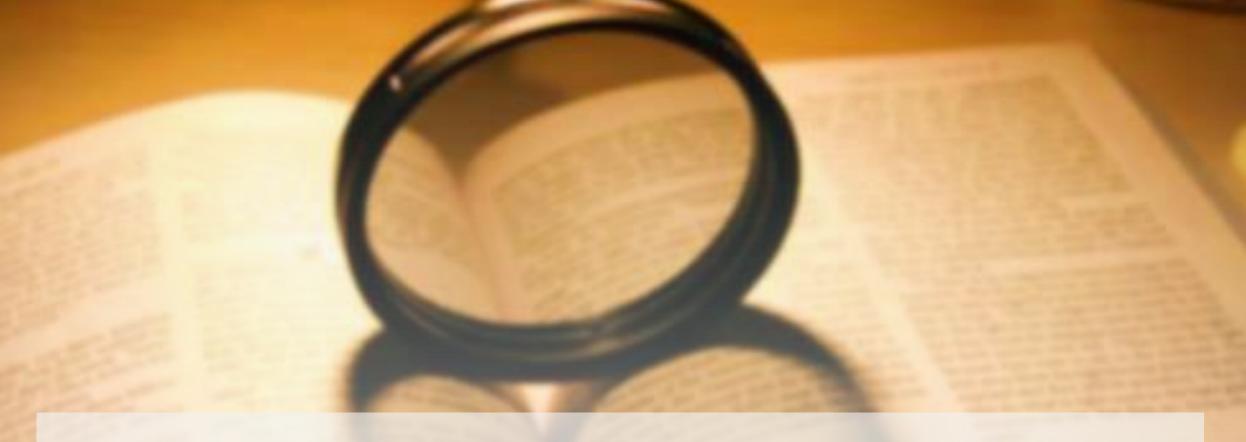
Data Structures
Chapter 5
Tree

- 1. introduction
- 2. Binary tree
- 3. Binary search tree
- 4. Tree balancing



다윗은 당시에 하나님의 뜻을 따라 섬기다가 잠들어 그 조상들과 함께 묻혀 썩음을 당하였으되. 행13:36

For when David had served God's purpose in his own generation, he fell asleep; he was buried with his fathers and his body decayed. Acts 13:36



다윗은 당시에 하나님의 뜻을 따라 섬기다가 잠들어 그 조상들과 함께 묻혀 썩음을 당하였으되. 행13:36

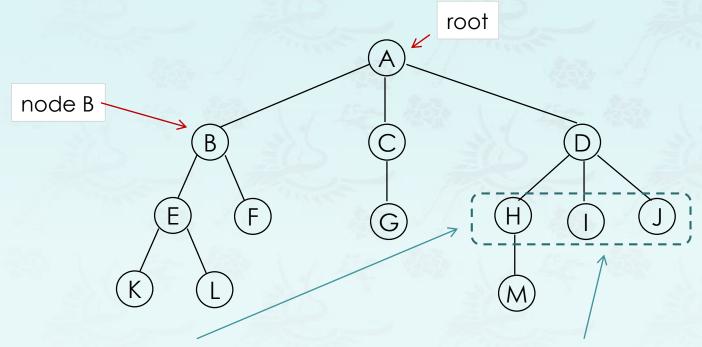
For when David had served God's purpose in his own generation, he fell asleep; he was buried with his fathers and his body decayed. Acts 13:36

하나님이 우리를 구원하사 거룩하신 소명으로 부르심은 우리의 행위대로 하심이 아니요 오직 자기의 뜻과 영원전부터 그리스도 예수 안에서 우리에게 주신 은혜대로 하심이라 (딤후1:9)

Data Structures Chapter 5 Tree

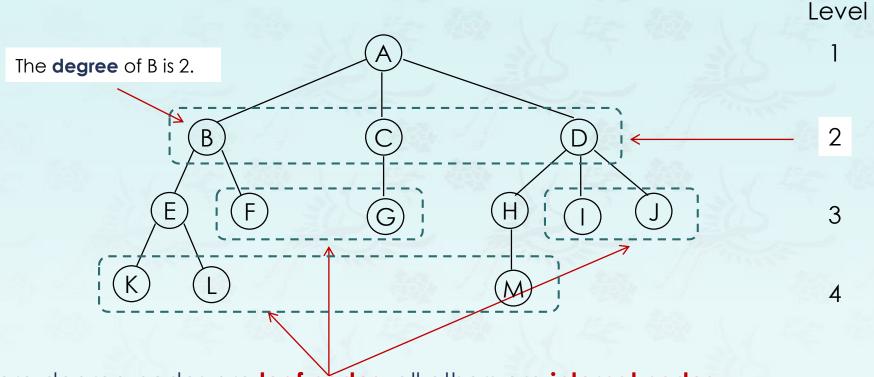
- 1. introduction
 - Definition and Terminology
- 2. Binary tree
- 3. Binary search tree
- 4. Tree balancing

- A tree data structure: it is like a linked list that has a first node, this node is called as the
 root of the tree.
- Example. A tree with a root storing the value 'A'



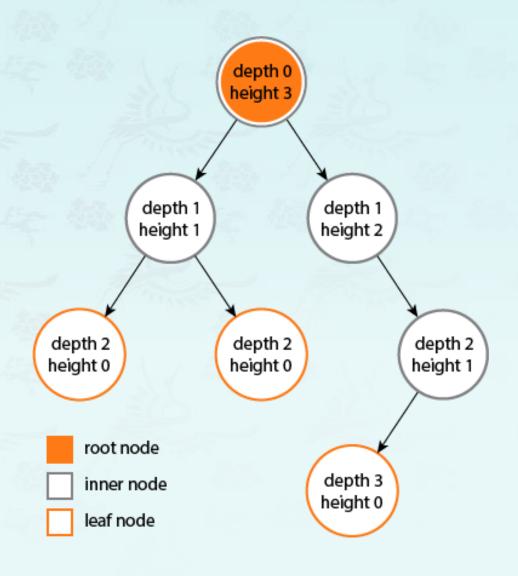
- The children of D are H, I, and J; H, I, and J are siblings.
- The parent of D is A.

Definition. child, parent, sibling, degree, leaf nodes, level, and internal node

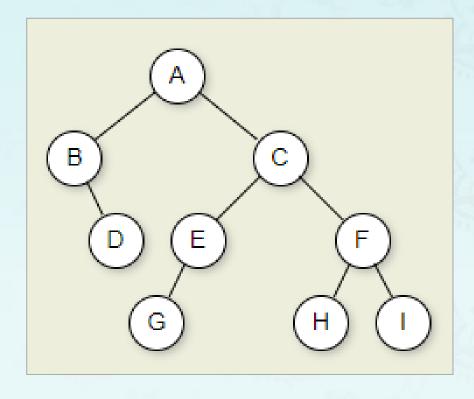


- Zero degree nodes are leaf nodes, all others are internal nodes.
 - An internal node is any node that has at least one non-empty child.
- The degree of a node is the number of children.
- The degree of a tree is the maximum of the degree of the nodes in the tree.

- Definition. height, depth, and level
- The height of a node is the number of edges on the longest path from the node to a leaf.
 - A leaf node will have a height of 0.
 - The height of a tree is the height of root.
 - The height of a tree with 1 node is 0.
 - The height of a tree is the maximum depth.
 - The height of a tree is the depth of the deepest node in the tree.
- The depth of a node is the number of edges from the node to the tree's root node.
 - A root node will have a depth of 0.
- The level of a node is defined by 1 + the number of connections between the node and the root.



Review



- A binary tree.
- Node A is the root.
- Nodes B and C are A's children.
- Nodes B and D together form a subtree.
- Node B has two children: Its left child is the empty tree and its right child is D.
- Nodes A, C, and E are ancestors of G.
- Nodes D, E, and F make up level 2 + 1 of the tree;
 node A is at level 0 + 1.
- The edges from A to C to E to G form a path of length 3.
- Nodes D, G, H, and I are leaves.
- Nodes A, B, C, E, and F are internal nodes.
- The depth of I is 3.
- The height of this tree is 3.

Introduction – Representation of trees

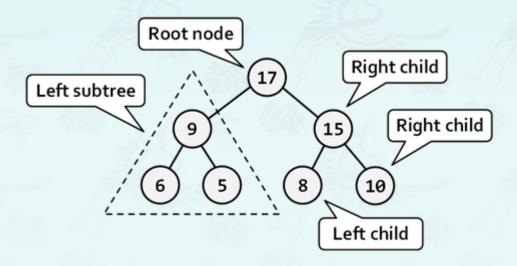
Exercise. The tree representing the HTML document below:

```
<html>
     <head>
           <title>Hello World!</title>
     </head>
     <body>
           <h1>This is a <u>Heading</u></h1>
           This is a paragraph with some <u>underlined</u> text.
     </body>
</html>
                                                              html
                                           head
                                                                                bodv
😻 Hello World! - Mozilla Firefox
  Edit View Go Bookmarks Tools Help
                                            title
          http://cheetah.vls.uwaterloo.ca/~dwh
Quest G G-Mail G G-News 🚁 GBM & ECE UG 🛊 CBC & SE 240 | "Hello World!"
                                                          "This is a "
This is a Heading
                                                                        "Heading"
This is a paragraph with some underlined text.
                                                                                                               text."
                                                                      "This is a paragraph with "
Done
                                                                                                  "underlined"
```

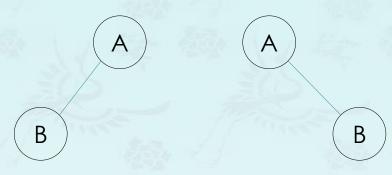
Data Structures Chapter 5 Tree

- 1. introduction
- 2. Binary tree
 - Definition and Properties
 - Traversal
 - Coding
- 3. Binary search tree
- 4. Tree balancing

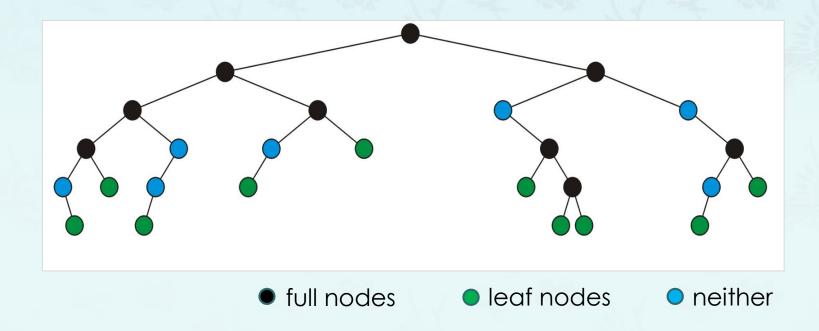
- Definition: A tree such that each node has exactly two children.
 - Notice, exactly two children not up to two children! Because exactly two children means a left child and/or right child, no middle child.
 - Each child is either empty or another binary tree.
 - Given this constraint, we can label the two children as left and right nodes or subtrees.



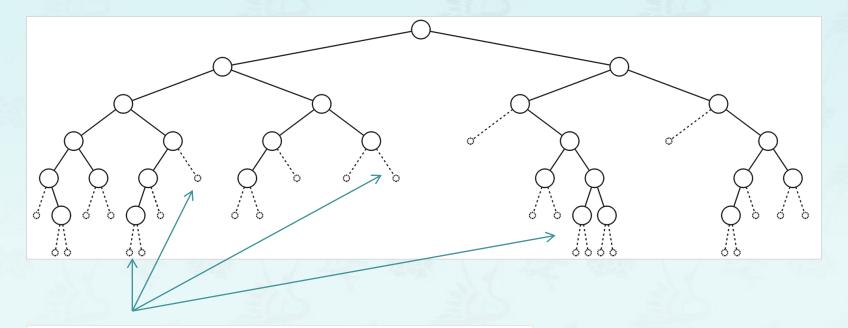
- **Example:** two binary trees with two nodes
 - Q: Are they two different binary trees?
 - A: Yes!



- Definition: A full node is a node where both left and right sub-trees are non-empty trees:
 - Q: How many full nodes are there?
 - Q: How many leaf nodes are there?
 - Q: What is the height of the tree?
 - Q: What is the degree of the tree?



 Definition: An empty node or null sub-tree is a location where a new leaf node (or a sub-tree) could be inserted.



null link, graphically, the missing branches

Binary trees - ADT

- Objects: a finite set of nodes either empty or consisting of a root node, leftBinaryTree, and rightBinaryTree.
- Functions:

boolean empty(bt)

binaryTree new Node{key, left, right}

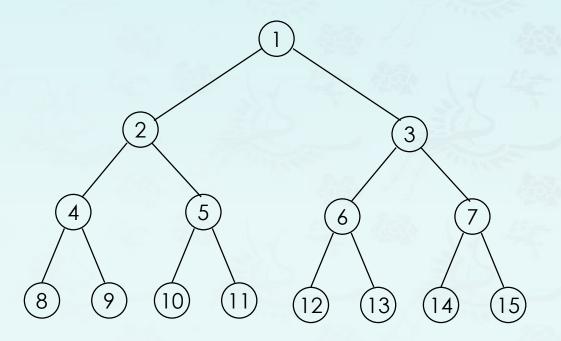
binaryTree left(bt)

element getKey(bt)

binaryTree right(bt)

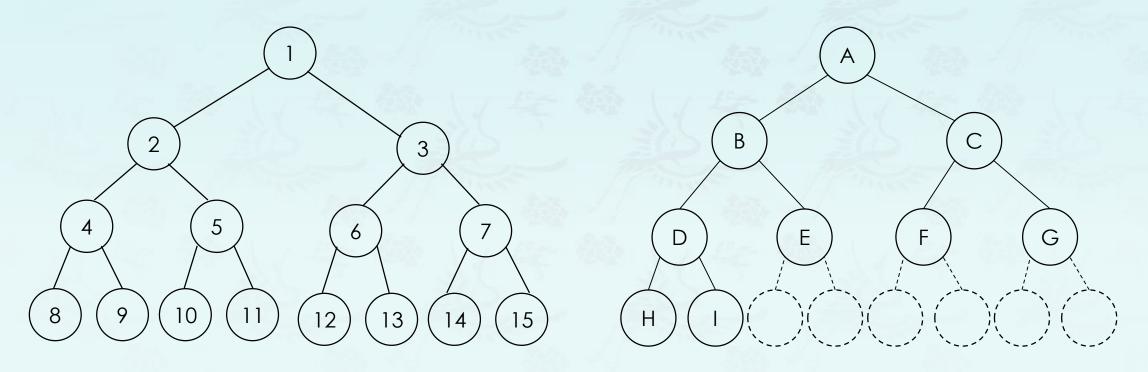
Observation:

- Q: Maximum number of nodes in binary trees in each level and all levels?
- Q: What is the max level k if there are n nodes? k(n) = ?



| Level | Node Numbers at Each Level | Total Numbers of Nodes |
|-------|-------------------------------|---------------------------|
| 1 | $1 = 2^0$ | $1 = 2^1 - 1$ |
| 2 | $2 = 2^1$ | $3 = 2^2 - 1$ |
| 3 | $4 = 2^2$ | $7 = 2^3 - 1$ |
| 4 | $8 = 2^3$ | $15 = 2^4 - 1$ |
| | | |
| 11 | $1024 = 2^{10}$ | $2047 = 2^{11} - 1$ |
| | | |
| k | 2^{k-1} | $2^k - 1$ |
| h | 2 ^h | 2 ^{h+1} - 1 |

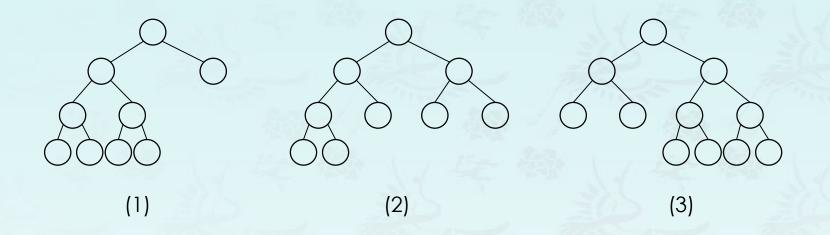
- **Definition:** A full binary tree of level k is a binary tree having $2^k 1$ nodes, $k \ge 0$.
- **Definition**: A binary tree with n nodes and level k is **complete** iff its nodes correspond to the nodes numbered from 1 to n in the full binary tree of level k.



A **full** binary tree

A **complete** binary tree

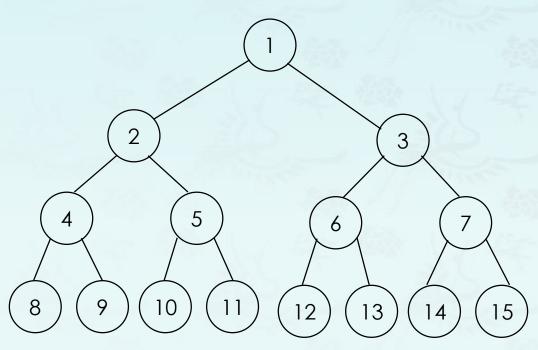
Q: Identify a complete binary tree.



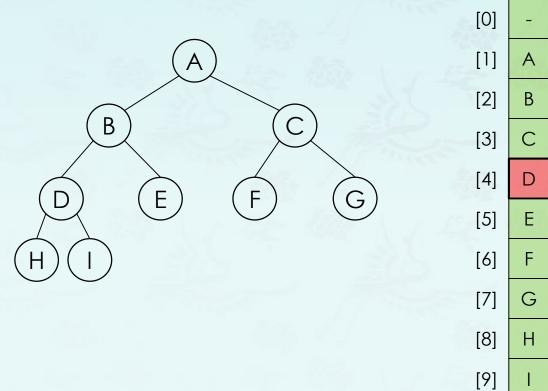
Q: Meanings of a complete tree in terms of ADT?

A: Removals of a node are only allowed from the "last" position. There is one position available to insert a node every time!

- Q: What is the potential problem when you use an one-dimensional array to represent a binary tree in memory?
- A: It is good for a full binary tree, but not good memory usage for a skewed or complete binary tree.



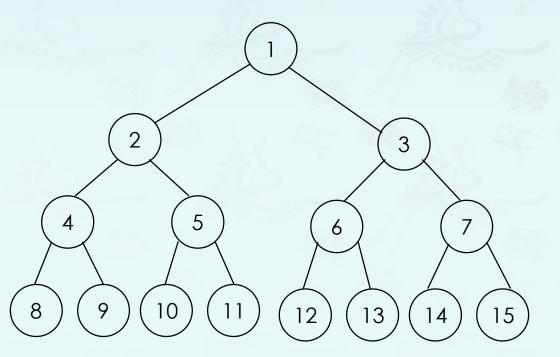
• Q: Let's suppose that you have a **complete binary tree** in an array. Find its parent, left child and right child at node D.



Solution:

parent(x = 4) is at 4/2 = 2 leftChild(4) is at 2x4 = 8rightChild(4) is at 2x4 + 1 = 9

- Q: Let's suppose that you have a complete binary tree in an array, how can we locate node x's parent or child?
- A complete binary tree with n nodes, any node index i, $1 \le i \le n$, we have
 - parent(i) is at $\lfloor i/2 \rfloor$ If i = 1, i is at the root and has no parent
 - leftChild(i) is at 2i if 2i <= n. If 2i > n, then i has no left child.
 - rightChild(i) is at 2i+1 if 2i+1 <= n. If 2i+1 > n, then i has no right child.

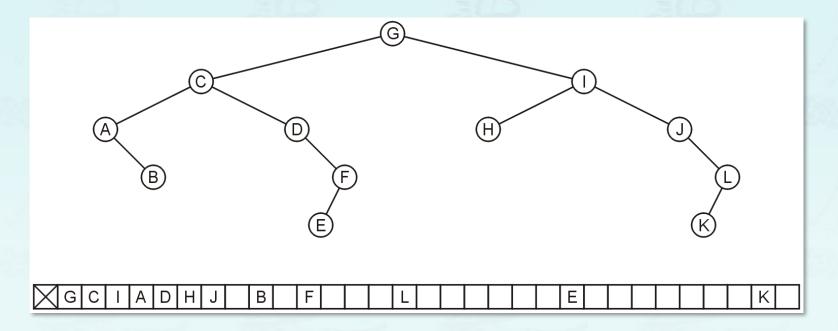


Wow! Can we use this to all binary trees? Why not?

Problem remains:

The problem with storing an arbitrary binary tree using an array is the inefficiency in memory usage.

- Q: Can we use this array rep. to store all binary trees? Why not?
- **A: For example,** the tree has 12 nodes, and requires an array of 32 elements. Adding one extra node, as a child of node K or E **doubles** the required memory for the array!



- A. In the worst case a skewed tree of level k will require $2^k 1$ space which is $O(2^k)$. Of these, only k will be used.
- Q. What happens when k = n? (Is there such a tree?)

- (1) The maximum number of **nodes on level k** of a binary tree is 2^{k-1} , $k \ge 1$
- (2) The maximum number of **nodes in a binary tree of level k** is $2^k 1$, $k \ge 1$
- (3) The maximum level of a **complete binary tree** with **n** nodes is $k(n) = \lceil \log_2 (n+1) \rceil$, $\lceil x \rceil$ is the smallest integer $\geq x$.

$$n = 2^k - 1$$

$$n + 1 = 2^k$$

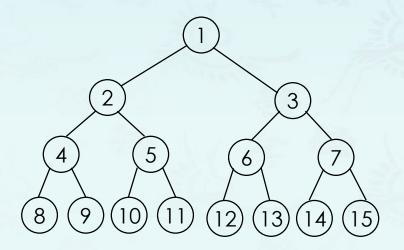
$$\log(n+1) = \log 2^k$$

$$\log(n+1) = k$$

$$k(n) = \lceil \log(n+1) \rceil \qquad \text{since k is an integer, and includes}$$

$$k(n) = \lceil \log(n) \rceil + 1 \qquad \text{the max level of complete binary tree.}$$

- Observation: The max level of a full binary tree of n nodes is $k = \lfloor \log(n) \rfloor + 1$:
 - Many operations with trees have a run time that goes with the max level of some path within the tree;
 - If we have a full binary tree (or something close to it), we know that those operations run in $O(\log n)$.



B C C F G H I () () () () () ()

A full binary tree

A complete binary tree

Binary trees - Linked representation

Node representations:

```
struct TreeNode{
    int         key;
    TreeNode* left;
    TreeNode* right;
};
using tree = TreeNode*;

TreeNode* t = new TreeNode(9);
tree t = new TreeNode(9);
```

Recursion & Tree Structure

```
struct TreeNode{
    int         key;
    TreeNode* left;
    TreeNode* right;
};
using tree = TreeNode*;
```

```
struct TreeNode{
  int          key;
  TreeNode* left;
  TreeNode* right;

TreeNode(int k, TreeNode* l, TreeNode* r) {
     key = k; left = l; right = r;
  }
  TreeNode(int k) : key(k), left(nullptr), right(nullptr) {}

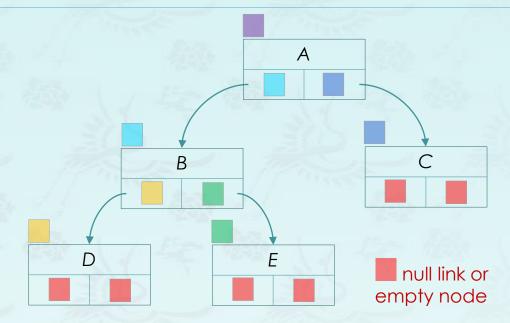
  ~TreeNode(){}
};
using tree = TreeNode*;
```

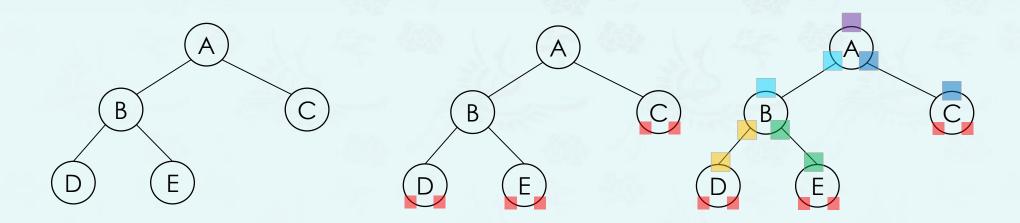
Binary trees - Linked representation

Node representations:

```
struct TreeNode{
    int         key;
    TreeNode* left;
    TreeNode* right;
};
using tree = TreeNode*;
```

```
TreeNode* t = new TreeNode(9);
tree t = new TreeNode(9);
```



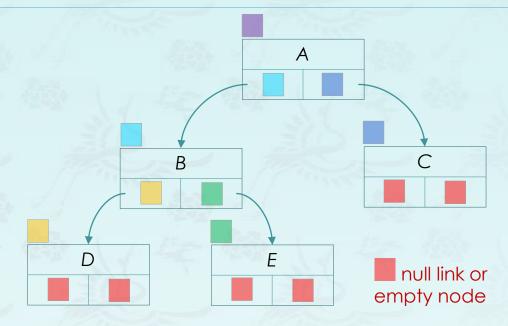


Binary trees – Linked representation

Node representations:

```
struct TreeNode{
    int         key;
    TreeNode* left;
    TreeNode* right;
};
using tree = TreeNode*;
```

```
TreeNode* t = new TreeNode(9);
tree t = new TreeNode(9);
```



- Q. Is this node structure good enough?
 - Not easy to find its parent node. Parent field could be added if necessary.
- Q. It is similar to a doubly-linked list(DLL). What is different?
 - One head, but many tails. Null points empty node conceptually.

THE PROPERTY OF THE PROPERTY O

Data Structures Chapter 5 Tree

- I. introduction
 - Definition and Terminology
- 2. Binary tree
 - Definition and Properties
 - Traversal
 - Coding
- 3. Binary search tree
- 4. Tree balancing