structures and Algorithms in C++ (2nd Edition), 2001
- https://www.geeksforgeeks.org/inorder-traversal-of-an-n-ary-tree/
- http://www2.cs.uregina.ca/~beattieb/CS340/notes/09%20-%20Non-Binary%20Trees.pdf