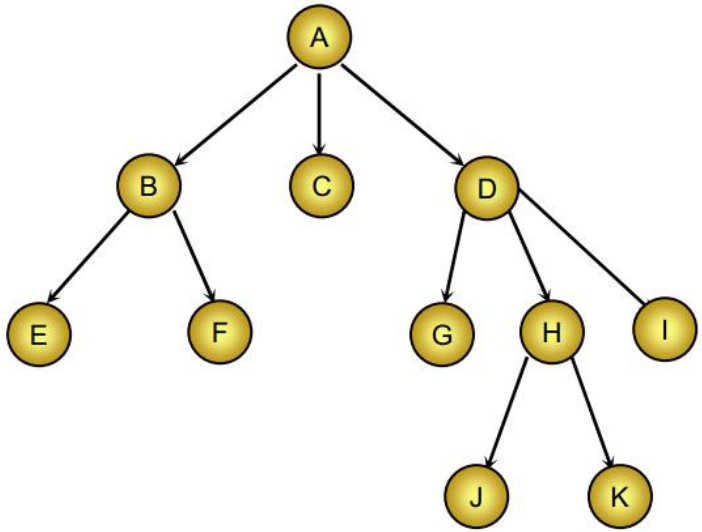
Bài 6. Cây

6.1. Cây (tree) – định nghĩa

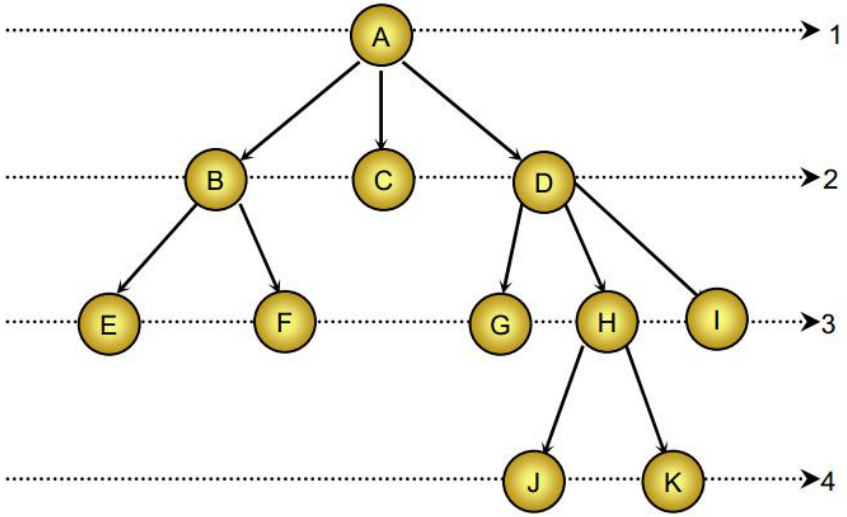
- Cây là một cấu trúc dữ liệu gồm một tập hữu hạn các nút, giữa các nút có một quan hệ phân cấp gọi là quan hệ "cha – con".
- Trong cây, có một nút đặc biệt, không có nút cha, gọi là gốc (root) của cây



- Có thể định nghĩa cây bằng các đệ quy như sau:
  - Mỗi nút là một cây, và là gốc của cây ấy
  - Nếu n là một nút và n_1, n_2, …, n_k lần lượt là gốc của các cây T_1, T_2, …, T_k; các cây này đôi một không có nút chung. Thì nếu cho nút n trở thành cha của các nút n_1, n_2, …, n_k ta sẽ được một cây mới T. Cây này có nút n là gốc còn các cây T_1, T_2, …, T_k trở thành các cây con (subtree) của gốc.
- Cây không có nút nào mà ta gọi là cây rỗng (null tree).

6.1. Cây (tree) – thuật ngữ

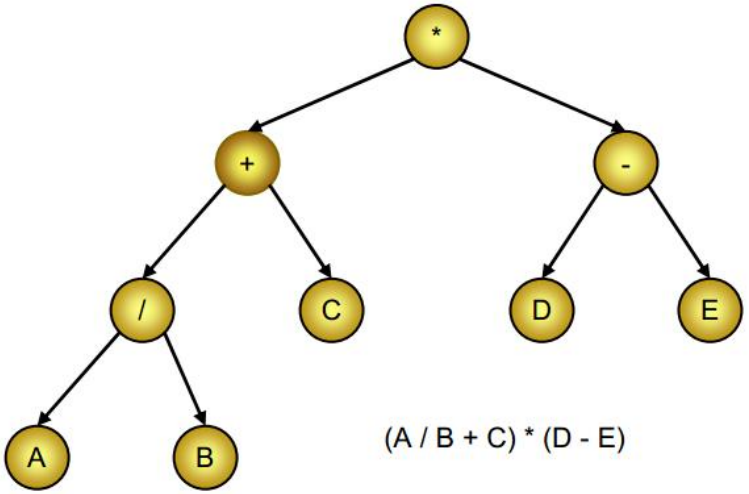
- Số các con của một nút được gọi là **cấp** của nút đó
- Nút có cấp bằng o được gọi là **nút lá** (leaf node) hay nút ngoài, hay nút tận cùng
- Những nút không phải là lá được gọi là **nút nhánh** (branch node) hày nút trong



- Cấp cao nhất của một nút trên cây gọi là **cấp của cây** đó
- Gốc của cây người ta gán cho số **mức** là 1, nếu nút cha có mức là i thì nút con sẽ có mức là i + 1
- **Chiều cao** (height) hay chiều sâu (depth) của một cây là số mức lớn nhất của nút có trên cây đó
- Một tập hợp các cây phân biệt được gọi là **rừng** (forest), một cây cũng là một rừng. Nếu bỏ nút gốc trên cây thì sẽ tạo thành một rừng các cây con.

6.1. Cây (tree) – ví dụ

- Mục lục của một cuốn sách với phần, chương, bài, mục v.v...
- Cấu trúc thư mục trên đĩa cũng có cấu trúc cây, thư mục gốc có thể coi là gốc của cây đó với các cây con là các thư mục con và tệp nằm trên thư mục gốc

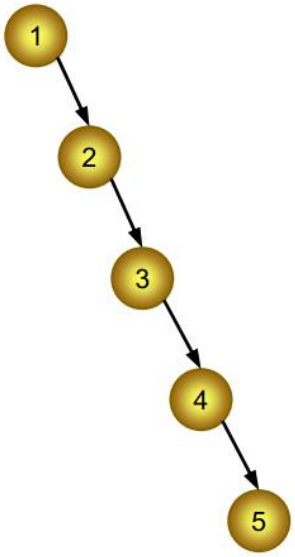


- Gia phả của một họ tộc cũng có cấu trúc cây
- Một biểu thức số học gồm các phép toán cộng, trừ, nhân, chia cũng có thể lưu trữ trong một cây mà các toán hạng được lưu trữ ở các nút lá, các toán tử được lưu trữ ở các nút nhánh, mỗi nhánh là một biểu thức con.

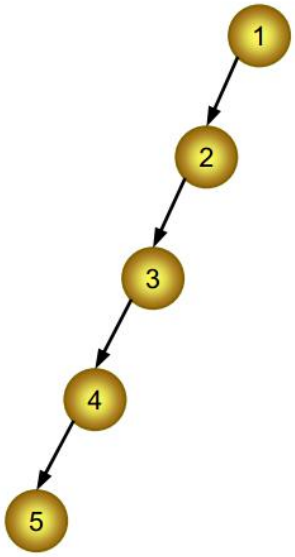
6.2. Cây nhị phân (binary tree)

- Cây nhị phân là cây có đặc điểm:
  - Mọi nút trên cây chỉ có tối đa **2 nhánh con**
  - Với mỗi nút, phân biệt cây con **trái** và cây con **phải** của nút đó
- Một số dạng đặc biệt của cây nhị phân
  - Cây nhị phân suy biến (degenerate binary tree): các nút không phải là lá chỉ có một nhánh con
  - Cây nhị phân hoàn chỉnh (complete binary tree): tất cả các nút nhánh đều có đúng 2 nút con
  - Cây nhị phân đầy đủ (full binary tree): là cây nhị phân hoàn chỉnh mà tất cả các nút lá đều có cùng mức

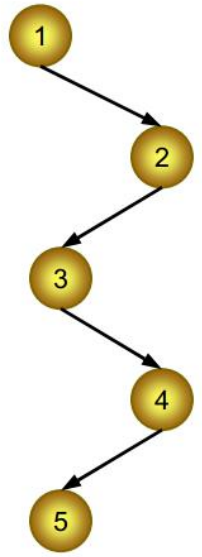
6.2. Cây nhị phân (binary tree)



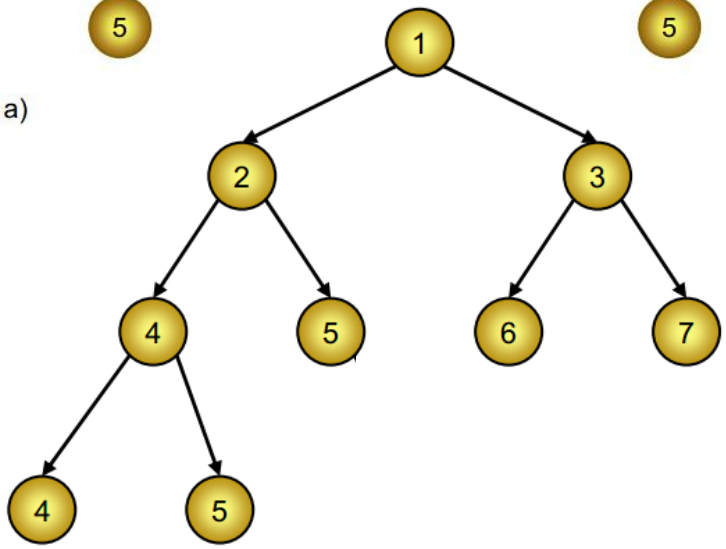
a)



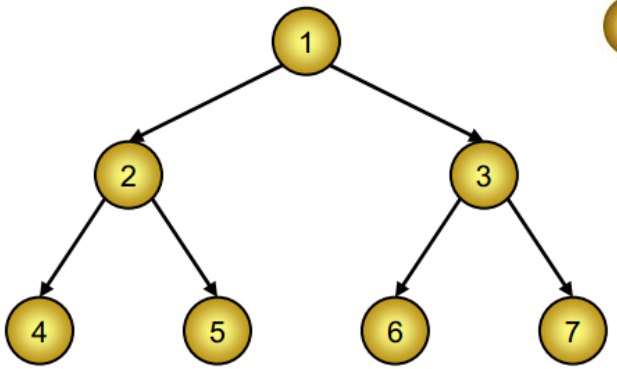
b)



c)



e)



f)

6.2. Cây nhị phân – tính chất

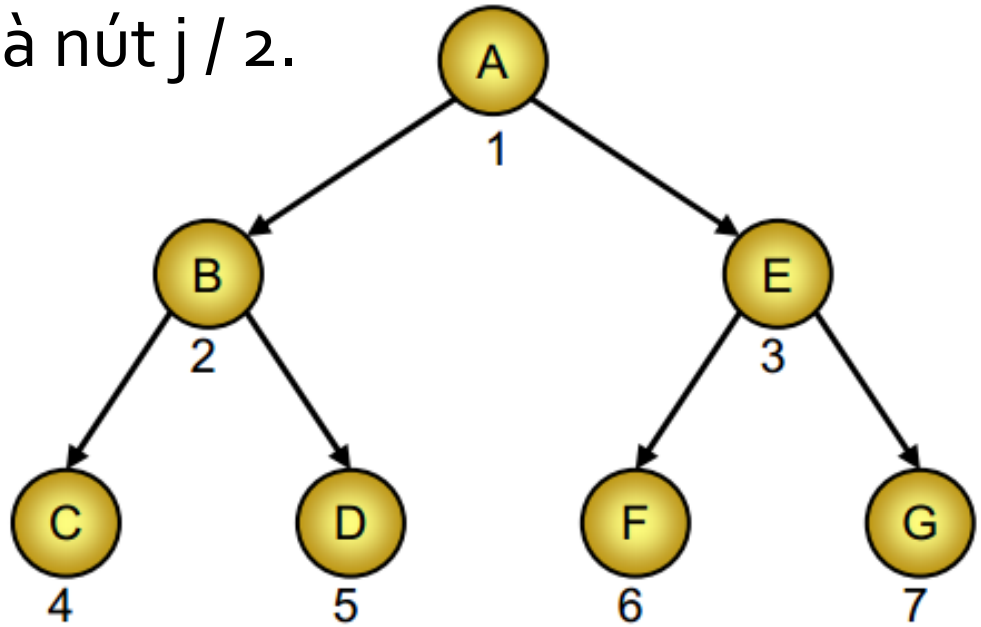
- Trong các cây nhị phân có cùng số lượng nút:
  - Cây nhị phân suy biến có chiều cao lớn nhất
  - Cây nhị phân đầy đủ thì có chiều cao nhỏ nhất
- Số lượng các nút trên mức ***i*** của cây nhị phân tối đa là 2^{i-1}, tối thiểu là 1 ($i \geq 1$).
- Số lượng các nút trên một cây nhị phân có chiều cao ***h*** tối đa là $2^0+2^1+..+2^{h-1}=\mathbf{2^h-1}$, tối thiểu là h ($h \geq 1$).
- Cây nhị phân đầy đủ có ***n*** nút thì chiều cao của nó là $h = \lfloor \log_2 n \rfloor + 1$.

6.3. Biểu diễn cây nhị phân

- Biểu diễn bằng mảng
- Biểu diễn bằng cấu trúc liên kết

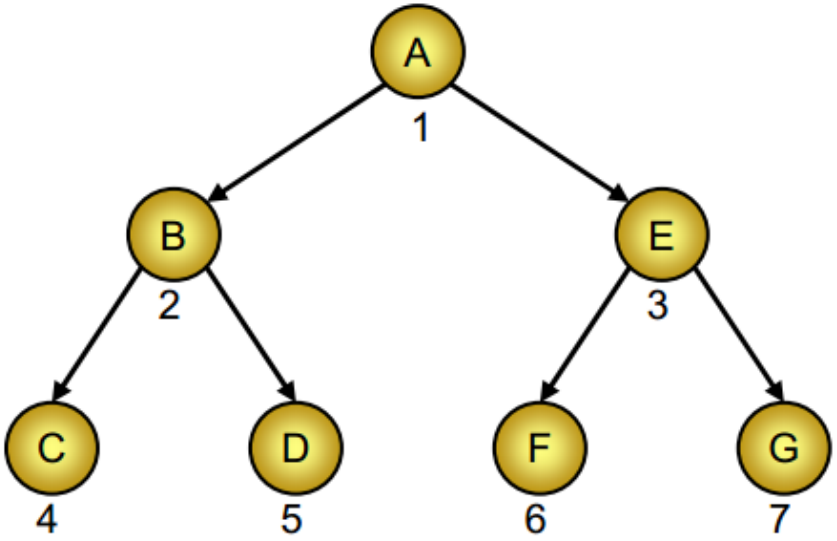
6.3.1. Biểu diễn cây nhị phân bằng mảng

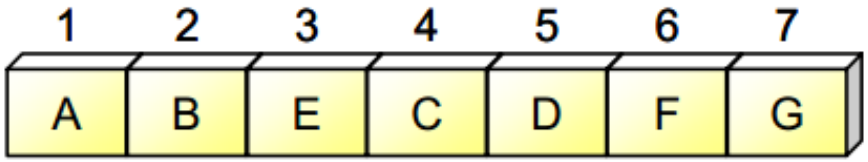
- Với một cây nhị phân **đầy đủ**, ta **đánh số** các nút theo thứ tự lần lượt từng mức một, từ mức 1 trở đi, và từ trái sang phải ở mỗi mức, khi đó:
  - Con của nút thứ i sẽ là các nút thứ 2i và 2i + 1.
  - Cha của nút thứ j là nút j / 2.



6.3.1. Biểu diễn cây nhị phân bằng mảng

- Từ đó có thể lưu trữ cây bằng một mảng T, nút thứ i của cây được lưu trữ bằng phần tử T[i]



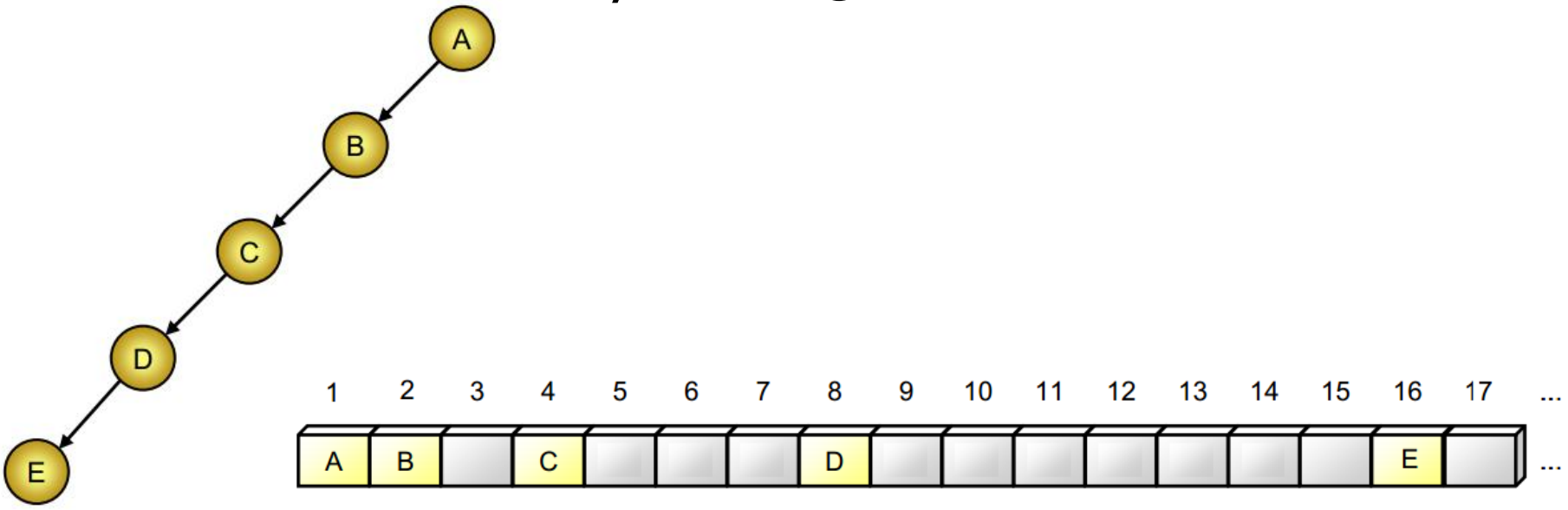


6.3.1. Biểu diễn cây nhị phân bằng mảng

- Đối với cây nhị phân **không đầy đủ**, ta có thể thêm vào một số **nút giả** để được cây nhị phân đầy đủ và:
  - gán những **giá trị đặc biệt** cho những phần tử trong mảng T tương ứng với những nút này
  - hoặc dùng thêm một **mảng phụ** để đánh dấu những nút nào là nút giả tự ta thêm vào
- Nhược điểm: lãng phí bộ nhớ để lưu các nút giả (có thể rất nhiều)

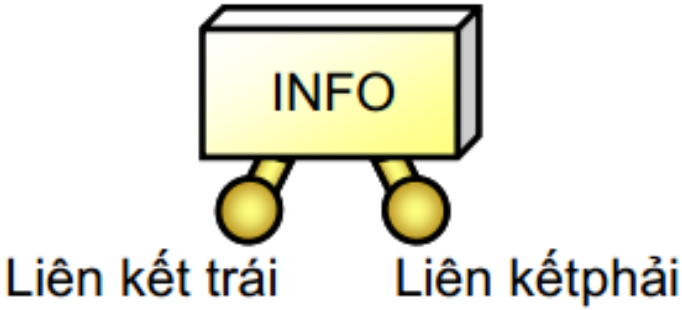
6.3.1. Biểu diễn cây nhị phân bằng mảng

- Ví dụ: với cây nhị phân 5 mức, ta cần mảng với $2^5-1 = 31$ ô nhớ để lưu trữ cây. Tuy nhiên với cây nhị phân suy biến, số ô nhớ thực sự chứa nút của cây chỉ là 5



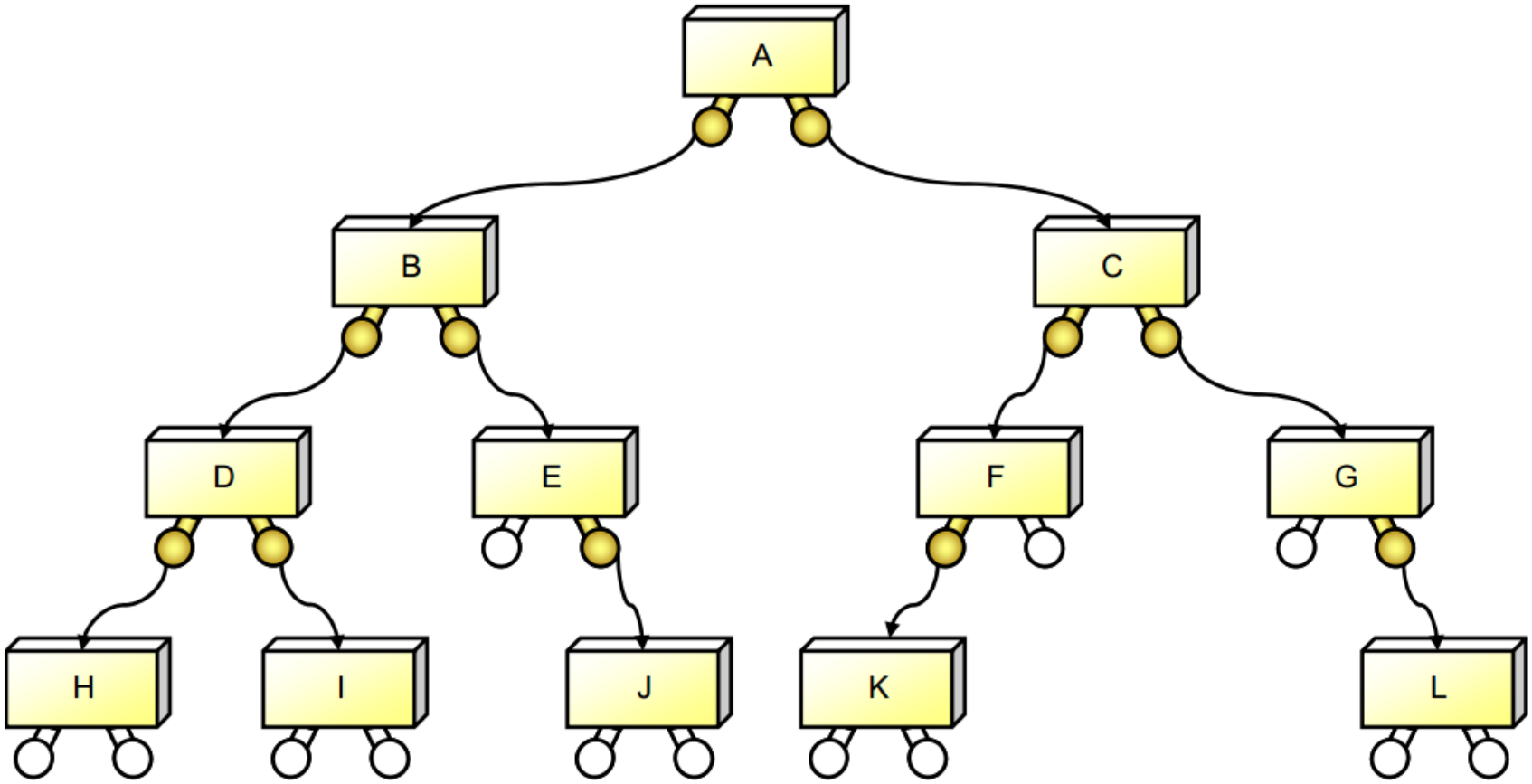
6.3.2. Biểu diễn cây nhị phân bằng cấu trúc liên kết

- Mỗi nút của cây là một cấu trúc gồm 3 trường:
  - Trường **Info**: Chứa giá trị lưu tại nút đó
  - Trường **Left**: Chứa liên kết (con trỏ) tới nút **con trái**
  - Trường **Right**: Chứa liên kết (con trỏ) tới nút **con phải**



6.3.2. Biểu diễn cây nhị phân bằng cấu trúc liên kết

- Để duyệt cây nhị phân, ta chỉ cần giữ lại **nút gốc**



6.4. Các thao tác với cây nhị phân

- Chèn một nút mới vào cây nhị phân
- Xóa một nút khỏi cây nhị phân
- Duyệt cây nhị phân
- Tìm kiếm trên cây
- …
- Giả sử nút của cây được biểu diễn bởi cấu trúc sau:

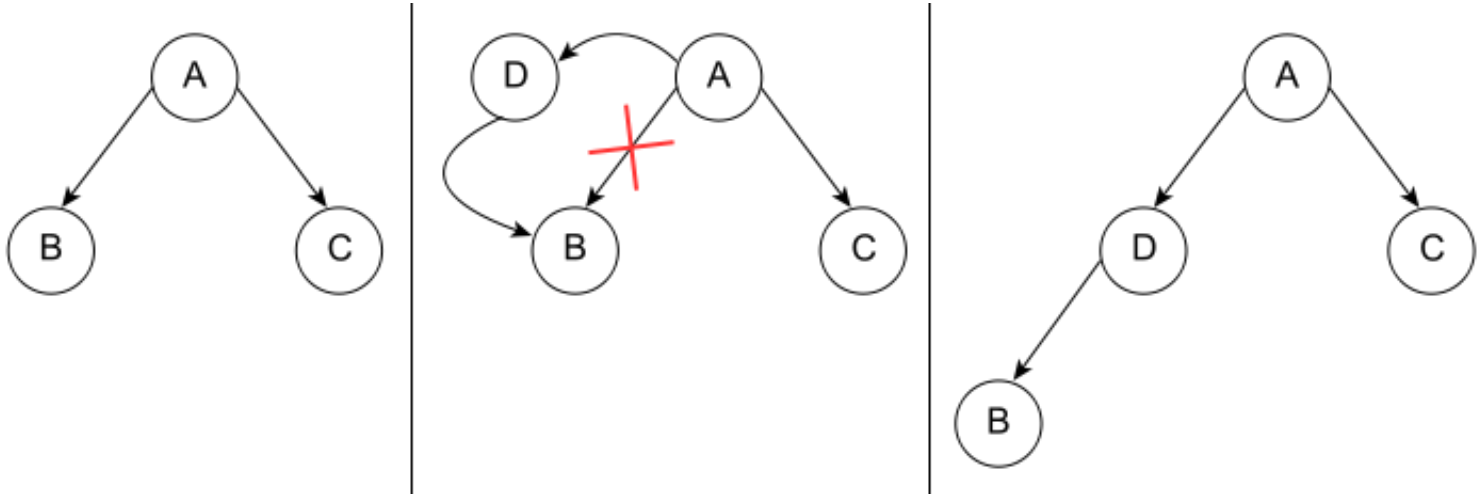
```
typedef struct node {
        int Info;
        TNode * Left;
        TNode * Right;
} TNode;
```

- Để thao tác trên cây, ta cần nắm được gốc:

```
TNode * root;
```

6.4.1. Chèn nút mới vào cây nhị phân

- Chèn nút mới vào làm nút con của nút p:
  - Tạo nút mới chứa giá trị cần chèn
    - `TNode * newNode = (TNode*) malloc(sizeof(TNode));`
    - `newNode->Info = value;`
    - `newNode->Left = newNode->Right = NULL;`
  - Chỉnh lại quan hệ cha-con giữa các nút



6.4.1. Chèn nút mới vào cây nhị phân

- Chèn nút mới vào làm nút con của nút p

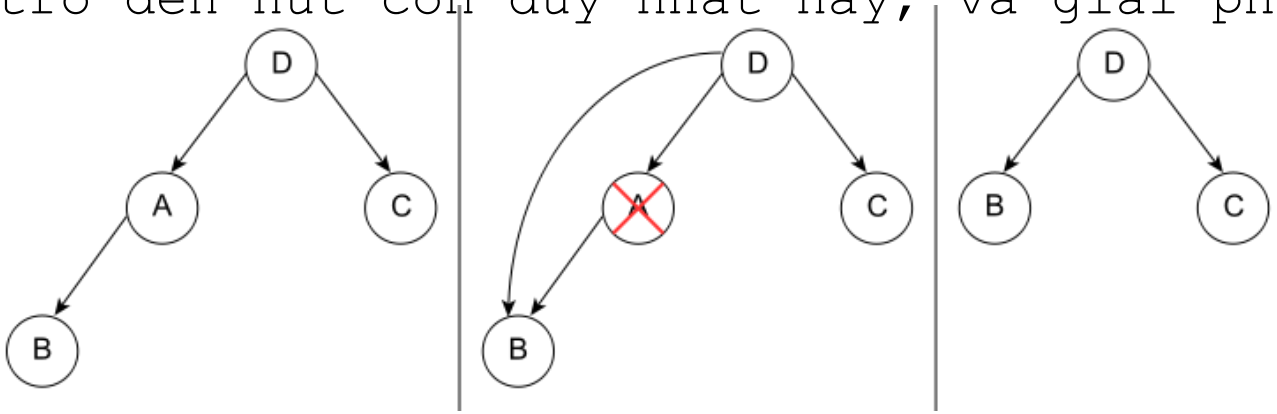
```
void Insert(TNode * p, int value) {
   <tạo nút mới newNode>
   if (p->Left == NULL) {
      p->Left = newNode;
   } else if (p->Right == NULL) {
      p->Right = newNode;
   } else {
      newNode->Left = p->Left;
      p->Left = newNode;
   }
}
```

- Chèn nút mới vào vị trí gốc của cây:

```
newNode->Left = root;
root = newNode;
```

6.4.2. Xóa nút khỏi cây nhị phân

- Xóa nút p khỏi cây nhị phân:
  - Nếu p là **nút lá** hoặc p chỉ có **1 nút con** thì rất đơn giản:
    - Nếu p là nút lá, chỉ cần giải phóng bộ nhớ bị chiếm bởi p, và gán liên kết tới nó = NULL
    - Nếu p có một nút con, chỉ cần chỉnh liên kết tới p trở đến nút con duy nhất này, và giải phóng p



  - Nếu p có **2 nút con**: không tồn tại cách xóa cụ thể, việc loại bỏ p và dồn các nút lại tùy thuộc vào bài toán cụ thể

6.5. Phép duyệt cây nhị phân

- Phép duyệt cây: là phép đi qua các nút trên cây một cách hệ thống, sao cho mỗi nút chỉ được xử lý đúng 1 lần
  - Duyệt theo thứ tự trước (preorder traversal): xử lý nút đang thăm trước, sau đó mới thăm các nhánh con
  - Duyệt theo thứ tự giữa (inorder traversal): thăm nhánh trái trước, rồi mới xử lý nút đang thăm, cuối cùng thăm nhánh phải
  - Duyệt theo thứ tự sau (postorder traversal): thăm nhánh trái trước, đến nhánh phải, cuối cùng mới xử lý nút đang thăm

6.5.1. Duyệt theo thứ tự trước (preorder traversal)

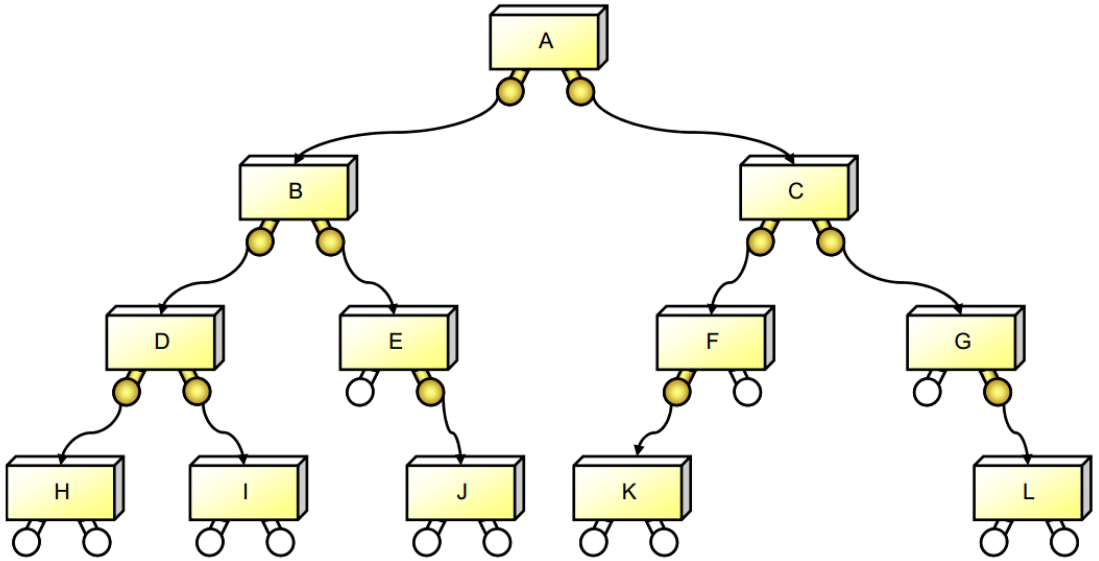
- Hàm đệ quy duyệt nhánh nhận ***node*** làm gốc

```
void Visit(TNode * node) {
  if (node != NULL) {
    <Output node->Info>
    Visit(node->Left);
    Visit(node->Right);
  }
}
```

- Để duyệt toàn bộ cậy, ta gọi `Visit(root);`

6.5.1. Duyệt theo thứ tự trước (preorder traversal)

```
Visit(A)
    in A
    Visit(B)
        in B
        Visit(D)
            in D
            Visit(H)
                in H
                Visit(NULL)
                Visit(NULL)
            Visit(I)
                in I
                Visit(NULL)
                Visit(NULL)
        Visit(E)
            in E
            Visit(NULL)
            Visit(J)
                in J
                Visit(NULL)
                Visit(NULL)
    Visit(C)
        in C
        … → F K G L
```



A B D H I E J C F K G L

6.5.1. Duyệt theo thứ tự giữa (inorder traversal)

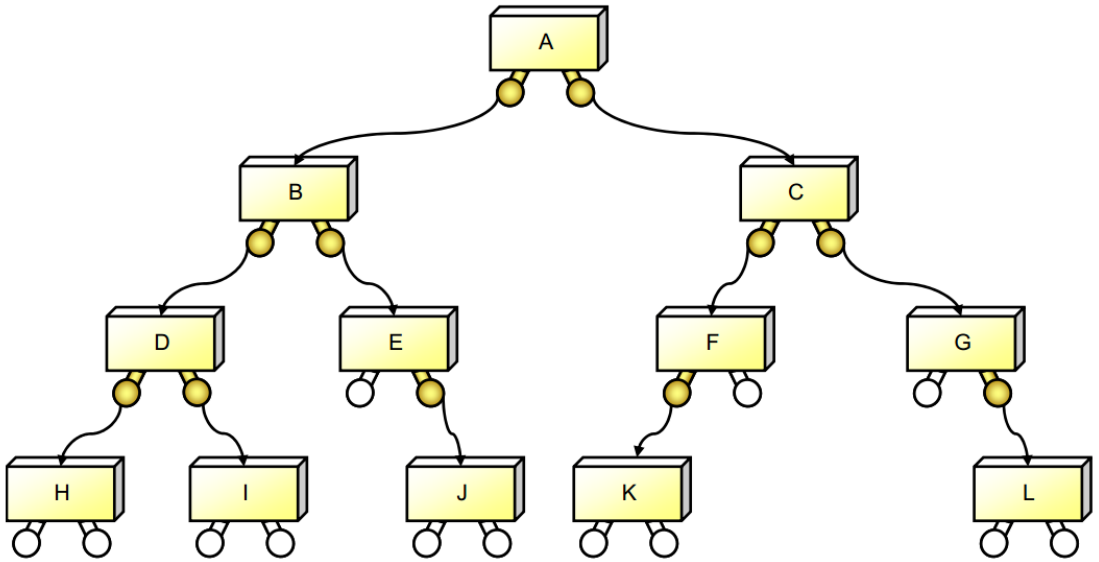
- Hàm đệ quy duyệt nhánh nhận ***node*** làm gốc

```
void Visit(TNode * node) {
  if (node != NULL) {
    Visit(node->Left);
    <Output node->Info>
    Visit(node->Right);
  }
}
```

- Để duyệt toàn bộ cậy, ta gọi `Visit(root);`

6.5.1. Duyệt theo thứ tự giữa (inorder traversal)

```
Visit(A)
   Visit(B)
      Visit(D)
         Visit(H)
            Visit(NULL)
            in H
            Visit(NULL)
         in D
         Visit(I)
            Visit(NULL)
            in I
            Visit(NULL)
      in B
      Visit(E)
         Visit(NULL)
         in E
         Visit(J)
            Visit(NULL)
            in J
            Visit(NULL)
   in A
   Visit(C)
      … → K F C G L
```



H D I B E J A K F C G L

6.5.1. Duyệt theo thứ tự sau (postorder traversal)

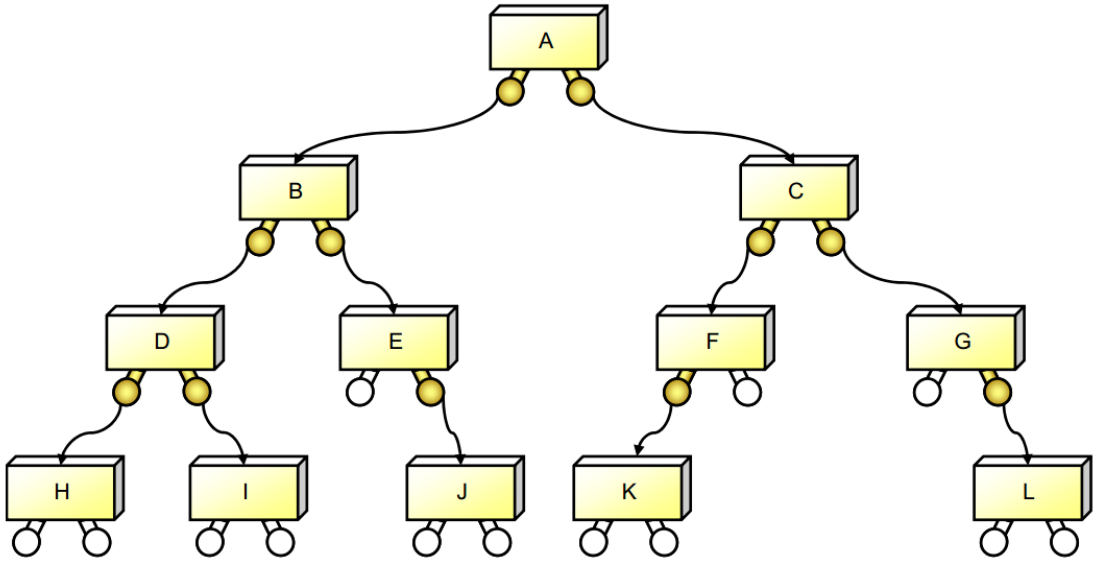
- Hàm đệ quy duyệt nhánh nhận ***node*** làm gốc

```
void Visit(TNode * node) {
  if (node != NULL) {
    Visit(node->Left);
    Visit(node->Right);
    <Output node->Info>
  }
}
```

- Để duyệt toàn bộ cậy, ta gọi `Visit(root);`

6.5.1. Duyệt theo thứ tự sau (postorder traversal)

```
Visit(A)
   Visit(B)
      Visit(D)
         Visit(H)
            Visit(NULL)
            Visit(NULL)
            in H
         Visit(I)
            Visit(NULL)
            Visit(NULL)
            in I
         in D
      Visit(E)
         Visit(NULL)
         Visit(J)
            Visit(NULL)
            Visit(NULL)
            in J
         in E
      in B
   Visit(C)
      … → K F L G
   in A
```



H I D J E B K F L G C A

6.6. Cây K-phân

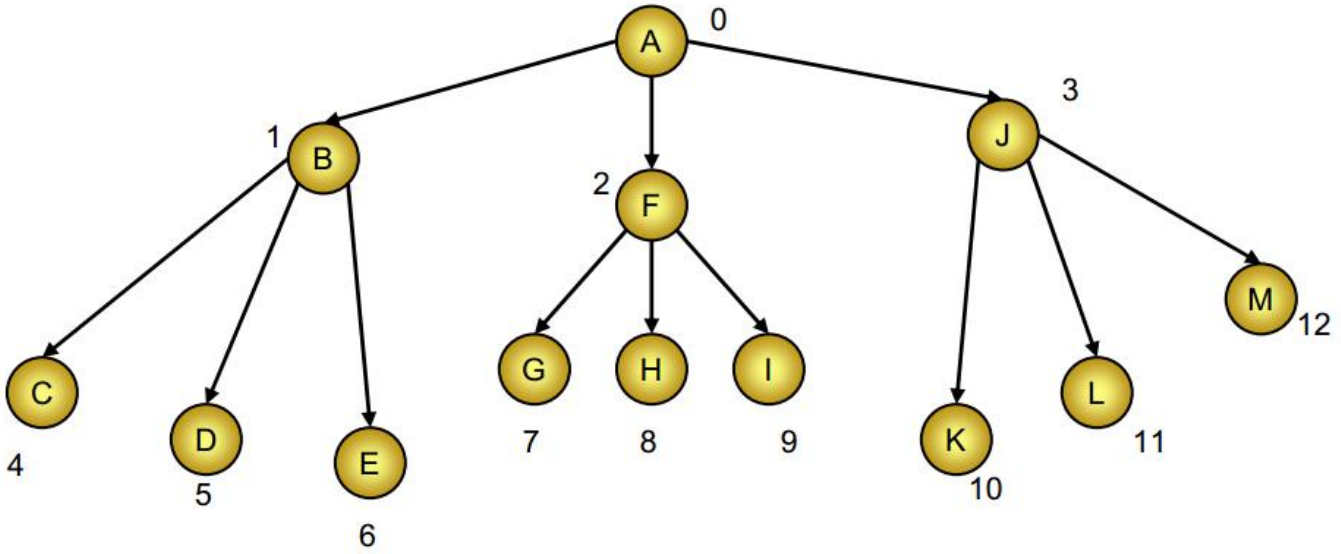
- Cây K-phân là một dạng cấu trúc cây mà mỗi nút trên cây có tối đa K nút con
- Tên gọi này bắt nguồn từ các hệ cơ số trong toán học: hệ nhị phân, hệ bát phân, hệ thập lục phân…
- 2 cách biểu diễn cây K-phân:
  - Biểu diễn bằng mảng
  - Biểu diễn bằng cấu trúc liên kết

6.6.1. Cây K-phân – biểu diễn bằng mảng

- Tương tự với cây nhị phân, ta có thể thêm vào cây K-phân một số **nút giả** để cho mỗi nút nhánh của cây K-phân đều có đúng K nút con:
  - các nút con được xếp thứ tự từ nút con thứ nhất tới nút con thứ K,
  - đánh số các nút trên cây K-phân từ 0, bắt đầu từ mức 1, và từ "trái qua phải" ở mỗi mức

6.6.1. Cây K-phân – biểu diễn bằng mảng

- Nút con thứ j của nút i là: i * K + j, với j = 1..K. Nút cha của nút x là nút (x-1) / K.
- Ta dùng mảng T đánh số từ 0 để lưu các giá trị trên các nút: Giá trị tại nút thứ i được lưu trữ ở phần tử T[i]



0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	F	J	C	D	E	G	H	I	K	L	M

6.6.2. Cây K-phân – biểu diễn bằng cấu trúc liên kết

- Khi biểu diễn cây K-phân bằng cấu trúc liên kết, mỗi nút của cây là một cấu trúc gồm hai trường:
  - Trường **Info**: Chứa giá trị lưu trong nút đó.
  - Trường **Links**: Là một **mảng** gồm K phần tử, phần tử thứ i (Links[i]) chứa liên kết (con trỏ) tới nút con thứ i.
- Để duyệt cây K- phân, chỉ cần giữ lại nút gốc

6.7. Cây tổng quát

- Là loại cây không có ràng buộc gì về số nút con của mỗi nút trên cây
- Xét 2 cách biểu diễn
  - Biểu diễn bằng mảng
  - Biểu diễn bằng cấu trúc liên kết

6.7.1. Cây tổng quát – biểu diễn bằng cấu trúc liên kết

- Khi lưu trữ cây tổng quát bằng cấu trúc liên kết, mỗi nút là một cấu trúc gồm ba trường:
  - Trường **Info**: Chứa giá trị lưu trong nút đó.
  - Trường **FirstChild**: Chứa liên kết (con trỏ) tới nút con đầu tiên (con cả)
  - Trường **Sibling**: Chứa liên kết (con trỏ) tới nút em kế cận bên phải (nút cùng cha với nút đang xét, khi sắp thứ tự các con thì nút đó đứng liền sau nút đang xét).