# 数据结构与算法

人工智能与大数据学院

## 本节课主要内容

- 第5章 树

① 掌握树和二叉树的递归定义
② 掌握树和二叉树的相关术语和概念
③ 掌握二叉树的基本性质
④ 掌握二叉树的存储结构
⑤ 掌握二叉树的基本操作

## 树的定义

1、树的定义

树是由n(n>=0)个结点组成的有限集合。

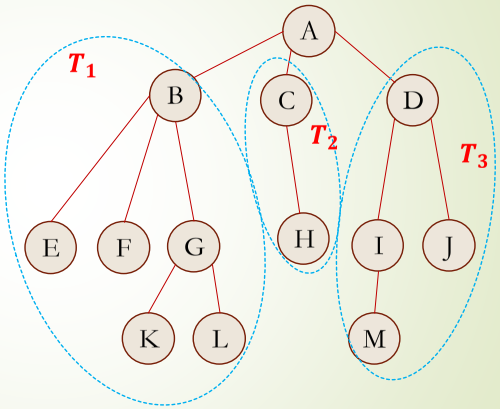
当n=0时，称这棵树为空树；

当n>0时，称这棵树为非空树；
在任一非空树中：
① 必有一个特定的称为根的结点
② 剩下的结点被分成m>=0个互不相交的集合T1,T2…，而且这些集合中的每一个又都是树

## 树的定义

树的特点：

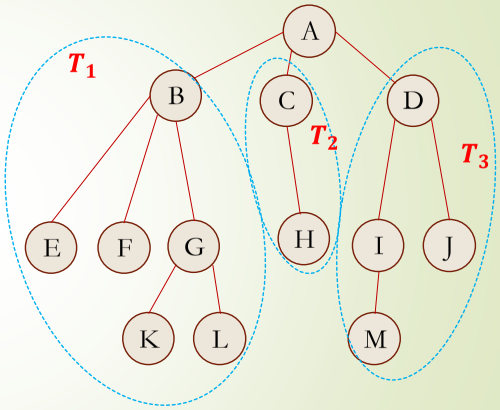
① 树的根结点没有双亲，除根结点以外，其他每个结点都有且仅有一个双亲。
② 树中所有结点都有零个或多个孩子结点。
③ 树是一种一对多的层次结构



## 森林的定义

森林是n(n>=0)棵互不相交的树的集合。

在数据结构中，树和森林的差异很小，一棵树是森林，任何一棵树删除根结点后剩下的部分也是森林。




## 二叉树的定义

二叉树是n(n>=0)个具有相同类型的数据元素（结点）的有限集合，这个集合或者是空的，或者是由一个根结点或两棵互不相交的称为左子树和右子树的二叉树组成。

```
typedef struct TreeNode {
    int value;              // 结点（节点）的值
    struct TreeNode *left;  // 左子结点（节点）
    struct TreeNode *right; // 右子结点（节点）
} TreeNode;
```

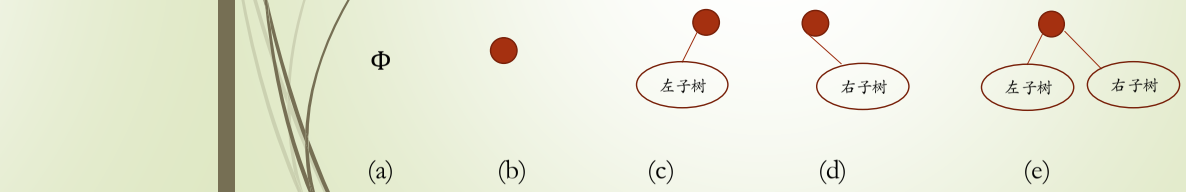
## 二叉树的定义

当n=0时，称这棵二叉树为**空二叉树**；

当n>0时，数据元素分为：一个称为**根**的数据元素(Root)和两个分别称为**左子树**和**右子树**的数据元素的集合，左、右子树互不相交，并且它们也都是二叉树。

在二叉树中，一个数据元素也称作一个结点。
二叉树的子树是有序的，若将其左右子树颠倒，就成为另一棵不同的二叉树。即使只有一棵子树，也要明确指出它是左子树还是右子树。

由于左右子树的有序性，二叉树具有五种基本形态，即***空二叉树、仅有根结点的二叉树、右子树为空的二叉树、左子树为空的二叉树和左右子树均非空的二叉树***



## 构造一棵二叉树——创建结点

```
TreeNode* createNode(int value) {
    TreeNode* node = (TreeNode*)malloc(sizeof(TreeNode));
    if (node == NULL) {
        printf("Error\n");
        exit(EXIT_FAILURE);
    }
    node->value = value;
    node->left = NULL;
    node->right = NULL;
    return node;
}
```

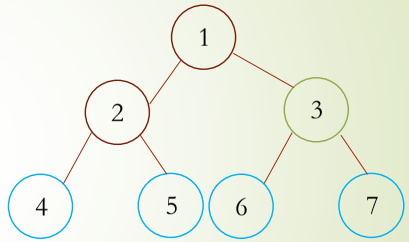
## 构造一棵二叉树——初始化一棵二叉树

```
TreeNode* createBinaryTree() {
    // 创建根节点
    TreeNode* root = createNode(1);

    // 创建左子树
    root->left = createNode(2);
    root->left->left = createNode(4);
    root->left->right = createNode(5);

    // 创建右子树
    root->right = createNode(3);
    root->right->left = createNode(6);
    root->right->right = createNode(7);

    return root;
}
```

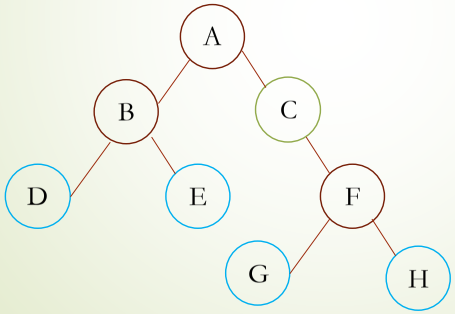



## 二叉树的相关概念

（1）结点的度

**结点所拥有的子树的数量称为该结点的度。**

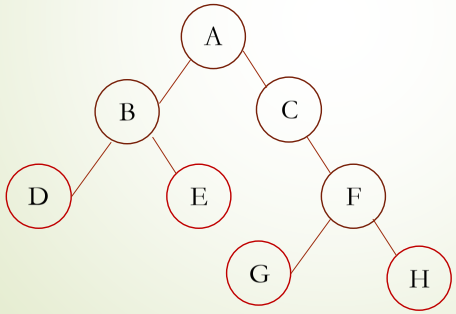
**在二叉树中，结点的度最大为2。**




## 二叉树的相关概念

（2）叶子

**在二叉树中，度为0的结点称为叶子。**



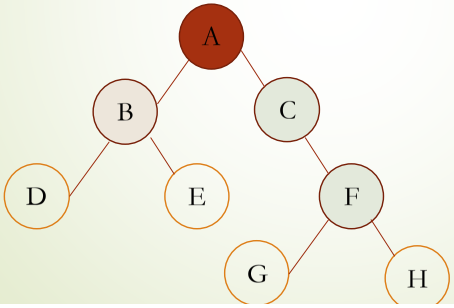
## 二叉树的相关概念

（3）孩子

结点的子树的根称为该结点的孩子。在二叉树中，孩子有左右之分，分别称为左孩子和右孩子。

（4）双亲

孩子结点的上层结点称为该孩子结点的双亲。



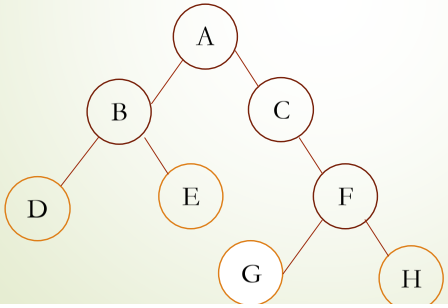

## 二叉树的相关概念

（5）子孙

以某结点为根的子树中的任一结点都称为该根结点的子孙。

（6）祖先

结点的祖先是从根结点到该结点所经分支上的所有结点。



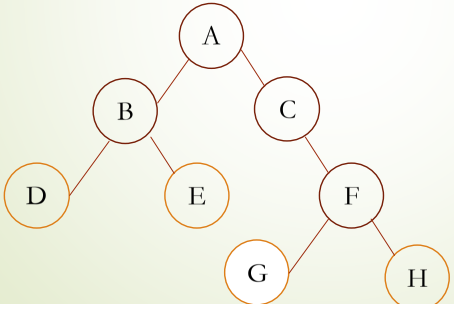

## 二叉树的相关概念

（7）结点的层次

从根结点起，根为第一层，它的孩子为第二层，孩子的孩子为第三层，依次类推，设某个结点的层次是L，则它的孩子第层次就为L+1。

（8）兄弟：同一双亲的孩子互为兄弟。

（9）堂兄弟：其双亲在同一层的结点互为堂兄弟。



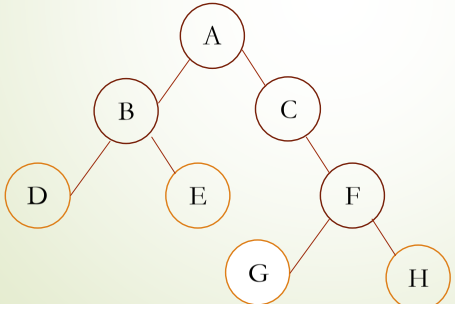

## 二叉树的相关概念

（10）二叉树的度

二叉树中最大的结点度数称为二叉树的度。下图二叉树的度为2，空二叉树和只有根结点的二叉树的度为0。

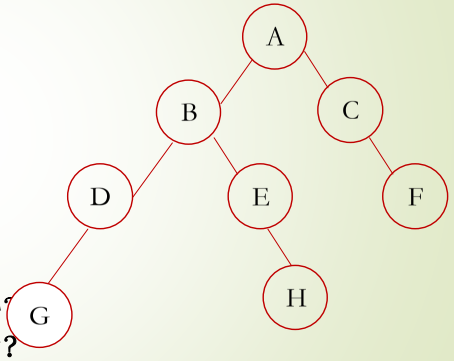
（11）二叉树的深度

二叉树中结点的最大层次数定义为二叉树的深度。下图二叉树的深度为4



分析如下几个问题：

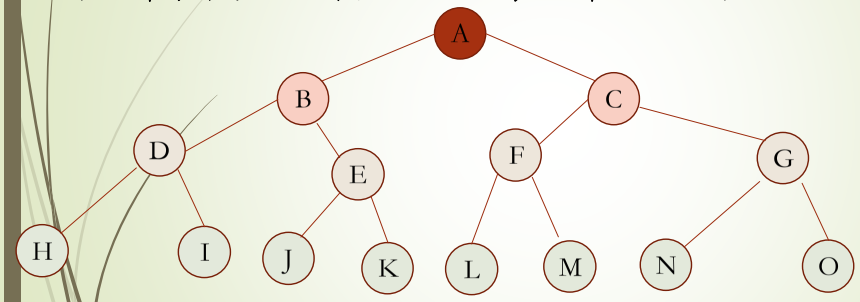
① 结点E的度是多少？
② 二叉树中的叶子结点有哪些？
③ 结点B的孩子是哪个结点？
④ 结点E的双亲是哪个结点？
⑤ 结点B的子孙有哪些结点？
⑥ 结点H的祖先有哪些结点？
⑦ 结点G是第几层次？
⑧ 满足兄弟关系的结点有哪些？
⑨ 满足堂兄弟关系的结点有哪些？
⑩ 二叉树的度和深度分别是多少？



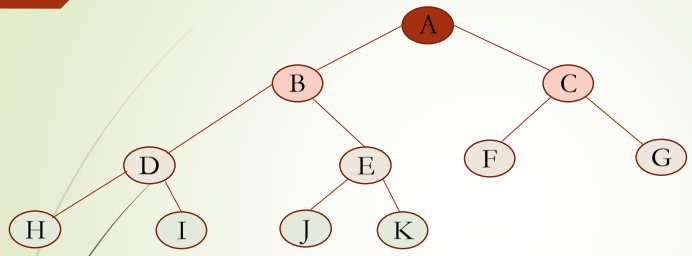

## 二叉树的相关概念

(12) **满二叉树**

在一棵二叉树中，如果所有分支结点都存在左子树和右子树，并且所有叶子结点都在同一层上，这样的二叉树称为满二叉树。

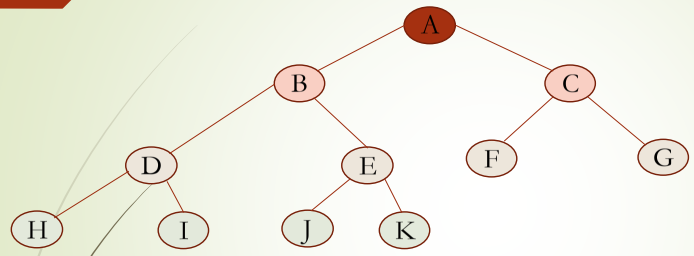



## 二叉树的相关概念




1、结点K的祖先包括哪些结点？
2、结点B的子孙包括哪些结点？
3、请列出该二叉树中属于堂兄弟关系的结点？
4、怎么样把这个二叉树变成满二叉树？

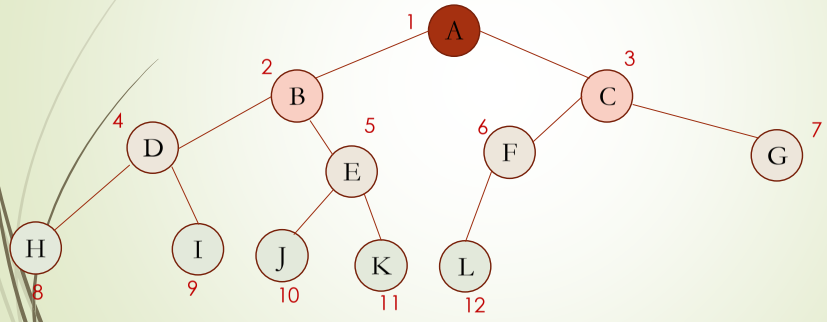
## 二叉树的相关概念




1、结点K的祖先包括哪些结点？ ABE
2、结点B的子孙包括哪些结点？ DEHIJK
3、请列出该二叉树中属于堂兄弟关系的结点？
DF/DG/EF/EG/HJ/HK/IJ/IK
4、怎么样把这个二叉树变成满二叉树？
给结点F补充左右孩子，给结点G补充左右孩子

（13）**完全二叉树**

**对深度为k的满二叉树中的结点按照从上至下，从左至右的顺序从1开始连续编号。对一棵具有n个结点，深度为k的二叉树，采用同样办法对树中结点进行编号。对于编号i的结点，其位置与满二叉树中的位置一致，则称此二叉树为一棵完全二叉树。**




## 二叉树的相关概念

（13）**完全二叉树**

- 对于一棵完全二叉树，叶子结点只可能出现在最下层和倒数第二层，而且最下层的叶子集中在树的最左部
- 一棵满二叉树一定是一棵完全二叉树，但是，一棵完全二叉树未必是一棵满二叉树

## 二叉树——基本性质

**【性质1】 一棵非空二叉树的第$i$层最多有$2^{i-1}$个结点。**

性质1表明的是二叉树每一层能拥有的最大结点数。

证明：对于非空二叉树，第1层只有1个根结点，所以结点最多为1，命题成立。假设非空二叉树的第j层最多有$2^{j-1}$个结点，而第j层的每一个结点，最多有两个孩子，因此第j+1层最多有$2 \times 2^{j-1} = 2^{(j+1)-1}$个结点。

## 二叉树——基本性质

**【性质1】 一棵非空二叉树的第$i$层最多有$2^{i-1}$个结点。**

**【例题】**

**一棵非空二叉树共有6层，其中第5层有18个结点【判断题】**

**解答：由二叉树的性质1可知，一棵非空二叉树的第5层最多有$2^{5-1} = 2^4 = 16$，因此本题错误。**

## 二叉树——基本性质

**【性质2】 深度为k的二叉树至多有$2^k-1$个结点（k>=1）**

性质2表明的是当二叉树的深度一定时，二叉树能拥有的最大结点数。

**证明：由性质1（一棵非空二叉树的第$i$层最多有$2^{i-1}$个结点）可知，深度为k的二叉树的最大结点数为：**

$$\sum_{i=1}^{k}(\text{第}i\text{层的最大结点数})=\sum_{i=1}^{k}2^{i-1}$$

**将上述公式展开，$X=2^0+2^1+2^2+\cdots+2^{k-1}$**
**两边同时乘以2，即$2X=2^1+2^2+\cdots+2^{k-1}+2^k$**
**下式减去上式可得，$X=-2^0+2^k=2^k-1$**

## 二叉树——基本性质

**【性质2】 深度为k的二叉树至多有$2^k-1$个结点（k>=1）**

**【例题】一棵深度为10的二叉树最少有多少个结点？最多有多少个结点？**

**解答：**
**（1）对于一棵深度为10的二叉树来说，每一层至少有1个结点，因此总共至少有10个结点**
**（2）根据二叉树基本性质2可知，至多有$2^{10}-1$，即1023个结点**

## 二叉树——基本性质

**【性质3】 对任何一棵二叉树T，如果叶子结点数为$n_0$，度为2的结点数为$n_2$，那么$n_0=n_2+1$**

证明：
二叉树的结点是由度为0的结点（叶子）、度为1的结点和度为2的结点组成。假设度为1的结点数为$n_1$，二叉树的总结点数为n，那么$n=n_0+n_1+n_2$

## 二叉树——基本性质

另一方面，二叉树中除根结点外，每一个结点都是其双亲结点的孩子结点，因此，二叉树的孩子结点数X=n-1。
二叉树中孩子结点是由度为2的结点和度为1的结点产生的，一个度为2的结点产生2个孩子结点，一个度为1的结点产生1个孩子结点，所以X又可以表示为$X=n_1+2n_2$。因此

$$n-1=n_1+2n_2$$

最后可得$n_0=n_2+1$
**该性质表明的是叶子结点数与度为2的结点数之间的关系**

## 二叉树——基本性质

【性质3】 对任何一棵二叉树T，如果叶子结点数为$n_0$，度为2的结点数为$n_2$，那么$n_0 = n_2 + 1$

【例题】若一棵二叉树具有10个度为2的结点，5个度为1的结点，那么该二叉树共有多少个叶子结点？总结点数为多少？

## 二叉树——基本性质

解答：

（1）根据二叉树的基本性质3，可知，叶子结点个数=度为2的结点个数+1=10+1=11

（2）总结点数=叶子结点数+度为1的结点数+度为2的结点数=11+5+10=26

说明：性质3明确给出了叶子结点数与度为2的结点数之间的关系。叶子结点数=度为2的结点数+1。

## 二叉树——基本性质

【性质4】 具有n个结点的完全二叉树的深度$k = \lfloor log_2 n \rfloor + 1$

证明：

结合完全二叉树的特点以及性质2（深度为k的二叉树至多有$2^k - 1$个结点）可知，

深度为k的完全二叉树至少有$2^{k-1}$个结点（完全二叉树的k-1层结点全部填满，最后一层至少有一个最左边的结点，即$(2^{k-1}-1)+1 = 2^{k-1}$）。

直接根据性质2可知，深度为k的完全二叉树至多有$2^k - 1$个结点。

## 二叉树——基本性质

【性质4】 具有n个结点的完全二叉树的深度$k = \lfloor log_2 n \rfloor + 1$

因此深度为k的完全二叉树的结点个数n满足如下公式：

$$2^{k-1} \le n < 2^k$$

两边取对数得，$k - 1 \le log_2 n < k$，即$k \le log_2 n + 1 < k + 1$

由于深度k为整数，所以$k = \lfloor log_2 n \rfloor + 1$

注意：

a. 该性质表明对于结点个数确定的完全二叉树，其深度也是确定的。

b. 但是对于一般的二叉树，当给定结点个数时，深度k是不能确定的。

## 二叉树——基本性质

【性质4】 具有n个结点的完全二叉树的深度$k = \lfloor log_2 n \rfloor + 1$

【例题】一棵具有128个结点的完全二叉树，它的深度是多少？

解答：根据二叉树的性质4可知
本例题中的完全二叉树的深度$k = \lfloor log_2 n \rfloor + 1 = log_2 128 + 1$
计算可得，k=8

## 二叉树——基本性质

【性质5】 对一棵具有n个结点的完全二叉树的结点按照从上到下、从左到右的顺序从1开始连续编号，那么对任一结点i，有：

（1）双亲结点的编号
如果i=1，则结点i是二叉树的根，无双亲；
如果i>1，则结点i的双亲是$\lfloor i/2 \rfloor$；

## 二叉树——基本性质

（1）双亲结点的编号
如果i=1，则结点i是二叉树的根，无双亲；
如果i>1，则结点i的双亲是$\lfloor i/2 \rfloor$；

证明：
假设结点i在第k层，双亲结点j在第k-1层的第q个结点。因此双亲的编号为：

$$j = 2^{k-2} - 1 + q$$

如果i是j的左孩子，那么

$$i = (2^{k-1} - 1) + (2(q-1) + 1) = 2^{k-1} + 2q - 2$$

如果i是j的右孩子，那么

$$i = (2^{k-1} - 1) + (2(q-1) + 2) = 2^{k-1} + 2q - 1$$

由上述公式可得:
如果i是j的左孩子，那么i=2j，即$j=i/2$
如果i是j的右孩子，那么i-1=2j，即$j=(i-1)/2$
因此，结点i的双亲$j = \lfloor i/2 \rfloor$

## 二叉树——基本性质

【性质5】 对一棵具有n个结点的完全二叉树的结点按照从上到下、从左到右的顺序从1开始连续编号，那么对任一结点i，有：

（2）左孩子结点的编号
如果i>n/2，则结点i无左孩子，为叶子结点；
如果i<=n/2，则结点i的左孩子是2i；

证明：
刚才推导，已知结点j和其双亲结点i，则满足如下关系：
如果j是i的左孩子，那么j=2i，因此如果i>n/2，则左孩子结点j应该大于n，错误。因此i不能超过n/2。
对于i<=n/2，则结点i的左孩子是2i；

## 二叉树——基本性质

【性质5】 对一棵具有n个结点的完全二叉树的结点按照从上到下、从左到右的顺序从1开始连续编号，那么对任一结点i，有：

（3）右孩子结点的编号

如果 i > (n-1)/2，则结点i无右孩子；

如果 i <= (n-1)/2，则结点i的右孩子是2i+1

证明：

刚才推导，已知结点j和其双亲结点i，则满足如下关系：

如果j是i的右孩子，那么j=2i+1，因此如果 i > (n-1)/2，则右孩子结点j应该大于n，错误。因此i不能超过(n-1)/2。

对于 i <= (n-1)/2，则结点 i 的右孩子是2i+1；

## 二叉树——基本性质

【性质5】 对一棵具有n个结点的完全二叉树的结点按照从上到下、从左到右的顺序从1开始连续编号，那么对任一结点i，有：

（1）双亲结点的编号

如果 i =1，则结点 i 是二叉树的根，无双亲；

如果 i >1，则结点 i 的双亲是$\lfloor i/2 \rfloor$；

（2）左孩子结点的编号

如果 i > n/2，则结点 i 无左孩子，为叶子结点；

如果 i <= n/2，则结点 i 的左孩子是2i；

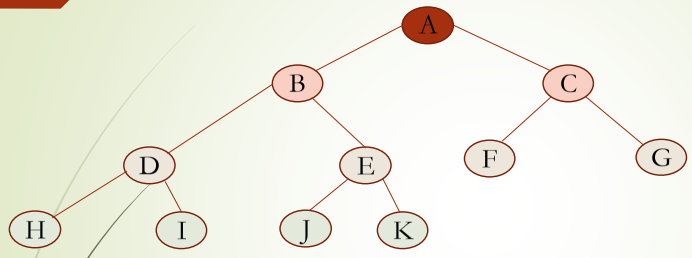
（3）右孩子结点的编号

如果 i > (n-1)/2，则结点i无右孩子；

如果 i <= (n-1)/2，则结点i的右孩子是2i+1

性质5表明的是如果完全二叉树的结点按照从上到下，从左到右的顺序编号，则双亲结点与左右孩子结点的关系可以由编号得到。

## 二叉树——基本性质

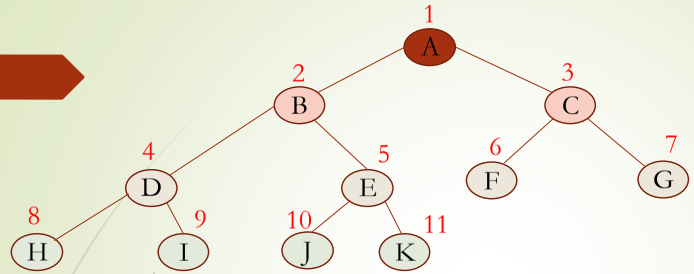



1、这个二叉树是否是完全二叉树?

2、这个二叉树一共多少个结点，它的深度是多少?

3、结点D的编号是多少？它的双亲的编号是多少？它的孩子的编号又是多少?

4、结点F的编号是多少？该结点为什么没有孩子?




1、这个二叉树是否是完全二叉树？是

2、这个二叉树一共多少个结点，它的深度是多少？共11个结点，深度为4，计算公式 $\lfloor log_2 n \rfloor + 1$

3、结点D的编号是多少？它的双亲的编号是多少？它的孩子的编号又是多少？结点D的编号是4，双亲的编号是$\lfloor i/2 \rfloor$=2，左孩子的编号是2i=8，右孩子的编号为2i+1=9

4、结点F的编号是多少？它的孩子的编号是多少？

结点F的编号是6，因为i>n/2，即6>11/2，因此没有左孩子；

因为i>(n-1)/2，即6>(11-1)/2，因此没有右孩子

# 二叉树——基本性质

【例题】对一棵具有20个结点的完全二叉树的结点按照从上到下、从左到右的顺序从1开始连续编号。

① 编号15的双亲结点是几号结点?

② 编号15是左孩子还是右孩子?

③ 编号10是否有左右孩子?如果有，编号各自是多少?

## 一、练习题

1、结合二叉树的基本性质，分析并回答下列问题

（1）一棵非空二叉树的第10层最多有多少个结点？

性质1:一棵非空二叉树的第$i$层最多有$2^{i-1}$个结点

（2）一棵深度为6的二叉树的结点总数最多有多少？

性质2: 深度为k的二叉树至多有$2^k-1$个结点

（3）假设完全二叉树的根结点为第1层，树中第10层有5个叶子结点，该完全二叉树有多少个结点？

性质2: 深度为k的二叉树至多有$2^k-1$个结点，以及完全二叉树的定义

（4）一棵具有100个结点的完全二叉树的结点从上到下、从左到右的顺序从1开始连续编号，结点33的双亲结点、左孩子结点及右孩子结点分别是几号结点

性质5: 结点 i 的双亲是$\lfloor i/2 \rfloor$,左孩子结点2i，右孩子结点2i+1

## 一、练习题

2、结合二叉树的基本性质，判断下面说法的正确性

（1）一棵具有10个结点的二叉树的深度为4

性质4: 具有n个结点的完全二叉树的深度：$\lfloor log_2(n) \rfloor + 1$

（2）某棵具有9个结点的完全二叉树，它的叶子结点个数为5，度为2的结点个数为4

性质3: n0=n2+1

（3）在（2）中树的基础上，将其补成满二叉树，需要增加6个结点

计算题：

（4）如果要将深度为10，总结点个数为10的二叉树补全成完全二叉树，

至少需要增加 ？个结点

最多需要增加 ？个结点

# 二叉树——存储结构

## 1、二叉树的顺序存储结构

### （1）完全二叉树的顺序存储结构

根据二叉树基本性质5可知，在完全二叉树中，结点$i$的双亲为$\lfloor i/2 \rfloor$，结点i的左孩子为2i，结点i的右孩子为2i+1。

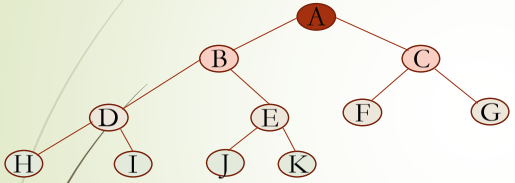
*存储方法：*

将完全二叉树的结点按照从上到下，从左到右的顺序从1开始连续编号，把编号为i的结点存放在线性存储单元的第i个位置，那么它的左孩子结点存放在2i位置，右孩子结点放在2i+1位置。

# 二叉树——存储结构

## 1、二叉树的顺序存储结构

### （1）完全二叉树的顺序存储结构




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

# 二叉树——存储结构

## 1、二叉树的顺序存储结构

### （2）一般二叉树的顺序存储结构

问题：如果把它的结点按照从上到下、从左到右的顺序存储在一维存储单元中，无法反映结点之间的双亲孩子关系

**存储方法：**

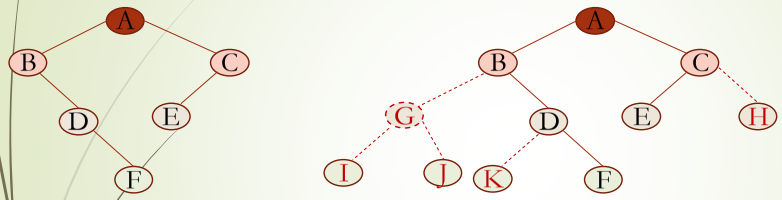
- 将一棵一般二叉树补全为一棵完全二叉树
- 将它按照上述完全二叉树的顺序存储方式进行存储

**注意：**

**新补上的结点在存储时只占位置，不存放结点数据**

## 1、二叉树的顺序存储结构

### （2）一般二叉树的顺序存储结构




| | A | B | C | ^ | D | E | ^ | ^ | ^ | ^ | F |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

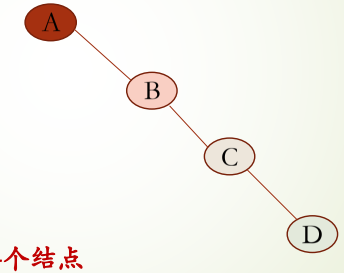
**新补上的结点在存储时只占位置，不存放结点数据**

# 二叉树——存储结构

## 1、二叉树的顺序存储结构

### （2）一般二叉树的顺序存储结构

**对于一般的二叉树，需要增加一些结点才能存储其结点数据和结构关系，这会造成存储空间的浪费。**

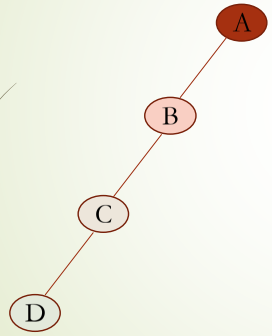



**对于深度为4的右偏斜二叉树，共4个结点**

**需要增加11个结点才能存储这个二叉树。**

## (2) 一般二叉树的顺序存储结构

**思考：对于一棵深度为4的4个结点的左偏斜二叉树，至少需要增加多少结点才能存储这个二叉树?**




# 二叉树——存储结构

## 2、二叉树的链式存储结构

**链表可以用来表示一维的线性结构，也可以用来表示非线性的二叉树结构。**

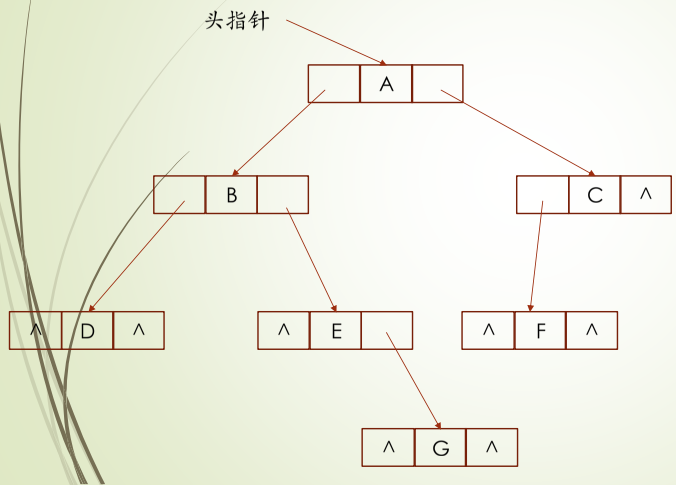
### (1) 二叉链表存储结构

二叉链表中每个结点包括3个域：数据域、左孩子指针域和右孩子指针域。左、右孩子指针域分别指示左右孩子结点的存储地址。每个结点的两个指针域分出了两个叉，因此该存储结构被形象地称为二叉链表。


**二叉链表结点的存储结构**

### (1) 二叉链表存储结构

当结点没有左（右）子树时，左（右）孩子指针域lchild(rchild)的值为空，用“^”或NULL表示。




## 2、二叉树的链式存储结构

### (1) 二叉链表存储结构

在二叉链表存储结构中，度为1的结点有1个空指针域，度为0的结点（叶子结点）有两个空指针域，度为2的结点没有空指针域。因此，

在具有n个结点的二叉树的二叉链表存储结构中，空指针域的个数=$2n_0 + n_1 = n_0 + \boxed{n_0} + n_1 = n_0 + \boxed{n_2 + 1} + n_1 = n_0 + n_2 + n_1 + 1$，

**在具有n个结点的二叉树的二叉链表存储结构中，空指针域的个数为n+1**

## 2、二叉树的链式存储结构

### （1）二叉链表存储结构

在二叉链表存储结构中，从结点的左右指针域可以找到该结点的左右孩子，很快也很方便。

问题：如果要找该结点的双亲结点，怎么办呢？

### （2）三叉链表存储结构

在二叉链表结点存储结构的基础上，再增加一个域，用来记录该结点的双亲结点的地址，就可以容易的查找该结点的双亲。

| lchild | data | parent | rchild |
|---|---|---|---|

三叉链表结点的存储结构

在三叉链表中，除根结点的parent域为空外，其余结点的parent域都不为空，指向其双亲。但是增加了一定的空间开销

## 二叉树的基本操作

1、初始化
2、求根
3、求双亲

```
typedef struct TreeNode {
    int value;              // 节点的值
    struct TreeNode *left;    // 左子节点
    struct TreeNode *right;   // 右子节点
} TreeNode;

TreeNode* initateEmptyTree() {
    TreeNode* root = NULL;
    return root;
}
```

## 二叉树的基本操作

1、初始化
2、求根
3、求双亲

```
typedef struct TreeNode {
    int value;              // 节点的值
    struct TreeNode *left;    // 左子节点
    struct TreeNode *right;   // 右子节点
} TreeNode;


TreeNode* getRoot(TreeNode*root) {
    return root;
}

root = getRoot(root->left);
```

## 二叉树的基本操作

1、初始化
2、求根
3、求双亲

```
typedef struct TreeNode {
    int value;              // 节点的值
    struct TreeNode *left;    // 左子节点
    struct TreeNode *right;   // 右子节点
} TreeNode;
```

```
typedef struct TreeNode2 {
    int value;              // 节点的值
    struct TreeNode2 *left;    // 左子节点
    struct TreeNode2 *right;   // 右子节点
    struct TreeNode2 *parent;  //双亲结点
} TreeNode2;
```