5.2 Binary Trees

5.2

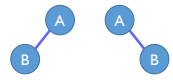
Binary Tree Definition

A ***binary tree*** is a finite set of nodes that either is ***empty*** or consists of a ***root*** and two disjoint binary trees called the ***left subtree*** and the ***right subtree***.

- Binary tree != Regular tree

	Binary Tree	Regular Tree
Has zero nodes	YES	NO
Order of the children	Important	Doesn't matter

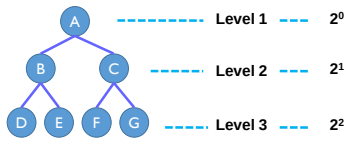
The same trees but different binary trees



5.2.2

Properties of BT: Maximum number of nodes

- The max. # of nodes on level i is 2^{i-1}. The max. # of nodes in a binary tree with depth k is $2^k - 1$



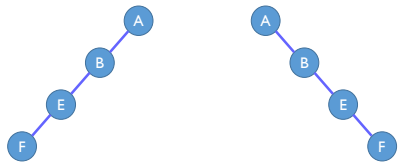
Total # of node is $2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^{k-1} = 2^k - 1$

Properties of BT: # of Leaf Nodes

- if n_0 = number of leaf nodes and n_2 = number of degree-2 nodes, then $n_0 = n_2 + 1$
- Proof:
  - $n = n_0 + n_1 + n_2$, where n_1 is # of degree-1 nodes
  - $n = B + 1$, where B is # of branches
  - $B = n_1 + 2n_2$ (all branches B stem from a node of degree 1 or 2)
  - $n_0 + n_1 + n_2 = n_1 + 2n_2 + 1$
  - $n_0 = n_2 + 1$

15

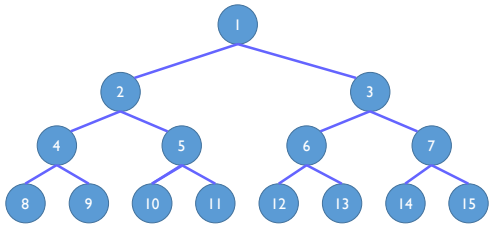
Skewed Binary Tree



Skewed to the left Skewed to the right

Full Binary Tree

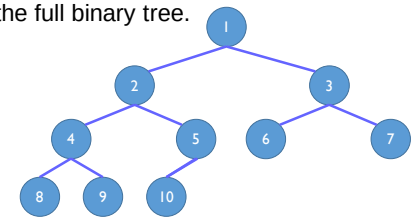
A binary tree of depth k and $2^k - 1$ nodes



A full binary tree of depth 4

Complete Binary Tree

- A binary tree of depth k with n node is called **complete** iff its nodes corresponding to the nodes numbered from 1 to n are in the full binary tree.



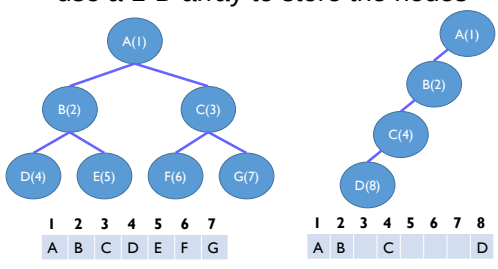
A complete binary tree of depth 4

5.2.3 BINARY TREE REPRESENTATION

19

Array Representation

- The numbering scheme suggests to use a 1-D array to store the nodes



Array Representation

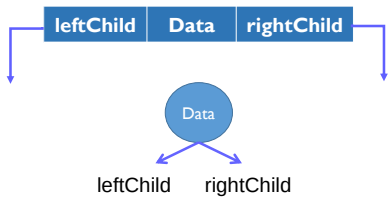
- Advantages: Easy to determine the locations of the parent, left child, and right child of any node.
- Let node i be in position i (array[0] is empty)
  - **parent(i) =** $\lfloor i/2 \rfloor$ if $i \neq 1$. If $i = 1$, i is the root and has no parent.
  - **leftChild(i) =** $2i$ if $2i \leq n$. If $2i > n$, then node i has no left child.
  - **rightChild(i) =** $2i + 1$ if $2i + 1 \leq n$, if $2i + 1 > n$, then node i has no right child.

Array Representation

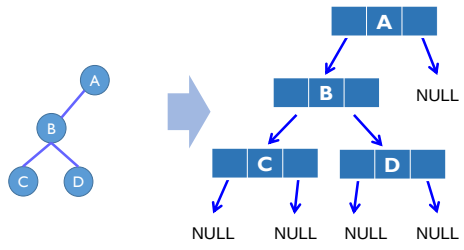
- Disadvantages:
  - Wasted space for a skewed tree.
  - Insertion and deletion of nodes require moving a large parts of existing nodes

Linked List Representation

- Similar to Chain structure in Ch. 4!
- Each tree node consists of three fields
  - Data, leftChild, and rightChild



Linked List Representation



ADT: Tree

```
template <class T > class Tree; // Forward declaration

template < class T >
Class TreeNode {
friend class Tree <T>;
private:
       T data;
       TreeNode<T>* leftChild;
       TreeNode<T>* rightChild;
};

template <class T>
Class Tree {
public:
       // Constructor
       Tree(void) {root=NULL;}

       // Tree operations here…

private:
      TreeNode<T> *root;
};
```