Chapter 4: Binary Search Tree

Lâm Hoài Bảo - FSE – CICT
Trương Minh Thái – FSE - CICT

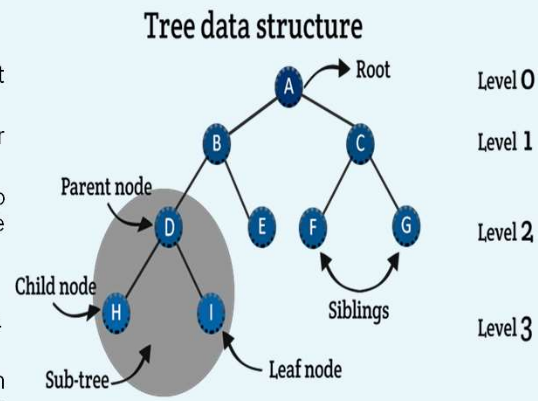
Agenda

- Concepts
- Binary tree
- Binary Search Tree (BST)
- Summary

Introduction

The Tree is a hierarchical DS

- A finite set of elements is called nodes. There is a distinct node called the root node.
- Each node represents an element in the set under consideration.
- In this set of nodes, they have the "parent-child" relationship called parenthood, which defines the structural system on the nodes.
- Each node, except the root node, has only one parent node.
- A node can have many child nodes (Child) or no child nodes. Groups of nodes with the same parent are called Siblings.
- The parent-child relationship is represented by the convention that the parent node is on the top line, the child node is on the bottom line and is connected by a line segment.
- Part of a tree data structure (Sub-tree) can be viewed as a complete tree



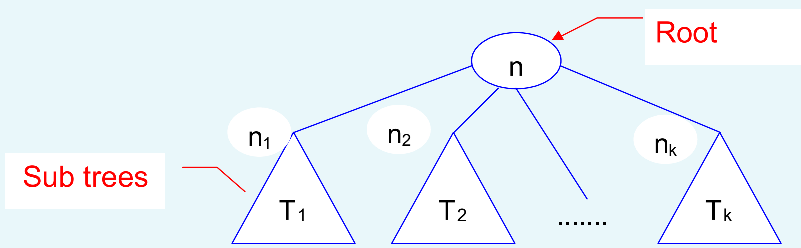
Characteristics

- A tree DS is a collection of objects (or entities) called nodes which are connected together to illustrate a hierarchy
- Is a non-linear DS since elements are distributed in different levels

CANTHO UNIVERSITY

Recursive definition

- A single node is a tree, that node is called the root of the tree.
- Given a single node n and k distinct trees T_1, T_2, ...T_k of which roots are n_1, n_2,...n_k, respectively. It possible to create a new tree of root is n and its sub trees are T_1, T_2, ... T_k.



CANTHO UNIVERSITY

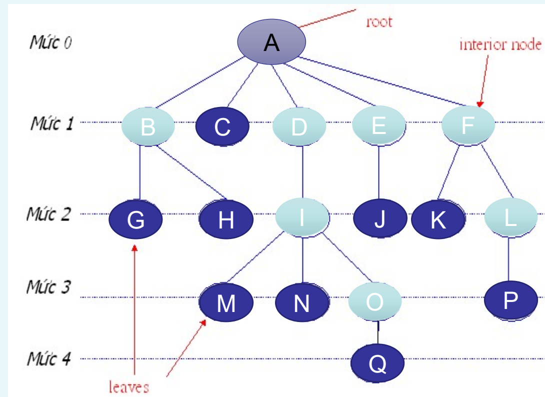
Terminologies

- The labeled tree
  – Stores the association of a label or value with a node in the tree
  – The label is the value stored at that node.
  – The key of a node can be just a part of this stored content in the node.
  – Note: The label is sometimes also called the key of the node.

CANTHO UNIVERSITY

Terminologies

- The root node is a node without a parent.
- A leaf node is a node that has no children.
- Interior node: is not a leaf node and is not a root node
- Degree of a node: number of sub trees of that node
  - Example: degree(I) = 3, degree(F) = 2
  - Interior node: node with degree != 0 and not the root



- A is the root node, there is no true ancestor
- C, G, H, J, K, M, N, P, and Q have no real descendants so are called leaf nodes
- B, D, E, F, I, L, and O are neither leaf nodes nor root nodes, so they are called interior nodes.

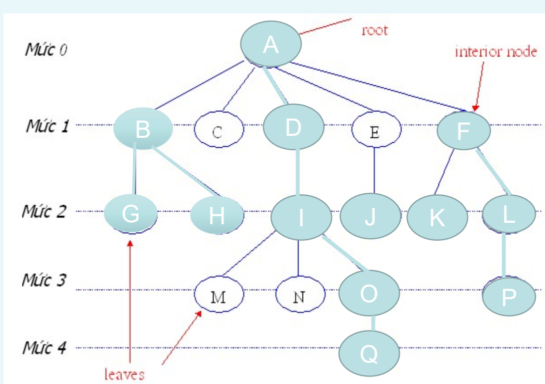
www.ctu.edu.vn 7

7

CANTHO UNIVERSITY

Terminologies

- A subtree of a tree is a node along with all its descendants
- The height of a node is the length of the maximum path from that node to the leaf
- The height of the tree is the height of the root node.
- The depth (level) of a node is the length of the path from the root node to that node
- (note: Nodes with the same depth i are called nodes with the same level i)



- B, G, and H are a subtree of the tree whose root node is A
- The height of node A is 4 (the maximum path length from A to leaf is 4: A, D, I, O, Q).
- The height of F = 2.
- The height of the tree in the figure below = height of node A = 4
- A is the root node with depth = 0 (level 0)
- G, H, I, J, K, and L have depth 2 and we call them the same level 2.

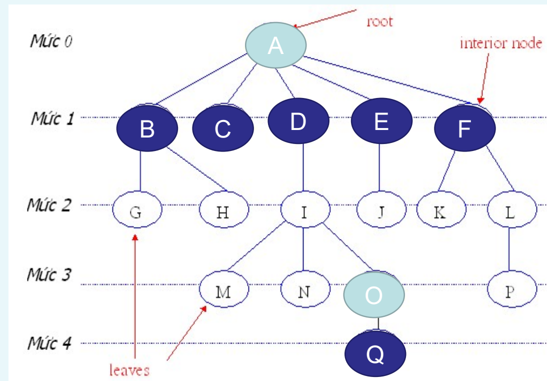
www.ctu.edu.vn 8

8

Terminologies

CANTHO UNIVERSITY

- Degrees of nodes and trees
  - The degree of a node is the number of subtrees of that node, and the degree of a leaf node = 0.
  - The degree of a tree is the largest degree of the nodes in the tree.
  - An n-ary tree is a tree with degree n.



- Node A has a degree of 5 (degree 5), node C has a degree of 0 and node O has a degree of 1
- The tree with root node A (tree A) in the figure has a degree of 5. Subtree D has a degree of 3
- The degree of tree A is 5, which is called a 5-ary tree.

www.ctu.edu.vn 9

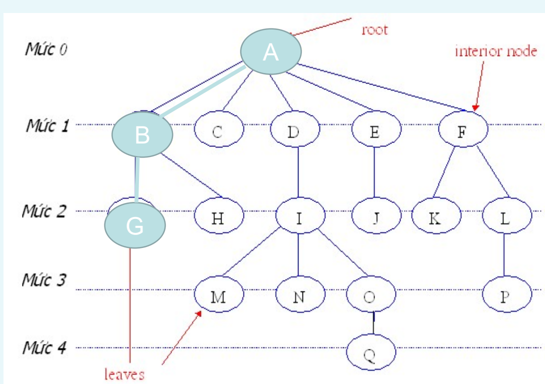
9

Terminologies

CANTHO UNIVERSITY

The path in the tree:

- Path is a series of node n_1, n_2, ..., n_k such that n_i is parent of n_{i+1} (i=1.. k-1).
- The path length is defined as the number of nodes on the path minus 1 (length=k-1, k is the number of nodes)
- If there is a path from node a to node b, then we say a is the ancestor of b, and b is called the descendant of node a



- A, B, G is a path from node A to G and has a path length of 3-1 =2
- A is the root node, there is no real predecessor
- There is a path from B to G, so B is a true ancestor of G and G is a true descendant of B

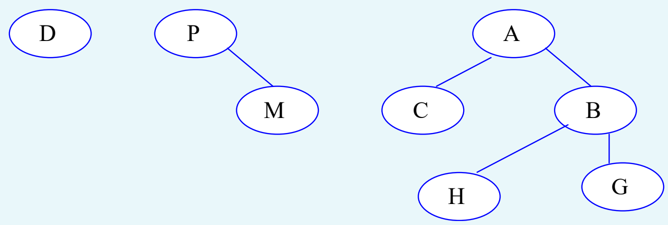
www.ctu.edu.vn 10

10

Terminologies

CANTHO UNIVERSITY

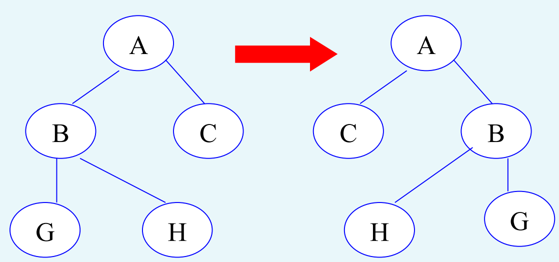
- Forest
  - A forest is a collection of many trees.



Terminologies

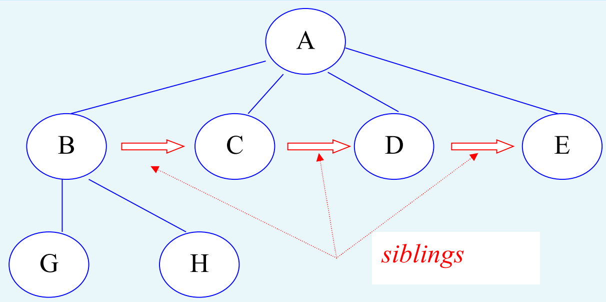
CANTHO UNIVERSITY

- Trees have order
  - If we distinguish the order of nodes in the same tree, we call it ordered. Otherwise, it is called an unordered tree.
  - In the tree, there is order, the conventional order is from left to right.



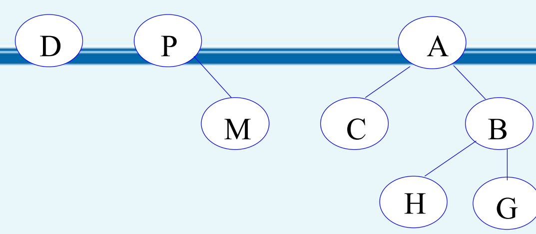
The two order trees have different orders

Terminologies

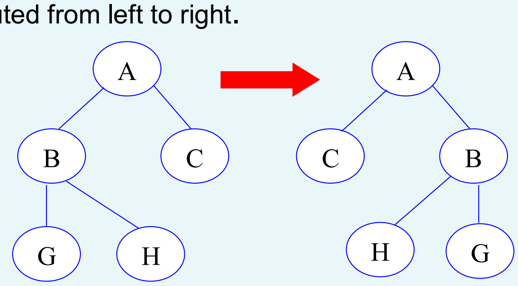


- The child nodes with the same parent node are called sibling nodes.
- Expanding the "left to right" relationship of siblings:
  - If a and b are siblings and a is on the left of b then the descendants of a are "on the left" of all descendants of b

Terminologies



- A forest is as collection of trees
- Order tree
  - In general, nodes in a tree are order distinctions.
  - Order is computed from left to right.

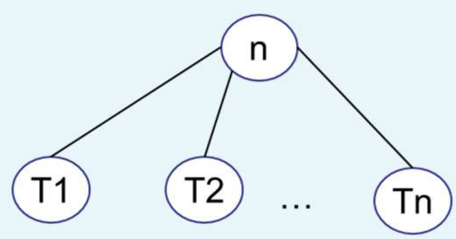


Tree traversal

- Traversal is a way to visit nodes of a tree depending on an order.
- List of traversed nodes: is a list of nodes which the traversal procedure visits.
- Traversal methods:
  - Depth First Search (DFS)
    - Preorder
    - Inorder
    - Postorder
  - Breadth First Search (BFS)
    - Level

Tree traversal

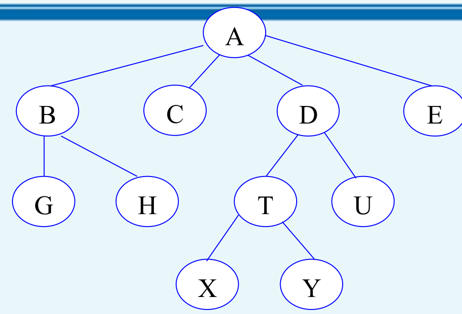
- DFS
  - Empty tree: all traversed expressions are empty.
  - Else, assume T has the root n and sub trees T_1, T_2 ,…,T_k
    - Preorder expression of T is n, follow by Preorder expression of T_1, T_2 ,…,T_k , respectively.
    - Inorder expression of T is Inorder expression of T_1, then n, follow by Inorder expression of T_2 ,…,T_k , respectively.
    - Postorder expression of T is Postorder expressions of T_1, T_2 ,…,T_k , respectively; follow by n.



- BFS
  - For each level of the tree
    - List all nodes from left to right

CANTHO UNIVERSITY

Example of traversal



- Preorder expression: A B G H C D T X Y U E
- Inorder expression: G B H A C X T Y D U E
- Posorder expression G H B C X Y T U D E A
- Level expression A B C D E G H T U X Y

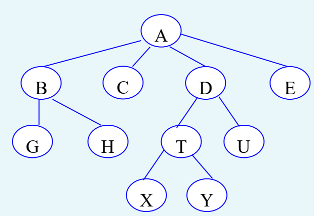
www.ctu.edu.vn 17

17

CANTHO UNIVERSITY

Recursive traversal algorithms

```
void PREORDER(node n){
    print "node n";
    for (each subtree c of node n in order from left to right)
        PREORDER(c);
} //PREORDER
```



Duyệt tiền tự:

A B G H C D T X Y U E

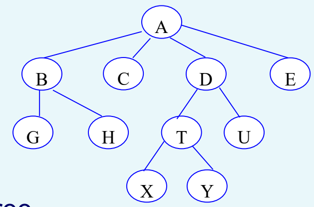
www.ctu.edu.vn 18

18

Recursive traversal algorithms

CANTHO UNIVERSITY

```
void INORDER(node n){
    if (n is leaf)
        print "node n"
    else {
        INORDER(leftmost child of n)
        print "node n";
        for(each subtree c of node n, except the leftmost subtree,
                in order from left to right)
              INORDER(c);
     }
} //INORDER
```

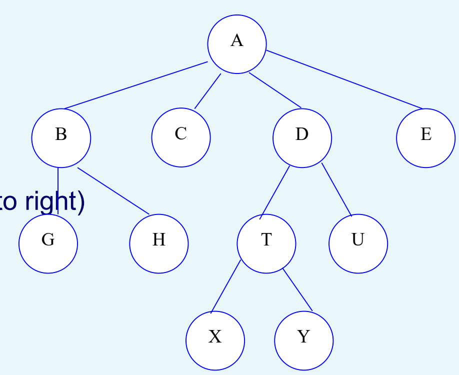


Duyệt trung tự:
G B H **A** C X T Y D U E

Recursive traversal algorithms

CANTHO UNIVERSITY

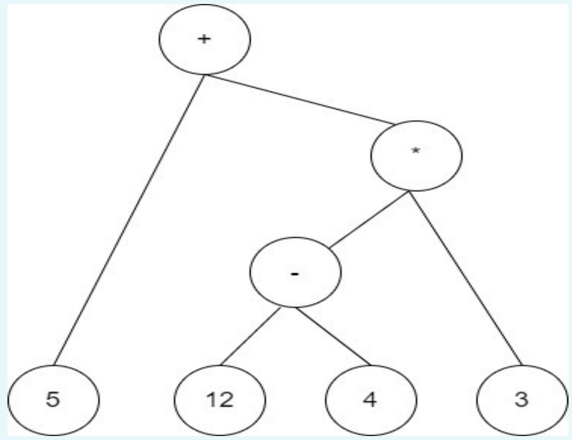
```
void POSORDER(node n){
    if (n is leaf)
        print "node n"
    else {
        for (each child node c of node n, in order from left to right)
          POSORDER(c);
        print "node n";
    }
}; //POSORDER
```



Duyệt hậu tự:
G H B C X Y T U D E **A**

Expression tree

- An expression tree represents an expression where each node describes either an operand or an operator.
  - 5+(12-4)*3
- An expression with binary operators can be illustrated as a binary tree



Construct an expression tree (home work)

- Specify priority of each operator: ‘(‘: 0, ‘+’, ‘-’: 1, ‘*’, ‘/’: 2
- Maintain 2 stack: ops for operators, nodes for nodes of the expression
- For each token in the expression st:
  - If token is ‘(‘, add it to the ops
  - If token is an operand, create a node of that token and add to nodes
  - If token is ‘)
    - While token at the top of ops is not ‘(‘
      - Pop 2 nodes from stack nodes
      - Pop an operator from stack ops
      - Create a new node: the root is the operator, 2 children are 2 recent popped nodes.
      - Add new node to stack nodes
    - Pop ‘(‘ from ops

Construct an expression tree (cont)

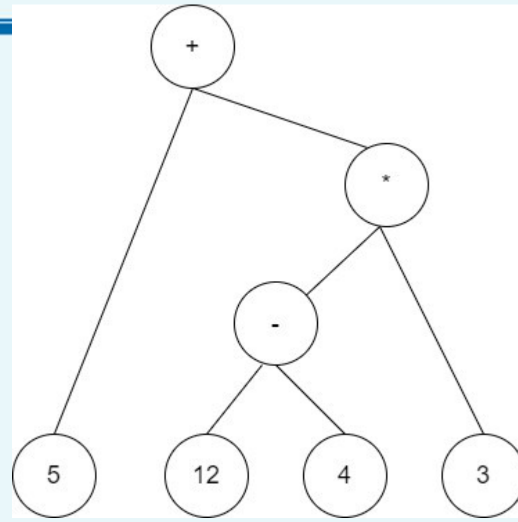
- For each token in the expression st:
  - If token is '(', …
  - If token is an operand, …
  - If token is ') …
  - Is token is an operator
    - While priority(operator at the top of ops) >= priority(token)
      - Pop 2 nodes from stack nodes, pop an operator from ops
      - Create a new node and add the new node to stack nodes.
    - If token is an operator
      - Create a new node
      - Pop 2 values from the stack, make them be children of the new node, push the new node to the stack
      - Push token to stack ops
- If size(nodes) > 1, repeat popping 2 nodes from nodes, 1 operator from ops, create a new node and push to stack nodes.

Agenda

- Concepts
- Binary tree
- Binary Search Tree (BST)
- Summary

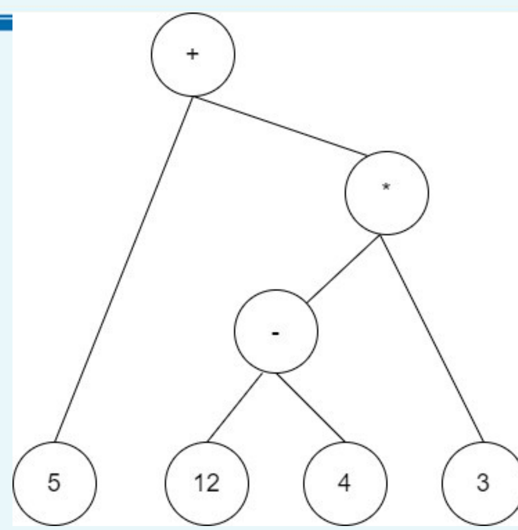
Binary tree

- A tree with degree 2
  - A node may 2 children: one left child and one right child
- An expression tree with binary operators is an example of binary tree



Binary tree traversal

- DFS
  - Preorder (NLR): visit the root node, Preorder left child, Preorder right child
  - Inorder (LNR): Inorder left child, visit the root node, Inorder right child
  - Posorder (LRN): Posorder left child, Posorder right child, visit the root node
- BFS
  - Visit nodes in each level of the tree

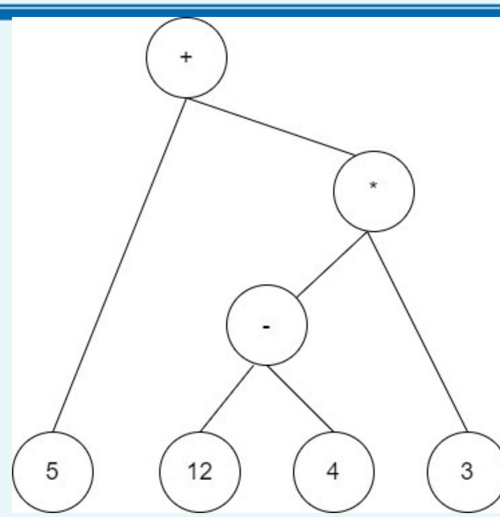


CANTHO UNIVERSITY

Expression tree traversal

- Preorder ~ prefix expression
- Inorder ~ infix expression
- Posorder ~ postfix expression

- Example
  - Prefix: + 5 * - 12 4 3
  - Infix: 5 + 12 - 4 * 3
  - Postfix: 5 12 4 - 3 * +



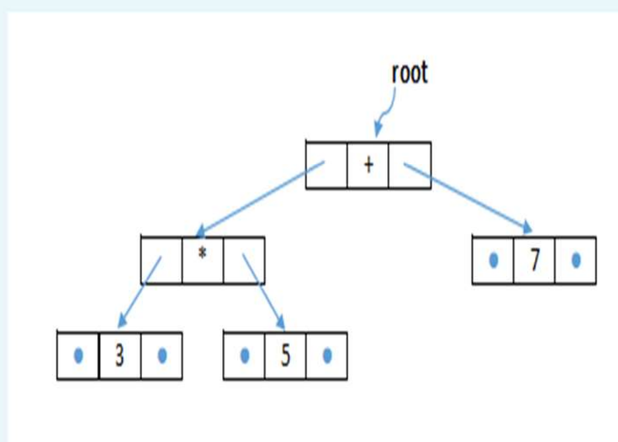
CANTHO UNIVERSITY

Declaration

```c
typedef … DataType;
struct NodeTag{
    DataType data;
    struct NodeTag* left;
    struct NodeTag* right;
} Node;
typedef Node* Tree;
```

Initialize an empty tree

```c
        void makenull(Tree *pT){
    (*pT) = NULL;
   }
```



Create a tree

- Create a tree from 2 sub trees where the root is the node with value v
- Algorithm

```
Tree create(DataType v, Tree l, Tree r){
    Tree N;
    N=(Node*)malloc(sizeof(Node));
    N->data = v;
    N->left = l;
    N->right = r;
    return N;
}
```

www.ctu.edu.vn 29

29

Binary tree installation

- **Get the left child**

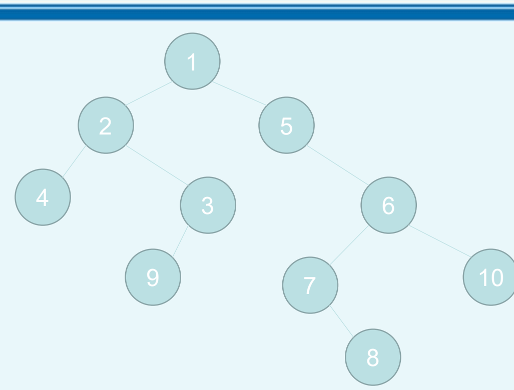
```
TTree LeftChild(Tree n){
    if (n!=NULL) return n->left;
    else return NULL;
}
```

- **Get the right child**

```
TTree RightChild(Tree n){
    if (n!=NULL) return n->right;
    else return NULL;
}
```

- **Check if a node is a leaf or not.**

```
int IsLeaf(Tree n){
    if(n!=NULL)
   return(LeftChild(n)==NULL)
                  &&(RightChild(n)==NULL);
   else return 0;
}
```



www.ctu.edu.vn 30

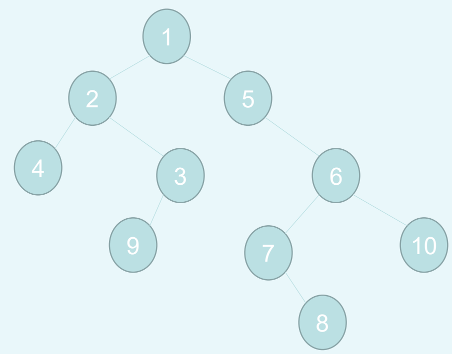
30

CANTHO UNIVERSITY

Binary tree installation

- PreOrder (NLR)

```
void NLR(Tree root){
    if (!emptyTree(root)){
         //If root is not NULL
        printf("%s ", root->data);
        LNR(root->left);
        LNR(root->right);
    }
}
```

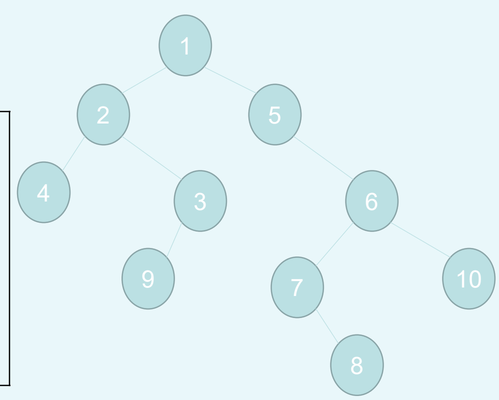


CANTHO UNIVERSITY

Binary tree installation

- InOrder (LNR)

```
void LNR(Tree root){
    if (!emptyTree(root)){
         //If root is not NULL
        LNR(root->left);
        printf("%s ", root->data);
        LNR(root->right);
    }
}
```

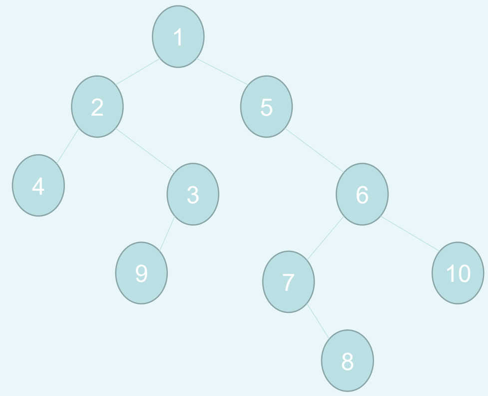


Binary tree installation

CANTHO UNIVERSITY

- **PosOrder (LRN)**

```
void LRN(Tree root){
    if (!emptyTree(root)){
         //If root is not NULL
        LNR(root->left);
        LNR(root->right);
        printf("%s ", root->data);
    }
}
```



www.ctu.edu.vn 33

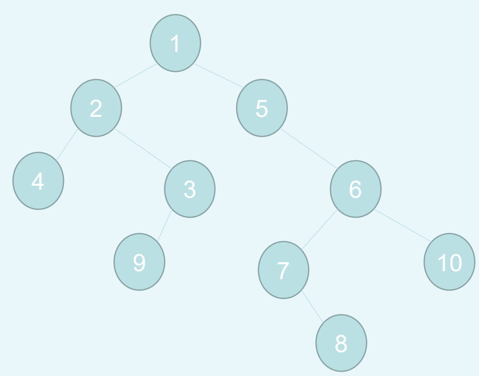
33

Binary tree installation

CANTHO UNIVERSITY

- **Determine the number of nodes in the tree**

```
int nb_nodes(Tree T){
  if(emptyTree(T))
     return 0;
  else
     return 1
          + nb_nodes(leftChild(T))
          + nb_nodes(rightChild(T));
  }
```



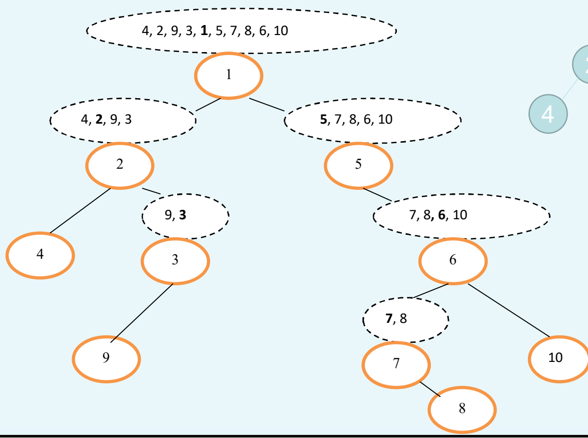
www.ctu.edu.vn 34

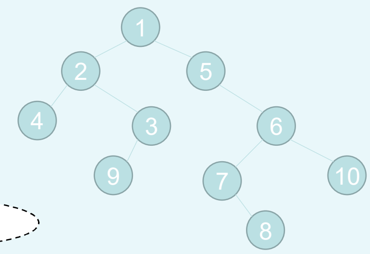
34

CANTHO UNIVERSITY

A binary tree construction

– PreOder(NLR): **1**, 2, 4, 3, 9, 5, 6, 7, 8, 10
– InOrder (LNR): 4, 2, 9, 3, **1**, 5, 7, 8, 6, 10
– PosOrder (LRN): 4, 9, 3, 2, 8, 7, 10, 6, 5, **1**





www.ctu.edu.vn 35

35

CANTHO UNIVERSITY

Agenda

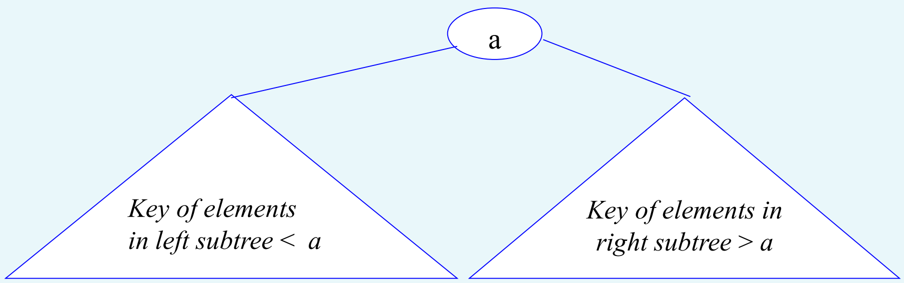
- Concepts
- Binary implementation
- Binary Search Tree (BST)
- Summary

www.ctu.edu.vn 36

36

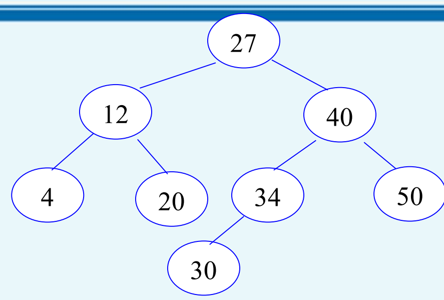
Binary search tree (BST)

- A BST tree is a binary tree in which the label (key) at each node is greater than the label (key) of all nodes in the left subtree and smaller than the label (key) of all nodes in the right subtree



Binary search tree (BST)

Example



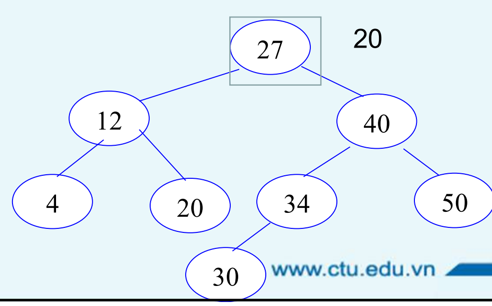
- In a binary search tree, there are no two nodes with the same key.
- The subtree of a binary search tree is a binary search tree.
- the traversal of Inorder to create a sequence of labels with increasing values: 4, 12, 20, 27, 30, 34, 40, 50

The BST installation

```
typedef ... KeyType;
typedef struct Node{
    KeyType Key;
    Node* Left,Right;
}
typedef Node* Tree;
```


The BST installation

- **Search for a node with key x**
  - Starting from the root node, we proceed with the following steps:
    - If the root node is NULL then key x is not in the tree.
    - If x is equal to the root node key, the algorithm stops because x has been found in the tree.
    - If x is less than the root node's key, then find (recursively) the key x on the left subtree
    - If x is greater than the root node's key, then find (recursively) the key x on the right subtree



The BST installation

```
Tree Search(KeyType x,Tree Root){
  if (Root == NULL) //không tìm thấy x
      return NULL;
  else if (Root->Key == x) // tìm thấy khoá x
      return Root;
 else if (Root->Key < x)
      //tìm tiếp trên cây bên phải
      return Search(x,Root->right);
 else
       //tìm tiếp trên cây bên trái
       return Search(x,Root->left);
 }
```


The BST installation

- Add a node with key x to the BST tree
  - Search to see if x already exists in the tree.
  - If found, the algorithm ends
  - If not found, add x to the tree.

  Adding x to BST must ensure that the properties of the BST are not disrupted..

Algorithm

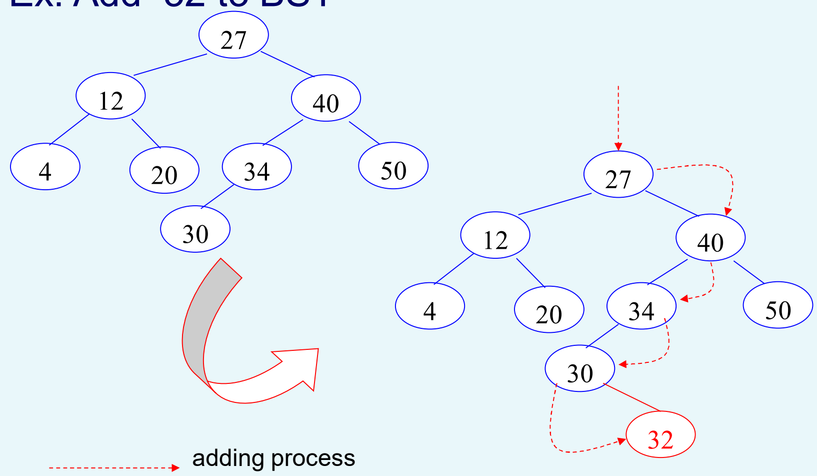
Starting from the root node, we proceed with the following steps:

- If the root node is NULL, then key x is not yet in the tree, so we add a new node.
- If x is equal to the root node key, the algorithm stops because x is already in the tree.
- If x is less than the root node's key, add (recursively) x to the left subtree
- If x is greater than the root node's key then add (recursively) x to the right subtree

CANTHO UNIVERSITY

The BST installation

- Ex: Add 32 to BST



CANTHO UNIVERSITY

The BST installation

```
void insertNode(KeyType X, Tree *T)
{
  if((*T) == NULL)
  {
     (*T)      = (Node*)malloc(sizeof(Node));
     (*T)->Key = X;
     (*T)->left = NULL;
     (*T)->right = NULL;
  }
  else
  if((*T)->Key == X)
       printf("Key %d is existing in BST", X);
  else
       if( (*T)->Key > X)
              InsertNode(X,&(*T)->left);
       else
              InsertNode(X,&(*T)->right);
}
```

The BST installation

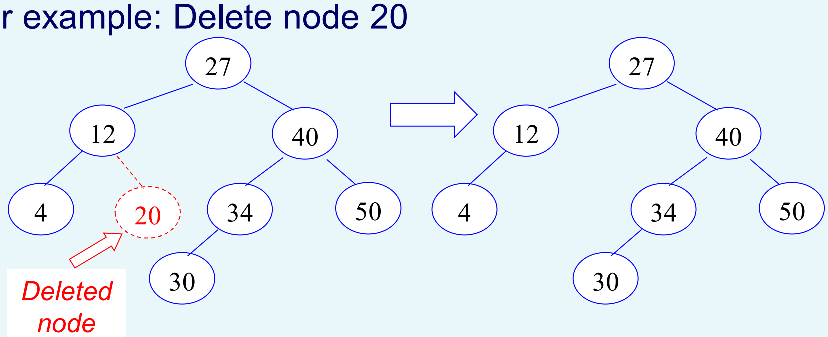
Delete a node with key x from the BST tree

- Find x in the tree.
  - If found x then delete the node containing key x.
  - Otherwise, the algorithm ends

Note that when deleting a node with key x, 3 cases can happen

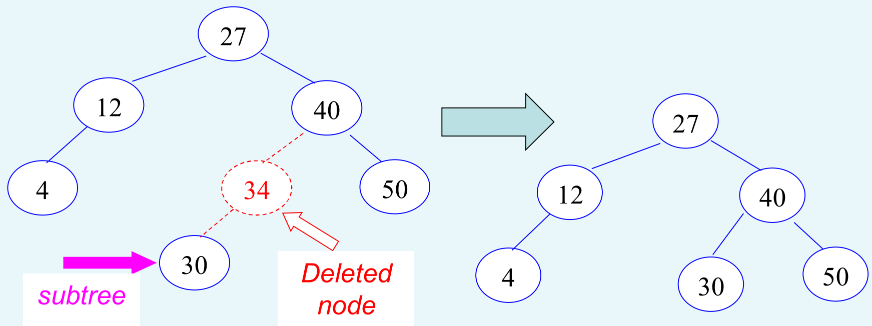
The BST installation

- **Case 1**
  - The node containing key x is a leaf node, then replace this node with NULL
  - For example: Delete node 20



The BST installation

- **Case 2**
  - The node containing key x has only one subtree, then replace this node with its subtree
  - For example, delete the node 34



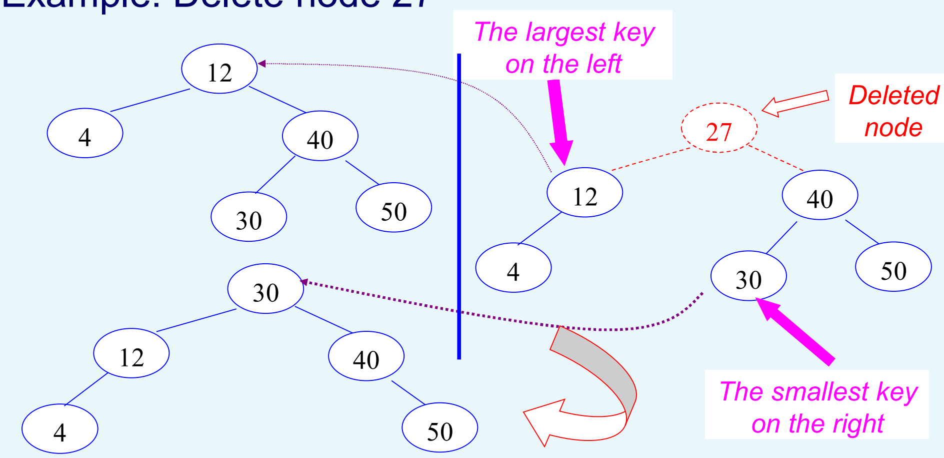
The BST installation

- **Case 3**
  - The node containing key x has two subtrees, then replace this node with one of two solutions:
    - The node with the largest key of the left subtree (rightmost node of the left subtree)
    - The node with the smallest key of the right subtree (leftmost node of the right subtree)

The BST installation

CANTHO UNIVERSITY

- Example: Delete node 27



www.ctu.edu.vn 49

49

The BST installation

CANTHO UNIVERSITY

Algorithm to delete a node with the smallest key

```
KeyType deleteMin(Tree *T)
{
 KeyType k;
 if((*T)->left == NULL)
 {
     k = (*T)->Key;
     (*T) = (*T)->right;
     return k;
 }
 else return deleteMin(&(*T)->left);
}
```

www.ctu.edu.vn 50

50

The BST installation

```
void deleteNode(KeyType X, Tree *T)
{
  if((*T)!=NULL) //Kiem tra cay khac rong
    if(X < (*T)->Key) //Hy vong X nam ben trai cua nut
        deleteNode(X,&(*T)->left);
    else
      if(X > (*T)->Key) //Hy vong X nam ben phai cua nut
          deleteNode(X,&(*T)->right);
      else // Tim thay khoa X tren cay
        if(((*T)->left==NULL)&&((*T)->right==NULL))//X la nut la
          (*T)=NULL; // Xoa nut X
        else // X co it nhat mot con
          if((*T)->left==NULL) //Chac chan co con phai
              (*T) = (*T)->right;
          else
            if((*T)->right==NULL) //Chac chan co con trai
                (*T) = (*T)->left;
            else  // X co hai con
                (*T)->Key = deleteMin(&(*T)->right);
}
```


Exercise

Write a function to delete a node

*void deleteNode(KeyType X, Tree *T)*

According to the strategy of the largest node of the left subtree

Exercise

Construct a binary tree given by two traversal lists as follows:

NLR: A,B,C,D,E,F,H,K,G,J,I

LNR: B,D,C,E,A,H,K,F,J,G,I

Exercise

a. Construct a BST given by lists as follows:

90, 30, 50, 10, 25, 35, 20, ~~30~~, 15, 80, 75, 45, 65, 5, 55, 100.

b. Reconstruct BST after deleting 35, deleting 65, adding 43, deleting 50.

Some applications

- Describe hierarchy: Example: file system
- **Organize data**: insert, delete, look up effectively
- Trie: store dictionary
- Heap: an array based tree implementing priority queue
- B/B+ tree: database index
- Routing table

...

Agenda

- Concepts
- Binary tree
- Binary Search Tree (BST)
- Summary

CANTHO UNIVERSITY

Summary

- Tree: a DS to manipulate hierarchical data
- Expression tree is a tree representing an expression
- Remaining of the course: Organize data supporting basic operators: insert, delete, look up.

www.ctu.edu.vn 57

57

CANTHO UNIVERSITY

Q&A

www.ctu.edu.vn

58