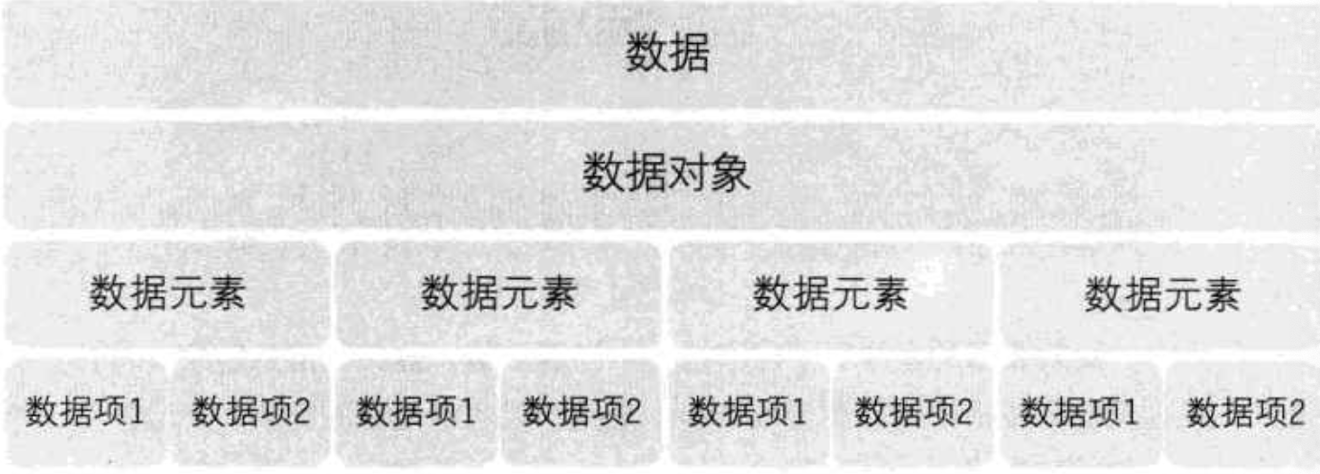
**笔记**

**第一章**

**1.4 基本概念和术语**



1.4.1 数据

数据：是描述客观事物的符号，是计算机中可以操作的对象，是能被计算机识别，并输入给计算机处理的符号集合。

1.4.2 数据元素

数据元素：是组成数据的、有一定意义的基本单位，在计算机中通常作为整体处理。也被称为记录。

如：在人类中，数据元素是人

1.4.3 数据项

数据项：一个元素可以由若干个数据项组成，数据项是数据不可分割的最小单位。

如：人这个数据元素可以有眼、耳、口、手这些数据项，也可以有性别、年龄、出生地等数据项

1.4.4 数据对象

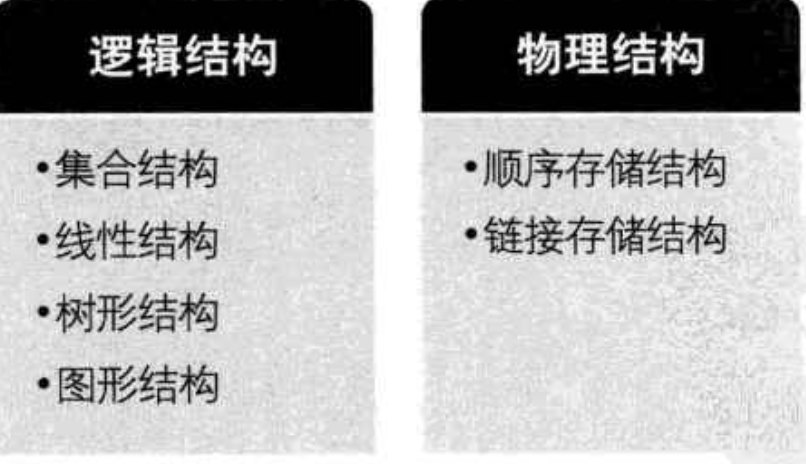
数据对象：是性质相同的数据元素的集合，是数据的子集。

1.4.5 数据结构

数据结构：是相互之间存在一种或多种特定关系的数据元素的集合。

**1.5 逻辑结构与物理结构**

数据结构分为逻辑结构与物理结构

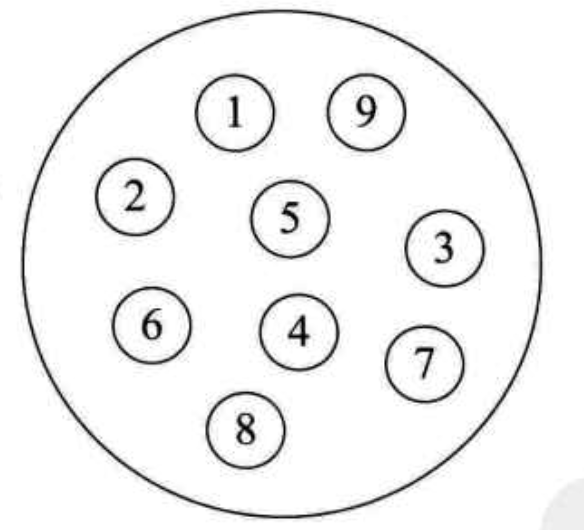


1.5.1 逻辑结构

逻辑结构：是指数据对象中数据元素之间的相互关系。

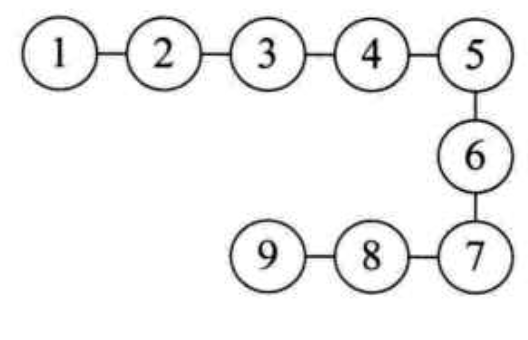
1. 集合结构

集合结构：集合结构中的数据元素除了同属于一个集合外，他们之间没有其他关系。（各数据元素是平等的）



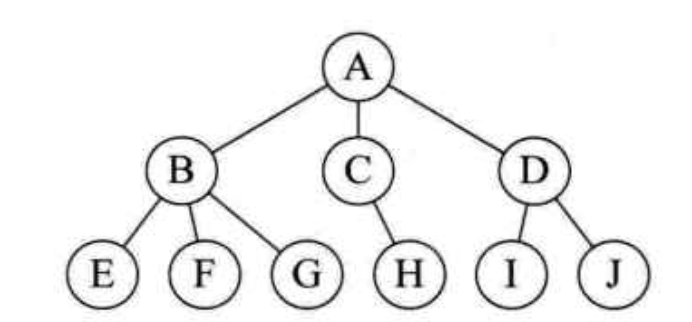
1. 线性结构

线性结构：线性结构中的数据元素之间是一对一的关系。



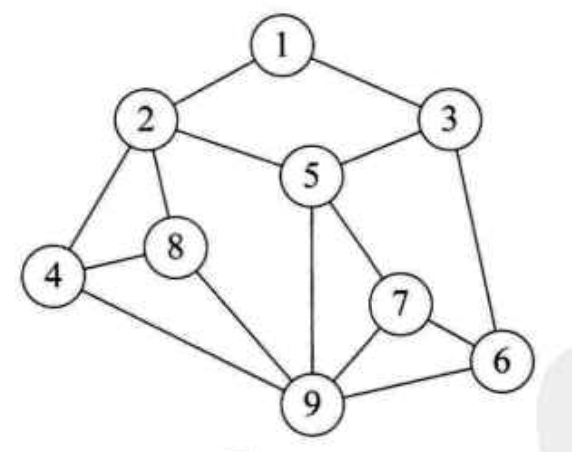
1. 树形结构

树形结构：树形结构中的数据元素之间存在一种一对多的层次关系



1. 图形结构

图形结构：图形结构的数据元素是多对多的关系

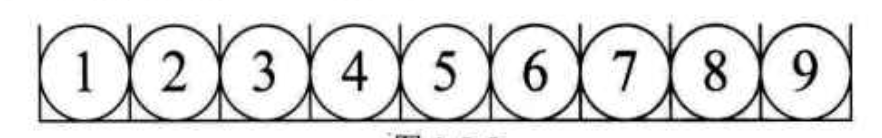


1.5.2 物理结构

物理结构：是指数据的逻辑结构在计算机中的存储形式，为顺序存储和链式存储

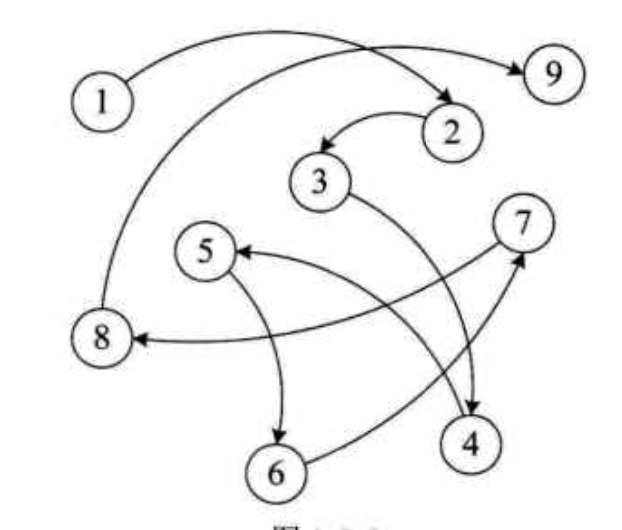
1. 顺序存储结构

顺序存储结构：是把数据元素存放在地址连续的存储单元里，其数据间的逻辑关系和物理关系是一致的。（如数组）



1. 链式存储结构

链式存储结构：是把数据元素存放在任意的存储单元里，这组存储单元可以是连续的，也可以是不连续的。（如链表）



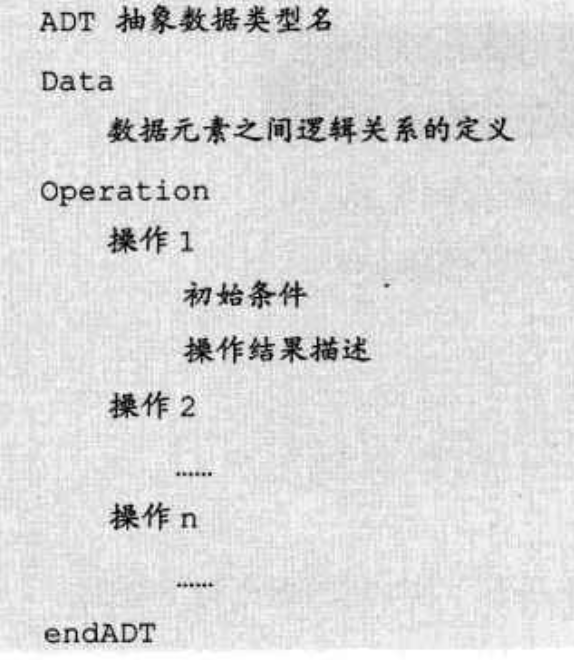
**1.6 抽象数据类型**

1.6.1 数据类型

1.6.2 抽象数据类型

抽象数据类型：是指一个数学模型及定义在该模型上的一组操作。

一个抽象数据类型定义了：一个数据对象、数据对象中各数据元素之间的关系及对数据元素的操作



**第二章 算法**

算法：是解决特定问题求解步骤的描述，在计算机中表现为指令的有限序列，并且每条指令表示一个或多个操作。

**2.5 算法的特性**

算法具有五个基本特性：输入、输出、有穷性、确定性和可行性

2.5.1 输入输出

算法具有零个或多个输入（仅打印不需输入）

算法至少有一个或多个输出

2.5.2 有穷性

有穷性：指算法在执行有限的步骤之后，自动结束而不会出现无限循环，并且每一个步骤在可接受的时间内完成。

2.5.3 确定性

确定性:算法的每一步骤都具有确定的含义，不会出现二义性（相同的输入只能有唯一的输出结果）

2.5.4 可行性

可行性：算法的每一步都必须是可行的，也就是说，每一步都能够通过执行有限次数完成

**2.9 算法时间复杂度**

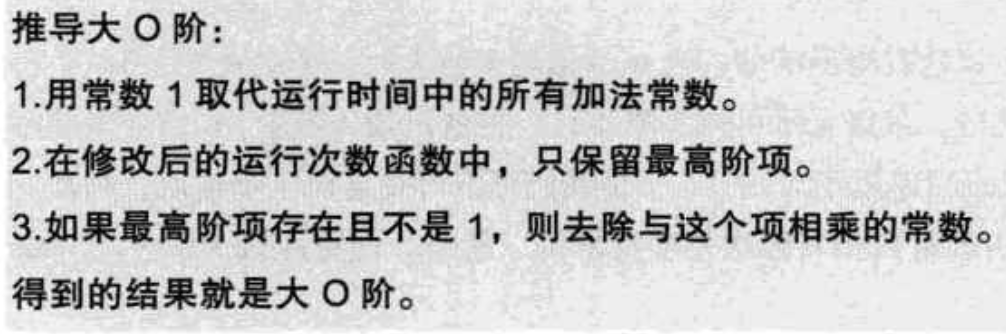
2.9.1 算法时间复杂度定义

在进行算法分析时，语句总的执行次数T(n）是关于问题规模n的函数，

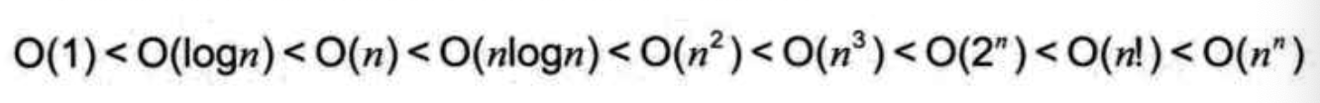
进而分析T（n）随n的变化情况并确定T（n）的数量级。算法的时间复杂度，也就是算法的时间度量，

记作：T(n) = O（f(n)）。它表示随问题规模n的增大，算法执行时间的增长率和f(n)的增长率相同，

称作算法的渐进时间复杂度，简称为时间复杂度。其中f(n)是问题规模n的某个函数。



时间复杂度排序：



**2.12 算法空间复杂度**

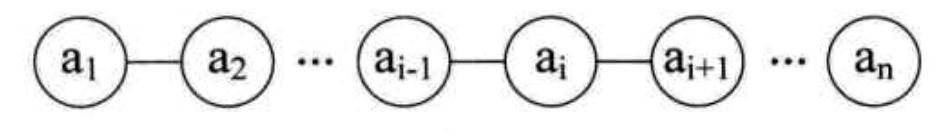
算法空间复杂度通过计算算法所需的存储空间实现，算法空间复杂度的计算公式记作：S(n) = O(f(n)),

其中，n为问题的规模，f(n)为语句关于n所占存储空间的函数。

**第三章 线性表**

**3.2 线性表的定义**

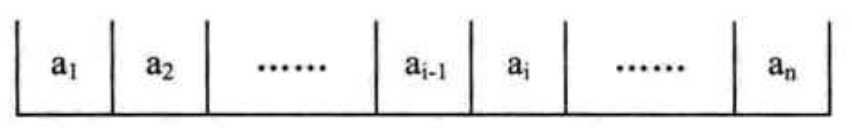
线性表（List）： 零个或多个数据元素的有限序列

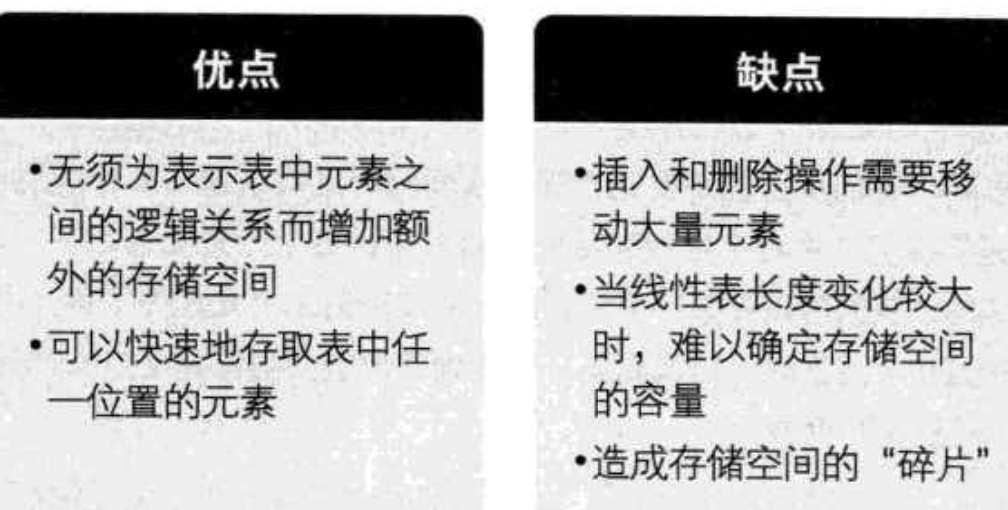


**3.4 线性表的顺序存储结构**

3.4.1 顺序存储定义

线性表的顺序存储结构，指的是用一段地址连续的存储单元依次存储线性表的数据元素。

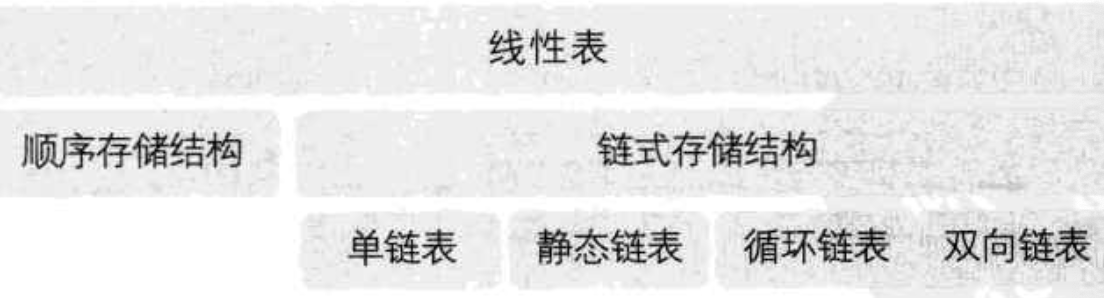




**3.6 线性表链式存储结构**



总结：



**第四章 栈与队列**

栈是限定仅在表尾进行插入和删除操作的线性表。

队列是只允许在一端进行插入操作、而在另一端进行删除操作的线性表。

**4.6 栈的链式存储结构及实现**

4.8.2 递归定义

递归：把一个直接调用自己或通过一系列的调用语句间接地调用自己的函数，称作递归函数

**4.10 队列的定义**

队列是只允许在一端进行插入操作、而在另一端进行删除操作的线性表。（如：键盘输入字母或数字）

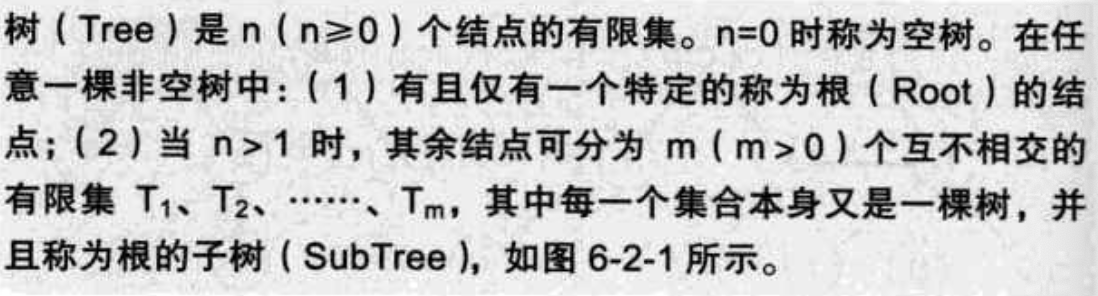
队列是一种先进先出（First In First Out）的线性表，简称FIFO。

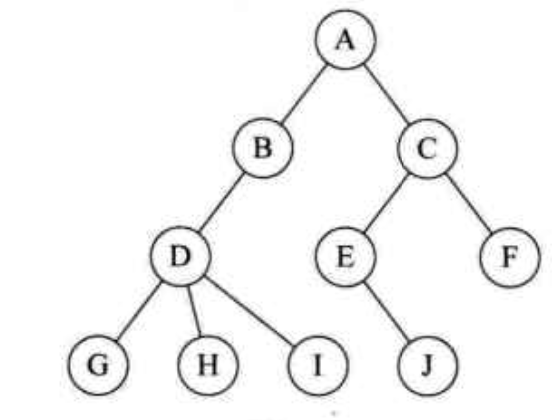
允许插入的一端称为队尾，允许删除的一端称为队头。

**第五章 串**

串：串（string）是由零个或多个字符串组成的有限序列，又名叫字符串

**第六章 树**





注意：

1. N > 0时根结点是唯一的，不可能存在多个根节点。
2. m > 0时，子树的个数没有限制，但它们一定是互不相交的。

6.2.1 结点分类

结点拥有的子树数称为结点的度。

度为0的结点称为叶结点或终端结点；度不为0的结点称为非终端结点或分支结点。

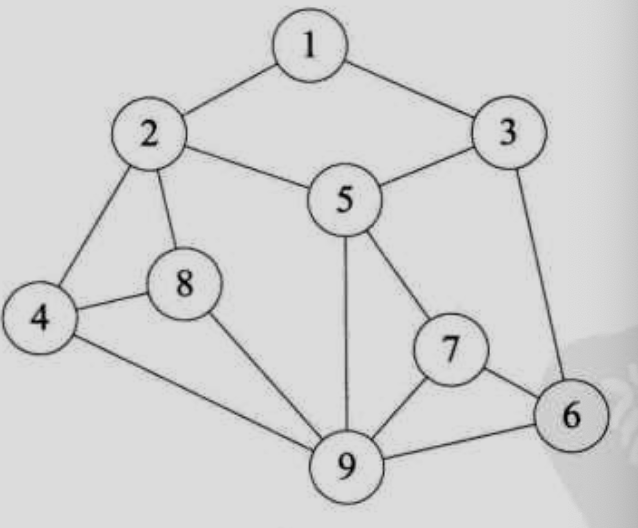
除根结点外，分支结点也称为内部结点。

树的度是树内各结点的度的最大值。

**第七章 图**

图：图是由顶点的有穷非空集合和顶点之间边的集合组成，

通常表示为：G(V,E)，其中，G表示一个图，V是图G中顶点的集合，E是图G中边的集合。



1.图中的数据元素——顶点

2.图中顶点的集合V——有穷非空

3.顶点之间的逻辑关系边——可以是空的

7.2.1 各种图定义

无向边：若顶点Vi到Vj之间的边没有方向，则称这条边为无向边（Edge）

用无序偶对（Vi，Vj）来表示。如果任意两个顶点之间的边都是无向的，则称该图为无向图

有向边：若从顶点Vi到Vj的边有方向，则称这条边为有向边，也称为弧（Arc）

用有序偶对<Vi，Vj>来表示，Vi称为弧尾，Vj称为弧头。（先后不能乱）

如果任意两个顶点之间的边都是无向的，则称该图为有向图

注意：无向边用小括号“（）”表示，有向边用尖括号“<>”表示

**第八章 查找**

**8.4 有序表查找**

8.4.1 折半查找

Mid = (low + high)/2

8.4.2 插值查找(适用于表长较大，关键字分布较均匀的查找表)

折半查找: mid = low + 1/2(high - low)

插值查找：

mid = low + (key - a[low])/(a[high] - a[low])\*(high - low)

8.4.3 斐波那契查找（适用于数据量较大的数据）