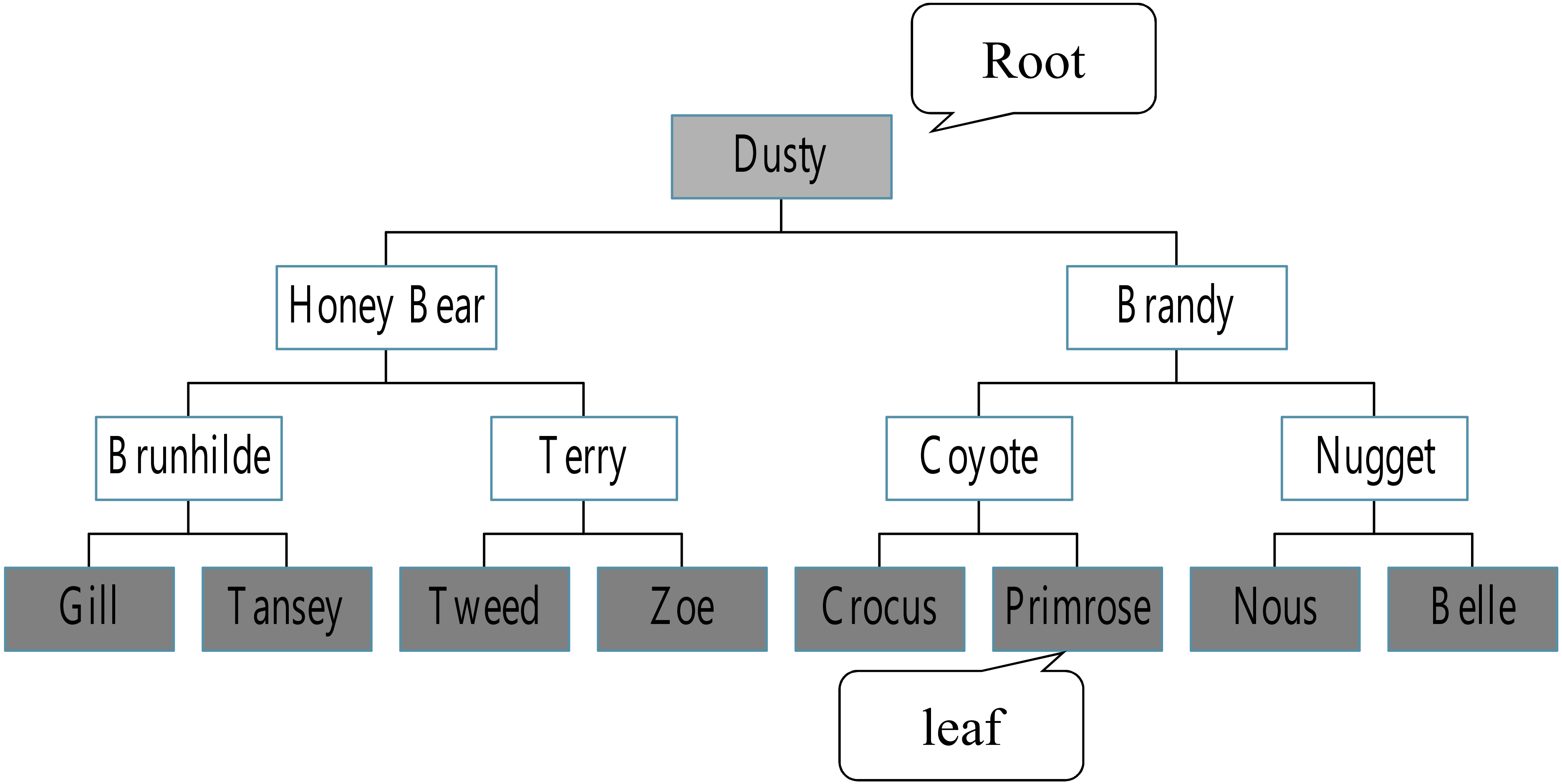
CHAPTER 5

Trees

All the programs in this file are selected from

Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed
"Fundamentals of Data Structures in C",
Computer Science Press, 1992.

Trees


Root
Dusty
Honey Bear
Brandy
Brunhilde
Terry
Coyote
Nugget
Gill
Tansey
Tweed
Zoe
Crocus
Primrose
Nous
Belle
leaf

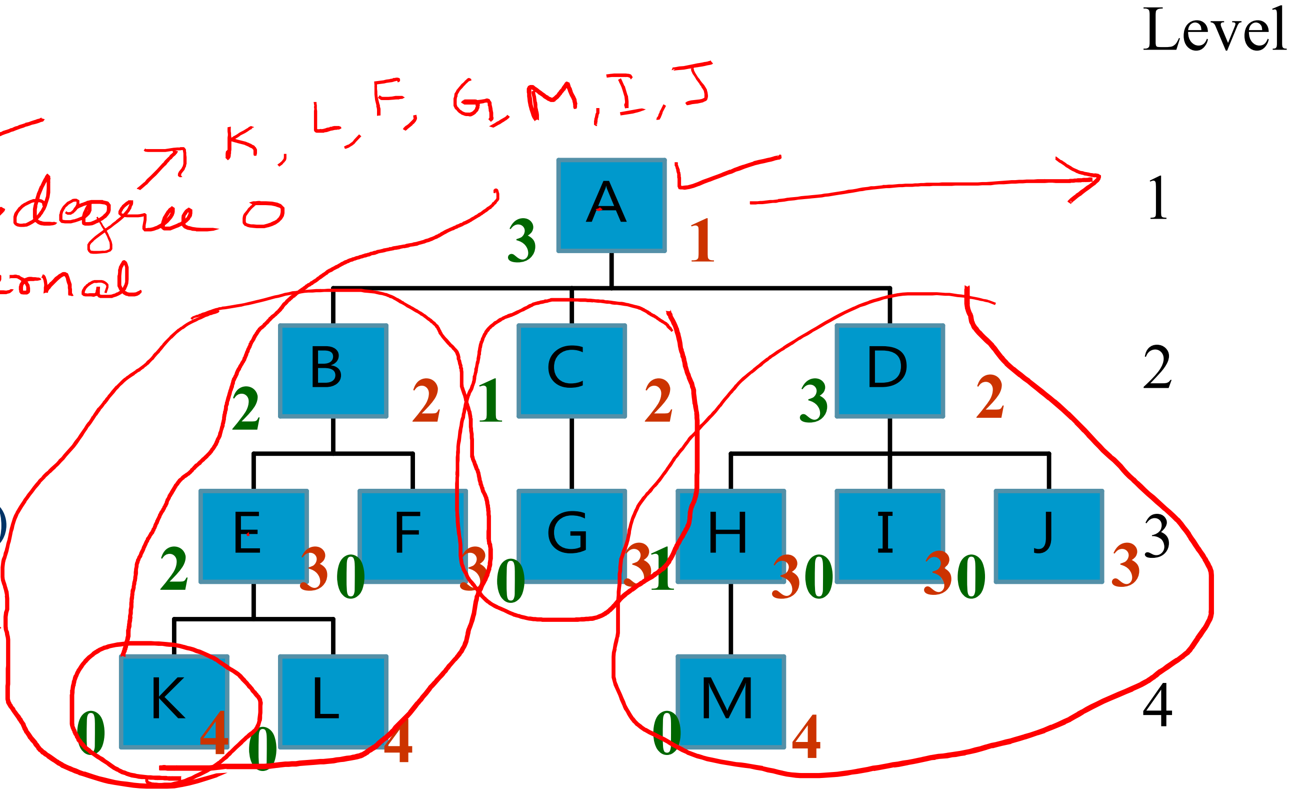
Definition of Tree

- A tree is a finite set of one or more nodes such that:
- There is a specially designated node called the root.
- The remaining nodes are partitioned into n>=0 disjoint sets T_1, ..., T_n, where each of these sets is a tree.
- We call T_1, ..., T_n the subtrees of the root.

Level and Depth

Nonterminal : A, B, C, D, E, H

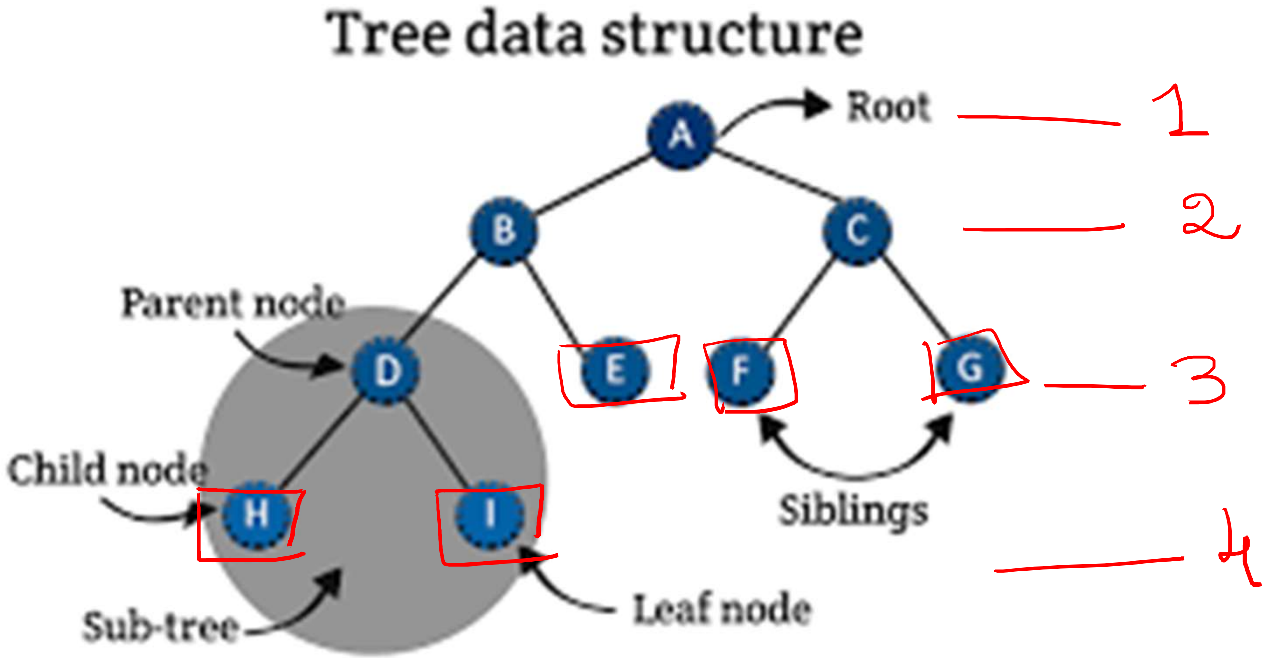
- node (13)
- degree of a node
- leaf (terminal) → degree 0 → K, L, F, G, M, I, J
- nonterminal / internal
- parent
- children
- sibling
- degree of a tree (3)
- ancestor
- level of a node
- height of a tree (4)



Terminology

- The degree of a node is the number of subtrees of the node
  - The degree of A is 3; the degree of C is 1.
- The node with degree 0 is a leaf or terminal node.
- A node that has subtrees is the *parent* of the roots of the subtrees.
- The roots of these subtrees are the *children* of the node.
- Children of the same parent are *siblings*.
- The ancestors of a node are all the nodes along the path from the root to the node.

Example:


Tree data structure
Root
A
B
C
Parent node
D
E
F
G
Child node
H
I
Siblings
Sub-tree
Leaf node
1
2
3
4

Representation of Trees

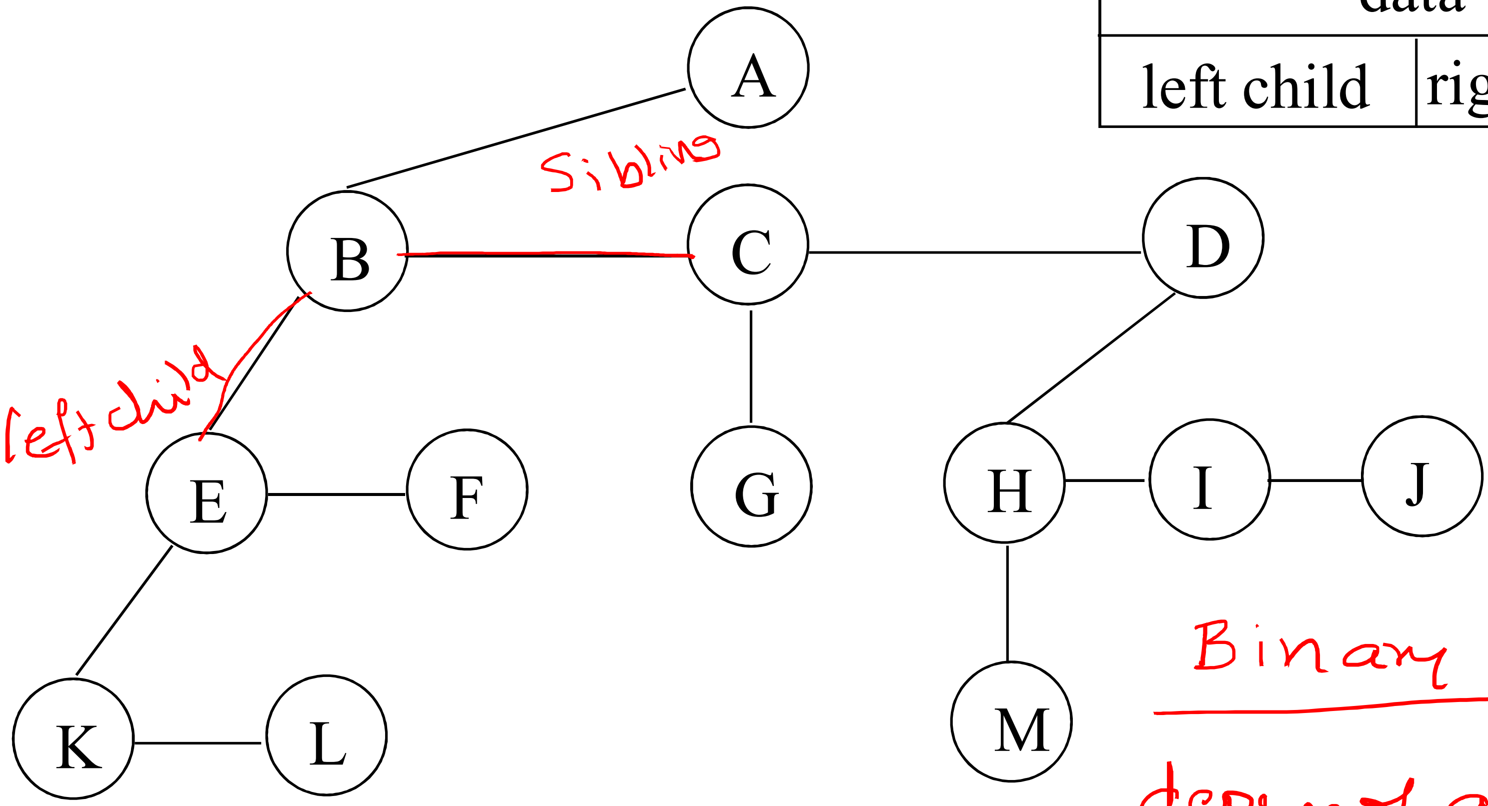
- List Representation
  - (A (B (E (K, L), F), C (G), D (H (M), I, J)))
  - The root comes first, followed by a list of sub-trees

data	link 1	link 2	...	link n

How many link fields are needed in such a representation?

Left Child - Right Sibling

data	
left child	right sibling



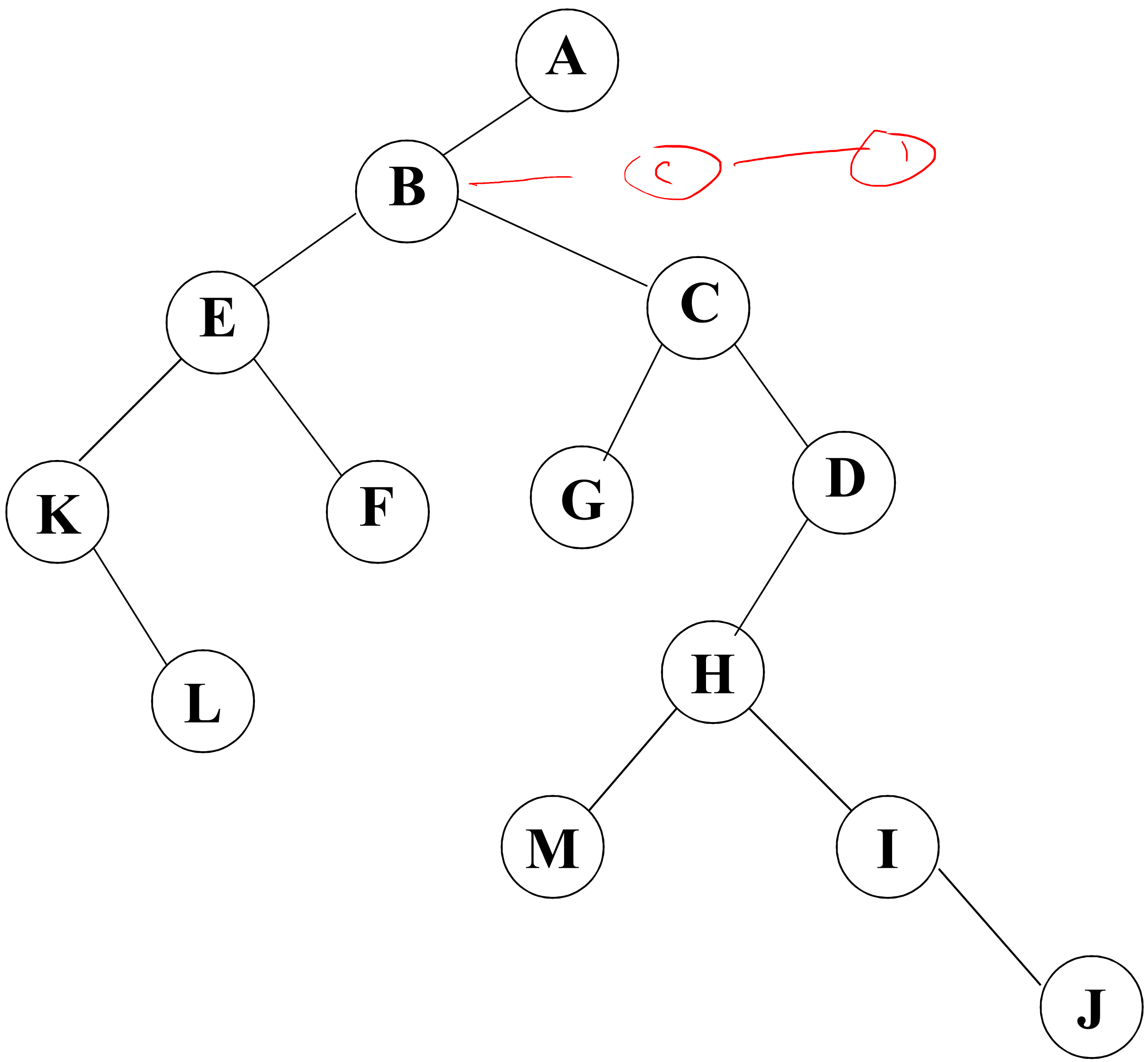
Binary tree

degree of a node ≤ 2

Binary Trees

- A binary tree is a finite set of nodes that is either empty or consists of a root and two disjoint binary trees called *the left subtree* and *the right subtree*.
- Any tree can be transformed into binary tree.
  - by left child-right sibling representation
- The left subtree and the right subtree are distinguished.

***Figure 5.6:** Left child-right child tree representation of a tree (p.191)



Abstract Data Type Binary_Tree

structure *Binary_Tree*(abbreviated *BinTree*) is

objects: a finite set of nodes either empty or consisting of a root node, left *Binary_Tree*, and right *Binary_Tree*.

functions:

for all *bt*, *bt1*, *bt2* $\in$ *BinTree*, *item* $\in$ *element*

Bintree Create()::= creates an empty binary tree

Boolean IsEmpty(*bt*)::= if (*bt*==empty binary tree) return *TRUE* else return *FALSE*

BinTree MakeBT(*bt1*, *item*, *bt2*)::= return a binary tree
whose left subtree is *bt1*, whose right subtree is *bt2*,
and whose root node contains the data *item*

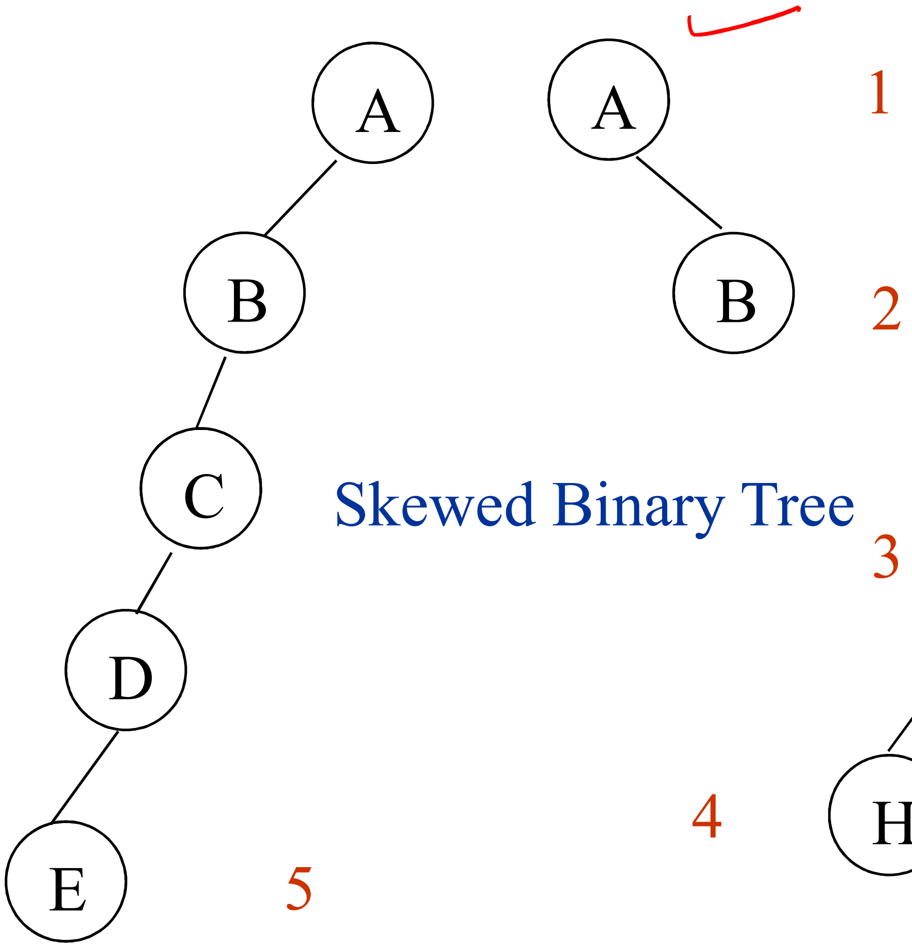
Bintree Lchild(*bt*)::= if (IsEmpty(*bt*)) return error
else return the left subtree of *bt*

element Data(*bt*)::= if (IsEmpty(*bt*)) return error
else return the data in the root node of *bt*

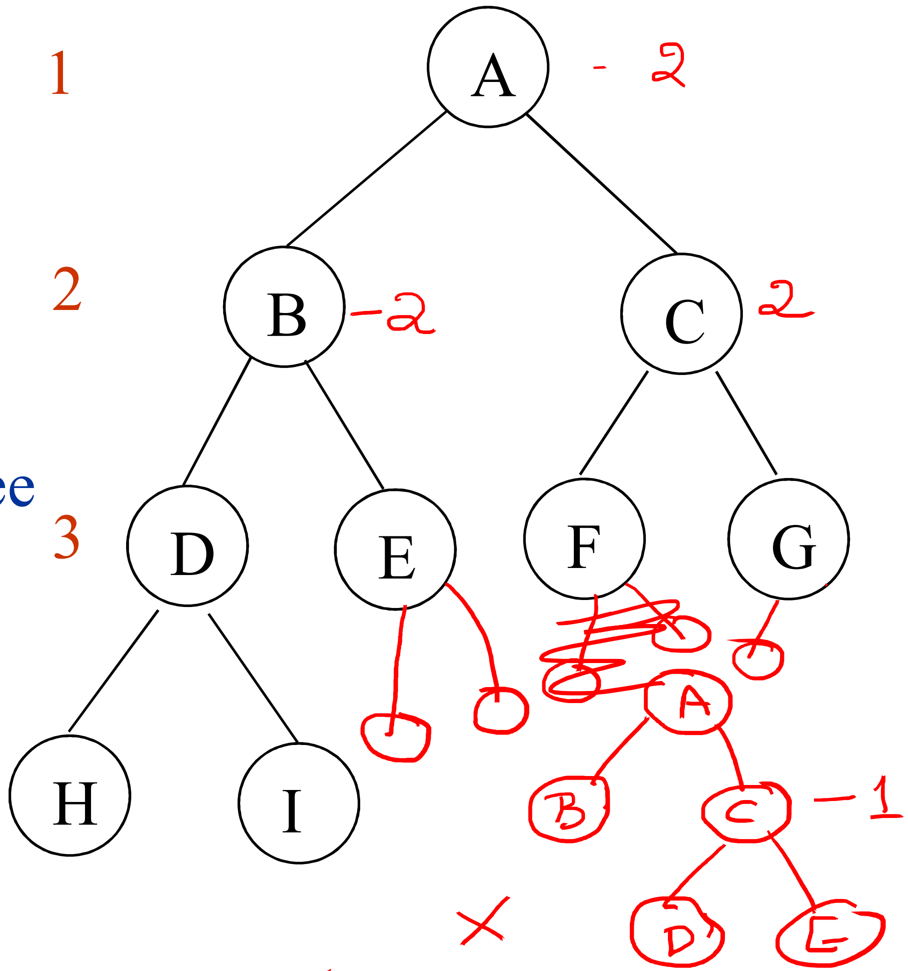
Bintree Rchild(*bt*)::= if (IsEmpty(*bt*)) return error
else return the right subtree of *bt*

Samples of Trees

Complete Binary Tree



Skewed Binary Tree

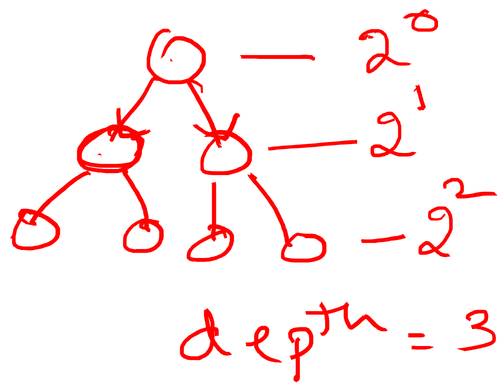


not a complete binary

Maximum Number of Nodes in BT

i-1

- The maximum number of nodes on level i of a binary tree is 2^{i-1}, i>=1. root is at level 1
- The maximum nubmer of nodes in a binary tree of depth k is 2^k-1, k>=1.



Prove by induction.

$$\sum_{i=1}^{k} 2^{i-1} = 2^k - 1$$

Relations between Number of Leaf Nodes and Nodes of Degree 2

For any nonempty binary tree, T, if n_0 is the number of leaf nodes and n_2 the number of nodes of degree 2, then $n_0=n_2+1$

proof:

Let n and B denote the total number of nodes & branches in T.

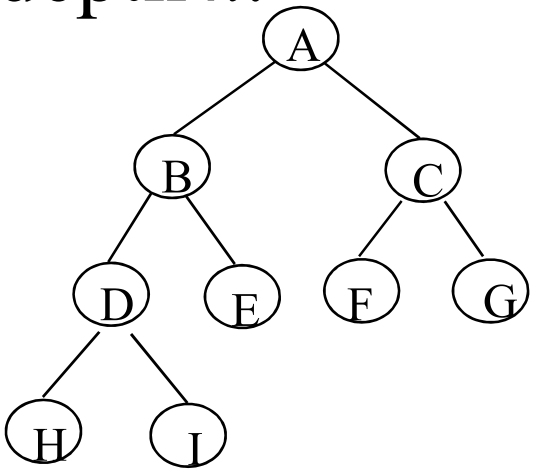
Let n_0, n_1, n_2 represent the nodes with no children single child, and two children respectively.

$n= n_0+n_1+n_2$, $B+1=n$, $B=n_1+2n_2$ ==> $n_1+2n_2+1= n$

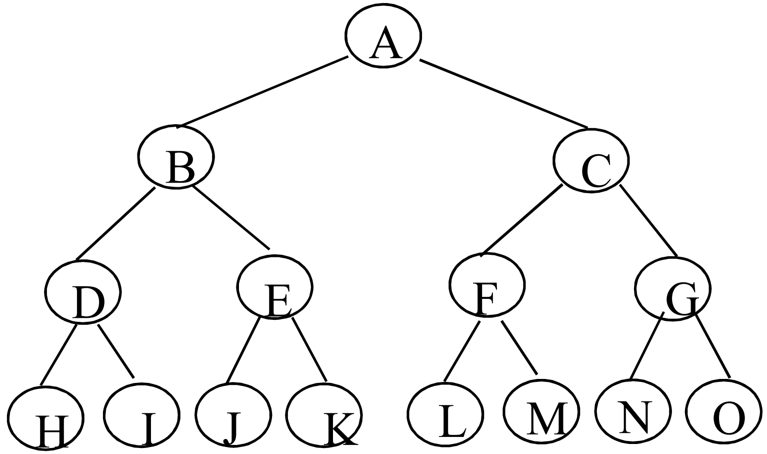
$n_1+2n_2+1= n_0+n_1+n_2$ ==> $n_0=n_2+1$

Full BT VS Complete BT

- A full binary tree of depth k is a binary tree of depth k having 2^k-1 nodes, k>=0.
- A binary tree with n nodes and depth k is complete *iff* its nodes correspond to the nodes numbered from 1 to n in the full binary tree of depth k.



Complete binary tree

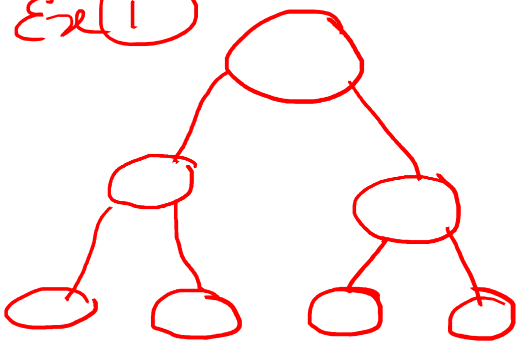


Full binary tree of depth 4

Binary Tree ~~Representations~~ Types

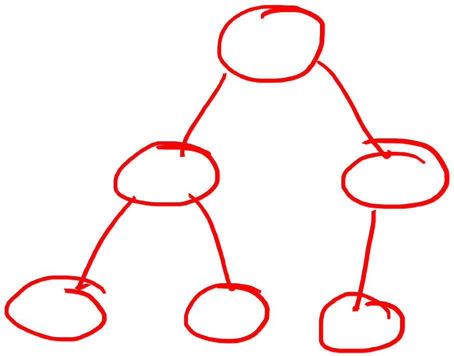
Complete BT

Ex 1



3 levels = $2^3 - 1 = 7$

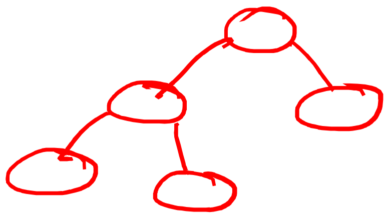
Fully BT (all the levels should be complete)



⇒ complete BT
not a fully BT

Strictly BT

0 or 2



BT Representation

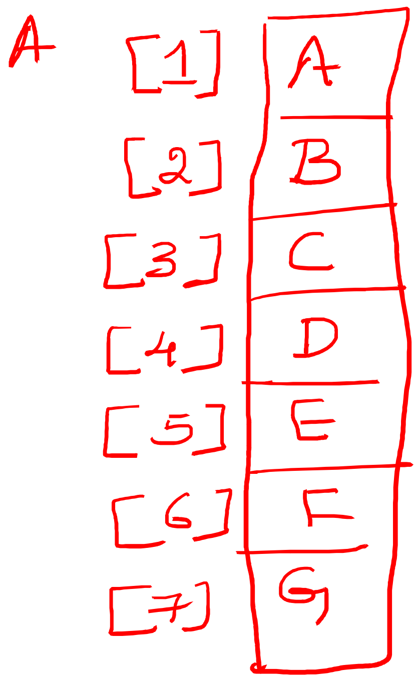
1. Sequential/Array

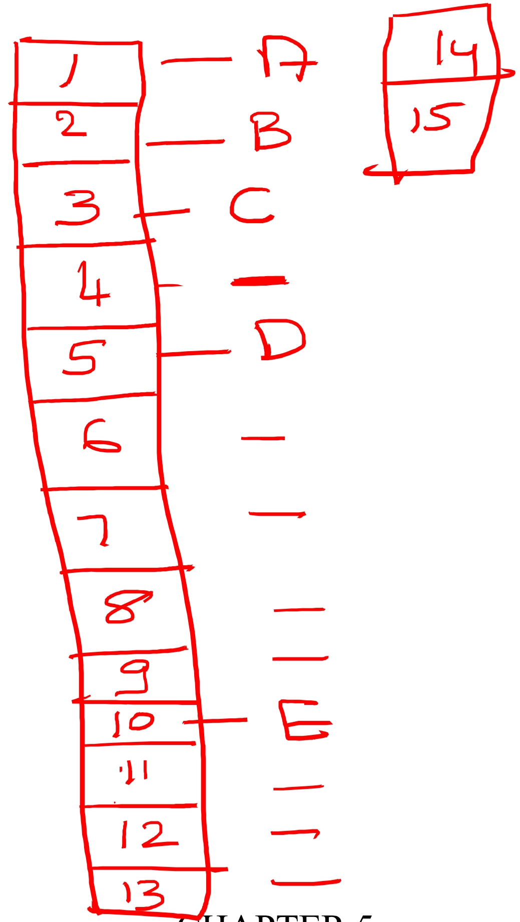
2. Linked list/Dynamic

Parent node i

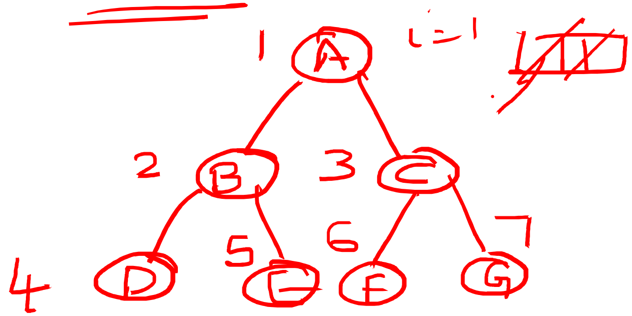
lchild $= 2i$

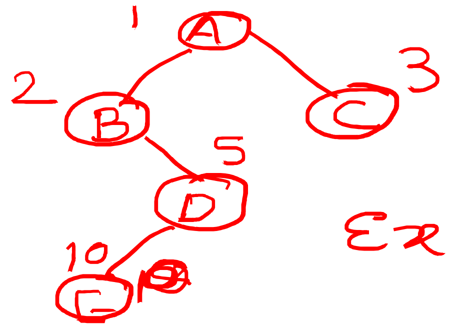
rchild $:= 2i+1$





Ex: 1





Ex-2

level = 4

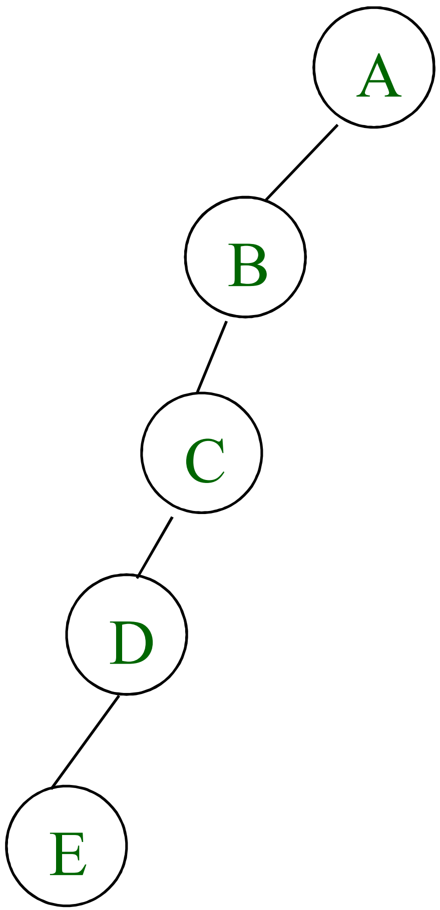
$2^4 - 1 = 15$

Binary Tree Representations

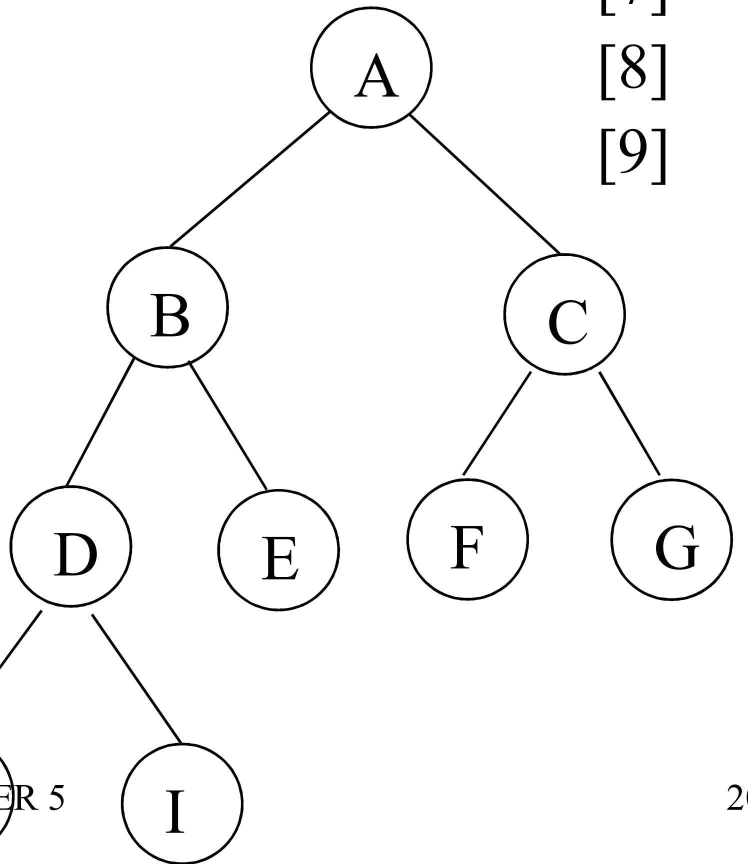
- If a complete binary tree with n nodes (depth = $\log n + 1$) is represented sequentially, then for any node with index i, $1 <= i <= n$, we have:
  - *parent*(i) is at $i/2$ if $i != 1$. If $i=1$, i is at the root and has no parent.
  - *left_child*(i) ia at $2i$ if $2i <= n$. If $2i > n$, then i has no left child.
  - *right_child*(i) ia at $2i+1$ if $2i + 1 <= n$. If $2i + 1 > n$, then i has no right child.

Sequential Representation

(1) waste space
(2) insertion/deletion problem



[1]	A
[2]	B
[3]	--
[4]	C
[5]	--
[6]	--
[7]	--
[8]	D
[9]	--
.	.
[16]	E



[1]	A
[2]	B
[3]	C
[4]	D
[5]	E
[6]	F
[7]	G
[8]	H
[9]	I

Linked Representation

```
typedef struct node *tree_pointer;
typedef struct node {
 int data;
 tree_pointer left_child, right_child;
};
```

left_child	data	right_child