物理存储、逻辑结构及运算

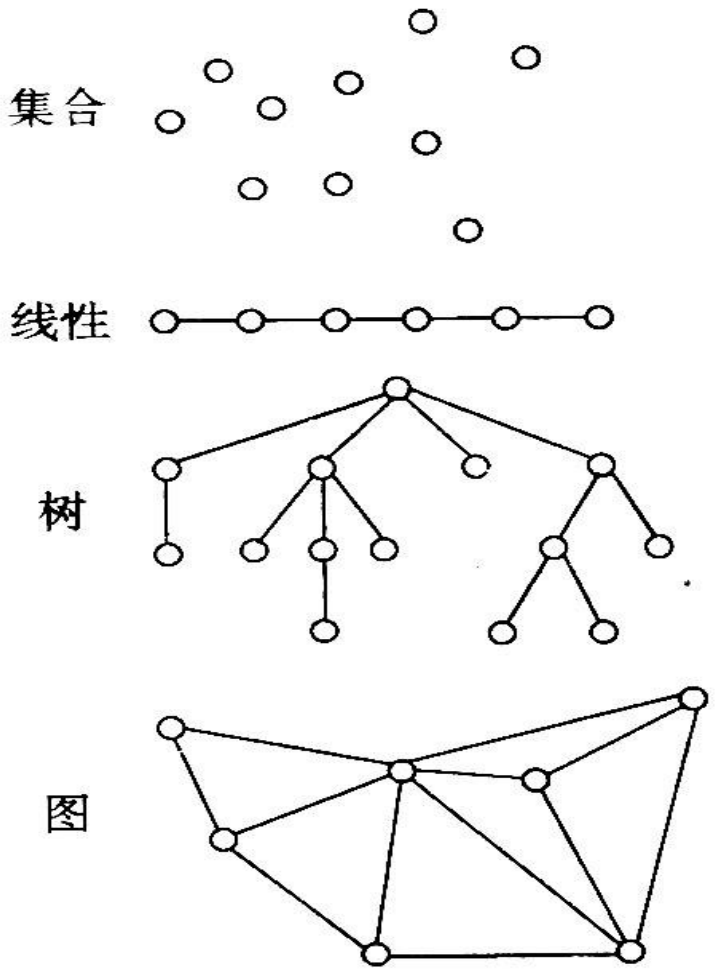
程序=算法+结构

--正确、健壮、效率、低存储

1. 顺序的
2. 链式的
3. 散列的
4. 索引的—B树

从物理存储看逻辑概念

- 相同的物理存储，不同的逻辑结构及其运算
  - 物理上是顺序存储的，但逻辑上可以表示集合、线性表、树和图
- 相同的逻辑结构，不同的运算



TSMC@2010, Rong Chen, All rights reserved.

数组

- 数组逻辑上可以表示集合、线性表、树和图

```
typedef struct {
   ElemType *elem;
   int length;
   int listsize;
}SqList; //P23
```

```
typedef struct {
   SElemType *base;
   SElemType *top;
   int stacksize;
}SqStack; //P58
```

```
typedef struct {
   QElemType *base;
   int front;
   int rear;
}SqQueue; //P70
```

```
typedef struct {
   char ch[MAXLEN+1];
   int length;
} SString; //P90串模式匹配
```

```
typedef unsigned char
SString[MAXLEN+1];
```

```
typedef struct {
   VertexType vex[MVNum];
   ArcType arcs[MVNum][MVNum];
   int vexnum, arcnum;
}AMGraph; //P154
```

```
typedef struct {
   ElemType *R;
   int length;
}SSTable; //P192查找
```

```
typedef struct {
   RcdType r[MAXSIZE+1];
   int length;
}SqList ; //P251堆排序
```

TSMC@2010, Rong Chen, All rights reserved.

顺序结构

- 顺序表—数据元素顺序存放，使用下标访问
- 静态链表

```
typedef struct {int weight;
   int parent,lchild,rchild;
}HTNode, *HuffmanTree;
```

  - P138赫夫曼树*HuffmanTree, P258链式基数排序
- 基本运算：查找、插入、删除
- 注意：
  - P225哈希表HashTable的物理结构与顺序表类似，但数据元素的访问采用哈希函数，所以归入散列结构。

TSMC@2010, Rong Chen, All rights reserved.

链式结构

- 单链表
  - P30单链表*LinkList，P60链栈，P73链队
  - P50顺序表vs.单链表
- 广义表
  - P104表结点和原子结点构成的*GList
- 树/二叉树
  - P121二叉树*BiTree，P129线索二叉树*BiThrTree，P199二叉排序树*BSTree及平衡二叉树，P212 B树*BTree
- 图
  - P156邻接表(=一维数组+多个单链表)ALGraph

```
typedef struct LNode{
   ElemType data;
   struct LNode *next;
}*LinkList; //P30
```

```
typedef struct BiTNode{
   ElemType data;
   struct BiTNode *lchild, *rchild;
}*BiTree; //P121
```

```
typedef struct ArcNode{
   int adjvex;
   struct ArcNode *nextarc;
}ArcNode; //P156
```

TSMC@2010, Rong Chen, All rights reserved.

栈 vs. 队列

	空	满
顺序栈s	**s.top==−1**	**s.top==MaxSize−1**
链栈ls	**ls−>next==NULL**	不成立
顺序队q (循环队列)	**q.rear==q.front**	**(q.rear+1)%Maxsize==q.front**
链队lq	**lq−>rear==NULL** **lq−>front==NULL**	不成立

```
typedef struct {
   SElemType *base;
   SElemType *top;
   int stacksize; //MAXSIZE
}SqStack; //P58
```

```
typedef struct {
   SElemType *base;
   int top;
}SqStack;
```

TSMC@2010, Rong Chen, All rights reserved.

散列结构

- 哈希表
  - 哈希表建立了记录关键字和存储位置之间的对应关系，查找非常快。而顺序表上没有这种关系，所以查找记录时要和关键字进行一列的比较。
- 哈希函数H(*key*)的构造方法
  - 除留余数、 折叠法、平方取中、直接定址、数字分析、随机法
- 冲突解决
  - 开放定址法（线性探测再散列P223）、链地址法、再哈希法、公共溢出区
  - 评价：平均查找长度ASL_{succ}和ASL_{unsucc}

TSMC@2010, Rong Chen, All rights reserved.

相同的逻辑结构，不同的运算

- 栈和队列
  - 不同特点：FILO vs. FIFO
  - 初始化，压栈/入队，弹栈/出队，空，满
- 栈与递归—LDR、DFS、DeleteBST、InsertAVL、QSort
- 串的模式匹配—next[]和nextval[]
- 数组—计算任意数据元素的存储位置
- 广义表—取头、取尾、表长、表深度
- 顺序表
  - 查找：顺序查找、折半查找、分块查找
  - （内部）排序
    - 直接插入排序、折半插入排序、表插入排序、希尔排序
    - 起泡排序、快速排序
    - 锦标赛排序/树形选择排序、堆排序
    - 归并排序
    - 链式基数排序

TSMC@2019, Rong Chen, All rights reserved.

树/二叉树

- 二叉树的5个性质P118
  - 完全二叉树
    - 1度结点的个数不超过1个；叶子结点个数=⌈n/2⌉；最后一个非终端结点位置=⌊n/2⌋
- 树和森林与二叉树的转换
- 遍历

二叉树	先序遍历DLR	先根遍历树/先序遍历森林	树/森林
	中序遍历LDR	后根遍历树/中序遍历森林	
	后序遍历LRD		

二叉树	先序遍历DLR	深度优先搜索DFS	图
	中序遍历LDR		
	后序遍历LRD		
	层次遍历	广度优先搜索BFS	

TSMC@2010, Rong Chen, All rights reserved.

树-是非判断

1. 给定一棵二叉树的按层次遍历序列和后序遍历序列，可以唯一地确定这颗二叉树。
2. 给定一棵二叉树的先序遍历序列和后序遍历序列，可以唯一地确定这颗二叉树。
3. 书中算法构造的Huffman树是唯一的。
4. 在Huffman树中不存在度为1的结点。
5. 在Huffman树中，权值相同的叶子结点都在同一层上。
6. 在Huffman树中，权值较大的叶子结点离根较近。
7. 在Huffman编码中，频率相同的其编码也相同。

TSMC@2010, Rong Chen, All rights reserved.

树/二叉树

- 线索二叉树P128
  - 采用左右孩子链式存储，n个结点的二叉树中有n+1个空指针
- 最优二叉树（哈夫曼树）P136
  - 树的带权路径长度WPL=$\Sigma w_i l_i$
  - 如何根据给定的n个权值w_i构造赫夫曼树，再得到赫夫曼编码？
- 折半查找过程的判定树P196
  - 查找成功或查找失败的平均查找长度ASL =$\Sigma P_i C_i$
- 动态查找—平均查找长度ASL =$\Sigma P_i C_i$
  - P198二叉排序树T—“左小右大”
    - 找不到数据元素e时将它插入树T；从T中删除一个关键字等于key的元素e
    - 插入结点而失衡，平衡处理—LL型和LR型
  - P205平衡二叉树/AVL树—“左小右大+|左右子树深度差|≤1”
  - P210平衡的多路查找树/m阶的B树—“叶子同层”
    - 插入：最低层非终端结点中添加一个关键字，关键字个数=m时该结点一分为二，关键字$K_{\lceil m/2 \rceil}$及指针插入到双亲结点；
    - 删除：找到关键字所在结点，删除之，关键字个数少于$\lceil m/2 \rceil$时合并结点。

TSMC@2010, Rong Chen, All rights reserved.

图

- AOV网—有向无环图DAG
- AOE网—带权的DAG—关键路径
- 邻接矩阵 vs. 邻接表
- 遍历：深度，广度
- 求最小生成树
  - Kruskal（克鲁斯卡尔）算法
  - Prim（普里姆）算法
- 求最短路径
  - Dijkstra（迪杰斯特拉）算法

TSMC@2010, Rong Chen, All rights reserved.

图-是非判断

1. 关键路径是由权值最大的边构成的。
2. 在AOE网中，减小任一关键活动上的权值后，整个工期也就相应减小。
3. 在AOE网中工程工期为关键活动上的权值之和。
4. 在关键路径上的活动都是关键活动，而关键活动也必在关键路径上。
5. 关键活动不按期完成就会影响整个工程的完成时间。
6. 任何一个关键活动提前完成，将使整个工程提前完成。
7. 所有关键活动若提前完成，则整个工程将提前完成。
8. 存在环路的有向图可以进行拓扑排序。
9. AOE网络中不存在环路。

TSMC@2010, Rong Chen, All rights reserved.

习题选讲1

- 证明题：用归纳法证明二叉树性质，性质3中n=B+1，例如

  证明：若一棵非空5叉树上只存在度为5、1、0的三种结点，且度为5的结点数有m个，度为1的结点数有5个，叶子结点数有n0个，试证明：n0 = 4*m + 1

  提示：m+5+n0=(5*m+5*1)+1

- 反证法证明：若借助栈由输入序列12...*n*得到的输出序列为$p_1p_2...p_n$（它是输入序列的一个排列），则在输出序列中不可能出现这样的情形：存在着$i<j<k$使$p_j<p_k<p_i$。

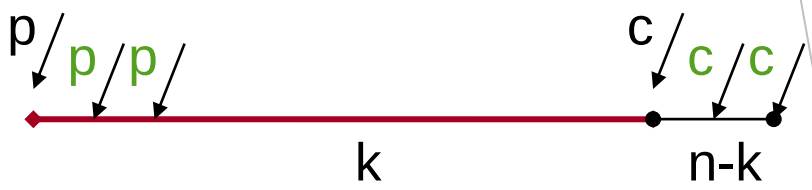
TSMC@2010, Rong Chen, All rights reserved.

习题选讲2

- 平衡二叉树（AVL树）的生成过程P206
- B-树的生成、插入和删除P214
  - 严题集9.14
- 堆排序P253（建初始堆如图8.12）
  - 题11-2-13
- 串的模式匹配P94
  - KMP算法的匹配过程
- 程序设计—单链表LinkList L
  - output_k(&L, k)输出表L的倒数第k个元素
  - 设单链表是由自然数组成的，reorder(&L)将奇数结点放在偶数结点之前。

TSMC@2010, Rong Chen, All rights reserved.

```
void output_k(LinkList &L, int k) {
LinkList pre, p;
int i=0;
1 if ( k <= 0 ) { printf(“XXX");    return; }    //判断k的合法性
2 pre=L->next;
3 c=L->next;
4 if ( !c ) { printf(“XXX");   return; }
//定位倒数第k个元素的位置。若k超出单链表的范畴，报错
5 while( c  &&  i < k ) { c=c->next;   i++; }
6 if ( i < k ) { printf("给定的k值超出单链表的长度！ ");    return; }
7 while( c ){
8      c=c->next;    p=p->next; }
9 printf(k, p->data); //输出倒数第k个元素
}//output_k
```



TSMC@2010, Rong Chen, All rights reserved.

```
Status reorder1(LinkList &L) {
    //[1]"顺藤摸瓜"，将偶数结点从原表L中脱离(即删除)，加入到偶数结点组成
    的新表Le
    p=L;
    e=Le;
    while (p) {
     p=p->next;
     if (偶数结点) {
          //将结点p从L中删除
          e->next = p;
     }
    }
    //[2]将L3链在L的表尾
    //L的表尾=Le;
    return OK;
}
```

TSMC@2010, Rong Chen, All rights reserved.

谢谢各位同学
祝大家心想事成，新年如意！