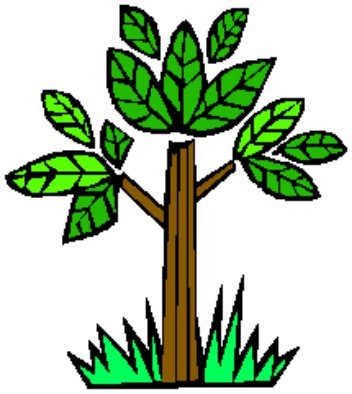
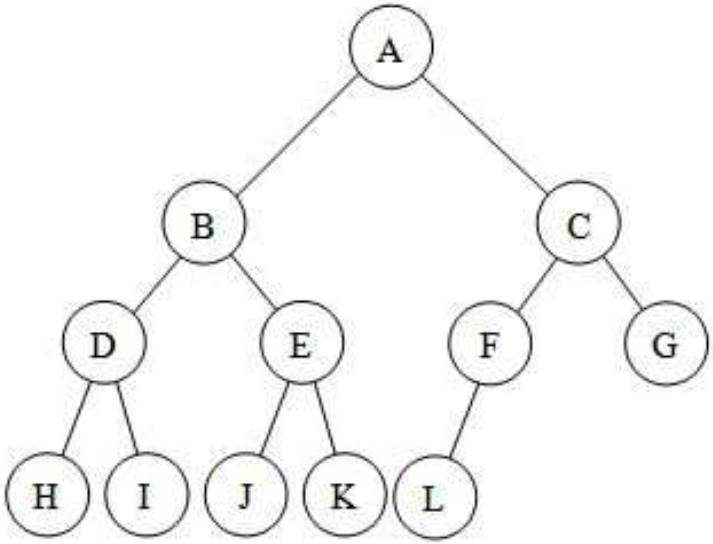
Cây nhị phân

- Các khái niệm và thuật ngữ cơ bản
- Cài đặt cấu trúc dữ liệu
- Duyệt cây
- Cây nhị phân tìm kiếm – Binary Search Tree
- Hàng đợi ưu tiên – Priority Queue



Các khái niệm và thuật ngữ cơ bản

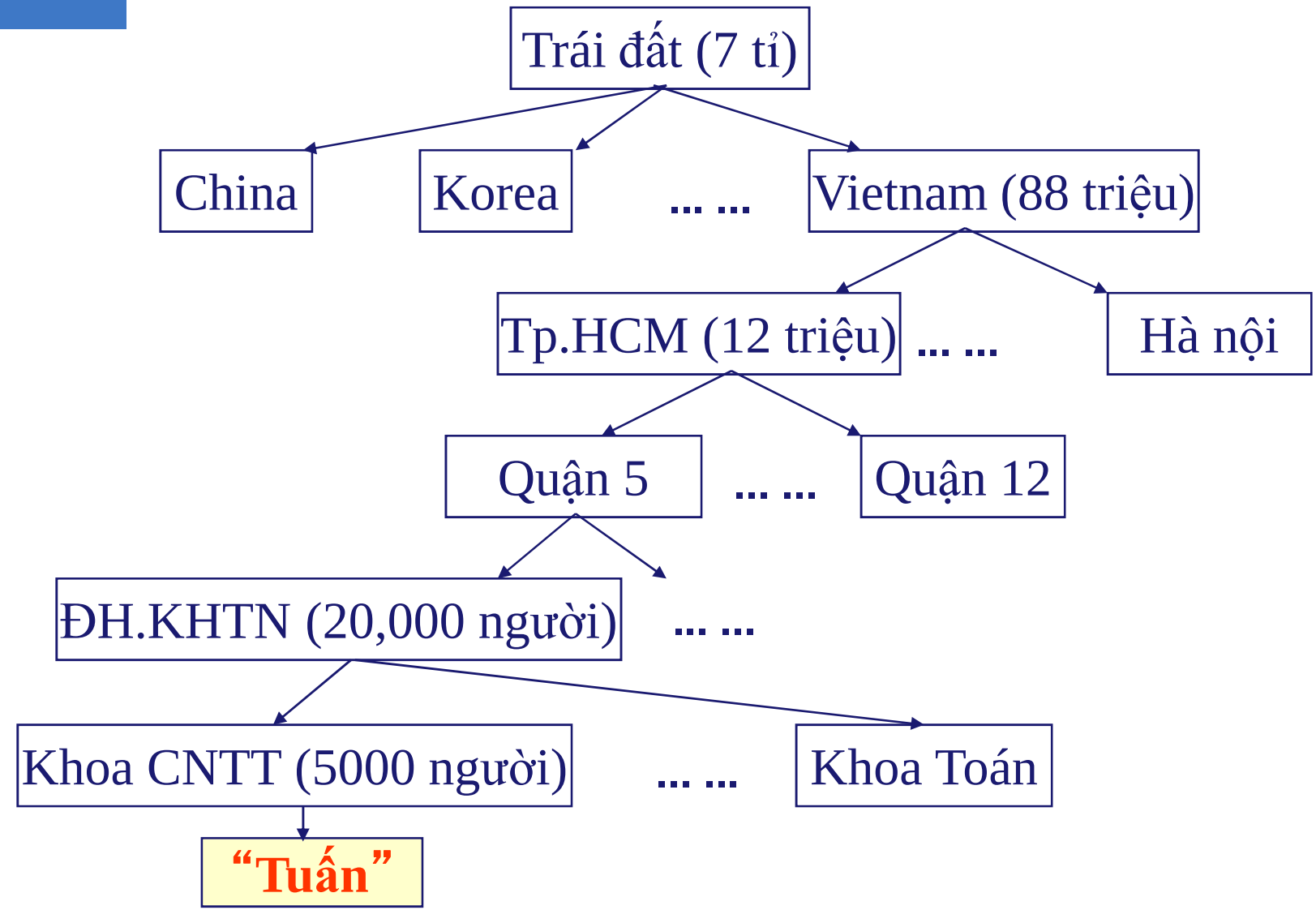
- Các ví dụ
- Đặc điểm của cấu trúc cây
- Tree ADT
- Các thuật ngữ liên quan
- Các định lý



Các ví dụ (1)

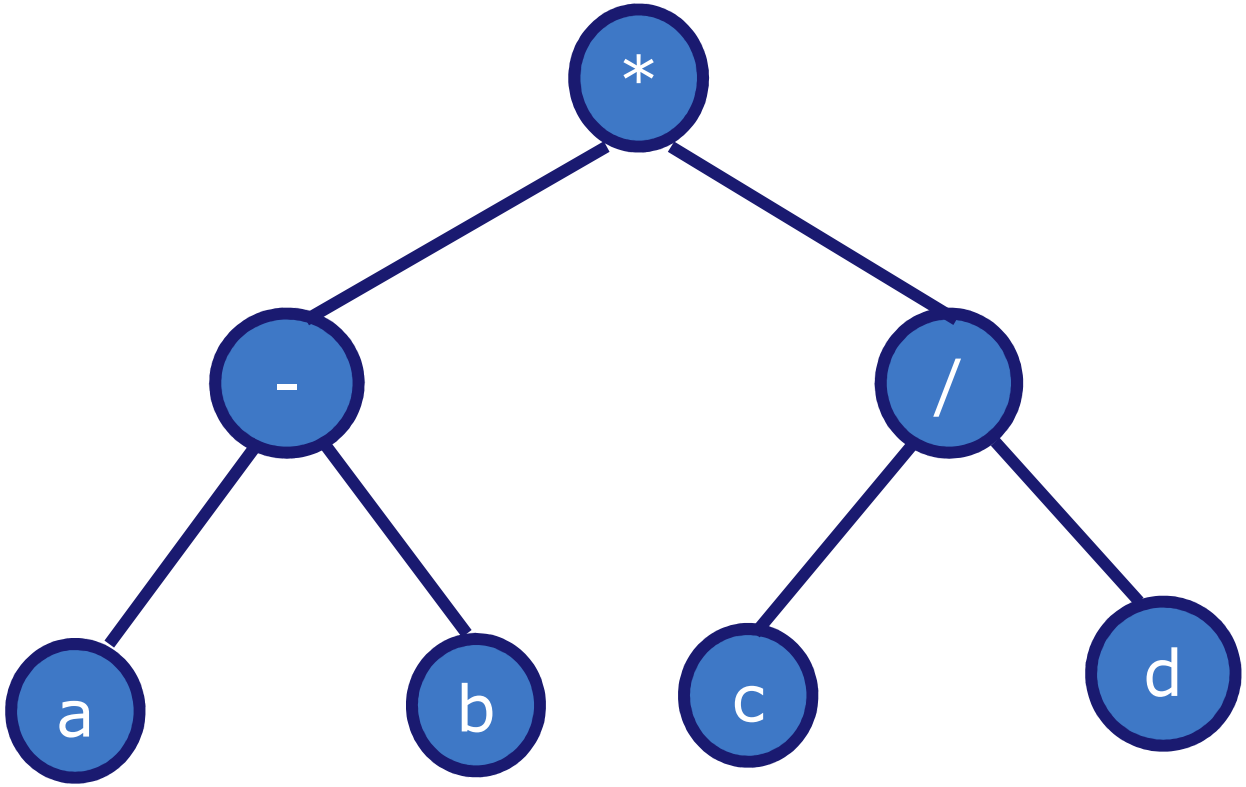
- Ví dụ 1: cách lưu trữ phân cấp → bài toán đưa thư
  - Cần tìm 1 người: Tuấn, khoa CNTT, ĐH KHTN, Quận 5, Tp.HCM, Việt nam
  - Cách tìm ra "Tuấn" nhanh nhất ?
  - Sử dụng mảng (array) ?
  - Sử dụng danh sách liên kết (linked list) ?

Các ví dụ (2)


Trái đất (7 tỉ)
China
Korea
... ...
Vietnam (88 triệu)
Tp.HCM (12 triệu)
... ...
Hà nội
Quận 5
... ...
Quận 12
ĐH.KHTN (20,000 người)
... ...
Khoa CNTT (5000 người)
... ...
Khoa Toán
"Tuấn"

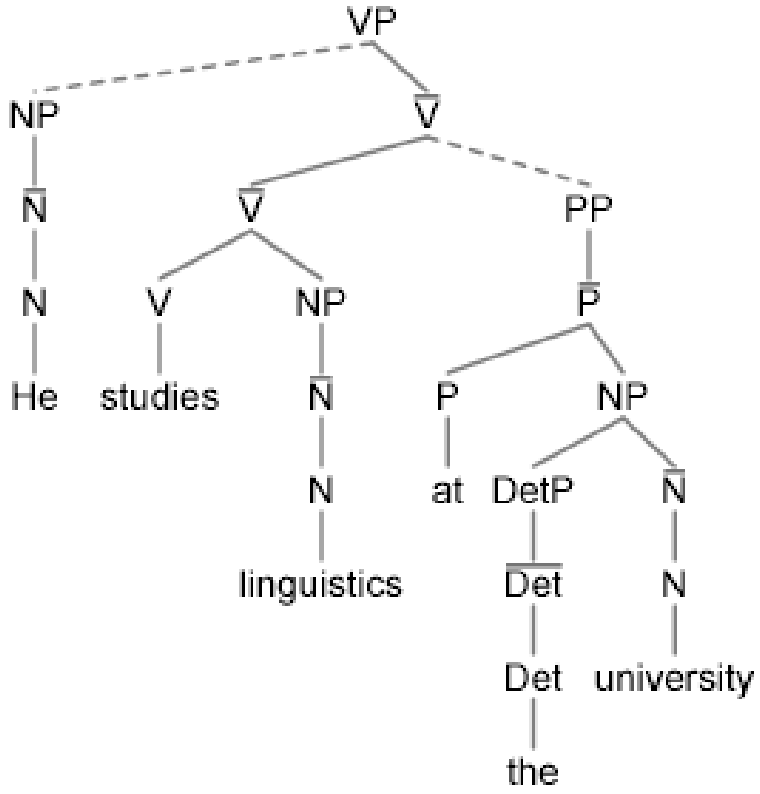
Các ví dụ (3)

- Ví dụ 2: cây biểu thức (a-b)*(c/d)



Các ví dụ (4)

- Ví dụ 3: cây ngữ pháp – mô tả các thành phần ngữ pháp trong một câu

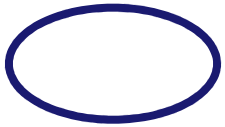


Đặc điểm của cấu trúc cây

- Cây là 1 cấu trúc dữ liệu quan trọng để biểu diễn tính "kế thừa", "phân cấp"
  - Cây gia phả (trong các dòng họ)
  - Cây phân cấp các loài (trong sinh vật)
  - …
- Linked List
  - Chèn/xóa phần tử: O(1)
  - Tìm kiếm: O(n)
- Cây nhị phân tìm kiếm
  - Thêm/xóa phần tử: $O(\log_2 n)$
  - Tìm kiếm: $O(\log_2 n)$

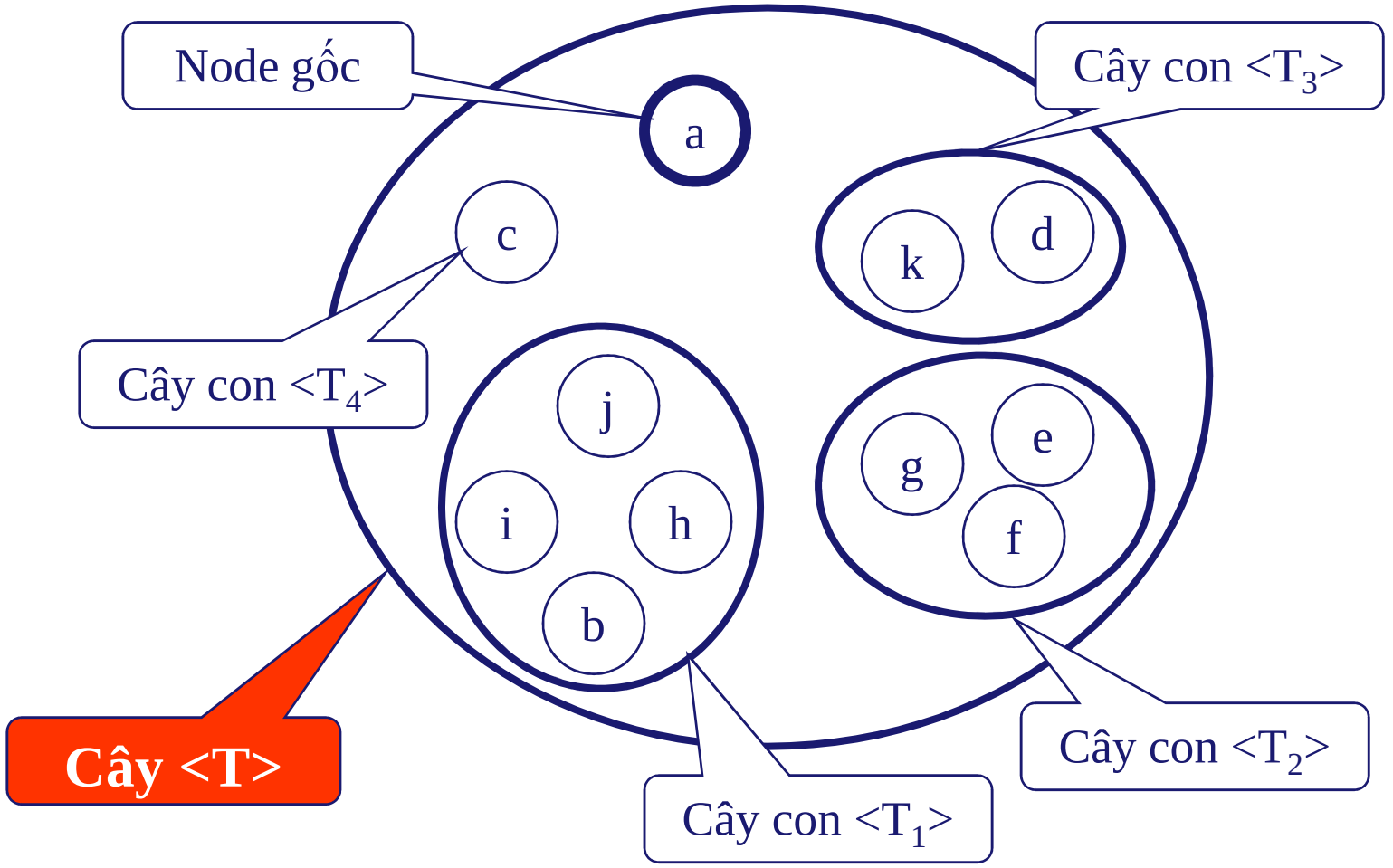
Tree ADT (1)

- Một cây <T> (Tree) là:
  - Một tập các phần tử, gọi là các node $p_1, p_2, \ldots, p_N$
  - Nếu N=0, cây <T> gọi là cây rỗng (NULL)
  - Nếu N>0:
    - Tồn tại duy nhất 1 node p_r gọi là gốc của cây
    - Các node còn lại được chia thành m tập hợp không giao nhau: $T_1, T_2, \ldots, T_m$
    - Mỗi $<T_i>$ là 1 cây con của cây <T>

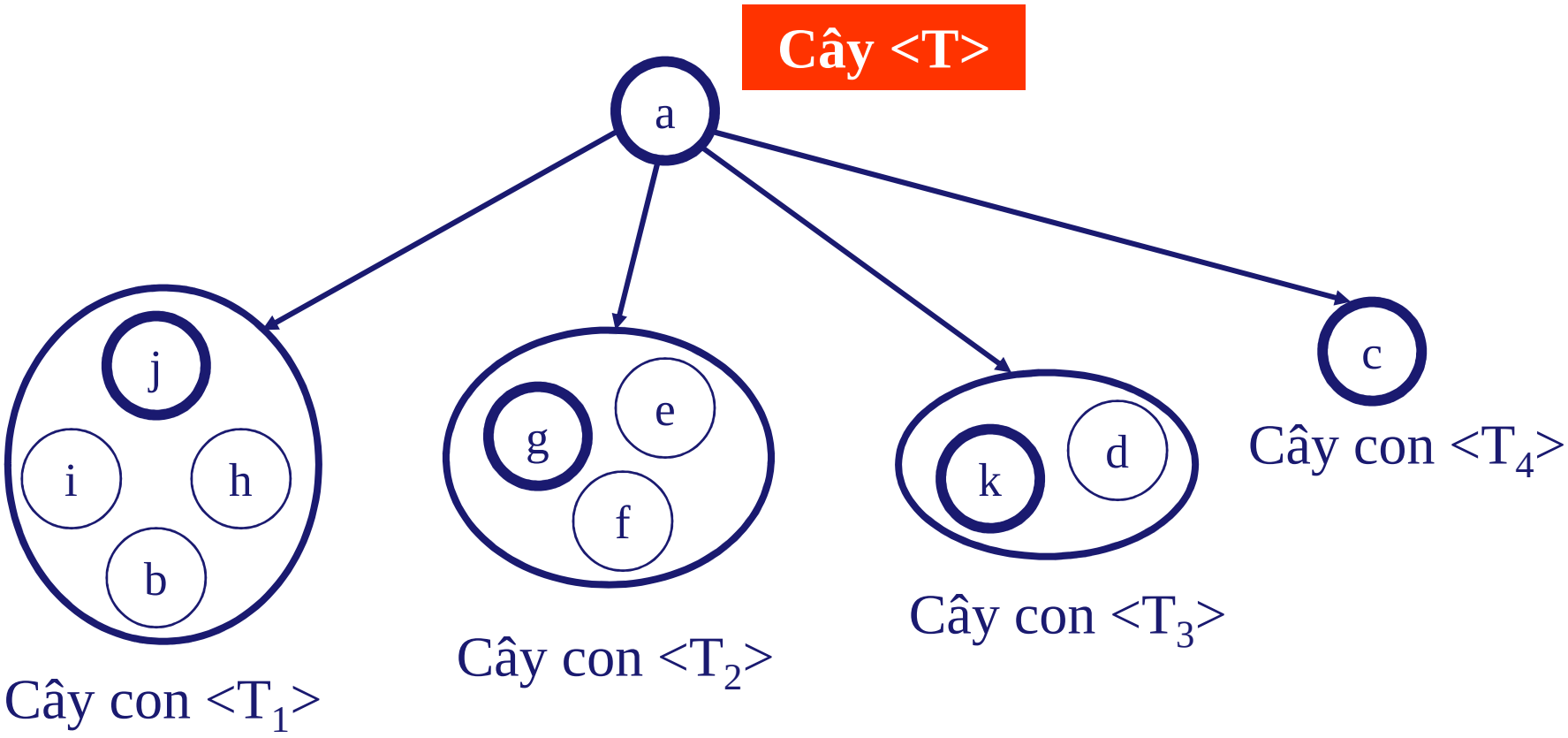


Tập rỗng → Cây <T> rỗng (NULL)

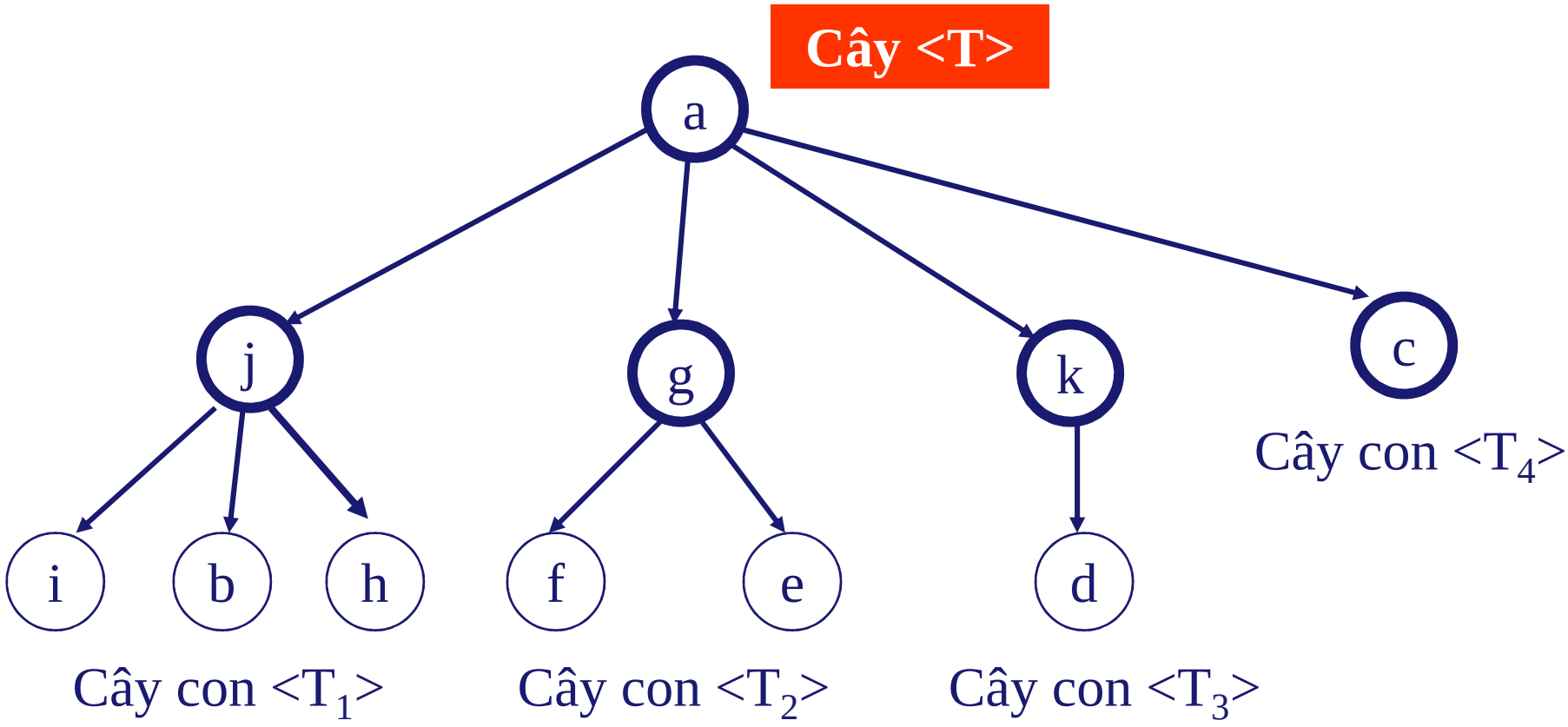
Tree ADT (2)


Node gốc
a
Cây con <T3>
c
k
d
Cây con <T4>
j
g
e
i
h
f
b
Cây <T>
Cây con <T1>
Cây con <T2>

Tree ADT (3)



Tree ADT (4)



Tree ADT (5)

- Các tính chất của cây:
  - Node gốc không có node cha
  - Mỗi node con chỉ có 1 node cha
  - Mỗi node có thể có nhiều node con
  - Không có chu trình

Tree ADT (6)

- Các thao tác cơ bản trên cây:
  - Khởi tạo cây rỗng
  - Xóa cây
  - Thêm một node
  - Xóa một node
  - Duyệt cây
  - Kiểm tra cây rỗng
  - Đếm số node trong cây
  - Tính chiều cao của cây

Các thuật ngữ liên quan (1)

- Node: là 1 phần tử trong cây. Mỗi node có thể chứa 1 dữ liệu bất kỳ
- Nhánh (Branch): là đoạn nối giữa 2 node
- Node cha (Parent node)
- Node con (Child node)
- Node anh em (sibling nodes): là những nút có cùng node cha
- Bậc của 1 node p: là số node con của p
  - Bậc (a) = 4; Bậc (j) = 3; Bậc (g) = 2;
  - Bậc (k) = 1; Bậc (c) = 0

Các thuật ngữ liên quan (2)

- Node gốc (Root node): node không có node cha
- Node lá (Leaf node): node có bậc = 0 (không có node con)
- Node nội (Internal node): là node có node cha và có node con
- Cây con (Subtree)
  - *Trắc nghiệm*: có bao nhiêu cây con trong cây <T> ?

Các thuật ngữ liên quan (3)

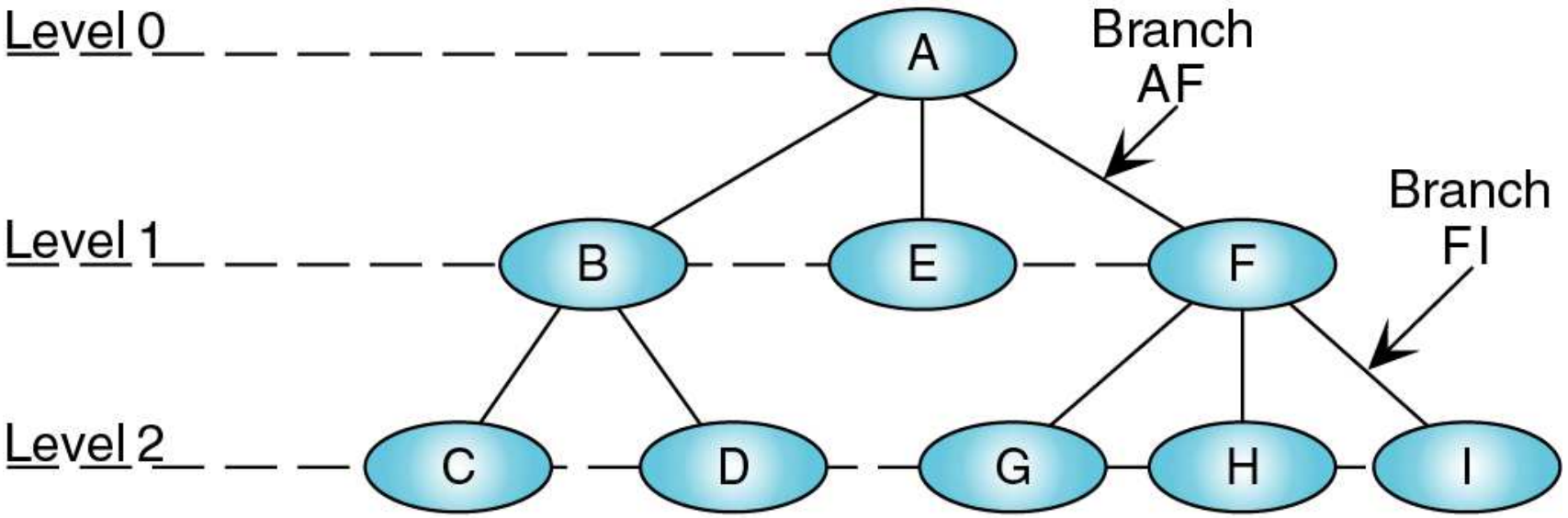
- Bậc của cây: là bậc lớn nhất của các node trong cây
  - Bậc (<T>) = max {bậc (p_i) / $p_i \in$ <T>}
  - Bậc của cây <T> ?
- Đường đi (Path) giữa node p_i đến node p_j: là dãy các node liên tiếp từ p_i đến p_j sao cho giữa hai node kề nhau đều có nhánh
  - Path(a, d) ?

Các thuật ngữ liên quan (4)

- Mức (Level):
  - Mức (p) = 0 nếu p = root
  - Mức (p) = 1 + Mức (Cha (p)) nếu p!=root

- Chiều cao của cây (Height - h_T): đường đi dài nhất từ node gốc đến node lá
  - h_T = max {Path(root, p_i) / p_i là node lá $\in$ <T>}
  - h_T ?

Các thuật ngữ liên quan (5)



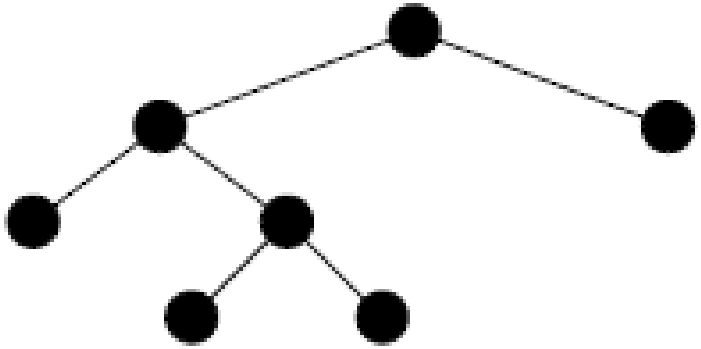
Parents: A, B, F
Children: B, E, F, C, D, G, H, I
Siblings: {B,E,F}, {C,D}, {G,H,I}

Leaves C,D,E,G,H,I
Internal nodes B,F

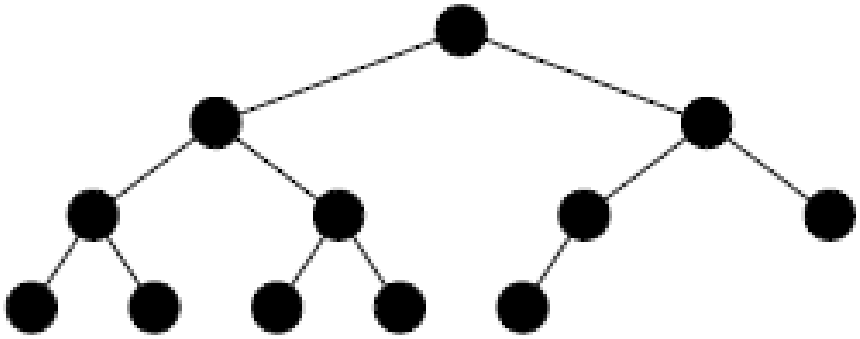
Các thuật ngữ liên quan (6)

- Cây nhị phân (binary tree)
  - Cây nhị phân là cây có bậc = 2
- Cây nhị phân đầy đủ (full binary tree)
  - Mỗi node có 0 hoặc 2 node con
- Cây nhị phân hoàn chỉnh (complete binary tree)
  - Từ mức 0 đến mức h-1: có đủ số node (completely full)
  - Mức h: các node được thêm vào cây từ trái sang phải

Các thuật ngữ liên quan (7)



Full but not complete

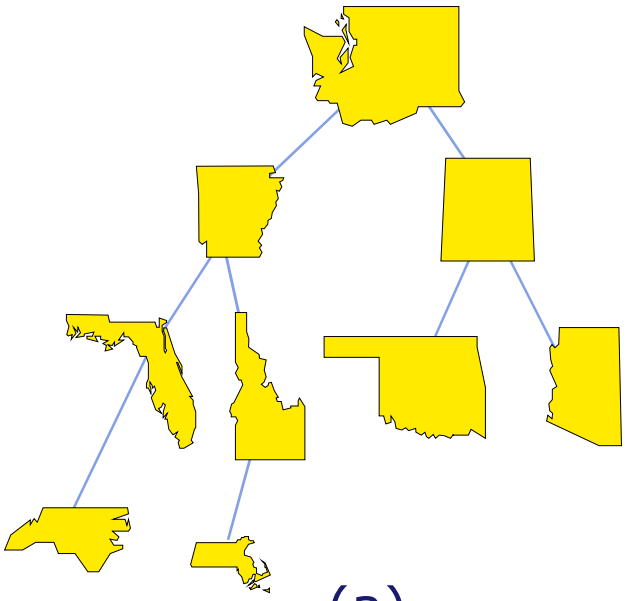


Complete but not full

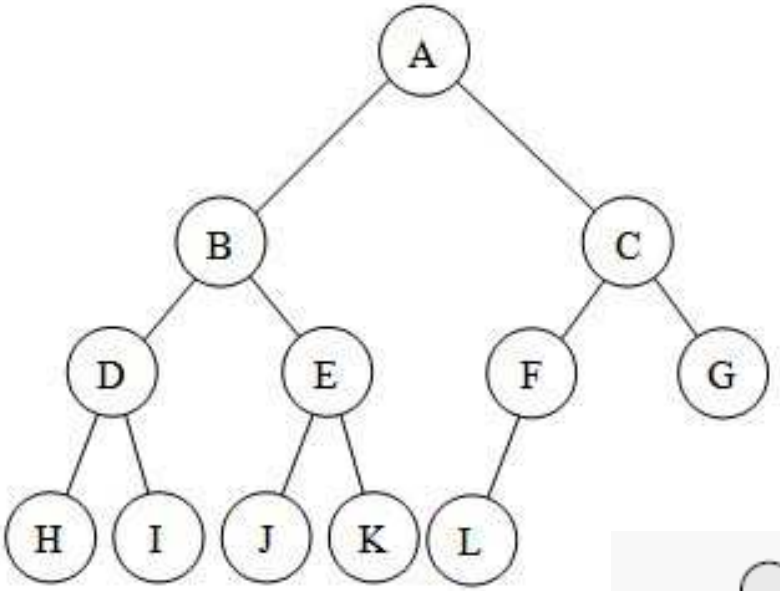
?

Complete and full

Các thuật ngữ liên quan (8)



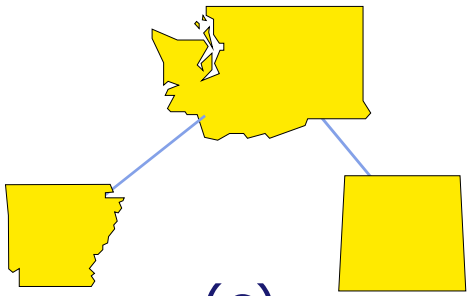
(a)



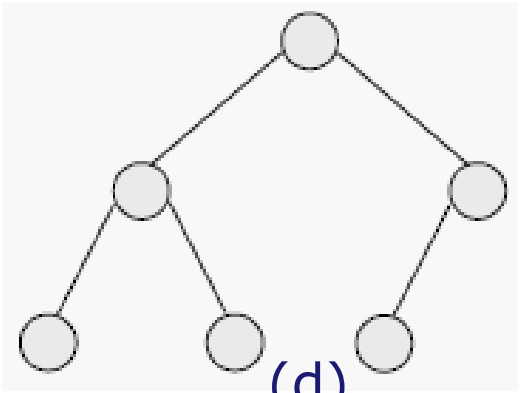
(b)

Complete ?

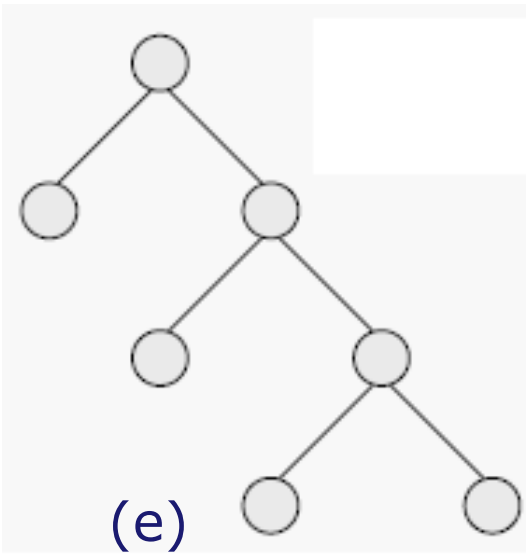
Full ?



(c)



(d)



(e)

Các định lý (1)

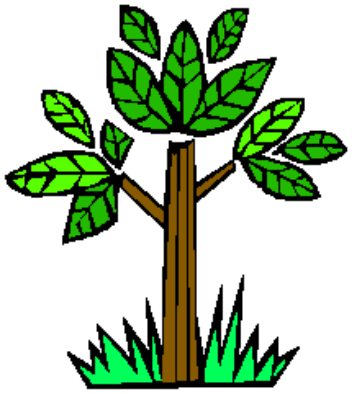
- Cho T là một cây nhị phân đầy đủ (full binary tree). Gọi N là số node, L là số node lá, I là số node nội (tinh ca node goc)
  - L = I + 1
  - N = 2I + 1
  - I = (N – 1)/2
  - L = (N + 1)/2
  - N = 2L – 1
  - I = L – 1

Các định lý (2)

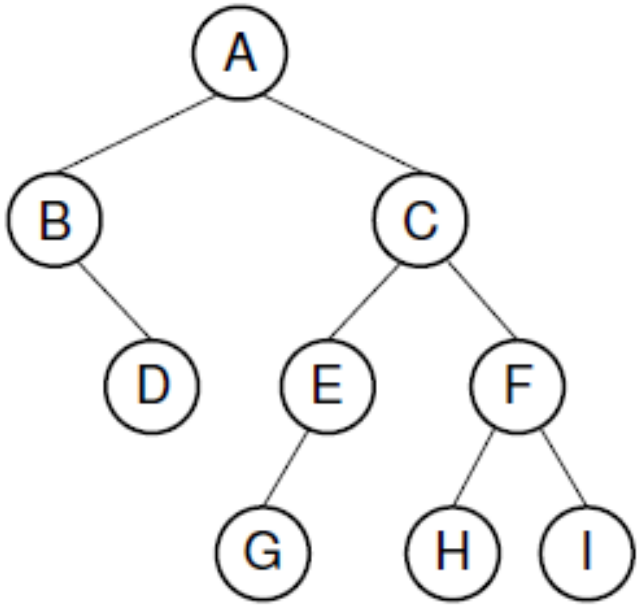
- Nếu T là một cây nhị phân có h level thì số node tối đa của T là $2^h - 1$
- Nếu T là một cây nhị phân có h level thì số node lá tối đa là 2^{h-1}
- Nếu T là một cây nhị phân, có không quá 2^h node tại mức h ($h \geq 0$)
- Nếu T là một cây nhị phân có N node thì số mức tối thiểu của T là $\lceil \log_2(N + 1) \rceil$
- Nếu T là một cây nhị phân có L node lá thì số mức tối thiểu của T là $\lceil \log_2 L \rceil + 1$

Cây nhị phân

- Các khái niệm và thuật ngữ cơ bản
- Cài đặt cấu trúc dữ liệu
- Duyệt cây
- Cây nhị phân tìm kiếm – Binary Search Tree
- Hàng đợi ưu tiên – Priority Queue



Cài đặt cây nhị phân bằng mảng (1)



# index	Node	Left	Right
0	A	1	2
1	B	-1	3
2	C	4	5
3	D	-1	-1
4	E	6	-1
5	F	7	8
6	G	-1	-1
7	H	-1	-1
8	I	-1	-1

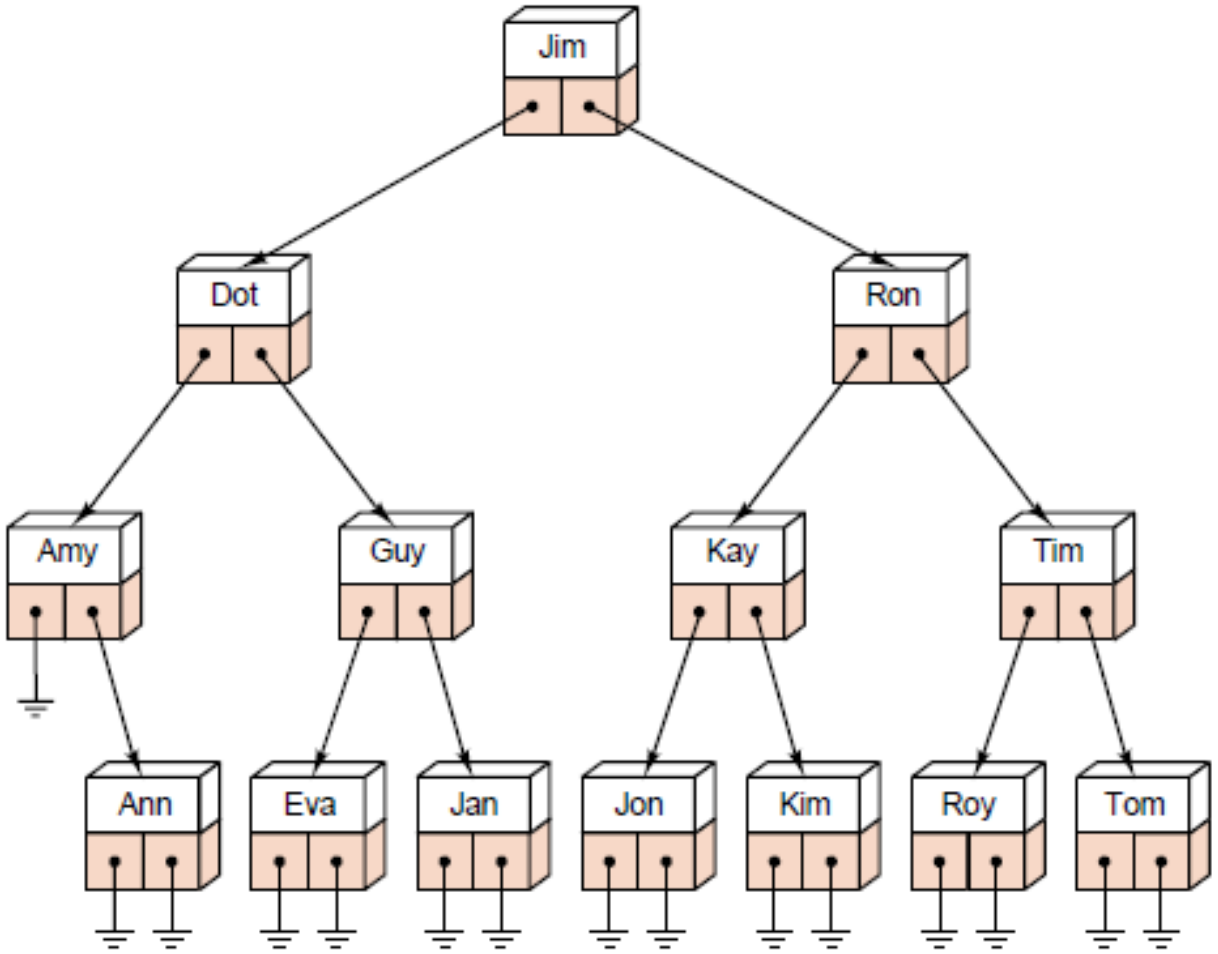
Cài đặt cây nhị phân bằng mảng (2)

```
template <class T> class BINARY_TREE {
    private:
           struct TreeNode  {
                      T                    data;      // data of node
                      int                  left;      // index to left child
                      int                  right;     // index to right child
           };
           int        root;                            // index to root of tree
           int        maxSize;                         // max number of node in tree
           TreeNode   *nodes;                          // array to store nodes of tree

           void       LNR(int p);
           void       NLR(int p);
           void       LRN(int p);
   public:
           BINARY_TREE(int size);                      // init a tree with 'size' node
           BINARY_TREE(const BINARY_TREE &aTree);      // copy constructor
           ~BINARY_TREE();                             // destructor

           // operations
           bool       isEmpty();
           int        countNode();
           int        height();
           bool       insert(T newItem);
           bool       remove(T item);
           void       preorder();                      // call NLR(root)
           void       inorder();                       // call LNR(root)
           void       postorder();                     // call LRN(root)
}; // end class
```

Cài đặt cây nhị phân bằng con trỏ (1)



A linked binary tree

Cài đặt cây nhị phân bằng con trỏ (2)

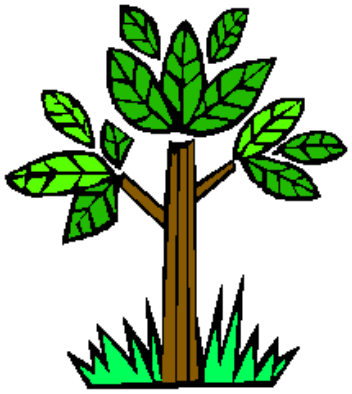
```
template <class T> class BINARY_TREE {
    private:
          struct TreeNode  {
                    T                   data;     // data of node
                    TreeNode            *left;    // pointer to left child
                    TreeNode            *right;   // pointer to right child
          };
          TreeNode            *root;              // pointer to root of tree

          void      LNR(TreeNode *p);
          void      NLR(TreeNode *p);
          void      LRN(TreeNode *p);
   public:
          BINARY_TREE();                          // default constructor
          BINARY_TREE(const BINARY_TREE &aTree);  // copy constructor
          ~BINARY_TREE();                         // destructor

          // operations
          bool      isEmpty();
          int       countNode();
          int       height();
          bool      insert(T newItem);
          bool      remove(T item);
          void      preorder();                   // call NLR(root)
          void      inorder();                    // call LNR(root)
          void      postorder();                  // call LRN(root)
}; // end class
```

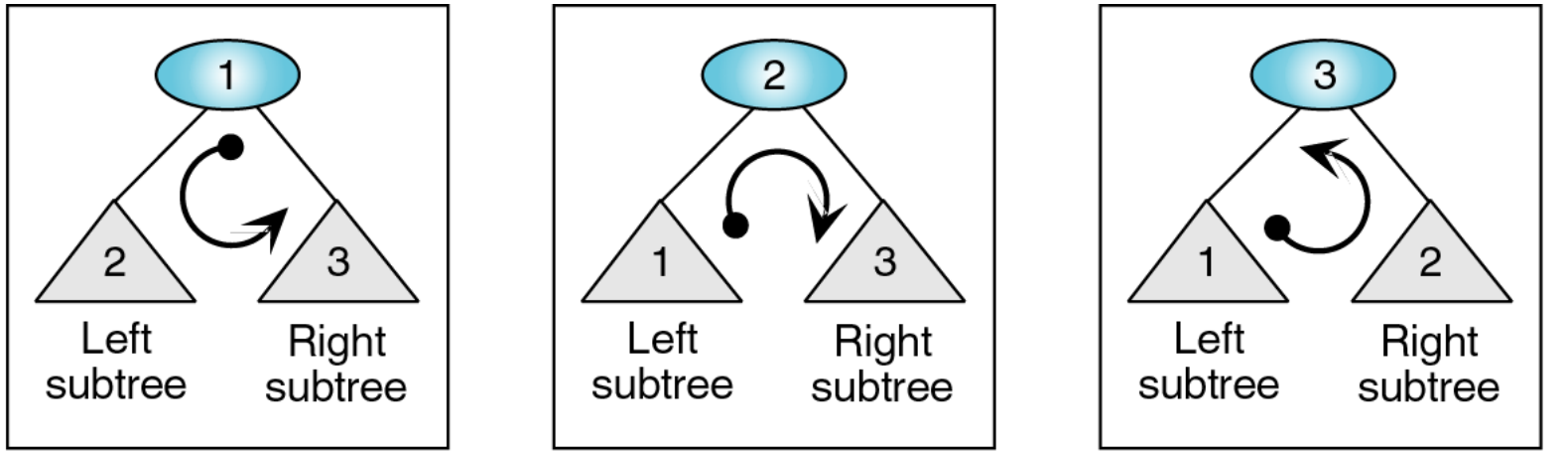
Cây nhị phân

- Các khái niệm và thuật ngữ cơ bản
- Cài đặt cấu trúc dữ liệu
- Duyệt cây
- Cây nhị phân tìm kiếm – Binary Search Tree
- Hàng đợi ưu tiên – Priority Queue



Duyệt cây (1)

- Có 3 cách duyệt cây:
  - Duyệt gốc trước (Pre-Order) - NLR
  - Duyệt gốc giữa (In-Order) - LNR
  - Duyệt gốc sau (Post-Order) - LRN

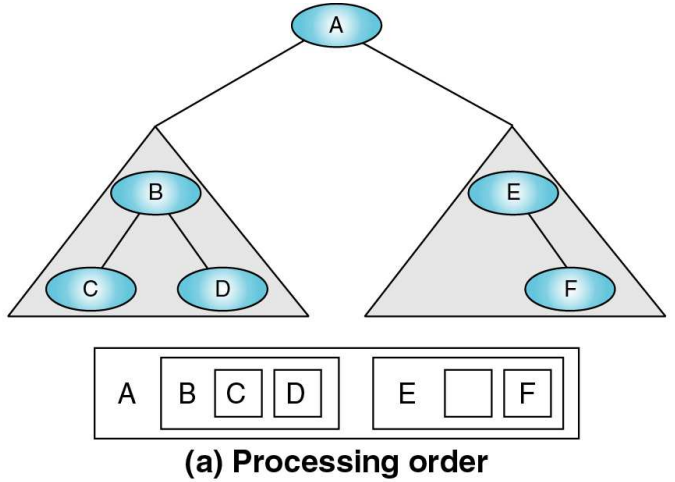


(a) Preorder traversal **(b) Inorder traversal** **(c) Postorder traversal**

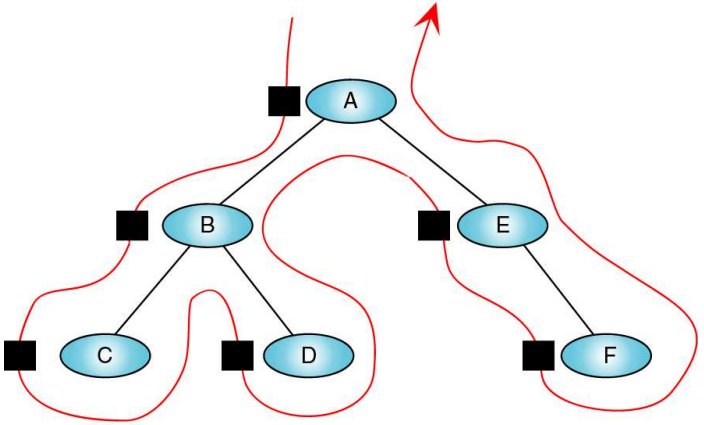
Duyệt cây (2)

- Duyệt gốc trước (Pre-Order) - NLR

```
template <class T>
void BINARY_TREE<T>::NLR(TreeNode *p)
{
   if (p==NULL) return;
   "Xử lý nút gốc p"
   NLR(p->left);
   NLR(p->right);
}
```

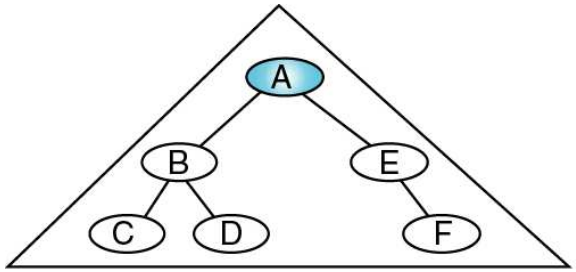


(a) Processing order

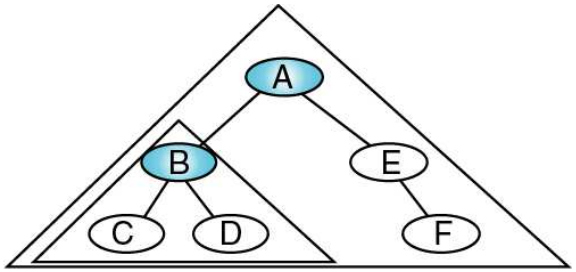


(b) "Walking" order

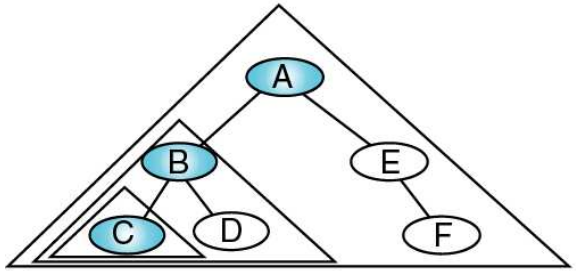
Duyệt cây (3)



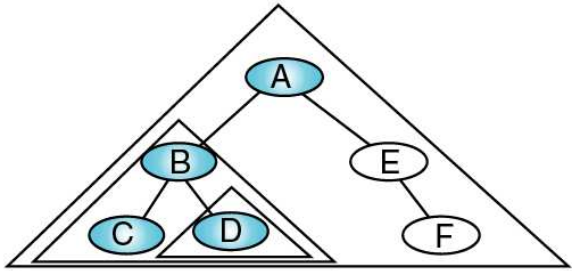
(a) Process tree A



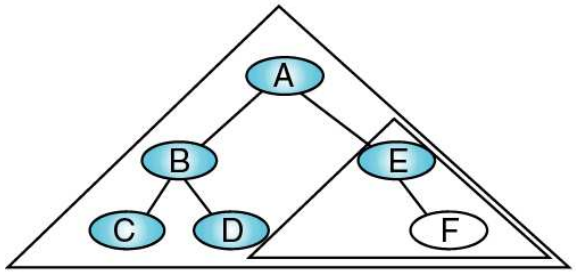
(b) Process tree B



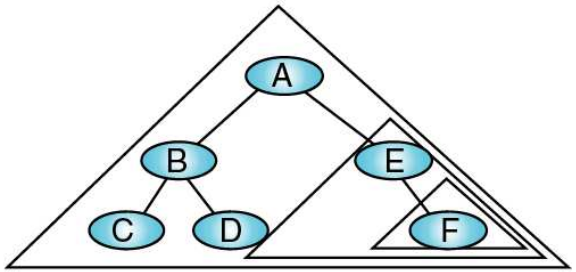
(c) Process tree C



(d) Process tree D



(e) Process tree E



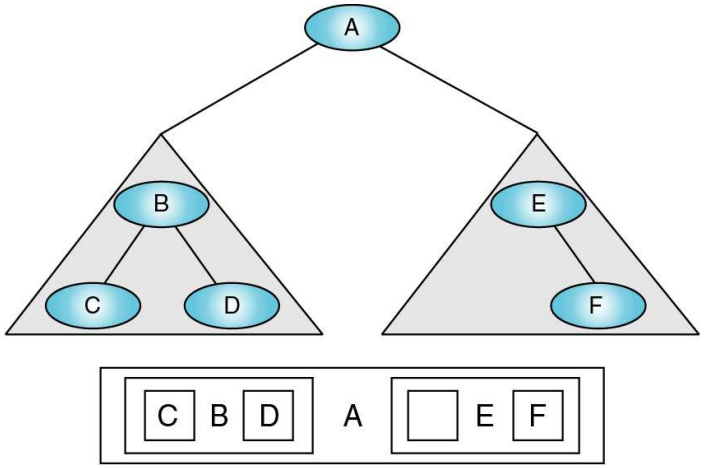
(f) Process tree F

Minh họa cách duyệt "gốc trước"

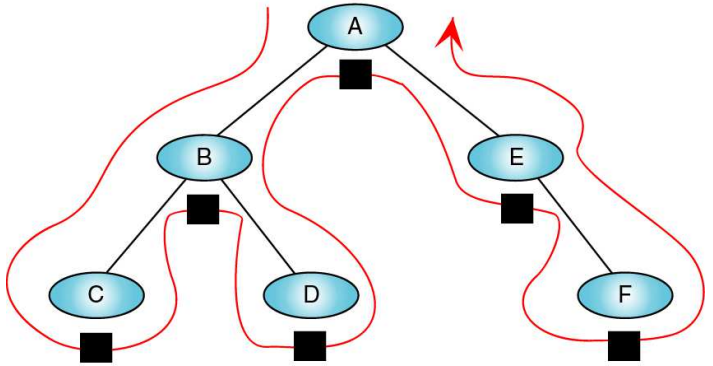
Duyệt cây (4)

- Duyệt gốc giữa (In-Order) - LNR

```
template <class T>
void BINARY_TREE<T>::LNR(TreeNode *p)
{
   if (p==NULL) return;
   LNR(p->left);
   "Xử lý nút gốc p"
   LNR(p->right);
}
```



(a) Processing order

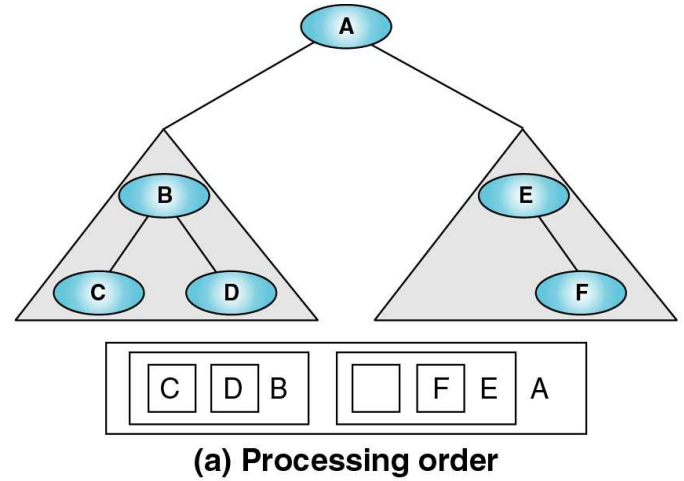


(b) "Walking" order

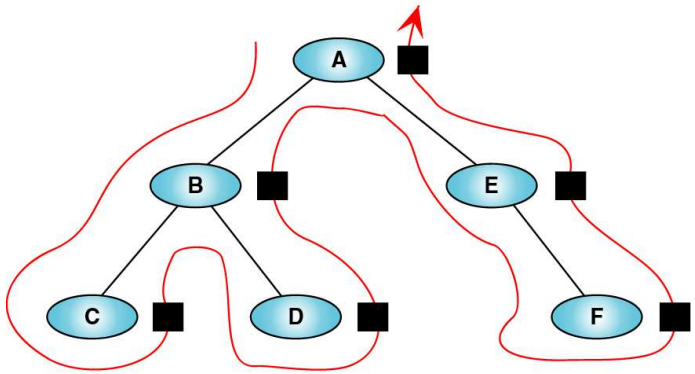
Duyệt cây (5)

- Duyệt gốc sau (Post-Order) - LRN

```
template <class T>
void BINARY_TREE<T>::LRN(TreeNode *p)
{
   if (p==NULL) return;
   LRN(p->left);
   LRN(p->right);
   "Xử lý nút gốc p"
}
```



(a) Processing order



(b) "Walking" order