Chương 4: Cây

Ths. Phạm Thanh An
Bộ môn Khoa học máy tính - Khoa CNTT
Trường Đại học Ngân hàng TP.HCM

LOGO

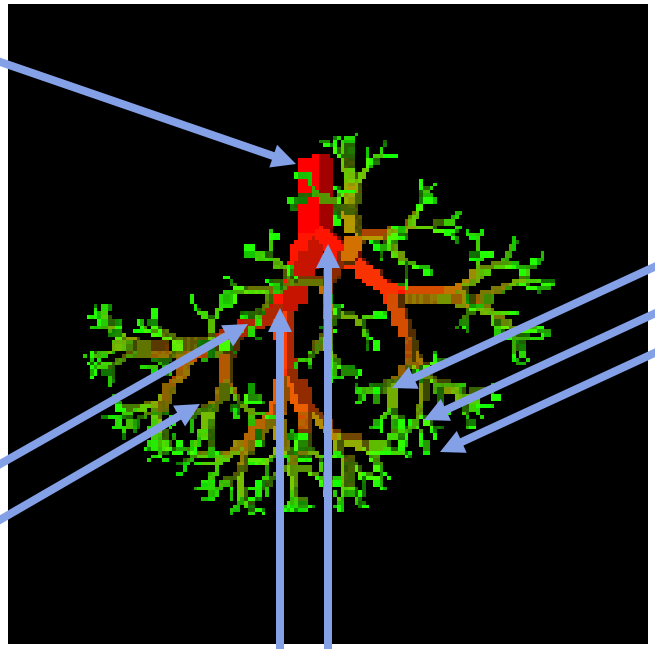
Mục tiêu

- Trang bị cho sinh viên các khái niệm và ứng dụng cây
- Cài đặt và thực hiện các phép toán trên cây, đặc biệt là các phép toán trên cây nhị phân nhị phân tìm kiếm.

Nội dung

- Định nghĩa và các khái niệm
- Cây nhị phân
- Cây nhị phân tìm kiếm (BST)
- Cây tổng quát

Cây (trong máy tính)



Khái niệm về cây (tree)

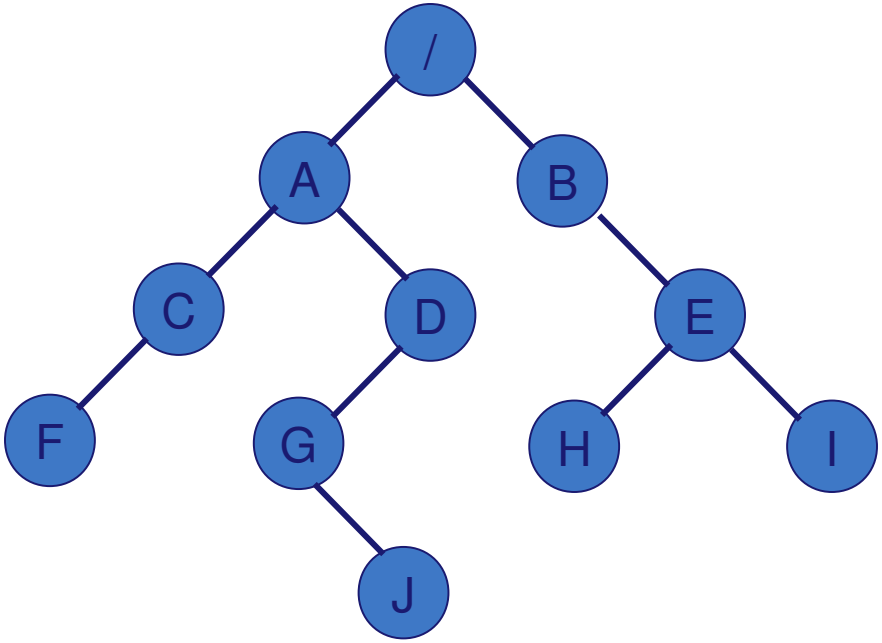
- Là tập hữu hạn các nút (tree node), sao cho
  - Có một nút gọi là nút gốc (root)
  - Các nút còn lại được phân hoạch thành n tập riêng biệt T1, T2 , ... , Tn, mỗi tập Ti là một cây
  - Giữa các nút có quan hệ phân cấp (hierarchical relationship) gọi là “quan hệ cha con”
- Cây không có nút gọi là cây rỗng (null tree)

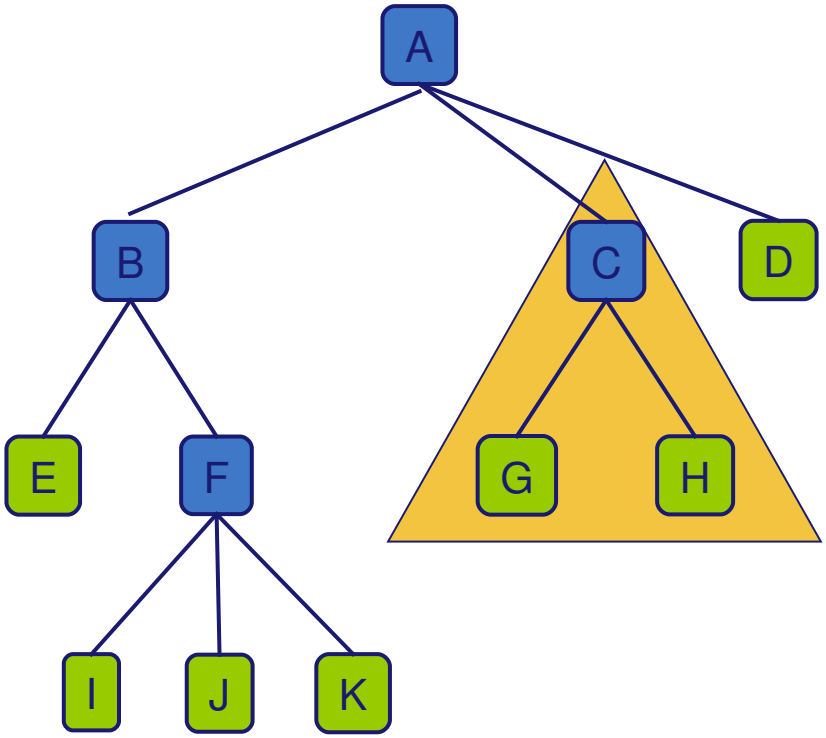
Biểu diễn cây

- Bằng đồ thị
- Bằng giản đồ
- Bằng danh sách (các dấu ngoặc lồng nhau)
- Bằng phương pháp Indentatio

Biểu diễn cây

- Bằng đồ thị

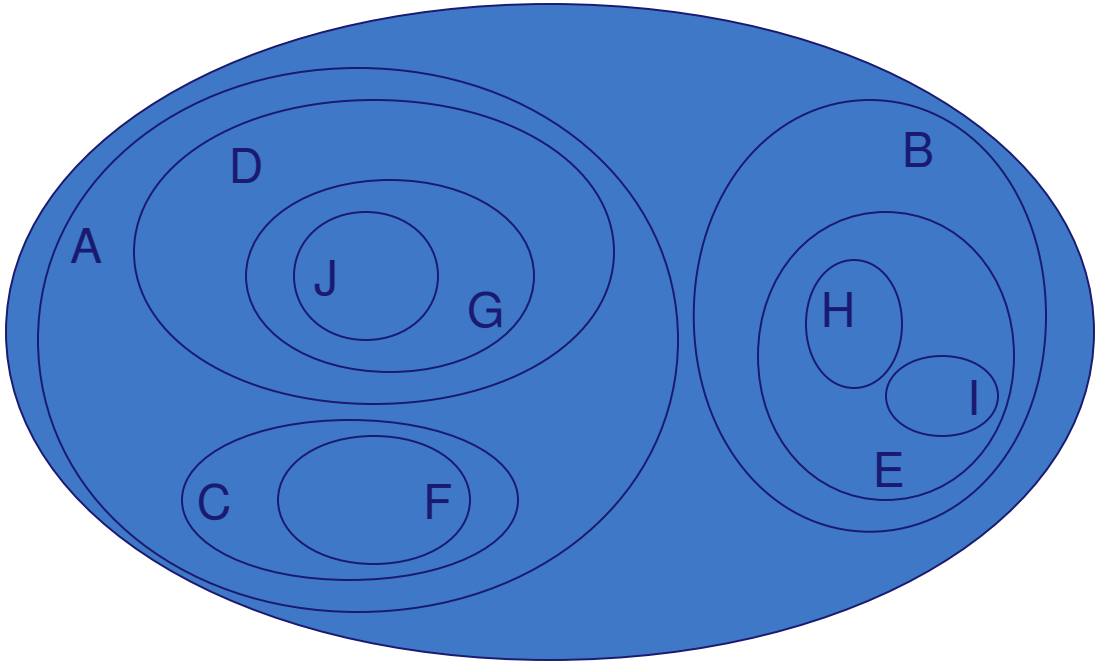




Cây con

Biểu diễn cây

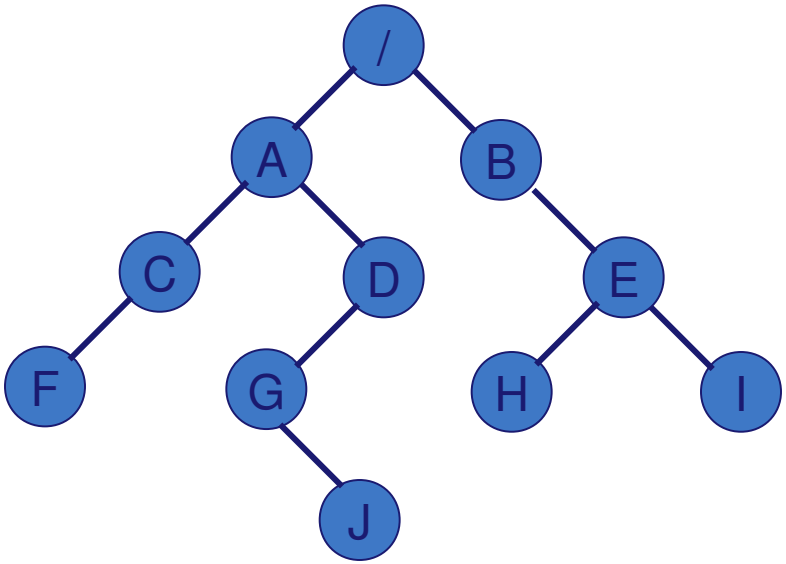
- Bằng giản đồ

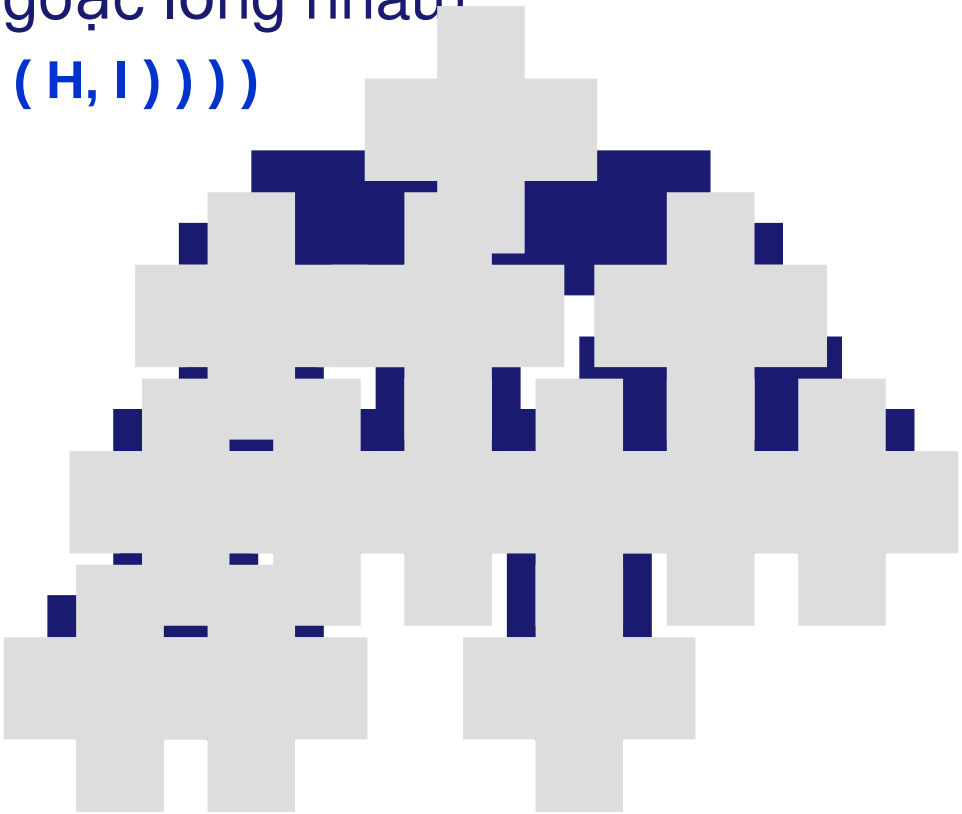


Biểu diễn cây

- Bằng danh sách (các dấu ngoặc lồng nhau)

  (/(A (C (F), D (G (J)))), (B (E (H, I))))



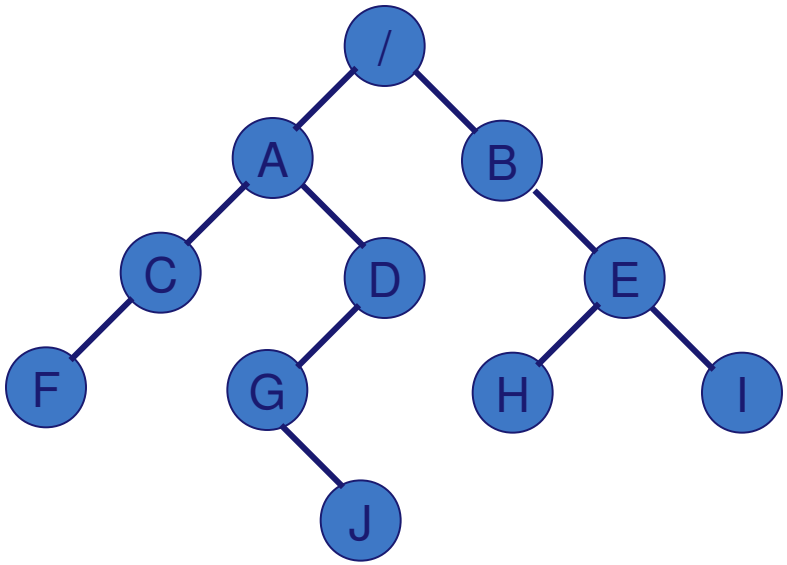


(A (B (E (K, L), F), C (G), D (H (M), I, J)))

Biểu diễn cây

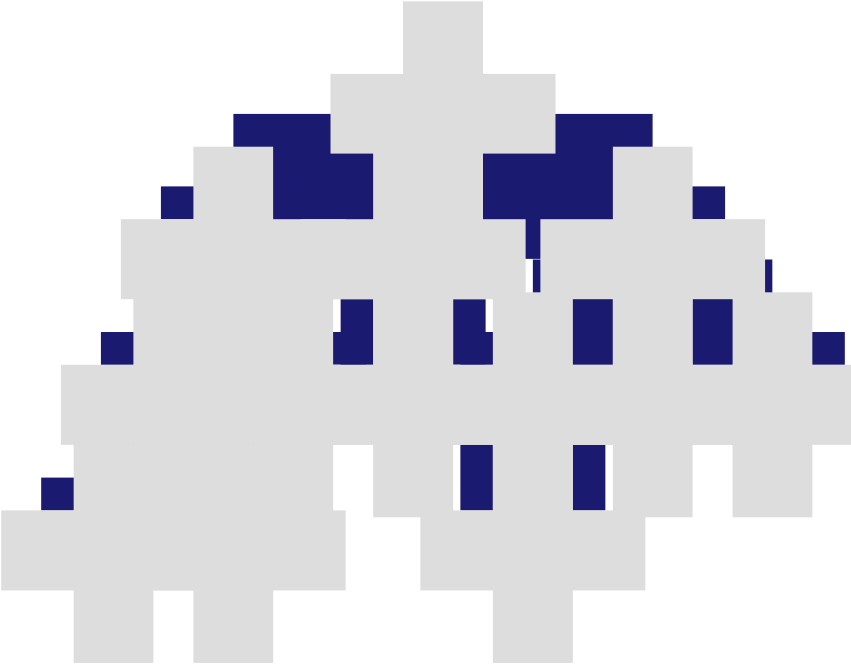
- Bằng phương pháp Indentatio

```
A
    C
        F
    D
        G
            J
B
    E
        H
        I
```



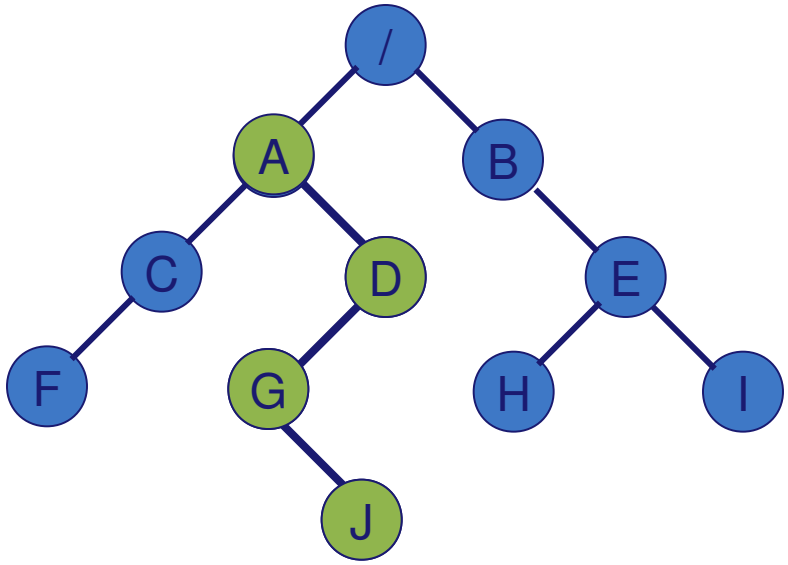
Các thuật ngữ

- Bậc của nút và bậc của cây
  - Nút A: bậc 3, nút C bậc 1
  - Bậc của cây: 3
- Nút gốc, Nút lá và nút nhánh
- Nút cha (Parent), nút con (children)



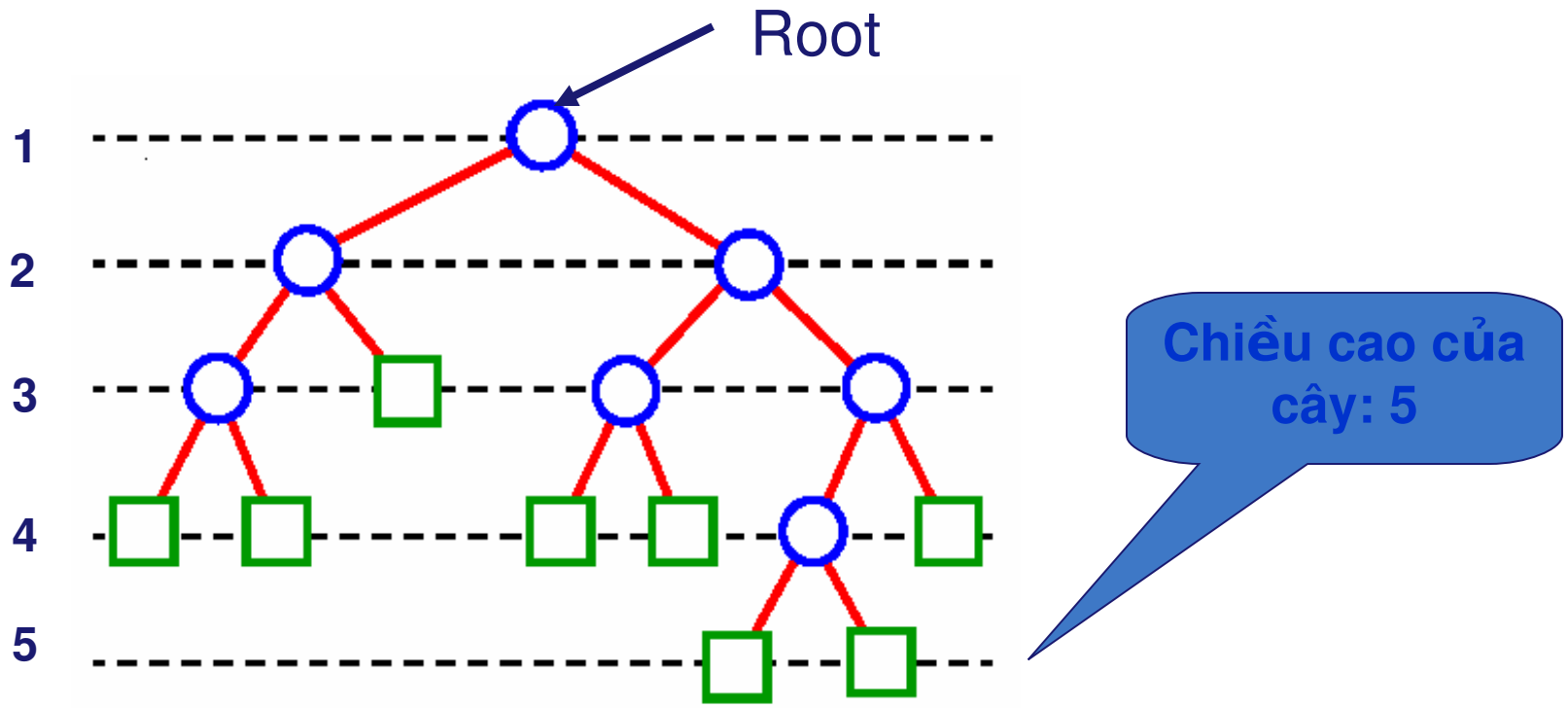
Các thuật ngữ

- Đường đi (path)



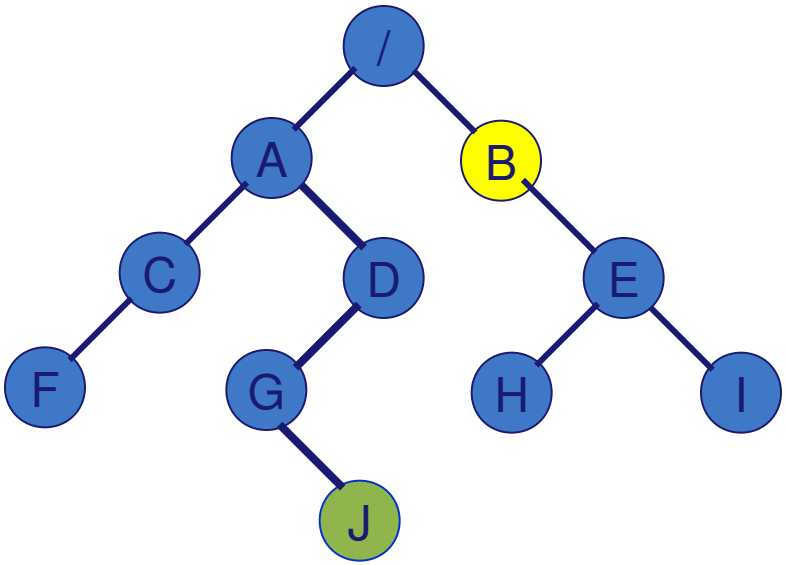
Các thuật ngữ

- Mức của nút và chiều cao của cây



Các thuật ngữ

- Tổ tiên (ancestors) của một nút
  - Tổ tiên của nút J
- Con cháu (Descendant) của một nút:
  - Con cháu của B
- Các con của cùng một cha gọi là anh em ruột (siblings)

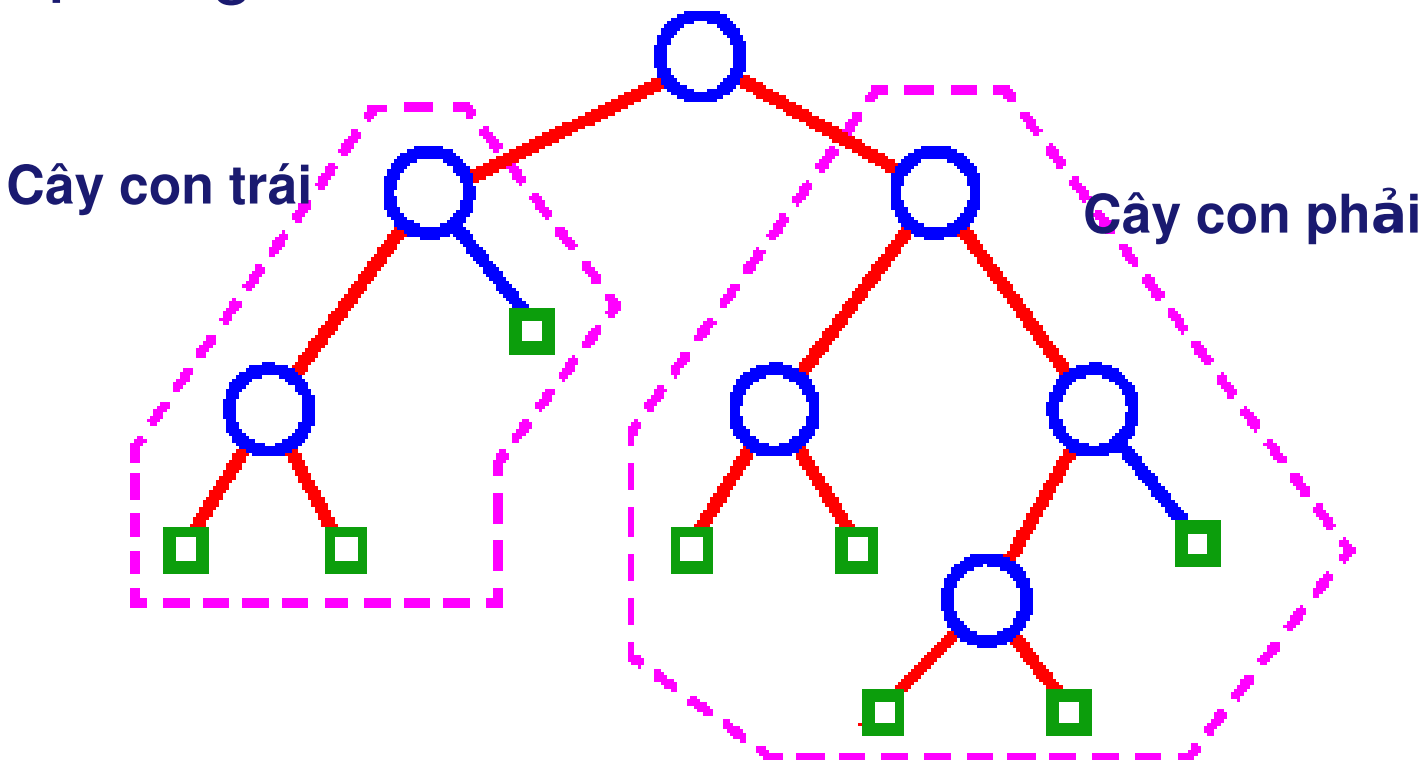


Cây có thứ tự và Rừng

- Cây có thứ tự (ordered tree)
  - Một cây gọi là có thứ tự khi ta thay đổi vị trí của các cây con, ta nhận được một cây mới
- Rừng (forest)
  - Tập hợp hữu hạn các cây phân biệt
  - Nếu bỏ đi nút gốc của một cây, ta sẽ thu được một rừng gồm nhiều cây phân biệt

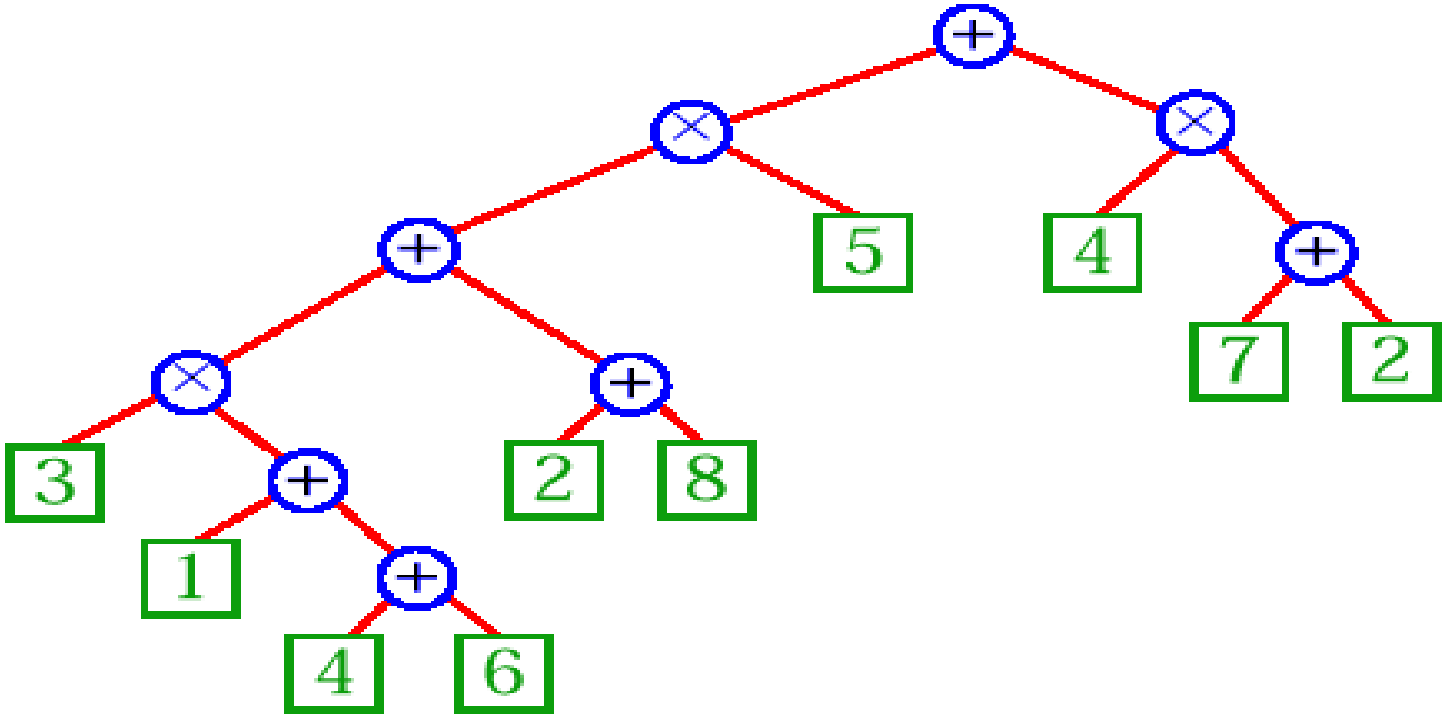
Cây nhị phân

- Định nghĩa



Cây nhị phân

- Cây nhị phân biểu diễn biểu thức toán học

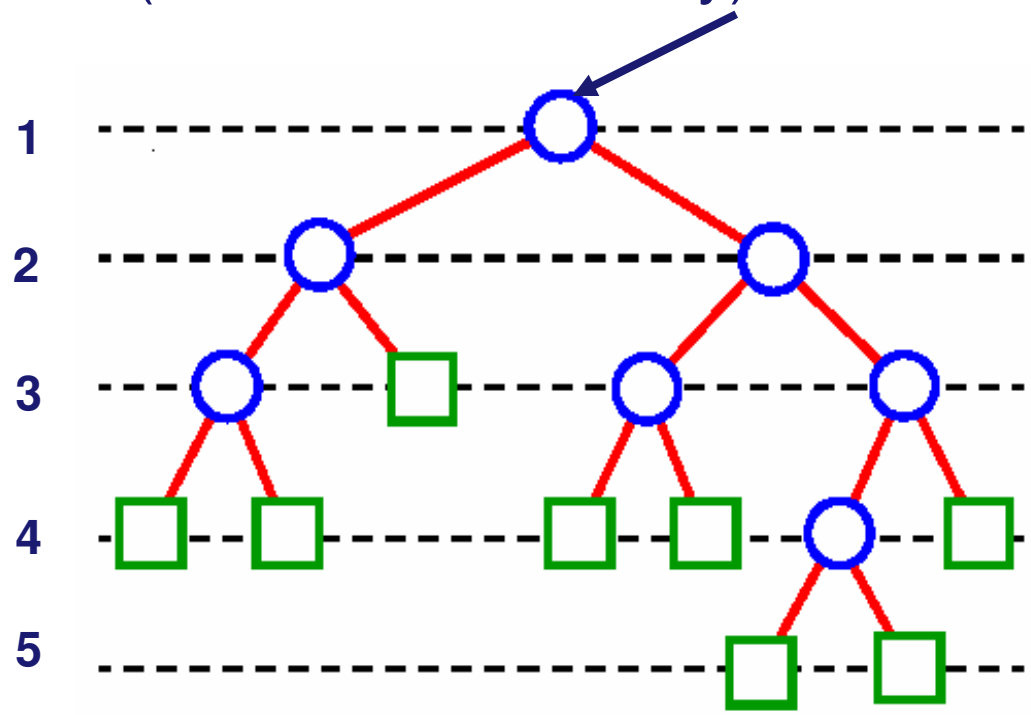


$$((((3 \times (1 + (4 + 6))) + (2 + 8)) \times 5) + (4 \times (7 + 2)))$$

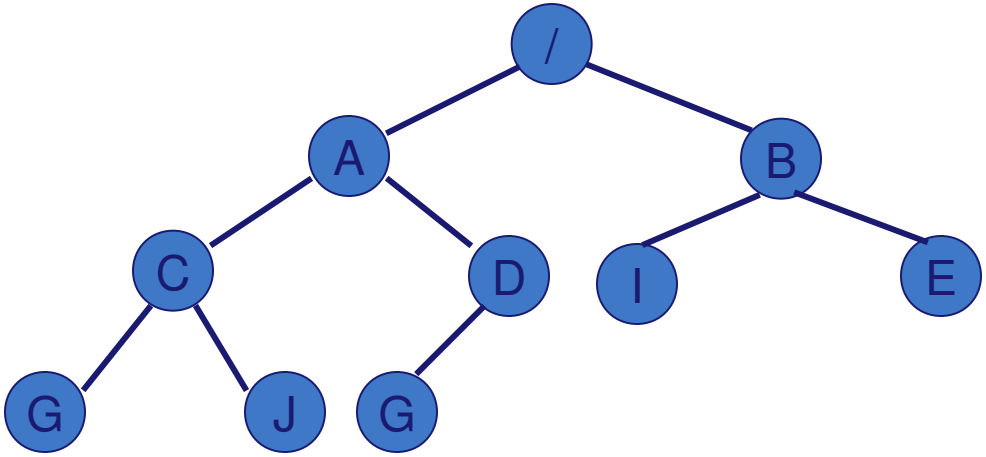
Tính chất của cây nhị phân

- Số nút tối đa mức i trong cây 2^{i-1}
- Số nút tối đa trong cây là 2^h-1 (h chiều cao của cây)

- Chiều cao của cây h $\log_2 N$ *(N là số nút trong cây).*

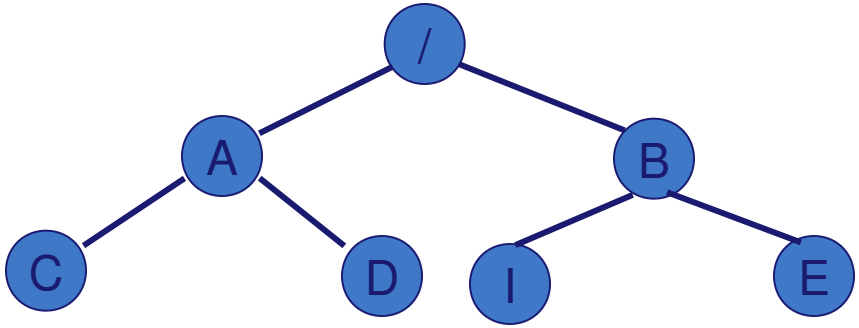


Cây nhị phân hoàn chỉnh



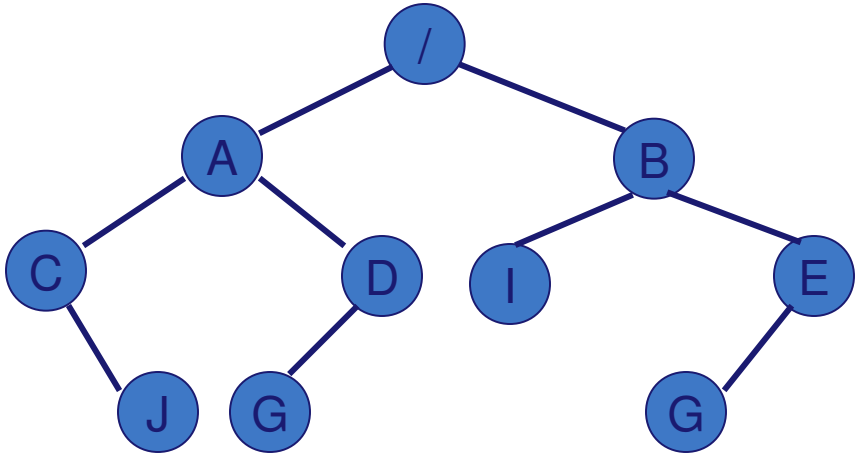
Các nút ứng với các mức trừ mức cuối đều đạt tối đa,
ở mức cuối, các nút đều đạt về phía trái

Cây nhị phân đầy đủ



Các nút đạt tối đa ở cả mọi mức

Cây nhị phân gần đầy



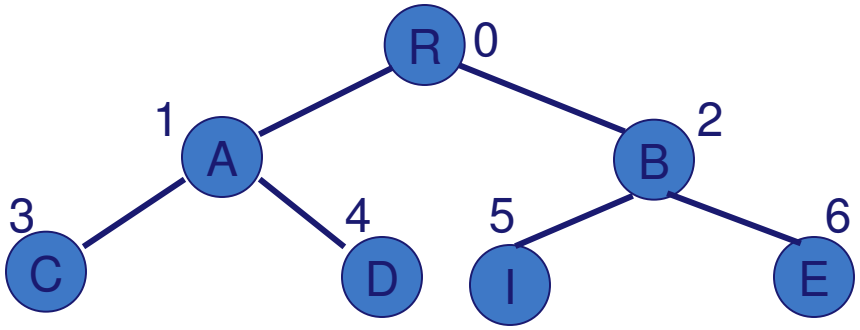
Các nút ứng với các mức trừ mức cuối đều đạt tối đa,
ở mức cuối, các nút không dạt đều về phía trái

Tổ chức lưu trữ cây nhị phân

- Sử dụng mảng một chiều (lưu trữ kế tiếp)
  - Đánh số thứ tự từ gốc, tại mỗi mức, đánh số các nút từ trái sang phải, từ mức thấp đến mức cao
- Sử dụng liên kết(Lưu trữ liên kết)
  - Quản lý cây thông qua nút gốc (root)
  - Mỗi nút cấp phát động, bao gồm dữ liệu và hai liên kết pLeft, pRight, liên kết tới cây con trái và cây con phải
  - Nút lá có hai liên kết trái phải đều rỗng

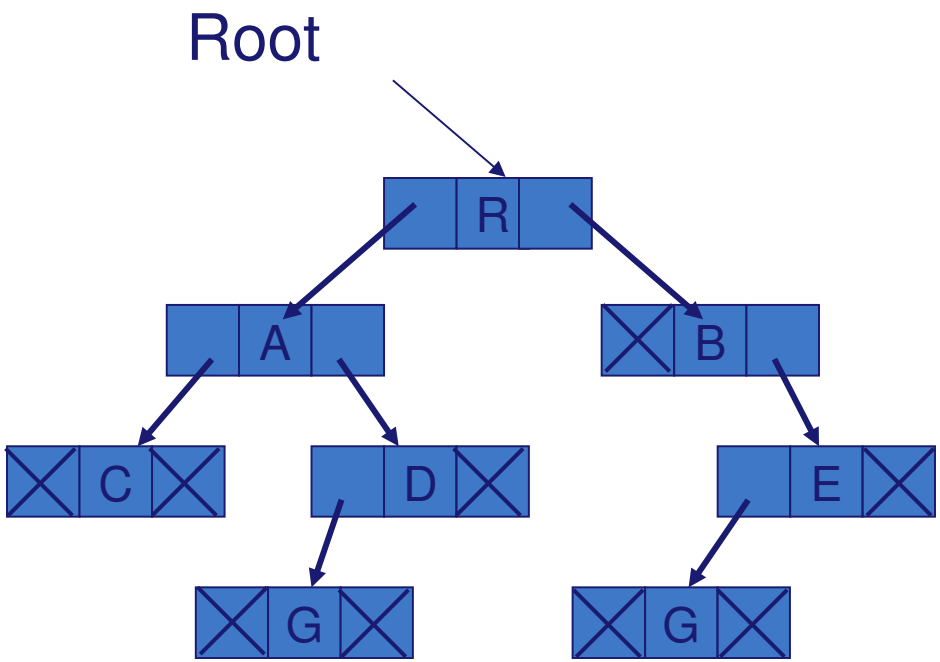
Lưu trữ kế tiếp cây nhị phân

- Con của nút thứ *i* là nút thứ *2i*+ 1và *2i+2*
- Cha của nút thứ *j* là nút *[(j-1)/2]*



R	A	B	C	D	I	E
V[0]	V[1]	V[2]	V[3]	V[4]	V[5]	V[6]

Sử dụng Liên kết

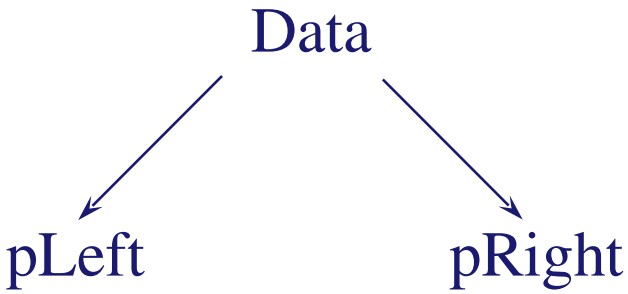


Sử dụng Liên kết

- Cấu tạo của nút
  - Tạo lập bằng cách cấp phát bộ nhớ động
  - Mỗi nút gồm có các thông tin:
    - Dữ liệu (data)
    - 2 liên kết pLeft, pRight liên kết đến nút con trái và nút con phải

Cấu trúc của nút

```
Class Node {
   int Data;
   Node pLeft; // liên kết đến nút con trái
   Node pRight; // liên kết đến nút con phải
};
Node root  = NULL; //gốc của cây
```

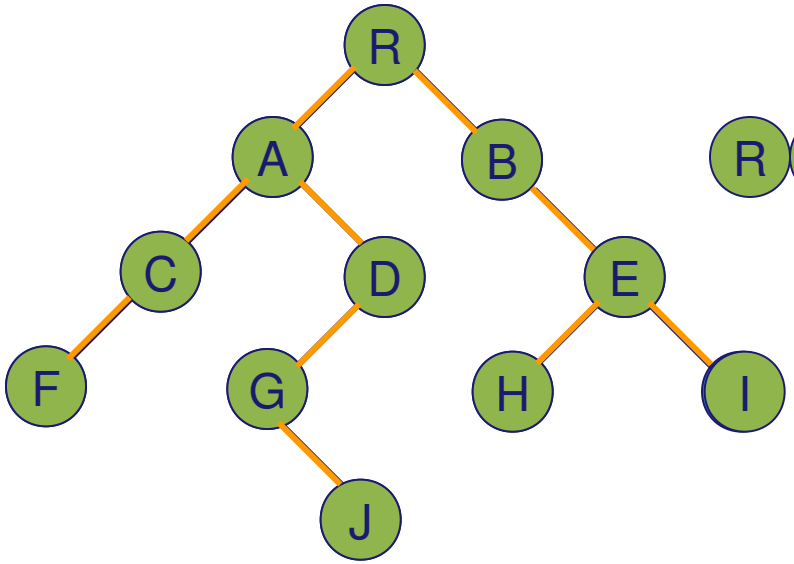


Phép duyệt cây nhị phân

- Định nghĩa
  - là phép xử lý các nút trên cây, mỗi nút một lần
- Duyệt cây theo thứ tự trước (preorder)
- Duyệt cây theo thứ tự giữa (inorder)
- Duyệt cây theo thứ tự sau (postorder)

Duyệt cây theo thứ tự trước

- Duyệt cây theo thứ tự trước (NLR)- Đệ qui
  - Thăm gốc
  - Duyệt cây con trái theo thứ tự trước
  - Duyệt cây con phải theo thứ tự trước

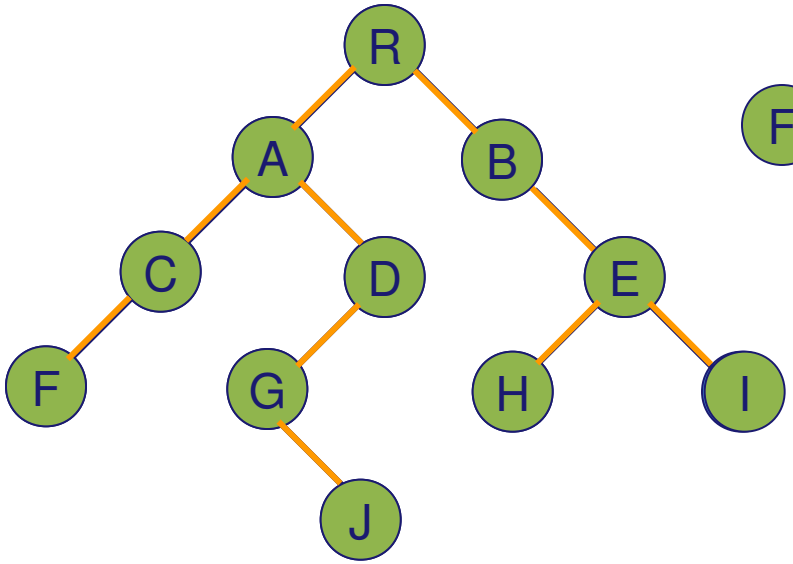


Duyệt theo thứ tự trước

```
void preorder(Node root)
{
    if (root != NULL) {
        - In ra : root.data;
        preorder(root.pLeft);
        predorder(root.pRight);
    }
}
```

Duyệt cây theo thứ tự giữa

- Duyệt cây theo thứ tự giữa (LNR)
  - Duyệt cây con trái theo thứ tự giữa
  - Thăm gốc
  - Duyệt cây con phải theo thứ tự giữa

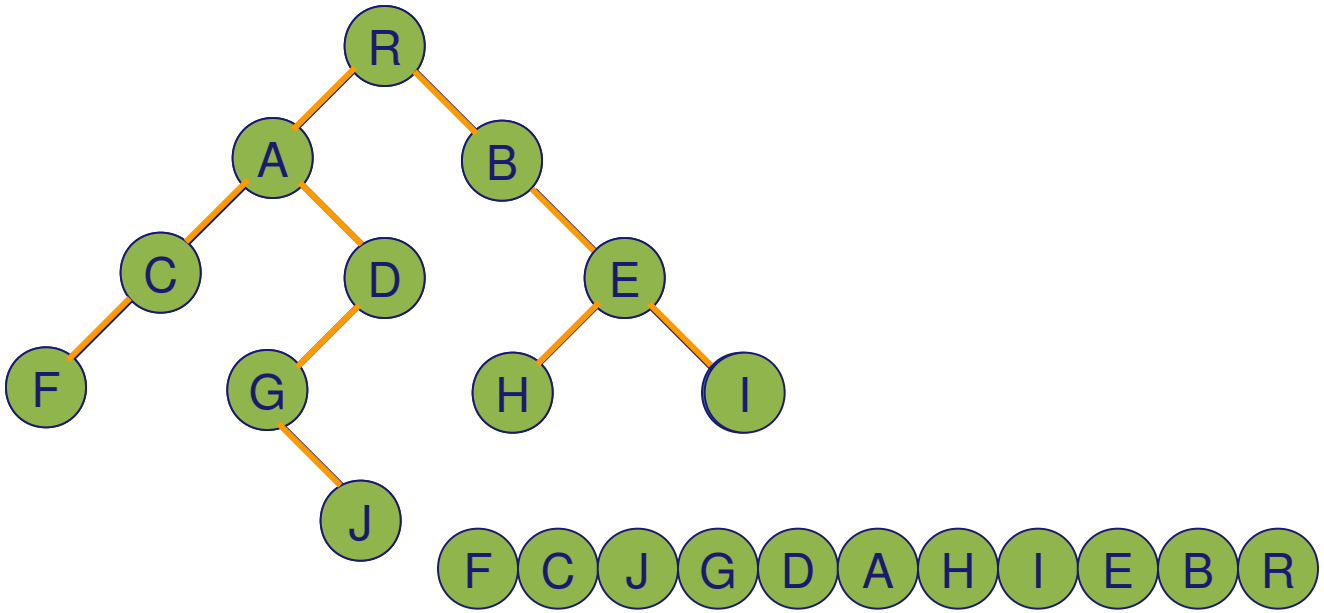


Duyệt cây theo thứ tự giữa

```
void inorder(Node root)
{
    if (root !=  NULL) {
        inorder(root.pLeft);
         In ra:  root.data;
        indorder(root.pRight);
    }
}
```

Duyệt cây theo thứ tự sau

- Duyệt cây theo thứ tự sau (LRN)
  - Duyệt cây con trái theo thứ tự sau
  - Duyệt cây con phải theo thứ tự sau
  - Thăm gốc

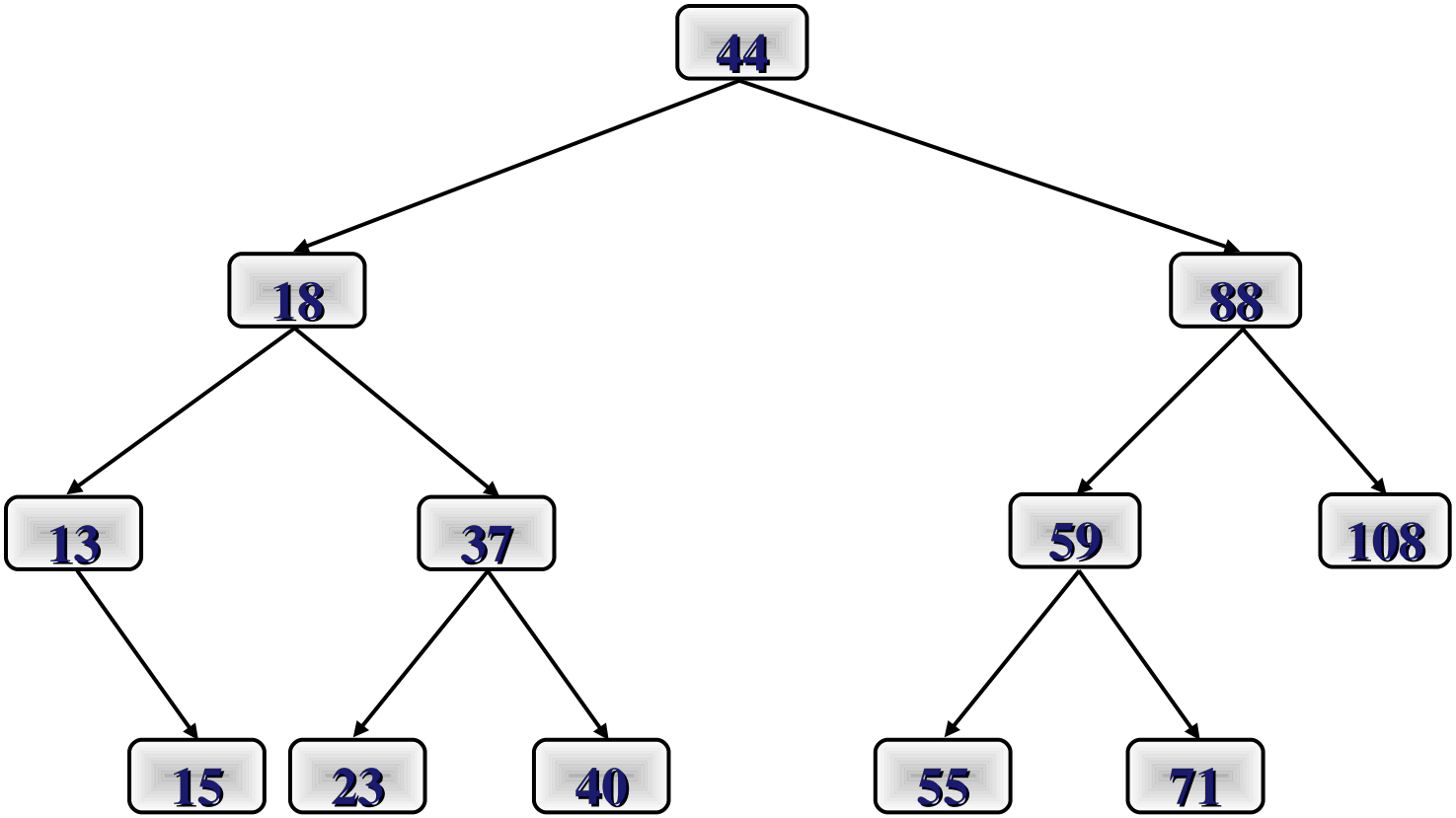


Duyệt cây theo thứ tự sau

```
void postorder(Node root)
{
    if (root !=null) {
        postorder(root.pleft);
        postdorder(root.pright);
        In ra: root.data;
    }
}
```

Cây nhị phân tìm kiếm

- Định nghĩa: (Binary Search Tree – BST)



Cây nhị phân tìm kiếm

- Khai báo cây

```
Class BSTNode {
  int Data;
   BSTNode pLeft; //con trỏ đến nút con trái
   BSTNode pRight; //con trỏ đến nút con phải
};
BSTNode root = NULL; //gốc của cây
```

Cây nhị phân tìm kiếm

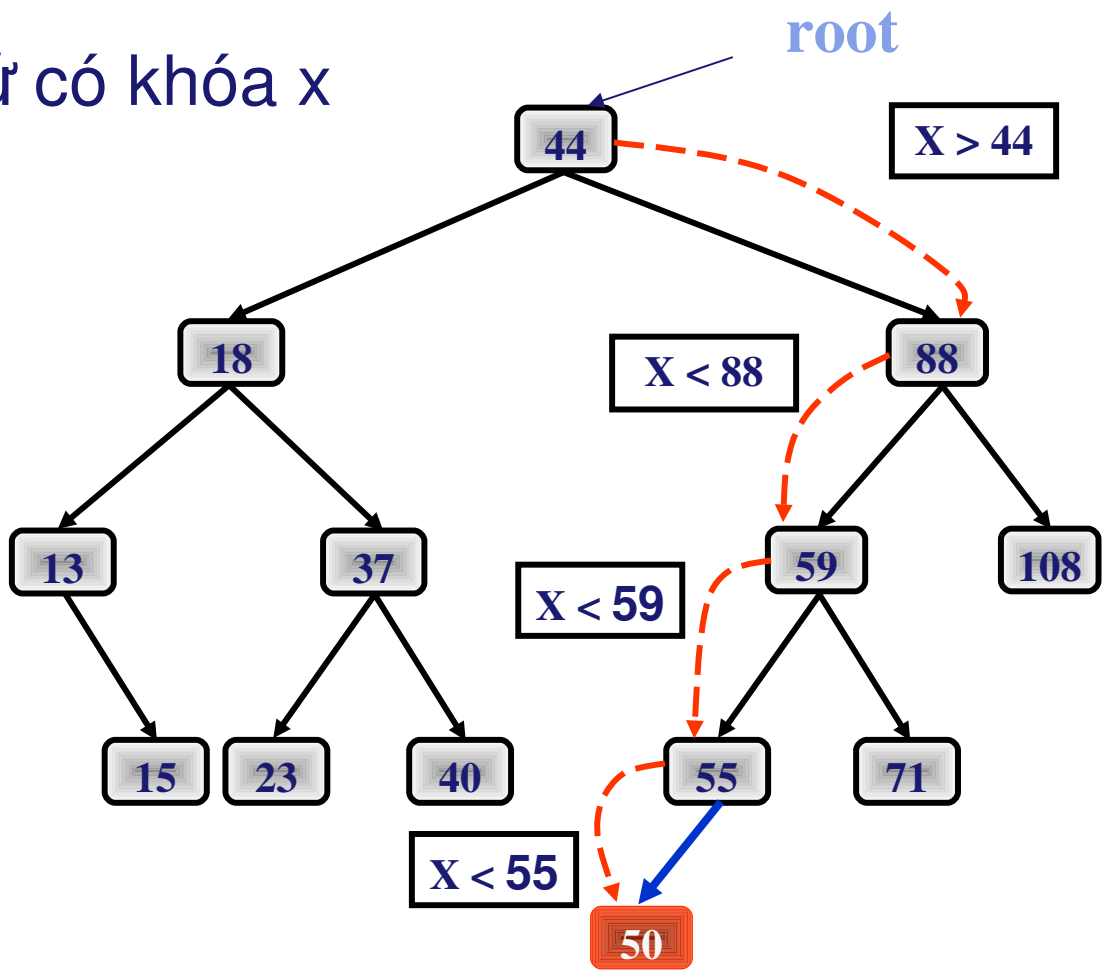
- Các thao tác trên cây BST
  - Tìm nút có khóa x
  - Xóa một nút có khóa là x
  - Tìm nút có khóa nhỏ nhất
  - Tìm nút có khóa lớn nhất
  - Giải phóng cây

Cây nhị phân tìm kiếm

- int Insert(int X, BSTNode root);
- int Delete(int X, BSTNode root);
- BST_Node Find(int X, BSTNode root);
- BST_Node FindMin(BSTNode root);
- BST_Node FindMax(BSTNode root);
- void MakeEmpty(BSTNode root);

Thêm một phần tử vào cây nhị phân tìm kiếm

- Thêm vào phần tử có khóa x



Thêm một phần tử vào cây nhị phân tìm kiếm

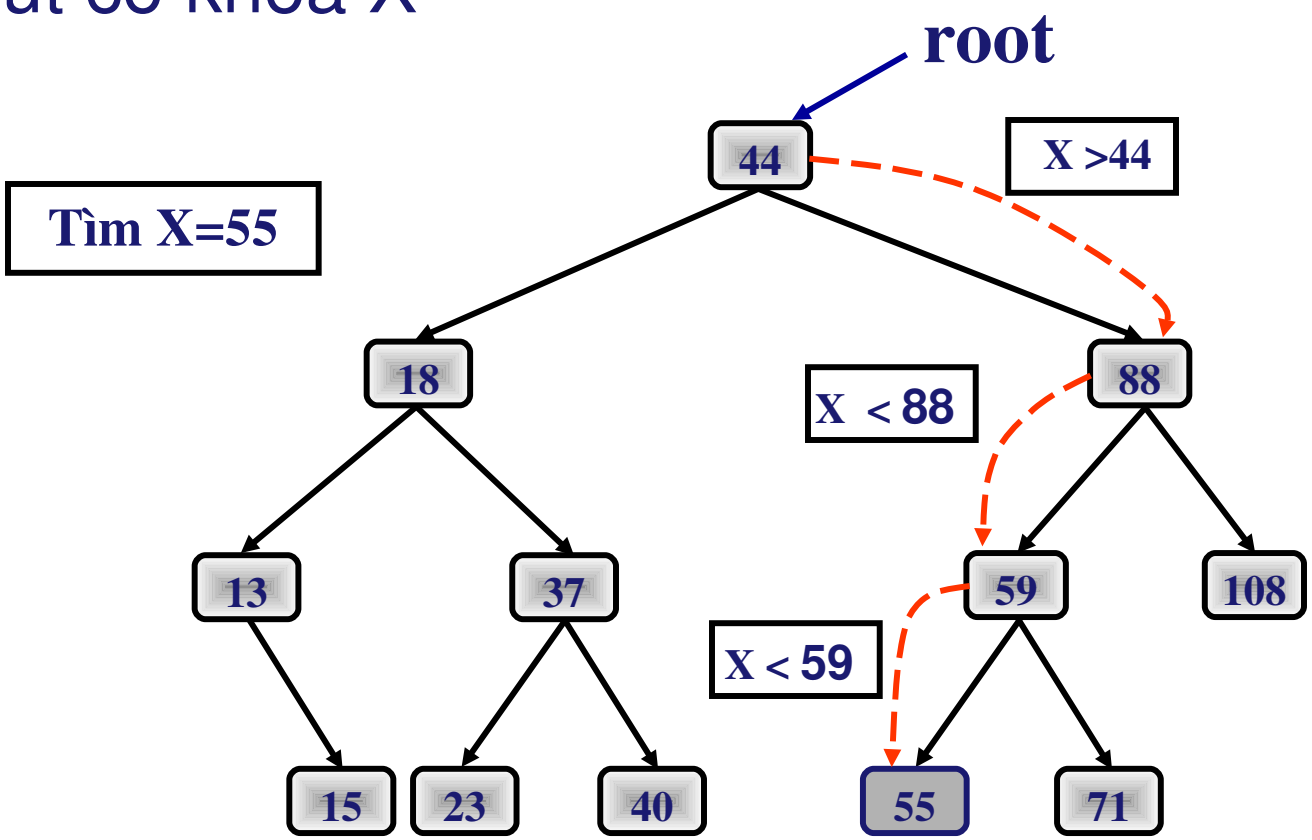
```
int  Insert(int X, BST_Node root)
{  if  (root == NULL)
    {       root =new BSTreeNode ;
          if  ( root == NULL )
                return -1; // Không thể cấp phát bộ nhớ
          else
           {
              root.Data = X;
                root.pLeft = root.pRight = NULL;
                return 1; // Thêm vào thành công
           }
      }
```

Thêm một phần tử vào cây nhị phân tìm kiếm

```
else
  if (root.Data ==X)
                    return 0 ; // Đã tồn tại trong cây
  else
        if ( X< root.Data )
         return Insert( X, root.pLeft );
     else
       return Insert( X, root.pRight );
}
```

Tìm một nút có khóa X

- Tìm nút có khóa X



Tìm một nút có khóa X

```
BSTNode Find( int X, BSTNode root)
  {         if( root == NULL )
              return NULL;
          if ( X < root.data)
              return Find( X, root.pLeft );
          else if ( X > root.data)
              return Find( X, root.pRight );
          else
              return root;
   }
```

Tìm một nút có khóa X

- Tìm nút có khóa X, không dùng đệ qui

```
BTSNode Find2(int X, BTSNode root)
{
    BTSNode p = root;
    while (p != NULL)
    {
        if(X == p.data)   return p;
          else
                if(x < p.data) p = p.pLeft;
                else
            p = p.pRight;
     }
    return NULL;
}
```

Tìm nút có khóa nhỏ nhất

```
BSTNode FindMin( BSTNode root)
 {        if ( root == NULL )
              return NULL;
          else if( root.pLeft == NULL )
              return root;
          else
              return FindMin( root.pLeft );
 }
```

Tìm nút có khóa lớn nhất

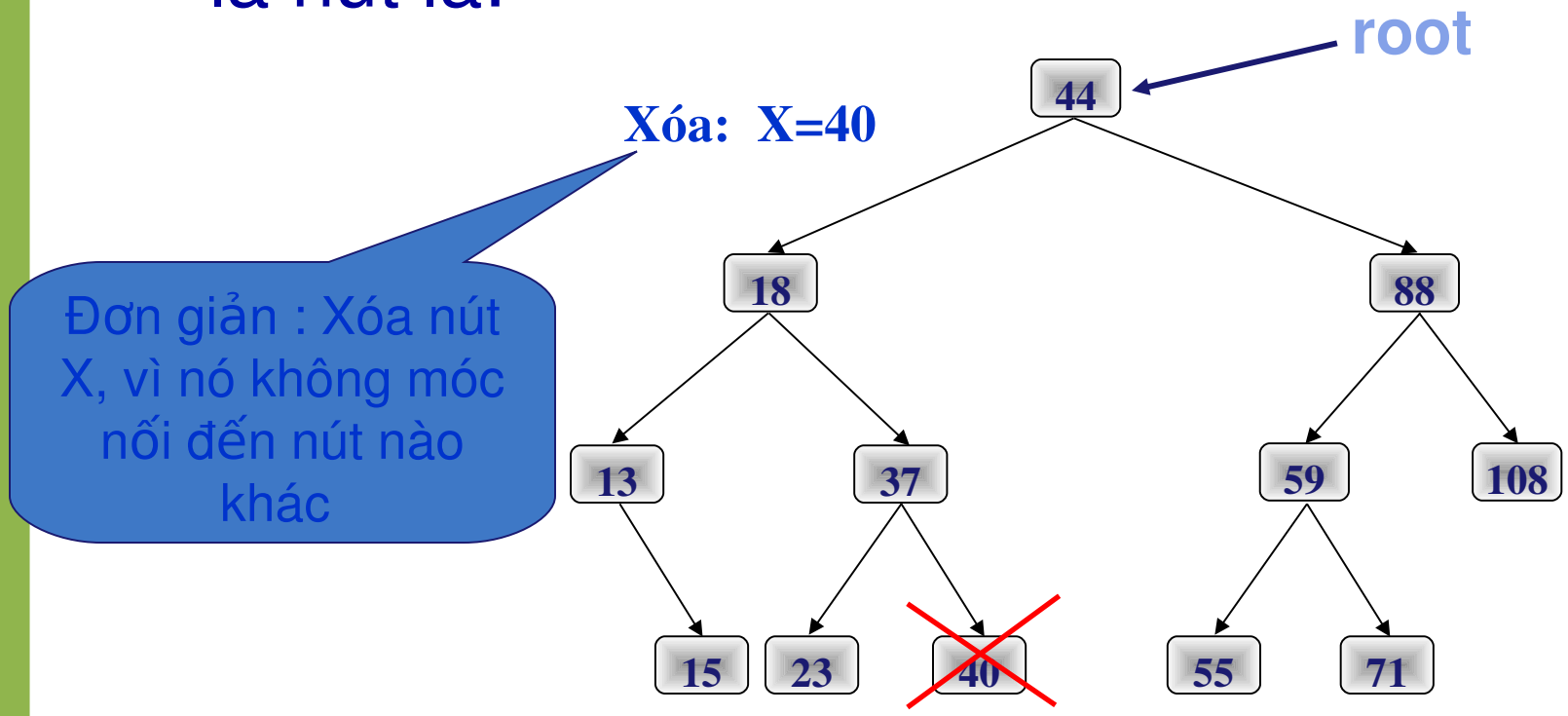
```
BST_Node FindMax(BSTNode root)
 {
        if (root != NULL )
              while (root.pRight != NULL )
                  root = root.pRight;
         return root;
  }
```

Xóa một nút có khóa X trên cây BST

- Xóa một nút có khóa X trên cây BST, có ba trường hợp:
  - Nút có khóa X là nút lá.
  - Nút có khóa X chỉ có 1 con (trái hoặc phải).
  - Nút có khóa X có đủ cả 2 con

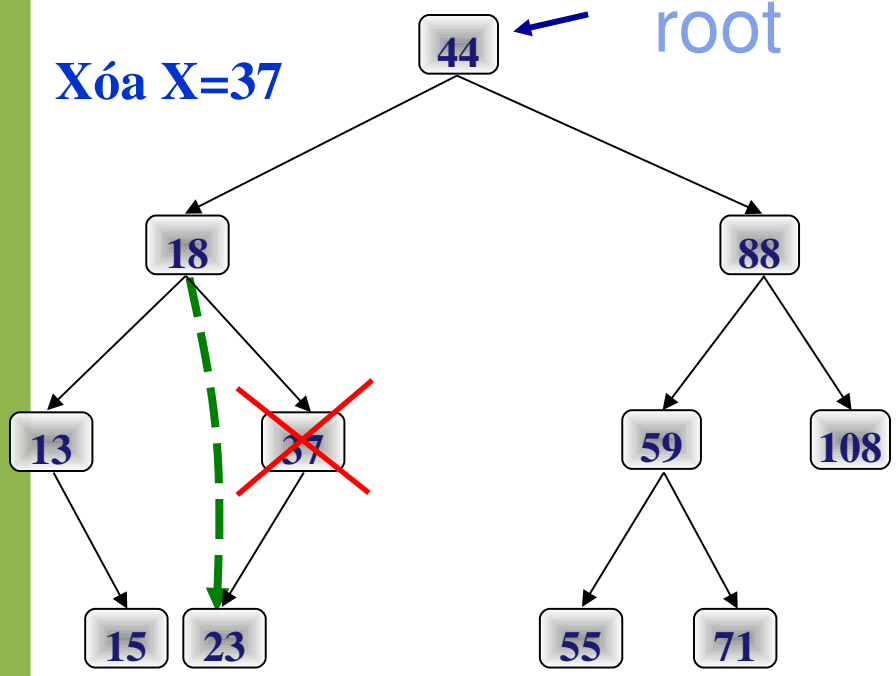
Xóa một nút có khóa X

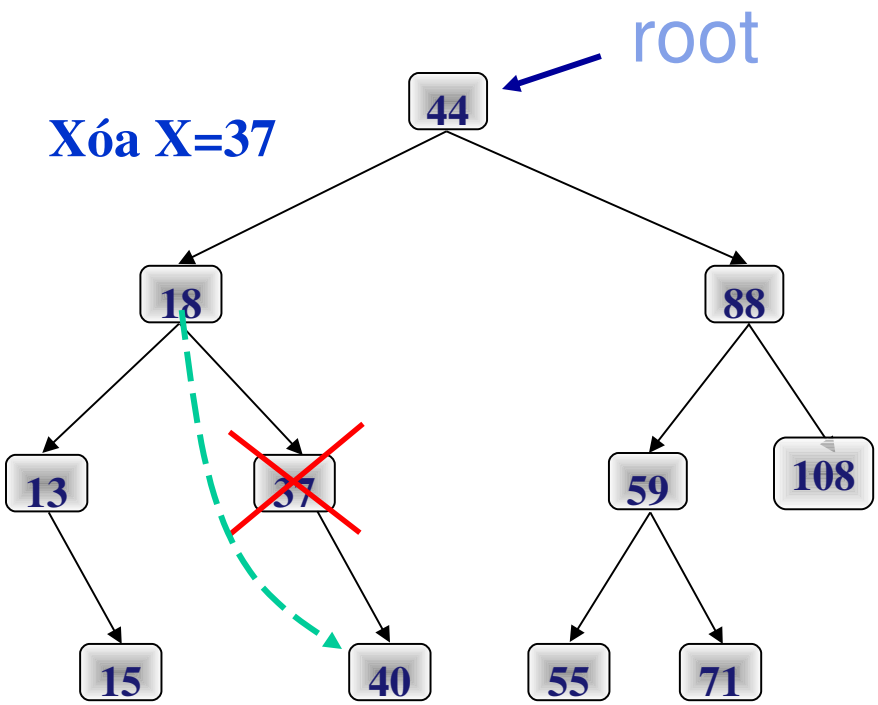
- Trường hợp 1 :Nút có khóa X trên cây BST là nút lá:



Xóa một nút có khóa X trên cây BST

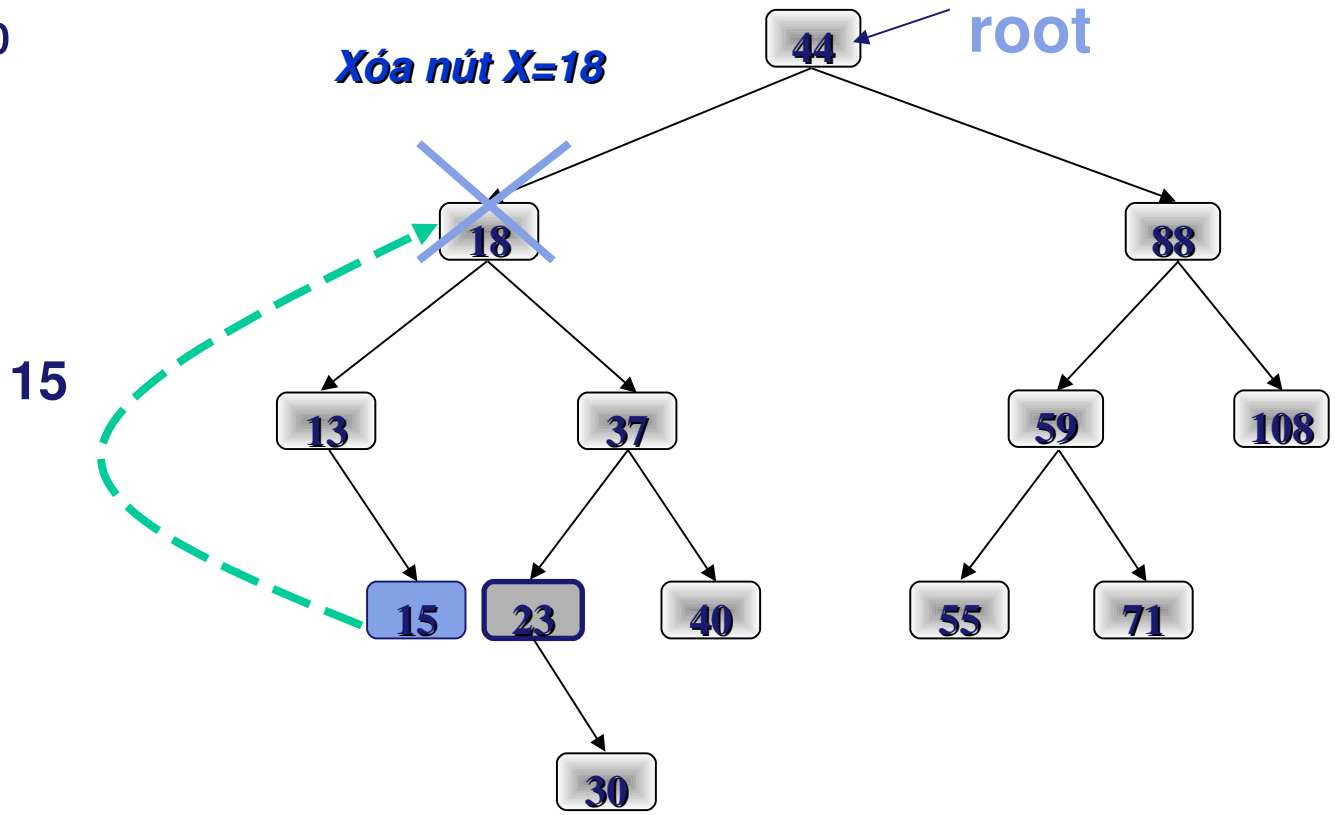
- Trường hợp 2 : Nút X có một cây con trái hoặc phải





Xóa một nút có khóa X trên cây BST

- Trường hợp 3 : Nút X có một cây hai con trái và phải
  - Hủy gián tiếp



Cây nhị phân tìm kiếm (Binary Search Tree – BST)

```
int Delete( int X, BSTNode root)
 {  BSTNode p;
    if ( root == NULL )
       return 0 ; // cây rỗng, không tim thấy
   else
       if ( X < root.Data)  // xóa trên cây con trái
          return Delete( X, root.pLeft );
        else
         if ( X > root.Data ) // xóa trên cây con phải
           retrurn Delete( X, root.pRight );
        else  // tìm ra nút cần xóa
```

Cây nhị phân tìm kiếm (Binary Search Tree – BST)

```
if ( root.pLeft && root.pRight )  // Có hai con
  {      p = FindMax(root.pLeft); //tìm nút có khóa lớn nhất trên con trái
         root.Data = p.Data;
         return Delete(root.Data, root.pLeft);
  }
  else  // có một con hoặc không có con
  {      p = root;
         if ( root.pLeft == NULL ) // xử lý như không có con
            root = root.pRight;
         else if ( root.pRight == NULL )
              root = root.pLeft;
         p = null;
   }
   return 1;
}
```

Giải phóng cây BST

```
void MakeEmpty( BST_Node root);
{
   if (root)
    {
        MakeEmpty(root.pLeft);
            MakeEmpty(root.pRight);
            delete root ;
    }
 }
```

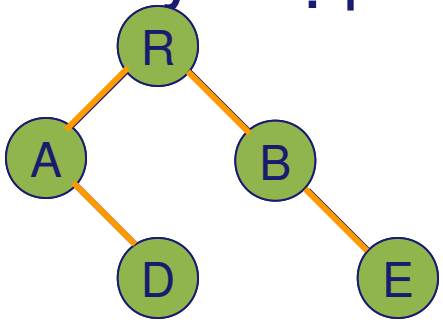
Cây nhị phân liên kết vòng

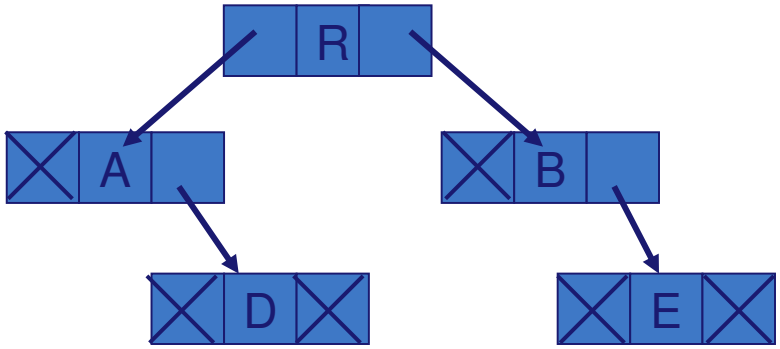
- **Định nghĩa**
  - Sử dụng liên kết NULL để lưu trữ liên kết tới nút kế tiếp trong phép duyệt cây nhị phân -> phép duyệt được thực hiện dễ dàng
  - Sử dụng giá trị kiểm tra liên kết thật (đến nút trong cây) hay liên kết giả (nút trong phép duyệt)
  - Ltype = true, nếu liên kết trái là liên kết thật
  - Rtype = true, nếu liên kết phải là liên kết thật

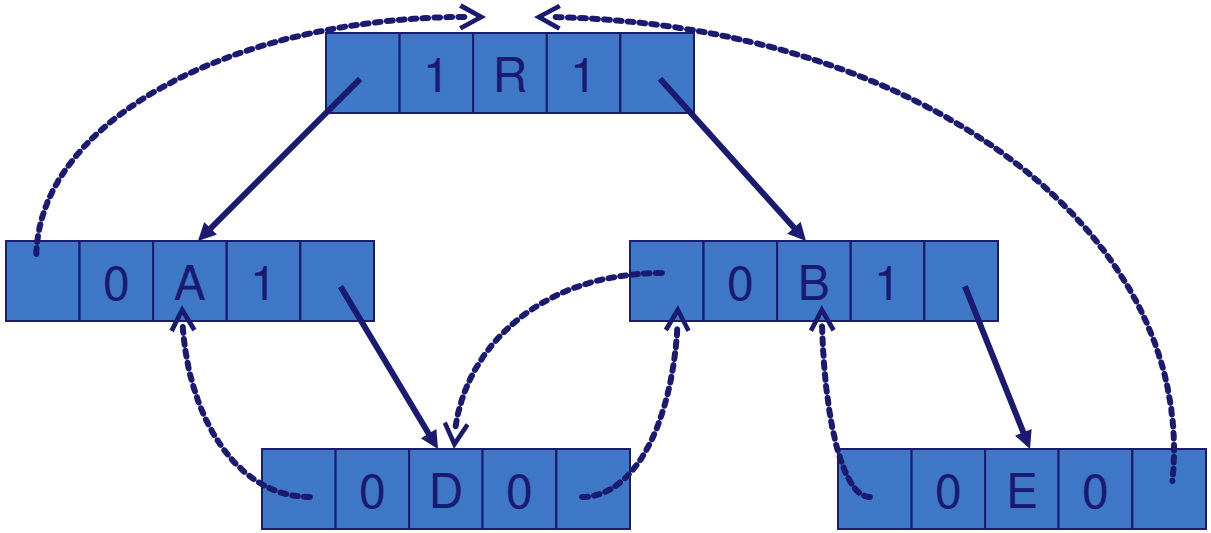
LPTR	LTYPE	INFO	RTYPE	RPTR

Cây nhị phân liên kết vòng

- Cây nhị phân liên kết vòng (NLR)







Cây tổng quát

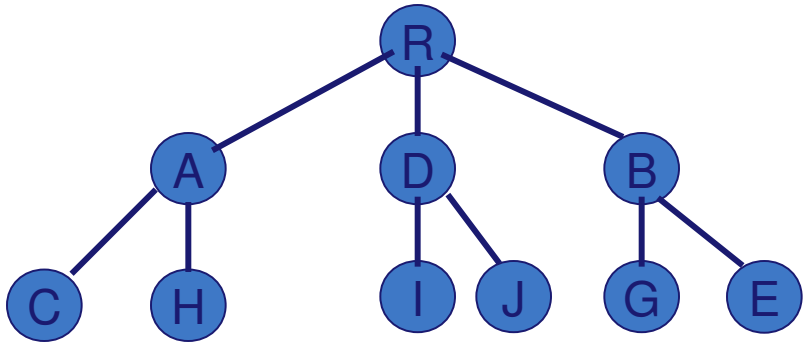
- Định nghĩa
  - Cây m phân là cây mà mỗi nút có tối đa m nút con (cây con)
  - Biểu diễn cây m phân bằng liên kết động
    - Mỗi nút có m+1 trường, với m mối nối
    - Với cây m phân đầy đủ, có n(m-1)+1 mối liên kết NULL

Cây tổng quát

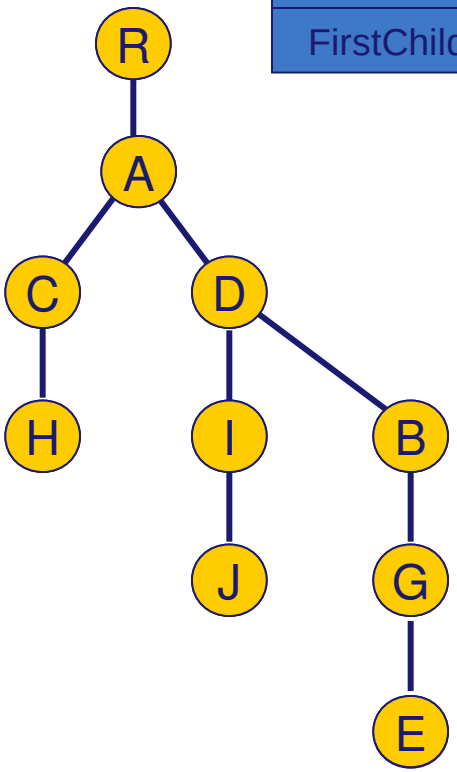
- Biểu diễn cây tổng quát
  - Biểu diễn cây tổng quát bằng cây nhị phân
  - Đối với một nút trên cây tổng quát
    - Một nút con nằm ở vị trí trái nhất (con cả 1)
    - Một nút kế cận với nút đang xét kể từ trái sang (em kế - 2)

Cây tổng quát

- Biểu diễn cây tổng quát



Data	
FirstChild	NextSibling




```
class TreeNode
        {int info;
         TreeNode FirstChild; // con cả
         TreeNode NextSibling; //em kê
        };
```

Cây tổng quát

- Phép duyệt cây tổng quát NLR(T)
  - Nếu T rỗng, dừng
  - Ngược lại, $T_1,\ldots,T_n$ là cây con gốc T
    - Thăm gốc của T
    - NLR(T_1), T_1 cây con thứ nhất của gốc T
    - Duyệt cây con $T_2,\ldots,T_n$ của T theo thứ tự trước

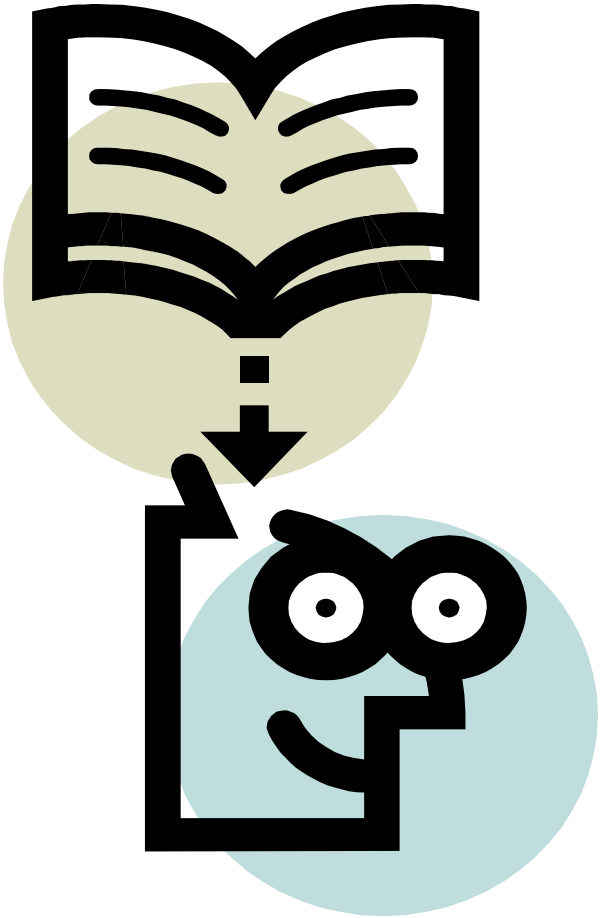
Cây tổng quát

- Phép duyệt cây tổng quát LNR(T)
  - Nếu T rỗng, dừng
  - Ngược lại, $T_1,\ldots,T_n$ là cây con gốc T
    - LNR(T_1), T_1 cây con thứ nhất của gốc T
    - Thăm gốc của T
    - Duyệt cây con $T_2,\ldots,T_n$ của T theo thứ tự giữa

Cây tổng quát

- Phép duyệt cây tổng quát LRN(T)
  - Nếu T rỗng, dừng
  - Ngược lại, $T_1,\ldots,T_n$ là cây con gốc T
    - LNR(T_1), T_1 cây con thứ nhất của gốc T
    - Duyệt cây con $T_2,\ldots,T_n$ của T theo thứ tự giữa
    - Thăm gốc của T

Cây tổng quát

- Duyệt cây theo mức
  - Duyệt cây theo chiều rộng
  - Ý tưởng
    - Tổ chức thành một hàng đợi
    - Đưa nút gốc vào hàng đợi
    - Lặp
      - Lấy một nút ra khỏi hàng đợi
      - Duyệt nút T
      - Đưa các nút con của T (nếu có) vào hàng đợi

Q&A