各章复习重点

单元 1 绪论

1．数据结构

数据元素间的关系称为结构，相互间存在一种或多种特定关系的数据元素的集合称为数据结构。

- 逻辑结构
- 物理结构
- 逻辑结构：元素间的逻辑关系，与计算机无关
- 物理结构：把数据存储到计算机中，并具体体现数据之间的关系。简言之，是逻辑结构在计算机中的表示（包括数据元素和关系的表示），同一种逻辑结构可以对应不同的物理结构

2．基本的数据结构

- 集合：属于同一集合
- 线形：一对一，数据元素之间存在一对一的关系，线性表除第一个元素和最后一个元素外每个元素有一个直接前驱和直接后继
- 树形：一对多
- 图：多对多

3．算法：解决特定问题的方法

- 算法的 5 个特征：有穷、确定、可行、零个或多个输入、一个或多个输出
- 时间复杂度

基本操作、频度、问题规模、数量级、时间复杂度与实现算法的软、硬件无关

n 个矩阵的乘积算法的基本操作为乘法，时间复杂度为 $O(n^3)$

要在 n 个数据元素中找最大元素，基本操作为比较，比较次数为 n-1，时间复杂度为 O(n)

单元 2 线性表

1．线性表的定义

属于同一个数据对象的数据元素的有限序列

2．顺序存储（顺序表）

逻辑结构与存储结构一致，可用数组或指针实现，能随机访问，如果线性表存储后最常用的操作是取第 i 个结点及其前驱，采用顺序表较方便，但顺序表插入删除操作平均而言移动元素次数较多，效率很低. 插入位置 i，移动元素次数为 n-i+1.删除位置是 i，移动次数 n-i

3．链式存储（链表）

以结构变量存储结点，动态生成结点，以指针链接结点，能有效利用存储空间，插入删除方便，但不能随机访问.单向链表可从某结点访问到后继结点

4．单向链表操作的关键步骤

- 建立链表的头插法：指针变量 p 开辟单元，生成结点，指针变量 q 始终指向头结点。操作为：

  p->next=q->next;

  q->next=p;

  尾插法：指针变量 q 始终指向尾结点，p 指针开辟单元，生成结点。

  q->next=p;

  q=p;

  插入：p 所指向结点的后面插入新结点 s 所指结点。

  s->next=p->next;

  p->next=s;

  删除：p，q 指向相邻结点，q 所指结点是 p 所指结点的后继，删除 q 所指结点。

  p->next=q->next;

  遍历：

  p=p->next;

5．单向循环链表

- 单向链表中，如果有 p->next==NULL，令 p->next=head；则成为单向循环链表，可从某结点访问到任一结点，但访问前驱要通过头结点.单向循环链表中，若 p 指向尾结点，则 p->next==head

6．双向链表

- 每个结点有两个指针域，一个指向直接后继，一个指向直接前驱.头结点的 prio 指向尾结点，尾结点的 next 指向头结点，从任一结点可访问前驱和后继。

7．单向链表为空的判断条件是 head==NULL，但带头结点的单向链表为空的判断条件为 head->next==NULL。

8．链式存储的线性表都不能随机访问

单元 3 栈和队列

1．栈和队列是运算受限制的线性表

2．栈：后进先出（LIFO），栈的插入删除操作在栈顶进行

例：进栈顺序为 b，c，d，e，f。出栈可能为 f，e，d，c，b；b，c，d，e，f；c，b，e，d，f……但不可能是 e，d，f，b，c。

3．顺序栈

- 相当于线性表的顺序存储结构，可用一维数组实现
- 设置栈顶指针 top，在栈顶进行操作（插入、删除等）

4．链栈

- 相当于线性表的链式存储结构，设头指针变量为 top

出栈：用 x 保存被删结点的值，在栈中删除结点。x=top->data;top=top->next;

进栈：设栈顶指针为 h，要插入 s 所指结点，操作为 s->next=h;h=s;相当于在单向链表的头插法

5．队列（FIFO）

入队：1，2，3，4，5　　出队：1，2，3，4，5

6．顺序队列

- 可以用一维数组来实现
- 设置指针 front、rear 分别指向队列的队头元素和队尾元素

7．链队列

- 是在表头删除在表尾插入的单链表
- 设置头指针 front，尾指针 rear
- 在链队中插入 s 所指结点的操作为：rear->next=s;rear=s;
- 在链队中删除结点相当于删除链表中的第一个结点（若要保存被删结点，可先保留，再删除）

单元 4 字符串

1．每个字符占一个字节，串的最基本的存储方式是顺序存储、链式存储

2．C 字符串的特点是在串尾自动加一个结束符

3．有关串的运算的函数（求串长、复制、连接、比较、查找字符、查找子串），要求能掌握函数的功能。例 StrCmp (“a”，“A”)的值为 1，上述函数是字符串比较函数。两个串相等的充要条件是串长相等，对应位置字符相等。例 StrCat (“ab”，“cd”)的功能是串连接。

4．从字符串中查找子串的方法称为模式匹配算法，例求串 q 在 p 中首先出现的位置。

单元 5 数组和广义表

1．特殊矩阵，如对称矩阵的压缩存储结构，矩阵元素与一维数组元素的对应。设数组下标从 1 开始，矩阵元素 a4，3，（1+2+3）+3=9，a4，3 对应一维数组下标为 9，a7，6 对应一维数组下标为（1+2+3+ ••• +6）+6=27.ai，j 对应下标为 i(i-1)/2+j

2．稀疏矩阵的三元组存储结构（行，列，非零元）

单元 6 树和二叉树

1．树的定义

连通不含回路的图

树的边数 m 和顶点数 n 有关系 n=m+1，顶点

数等于边数加 1

2．二叉树的性质

- 二叉树上终端结点数（叶结点数）等于双分子结点数（度数为 2 的结点数）加 1。

  例有 n 个叶结点的二叉树，每个结点度数为 2，则有 2n-1 个结点
- 二叉树第 i 层上至多有 2i-1 个结点
- 深度为 h 的二叉树至多有 2h-1 个结点
- 二叉树中编号为 i 的结点，左孩子结点编号为 2i，右孩子结点为 2i+1
- 满二叉树、完全二叉树

设有一棵完全二叉树有 18 个结点，最高层有 18-（1+2+4+8）=3 个结点

例有一棵二叉树，有 2n-2 条边，每一个非叶结点度数都为 2，则共有 2n-2+1=2n-1 个顶点，有 n 个叶结点，n-1 个非叶结点

3．二叉树的存储结构

- 顺序存储

对结点编号，以编号为下标把结点存储到一维数组中

- 链式存储结构

  Left data right
- 链式存储的二叉树的空指针域

设结点数为 n，共有 2n 个指针域，有 n-1 个指针域指向 n-1 个结点（根结点除外）所以有 2n-(n-1)个指针没有指向（空指针域），即 n+1 个空指针域

- 二叉树的遍历

访问每个结点一次且只一次

遍历树的三个子问题：根结点、左子树、右子树。规定先左后右，以根结点的访问顺序分为先、中、后序遍历.另外还有层次遍历，共四种遍历方法

- 二叉树的递归遍历算法，递归调用、输出结点信息
- 结点的权和带权路径长度

从根结点到该结点的路径长度与该结点上权的乘积

- 树的带权路径长度

树中所有叶子结点的带权路径长度之和

- 哈夫曼树（最优树）

n 个带权叶结点构成的所有二叉树中，带权路径长度 WPL 最小的二叉树

- 构造 Huffman 树的算法

设 n 个权值{w1，w2，•••，wn}

（1）在权值集合中取权值最小的作为结点，以它们的权值之和作为它们的父结点的权值

二叉树的存储结构

- 构造 Huffman 树的算法

（2）在剩余的权值集合中，加入上述父结点的权值得到新权值集合，重复步骤（1），直到权值集合中只剩下一个权值，生成一棵有 n 个结点的 Huffman 树

- 哈夫曼编码

在 Huffman 树中，让每个分支结点的左、右分支分别用 0、1 编码，从根结点到叶结点的路径上所经分支的 0、1 编码序列为该叶结点的二进制编码，所有叶结点的编码集合为哈夫曼编码

Huffman 树的特点之一：除叶结点外，每一个结点度数都为 2。设一棵哈夫曼树有 n 个非叶结点，则有 n+1 个叶结点，共有 2n+1 个结点

单元 7 图

1．图的存储结构

2．图的遍历

- 图的广度优先遍历的规则和步骤

（1）访问 vi，访问 vi 的所有未被访问过的邻接点 vi1、vi2、••• vit

- 图的广度优先遍历的规则和步骤

（2）按照 vi1、vi2、••• vit 的次序，访问每一个顶点所有未被访问过的邻接点，依次类推直到和 vi 有路径相通的顶点都访问过为止

- 图的深度优先遍历的规则和步骤

访问 vi（初始点），从 vi 的任一个未被访问过的邻接点出发继续深度优先搜索遍历，若搜索过程中某一结点的邻接点全被访问过，则退回上一个结点，继续深度搜索遍历，直到退回到初始点且没有未被访问过的邻接点

3．图的最小生成树

- 生成树

设有连通图 G，取 G 的全部顶点和部分边构成子图 G’，若 G’连通且不含有回路，则 G’是生成树

- 带权图

边上带有权的图

- 连通图一定存在生成树，且不一定唯一

- 树权

树中所有边的权值之和

- 最小生成树

连通图中具有最小权的生成树

- 带权连通图一定存在最小生成树，且不一定唯一

单元 8 查找

1．查找表、关键字：主关键字

2．线性表的查找

- 顺序查找：从表的某一端开始逐次进行比较
- 折半查找：针对有序的顺序表，设置 low、high，令 mid=（low+high）/2

3．折半查找对应的判定树

树中结点相应于查找表中的记录，结点值相应于记录在查找表中的位置

4．利用折半查找的判定树，求成功查找到某一元素、查不到某一元素的比较次数、等概率条件下成功查找的平均查找长度

5．分块查找的数据结构

- 查找表分块
- 索引表（块内最大关键字值、块起始地址）

6．二叉排序树

- 二叉排序树定义

若左子树非空，则左子树所有结点的值小于根结点的值；

若右子树非空，则右子树所有结点的值大于根结点的值；

- 二叉排序树定义

左、右子树也分别是一棵二叉排序树（每个结点的值都大于它的左子树上所有结点的值，小于它的右子树上所有结点的值）二叉排序树中任一棵子树也是二叉排序树。

- 二叉排序树的建立

实际上是从空树逐次插入的过程

- 二叉排序树的查找
- 二叉排序树的插入

7．哈希函数

记录的关键字值与该记录存储地址之间构造的对应关系。

单元 9 排序

1．插入排序

- 直接插入

第 i 趟插入是指前 i-1 个已有序，第 i 个元素逐次与前 i-1 个元素比较找到插入位置。插入后得到有 i 个元素的有序序列。

- 折半插入

采用折半查找法找到插入位置，加快了查找速度

2．交换排序

- 冒泡排序

n 个元素通常需要 n-1 趟冒泡

第 i 趟冒泡要进行 n-i 次元素比较

某趟冒泡中若没有进行元素的交换，则表

明已排好序，可设立标志位结束冒泡过程

- 快速排序

一趟划分执行步骤：设置分割元素（第一个元素），逐次轮换从后向前、从前向后扫描，必要时交换记录位置，最终使划分元素到位，完成一次分割，递归调用一趟划分函数，实现快速排序

3．选择排序

- 简单选择排序

逐次从 n 个元素、n-1 个元素•••，查找最小元素的位置，并逐一排序到位

- 堆排序

大根堆、小根堆

堆与特殊完全二叉树的对应

筛选：输出堆顶元素（存入到最后一个元素的位置），以最后一个元素替换，并自顶向下重新调整为堆

建初始堆：从最后一个非叶子结点开始直到第一个结点，从下到上逐次筛选

4．归并排序

- 归并：把两个有序序列合成一个有序序列（逐次比较）
- 一趟归并算法
- 对待排序列逐次施行（1，1）归并、（2，2）归并，……最终完成排序