



《数据结构》总复习



西北大学

第一章 绪论

1.1 理解基本概念

数据、数据元素、数据对象、数据结构

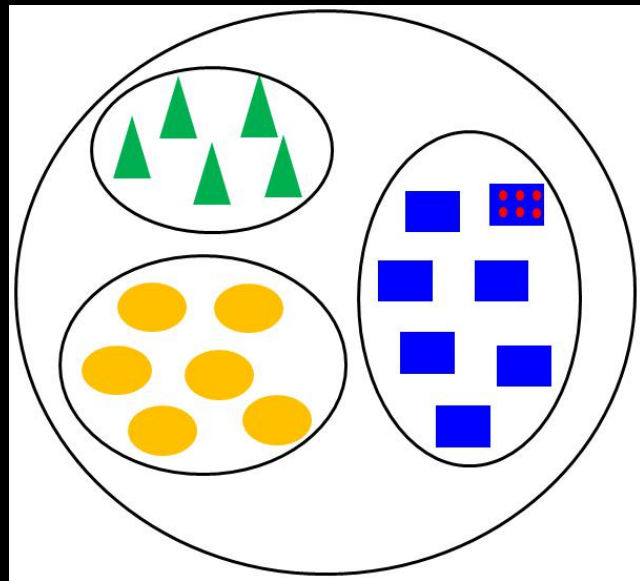
逻辑结构：线性

非线性（集合、树、图）

物理结构：顺序

非顺序（链式、索引、哈希）

ADT：数据对象、元素关系、一组操作



1.2 算法特性与性能分析

- 1、算法的**定义**：解决特定问题的一系列操作。
- 2、算法的5大**特性**（要素）：输入、输出、确定性、可行性和有限性。
- 3、算法**设计要求**：正确性、可读性、鲁棒性、高效率低存储

特性和设计要求不要搞混了哦。

4、算法分析：

时间复杂度 $T(n)$ ：寻找基本操作；计算其执行次数，一般结果为问题规模 n 的多项式；时间复杂度为该多项式的最高次幂。

$T(n)=O(1)$ 的含义：常量时间复杂度，表示算法执行时间与问题规模无关。

空间复杂度 $S(n)$ ：算法执行时所需要的辅助空间。

$S(n)=O(1)$ 的含义：常量空间复杂度，表示算法执行时需要的辅助空间与问题规模无关，也称为算法原地工作。

题1.1 算法的定义与设计要求。

题1.2 数据元素间的逻辑结构关系有哪些。

题1.3 通常从 时间复杂度 和 空间复杂度 来评价算法的优劣。

题1.4 划线部分的语句频度为 ()

int i,j; $\frac{m(m-1)}{2}$

for(i=0;i<m;i++)

for(j=0;j<i;j++) i 次

a[i][j]=i*j;

$$\sum_{i=0}^{m-1} i = \frac{m(m-1)}{2}$$

题1.5 分析以下程序段的时间复杂度为 ()。

void Test() $O(n^2)$
{ s=0;
for(i=1;i<=n;i++)
{
s++;
funB(); //T(n)=O(n)
}
}

第二章 线性表

2.1 线性表的定义、特点

线性表为 $n(n \geq 0)$ 个数据元素的有限序列。其特点为：存在唯一首元素、尾元素，除首元素和尾元素外，其余每个元素只有一个前驱和后继。

2.2 线性表的存储表示

1. 线性表的顺序存储
2. 线性表的链式存储
3. 其他形式的链表：循环单链表、带尾指针的循环链表、双向链表、双向循环链表、静态链表、
4. 根据实际问题，灵活选择存储方式。
5. 各种链表判空条件

不带头结点的单链表、带头结点的单链表

带头结点的单循环链表、带头结点的双向循环链表

6、顺序存储与链式存储的优缺点

- ① 顺序表是静态分配，无需为了表示元素间的关系而增加额外的辅助空间，存储密度大；可按元素序号随机访问；但进行插入、删除时，需要移动大量元素；
- ② 链表是动态分配，需要增加指针域来表示元素间的逻辑关系，存储密度小；不能进行随机访问；插入、删除时不需要移动元素，只需要修改相关指针域即可。

2.3 线性表的基本运算

1、在顺序表中执行插入、删除运算要点：

- ① 插入时，应先判断表是否已满以及插入位置的合法性。在第 i 个位置插入时，需要把从第 i 个位置到最后一个的元素($n-i+1$)向后移动。
- ② 删除时，应先判断表是否空以及删除位置的合法性。删除第 i 个位置元素时，需要把从 $i+1$ 到最后一个的元素($n-i$)向前移动。

2、在单链表中执行插入、删除运算要点：

- ① 插入、删除时，需要找到被插入或删除结点的前驱结点。
- ② 指针更改顺序很重要，防止后继结点丢失。
- ③ 删除的结点要记着释放空间。

2.4 综合应用

算法设计：根据给定需求，在顺序存储或链式存储上实现相应操作。

题2.1 一维数组和顺序表的异同。

同：一维数组与顺序表都可按元素下标直接（或随机）存取元素；

异：

- ① 一维数组中各元素间可以有空元素，而顺序表中的元素必须按顺序存放；
- ② 一维数组的基本操作只有按下标存取，而顺序表可以有线性表的所有操作，如插入、删除等；
- ③ 一维数组的大小一经分配便不可变，而顺序表的长度是可变的。

题2.2 线性表的特点是：除第一个元素外其他每个元素有且只有一个直接前驱，除最后一个元素外其他每个元素都有且只有一个直接后继。那么，循环链表的每一个结点都有直接后继，双向链表的每一个结点都有直接前驱和直接后继，它们还是线性表吗？

解：循环链表和双向链表示存储结构，线性表是逻辑结构。线性与非线性是从逻辑上划分的。因此，循环链表和双向链表是线性表，它们是线性表的不同存储形式。

题2.2 在一个单链表L中，P为中间某结点，在P前插入S结点，可否在 $O(1)$ 时间复杂度内完成。

提示：采用交换方式。

将S插入在P后面，再交换P与S 的值

题2.3 长度为 n 的非空线性表采用顺序存储结构，在表的第 i 个位置删除一个数据元素， i 的合法值应该是（D）

A) $i > 0$

B) $1 \leq i \leq n+1$

C) $1 \leq i \leq n-1$

D) $1 \leq i \leq n$

若该题目改为插入位置，则应该选B

题2.4若线性表最常用的操作是存取第 i 个元素及其前趋的值，则采用（D）存储方式节省时间。

A.单链表

B.双链表

C.单循环链表

D.顺序表

题2.5 某线性表最常用的操作是在最后一个结点之后插入一个结点或删除第一个结点，故采用（D）存储方式最节省运算时间。

A.单链表

B.仅有头结点的单循环链表

C.双链表

D.仅有尾指针的单循环链表

题2.6 链表不具有的特点是 (B)

A.插入、删除不需要的移动元素

B.可随机访问任一元素

C.不必事先估计存储空间

D.所需空间与线性长度成正比

题2.7 设rear是指向非空的带头结点的单循环链表的尾指针，若想删除链表第一个结点，则执行（D）

A. `s=rear; rear=rear->next; free(s);`

B. `rear=rear->next; free(rear);`

C. `rear=rear->next->next; free(rear);`

D. `s=rear->next->next; rear->next->next=s->next; free(s);`

题2.8 在一个长度为 n 的顺序表中第 i 个元素 ($1 \leq i \leq n+1$) 之前插入一个元素时, 需向后移动 $n-i+1$ 个元素。

题2.9 编写算法void Adjust(LinkList L), 其功能是: 在带头结点的单链表L中, 将所有奇数元素放在链表的前半部分, 偶数元素放在链表的后半部分。

题2.10编写算法, PolyNode* Derivative(PolyNode *PL), 其功能是: 求某一元多项式的导数。参数为一元多项式单链表, 返回值为该一元多项式导数的一元多项式链表头指针。(带头结点)

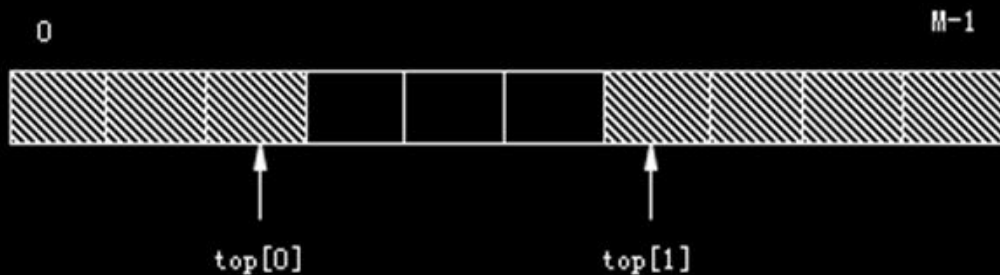
第三章 栈和队列

栈和队列是两种特殊的线性表，它们的逻辑结构和线性表相同，只是其操作的位置受限制。

3.1 栈的定义、存储与基本运算

- 1、只允许在表的末端（栈顶）进行插入和删除的线性表，具有LIFO特性。
- 2、栈的存储结构：顺序存储和链式存储。
- 3、什么时候用栈？

递归？ 函数调用？ 括号匹配？ 表达式求值？ BFS？



4、两栈共享空间：

初始化： $\text{top}[0] = -1$ $\text{top}[1] = M$

栈满： $\text{top}[0] + 1 == \text{top}[1]$

栈空： 1号栈空 $\text{top}[0] == -1$ 2号栈空 $\text{top}[1] == M$

进栈： 根据进哪个栈，具体操作不同

出栈： 根据进哪个栈，具体操作不同

题3.1 设输入序列为1、2、3、4，则借助栈所得到的输出序列不可能是（B）

A.1、2、3、4

B.4、1、2、3

C.1、3、4、2

D.4、3、2、1

题3.2 链式栈只能顺序存取，而顺序栈不但能顺序存取，还能直接存取，这种说法对吗？

题3.3向一个带头结点的栈顶指针为 top 的链栈中插入一个 S 所指结点时, 则执行 ()



A. $top \rightarrow next = S;$

B. $S \rightarrow next = top \rightarrow next; top \rightarrow next = S;$

C. $S \rightarrow next = top; top = S;$

D. $S \rightarrow next = top; top \rightarrow next;$

3.3 栈与递归

- 1、什么是递归：自己定义或调用自己。
- 2、递归的优点：结构清晰、正确性容易证明。
- 3、递归适合条件：原问题可以层层分解为类似子问题，子问题规模更小；最小子问题有解。
- 4、递归的缺点（为什么要消除递归）：时空效率低；无法得到递归过程的某中间状态。

5、递归进层：

- ① 保留本层参数与返回地址；（保存断点，进栈）
- ② 为被调函数的局部变量分配存储空间，给下层参数赋值；
- ③ 转移到被调函数入口。

6、递归退层：

- ① 保存被调函数计算机结果；
- ② 释放被调函数的数据区，恢复上层参数；（出栈）
- ③ 转入到保存的返回地址继续执行。

3.3 队列的定义、存储与基本运算

- 1、允许在表的一端(队尾)进行插入，另一端（队头）进行删除的线性表，具有FIFO特性。
- 2、一串元素依次进队，其出队序列只有一种。
- 3、队列的存储结构：顺序存储和链式存储。

4、循环队列（假设牺牲一个空间来区分队空和队满，front指向队头元素，rear指向队尾的后一个单元）：

① 队满 $(rear+1)\%maxSize==front$ 队空 $front==rear$

② 进队： $rear=(rear+1)\%maxSize$

③ 出队： $front=(front+1)\%maxSize$

④ 元素个数计算： $(rear-front+maxSize)\%maxSize$ 。以上假设front指向队头元素，rear指向队尾元素的后一个单元。

也可用增加一个标志的方法来区分队空和队满

题3.4 栈和队列的共同点是（C）。

A.都是先进后出

B.都是先进先出

C.只允许在端点处插入和删除元素

D.没有共同点

题3.5 线性表、栈和队列都是线性结构，可以再线性表的两端位置插入和删除元素；对于栈只能在一端（栈顶）位置插入和删除元素；对于队列只能在一端（队尾）位置插入元素和在另一端（队头）位置删除元素。

类似题目：栈、队列和线性表有什么异同。

题3.6 排队是日常生活中常见的一种现象，比如：在商店排队付款，当地一位顾客完成付款离开后，其他顾客依次前移。下面用数据结构中的队列来模拟这种排队现象。

```
#define QUEUE 40  
struct Queue {  
    int queue[QUEUE];  
    int Rear;      // Rear记录队列尾  
};
```

如何初始化？进队？入队？

3.4 综合应用

栈：括号匹配、四则运算、进制转换等。

队列：键盘输入缓冲等。

根据实际问题，选择栈或队列完成算法设计。

第四章 串

4.1 串的定义和基本术语

- 1、串的定义：串是特殊的线性表，其特殊性在于组成串的元素为字符。
- 2、空串：长度为0的字符串
- 3、空格串：由空格组成的字符串
- 4、子串：主串的若干连续字符组成的串。
- 5、经典的串模式匹配算法时间复杂度为 $O(mn)$ ，KMP算法的时间复杂度为 $O(m+n)$

题4.1 长度为 n 的字符串，其子串个数为（ B ）。（假设空串不是任何串的子串）

A. n

B. $n*(n+1)/2$

C. $n*(n-1)/2$

D. $n*(n-1)$

第五章 数组和广义表

5.1 数组

1. 数组可以看成是一般线性表的扩充。一维数组即为线性表，而二维数组可以定义为“其数据元素为一维数组(线性表)”的线性表，多维数组依次类推。
2. 数组的基本操作：获取指定位置元素和修改指定位置元素。
3. 数组存储方式：顺序存储，可按行或按列存储

4、一维数组、二维数组和三维数组中某元素地址的计算：

① 一维数组A[1...n]，每个元素占k个字节：

$$\text{Loc}(A[i]) = \text{Loc}(A[1]) + (i-1) * k$$

① 二维数组A[1...m][1...n]，每个元素占k个字节：

按行存储时： $\text{Loc}(A[i][j]) = \text{Loc}(A[1][1]) + ((i-1) * n + j - 1) * k$

按列存储时： $\text{Loc}(A[i][j]) = \text{Loc}(A[1][1]) + ((j-1) * m + i - 1) * k$

③ 三维数组A[1...m][1...n][1...r]，每个元素占k个字节：

按行-列-纵存储时： $\text{Loc}(A[i][j][k]) =$

$$\text{Loc}(A[1][1][1]) + ((i-1) * n * r + (j-1) * r + k - 1) * k$$

按列存储时：

5、特殊矩阵压缩存储

- ① 特殊矩阵：元素分布有规律或非零元素很多的矩阵。如上三角矩阵、下三角矩阵、对称矩阵、带状矩阵、稀疏矩阵
- ② 压缩原则：值相同的元素且分布有规律的元素只分配一个空间；
领元素不分配空间。

$$\begin{bmatrix} a_{1,1} & & & \\ a_{2,1} & a_{2,2} & & \\ \vdots & \vdots & \ddots & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ & a_{2,2} & \cdots & a_{2,n} \\ & & \ddots & \vdots \\ & & & a_{n,n} \end{bmatrix}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & 0 \\ & a_{3,2} & a_{3,3} & a_{3,4} & \\ & & \ddots & \ddots & \ddots \\ 0 & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

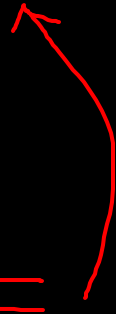
③ 上三角矩阵、下三角矩阵、对称矩阵、带状矩阵一般压缩在一维数组中；

稀疏矩阵一般用三元组或十字链表压缩存储。

③ 掌握压缩后的存储地址或下标计算关系。（注意下标从0还是1开始）

题5.1 什么是特殊矩阵？其压缩原则是什么？

题5.2 三维数组 $a[4][5][6]$ （下标从0开始， a 有 $4*5*6$ 个元素），每个元素的长度是2，则 $a[2][3][4]$ 的地址是_____。（设 $a[0][0][0]$ 的地址是1000，数据以行为主序方式存储）。

$$1000 + (2 \times 5 \times 6 + 3 \times 6 + 4) \times 2 =$$


题5.3 设矩阵A是一个对称矩阵，为了节省存储，将其下三角部分按行序存放在一维数组B[1,n(n-1)/2]中，对任一下三角部分中任一元素 A_{ij} ，在一组数组B的下标位置K的值是（ B ）

A. $i(i-1)/2+j-1$ B. $i(i-1)/2+j$

C. $i(i+1)/2+j-1$ D. $i(i+1)/2+j$

题5.4 稀疏矩阵一般的压缩存储方法有两种，即 ()

- A. 二维数组和三维数组 B. 三元组合散列
C. 三元组和十字链表 D. 散列和十字链表

	row	col	e
1	1	2	12
2	1	3	9
3	3	1	-3
4	3	6	14
5	4	3	24
6	5	2	18
7	6	1	15
8	6	4	-7

该稀疏矩阵有多少行？多少列？

无法确定

5.2 广义表

1、定义：广义表是特殊的线性表，其特殊性在于广义表中的 d_i 既可以是单个元素,还可以是一个广义表。

2、表头：广义表中的第一个元素

表尾：除了第一个元素外，其余元素构成的广义表

3、广义表的存储结构：头尾链、同层结点链。

题5.4 已知广义表 $LS=((a,b,c),(d,e,f))$,运用head和tail函数取出LS中原子e的运算是 (A)。

A. head(tail(head(tail(LS))))

B. tail(head(LS))

C. head(tail(LS))

D. head(tail(tail(head(LS))))

题5.5 广义表 $((a,b,c,d))$ 的表尾是 (B)

A. a B. ()

C. (a,b,c,d) D. ((a,b,c,d))

第六章 树和二叉树

6.1 二叉树

1、二叉树的5个基本性质。

性质1:在二叉树的第 i 层上至多有 2^{i-1} 个结点($i \geq 1$)。

性质2:深度为 k 的二叉树至多有 $2^k - 1$ 个结点 ($k \geq 1$) 。

性质3:对任意一棵二叉树 T ，若终端结点数为 n_0 ，而其度数为2的结点数为 n_2 ，则 $n_0 = n_2 + 1$ 。（需要掌握其证明过程）

完全二叉树中，度为1的结点个数最多1个

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

性质5：对于具有 n 个结点的完全二叉树，如果按照从上到下和从左到右的顺序对二叉树中的所有结点从1开始顺序编号，则对于任意的序号为 i 的结点有：

(1) 如 $i=1$ ，则序号为 i 的结点是根结点，无双亲结点；如 $i>1$ ，则序号为 i 的结点的双亲结点序号为 $\lfloor i/2 \rfloor$ 。

(2) 如 $2 \times i > n$ ，则序号为 i 的结点无左孩子；如 $2 \times i \leq n$ ，则序号为 i 的结点的左孩子结点的序号为 $2 \times i$ 。

(3) 如 $2 \times i + 1 > n$ ，则序号为 i 的结点无右孩子；如 $2 \times i + 1 \leq n$ ，则序号为 i 的结点的右孩子结点的序号为 $2 \times i + 1$ 。

2、二叉树的顺序存储结构和链式存储结构。

顺序存储：将二叉树补为完全二叉树，从上到下，从左到右依次存储每个结点，利用性质5来计算结点之间的关系。

链式存储：用二叉链表来表示。 n 个结点的二叉树，共有 $n+1$ 个空链域， $n-1$ 个分分支。

3、二叉树的遍历与应用

- ①给定二叉树，写出先序、中序和后序序列。
- ②给定二叉树的先序和中序，或者后序和中序，确定二叉树。
- ③二叉树遍历的应用。 算法设计
- ④二叉树的层次遍历。

4、二叉树遍历算法的非递归算法。

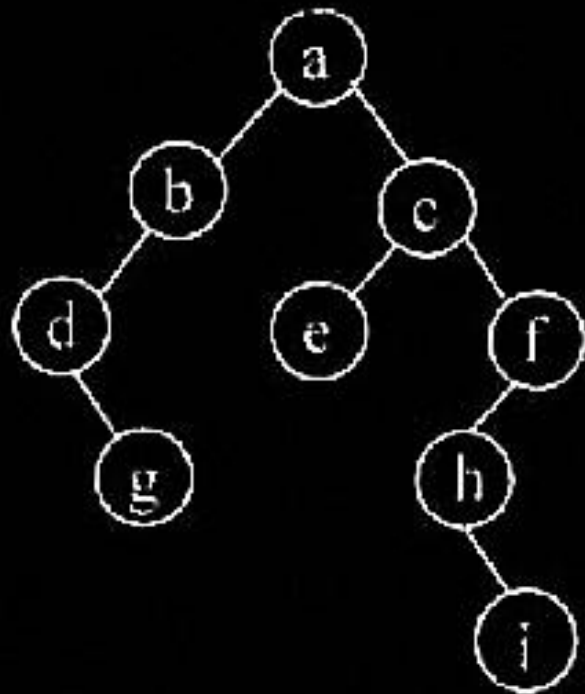
5、线索二叉树（目的；如何判断某结点没有左、右孩子；如何查找某结点的前驱或后继；手工线索化）

题6.1 如图所示的二叉树，回答以下问题：

(1) 其中序遍历序列为_____；

(2) 其前序遍历序列为_____；

(3) 其后序遍历序列为_____。

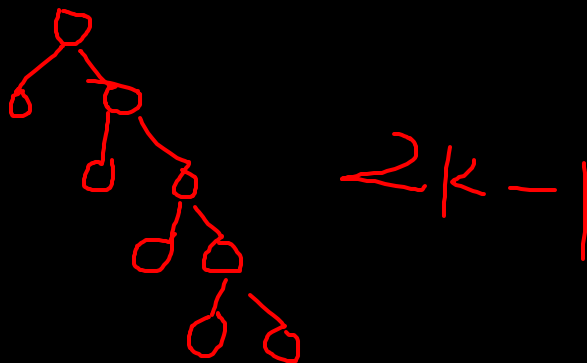


该二叉树的顺序存储？

该二叉树的扩展先序？

题6.2 深度为 k 的二叉树上只有度为0和度为2的结点，则这类二叉树上所含结点总数最少有 $(2^k - 1)$ 个。

题6.3 对任意一棵树，设它有 n 个结点，这 n 个结点的度之和为 $(n - 1)$ 。



题6.5 已知二叉树的前序遍历序列是abdgcefh，中序遍历序列是dgbaechf，则其后遍历序列为（D）

A.bdgcefha B.gdbecfha C.bdgaechf D.gdbaehfc

题6.6 在一棵非空二叉树中，叶节点的总数比度为2的节点总数多（C）个。

A.-1 B.0 C.1 D.2

题6.7 设树T的度为4，其中度为1,2,3和4的结点个数分别为4,2,1,1。

则T中的叶子数为 (D)

A.5

B.6

C.7

D.8

$$\begin{aligned}n_0 &= 3n_4 + 2n_3 + n_2 + 1 \\&= 3 \times 1 + 2 \times 1 + 2 + 1 \\&= 8\end{aligned}$$

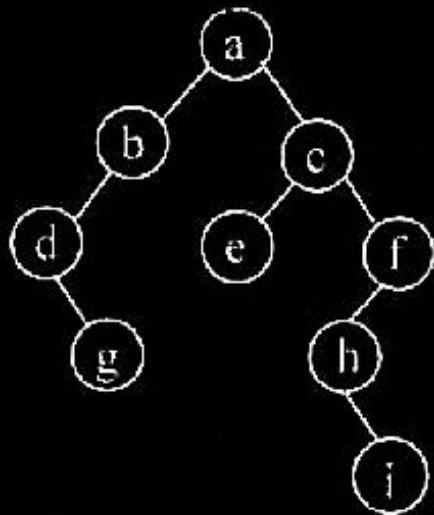
题6.8 将算术表达式 $((a+b)+c*(d+e)+f)*(g+h)$ 转化为二叉树。

题6.9 表达式 $A+((B*C-D)/E+F*G/H)+I/J$ 的后缀表达式是_____。

中缀表达式相当于对二叉树表达式进行中序遍历；

后缀表达式相当于对二叉树表达式进行后序遍历。

题6.10 已知一棵二叉树的前序遍历的结果是ABDGCEFHI，中序遍历的结果是DGBAEC HIF，画出这棵二叉树，以及对二叉树进行后序线索化后的后继线索（用带箭头的线表示）。



6.11 算法设计

- ① 统计二叉树中所有度为1的结点的个数；
- ② 给二叉树中的每个结点设置其层次；
- ③ 在中线线索二叉树中查找某结点P的前驱结点。

6.2 树

1. 树的三种存储形式：双亲表示法、孩子链表示法、孩子兄弟表示法

2. 树和二叉树之间的转换

树的先根遍历《=》二叉树的先序遍历

树的后根遍历《=》二叉树的中序遍历

3. 树和森林之间的转换

森林的先序、中序和后序遍历与二叉树的先序、中序和后序遍历一一
对应

6.3 Huffman树

- 1、Huffman树的构造过程
- 2、前缀码概念
- 3、计算带权路径长度WPL。

题6.8 下面几个符号串编码集合中，不是前缀编码的是（B ）

A. {0, 10, 110, 1111}

B. {11, 10, 001, 101, 0001}

C. {00, 010, 0110, 1000}

D. {b,c,aa,ac,abs,abb,abc}

题6.9 假设字符a,b,c,d,e的频度分别为34%, 14%, 25%, 12%, 15%, 计算Huffman编码。请写出各个符号的huffman编码及编码树。（要求Huffman树的右子树不大于左子树，且左子树标为0）。

第七章 图

7.1 图的基本术语与存储结构

1. 基本术语：有向图、无向图、稠密图、稀疏图、连通图、连通分量、强连通图、强连通分量、简单路径、出度、入度、度、生成树、网。
2. 图的邻接矩阵存储方式与基本操作
 - (1) 采用两个数组来表示图：一个是用于存储顶点信息的一维数组，另一个是用于存储图中顶点之间关联关系的二维数组（邻接矩阵）。

(2) 无向图的邻接矩阵是对称的，有向图的邻接矩阵不一定对称。
图的邻接矩阵存储结构所需空间与顶点 n 有关系，其空间复杂度为 $O(n^2)$ ，与边 e 无关。因此适合存稠密图。

(3) 基本操作

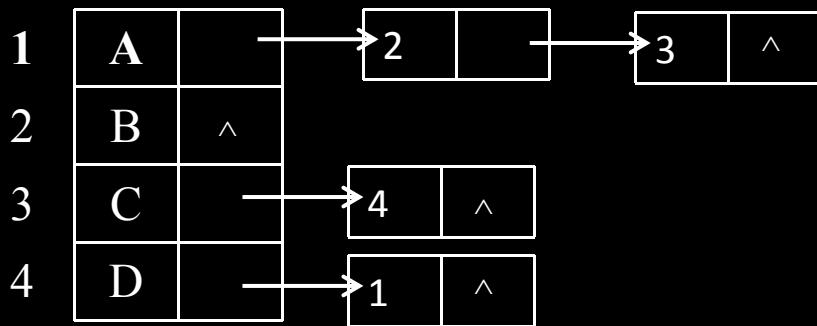
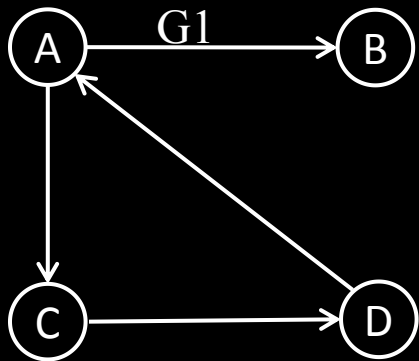
有向图中求顶点 i 出度：第 i 行中非0且非无穷的元素个数

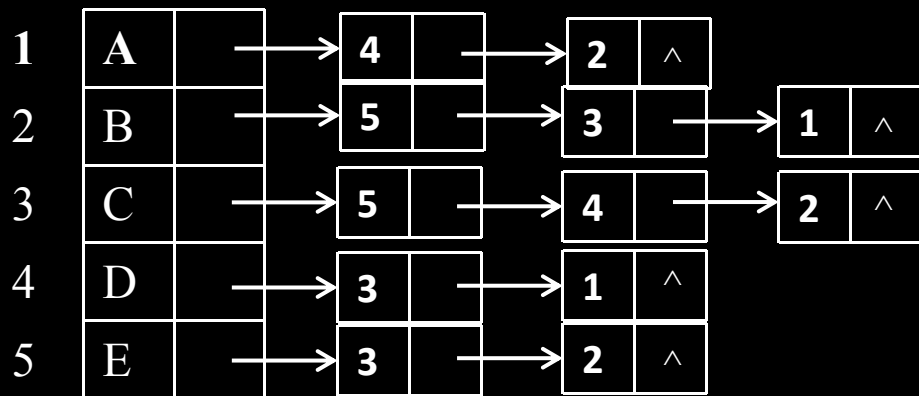
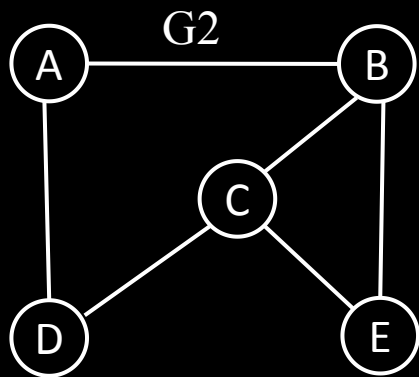
有向图中求顶点 i 入度：第 i 列中非0且非无穷的元素个数

无向图中求顶点 i 的度：第 i 行(列)中非0且非无穷的元素个数

3. 图的邻接表存储方式与基本操作

(1) 基本思想：只存有关联的信息。表头结点表和边表。如图：





当顶点在图中的位置确定后
 图的邻接矩阵表示法唯一的
 图的邻接表表示法是不唯一的。

(2) 有向图的边只存一次，无向图的边存2次。图的邻接表存储结构所需空间与顶点 n 和边 e 都有关系，其空间复杂度为 $O(n+e)$ ，因此适合存稀疏图。

(3) 基本操作

有向图中求顶点 i 出度：第 i 个结点的边表结点个数。

有向图中求顶点 i 入度：遍历所有结点的边表结点，统计邻接点为 i 的结点个数。

无向图中求顶点 i 的度：第 i 个结点的边表结点个数。

由于求入度效率低，可以建立逆邻接表。

题7.1 有 n 个结点的有向图的边数最多有 (B)

A. n B. $n(n-1)$ C. $n(n-1)/2$ D. $2n$

题7.2 在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和的__1__倍。

题7.3 假定一个图具有 n 个顶点和 e 条边，则采用邻接矩阵表示时，其相应的空间复杂度为__ $O(n^2)$ __。

7.2图的遍历及其应用

知识点:

1、图的深度优先遍历

类似于二叉树的先根遍历，通常用递归或栈来实现。

2、图的广度优先遍历

类似于二叉树的层次遍历，通常用队列来实现。

树和图都提到遍历。

遍历： 把树或图中的每个结点访问且仅访问一遍。

树的遍历的方法：二叉树有先序、中序、后序、层次遍历，可采用递归或非递归实现；树有先根、后根、层次遍历。

图在遍历时因为结构复杂，需要设置访问标志数组visited[]来对每个结点的访问状态进行标记。图有DFS和BFS。

图的DFS相当于树的先根遍历；BFS相当于树的层次遍历。

- ① 根据图的邻接矩阵或邻接表存储结构，给出图的逻辑结果；
- ② 根据图中顶点间的关系，给出图的邻接矩阵或邻接表存储结构；
- ③ 从某点出发的图的深度或广度优先遍历序列、广度或深度优先生成树。
- ④ 在邻接矩阵和邻接表上的图的深度和广度优先遍历算法。

3、图的应用

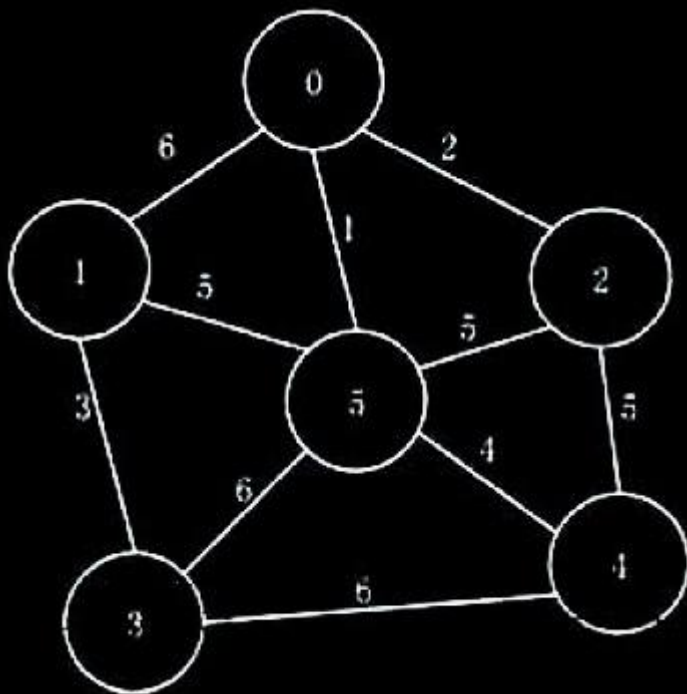
- (1) 最小生成树 (Prim算法和Kruskal算法)
 $O(n^2)$ $O(e \log_2 e)$
- (2) 拓扑排序
- (3) 关键路径
- (4) 最短路径

题7.4 关键路径是AOE网中从源点到汇点的（最长）路径。

题7.6 已知二维数组表示的图的邻接矩阵如下图所示。试分别画出自顶点1出发进行遍历所得的深度优先生成树和广度优先生成树。

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	1	0	1	0
2	0	0	1	0	0	0	1	0	0	0
3	0	0	0	1	0	0	0	1	0	0
4	0	0	0	0	1	0	0	0	1	0
5	0	0	0	0	0	1	0	0	0	1
6	1	1	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	1
8	1	0	0	1	0	0	0	0	1	0
9	0	0	0	0	1	0	1	0	0	1
10	1	0	0	0	0	1	0	0	0	0

题7.7 用克鲁斯卡尔算法将下列的图构造成最小生成树，画出生成过程。



第八章 查找

8.1 基于线性表的查找

1、查找成功的平均查找长度 $ASL = \sum_{i=1}^n P_i C_i$

2、顺序查找：既适合顺序表，也适合链表。查找成功的平均查找长度为 $(n+1)/2$ 。

3、折半查找：要求元素有序且顺序存储，用折半判定树来分析其查找性能。其平均查找长度为 $\log_2 n$ 。

4、分块查找：数据分块后，块内有序，块间无序，每块的最大或最小元素组成索引表。索引表由于有序，可按折半查找，块内因为无序，只能顺序查找。其查找性能由两部分组成。

题8.1 链表适用于（ A ）查找。

A.顺序 B.二分法

C.顺序，也能二分法 D.随机

题8.2 有一个长度为12的有序表，按二分查找法对该表进行查找，在表内各元素等概率情况下查找成功所需的平均比较次数为（ B ）

A.35/12 B.37/12 C.39/12 D.43/12

题8.3 已知有序表为(13,19,24,35,47,50,62,83,95,115,138)，当用二分法查找19时，需（4）次查找成功。（中间位置计算时下取整）

题8.4 折半查找的前提条件是什么。

题8.5 分块查找的基本思想。

8.2 基于树的查找

1、二叉排序树的定义：二叉树排序树或者是一棵空树，或者是具有如下性质的二叉树：

（1）若它的左子树非空，则左子树上所有结点的值均小于根结点的值；

（2）若它的右子树非空，则右子树上所有结点的值均大于（或大于等于）根结点的值；

（3）它的左右子树也分别为二叉排序树。

- 2、给定一个序列，能构建一棵二叉排序树，并分析其查找性能。
- 3、二叉排序树按照中序遍历，会得到一个非递减的序列。
- 4、二叉排序树的平均查找性能与折半类似 $O(\log_2 n)$ ，但其插入和删除元素时不需要移动元素；但在最坏情况下（初始序列有序时），会蜕化为顺序查找 $(n+1)/2$ 。
- 5、平衡二叉排序树的定义：每个结点的左右子树高度之差的绝对值小于或等于1。

平衡因子：左右子树高度之差

失衡类型：LL, LR, RR, RL

题8.6 设有一组记录的关键字为 $\{19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79\}$ ，构造二叉排序树，并计算等概率情况下，查找成功的平均查找长度。

题8.7 平衡因子的取值范围是 0, -1, 1。

8.3 哈希查找

1、基本思想：在元素的关键字 k 和元素的存储位置 p 之间建立一个对应关系 H ，使得 $p=H(k)$ ， H 称为哈希函数。创建哈希表时，把关键字为 k 的元素直接存入地址为 $H(k)$ 的单元；以后当查找关键字为 k 的元素时，再利用哈希函数计算出该元素的存储位置 $p=H(k)$ ，从而达到按关键字直接存取元素的目的。

2、哈希函数：除留余数法考的多

3、处理冲突的方法：线性探测再散列、二次探测再散列、再哈希法和链地址法。

4、装填因子= $\frac{\text{哈希表中元素个数}}{\text{哈希表的长度}}$

5、查找成功和不成功的平均查找长度

6、影响哈希查找效率的因素：

哈希函数；处理冲突的方法；装填因子。

题8.6 在各种查找方法中，平均查找长度与结点个数 n 无直接关系的查找方法是 哈希查找。

题8.7 设有一组记录的关键字为 $\{19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79\}$ ，散列函数为 $H(\text{key}) = \text{key} \text{ MOD } 13$ ，采用线性探测再散列解决冲突，构造哈希表，并计算查找成功与不成功的平均查找长度。

第九章 排序

9.1 排序基本概念

1、内部排序与外部排序：**内部排序**是整个排序过程完全在内存中进行；当待排序记录数据量太大时，内存无法容纳全部数据，排序需要借助外部存储设备才能完成，称为**外部排序**。

2、排序的稳定性。

是否稳定的判断方法：排序过程中，需要交换或移动数据。若交换或移动元素时，中间可能会隔若干个元素时（空运移动），这种排序方法必定不稳定，如希尔、简单选择、快速和堆。

理解稳定排序的意义。

9.2简单排序

- 1、直接插入排序、冒泡排序、简单选择排序。
- 2、掌握每种排序算法基本思想，并能针对给定的某序列，写出完整排序过程。
- 3、三种排序算法的时间复杂度均为 $O(n^2)$ ，空间复杂度为 $O(1)$ 。
- 4、直接插入和冒泡排序的最好情况均为元素已经有序，最坏情况为逆序。简单选择排序与元素初始序列的排列无关。

9.3 改进排序

- 1、希尔排序、快速排序、堆排序、归并排序
- 2、掌握每种排序算法基本思想，并能针对给定的某序列，写出完整排序过程。
- 3、希尔的最好情况是元素有序；最坏情况是元素逆序；
堆、归并算法性能和元素的初始序列无关。

快速排序的最坏情况是元素正序或逆序，最好情况是中轴元素每次能将待排序元素划分成两个规模相当的子序列。一般元素越乱性能会较好。

4、希尔排序的时间复杂度为 $O(n^{1.5})$,快速排序的平均时间复杂度和堆、归并排序的时间复杂度均为 $(n\log_2 n)$,但快速排序在最坏情况下的时间复杂度为 $O(n^2)$

5、希尔排序的空间复杂度为 $O(1)$,快速排序的空间复杂度为 $O(\log_2 n)$,堆排序的空间复杂度为 $O(1)$,归并排序的空间复杂度为 $O(n)$ 。

题9.1 给定关键字序列{33,39,24,115,27,51,62,83,95, 15, 38}, 写出直接插入、简单选择、冒泡的前3趟排序结果。

题9.2 关键字序列{33,39,24,115,27,51,62,83,95, 15, 38}, 以第一个元素为中轴, 一趟快速排序后的结果为

15,27,24,33,115,51,62,83,95,39,38。

题9.3 关键字序列{33,39,24,115,27,51,62,83,95, 15, 38}, 用堆排序进行从小到大排序, 建初堆时, 应该从27元素开始筛选。

题9.4 快速排序在下列（C）情况下最易发挥其长处。

A.被排序的数据中含有多个相同排序码

B.被排序的数据已基本有序

C.被排序的数据完全无序

D.被排序的数据中的最大值和最小值相差悬殊

题9.5 在堆排序和快速排序中，若原始记录接近正序或反序，则选用 堆排序，若原始记录无序，则最好选用 快速排序。

题9.6 在对一组记录（54,38,96,23,15,72,60,45,83）进行直接插序排列时，当把第7个记录60插入到有序表时，为寻找插入位置需比较3次。

题9.7 排序算法中的比较次数与初始元素序列的排列无关的排序算法有简单选择，堆，归并。

题9.8 分别采用堆排序，快速排序，冒泡排序和归并排序，对初态为有序的表，则最省时间的是冒泡排序算法，最费时间的是快速排序算法。

题9.9 有一种排序算法，其时间复杂度为 $O(n^2)$ ，关键字比较次数与待排序记录的初始排列顺序无关且排序不稳定，则该排序算法是简单选择排序。

题9.10 对记录 (54,38,96,23,15,72,60,45,83) 从小到大进行排序：

- (1) 写出用第一个元素做基准的一趟快速排序结果。
- (2) 图示建好的初堆，及2趟堆排序结果。