数据结构与算法（标注版）

1.1 算法

1、**算法**是指解题方案的准确而完整的描述。换句话说，算法是对特定问题求解步骤的一种描述。

```
I=1                    ，变量 I 赋值为 1
S=0                    ，变量 S 赋值为 0
DO WHILE I<=100        ，循环结构，当 I 小于 100 时循环
S=S+I                  ，对 I 的值累加求和
I=I+1                  ，变量 I 的值没循环一次增加 1
LOOP                   ，循环结构，遇到 loop 返回 DO 处重新循环
MSGBOX  S              ，输出求和变量 S 的值
```

算法规定了解决某类问题所需的**操作语句**以及**执行顺序**，使其能通过有限的指令语句，在一定时间内解决问题。

算法是一个**操作序列、有限长度，目的是解决某类问题**。

***：**算法不等于程序，也不等于计算方法。程序的编制不可能优于算法的设计。

2、算法的基本特征

（1）可行性。针对实际问题而设计的算法，执行后能够得到满意的结果。

（2）确定性。每一条指令的含义明确，无二义性。并且在任何条件下，算法只有唯一的一条执行路径，即相同的输入只能得出相同的输出。

（3）有穷性。算法必须在有限的时间内完成。有两重含义，一是算法中的操作步骤为有限个，二是每个步骤都能在有限时间内完成。

（4）拥有足够的情报。指的是有足够的输入和输出。

***：**综上所述，所谓算法，是一组严谨地定义运算顺序的规则，并且每一个规则都是有效的，且是明确的，此顺序将在有限的次数下终止。

3、算法的基本要素

一个算法通常由两种基本要素组成：一是对数据对象的运算和操作；二是算法的控制结构。

(1）算法中对数据的运算和操作

每个算法实际上市按解题要求从环境能进行的所有操作中选择合适的操作所组成的一组指令序列。因此，计算机算法就是计算机能处理的操作所组成的指令序列。

在一般的计算机系统中，基本的运算和操作有以下四类：

①算术运算：主要包括加、减、乘、除等运算；

②逻辑运算：主要包括与(AND)、或(OR)、非(NOT)等运算；

③关系运算：主要包括大于、小于、等于、不等于等运算

④数据传输：主要包括赋值、输入、输出等操作。

(2)算法的控制结构

顺序、选择和循环。

4、算法的基本方法（计算机解题的过程实际上是在实施某种算法）

（1）列举法（列举所有解决方案）

根据提出的问题，列举所有可能的情况，并用问题中给定的条件检验哪些是需要的，哪些是不需要的。

（2）归纳法（特殊->一般）适合于列举量为无限的情况

通过列举少量的特殊情况，经过分析，最后找出一般的关系。

（3）递推法 （已知->未知）

从已知的初始条件出发，逐次推出所要求的各中间结果和最后结果。

（4）递归法（逐层分解）

将一个复杂的问题归结为若干个较简单的问题，然后将这些较简单的每一个问题再归结为更简单的问题

（5）减半递推法

“减半”是指将问题的规模减半，而问题的性质不变，所谓“递推”是指重复“减半”的过程。

（6）回溯法

复杂应用，找出解决问题的线索，然后沿着这个线索逐步多次“探”、“试”。

5、算法复杂度主要包括时间复杂度和空间复杂度。

算法的复杂度是衡量算法好坏的量度。

（1）算法时间复杂度是指执行算法所需要的计算工作量，可以用执行算法的过程中所需基本运算的执行次数来度量。

影响计算机工作量的主要因素：

第一：基本运算次数；　　第二：问题规模。

下面的方法不能用来度量算法的时间复杂度：

第一：算法程序的长度或算法程序中的语句（指令）条数；

第二：算法程序所执行的语句条数；

第三：算法程序执行的具体时间

（2）算法空间复杂度是指执行这个算法所需要的内存空间。

一个算法所用的内存空间包括： 1）算法程序所占用的存储空间；

2）输入的初始数据所占的存储空间；3）算法执行过程中的额外空间。

6、考题练习：

1）下列叙述正确的是（）

（A）算法就是程序

（B）算法强调的是利用技巧提高程序执行效率

（C）设计算法时只需考虑结果的可靠性

（D）以上三种说法都不对

2）下面叙述正确的是（）

（A）算法的执行效率与数据的存储结构无关

（B）算法的空间复杂度是指算法程序中指令（或语句）的条数

（C）算法的有穷性是指算法必须能在执行有限个步骤之后终止

（D）以上三种描述都不对

3）下列叙述中正确的是（）

（A）一个算法的空间复杂度大，则其时间复杂度也必定大

（B）一个算法空间复杂度大，则其时间复杂度必定小

（C）一个算法的时间复杂度大，则其时间复杂度必定小

（D）以上三种说法都不对

4）算法的空间复杂度是指（　）

（A）算法程序中变量的个数

（B）算法程序中的指令条数

（C）算法程序中各控制变量所占的额外空间

（D）算法执行过程中所需要的存储空间

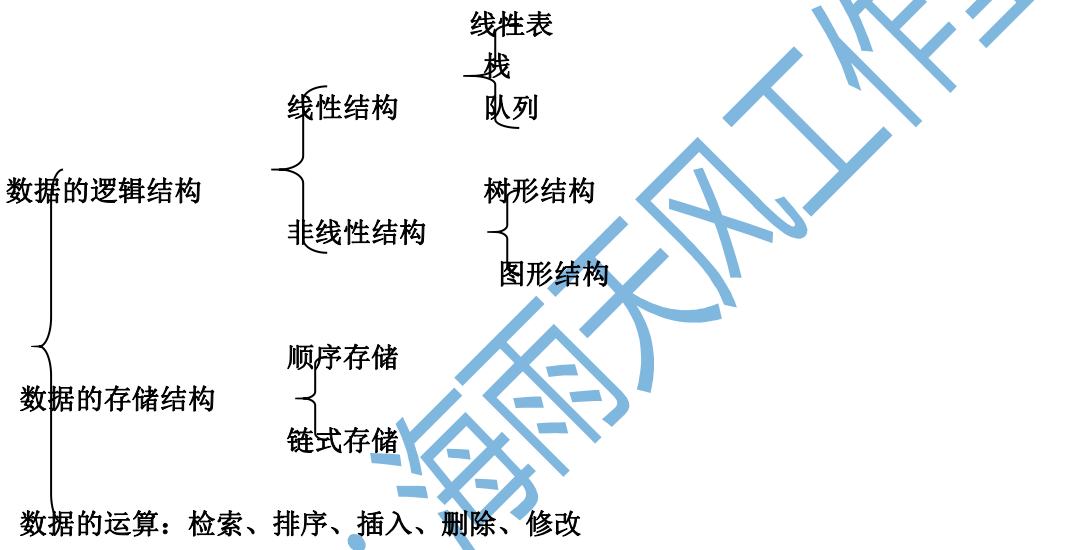
1.2 数据结构的基本概念

1、基本概念：

1）数据：在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。

2）数据元素：数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

3）数据结构：是指相互有关联的数据元素的集合。



2、逻辑结构：数据元素之间的逻辑关系，从逻辑关系上描述数据，它与数据的存储无关，是独立于计算机的。

3、存储（物理）结构：是指数据元素及其关系在计算机内存中的表示，即数据的逻辑结构在计算机存储空间中的存放形式。

注意：对于同一个逻辑结构来说，采用不同的存储结构，其数据处理的效率是不同的。因此，在数据处理时，选择合适的存储结构很重要。

各数据元素在计算机存储空间中的位置关系与它们的逻辑关系不一定相同。

由于数据元素在计算机存储空间中的位置关系可能与逻辑关系不同，因此，为了表示存放在计算机存储空间中的各数据元素之间的逻辑关系（即前后间关系）在数据结构中，不仅要存放各数据元素的本身，而且还需要存放各数据元素之间的前后件关系的信息。

4、逻辑结构与物理结构的关系：

（1）一种逻辑结构可以用不同的物理结构来实现

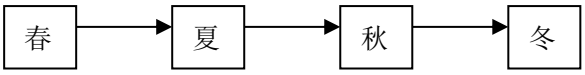
（2）逻辑结构决定了算法的设计

（3）物理结构决定了算法的实现

数据结构的图形表示：表示数据结构的常用方法：二元关系表和图形表示。

例：一年四季的数据结构可以表示成：

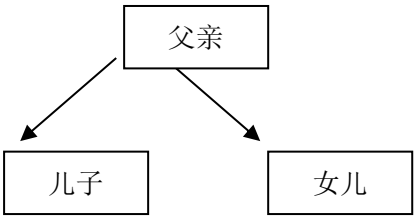
用图形表示为：



1）　数据元素：用中间标有元素值的方框来表示，称为**数据结点**，简称为**结点**。

2）　元素之间的前后关系：用一条有向线段从前件结点指向后件结点（有时可以省略箭头）

例如：家庭成员数据结构可以表示称为：



在数据结构中，没有前件的结点称为根结点，没有后件的结点称为终端结点（也称为叶子结点）其它的称为内部结点。

5、线性结构与非线性结构

数据结构的分类：

根据数据结构中各数据元素之间的前后关系的复杂度，一般将数据结构分成两大类：线性结构和非线性结构。

注意：线性结构与非线性结构是在数据的逻辑结构概念下的一种划分方法，与数据的存储结构无关。前面说过，一种数据的逻辑结构根据需要可以表示成多种存储结构，但只要数据的逻辑结构是属于线性结构，则该逻辑结构的任意一种存储结构也都属于线性结构。

6、线性结构的定义：

如果一个非空的数据结构满足下列两个条件：

（1）　有且只有一个根结点（每有前件的结点称为根结点）

（2）　每个结点最多有一个前件，也最多有一个后件。则称该数据结构为线性结构，线性结构又称为线性表。

（3）　线性结构的操作。

在一个线性结构中插入或删除任何一个结点后还应是线性结构。

1.3 线性表及其顺序存储结构

1、线性表就是线性结构。**线性表**由一组数据元素构成，数据元素的位置只取决于自己的序号，元素之间的相对位置是线性的。线性表是由 n(n≥0)个数据元素组成的一个有限序列，表中的每一个数据元素，除了第一个外，有且只有一个前件，除了最后一个外，有且只有一个后件

2、线性表的顺序存储结构也称为顺序表。

3、线性表的顺序存储结构具有两个基本特点：(1)线性表中所有元素所占的存储空间是连续的；(2)线性表中各数据元素在存储空间中是按逻辑顺序依次存放的，其前后两个元素在存储空间中是紧邻的，且前件元素一定存储在后件元素的前面。

4、线性表的长度：是指线性表中数据元素的个数，当 N=0 时，称为空表。

5、通常定义一个一维数组来表示线性表的顺序存储空间，用一维数组来存放线性表时，该一维数组的长度不要定义为太短，也不要定义为太长。

6、顺序表（线性表的顺序结构）的插入运算：在一般情况下，要在第 i（1≤i≤n）个元素之前插入一个新元素时，首先要从最后一个（即第 n 个）元素开始，直到第 i 个元素之间共 n-i+1 个元素依次

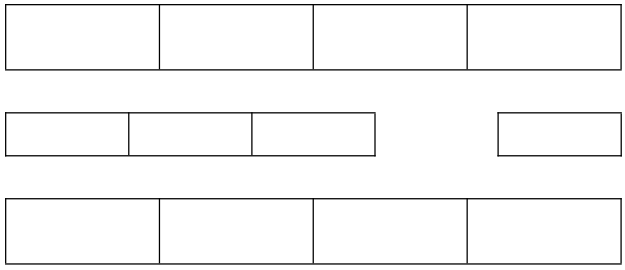
向后移动一个位置，移动结束后，第 i 个位置就被空出，然后将新元素插入到第 i 项。插入结束后，线性表的长度就增加了 1。

7、顺序表的删除运算：在一般情况下，要删除第 i（1≤i≤n）个元素时，则要从第 i+1 个元素开始，直到第 n 个元素之间共 n-i 个元素依次向前移动一个位置。删除结束后，线性表的长度就减小了 1。

*：进行顺性表的删除运算时也需要移动元素，在等概率情况下，平均需要移动（n-1）/2 个元素。

插入、删除运算不方便。

例题：选择正确表示线性表(A1,A2,A3,A4)的顺序结构是（　）



综上所述：线性表的顺序存储结构对于经常需要变动的大线性表就不合适了，因为插入和删除的效率比较低。

线性表的顺序存储的缺点

第一、在插入和删除时需要移动元素（除栈和队列之外）

第二、上溢（或下溢）

1.4 栈

1、栈的定义：展示一种特殊的线性表，即限定在一端进行插入与删除的线性表。

2、栈顶、栈底的定义：在栈中，允许插入与删除的一端称为栈顶，不允许插入与删除的另一端称为栈底。栈顶元素总是最后被插入的元素，栈底元素总是最先被插入的元素。

3、栈的操作原则：是按照“先进后出”或“后进先出”的原则组织数据的。

4、栈的基本运算：1）插入元素称为入栈运算；

2）删除元素称为退栈运算；

3）读栈顶元素是将栈顶元素赋给一个指定的变量，此时指针无变化。

5、栈具有记忆功能。

6、注意：1）在顺序存储结构下，栈的插入与删除运算都不需要移动表中其他数据元素；

2）在栈中，栈顶指针动态反映了栈中元素的变化情况。

7、在顺序结构下的读栈顶元素是指将栈顶元素赋值给一个指定的变量。（读不是删除，删除一定要先读。在只读的情况下，原来是多少，读后还是多少）

8、读栈顶元素过程中应注意的问题：

1）读栈顶元素不删除栈顶元素，只是将它的值赋给一个变量。因此在这个运算中栈顶**指针不会改变；**

2）当栈顶指针为 0 时，说明栈为空，读不到栈顶元素。

1.5 队列及其运算

1、队列的定义：队列是指允许在一端（队尾）进入插入，而在另一端（队头）进行删除的线性表。

2、允许插入的一端叫队尾。尾指针（Rear）指向队尾元素。

允许删除的一端叫对头。头指针（front）指向排头元素的前一个位置（队头）。

3、数据操作方法：队列是“先进先出”或“后进后出”的线性表。

4、队列运算包括：1）入队运算：从队尾插入一个元素；

2）退队运算：从队头删除一个元素。

5、在顺序存储结构下，队列的插入与删除运算都不需要移动表中其他数据元素。

6、循环队列及其运算：所谓循环队列，就是将队列存储空间的最后一个位置绕到第一个位置，形成逻辑上的环状空间，供队列循环使用。在循环队列中，用队尾指针 rear 指向队列中的队尾元素，用排头指针 front 指向排头元素的前一个位置，因此，从头指针 front 指向的后一个位置直到队尾指针 rear 指向的位置之间，所有的元素均为队列中的元素。

7、确定循环队列中元素个数的方法如下：

设循环队列的容量为 M，

如果 rear>front，则循环队列中的元素个数为 rear-front；

如果 rear<front，则循环队列中的元素个数为 M+(rear-front)；

8、循环队列的运算有两个：

1）入队运算：是在循环队列的队尾加入一个新元素，也就是 rear+1；

2）退队运算：是在先能换队列的排头位置退出一个元素并赋给指定的变量，也就是 front+1。

1.6 线性链表

对于大的线性表或者变动频繁的线性表不宜用顺序存储，应该用链式存储。

1、链式存储方式的特点：

（1）在链式存储结构中，存储数据结构的存储空间可以是不连续的。

（2）各结点的存储顺序与数据元素之间的逻辑关系可以不一致。

2、线性链表：线性表的链式存储结构称为线性链表，是一种物理存储单元上**非连续**、**非顺序**的存储结构，数据元素的逻辑顺序是通过链表中的指针链接来实现的。

3、线性表顺序存储的缺点：

（1）插入或删除的运算效率很低。在顺序存储的线性表中，插入或删除数据元素时需要移动大量的数据元素；

（2）线性表的顺序存储结构下，线性表的存储空间不便于扩充；

（3）线性表的顺序存储结构不便于对存储空间的动态分配。

注意：线性链表是线性表的链式存储结构；带链的栈是栈的链式存储结构；带链的队列是队列的链式存储结构。

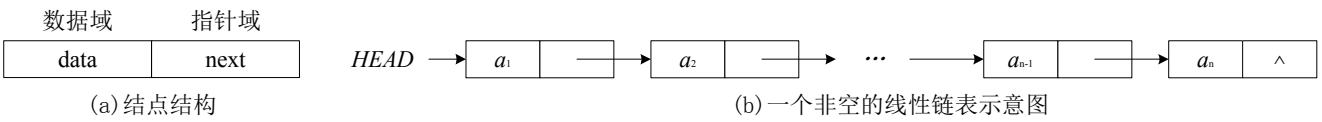
4、线性链表的存储结点的定义：为了适应线性链表的链式存储结构，计算机存储空间被划分为一个一个小块，每个小块占若干个字节，通常称这些小块为存储结点。

存储结点=数据域+指针域

头指针：在线性链表中，头指针（head）很关键，不的丢失。

最后一个结点的指针域：线性链表的最后一个结点的指针域为空（用 NULL 或 0 来表示）

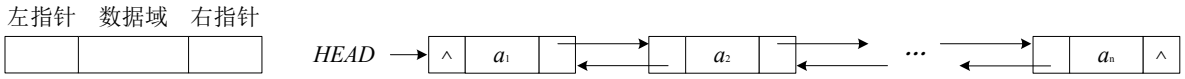
5、在链式存储方式中，每个结点由两部分组成：一部分用于存放数据元素的值，称为数据域；另一部分用于存放指针，称为指针域，用于指向该结点的前一个或后一个结点（即前件或后件），如下图所示：



(a)结点结构　　(b)一个非空的线性链表示意图

线性链表分为单链表、双向链表和循环链表三种类型。

在单链表中，每一个结点只有一个指针域，由这个指针只能找到其后件结点，而不能找到其前件结点。因此，在某些应用中，对于线性链表中的每个结点设置两个指针，一个称为左指针，指向其前件结点；另一个称为右指针，指向其后件结点，这种链表称为双向链表，如下图所示：



(a)结点结构　　(b)一个非空的双向链表示意图

3、线性链表的基本运算

（1）在线性链表中包含指定元素的结点之前插入一个新元素。

*：在线性链表中插入元素时，不需要移动数据元素，只需要修改相关结点指针即可，也不会出现“上溢”现象。

（2）在线性链表中删除包含指定元素的结点。

*：在线性链表中删除元素时，也不需要移动数据元素，只需要修改相关结点指针即可。

（3）将两个线性链表按要求合并成一个线性链表。（4）将一个线性链表按要求进行分解。

（5）逆转线性链表。　　（6）复制线性链表。

（7）线性链表的排序。　　（8）线性链表的查找。

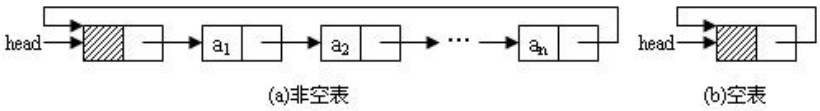
*：线性链表不能随机存取。

4、循环链表及其基本运算

在线性链表中，其插入与删除的运算虽然比较方便，但还存在一个问题，在运算过程中对于空表和对第一个结点的处理必须单独考虑，使空表与非空表的运算不统一。为了克服线性链表的这个缺点，可以采用另一种链接方式，即循环链表。

与前面所讨论的线性链表相比，循环链表具有以下两个特点：1）在链表中增加了一个表头结点，其数据域为任意或者根据需要来设置，指针域指向线性表的第一个元素的结点，而循环链表的头指针指向表头结点；2）循环链表中最后一个结点的指针域不是空，而是指向表头结点。即在循环链表中，所有结点的指针构成了一个环状链。

下图 a 是一个非空的循环链表，图 b 是一个空的循环链表：



(a)非空表　　(b)空表

循环链表的优点主要体现在两个方面：一是在循环链表中，只要指出表中任何一个结点的位置，就可以从它出发访问到表中其他所有的结点，而线性单链表做不到这一点；二是由于在循环链表中设置了一个表头结点，在任何情况下，循环链表中至少有一个结点存在，从而使空表与非空表的运算统一。

*：循环链表是在单链表的基础上增加了一个表头结点，其插入和删除运算与单链表相同。但它可以从任一结点出发来访问表中其他所有结点，并实现空表与非空表的运算的统一。

1.6 树与二叉树

1、树的基本概念

树是一种简单的非线性结构。在树这种数据结构中，所有数据元素之间的关系具有明显的层次特性。在树结构中，每一个结点只有一个前件，称为**父结点**。没有前件的结点只有一个，称为树的根结点，简称树的根。每一个结点可以有多个后件，称为该结点的**子结点**。没有后件的结点称为**叶子结点**。

在树结构中，一个结点所拥有的后件的个数称为该结点的度，所有结点中最大的度称为树的度。树的最大层次称为树的深度。

结点的定义：数据元素+若干指向子树的分支。

结点的度得定义：子树的个数，即拥有的后件个数称为该结点的度。

树的度得定义：树中所有结点的度的最大值。

叶子结点的定义：度为零的结点，即没有后件的结点。

分支结点的定义：度大于非零的结点。

结点的层次（深度）的定义：假设根结点的层次为 1，第 I 层得结点的子树根结点的层次为 I+1。

树的深度的定义：树中叶子结点所在的最大层次。

2、二叉树及其基本性质

（1）什么是二叉树

二叉树是一种很有用的非线性结构，它具有以下两个特点：1）非空二叉树只有一个根结点；2）每一个结点最多有两棵子树，且分别称为该结点的左子树与右子树。

*：根据二叉树的概念可知，二叉树的度可以为零（叶结点）、1（只有一棵子树）或 2（有 2 棵子树）。

（2）二叉树的基本性质

性质 1　在二叉树的第 k 层上，最多有　　个结点。

性质 2　深度为 m 的二叉树最多有个　　2^m-1 个结点$(k\geq 1)$

性质 3　在任意一棵二叉树中，度数为 0 的结点（即叶子结点）总比度为 2 的结点多一个。**性质 4**　具有 n 个结点的二叉树，其深度至少为　　，其中 $[\log_2 n]+1$ 表示取 $[\log_2 n]$ 的整数部分$\log_2 n$

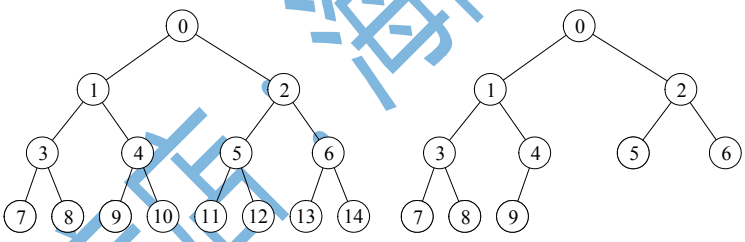
3、满二叉树与完全二叉树

满二叉树：除最后一层外，每一层上的所有结点都有两个子结点。

完全二叉树：除最后一层外，每一层上的结点数均达到最大值；在最后一层上只缺少右边的若干结点。

*：根据完全二叉树的定义可得出：度为 1 的结点的个数为 0 或 1。

下图 a 表示的是满二叉树，下图 b 表示的是完全二叉树：



(a)满二叉树　　(b)完全二叉树

完全二叉树还具有如下两个特性：

性质 5　具有 n 个结点的完全二叉树深度为　　$[\log_2 n]+1$

性质 6　设完全二叉树共有 n 个结点，如果从根结点开始，按层序(每一层从左到右)用自然数 1,2,…，n 给结点进行编号，则对于编号为 k（k=1，2，…，n）的结点有以下结论：

①若 k=1，则该结点为根结点，它没有父结点；若 k>1，则该结点的父结点的编号为 INT(k/2)。

②若 2k≤n，则编号为 k 的左子结点编号为 2k；否则该结点无左子结点（显然也没有右子结点）。

③若 2k+1≤n，则编号为 k 的右子结点编号为 2k+1；否则该结点无右子结点。

4、二叉树的存储结构

在计算机中，二叉树通常采用链式存储结构。

与线性链表类似，用于存储二叉树中各元素的存储结点也由两部分组成：数据域和指针域。但在二叉树中，由于每一个元素可以有两个后件（即两个子结点），因此，用于存储二叉树的存储结点的指针

域有两个：一个用于指向该结点的左子结点的存储地址，称为左指针域；另一个用于指向该结点的右子结点的存储地址，称为右指针域。

*：一般二叉树通常采用链式存储结构，对于满二叉树与完全二叉树来说，可以按层序进行顺序存储。

5、二叉树的遍历

二叉树的遍历是指不重复地访问二叉树中的所有结点。二叉树的遍历可以分为以下三种：

（1）前序遍历（DLR）：若二叉树为空，则结束返回。否则：首先访问根结点，然后遍历左子树，最后遍历右子树；并且，在遍历左右子树时，仍然先访问根结点，然后遍历左子树，最后遍历右子树。

（2）中序遍历（LDR）：若二叉树为空，则结束返回。否则：首先遍历左子树，然后访问根结点，最后遍历右子树；并且，在遍历左、右子树时，仍然先遍历左子树，然后访问根结点，最后遍历右子树。

（3）后序遍历（LRD）：若二叉树为空，则结束返回。否则：首先遍历左子树，然后遍历右子树，最后访问根结点，并且，在遍历左、右子树时，仍然先遍历左子树，然后遍历右子树，最后访问根结点。

1.7 查找技术

查找：根据给定的某个值，在查找表中确定一个其关键字等于给定值的数据元素。

查找结果：（查找成功：找到；查找不成功：没找到。）

平均查找长度：查找过程中关键字和给定值比较的平均次数。

1、顺序查找

基本思想：从表中的第一个元素开始，将给定的值与表中逐个元素的关键字进行比较，直到两者相符，查到所要找的元素为止。否则就是表中没有要找的元素，查找不成功。

在平均情况下，利用顺序查找法在线性表中查找一个元素，大约要与线性表中一半的元素进行比较，最坏情况下需要比较 n 次。

顺序查找一个具有 n 个元素的线性表，其平均复杂度为 O（n）。

下列两种情况下只能采用顺序查找：

1）如果线性表是无序表（即表中的元素是无序的），则不管是顺序存储结构还是链式存储结构，都只能用顺序查找。

2）即使是有序线性表，如果采用链式存储结构，也只能用顺序查找。

2、二分法查找

二分法查找只适用于顺序存储的有序表。在此所说的有序表是指线性表中的元素按值非递减排列。对于长度为n的有序线性表，在最坏情况下，二分查找只需要比较$\log_2 n$次，而顺序查找需要比较n次。

1.8 排序技术

（1）交换类排序法

交换类排序法是指借助数据元素之间的互相交换进行排序的一种方法。冒泡排序法与快速排序法都属于交换类的排序方法。假设线性表的长度为n，则在最坏情况下，冒泡排序需要经过n/2遍的从前往后的扫描和n/2遍的从后往前的扫描，需要的比较次数为n(n-1)/2。在快速排序过程中，随着对各子表不断地进行分割，划分出的子表会越来越多，但一次又只能对一个子表进行再分割处理，需要将暂时不分割的子表记忆起来，这就要用一个栈来实现。

（2）插入类排序法

插入排序是指将无序序列中的各元素依次插入到已经有序的线性表中。在简单插入排序法中，每一次比较后最多移掉一个逆序，因此，这种排序方法的效率与冒泡排序法相同。在最坏情况下，简单插入排序需要n(n-1)/2次比较。

（3）选择类排序法

选择排序法的基本思想是扫描整个线性表，从中选出最小的元素，将它交换到表的最前面（这是它应有的位置）；然后对剩下的子表采用同样的方法，直到子表空为止。简单选择排序法在最坏情况

下需要比较 n(n-1)/2 次。