第一章 绪论

逻辑结构：线性结构（2 线性表；3 栈和队列；4 数组和广义表；5 串）和非线性（6 树；7 图）

8 查找 ；9 排序

存储结构：顺序存储、链式存储、索引存储、散列存储

顺序存储和链式存储的优缺点：

- 顺序存储优点：存取方式是随机的，读取数据比较方便

  缺点：插入和删除操作需要移动大量的数据
- 链式存储方式的优缺点与顺序相反。

判断算法好坏的标准：时间复杂度和空间复杂度

第二章 线性表

线性表：由 n（n>=0）个相同数据类型的元素组成的有限序列。

逻辑特征：（1：1）

- 第一个结点是开始结点，无前驱有唯一的后继
- 最后一个结点是终端节点，无后继有唯一的前驱
- 中间结点，有唯一前驱和唯一的后继

逻辑结构的定义方式：（元素 1，元素 2，。。。）

存储结构：（物理结构）

- 顺序存储(用数组体现)：预先分配一段连续的区域，静态分配

  int a[10]; a 表示 a[0]的地址

  顺序表：两层含义：逻辑结构上线性结构

  物理上顺序存储

- 链式存储(用指针):占用的区域不一定连续,动态分配;malloc free

链表：两层含义：逻辑结构上线性结构

物理上链式存储

顺序表的插入：maxsize 表示最多存放的元素的个数，size 表示有效元素的个数

1、 判断满不满：满的条件：size= =maxsize

2、 插入的位置是不是合法（i>=1&&i<=size+1）

3、 后移（先移动后面的）

4、 插入 size++;

顺序表的删除：

1、 判断空间是否是空 （size= =0）

2、 判断位置是不是合法 （1<=i<=size）

3、 前移（先移动前面的）

4、 Size- -

链表：

分类：单链表、循环链表、双链表

三个术语：头结点、头指针、开始结点

插入操作：p 结点的后面

1、 生成空间：s=(S *)malloc(sizeof(S));

2、 赋值：s->data=X;

3、 s->next=p->next;(修改新结点的指针)

4、 P->next=s;

链表的删除：p 结点的后面的结点

1、 q=p->next;

2、 p->next=q->next; //p->next=p->next->next;

3、 free(q);

删除 p 结点本身：

A、 寻找 p 结点的前驱结点,转化为删除 p 结点的前驱结点的后继

```
q=head;
while(q->next!=p)
    q=q->next;
q->next=p->next;
free(p);
```

B、 删除 p 的后继（p 结点不是终端结点，p 一定要有后继）

```
q=p->next;
p->data=q->data;
p->next=q->next;
```

free(q);

第三章 栈和队列

栈和队列是特殊的线性表。特殊在插入和删除操作都是在表的端点处进行。

栈 Stack：操作受限（插入和删除）的线性表。

在线性表的末尾做插入和删除，叫做栈顶 top　线性表的开始部分叫做栈底 bottom

特点：FILO

典型应用：函数的调用

存储方式：

- 顺序存储（用数组体现，静态分配存储空间）
- 链式存储（涉及到指针，动态分配存储空间）

顺序栈的插入：top 表示栈顶元素的下标，定义是 int 类型的，格式是 int top；maxsize 表示空间的大小，最多存放多少个元素

(在顺序表的基础上修改的，红颜色粗体部分去掉)

1、 判断满不满：满的条件是 top= = maxsize-1

~~2、 插入的位置是不是合法（i>=1&&i<=size+1）：去掉 原因是位置固定 size+1~~

~~3、 后移（先移动后面的）：去掉~~

4、 top++；插入（data[top]=X）

顺序栈的删除：(在顺序表的基础上修改的，红颜色粗体部分去掉)

1、 判断空间是否是空 （top= =-1 top<0）

~~2、 判断位置是不是合法 （1<=i<=size）~~

~~3、 前移（先移动前面的）~~

4、 top - -

链栈的插入操作：top 表示栈顶的指针，类似于 head，top 是指针变量 格式：数据类型 *top;

1、 生成空间：s=malloc（。。。。）;

2、 赋值：s->data=X;

3、 修改指针：s->next=top;(先修改的是新结点的指针)

4、 修改 top 指针：top=s;

链栈的删除：

1、 p=top;

2、 top=top->next;

3、 free(p);

进栈的顺序是 123，则可能的出栈的顺序是 123、132、213、231、~~312~~、321

进栈的顺序是 1234，则可能的出栈的顺序是 1234、1243、1324、1342、1432、2134、2143、2314、2341、2431、3214、3241、3421、4321

队列：

1、 定义：操作受限的线性表（在表的一端进行插入、在另一端进行删除）

2、 在线性表的末尾进行插入操作，叫做队尾 rear

在线性表的开始做删除操作，叫做队头 front

3、 特点：FIFO

4、 存储结构：

- 顺序存储（用数组体现，静态分配存储空间）
- 链式存储（用指针体现，动态分配存储空间）

入队列的顺序是 1234，则出队的顺序是 1234。

循环队列：队列的顺序存储结构

循环队列长度是 maxsize，则最多存放 maxsize-1 个元素

- Front 表示：队头元素的前一个位置
- Rear：队尾元素的下标

循环队列空的条件是：front= = rear

循环队列满的条件是：front==(rear+1)%maxsize

循环队列的元素的个数是：(rear-front+maxsize)%maxsize

循环队列的队头元素的下标：（front+1）%maxsize

备注：凡涉及到+1，都要%maxsize

顺序队列的插入：(在顺序表的基础上修改的，红颜色粗体部分去掉)

1、 判断满不满：满的条件：front= =（rear+1）%maxsize

2、 插入：rear=(rear+1)%maxsize

data[rear]=X;

顺序队列的删除：(在顺序表的基础上修改的，红颜色粗体部分去掉)

1、 判断空间是否是空：空的条件是 front= =rear

2、 删除：front=(front+1)%maxsize

链队列：队列的链式存储结构

Front 表示的队头元素的指针

Rear 表示队尾的指针

链队列的插入操作：

1、 p=malloc(…);

2、 p->data=X;

3、 p->next=NULL;（先修改新产生结点的指针）

4、 rear->next=p;

5、 rear=p;

链队列的删除:（带头结点的队列）

一、 删除真正的队头元素

1、 判断队列是否为空:空的条件是 front= = rear

2、 s=front->next;

3、 front->next=s->next;

4、 free(s);

二、 通过删除头结点 转化为删除队头元素

1、 判断队列是否为空:空的条件是 front= = rear

2、 s=front;

3、 front=front->next;

4、 free(s);

练习：

(10,20,30)

1、 线性表，画出带头结点的单链表

2、 栈，分别画出顺序存储结构 链式存储结构

3、 队列，分别画出顺序存储结构 链式存储结构

第四章 数组和广义表

数组：两种操作（查找和修改）

两种存储结构：按行优先、按列优先

运算：计算地址、计算按行优先时的地址按列优先时是哪个元素

广义表：（不考）

- 表头一定是原子。　（×）
- 表头一定是广义表。　（×）
- 表尾一定是原子。（×）
- 表尾一定是广义表。（√）

第五章 串

两个串相等的充要条件是：串长相等并且对应位置上的字符要相同。

空串和空白串区别：

串的运算：串的链接、求子串、求子串在主串中的位置（模式匹配）、串的比较、串的复制、串的替换 等等。

Int a[]=”123”;　（√）

Int a[10]; a=”123”;　（×）

Char name[10]=”gmm”;

For(i=0;i<size;i++)

If(strcmp(name,stu[i].name)= =0)

练习：

输出所有的“水仙花数”。所谓“水仙花数”是指一个 3 位数，其各位数字立方和等于该数本身。例如：153 是一个水仙花数，应为 $153=1^3+5^3+3^3$。

第六章 树

树：非线性结构

1、定义：有 n（n>=0）个结点组成的有限集合。对于非空树来说：有且仅有一个根结点；子树由 t1，t2.。。互斥的集合。

2、术语：结点的度、树的度、树的高度、路径、路径的长度、树的路径长度、树的带权路径长度

3、树的存储：双亲表示法、孩子表示法、双亲孩子表示法、孩子兄弟表示法

二叉树：

判断：二叉树是树；是度为 2 的树；是特殊的树；是度为 2 的有序树。（×）

5 个性质：

性质 1：二叉树第 i 层上最多 2^{i-1}（数学归纳法）

性质 2：深度为 k 的二叉树上最多 2^k-1（等比数列）

性质 3：二叉树上度为 0 的节点有 n0 个，度为 2 的节点有 n2 个，

则 n0=n2+1　　（N=n0+n1+n2=0*n0+1*n1+2*n2+1）

满二叉树 完全二叉树

- 满二叉树一定是完全二叉树。（√）
- 完全二叉树一定是满二叉树。（×）
- 完全二叉树的结点如果有左孩子，则它一定有右孩子。（×）
- 完全二叉树的结点如果有右孩子，则它一定有左孩子。（√）

性质 4：有 n 个节点的完全二叉树，深度为：。。。。。（假设深度 k，$2^{k-1}-1+1 \leqslant n \leqslant 2^{k}-1$）　　「　　」

性质 5：把完全二叉树编码，从上到下，同一层上从左向右，

- I=1：是根结点，没有双亲；i>1：有双亲，双亲编码是」i/2」；
- I:2i 是左孩子，如果存在右孩子，则右孩子的编码是 2i+1；
- I:2i+1 是它的右孩子，左孩子的编码是 2i。

练习：

1、有 1001 个结点的完全二叉树，树的高度为（ ），其中叶子结点的个数为（ ），度为 1 的结点的个数为（ ），度为 2 的结点的个数为（ ）。

2、已知二叉树有 52 个叶子结点，度为 1 的结点个数为 30 则总结点个数为（）。

二叉树的存储：

- 顺序存储---→完全二叉树的存储（下表为 0 的空间没用　性质 5 来判断双亲和孩子的关系）
- 链式存储（二叉链表、带双亲的二叉链表（三叉链表））-→

一般二叉树

二叉树的遍历：先序遍历（第一个节点是根结点）、中序遍历（来判断左右子树的结点）、后序遍历（最后一个节点是根结点）

根据先序和中序、中序和后序得到二叉树。

练习：

1、已知二叉树的中序遍历序列是 ACBDGHFE，后序遍历序列是 ABDCFHEG，请构造一棵二叉树。

2、已知二叉树的层次遍历序列为 ABCDEFGHIJK，中序序列为 DBGEHJACIKF，请构造一棵二叉树。

3、已知二叉树的前序、中序和后序遍历序列如下，其中有一些看不清的字母用*表示，请先填写*处的字母，再构造一棵符合条件的二叉树。

(1)前序遍历序列是：*BC***G*

(2)中序遍历序列是：CB*EAGH*

(3)后序遍历序列是：*EDB**FA

树、森林和二叉树之间的转换：

- 树----→二叉树：加线、抹线、调整（结论：转换后的二叉树没有右子树）
- 二叉树-----→树：加线、抹线、调整
- 森林-----→二叉树：
- 二叉树-----→森林：

树、森林和二叉树的遍历

1、树的先跟遍历次序与它所对应的二叉树的先序遍历次序相同；

2、树的后跟遍历次序与它所对应的二叉树的中序遍历次序相同；

3、森林的先序遍历次序与它所对应的二叉树的先序遍历次序相同；

4、森林的中序遍历次序与它所对应的二叉树的中序遍历次序相同；

哈弗曼树及其编码：

- 术语：路径、路径的长度、树的路径长度（从根结点到每一个叶子结点的路径长度之和）、带权路径长度（根结点到叶子结点的路径长度*权值　和）
- 构造：每次找两个权值最小的结点进行合并，得到的新节点的权值放到最后，循环合并，直到只剩下一个结点的时候结束过程，这时候的二叉树就是所求的最优二叉树（哈夫曼树）。
- 哈弗曼编码：左分支为 0　右分支为 1，从根结点到叶子结点所经过的边的编码就是它所对应的哈弗曼编码。

第七章 图

1、定义：

- 图：Graph 两个集合 V(vertex)　E
- 有向图：V E　　<V,W >　弧头（箭头指向谁　终点）弧尾（起点）
- 无向图：　（V,W）==（W,V）

2、术语：

- 完全图（任意两个定点之间都有边　无向图 n*（n-1）/2 有向图 n*(n-1)）
- 结点的度：与这个顶点相关联的边的条数
- 入度：针对的有向图　以这个顶点为弧头
- 出度：针对有向图　以这个顶点为弧尾
- 边的个数：所有顶点的度的总和/2
- 连通图：

如果两个顶点之间有路径，那么这两个顶点就是连通的

连通图：无向图 任意两个顶点之间都有路径

判断：　完全图一定是连通图（√）

连通图一定是完全图（×）

- 连通分量：非连通图的极大连通子图
- 强连通图 强连通分量 ：有向图
- 生成树：连通图，有 n 个顶点，至少需要 n-1 边使得它连通

连通图的极小连通子图

3、　　图的存储：

（1）邻接矩阵：

如果有 n 个顶点的无向图，存储空间压缩 n*(n+1)/2

如果有 n 个顶点的有向图，存储空间不能压缩 n*n

有向图的连接矩阵　　第 i 行非零元素的个数------→出度

第 i 列非零元素的个数------→入度

无向图的连接矩阵：第 i 行（或者列）非零元素的个数------→度

（2）邻接表：

无向图中顶点 Vi 的度是邻接表中结点的个数；

有向图中，顶点 Vi 的出度是邻接表中结点的个数；

顶点 Vi 的入度是逆邻接表中结点的个数。

图的遍历：深度遍历 广度遍历

生成树：深度遍历的生成树　广度遍历生成树

最小生成树：

- Prim：把顶点分为两个集合，分别从这两个集合中各找出一个顶点，使得这两个顶点之间的边的权值是最小的，把绿颜色顶点合并到黄颜色中，继续。。。，直到所有顶点都合并到黄颜色中为止。
- Kruscal：把顶点分别作为集合，每次从边的集合中查找权值最小的两个顶点：如果这两个顶点分属于不同集合，合并；如果这两个顶点分属于同一集合，继续查找下一条权值较小的边。

最短路径：迪杰斯特拉算法

- 第一条最短路径：在所有的直达路径中，找出最短的一条路径
- 在后面的最短路径：要么直达；要么经过已经求出的最短路径的顶点。

拓扑结构：主要用于查看工序之间的前驱后继的关系。AOV 网

第八章 查找

1、线性表的查找（静态查找）

- 顺序查找：顺序表 链表
- 折半查找（二分查找）：有序的顺序表

2、树表的查找（动态查找）：二叉排序树（二叉查找树）

（1）定义：可以是空树，如果非空，满足 3 个条件。。。。

（2）查找

（3）插入（作为叶子结点插入）

（4）删除（3 种情况）：

- 删除叶子结点：直接删除，修改其双亲的左指针或者右指针为 NULL
- 删除只有左子树或者只有右子树：修改其双亲直接指向其左孩子或者右孩子
- 删除既有左子树又有右子树：用中序遍历的前驱结点代替删除结点，然后删除前驱结点。

第九章 排序

1、定义：

分类：内部排序、外部排序

稳定性：

2、重点排序方法

- 插入排序：直接插入排序（稳定）、希尔排序（不稳定）
- 交换排序：冒泡排序（稳定）、快速排序（不稳定）
- 选择排序：简单选择排序（稳定）
- 归并排序：2 路归并排序（稳定）

Memory：

Storage：

CPU:Centrol Processing Unit

OS:Operating System

操作系统的功能：作业管理 进程管理 存储管理 文件管理 设备管理