Tree Abstract Data Structure

Dr. Aniket Pingley

CSL 102 Data Structures

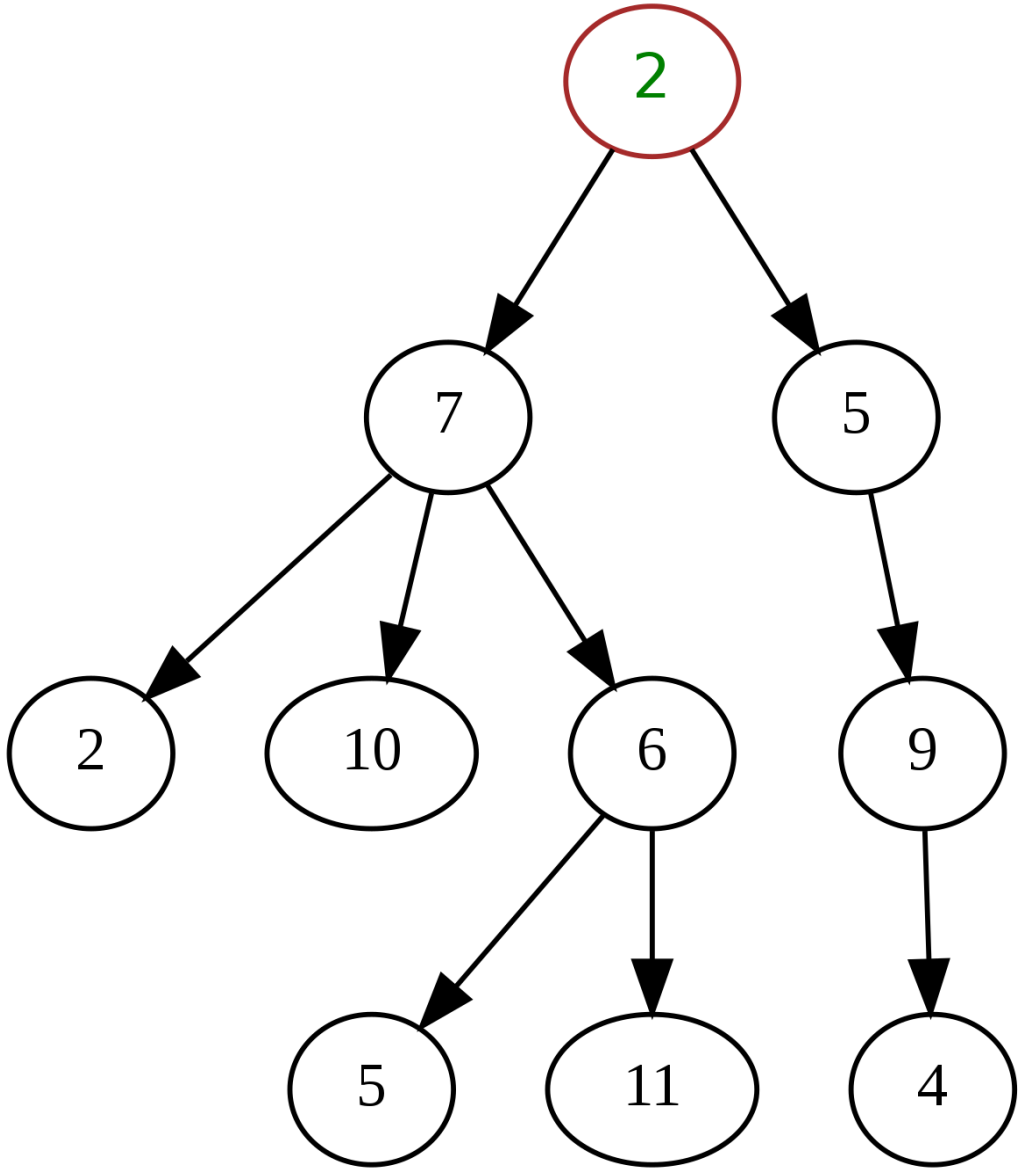
IIIT Nagpur

Need for Tree ADT

- Array, Linked List, Stack and Queue have linear ordering
- In many use cases, information is stored in a hierarchical manner
  - File system – folders and files
  - Geographical data – state, district, cities, pin code etc.
  - Hierarchy of human resource in an organization
- Arranging data in Tree ADT can provide better performance for operations such as insertion, deletion, read/access etc.

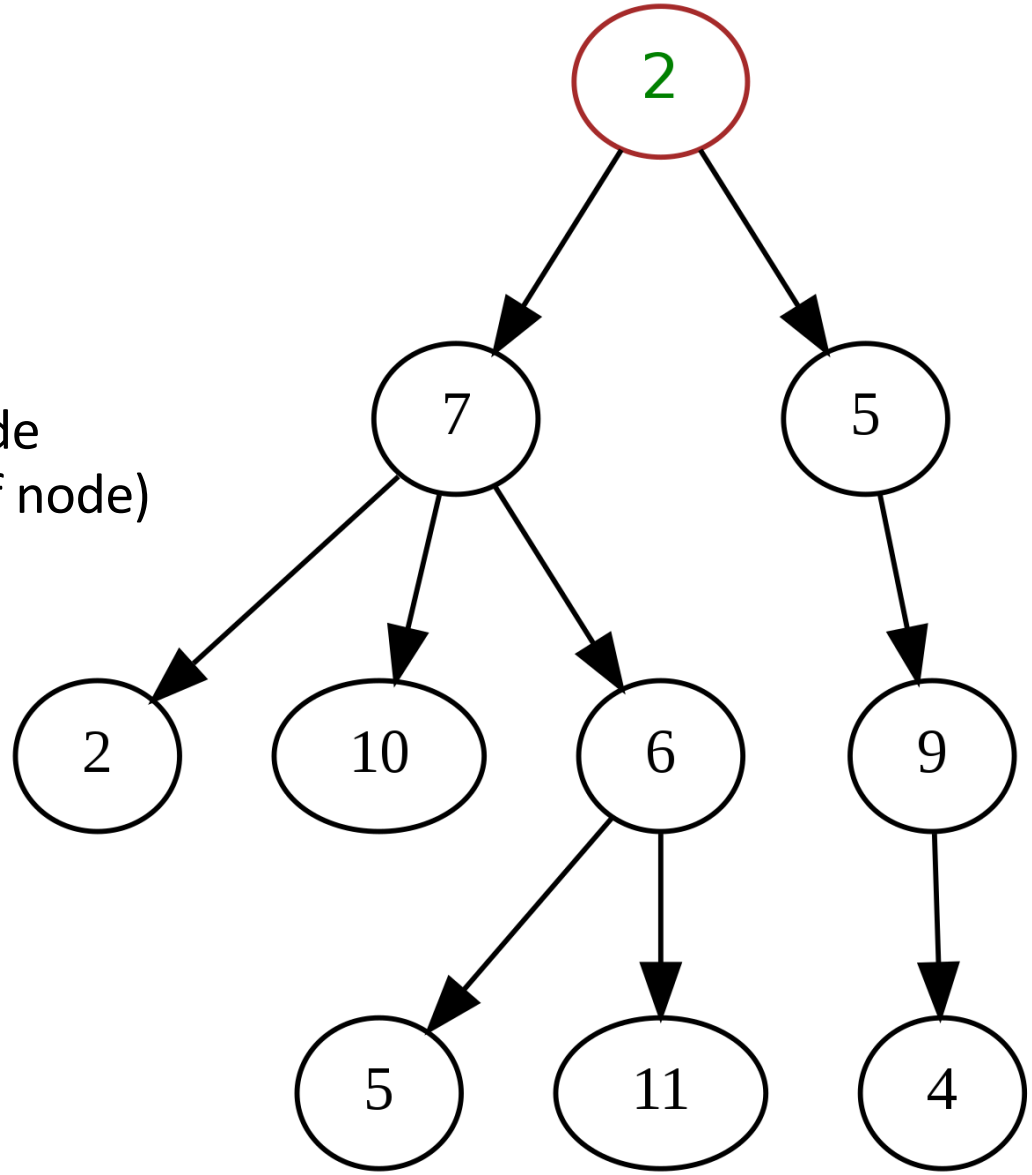
Nomenclature Part 1

- Root, Node, Edges, Leaves/Leaf nodes
  - 2 (top) is root
  - Each circle in the diagram is a node
  - Edges = lines with arrows
  - Nodes 2(bottom), 10, 6, 5, 11 & 4 are leaf nodes
- Parent, Children, Siblings
  - e.g., 6 is parent of 2, 10 and 6.
  - e.g., 2, 10 & 6 are siblings and children of 7
- Ancestors, Descendants
  - e.g., 7 is ancestor of 2, 10, 6, 5 & 11
  - e.g., 4, 9 and 5 are descendants of 2
- Subtree
  - e.g., 5 and its descendants together are a subtree



Nomenclature Part 2

- Path
  - e.g., 2->7->6->11 is a path from root to a leaf node
  - e.g., 7->10 is a path from node 7 to node 10 (leaf node)
- Length of Path:
  - Number of edges in a path
- Height
  - Node: length of longest path from a node to leaf
  - Tree: height of the root node (4 is this case)
- Depth
  - Node: length of path from root to a node
  - Tree: depth of the deepest node (4 in this case)
- Height of Tree = Depth of Tree



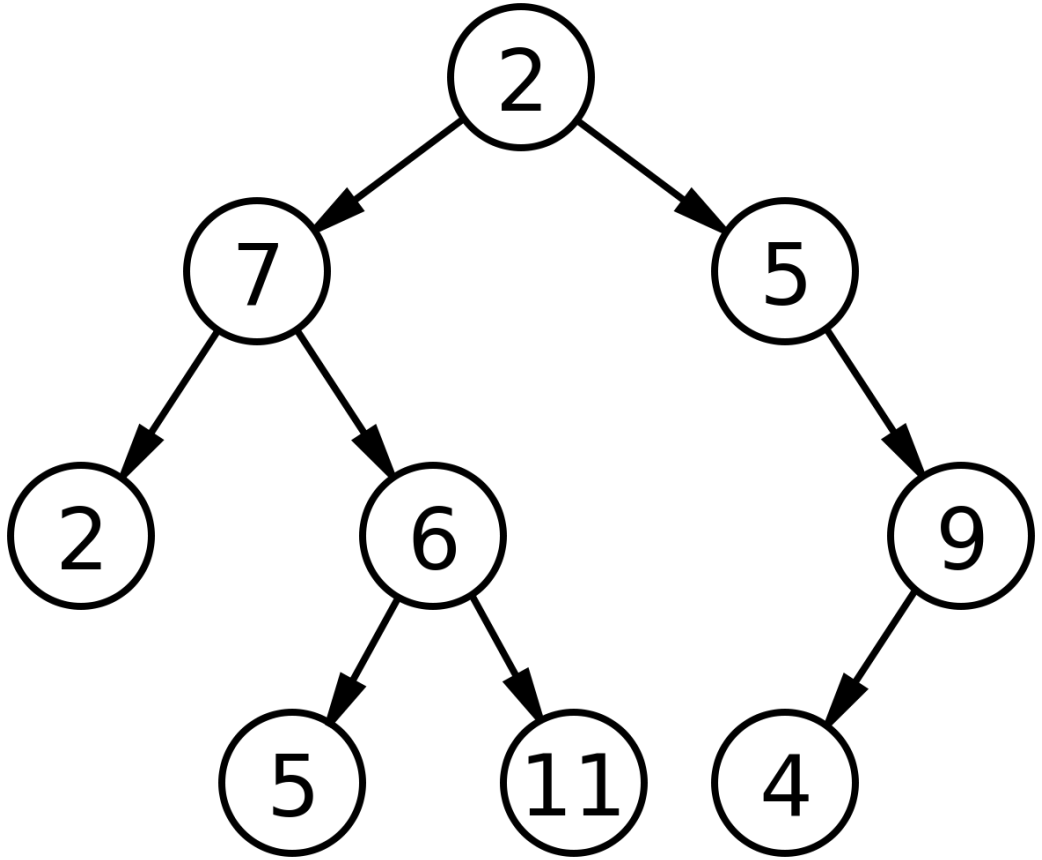
Formal Description

- A tree is a *nonlinear* data structure
- A tree can be empty with no nodes or a tree is a structure consisting of one node called the **root** and zero or one or more subtrees.
- A tree has following general properties:
  - One node is distinguished as a **root**;
  - Every node (exclude a root) is connected by a directed edge *from* exactly one other node; A direction is: *parent -> children*
- A tree with N nodes always has N-1 edges
- Two nodes in a tree have at most one path between them
- Trees cannot have cycles/loop

Binary Trees

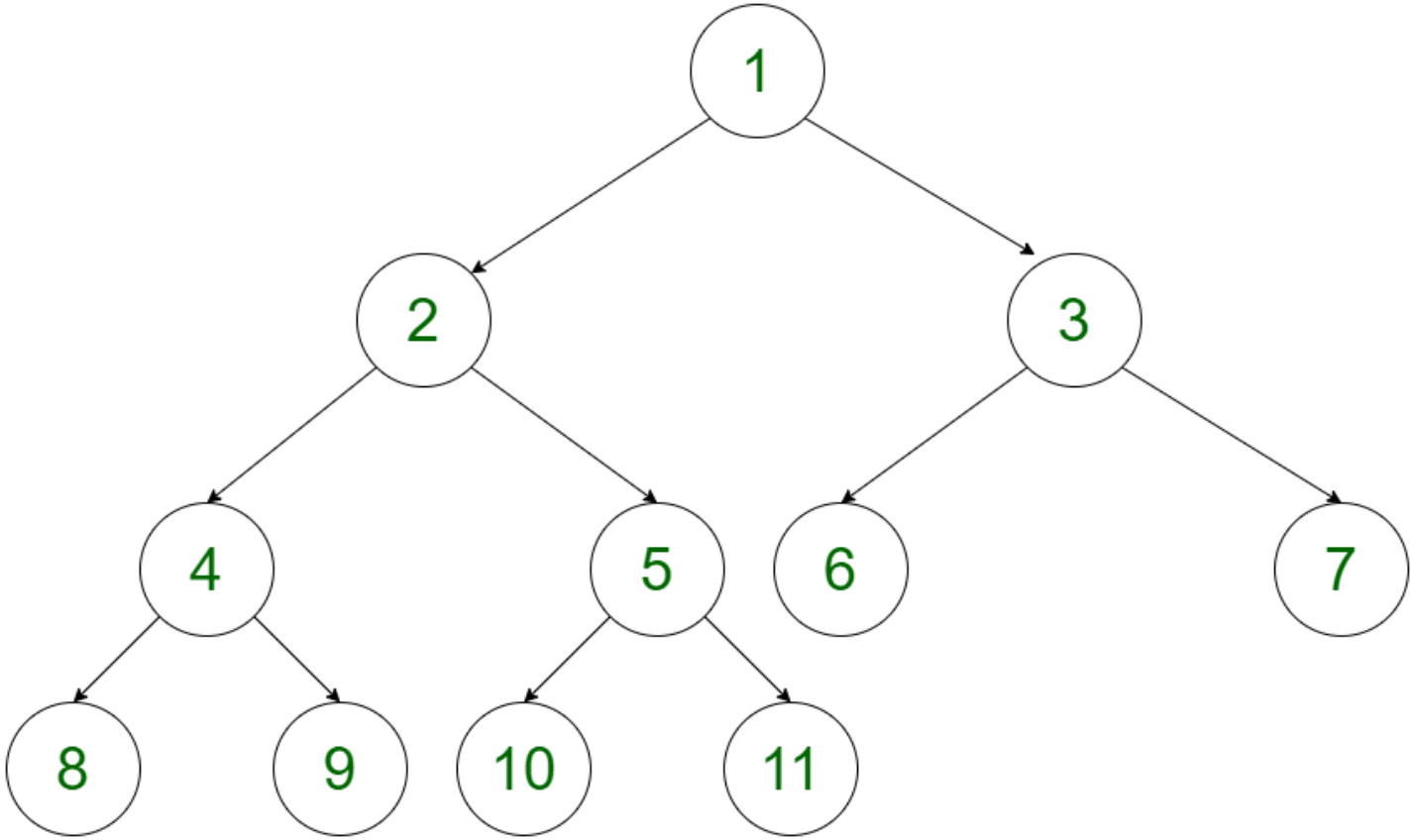
Every node has at most two children

```
struct node {
    short data;
    struct node* left;
    struct node* right;
};
```



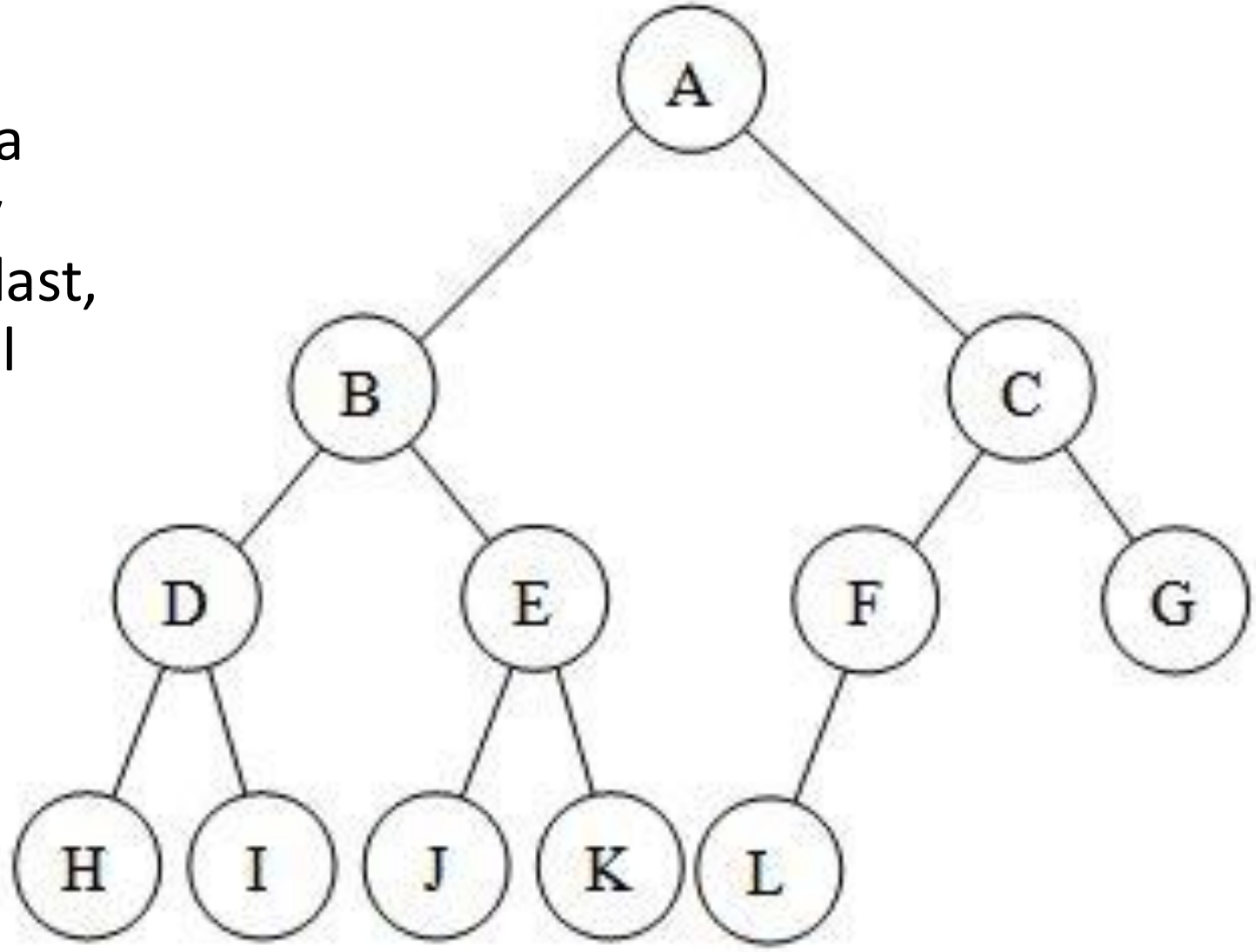
Full Binary Tree

Special kind of binary tree where each node has either ZERO or TWO children



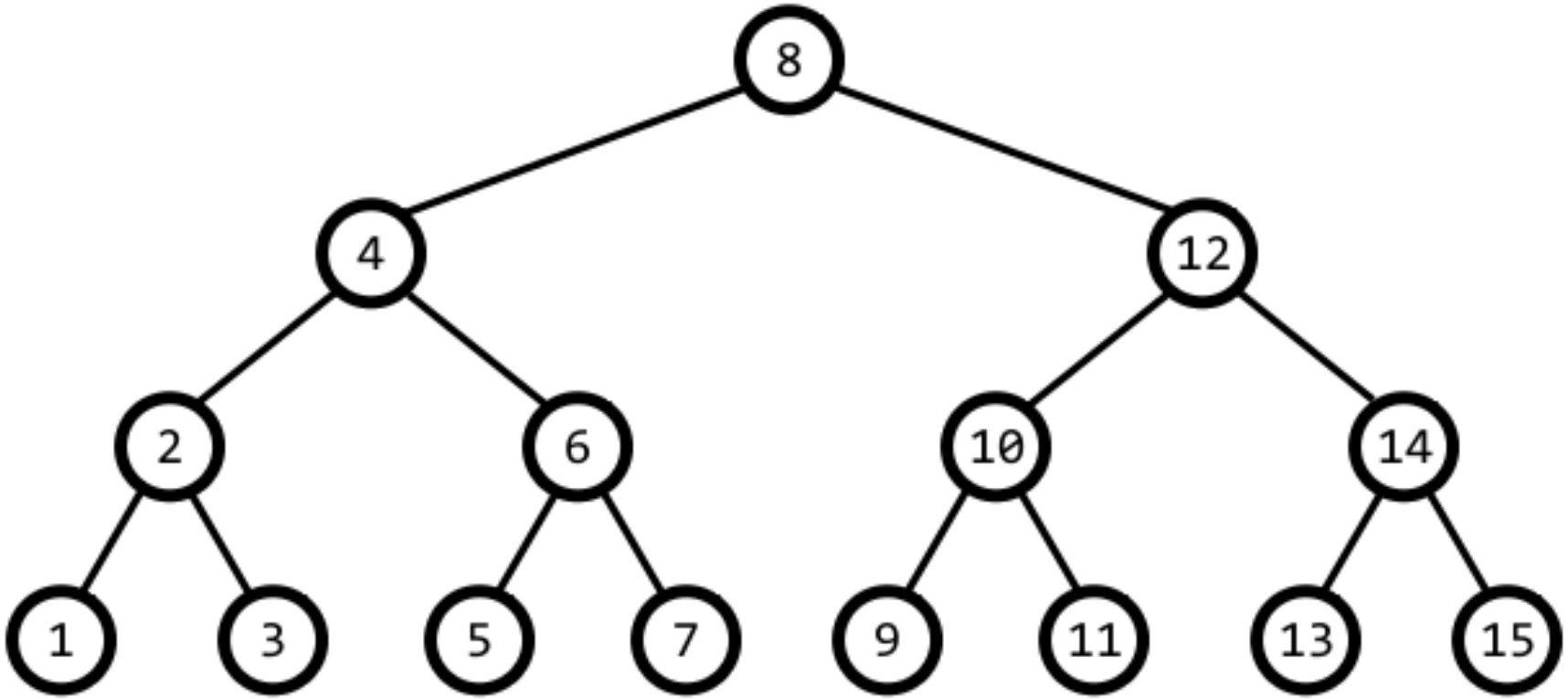
Complete Binary Tree

A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.



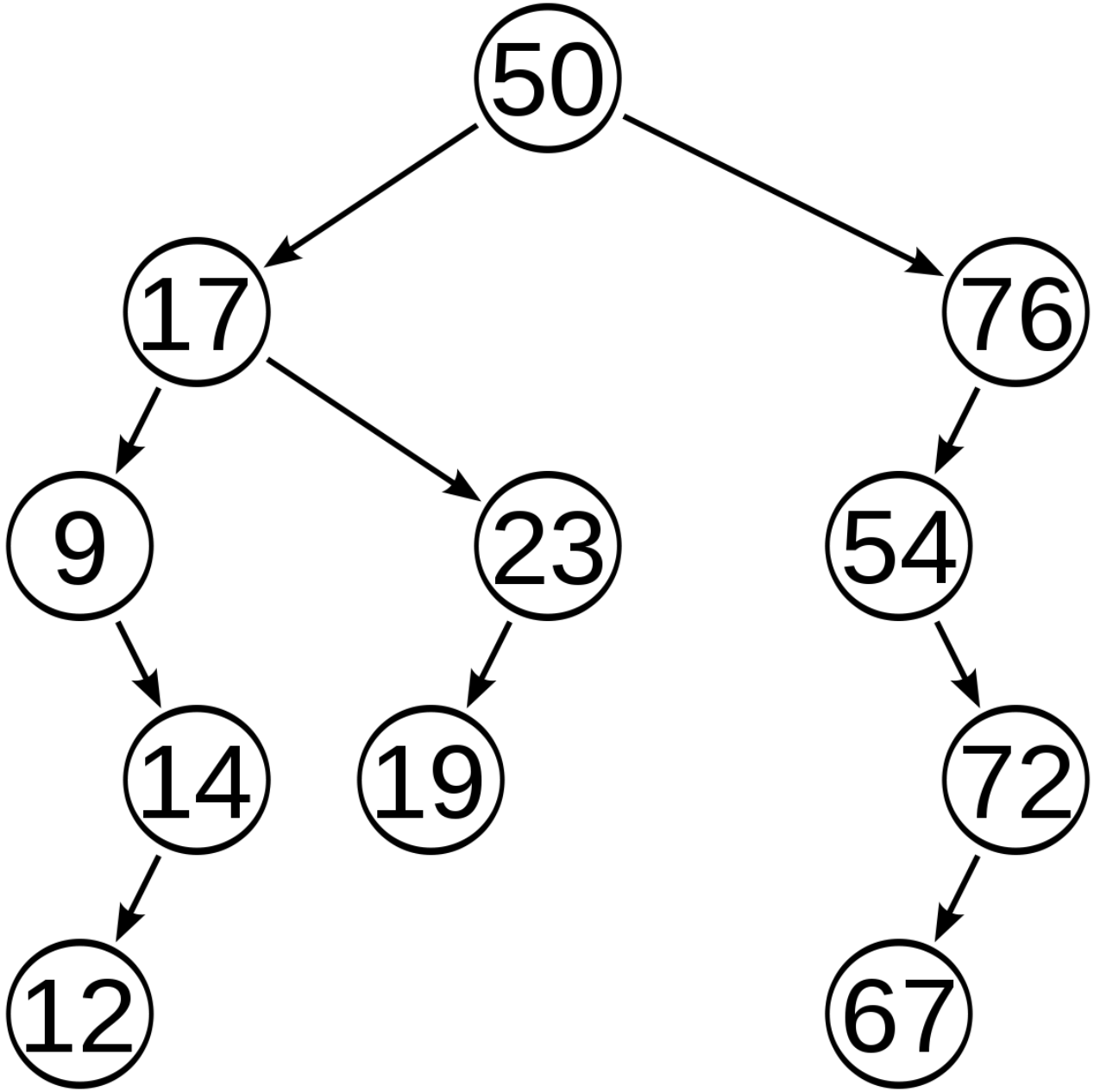
Perfect Binary Tree

All interior nodes have two children *and* all leaves have the same *depth*



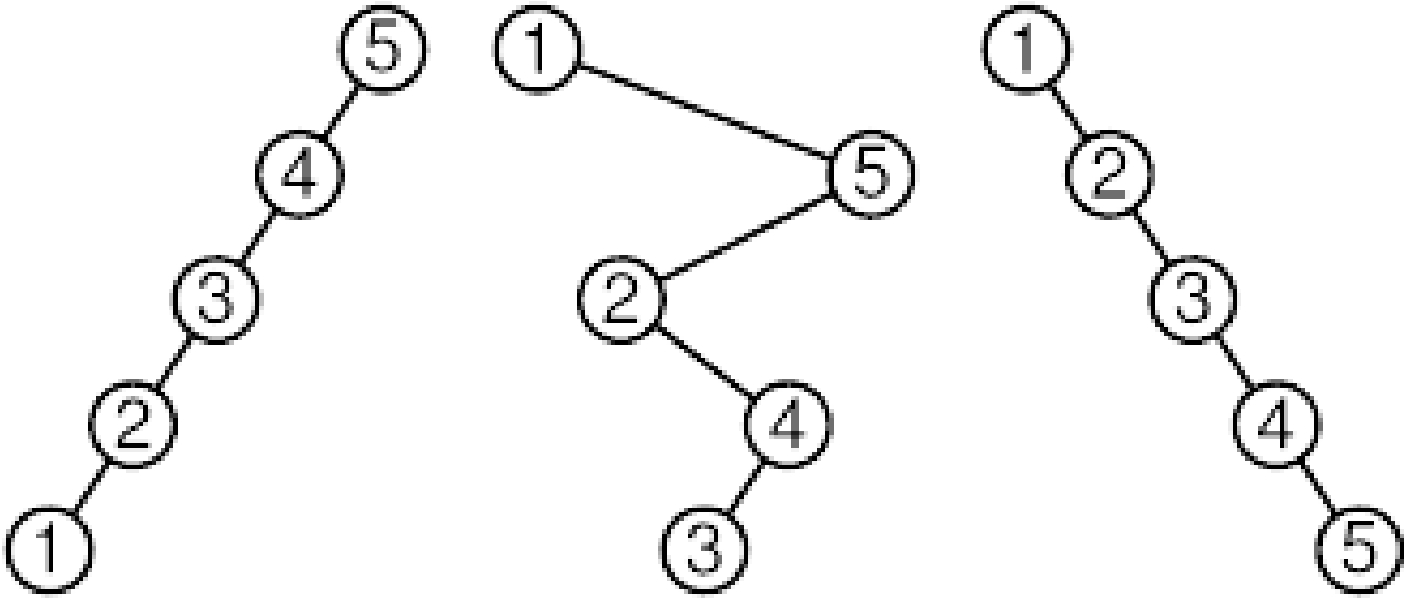
Balanced Binary Tree

In a balanced binary tree, the height of the left and the right subtrees of each node should vary by at most one



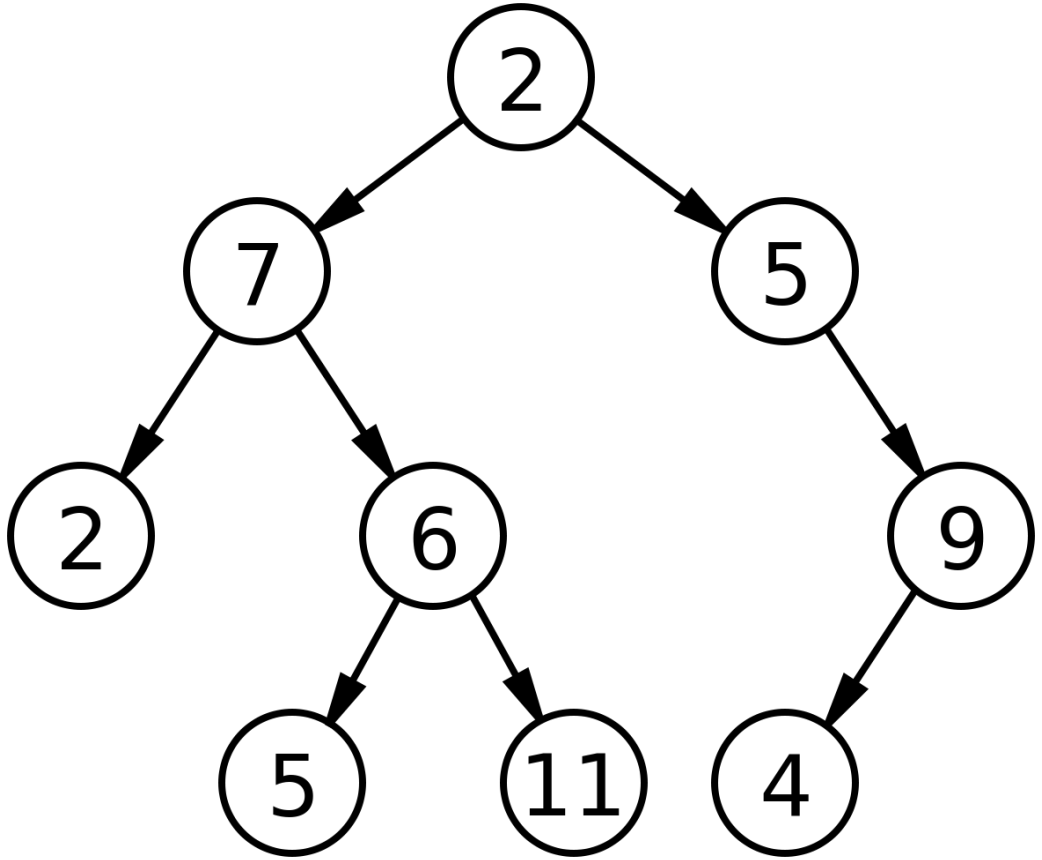
Degenerate Tree

A degenerate (or pathological) tree is where each parent node has only one associated child node, i.e., linked list data structure.



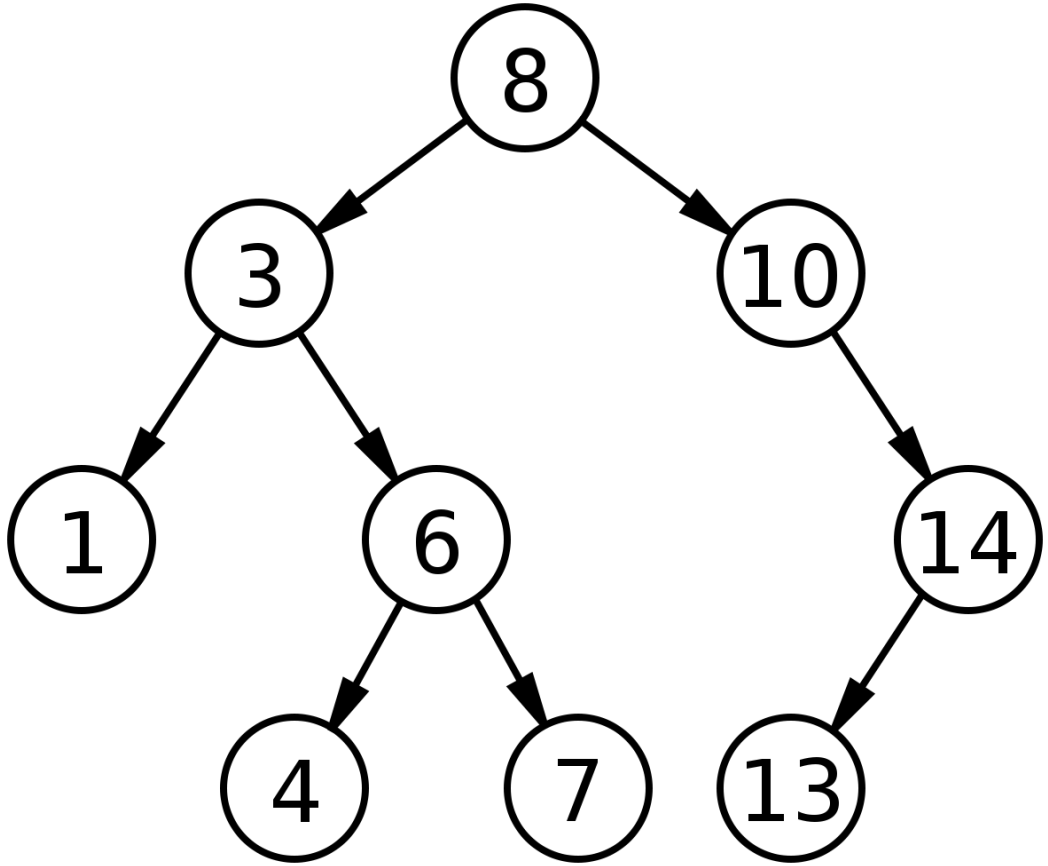
Binary Tree Trivia

- At a given depth/level, a maximum of 2^d nodes can exist, where d is the depth
- For a given depth/level, a maximum of $2^{(d+1)}$ -1 total number of nodes can exist in the tree, where d is the depth
- For N total number of nodes
  - Minimum depth = $\log(N) \leq d \leq \log(N+1)-1$



Tree Traversal

- Inorder
  - Visit all the nodes in the left subtree
  - Visit root node
  - Visit all the nodes in the right subtree
- Preorder
  - Visit root node
  - Visit all the nodes in the left subtree
  - Visit all the nodes in the right subtree
- Postorder
  - Visit all the nodes in the left subtree
  - Visit all the nodes in the right subtree
  - Visit root node



Binary Search Tree

- BST are binary trees
  - All values in the node's left subtree are less than the node value
  - All values in the node's right subtree are greater than the node value
  - All values is right subtree are greater than all values in left subtree