# 数据结构

## 一、数据

### 1.数据的基本概念

**程序=数据结构+算法**

数据元素(data element)：是数据的基本单位，是数据集合的个体。如一位学生完整信息的数据记录就是一个数据元素。数据元素通常由若干个数据项组成。数据项具有原子性，是不可分割的最小单位。如java中类的成员变量就是类的数据项。

数据对象(data object)：性质相同的数据元素的集合，是数据的子集。如一个学校的所有学生的集合就是数据对象。

数据结构(data structure)：是指相互之间存在一种或多种特定关系的数据元素的集合。其表现形式：**数据的逻辑结构、数据的存储结构**。

数据结构={D,S}。其中D是数据元素的集合，S是D中数据元素之间的关系集合。

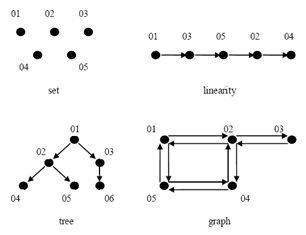
数据的**逻辑结构**：逻辑层面上数据之间的关系

**集合结构**：set={K,R}，K不为空，R为空。数据元素的集合非空，而数据元素之间除了同属一个集合之外不存在任何关系(关系集合为空)。

**线性结构**：除第一个元素以外每个数据元素有且仅有一个直接前驱元素，除最后一个元素以外每个数据元素有且仅有一个直接后续元素。数据元素之间是1对1的关系，即线性关系。如：线性表、栈、队列。

**树型结构**：除了第一个数据元素以外每个数据元素有且仅有一个直接前驱元素，但是可以有多个直接后驱元素。数据元素之间是1对N的联系。

**图形结构**：每个数据元素可以有多个直接前驱元素，也可以有多个直接后续元素。数据元素之间是M对N的联系。



数据的**(物理)存储结构**：物理层面

计算机的存储形式。每一种逻辑结构都有物理结构。

**顺序存储结构**：**数组**。数据元素的存储对应于一块连续的存储空间。数据元素之间的前驱和后续关系通过数据元素 在存储器中的相对位置来反映。

**链式存储结构**：**链表**。数据元素的存储对应的是不连续的存储空间。每个存储节点对应一个需要存储的数据元素。元素之间的逻辑关系通过存储节点之间的链接关系反映出来。

### 2.栈

栈是运算受限的线性表。仅允许在表的一端进行插入和删除操作，不允许在其他任何位置进行插入、查找、删除等操作。

表中进行插入、删除操作的一端称为栈顶（**top**），栈顶保存的元素称为栈顶元素。栈顶、栈底。当栈中没有数据元素时称为空栈；向一个栈插入元素又称为进栈或入栈；从一个栈中删除元素又称为出栈或退栈。

栈又称堆栈，又把**堆栈**称为**后进先出** (**Last in First Out**，简称**LIFO**)

### 2.队列

也是一种运算受限的线性表，其限制是仅允许在表的一端进行插入，而在表的另一端进行删除。

在队列中把插入数据元素的一端称为队尾（**rear**），删除数据元素的一端称为队首（**front**）。向队尾插入元素称为进队或入队，从队列中删除元素称为离队或出队。

在队列中，先进队的元素必定先离队，因此**队列**称为**先进先出表**(**First in First Out**，简称**FIFO**)

## 二、树

树是由一个集合以及在该集合上定义的一种关系构成的。集合中的元素称为树的结点。

空树中不包含任何结点。非空树有且仅有一个结点。

**树的层次(level)**从根结点开始定义：层次为0的结点为根结点。即**树的层次是从0开始的**。

**树的深度(depth)或高度**：**树中结点最大的层次数**。树中结点也有高度，其高度是以该结点为根的树的高度。

**结点的度(degree)**：**结点拥有的子树的数目称为结点的度**。度为0的结点称为**叶子**(leaf)或**终端结点**。度不为0的结点称为非终端结点或分支结点。

**树的性质**：

**性质1**：树中的结点数等于树的边数加1，也等于所有结点的度数之和加1。在树中结点总数与边的总数相等。

**性质2**：树中任意两个结点之间都存在唯一的路径。根结点开始，存在到其他任意结点的一条唯一路径，根到某个结点路径的长度，恰好是该结点的层次数。

有序树：将树中结点的各子树看成是从左至右是有次序的，则称该树为有序树。若不考虑子树的顺序则称为无序树

**m叉树**：树中所有结点最大度数为m的有序树称为**m**叉树。

**二叉树(binary tree)**：**每个结点的度均不超过2的有序树**。二叉树中有左子树、右子树。

**二叉树的第n层上最多有2的n次方个结点**。

**满二叉树**：每层结点都达到最大数。高度为n并且有个结点的二叉树。

**完全二叉树**：在一棵满二叉树中，在**最下层**从**最右侧起**去掉相邻的若干叶子结点，得到的二叉树。

满二叉树一定是一个完全二叉树，但完全二叉树不一定是一个满二叉树。(将一棵满二叉树的**最下层**从右侧开始去掉若干个子结点，得到完全二叉树。满二叉树是一个完全二叉树是因为该满二叉树相当于将最下层的所有结点都去掉了。)

对于完全二叉树、满二叉树可以使用顺序存储结构，且不会浪费存储结构。对于其他的树用顺序存储结构则会浪费空间，用链式存储结构。

二叉树的存储结构：顺序存储结构、链式存储结构

二叉树链表中存放的元素：数据、左孩子、右孩子

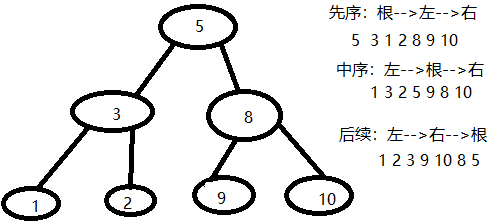
1.遍历二叉树：

先序遍历(**DLR**)：**根结点--->左子树--->右子树**

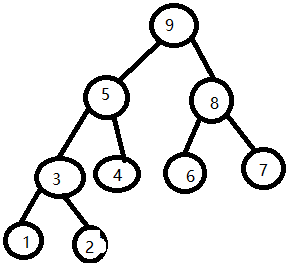
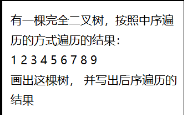
中序遍历(**LDR**)：**左子树--->根结点--->右子树**

后序遍历(**LRD**)：**左子树--->右子树--->根结点**

(记忆：这里的先、中、后序可看成是针对根结点的，先序则可看成根节点先遍历，中序则可看成根结点中间遍历，后序同样)



共有9个结点。假设是一个满二叉树，则2(n+1)-1>=9，计算得到n=3，即该完全二叉树的最大层次数为3，即有4层。画出这个完全二叉树：



十进制转换成二进制采用的是栈的先进后出形式。