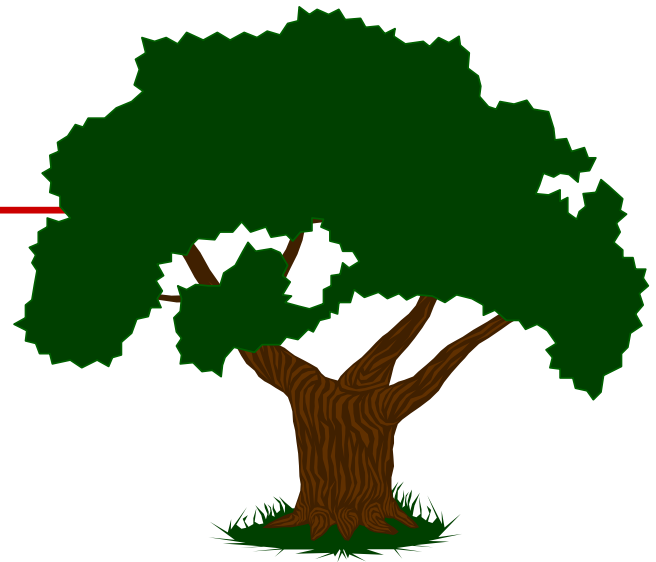
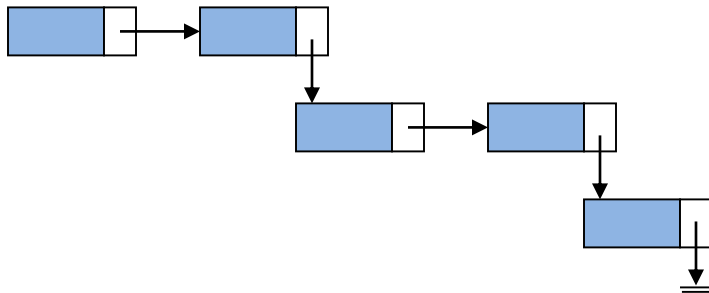
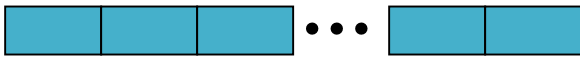


---

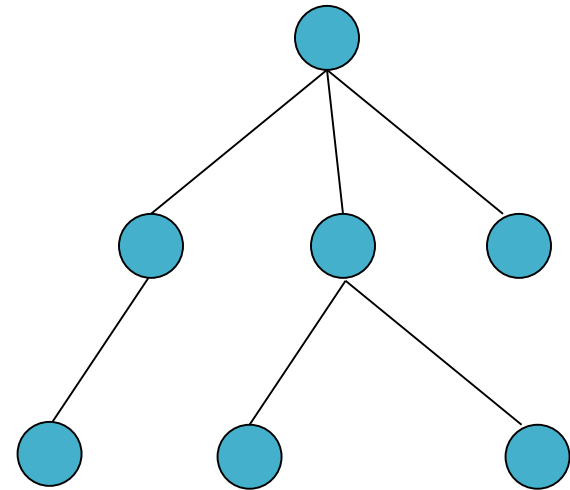
# Chương 6: Cấu trúc cây (Tree structure)



# Cấu trúc dữ liệu

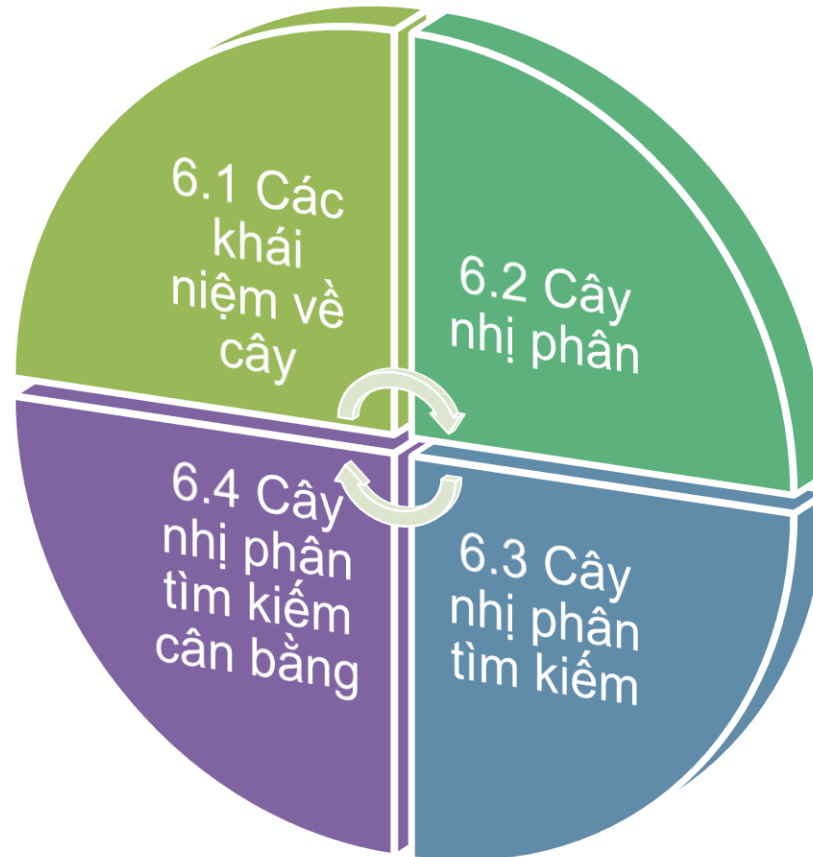


**Linear**



**Hierarchical structures**

# Nội dung



---

# Nội dung

## 6.1 Các khái niệm về cây

### 6.2 Cây nhị phân

### 6.3 Cây nhị phân tìm kiếm

### 6.4 Cây nhị phân tìm kiếm cân bằng

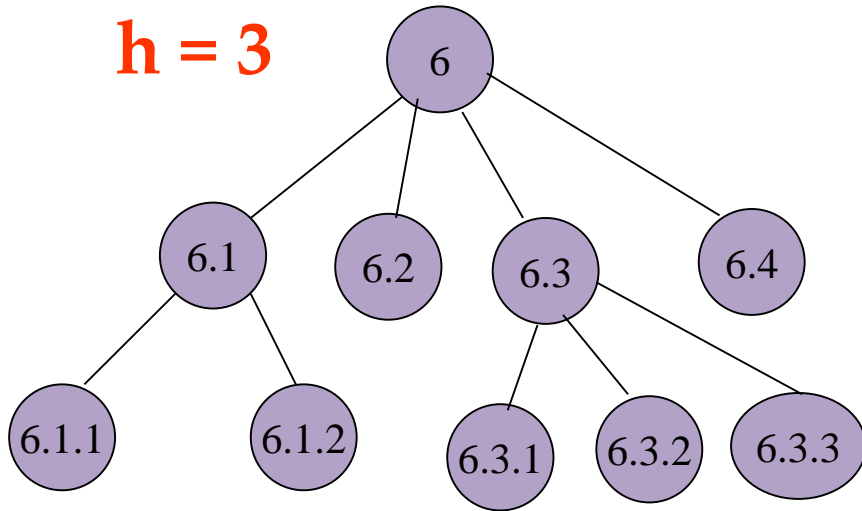
# 6.1 Các khái niệm về cây

## ■ Khái niệm cây:

- Cây là tập hợp các nút được tổ chức dưới dạng phân cấp
- Có một nút đặc biệt gọi là gốc (không có cha)
- Quan hệ giữa các nút trên cây là quan hệ cha-con và có dạng one – to – many.

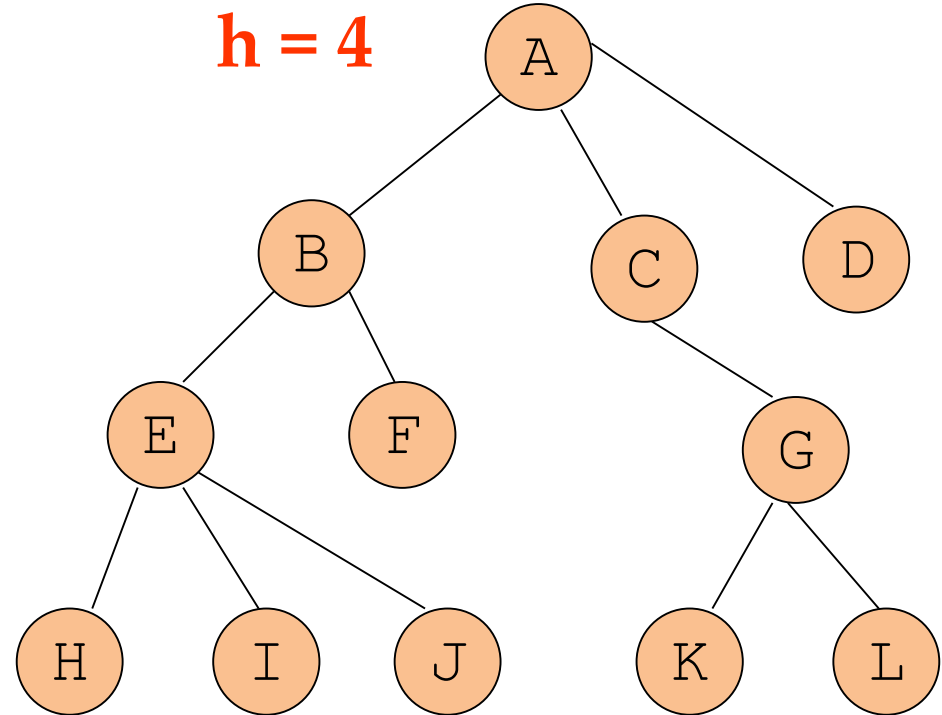
# 6.1 Các khái niệm về cây

$h = 3$



Cây mục lục

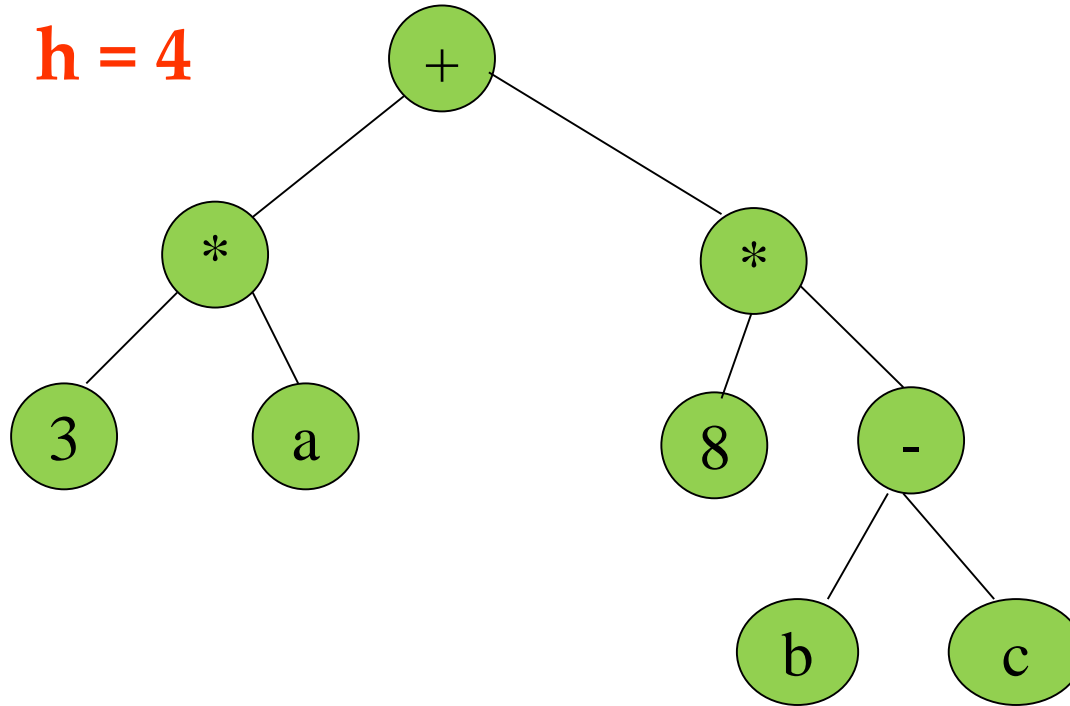
$h = 4$



Cây tập hợp

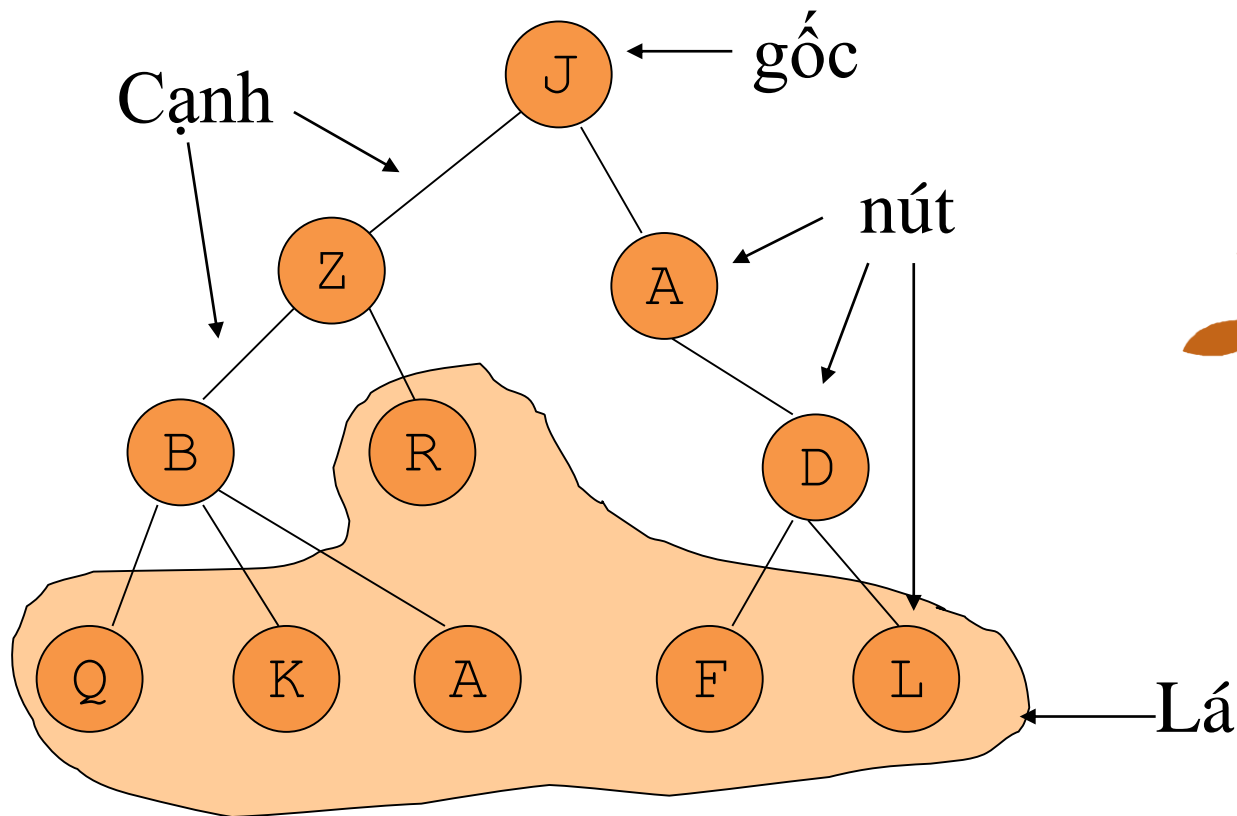
# 6.1 Các khái niệm về cây

$h = 4$



Cây biểu thức

# 6.1 Các khái niệm về cây

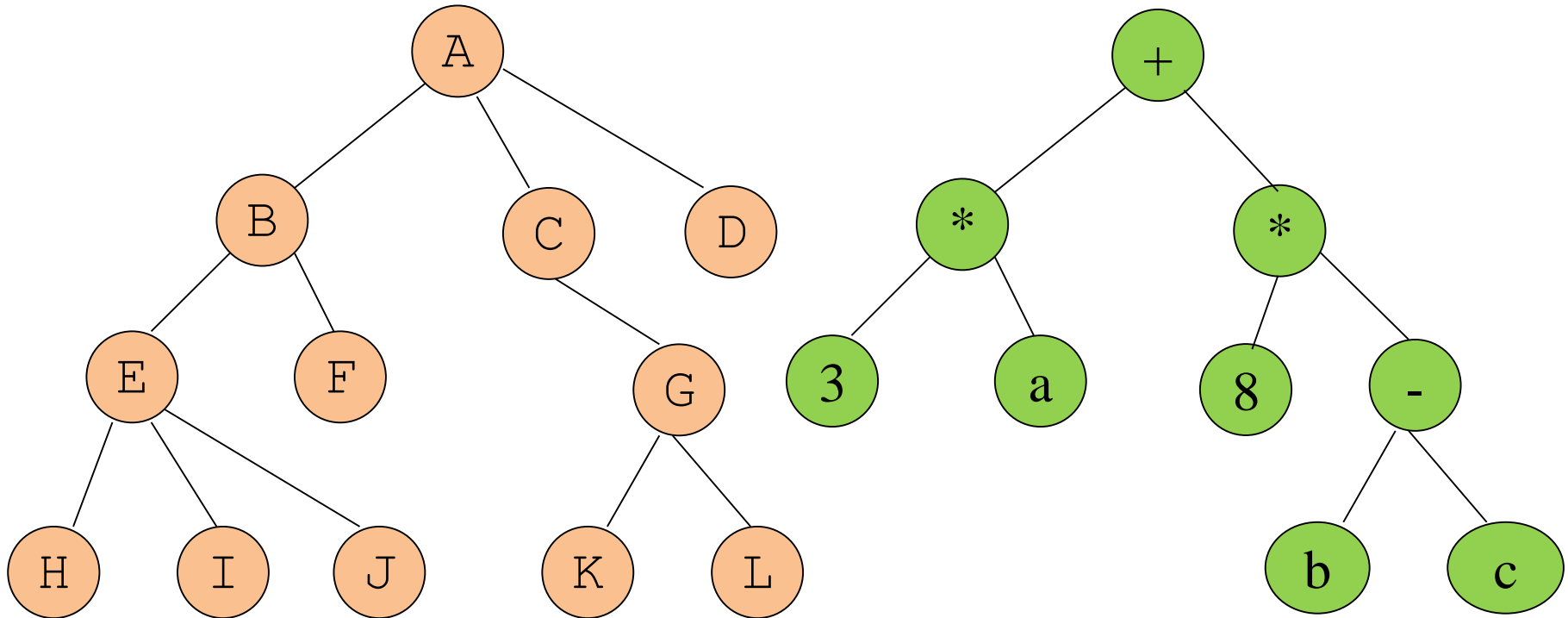




# 6.1 Các khái niệm về cây

## ■ Cấp (Degree):

- Cấp của nút là số con của nút đó
- Cấp của cây là cấp lớn nhất của nút trên cây

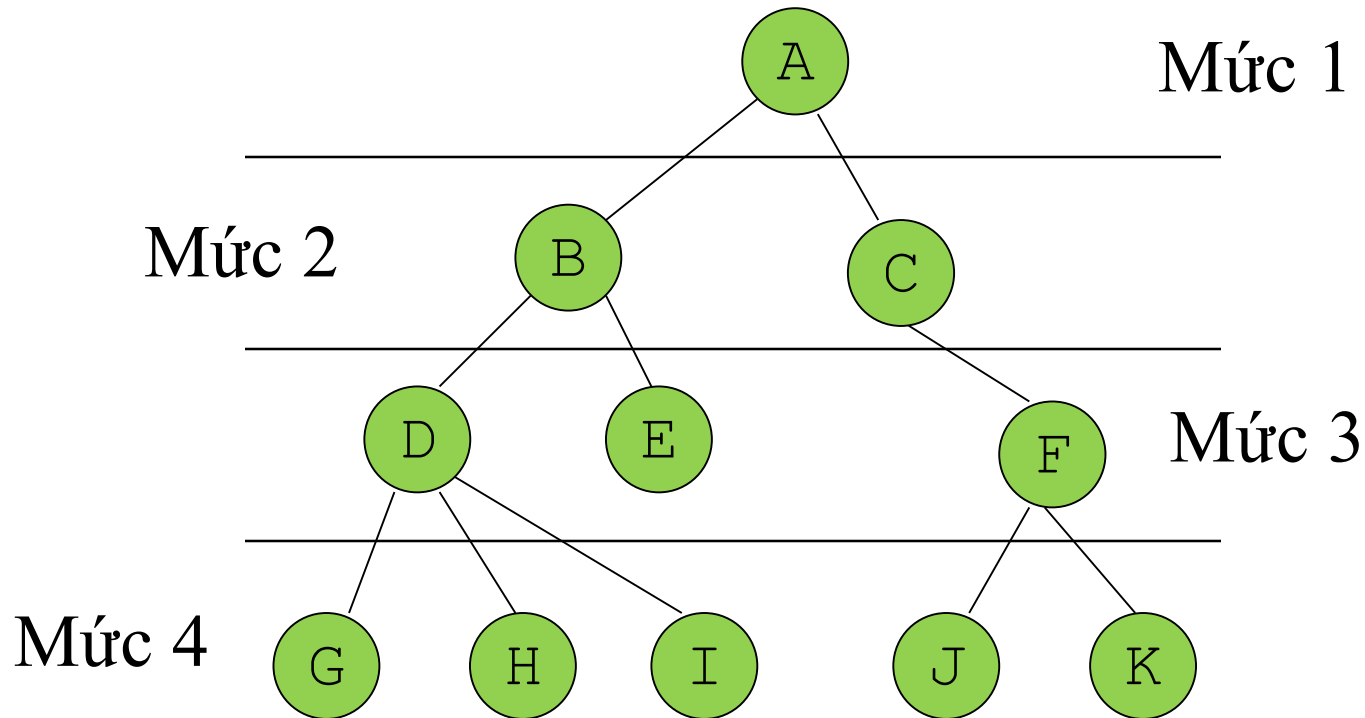


# 6.1 Các khái niệm về cây

■ Cấp (Degree)

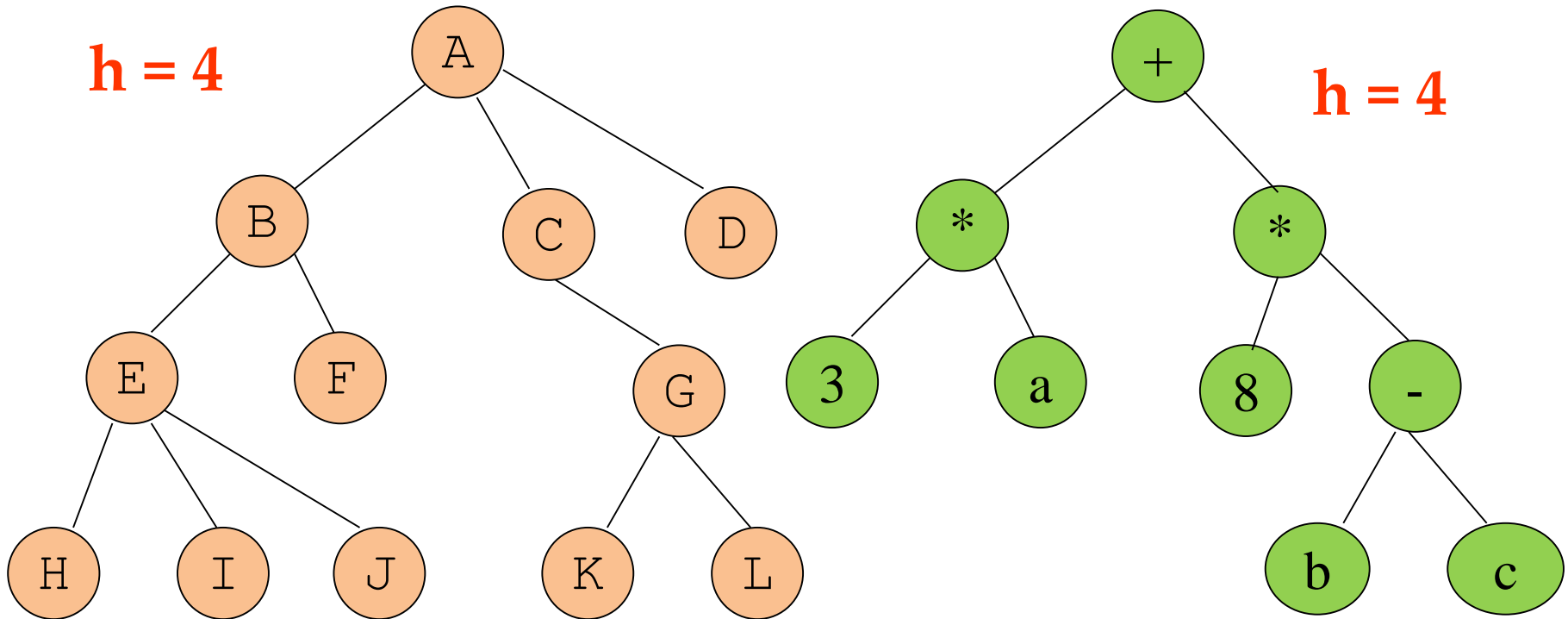
■ Mức (Level)

- Nút gốc có mức là 1
- Nếu nút cha có mức là  $i$  thì nút con có mức là  $i + 1$



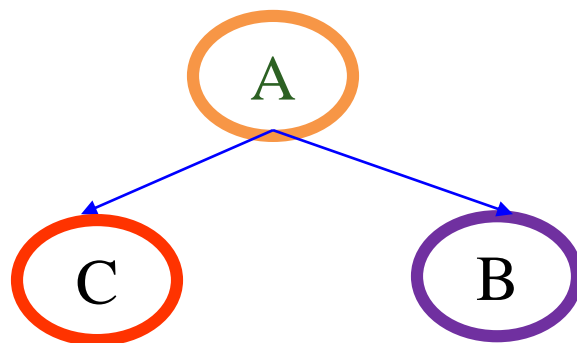
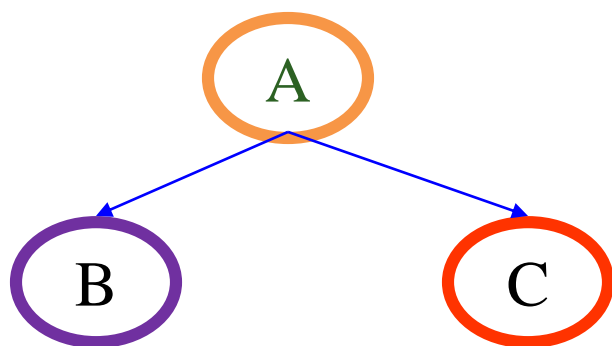
# 6.1 Các khái niệm về cây

- Cấp (Degree)
- Mức (Level)
- Chiều cao (Height): mức lớn nhất của nút trên cây



# 6.1 Các khái niệm về cây

- Cấp (Degree)
- Mức (Level)
- Chiều cao (Height): mức lớn nhất của nút trên cây
- Cây có thứ tự (Ordered Tree): là cây mà thứ tự các nút trên cây được coi trọng.
- Cây không có thứ tự (UnOrdered Tree): là cây mà thứ tự các nút trên cây không được coi trọng.



# 6.1 Các khái niệm về cây

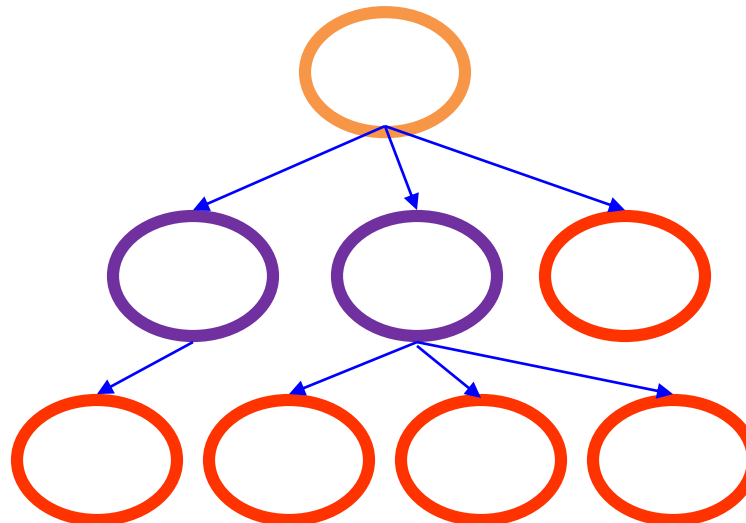
## ■ Thuật ngữ

- Nút gốc: không có nút cha
- Nút lá: không có nút con
- Nút trong: không phải nút gốc và nút lá

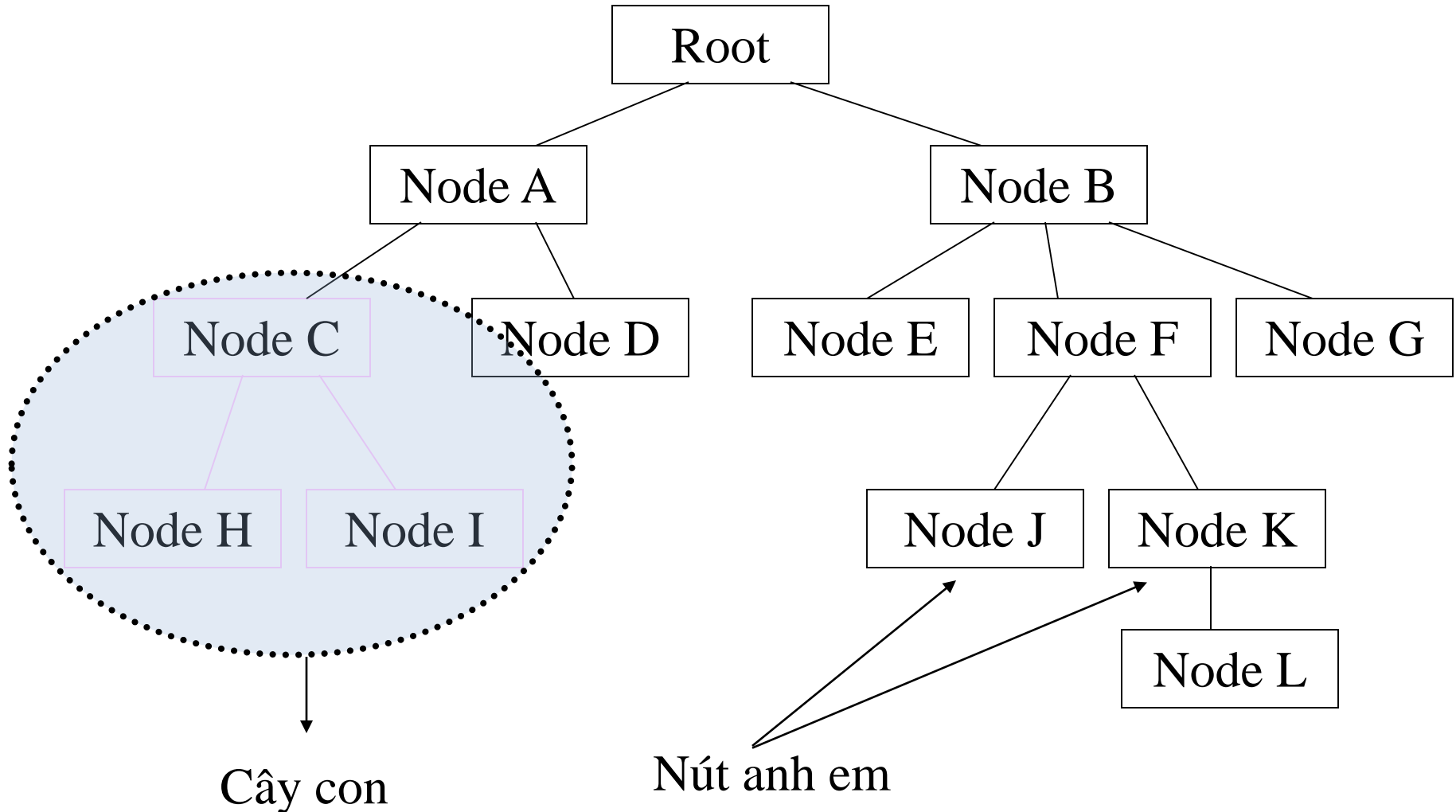
Nút gốc

Nút trong

Nút lá



# 6.1 Các khái niệm về cây



---

# Nội dung

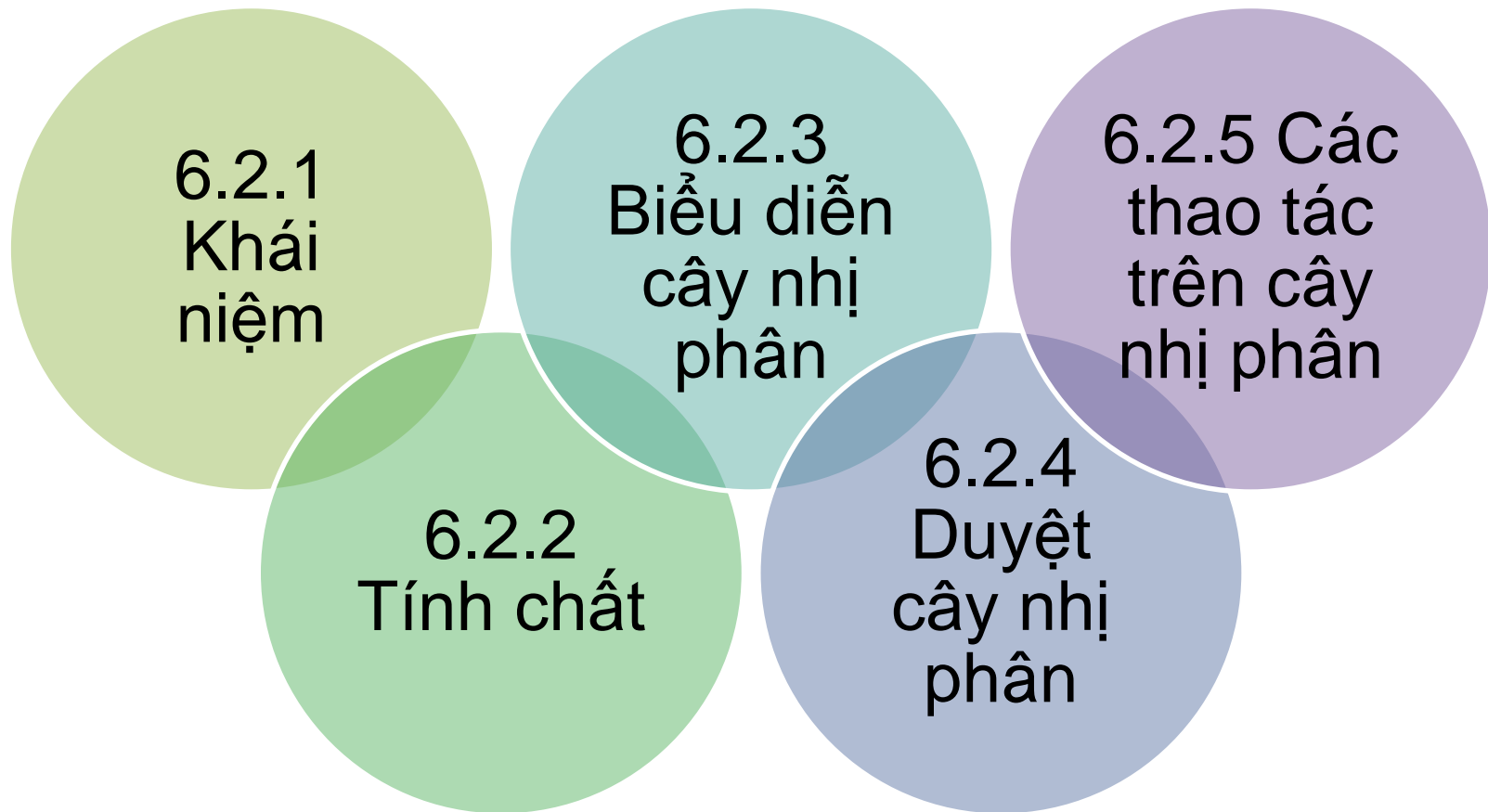
6.1 Các khái niệm về cây

**6.2 Cây nhị phân**

6.3 Cây nhị phân tìm kiếm

6.4 Cây nhị phân tìm kiếm cân bằng

## 6.2 Cây nhị phân

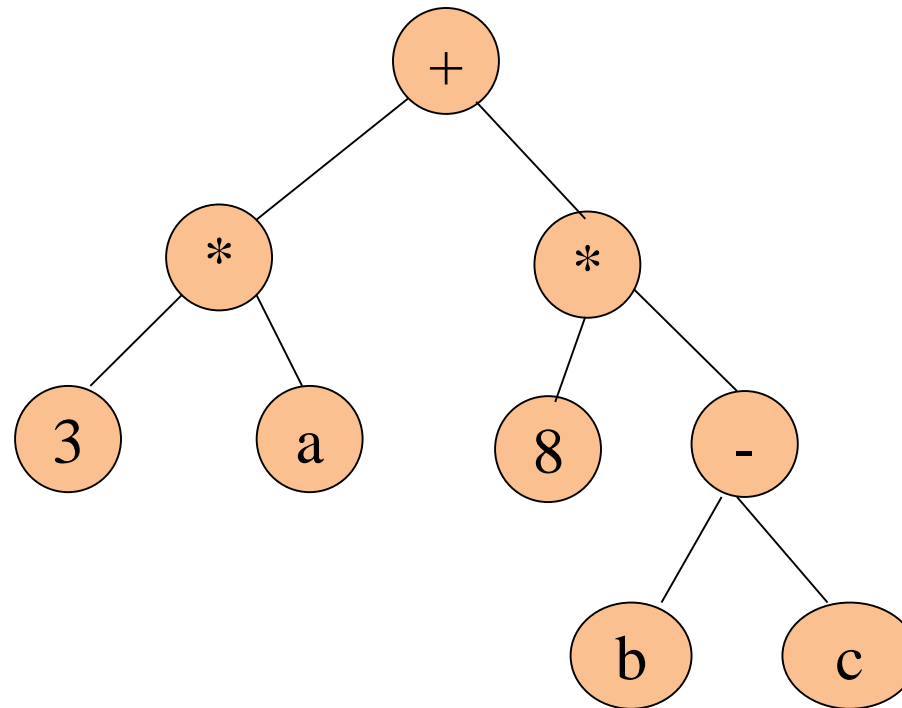




## 6.2.1 Khái niệm

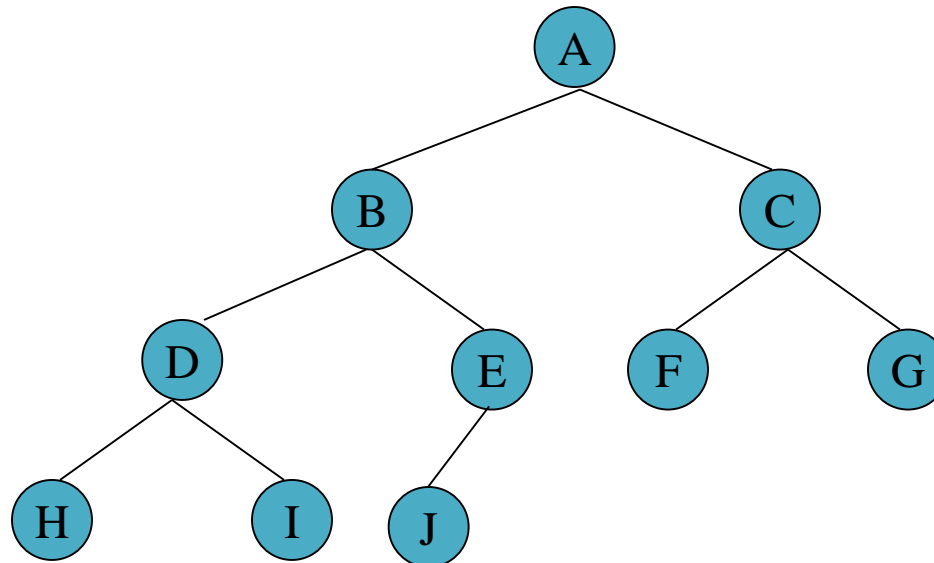
### ■ Khái niệm cây nhị phân

- Cây nhị phân là cây mà mỗi nút trên cây có tối đa là 2 con.
- Số con của 1 nút trên cây nhị phân  $\{0, 1, 2\}$



## 6.2.1 Khái niệm

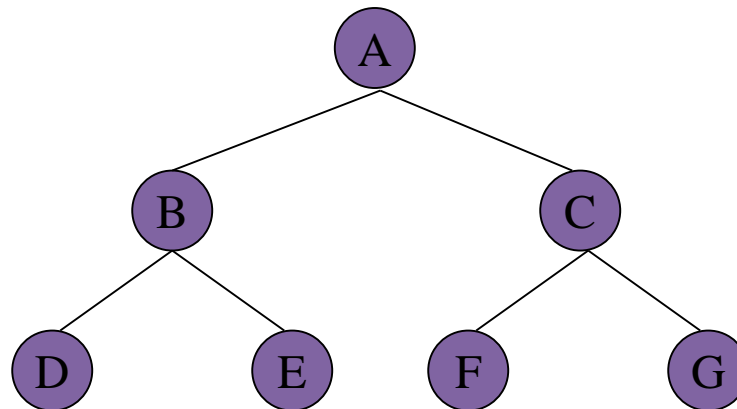
- Cây nhị phân hoàn chỉnh là cây nhị phân mà ở mức cuối còn thiếu các nút, nhưng các nút ở mức cuối đều dồn về bên trái



Cây nhị phân hoàn chỉnh

## 6.2.1 Khái niệm

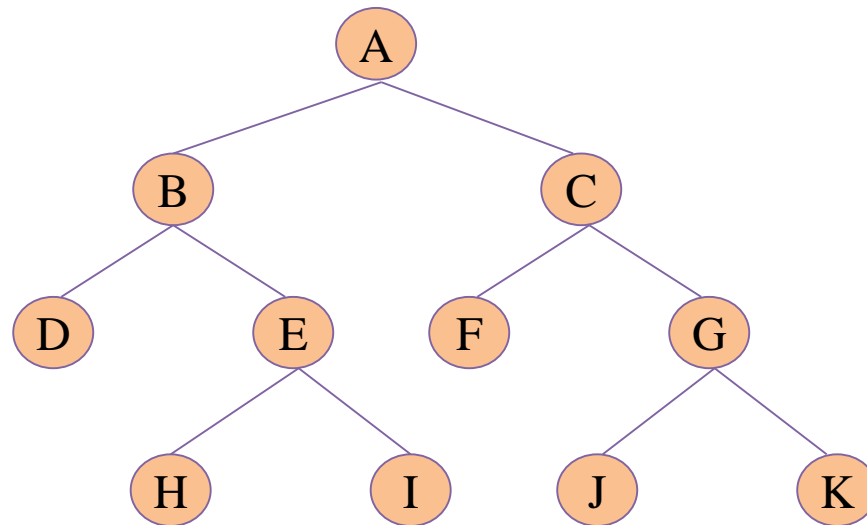
- Cây nhị phân hoàn chỉnh
- Cây nhị phân đầy đủ là cây nhị phân mà ở các mức đều đủ hết các nút



Cây nhị phân đầy đủ

## 6.2.1 Khái niệm

- Cây nhị phân hoàn chỉnh
- Cây nhị phân đầy đủ
- Cây nhị phân đúng là cây nhị phân mà nút gốc và các nút trong đều có đúng 2 con



**Cây nhị phân đúng**

## 6.2.2 Tính chất

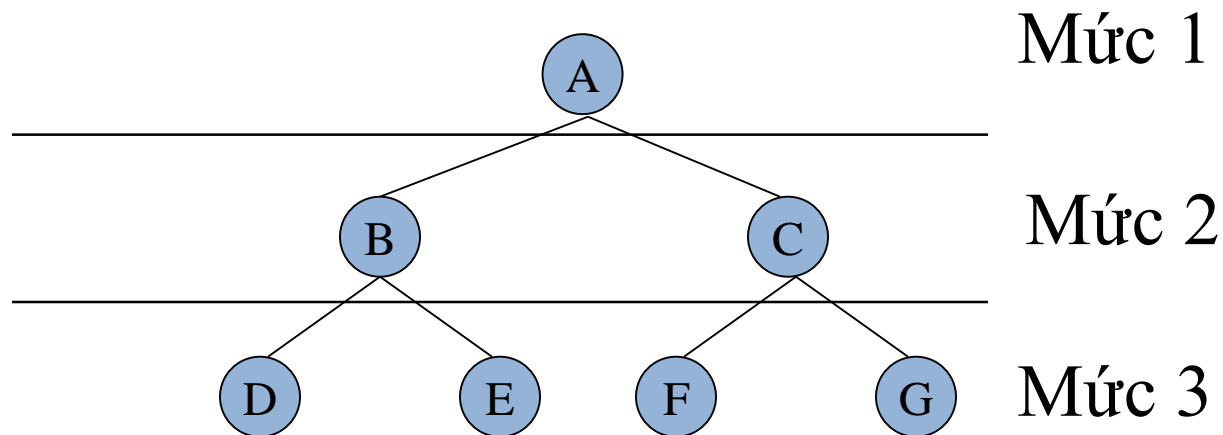
### ■ Tính chất của cây nhị phân

- Số nút tối đa của mức  $i$  trên cây nhị phân là  $2^{i-1}$
- Số nút tối đa trên cây nhị phân có chiều cao  $h$  là  $2^h - 1$
- Số nút tối đa trên cây nhị phân đúng có  $n$  nút lá là  $2*n - 1$

$$2^{1-1}=2^0=1$$

$$2^{2-1}=2^1=2$$

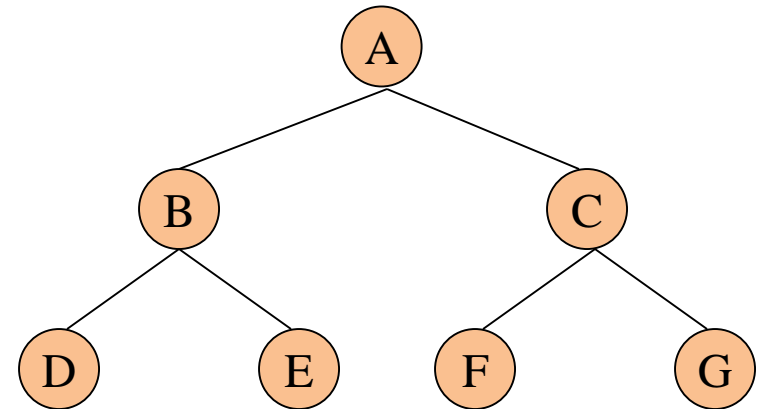
$$2^{3-1}=2^2=4$$



## 6.2.2 Tính chất

- Cây nhị phân đầy đủ với chiều cao  $h$ 
  - Phải là cây nhị phân đúng
  - Tất cả nút lá có chiều cao  $h$

Số nút trên cây NP đầy đủ có chiều cao  $h$  là  $(2^h - 1)$

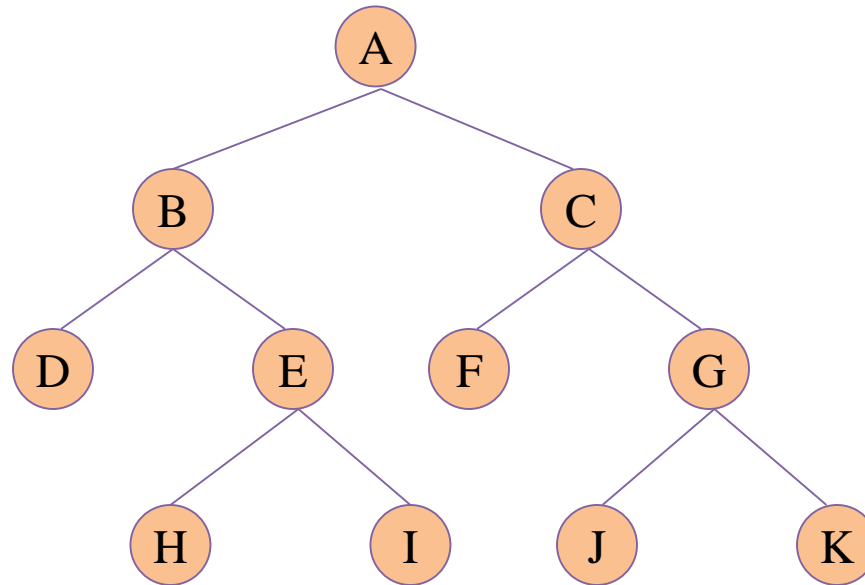


## 6.2.2 Tính chất

### ■ Cây nhị phân đúng:

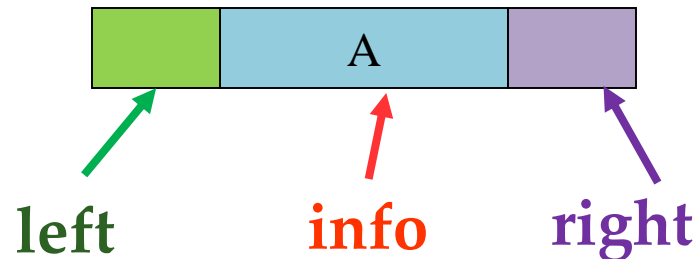
- Nút gốc và nút trung gian có đúng 2 con

### ■ Cây nhị phân đúng có $n$ nút lá thì số nút trên cây là $2n-1$



## 6.2.3 Biểu diễn cây nhị phân

- Lưu trữ cây nhị phân
  - Lưu trữ kế tiếp
  - Lưu trữ móc nối
- Lưu trữ móc nối: mỗi nút gồm các 3 thành phần
  - Phần data: chứa giá trị, thông tin...
  - Liên kết đến nút con trái (nếu có)
  - Liên kết đến nút con phải (nếu có)



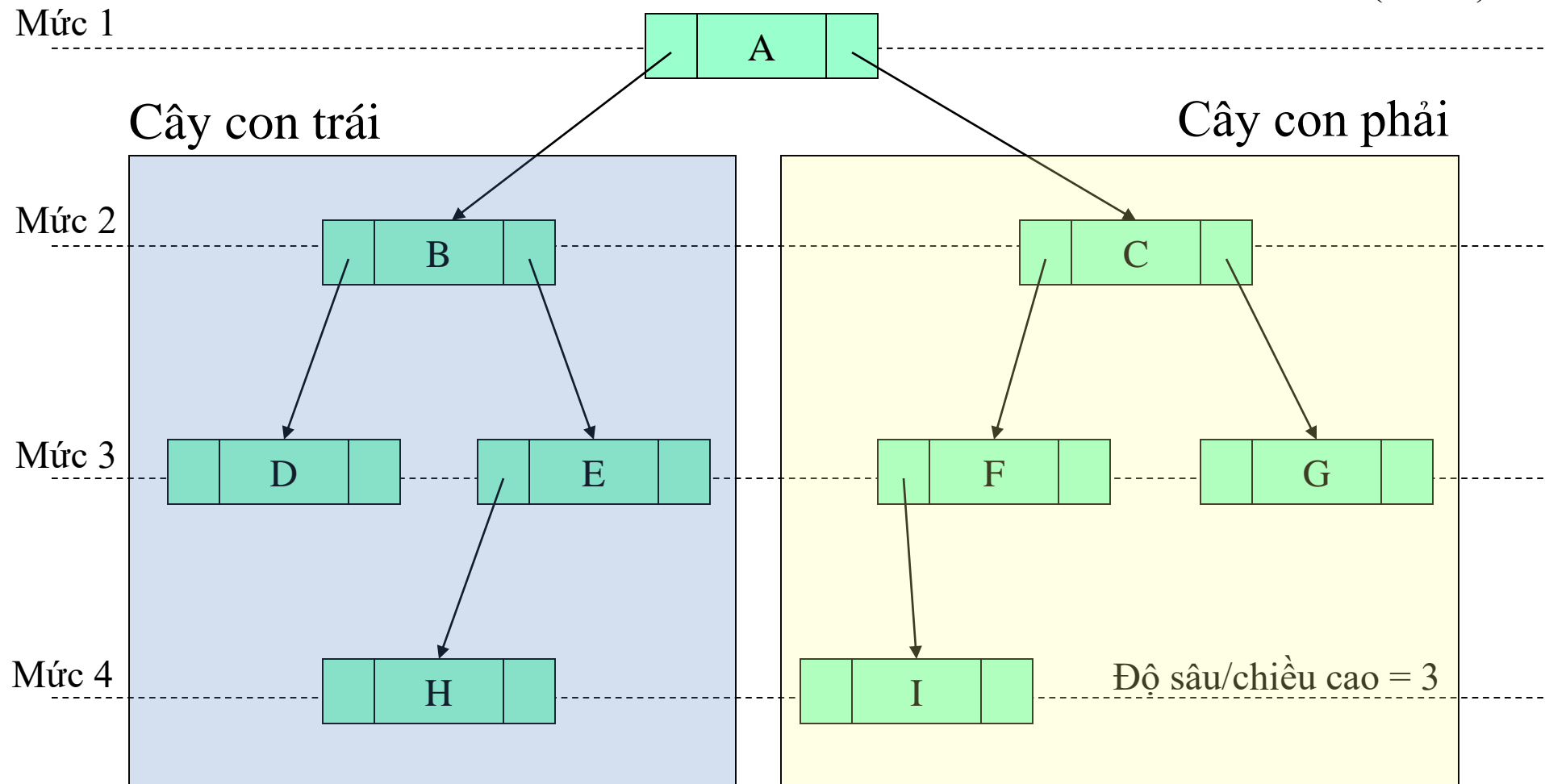


## 6.2.3 Biểu diễn cây nhị phân

- Cây nhị phân có thể rỗng (ko có nút nào)
- Cây NP khác rỗng có 1 nút gốc
  - Có duy nhất 1 đường đi từ gốc đến 1 nút
  - Nút không có nút con bên trái và con bên phải là nút lá

## 6.2.3 Biểu diễn cây nhị phân

Kích thước = 9 (số nút)



## 6.2 Cây nhị phân – Khai báo

- Khai báo cây: Cấu trúc của 1 nút và cây

```
typedef struct node
{
    DataType    info;
    node*       left;
    node*       right;
} NODE;
typedef NODE*  NodePtr;

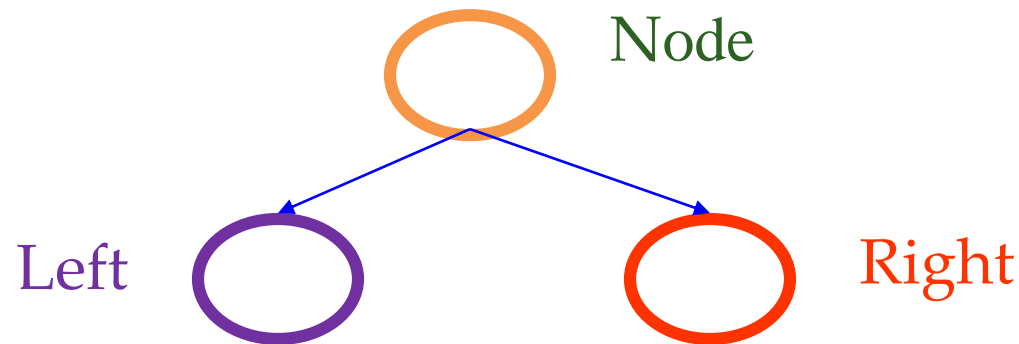
NodePtr      pTree;
```

pTree trỏ gốc của cây

## 6.2.4 Duyệt cây nhị phân

### ■ Duyệt cây nhị phân:

- Duyệt theo thứ tự trước: PreOrder **NLR**
- Duyệt theo thứ tự giữa: InOrder **LNR**
- Duyệt theo thứ tự sau: PostOrder **LRN**



## 6.2.4 Duyệt cây nhị phân

### ■ PreOrder - Duyệt cây theo thứ tự trước

```
void PreOrder (NodePtr    pTree)
{
    if (pTree != NULL)
    {
        cout<< pTree->info;
        PreOrder (pTree->left);
        PreOrder (pTree->right);
    }
}
```

## 6.2.4 Duyệt cây nhị phân

### ■ InOrder

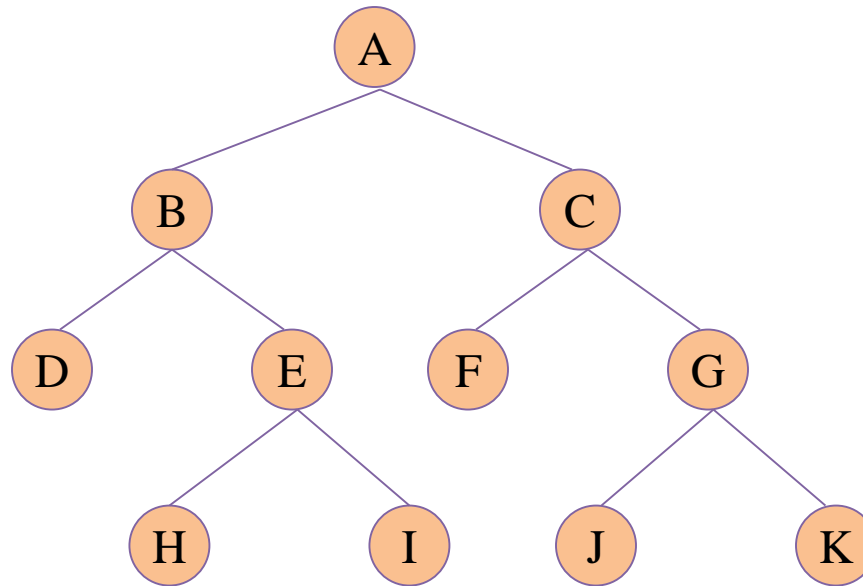
```
void InOrder (NodePtr pTree)
{
    if (pTree != NULL)
    {
        InOrder (pTree->left);
        cout<< pTree->info;
        InOrder (pTree->right);
    }
}
```

## 6.2.4 Duyệt cây nhị phân

### ■ PostOrder

```
void PostOrder (NodePtr pTree)
{
    if (pTree != NULL)
    {
        PostOrder (pTree->left);
        PostOrder (pTree->right);
        cout<< pTree->info;
    }
}
```

## 6.2.4 Duyệt cây nhị phân



NLR: ABDEHICFGJK

LNR: DBHEIAFCJGK

LRN: DHIEBFJKGCA



## 6.2.5 Các thao tác

- Các thao tác:
  - **CreateNode**, Init, IsEmpty
  - InsertLeft, InsertRight
  - DeleteLeft, DeleteRight
  - PreOrder, InOrder, PostOrder
  - Search
  - ClearTree



Phần minh  
hoạ dùng  
DataType là  
kiểu int

## 6.2.5 Các thao tác

### ■ InitTree: Khởi tạo cây nhị phân

```
void InitTree (NodePtr &pTree)
{
    *pTree = NULL;
}
```

### ■ IsEmpty: Kiểm tra cây có rỗng không

```
void IsEmpty (NodePtr &pTree)
{
    return (pTree == NULL) ;
}
```

## 6.2.5 Các thao tác

### ■ CreateNode: Tạo nút mới

```
1. NodePtr CreateNode (int x)
2. {
3.     NodePtr node;
4.     node = new NODE;
5.     node -> info = x;
6.     node -> left = NULL;
7.     node -> right = NULL;
8. }
```

## 6.2.5 Các thao tác

■ **InsertLeft**: bổ sung nút có khoá x làm con trái nút p

```
1. void InsertLeft(NodePtr p, int x)
2. { NodePtr node;
3.   if (p == NULL)           return;
4.   if (p -> left != NULL)   return;
5.   else
6.   {
7.       node = CreateNode(x);
8.       p -> left = node;
9.   }
10. }
```

## 6.2.5 Các thao tác

■ **InsertRight**: bổ sung nút có khoá x làm con phải nút p

```
1. void InsertRight(NodePtr p, int x)
2. { NodePtr node;
3.   if (p == NULL) return;
4.   if (p -> right != NULL) return;
5.   else
6.   {
7.       node = CreateNode(x);
8.       p -> right = node;
9.   }
10. }
```

## 6.2.5 Các thao tác

### ■ DeleteLeft: Loại bỏ con trái nút p

```
1. void DeleteLeft (NodePtr p)
2. {   NodePtr q = p->left;
3.     if (p == NULL)                return;
4.     if (p -> left == NULL)        return;
5.     else
6.         if (q -> left != NULL ||
              q -> right != NULL)    return;
7.         else
8.             {   delete      q;
9.                 p -> left = NULL;
10.            }
11. }
```

## 6.2.5 Các thao tác

### ■ DeleteRight: Loại bỏ con phải nút p

```
1. void DeleteRight(NodePtr p)
2. {   NodePtr q = p->right;
3.     if (p == NULL)                return;
4.     if (p -> right == NULL)        return;
5.     else
6.         if (q -> left != NULL ||
              q -> right != NULL)    return;
7.         else
8.             {   delete      q;
9.                 p -> right = NULL;
10.            }
11. }
```

## 6.2.5 Các thao tác

### ■ Search: tìm nút có khoá x

```
1. NodePtr Search(NodePtr pTree, int x)
2. {
3.     NodePtr p;      //con tro timkiem
4.     if (pTree == NULL) return NULL;
5.     if (pTree->info == x) return pTree;
6.     p = Search(pTree->left, x);
7.     if (p == NULL)
8.         p = Search(pTree->right, x);
9.     return p;
10.
11. }
```



## 6.2.5 Các thao tác

- ClearTree: xóa lần lượt nút bên trái, bên phải và nút cha.

```
1. void ClearTree (NodePtr  pTree)
2. {
3.     if (pTree != NULL)
4.     {
5.         ClearTree (pTree->left);
6.         ClearTree (pTree->right);
7.         delete      pTree;
8.     }
9. }
```

# Bài tập

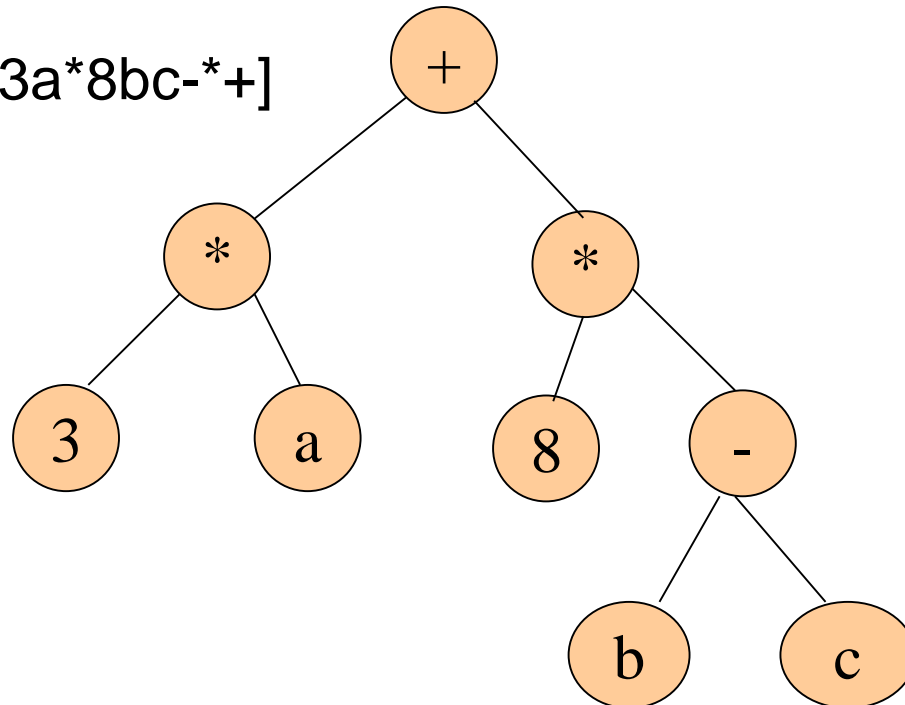
**Bài 1:** Cho cây nhị phân có nút gốc được trỏ bởi pTree, trường info chứa giá trị nguyên. Viết các giar thuật thực hiện các thao tác sau:

- Đếm số nút lá: CountLeaf
- Đếm số nút trên cây: CountNode
- Đếm số nút trong: CountInnerNode
- Xác định độ sâu/chiều cao của cây
- Tìm giá trị nhỏ nhất/lớn nhất trên cây
- Tính tổng giá trị các nút trên cây
- Đếm số nút có giá trị bằng x

# Bài tập

## ■ Bài 2: Viết dãy duyệt cây dưới đây:

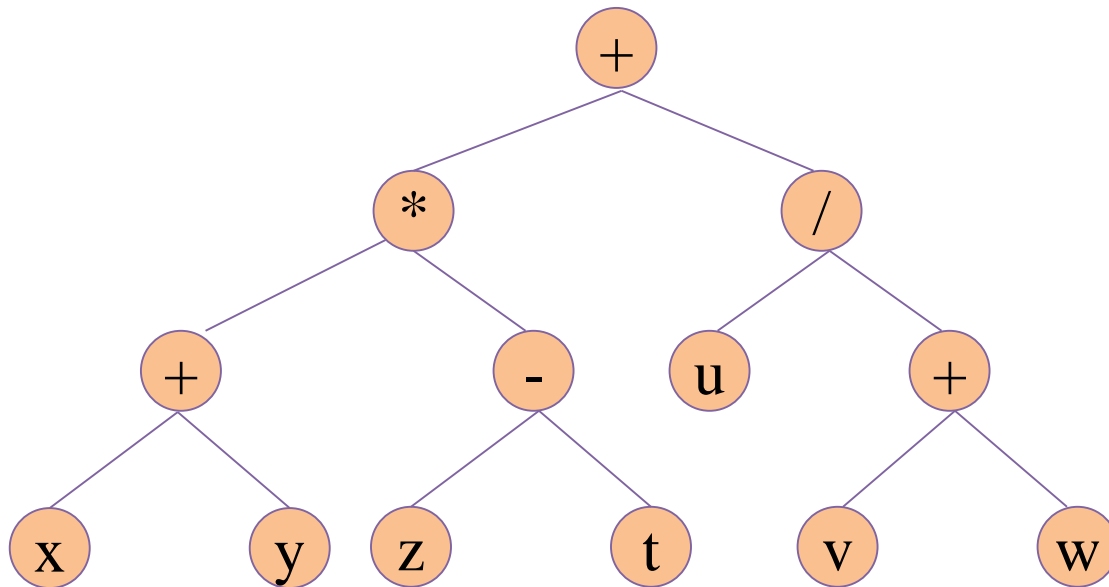
- Theo thứ tự trước  $[+*3a*8b-c]$
- Theo thứ tự giữa  $[3*a+8*b-c]$
- Theo thứ tự sau  $[3a*8bc-*+]$



# Bài tập

■ **Bài 3:** Cho biểu thức  $(x + y)^*(z - t) + u/(v + w)$

- Vẽ cây nhị phân biểu diễn biểu thức trên
- Viết dãy duyệt cây đó theo thứ tự trước và thứ tự sau



# Bài tập

## ■ Bài 4:

- Viết chương trình hoàn chỉnh thực hiện các thao tác trên cây nhị phân

# Tài liệu tham khảo

- [1]. Giáo trình Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh, NXB Đại học quốc gia TP HCM, 2013
- [2]. Cấu trúc dữ liệu & thuật toán, Đỗ Xuân Lôi, NXB Đại học quốc gia Hà Nội, 2010.
- [3]. Trần Thông Quế, *Cấu trúc dữ liệu và thuật toán (phân tích và cài đặt trên C/C++)*, NXB Thông tin và truyền thông, 2018
- [4]. Robert Sedgewick, *Cẩm nang thuật toán*, NXB Khoa học kỹ thuật, 2004 .
- [5]. PGS.TS Hoàng Nghĩa Tý, *Cấu trúc dữ liệu và thuật toán*, NXB xây dựng, 2014

