**数据结构总复习**

**一、基本要求**

数据结构与算法是计算机及其计算机相关专业的专业基础课，属于主干课程，它是计算机程序设计的理论和技术基础，它所涉及的内容很丰富，无论是对后续课程的学习，还是对从事计算机软件的开发都起着重要的作用。

数据结构主要研究把具有一定逻辑关系的一批数据按某种存储方式存放在计算机的存储器中，并在这批数据上定义一系列操作。如何进行操作，这就是算法问题，算法与数据结构是相互关联的，算法总是建立在一定的数据结构基础上的，合理的数据结构可使算法简单且高效。

学习数据结构目的是理解和掌握各种数据结构的定义及基本操作的实现，理解和掌握典型算法的基本思想、算法设计方法和计算算法的时间复杂度。

掌握的知识点如下：

⑴ 线性表、顺序表和链表。

要求掌握线性表的概念，两种存储结构的实现、优缺点及两种存储结构上的基本操作。

⑵ 栈与队列。

要求掌握栈和队列的概念，顺序栈、链栈的操作，栈的应用，循环队列、循环链队列的操作。

⑶ 串的基本运算和模式匹配。

掌握串的基本运算的含义，了解模式匹配算法和时间复杂度。

⑷ 多维数组和广义表。

掌握多维数组及特殊矩阵的地址公式，广义表的运算和存储。

⑸ 树和二叉树。

掌握树、二叉树的定义、术语，二叉树的性质、存储、遍历、应用，线索二叉树的概念，树与二叉树的关系。

⑹ 图的存储及其操作。

掌握图的定义、术语，图的存储，图的遍历、图的操作（最小生成树、拓扑排序、关键路径、最短路径）概念。

⑺ 表和树的查找。

掌握表和树查找的概念、平均比较次数，二叉排序树和平衡二叉树的插入、删除，了解B-树的定义。

⑻ Hash技术。

掌握哈希表构造、解决冲突的方法及哈希表的查找。

⑼ 排序算法。

掌握直接插入排序、冒泡排序、简单选择排序、快速排序、堆排序、归并排序和希尔排序算法和时间复杂度，了解基数排序、外排序的概念和算法。

**二、基本内容**

第一章 数据结构与算法概念

         数据结构：数据结构DS=(A,R)，其中A是数据元素的非空有限集合，R是定义在A上关系的非空有限集合。结构就是元素之间的关系。

         算法：算法就是解决问题的方法和步骤。

         算法的时间复杂度：算法中语句重复执行次数（或称语句频度）或算法中基本操作次数，一般用数量级符号○来描述。

         抽象数据类型：抽象数据类型ADT=(A,R,P)，其中A是数据元素的非空有限集合，R是定义在A上关系的非空有限集合，P是(A，R)上非空的基本操作集合。

【例1-1】     求表2-1中程序段的各语句的语句频度和时间复杂度。

表2-1 语句频度和时间复杂度计算

|  |  |  |
| --- | --- | --- |
| 语句 | 语句频度 | 说明 |
| for(i=0;i<N;I++)< SPAN> | n+1 | 当i=n时，跳出for循环，故加1 |
| for(j=0;j<I;J++)< SPAN> | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230502_103182203979.gif | 即：mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230502_555150745743.gif |
| a[i][j]=x++; | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230503_130417283333.gif |  |

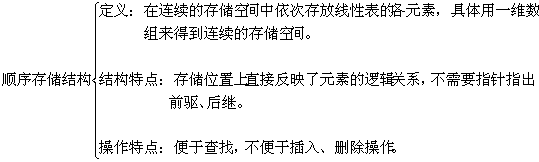
时间复杂度T(n)为所有语句频度之和，即T(n)=n+1+2mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://cc.usst.edu.cn/able.acc2.web/eWebEditor/uploadfile/20090109230502_103182203979.gif=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230504_984562625793.gif

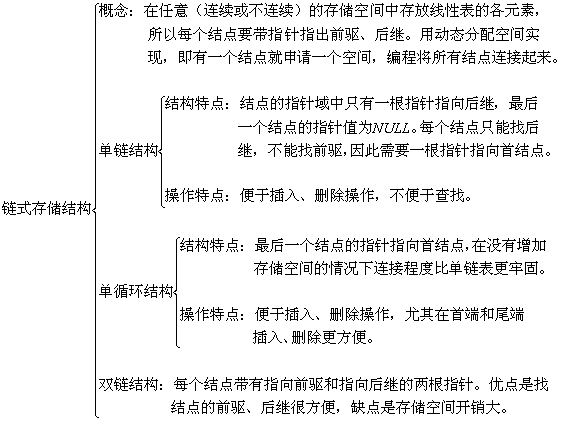
当n→∞时， mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230505_320336693778.gif 所以时间复杂度T(n)=○(n2)

第二章 线性表

         线性表的定义：线性表是n(n≥0)个元素的有限序列，当n=0，则称为空表；当n>0时，线性表通常表示为（a1 ，a2 ，．．．，an），其中a1无前驱，an无后继，其余结点有且只有一个前驱和一个后继。

         线性表的存储：线性表有顺序存储和链式存储两种存储结构。





         线性表的操作：主要操作有初始化表、判表空、求表长、查找、插入和删除元素等。线性表的存储结构不同，具体的操作实现就不同，如表2-2所示。

表2-2 线性表的两种存储结构的基本操作特点

|  |  |  |  |
| --- | --- | --- | --- |
|  | 顺序存储结构 | | 链式存储结构 |
|  | 静态数组 | 动态数组 |  |
| 类型定义 | #define ArSize 10  typedef struct {  ElemType elem[ArSize];  int length;} SqList; | typedef struct {  Elemtype \*elem;  int length, ArSize;  }sqList; | typedef struct node  {ElemType data;  struct node\*next;  }LNode; |
| 初始化表 | void initList(SqList \*L)  {L->length=0;} | void initList(sqList \*L, int n)  {L->elem=(ElemType\*)malloc(  n\*sizeof(ElemType));  L->ArSize=n;L->length=0;} | LNode \*initLK( )  {LNode \*head;  head=NULL;  return head;} |
| 查找 | 由于第i个元素ai的存储位置LOC(ai)=LOC(a1)+(i-1)×d，所以能随机查找任一个元素，时间复杂度为O(1) | | 只能进行顺序查找，查找一个元素平均比较次数为(n+1)/2 |
| 插入删除 | 插入、删除时大量结点要移动，在等概率下插入、删除一个元素平均移动结点的次数分别是n/2和(n-1)/2。 | | 插入、删除时只要修改相关的指针即可，时间复杂度都为O(1) |

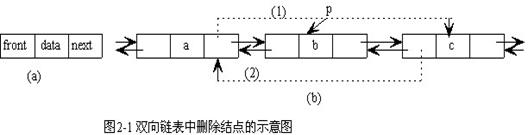
【例2-1】在非空的双链表中删除指针p所指向的结点。双链表的结点形式如图2-1(a)所示,删除p的操作步骤如图2-1(b)所示。

图2-1中（1）为p->front->next=p->next;(2)为p->next->front=p->front 。

【例2-2】在非空的双链表中指针p所指向的结点前插入一个新结点q。插入q的操作步骤如图2-2所示。

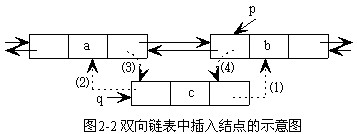


图2-2中（1）为q->next=p; (2)q->front=p->front;

(3)p->front->next=q; (4)p->front=q

注意：步骤（3）与（4）的顺序不能颠倒。

第三章 栈和队列

3.1 栈

         栈的定义:栈是一种只能在表的某一端（首端）进行操作的线性数据结构。栈的操作主要有进栈和出栈，因为只能在某一端进栈出栈，所以必定是先进后出的（FILO或LIFO）。

         栈的存储结构：有顺序栈（静态数组或动态数组）和链栈。

         栈的操作：初始化栈、进栈、出栈，通过栈顶指针（top）来操作如表2-3所示。

表2-3 顺序栈和链栈的操作

|  |  |  |  |
| --- | --- | --- | --- |
|  | 静态数组栈 | 动态数组栈 | 链栈 |
| 类型定义 | #define SSize 10  typedef struct {  ElemType elem[SSize];  int top;}SqSNode; | Typedef struct{  ElemType \*elem,\*top;  int SSize;  }DSNode; | typedef struct node  {ElemType data;  struct node\*next;  }LSNode; |
| 进栈 | SqSNode s;  s.elem[s.top++]=e; | DSNode s;  \*s.top++=e; | LSNode \*top,\*p;  p=new LSNode;p->data=e; p->next=top;top=p; |
| 出栈 | e=s.elem[--s.top]; | e=\*--s.top; | p=top;e=p->data;  top=top->next;delete(p) |
| 栈空 | s.top==0 | s.top==s.elem | top==NULL |
| 栈满 | s.top==SSize | s.top==s.elem+s.SSize | 空间分配失败p==NULL |

         栈的应用举例：表达式求值和递归的实现。

【例3-1】算术表达式a+b\*c/(d-e)-f的逆波兰式为abc\*de-/+f-;

【例3-2】已知下列算法

void p(int n)

{ if (n>1 &&n%2==1) p(n-1);

printf(“%2d”,n);

if (n>1 &&n%2==0) p(n-1);

}

用p(5)调用的结果是：4 2 1 3 5

分析：当遇到调用语句时将参数、返回地址进栈；当遇到函数结束时退栈返回。

3.2 队列

         队列的定义：队列是一种只能在表的尾端进行插入操作，在首端进行删除操作的线性数据结构。它是先进先出（FIFO）的线性表。

         队列的存储结构：有顺序队列—循环队列，设有首指针和尾指针；链队列—一般在单循环链表上实现，只设尾指针，不设首指针称循环链队列。

队列的操作：初始化队列、进队、出队，通过队列的首指针front和尾指针rear实现操作如表2-4所示。

表2-4 循环队列和循环链队列的操作

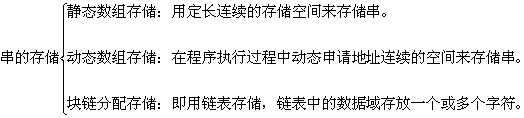
|  |  |  |
| --- | --- | --- |
|  | 循环队列 | 循环链队列 |
| 类型定义 | #define QSize 10  typedef struct {  ElemType elem[QSize];  int front,rear;}SqQNode; | typedef struct node  {ElemType data;  struct node\*next;  }LQNode; |
| 进队 | SqQNode Q;  Q.rear=(Q.rear+1)% QSize;  Q.elem[Q.rear]=e; | LQNode \*rear,\*p;  p=new LQNode; p->data=e; p->next=rear->next; rear->next=p; rear=p; |
| 出队 | Q.front=(Q.front+1)% QSize;  e=Q.elem[Q.front]; | p=rear->next; e=p->data;(无头链表)  rear->next=p->next;delete(p)； |
| 队空 | Q.front==Q.rear | rear==NULL |
| 队满 | Q.front==(Q.rear+1)%QSize | 空间分配失败p==NULL |

【例3-3】设循环队列Q，则当前循环队列中的元素个数是：

(Q.rear-Q.front+ QSize)%QSize.

第四章 串

         串、子串的定义：串是有限个字符组成的序列。子串是串中任意长度的连续字符构成的序列。

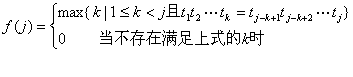
       

         串的运算：两个串的联接、比较，串的赋值，求串长，插入子串，删除子串和模式匹配。串的模式匹配算法有朴素的模式匹配算法和KMP算法等，设主串s、长度为m，子串t、长度为n，匹配算法如表2-5所示。

表2-5 模式匹配算法

|  |  |
| --- | --- |
| 朴素的模式匹配 | KMP算法 |
| 若s[i]==t[j]，则i++；j++; | 若s[i]==t[j]，则i++；j++; |
| 若s[i]!=t[j]，则i=i-j+2；j=1; | 若s[i]!=t[j]，则 j=next[j];i不动 |
| 时间复杂度T（m,n）=○(mn) | 时间复杂度T(m,n)=○(m+n) |

其中mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230510_904367637753.gif



【例4-1】已知主串s=”ABBABA”,子串t=”ABA”,求用朴素的模式匹配算法查找子串t的比较次数。

解：用朴素的模式匹配算法需要进行8次比较，子串位置是4。

【例4-2】求子串t=”ABCABCACAB”的next值。

解： j: 1 2 3 4 5 6 7 8 9 10

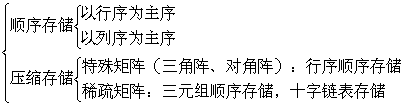
t[j]: A B C A B C A C A B

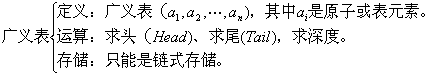
f[j]: 0 0 0 1 2 3 4 0 1 2

next[j]: 0 1 1 1 2 3 4 5 1 2

第五章 数组和广义表

         数组的定义：n维数组Ab1b2…bn ,A中的每个元素A[j1, j2,…,jn]其中1≤ji≤bi,1为每一维的下界，bi为第i维的上界。

         数组的存储

         

【例5-1】已知三维数组Amnp,将A以行序为主序存储在首地址为loc(A[1,1,1])的内存空间中，每个元素占L个单元，则数组A中任意一个元素的地址为

loc(A[i,j,k])= loc(A[1,1,1])+((i-1)\*n\*p+(j-1)\*p+k-1)\*L

若以列序为主序，则loc(A[i,j,k])= loc(A[1,1,1])+((k-1)\*m\*n+(j-1)\*m+i-1)\*L

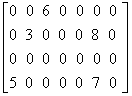
【例5-2】已知对称阵Ann, 以行序为主序存储A的下三角阵部分，内存首地址为loc(A[1,1])，每个元素占L个单元，则下三角阵中任意一个元素的地址为

loc(A[i,j])=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230512_169315138075.gif

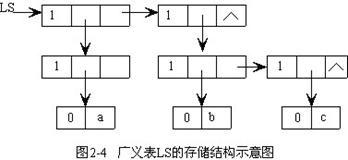
【例5-3】已知上三角矩阵Ann 以行序为主序存储在内存首地址为loc(A[1,1])，每个元素占L个单元，则上三角阵中任意一个元素的地址为

loc(A[i,j])= loc(A[1,1])+((2n-i+2)\*(i-1)/2+(j-i))\*L 当1≤i≤j时

【例5-4】 已知广义表LS=((a,(b)),(c)),LS深度为3（括号的重数），求头Head(LS)=(a,(b)),求尾Tail(LS)=((c)).

【例5-5】已知四行七列的稀疏矩阵M=，其三元组表为（(1,3,6),(2,2,3),(2,6,8),(4,1,5),(4,6,7)）。

【例5-6】已知广义表LS=((a),(b,c)),广义表的结点结构如图2-3 所示，LS的链式存储结构如图2-4 所示。

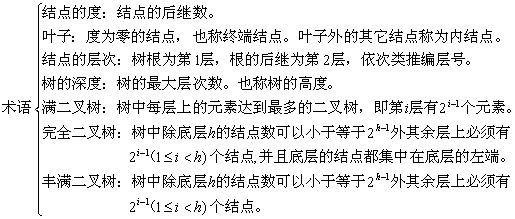


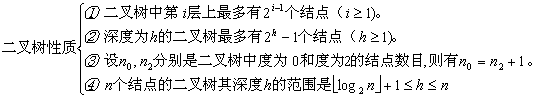
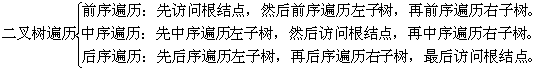
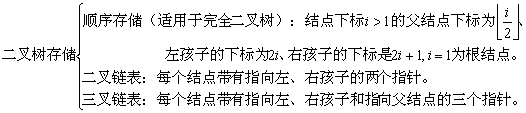


第六章 树和二叉树

6．1 二叉树

         二叉树的定义：二叉树是由一个特定的结点（无前驱）称为根和两个互不相交的左子树、右子树组成，其中左子树，右子树本身是二叉树。

         

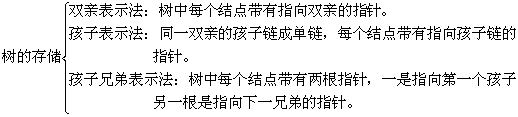
         线索二叉树：二叉树用二叉链表存储时，当二叉树有n个结点时，则有n+1个指针域为空，可以利用这些空指针域存放某种遍历次序下的前趋和后继结点的指针，这种二叉树就称为线索二叉树。在线索二叉树中用得较多的是中序线索二叉树，即树中任意结点p，若p左指针域空，则其左指针域指向p的中序前趋;若p右指针域空，则其右指针域指向p的中序后继。

         mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230515_612879218029.gif哈夫曼树（Huffman）：哈夫曼树又称最优二叉树。它是n个带权叶子结点构成的所有二叉树中带权路径长度WPL最小的二叉树。其中：

其中n表示叶子结点的数目，Wi和li分别表示叶子结点ki 的权值和根到ki 之间的路径长度。

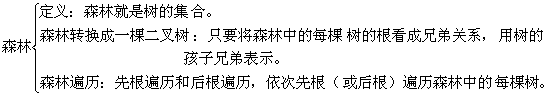
6．2 树和森林

         树的定义：树是由一个特定的结点（无前驱）称为根和m(m>0)个互不相交的树（称为子树）组成，其中当m=0时称为空树。

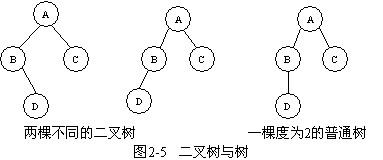
         

         mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230516_328634370493.gif

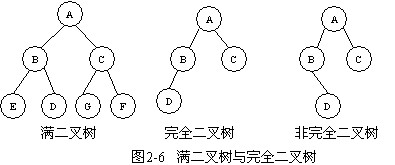
         树转换为二叉树：树用孩子兄弟表示法表示后即为二叉树了。

         

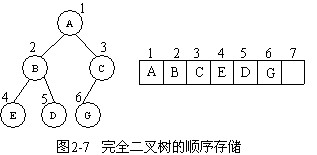
【例6-1】图2-5所示树与二叉树的区别，二叉树的子树有左右之分。



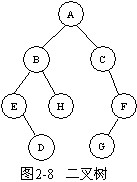
【例6-2】图2-6所示满二叉树和完全二叉树。



【例6-3】完全二叉树的顺序存储如图2-7所示。



【例6-4】n个结点的二叉树用二叉链表存储，那么有n+1个空指针，因为n个结点共有2n个指针空间，其中n-1个指向孩子结点。

【例6-5】对于图2-8所示的二叉树，它的

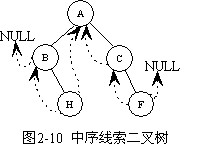
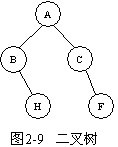
前序遍历序列为：ABEDHCFG

中序遍历序列为：EDBHACGF

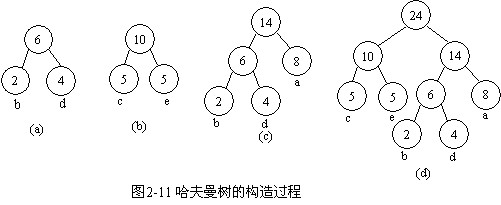
后序遍历序列为：DEHBGFCA

层次遍历序列为：ABCEHFDG

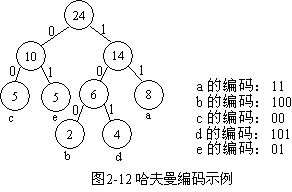
【例6-6】对于图2-9所示的二叉树，其中序线索二叉树如图2-10所示。



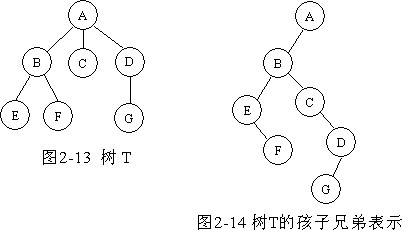
【例6-7】假设通信的电文仅有a,b,c,d,e五个字母组成，字母在电文中出现的次数分别是8，2，5，4，5。用哈夫曼算法构造的哈夫曼树过程如图2-11所示。



哈夫曼树中从根到每个叶子结点都有一条唯一的路径，对路径上的各分支约定左分支表示‘0’码，右分支表示‘1’码，则从根结点到叶子结点的路径上分支的码组成的字符串作为该叶子结点的编码，这就是哈夫曼编码。为上述五个字母设计的哈夫曼编码如图2-12所示。



【例6-8】对于图2-13的树T，它的孩子兄弟表示法如图2-14所示，即将树T转换为二叉树了。



【例6-9】图2-13中树T的前序遍历序列为：ABEFCDG, 后序遍历序列为：EFBCGDA。而图2-14中的二叉树的前序遍历序列为：ABEFCDG, 中序遍历序列为：EFBCGDA。这个例子验证了树和树转换为二叉树的前序遍历序列相同；树的后序与二叉树的中序相同。

第七章 图

7.1 图的定义：图G＝（V，E），其中V是顶点（结点）的有穷非空集合，E是V中顶点的序偶组成的有穷集，这些序偶称为边。

7.2 基本术语

         邻接点：，若存在一条边<Vi,vj>，则在无向图中称vi和vj互为邻接点，在有向图中称vj是vi 的邻接点。

         完全图：图G中任意两点都邻接，称该图为完全图。无向完全图有n（n-1）／2条边,有向完全图有n（n-1）条边。

         度、入度和出度：无向图中顶点v依附(关联)的边数称为v的度。对于有向图，顶点V的度分为入度和出度，v的入度是边的箭头指向v的入边数目；v的出度是箭头离开v的出边数目。有关度的结论有图中所有结点度数之和等于图的边数的两倍。

         连通和连通图：在无向图G中．若从顶点vi到vj有路径，则称vi和vj是连通的。若图G中任意两个顶点都是连通的，则称G连通图。

       连通分量：无向图G中的极大连通子图称为图G的连通分量。

         强连通图和强连通分量：在有向图G中，若任意两个顶点vi和vj都连通，即从vi到vj存在路径，从vj到vi也存在路径,则称该图是强连通图。有向图G中的极大强连通子图称为G的强连通分量。

         网：在一个图中，每条边可以标上具有某种含义的数值，该数值称为该边的权。边带权的图称为带权图，也称为网。

         生成树：连通图G有n个顶点，取G中n个顶点，取连接n个顶点的n-1条边所得的子图称为G的生成树。满足此定义的生成树可能有多棵，即生成树不唯一。

7.3 图的存储结构

         邻接矩阵：邻接矩阵是表示顶点之间相邻关系的矩阵。设图G=（V，E）有n个顶点，则G的邻接矩阵是一个n阶方阵：

A[i][j]=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230524_491609701918.gif

若图是带权(w)图,则：

A[i][j]=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230526_349467342713.gif

         邻接表：邻接表就是对图中的每个顶点ki建立一个单链表，把与ki相邻接的顶点放在一个链表中。

7.4 图的遍历

         深度优先搜索遍历(DFS)：首先访问指定的起始顶点v，然后选取与v邻接的未被访问的任意一个顶点w，访问之，再选取与w邻接的未被访问的任一顶点，访问之。重复进行如上的访问，当到达一个所有邻接顶点都被访问过时，则依次退回到最近被访问过的顶点，若它还有邻接顶点未被访问过，从这些未被访问过的顶点中取其中的一个顶点开始重复上述的访问过程，直到所有的顶点都被访问过为止。

         广度优先搜索遍历(BFS)：首先访问指定的起始顶点v，然后选取与v邻接的全部顶点w1,w2,...,wt,再依次访问与w1,w2,...,wt邻接的全部顶点（已被访问的顶点除外），再从这些被访问的顶点出发，逐次访问与它们邻接的全部顶点（已被访问的顶点除外）。依次类推，直到所有顶点都被访问过为止。

         深度优先生成树和广度优先生成树：在对具有n个顶点的连通图进行遍历时，要访问图中的所有顶点，在访问n个顶点过程中一定经过n-1条边，由深度优先遍历和广度优先遍历所经过的n-1条边是不同的，通常把由深度优先遍历所经过的n-1条边和n个顶点组成的图形就称为深度优先生成树。而由广度优先遍历所经过的n-1条边和n个顶点组成的图形就称为广度优先生成树。

7.5 最小生成树

         定义：生成树中边权之和定义为树权，在图的所有生成树中树权最小的那棵生成树就是最小生成树。

         最小生成树的算法：普里姆（Prim）算法和克鲁斯卡尔（Kruskal）算法，Prim算法的时间复杂度为O(n2),它适用于稠密图，而Kruskal算法的时间复杂度为O(elog2e), 它适用于稀疏图。

7.6 拓扑排序

         拓扑排序就是对一个有向无环图（AOV网）中的各顶点排成一个具有前后次序的线性序列。

         拓扑排序的方法：在图中始终找无前趋（入度为零）的顶点，找到无前趋的顶点输出并将其每个后继顶点的前趋数（入度数）减1，重复上述过程直至所有顶点都输出或无顶点可输出。只要图是无环的，一定能输出所有的顶点，否则说明有向图存在环。算法的时间复杂度为O(n+e)。

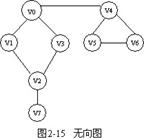
7.7 关键路径

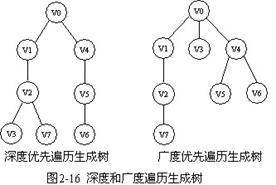
         带权有向图（AOE网）中从源点(表示工程开始)到汇点(表示工程结束)的路径长度最长的路径称为关键路径。关键路径上的所有活动都是关键活动,只要提高关键路径上的活动的速度就能缩短整个工程的工期。

7.8 最短路径

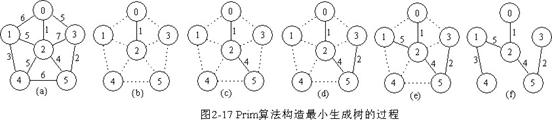
         最短路径是图中两个顶点之间的最短距离或最便宜的交通费用或途中所需的时间最少的问题。要找出最短路径方法很多。常用经典算法Dijkstra算法来求从一个源点到其他顶点的最短路径。



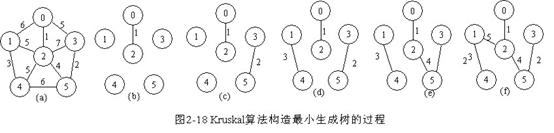
【例7-1】已知图G如图2-15所示从v0出发深度优先遍历序列是v0 v1 v2 v3 v7 v4 v5 v6或v0 v3 v2 v7 v1 v4 v5 v6 等，从v0出发广度优先遍历序列：v0 v1 v3 v4 v2 v5 v6 v7或v0 v4 v3 v1 v6 v5 v2 v7等。

【例7-2】对于图2-15所示无向图，从v0出发深度优先遍历生成树和广度优先遍历生成树如图2-16所示。

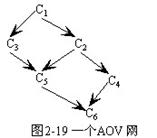
【例7-3】用Prim算法构造的最小生成树的过程如图2-17所示。



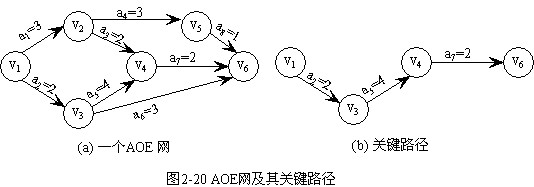
【例7-4】用Kruskal算法构造的最小生成树的过程如图2-18所示。

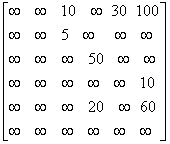
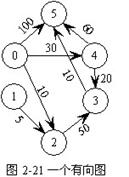


【例7-5】假设某专业中的六门课程的前修后继之间的关系如图2-19所示,课程的教学安排次序为：c1、c3、c2、c5、c4、c6。或c1、c2、c3、c5、c4、c6。这就是拓扑排序序列，方法是将图中的无前驱的结点输出，若有多个无前驱的结点，则用栈或队列暂存无前驱的结点，然后依次输出并将后件的前驱数减一。



【例7-6】已知AOE网如图2-20（a）所示，其中顶点表示事件，弧表示活动，弧上的数值表示活动需要的时间，关键路径如图2-20（b）所示。

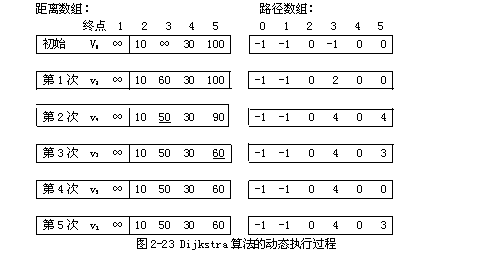




                                               2-22 有向图的邻接矩阵

【例7-7】有向图G如图2-21所示，G的邻接矩阵如图2-22所示：

从源点v0到各终点的最短路径值、路径和Dijkstra算法的动态执行过程如图2-23所示：



输出结果为：

v0到v2：v0→v2 距离10

v0到v4：v0→v4 距离30

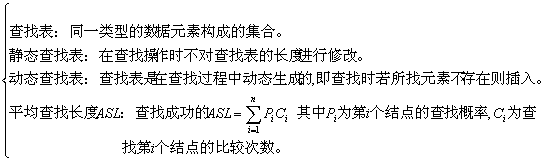
v0到v3：v0→v4→v3 距离50

v0到v5：v0→v4→v3→v5 距离60

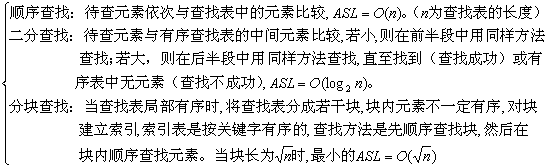
v0到v1：无路径 距离∞

第九章 查找

9.1 查找概念



9.2 静态查找表上的查找方法：

9.3 动态查找表的查找

1．二叉搜索(排序)树

         定义：非空的二叉排序树中的任意结点K，若k有左子树，则左子树中的每个结点的值都小于k的值；若k有右子树，则右子树中的每个结点的值都大于等于k的值。（注意：k是指树中的每个结点）

         查找：将待查元素与二叉排序树的根比较，若相等，则查找成功；若待查元素小于根，则在左子树中找；若待查元素大于根，则在右子树中找；重复这样的查找过程直至找到（查找成功）或子树空（查找不成功）止。

         插入：先查找插入位置，然后新结点挂在从根到插入位置这条路径的末端。

         删除：先查找到被删元素，①若被删元素是叶子，则直接删除；②若被删结点是单支（被删结点只有左子树或只有右子树）结点，则将单支的子树挂到被删结点的父结点上即从树上断开了被删结点；③若被删结点是双支（被删结点有左子树又有右子树）结点，则将左子树挂到被删结点的中序后继的左边或将右子树挂到被删结点的中序前驱的右边，这样变成删单支结点如情况②所述操作。

2．平衡二叉树（AVL树）

         定义：对于非空的平衡二叉树中的任意结点K，K的左子树和右子树的深度之差的绝对值不超过1。注意：k是指树中的每个结点。

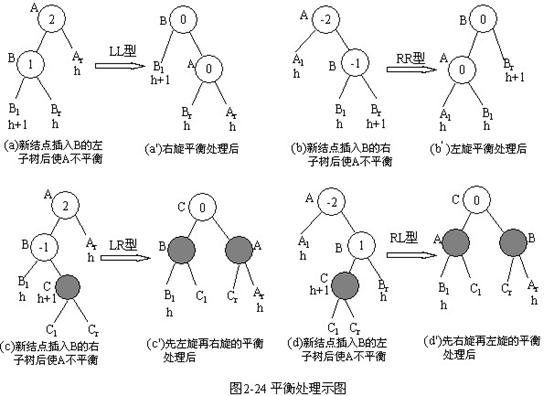
         插入：在平衡二叉排序树中插入结点，然后检查是否因插入结点而破坏平衡，若是，则找出其中最小的不平衡二叉树，按如图2-24所示情况进行调整。 

图2-24中结点的值为该结点的平衡值，阴影结点的平衡值为1或0或-1。

3． B\_树

         B\_树定义：一棵m阶B\_树，或是空树，或是满足下列特性的m叉树：

①树中每个结点最多有m棵子树；

②若根不是叶结点，则至少有两棵子树；

③除根外的每个非终端结点至少有mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230532_812845918800.gif棵子树；

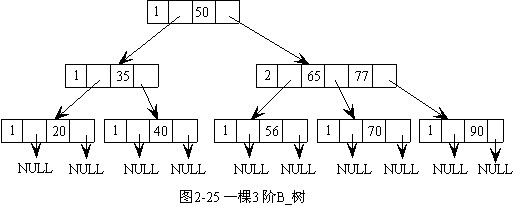
④所有非终端结点中都包含下列数据信息：

　　　　　　　　（n,A0,K1,A1,K2,A2，…，Kn,An）

其中Ki(i=1,2, …,n)为关键字且由小到大有序排列，Ai(i=0,1, …,n)为指向子树根的指针，且指针Ai-1 所指子树中的所有结点的关键字均小于Ki, An 所指结点的关键字均大于Kn, n为结点中的关键字数目(mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230533_826443523040.gif)。

⑤所有的叶子结点都出现在同一层次上，并且不带信息(看作外部结点或查找失败结点)。

一棵 3 阶B\_树如图2-25所示。



3 阶B\_树也称2-3树。

         B\_树的查找：类似二叉排序树的查找，首先在根结点包含的关键字中查找，若找到则成功返回；否则根据待查的关键字在相应的子树中继续查找，直至查找成功或查找失败。

         B\_树的插入：B\_树的结点插入只在底层进行，先查找插入位置，在底层某个位置中插入结点，若该结点中的关键字个数不超过m-1，则插入完成；否则进行结点的“分裂”处理。“分裂”就是把结点中处于中间位置上的关键字取出插到父结点中，并以该结点为分界线，把原结点分成两个结点，对父结点再进行关键字个数判断，“分裂”过程可能一直持续到树根。

         B\_树的删除：先找到被删结点，若被删结点不在底层，则用底层的后继（或前驱）替代，然后再删后继（或前驱），若被删结点关键字数目小于mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230534_619734189772.gif则进行“合并”，否则删除完成。

4．哈希表及其查找

         哈希表：设哈希函数H(Key),结点的存储方法为：H(关键字)=存储位置，即关键字作为函数的自变量，函数值解释为存储位置，按这种方法得到的存储结构图称为哈希表。

         解决冲突的方法：当哈希函数H(Key)不是一对一函数时则产生冲突，冲突即两个不同结点占同一存储空间。常见的处理冲突的方法有：

①开放地址法：

Hi=(H(Key)+di)%M i=1,2,…，M-1

其中H(Key)为哈希函数；M为哈希表的长度；di为增量序列。

当di=1，2，3，…，M-1时称为线性探测再散列。

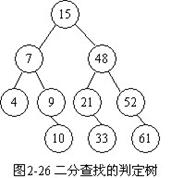
当di=12，-12，22，-22，…，±k2(k≤M/2)时称为二次探测再散列。

在发生冲突时顺序地到存储区的下一个单元进行探测。

②链地址法：每个结点增加一个链域，链域指向冲突结点。

【例9-1】已知关键字序列 4,7,9,10,15,21,33,48,52,61。写出用二分查找方法查找关键字52的比较次数并写出等概率的平均查找长度。

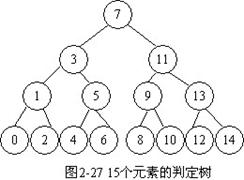
解：序列长度为10，求任意结点的查找次数可以用如图2-26所示的判定树描述。



从判定树可得查找52的比较次数为3次，ASL=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230537_323147846842.gif=2.9

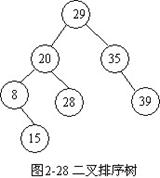
【例9-2】画出有序序列长度为15的判定树。

解：15个元素的判定树如图2-27所示：



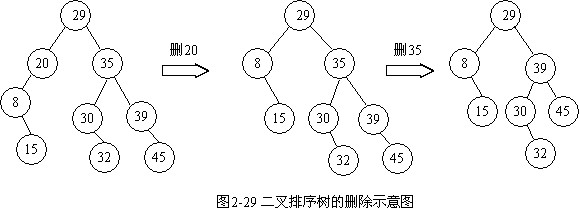
图中的数字为序列的关键字所在序列中的下标。

【例9-3】从空的二叉排序树开始依次插入29，20，35，8，15，39，28。画出该二叉排序树并计算平均查找长度。

span style='mso-ignore:vglayout;;z-index:23;margin-left:192px;margin-top:0px;width:160px;height:178px'解：二叉排序树如图2-28所示：

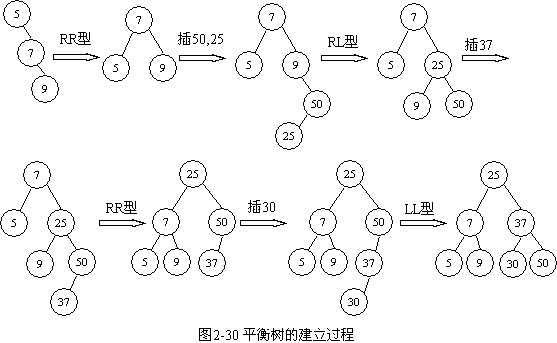
ASL=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230538_702224263652.gif=2.57

【例9-4】在如图2-29所示二叉排序树中删除结点20和35的过程。



【例9-5】从空的平衡树中依次插入5，7，9，50，25，37，30。画出建平衡二叉树的过程。

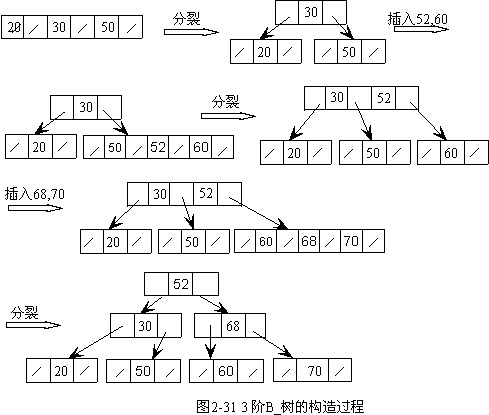
解：建平衡二叉树的过程如图2-30所示。

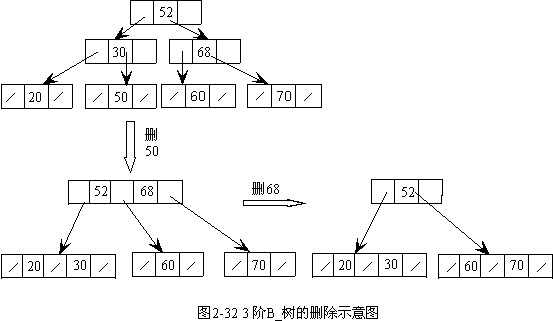


【例9-6】从空的2-3树开始依次插入20，30，50，52，60，68，70，画出建立2-3树的过程。并画出删除50和68后的2-3树状态。

解：2-3树的建立过程如图2-31所示，删除50，68后的B\_树状态如图2-32所示。

图中每个结点的结点数省略了。



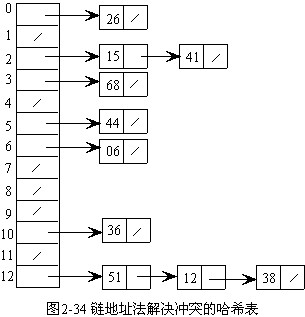


【例9-7】已知一组关键字为（26，36，41，38，44，15，68，12，06，51），哈希函数为H(key)=key%13,用线性探测再散列法解决冲突，画出哈希表并求平均查找长度。

解：哈希表如图2-33所示，平均查找长度ASL=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230545_288311280745.gif



【例9-8】对于例2.37的关键字序列用链地址法解决冲突的哈希表如图2-34所示, 平均查找长度ASL=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230546_432046562553.gif



第十章 排序

10.1 排序的基本概念

         排序：将无序序列调整为有序序列。

         稳定性：若待排序记录中有相同关键字，排序后相同关键字的相对位置发生变化，则称该排序是不稳定的；否则为稳定的。

         内部排序：指待排序记录全部存放在内存中进行排序的过程。

         外部排序：指待排序的数量很大，以至内存不能放下全部记录，在排序过程中尚需对外存进行访问的排序过程。

1．简单排序

（1）直接插入排序

         基本思想：将无序序列中从第二个开始的每个元素依次插入到有序序列中，当序列有n个元素时，则进行n-1次插入。

         算法：

void insertsort(int r[ ],int n)

/\*将数组r[1]～r[n]中的n个整数按非递减有序方式进行直接插入排序\*/

{int i,j;

for(i=2;i<=n;i++)

if ( r[i]<R[I-1])< SPAN>

{r[0]=r[i];j=i-1;

do{ r[j+1]=r[j];j--;

} while (r[0]<R[J]);< SPAN>

r[j+1]=r[0];

}

}

         稳定性：稳定的算法分析：如表2-6所示

表2-6 直接插入排序算法的时间复杂度分析

|  |  |  |  |
| --- | --- | --- | --- |
| 直接插入排序 | 最好情况 | 最差情况 | 一般情况 |
| 比较次数 | n-1 次 | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230548_146109730642.gif次 | ○(n2) |
| 移动次数 | 0次 | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230549_724652539362.gif次 | ○(n2) |
| 时间复杂度 | ○(n) | ○(n2) | ○(n2) |

（2）冒泡排序

         基本思想：小的往上冒方法：从无序序列的最后元素开始往前两两比较，第一次冒出最小元素，第二次在剩余的结点序列中冒出第二小的，这样共要冒出n-1个小的数。（也可以用大的往下沉方法）

         算法：

void bubblesort(int r[ ],int n)

/\*将数组r[0]～r[n-1]中的n个整数按非递减有序方式进行冒泡排序\*/

{int i,j,tag,temp;

for(i=0;i<N-1;I++)< SPAN>

{ tag=0;

for(j=n-1;j>=i+1;j--)

if(r[j]<R[J-1])< SPAN>

{temp=r[j];r[j]=r[j-1];r[j-1]=temp;tag=1;}

if(tag==0) break;

}

}

         稳定性：稳定的

         算法分析：如表2-7所示

表2-7 冒泡排序算法的时间复杂度分析

|  |  |  |  |
| --- | --- | --- | --- |
| 冒泡排序 | 最好情况 | 最差情况 | 一般情况 |
| 比较次数 | n-1 次 | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230549_954208615452.gif次 | ○(n2) |
| 移动次数 | 0次 | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230551_625508482814.gif次 | ○(n2) |
| 时间复杂度 | ○(n) | ○(n2) | ○(n2) |

（3）简单选择排序

         基本思想：第一次从无序序列中选出第一小的元素，然后从剩余的结点序列中选出第二小的元素，这样共选出n-1小的数。

         算法：

void selectsort(int r[ ],int n)

/\*将数组r[0]～r[n-1]中的n个整数按非递减有序方式进行选择排序\*/

{int i,j,k,temp;

for(i=0;i<N-1;I++)< SPAN>

{k=i;

for(j=i+1;j<N;J++)< SPAN>

if(r[j]< SPAN>

if(i!=k){ temp=r[i];r[i]=r[k];r[k]=temp;}

}

}

         稳定性：不稳定的

         算法分析：如表2-8所示

表2-8 简单选择排序算法的时间复杂度分析

|  |  |  |  |
| --- | --- | --- | --- |
| 简单选择排序 | 最好情况 | 最差情况 | 一般情况 |
| 比较次数 | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://cc.usst.edu.cn/able.acc2.web/eWebEditor/uploadfile/20090109230549_954208615452.gif次 | mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://cc.usst.edu.cn/able.acc2.web/eWebEditor/uploadfile/20090109230549_954208615452.gif次 | ○(n2) |
| 移动次数 | 0次 | 3(n-1)次 | ○(n) |
| 时间复杂度 | ○(n2) | ○(n2) | ○(n2) |

3．先进的排序

（1）希尔排序

希尔排序是直接插入排序方法的改进，基本思想是：先将整个待排序列分割成若干子序列（或称组），然后对各个子序列分别进行直接插入排序。待整个序列中的元素基本有序时，再对整个序列进行一次直接插入排序。具体做法是：先取一个小于n的整数d1作为第一个增量，把序列分成d1个子序列，将所有间隔距离为d1的元素放在同一子序列中，在各个子序列中进行直接插入排序，然后取第二个增量d2<D1,重复上述分组和排序，依次类推，直至所取的增量di=1为止，即所有元素在同一组中进行直接插入排序。

希尔排序中元素是跳跃式移动的，所以是不稳定的。尽管希尔排序中调用了多次直接插入排序，但初始子序列较短，花得时间不会太多，后面序列基本有序，花得时间接近直接插入排序的最好时间，所以希尔排序的时间复杂度接近○(nlog2n)。

（2）快速排序

         算法思想：选取序列的第一个结点，以它的关键字和序列中所有其它结点的关键字比较，将所有关键字较它小的结点放在它之前，将所有关键字较它大的结点放在它之后，则经过这样的一趟排序后，可按该结点所在位置为界，将序列分成两部分，然后再分别对这两部分重复上述过程，直至每一部分只剩一个结点为止。

         排序过程：

25 84 21 47 15 27 68 35 20

20 15 21 25 47 27 68 35 84

15 20 21 25 35 27 47 68 84

15 20 21 25 27 35 47 68 84

         算法：

void quicksort(node r[ ],int L,int R)

{int i,j;

       if(L<R)< SPAN>

        { i=L;j=R;r[0]=r[i];

          do{while (r[j].key>=r[0].key && i<>

             if (i

             while (r[i].key<=r[0].key && i<>

             if (i

            }while (i!=j);

         r[i]=r[0];

         quicksort(r, L, i-1);

         quicksort(r, i+1, R);

        }

  }

         稳定性：不稳定的

         算法分析：一般情况下的时间复杂度为O(nlog2n)，最坏情况下的时间复杂度O(n2)。

（3）堆排序

         堆的定义：n个元素的关键字序列{ K1,K2,…，Kn},当且仅当所有关键字都满足下列关系式时称为堆：mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230552_433280820545.gif (称为小根堆) 或 mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230553_408622642383.gif(称为大根堆) 其中 1≤i≤n/2 。

         堆排序的算法思想：首先将序列按堆的定义建成堆（如大根堆），从而堆顶的关键字最大，将堆顶与堆末交换，这样得到了最大数，并把末元素从堆中去掉，然后再将堆顶调整为堆，从而堆顶为次大数，得到了次大数。如此反复进行，直到堆中只剩一个元素止，此时数组的元素为递增（或非递减）有序序列。

         算法：

void heapsort(int r[ ], int n)

/\*将数组r[0]～r[n-1]中的n个整数按非递减有序方式进行堆排序\*/

{int i,t;

for(i=n/1-1;i>0;i--)

heapadjust(r,i,n-1);

for(i=n-1;i>0;i--)

{t=r[0];r[0]=r[i];r[i]=t; heapadjust(r , 0, i-1);}

}

void heapadjust(int r[ ],int s,int m)

/\*使r[s] ～r[m]为大根堆\*/

{int i,j,t;

t=r[s];

for(j=2\*s+1;j<=m;j=2\*j+1)

{if (j<M&&R[J]<R[J+1])J++;< SPAN>

if(t>=r[j])break;

r[s]=r[j];s=j;

}

r[s]=t;

}

         稳定性：不稳定的

         算法分析：堆排序的最好、最差情况下的时间复杂度都为O(nlog2n)。

（4）归并排序

         算法思想：将两个或多个有序序列合并成一个新的有序序列。

         稳定性：稳定的

         算法分析：归并排序的最好、最差情况下的时间复杂度都为O(nlog2n)。

（5）基数排序

         算法思想：基数排序是按组成关键字各个数位的值进行排序的，具体做法是：根据基数r（若关键字是十进制，则r=10）设置r个口袋（0～r-1），对关键字的第T位的值分配到相应的口袋中，当所有关键字分配完，则从0号～r-1号的次序进行收集，若最大关键字有d位，则上述过程重复d次。T可以从最高有效位开始，也可以从最低有效位开始。

         排序过程：如表2-9、表2-10所示

关键字序列：52，18，17，5，8，7，25

表2-9 基数排序的个位分配

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 口袋 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 个位分配 |  |  | 52 |  |  | 5  25 |  | 17  7 | 18  8 |  |

个位收集：52，5，25，17，7，18，8

表2-10 基数排序的十位分配

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 口袋 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 十位分配 | 5  7  8 | 17  18 | 25 | 52 |  |  |  |  |  |  |

十位收集：5，7，8，17，18，25，52

         稳定性：稳定的

         算法分析：基数排序的平均时间复杂度为O(d×(n+r))。

【例10-1】 给出一个例子说明简单选择排序是不稳定的。

解：关键字序列如下：

5，5，4，7，2

2，5，4，7，5

2，4，5，7，5

2，4，5，5，7

带下划线的是后一个相同关键字。

【例10-2】判断序列(100,70,33,65,24,56,48,92,86,33)是否为大根堆，如果不是，则把它调整为堆。

解：(100,70,33,65,24,56,48,92,86,33)不是堆，调整为大根堆：（100，92，56，86，33，33，48，65，70，24）

4．外排序

外排序就是对大型文件的排序，待排序的记录存放在外存上，在排序过程中，内存只存储文件的一部分记录，整个排序过程需要进行多次的内外存间的数据交换。

常用的外排序方法是归并排序，先将文件中的记录分段分别读入内存，用内部排序方法分别进行排序形成一个一个归并段，然后用某种归并方法进行一趟一趟地归并，使文件的有序段逐渐加长，直至整个文件归并为一个有序段为止。

**三、重点习题解析**

本章重点内容是栈，队列，二叉树性质及遍历，图的存储及遍历，二分查找，二叉排序树，哈希表和内部排序。本节给出相关习题的讨论。

本章难点内容是：时间复杂度计算，串模式匹配中的next函数，平衡二叉树，B\_树，算法设计方法等

(一)判断题

从下列各题叙述中分别选出5条正确的叙述。

1． ①顺序存储方式只能用于存储线性结构。

②顺序存储方式的优点是存储密度大，且插入、删除运算效率高。

③链表的每个结点中都恰好包含一个指针。

④散列法存储的基本思想是由关键码的值决定数据的存储地址。

⑤散列表的结点中只包含数据元素自身的信息，不包含任何指针。

⑥负载因子（装填因子）是散列法的一个重要参数，它反映散列表的装满程度。

⑦栈和队列的存储方式既可是顺序方式，也可是链接方式。

⑧用二叉链表法存储包含n个结点的二叉树，结点的2n个指针区域中有n+1个为空指针。

⑨用相邻矩阵法存储一个图时，在不考虑压缩存储的情况下，所占用的存储空间大小只与图中结点个数有关，而与图的边数无关。

⑩邻接表法只能用于有向图的存储，而相邻矩阵法对于有向图的存储都适用。

2． ①二叉树中每个结点有两个子结点，而对一般的树则无此限制，因此二叉树是树的特殊情形。

②当k≥1时，高度为k的二叉树至多有2k-1个结点。

③用树的前序遍历和中序遍历可以导出树的后序遍历。

④线索二叉树的优点是便于在中序下查找前趋结点和后继结点。

⑤将一棵树转换成二叉树后，根结点没有左子树。

⑥一棵含有n个结点的完全二叉树，它的高度是mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230553_922933809911.gif。

⑦在二叉树中插入结点，该二叉树便不再是二叉树。

⑧采用二叉链表作树的存贮结构，树的前序遍历和其相应的二叉树的前序遍历的结果是一样的

⑨哈夫曼树是带权路径长度最短的树，路径上权值较大的结点离根较近。

⑩用一维数组存贮二叉树时，总是以前序遍历顺序存贮结点。

3． ①一棵二叉树的层次遍历方法只有前序法和后序法两种。

②在哈夫曼树中，叶结点的个数比内部结点个数多1。

③完全二叉树一定是平衡二叉树。

④在二叉树的前序序列中，若结点u在结点v之前，则u一定是v的祖先。

⑤在查找树中插入一个新结点，总是插入到叶结点下面。

⑥树的后序序列和其对应的二叉树的后序序列的结果是一样的。

⑦对B\_树删除某一关键字值时，可能会引起结点的分裂。

⑧在含有n个结点的树中，边数只能是n-1条。

⑨最佳查找树就是检索效率最高的查找树。

⑩中序遍历二叉链表存储的二叉树时，一般要用堆栈；中序遍历检索二叉树时，也必须使用堆栈。

4． ①m阶B\_树每一个结点的后件个数都小于等于m。

②m阶B\_树每一个结点的后件个数都大于等于mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230554_932437439614.gif。

③m阶B\_树具有k个后件的非叶子结点含有k—1个键值。

④m阶B\_树的任何一个结点的左右子树的高度都相等。

⑤中序遍历一棵查找树的结点就可得到排好序的结点序列。

⑥用指针的方式存储一棵有n个结点的二叉树，最少要n+1个指针。

⑦任一查找树的平均查找时间都小于用顺序查找同样结点的线性表的平均查找时间。

⑧平衡树一定是丰满树。

⑨已知树的前序遍历并不能唯一地确定这棵树，因为不知道树的根结点是哪一个。

⑩不使用递归，也可以实现二叉树的前序、中序及后序遍历。

         判断题答案：

1．④⑥⑦⑧⑨　　　2．③④⑥⑧⑨ 3．②③⑧⑨⑩ 4．①③④⑤⑩

（二）填空题

1．算法好坏主要从 和 方面来衡量。

2．算术表达式a+b/(c+d)2f的逆波兰式是\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

3．在一个顺序存储的循环队列Q[0…M-1]，头尾指针分别是front和rear，判断队空的条件为\_\_\_\_\_\_\_\_，判断队满的条件为\_\_\_\_\_\_\_\_\_。

4．设二维数组a[10][10] 是对称阵，现将a中的上三角（含对角线）元素以行为主序存储在首地址为2000的存储区域中，每个元素占3个单元，则元素a[6][7]的地址为\_\_\_\_\_\_\_\_。

5．广义表( (a,b),(c) )的表头是 ，表尾是 。

6．假定一棵树的广义表表示为A（B（C，D（E，F，G），H（I，J））），则树中所含的结点数为\_\_\_\_\_\_个，树的深度为\_\_\_\_\_，树的度为\_\_\_\_\_\_。

7．在一棵三叉树中，度为3的结点数为2个，度为2的结点数有1个，度为1的结点数为2个，那么度为0的结点数有\_\_\_\_\_\_个。

8．一棵二叉树的结点数为18，则它的最小深度为\_\_\_\_\_\_\_，最大深度为\_\_\_\_\_\_\_。

9．对于一棵具有n个结点的二叉树，对应二叉链表中指针总数为\_\_\_\_\_\_个，其中\_\_\_\_\_\_个用于指向孩子结点，\_\_\_\_\_\_个指针空闲着。

10．某二叉树的前序遍历结点访问顺序是abdgcefh，中序遍历的结点访问顺序是dgbaechf，则其后序遍历的结点访问顺序是 。

11．有一棵50个结点的完全二叉树，其叶结点有 个。

12．设有一稀疏图G，则G采用\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_存储较省空间。

13．如果无向图G有n个顶点，那么G的一棵生成树有且仅有 条边。

14．如果无向图G有n个顶点、e条边且用邻接矩阵进行存储，那么深度优先遍历图G的时间复杂度为 。

15．假定对线性表（38，25，74，52，48）进行散列存储，采用H（K）=K%7作为散列函数，若分别采用线性探测法和链接法处理冲突，则对各自散列表进行查找的平均查找长度分别为\_\_\_\_和\_\_\_\_\_\_。

16．在待排序的元素序列基本有序的前提下，效率最高的排序方法是\_\_\_\_\_\_\_\_\_\_\_。

17．对于一个具有n个结点的序列，如果采用插入排序,所须的最大比较次数是\_\_\_\_\_\_?

所须的最大移动次数是\_\_\_\_\_\_?

18．设有1000个无序的元素，希望用最快的速度挑选出其中前10个最大的元素，最好选用\_\_\_\_\_\_\_\_排序法。

19．对于一个具有n个元素序列如果采用快速排序,那么所须的最少比较次数是\_\_\_\_\_\_\_;所须的最大比较次数是\_\_\_\_\_\_\_且此序列为 序列。

20．将两个各有n个元素的有序表归并成一个有序表，其最少的比较次数是 ，最多的比较次数是 。

         **填空题答案：**

1. 时间复杂度、空间复杂度 2.abcd+/f2+

3. 队空条件front==rear;队满条件front==（rear+1）%M

4. 上三角的地址公式loc(a[i][j])=loc(a[0][0])+(((2n-i+1)\*i/2+(j-i))×L 注意0≤i,j≤9,所以a[6][7]的地址为2138

5. 表头是(a,b)，表尾是 ((c)) 6. 10,4,3

7. m叉树中的叶子数=1+mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230557_891838396288.gif所以有6个叶子

8. 最小深度mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230557_766109806104.gif=5 最大深度是18 9. 2n, n-1, n+1

10. 画出二叉树后得gdbaehfca

11. 完全二叉树的叶结点数=mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230558_295170614149.gif n为结点数，所以有25个叶结点

12. 邻接表 13. n-1条边

14. ○（n2） 15. 2 , 1.2

16.插入排序和冒泡排序 17.(n+2)(n-1)/2 , (n+4)(n-1)/2

18.堆排序 19.nlog2n ,n(n-1)/2, 有序

20．当一个有序表的元素都比另一有序表的元素都小（或都大）时比较次数最少为n。最多比较次数为2n-1。

（三）简答题

1．简述顺序存储结构和链式存储结构的特点

答：顺序存储结构的优点无需为表示元素间的逻辑关系而增加额外的指针空间；可以随机存取表中的任一元素。缺点是必须事先进行空间分配，表的容量难以扩充；插入和删除操作时需移动大量结点，效率较低。

链式存储结构的优点是结点的存储采用动态存储，表的容量很容易扩充；插入和删除操作方便，不必移动结点，只要修改结点中的指针即可。缺点是每个结点中需要有指针空间，比顺序存储结构的存储密度小；只能进行顺序查找结点。

2．链表中为何要引入头结点？

答：链表进行插入和删除操作时要判断是否在链表的首端操作，若在第一结点前插入新结点和删除第一个结点则会引起首指针head值的改变；否则head的值不会改变。在链表前加一个头结点（只用指针域指向链表的首结点）就避免了两种情况的判断，使程序设计简单了，程序的结构更清楚。

2．  简述由二叉树的前序、中序和后序遍历序列确定二叉树

答：在三种遍历序列中，前序序列和中序序列、中序序列和后序序列能唯一确定一棵二叉树，因为前序序列或后序序列能确定二叉树的根结点而中序序列能确定根的左、右子树。前序序列和后序序列不能唯一确定一棵二叉树，但注意树的先根序列和后根序列能唯一的确定该树，因为树的后根序列就是二叉树的中序序列。

4．快速排序最坏情况的改进

答：当待排序的序列为有序序列时快速排序的效率很低，蜕变为冒泡排序了，为了避免这种情况，选序列的首元素为枢轴元素（或称基准元素）改为选序列的首元素、中间元素和末元素三个元素中中间大的元素为基准元素（简单的就用中间元素为基准），这可大大改善快速排序的性能。例如：

8，0，4，9，6，3，5，2，7，1

以中间大元素6为基准，基准元素与最后元素交换后为：

8，0，4，9，1，3，5，2，7，6

↑ ↑

ｉ　 　　　　　　　　　ｊ

将i,j指的内容比较，若i的内容比基准小，i推进，否则i停下，开始进行j的比较；若j的内容比基准大，j推进,，否则j停下，将i的内容与j的内容交换，重复上述过程，直至j<I< SPAN>止，将基准与i的内容交换，一次分段完成。,如下所示：

8，0，4，9，1，3，5，2，7，6

2，0，4，9，1，3，5，8，7，6

2，0，4，5，1，3，9，8，7，6

2，0，4，5，1，3，6，8，7，9

5．简述动态规划法的基本思想

答：为了节约重复求相同子问题的时间，引入一个表（数组），不管它们是否对最终解有用，把新的子问题的解答存于该表中，待以后遇到同样子问题时，就不再重复求该子问题，而直接从表中取出该子问题的解答，这就是动态规划法所采用的基本思想。

（四）选择题

1．循环队列用数组A[0…m-1]存放其元素值，已知其头尾指针分别是front和rear，则当前队列中的元素个数是 　　　 。

A．(rear-front+m)% m B．read-front+1

C．read-front-1 D．read-front

         **参考答案**A

2．递归算法的执行过程一般来说，可分成 (1) 和 (2) 两个阶段。

(1)A．试探 B．递推 C．枚举 D．分析

(2)A．回溯 B．回归 C．返回 D．合成

         **参考答案**(1) B (2) B

3．设哈希表长m=11，哈希函数H（key）=key%11。表中已有4个结点：addr（15）=4， addr（38）=5，addr（61）=6，addr（84）=7，其余地址为空，如果二次探测再散列处理冲突，关键字为49的结点地址是 。

A．8 B．3 C．5 D．9

         **参考答案**D

4．m阶B-树中所有非终端（除根之外）节点中的关键字个数必须大于或等于 。

A．mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://202.120.223.158/download/20090109230558_676372679147.gif－1 B． mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://cc.usst.edu.cn/able.acc2.web/eWebEditor/uploadfile/20090109230558_676372679147.gif+1 C．mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://cc.usst.edu.cn/able.acc2.web/eWebEditor/uploadfile/20090109230558_676372679147.gif－1 D．m

         **参考答案**C

5．一组记录的关键码为（46，79，56，38，40，84），则采用快速排序的方法，以第一个记录为基准得到的一次划分结果为 。

A．38，40，46，56，79，84 B．40，38，46，79，56，84

C．40，38，46，56，79，84 D．40，38，46，84，56，79

         **参考答案** C

6．若一个问题的求解既可以用递归算法，也可以用递推算法，则往往用 (1) 算法，因为 (2) 。

（1）A．先递归后递推 B．先递推后递归　　Ｃ．递归　　Ｄ．递推

　 （2）A．递推的效率比递归高　　　　　　　Ｂ．递归宜于问题分解

　 C．递归的效率比递推高 D．递推宜于问题分解

         **参考答案** （1）D （2）A

7．将一棵有100节点的完全二叉树从上到下、从左到右依次对结点进行编号，根结点的编号为1，则编号为49的结点的左孩子编号为 。

A． 99 B．98 C．50 D．48

         **参考答案** B

8．二叉树在线索化后，仍不能有效求解的问题是 。

A．前序线索二叉树中求前序后继 B．中序线索二叉树中求中序后继

C．中序线索二叉树中求中序前趋 D．后序线索二叉树中求后序后继

         **参考答案** D

9．判断线索二叉树中某结点P有左孩子的条件是 （1） 。若由森林转化得到的二叉树是非空的二叉树，则二叉树形状是 （2） 。

（1）A．P！=null B．P->lchild！=null C．P->ltag=0 D．P->ltag=1

（2）A．根结点无右子树的二叉树 B．根结点无左子树的二叉树

C．根结点可能有左子树和右子树 D．各结点只有一个孩子的二叉树

         **参考答案** （1）C （2）C

10．在一个单链表head中，若要在指针p所指结点后插入一个q指针所指结点，则执行\_\_\_\_\_。

A． p->next=q->next; q->next=p;

B． q->next=p->next; p=q;

C． p->next=q->next; p->next=q;

D． q->next=p->next; p->next=q;

         **参考答案**D

11．设二维数组a[0…m-1][0…n-1]按列优先顺序存储在首地址为loc(a[0][0])的存储区域中，每个元素占d个单元，则a[i][j]的地址为\_\_\_\_\_\_\_\_。

A． loc(a[0][0]) +(j×n+i) ×d B． loc(a[0][0]) +(j×m+i) ×d

C．loc(a[0][0]) +((j-1)×n+i-1) ×d D． loc(a[0][0]) +((j-1)×m+i-1) ×d

         **参考答案**B

12．如果一个栈的进栈序列是1,2,3,4且规定每个元素的进栈和退栈各一次,那么不可能得到的退栈序列\_\_\_\_\_。

A 4,3,2,1 B 4,2,1,3 C 1,3,2,4 D 3,4,2,1

         **参考答案** B

13．对n个元素进行快速排序时，最坏情况下的时间复杂度为　　 。

A．O（log2n） B．O（n） C．O（nlog2n） D．O（n2）

         **参考答案**D

14．任何一个基于“比较”的内部排序的算法，若对6个元素进行排序，则在最坏情况下所需的比较次数至少为 　　。

A．10 B．11 C ．21 D．36

         **参考答案**A

**四、模拟试题**

1．二叉树的前序、中序和后序遍历法最适合采用 　（１）　 来实现。

查找树中，由根结点到所有其它结点的路径长度的总和称为　（２） ，而使上述路径长度总和达到最小的树称为 （３） 。它一定是 （４） 。

在关于树的几个叙述中，只有 （５） 是正确的。

（１）A．递归程序 B．迭代程序 C．队列操作 D．栈操作

（２）A．路径和 B．内部路径长度 C．总深度 D．深度和

（３）A．B-树 B．B+树 C．丰满树 D．穿线树

（４）A．B-树 B．平衡树 C．非平衡树 D．穿线树

（５）A．用指针方式存储有n个结点的二叉树，至少要有n+1个指针

B．m阶B-树中，每个非叶子结点的后件个数≥mhtml:file://C:\Users\Administrator\Documents\常用文档（1）\数据结构课件\数据结构.mht!http://cc.usst.edu.cn/able.acc2.web/eWebEditor/uploadfile/20090109230558_676372679147.gif

C．m阶B-树中，具有k个后件的结点，必含有k-1个键值

D．平衡树一定是丰满树

         **参考答案**（１）A　　（２）B　（３）C　（４）B　　（５）C

２．一棵查找二叉树，其结点A、B、C、D、E、F依次存放在一个起始地址为n（假定地址以字节为单位顺序编号）的连续区域中，每个结点占4个字节：前二个字节存放结点值，后二个字节依次放左指针、右指针。若该查找二叉树的根结点为E，则它的一种可能的前序遍历为 （１） ，相应的层次遍历为 （２） 。在以上两种遍历情况下，结点C的左指针Lc的存放地址为　（３） ，Lc的内容为 （４） 。结点A的右指针Ra的内容为 （５） 。

（１）A．EAFCBD B．EFACDB C．EABCFD D．EACBDF

（２）A．EAFCBD B．EFACDB C．EABCFD D．EACBDF

（３）A．n+9 B．n+10 C．n+12 D．n+13

（４）A．n+4 B．n+8 C．n+12 D．n +16

（５）A．n+4 B．n+8 C．n+12 D．n +16

         **参考答案**（１）D　（２）A　（３）B　（４）A　（５）B

3．对于给定的一组关键字（12，2，16，30，8，28，4，10，20，6，18），按照下列算法进行递增排序，写出每种算法第一趟排序后得到的结果：希尔排序（增量为5）得到 （１） ，快速排序（选第一个记录为基准元素）得到 （２） ，基数（基数为10）排序得到　（３）　，二路归并排序得到 （４） ，堆排序得到 （５） 。

（１）A．2,4,6,8,10,12,16,18,20,28,30 B．6,2,10,4,8,12,28,30,20,16,18

　　 C．12,2,10,20,6,18,4,16,30,8,28 D．30,10,20,12,2,4,16,6,8,28,18

（２）A．10,6,18,8,4,2,12,20,16,30,28 B．6,2,10,4,8,12,28,30,20,16,18

C．2,4,6,8,10,12,16,18,20,28,30 　 D．6,10,8,28,20,18,2,4,12,30,16

（３）A．10,6,18,8,4,2,12,20,16,30,28 B．1,12,10,20,6,18,4,16,30,8,28

　　C．2,4,6,8,10,12,16,18,20,28,30 D．30,10,20,12,2,4,16,6,8,28,18

（４）A．2,12,16,8,28,30,4,6,10,18,20 　B．2,12,16,30,8,28,4,10,6,20,18

　　 C．12,2,16,8,28,30,4,6,10,28,18 　 D．12,2,10,20,6,18,4,16,30,8,28

（５）A．30,28,20,12,18,16,4,10,2,6,8 　 B．20,30,28,12,18,4,16,10,2,8,6

　　C．2,6,4,10,8,28,16,30,20,12,18 　 D．2,4,10,6,12,28,16,20,8,30,18

         **参考答案** （１）C 　（２）B 　（３）D 　（４）B　（５）C

4．在所有排序方法中，关键字比较的次数与记录的初始排列次序无关的是 （１） 。

从未排序序列中依次取出元素与已排序序列（初始时为空）中的元素进行比较，将其放入已排序序列的正确位置上的方法，称为 （２） 。设有1000个无序的元素，希望用最快的速度挑选出其中前10个最大的元素，最好选用 （３） 排序法。

（１）A．希尔排序 B．起泡排序 C．插入排序 D．选择排序

（２）A．希尔排序 B．起泡排序 C．插入排序 D．选择排序

（３）A．起泡排序 B．快速排序 C．堆排序 D．基数排序

         **参考答案** （１）D　（２）C 　（３）C

５．用某种排序方法对线性表（25,84,21,47,15,27,68,35,20）进行排序时,元素序列的变化情况如下：

①25,84,21,47,15,27,68,35,20 ②20,15,21,25,47,27,68,35,84

③15,20,21,25,35,27,47,68,84 ④15,20,21,25,27,35,47,68,84

则所采用的排序方法是 （1） 。下列（2）中不稳定的排序是 （2） 。

外排序是指 （3） 。

（1）A．选择排序 B．希尔排序 C．归并排序 D．快速排序

（2）A．直接插入排序 B．冒泡排序 C．Shell排序 D．归并排序

（3）A．用机器指令直接对硬盘中需排序数据排序

B． 把需排序数据，用其它大容量机器排序

C． 把外存中需排序数据一次性调入内存，排好序后再存储外存

D．对外存中大于内存允许空间的待排序的数据，通过多次内外间的交换实现排序。

         **参考答案**(1)D (2) C （3）D

６．在内部排序中，通常要对被排序数据进行多次扫描。各种排序方法有不同的排序实施过程和时间复杂性。对给定的整数数列（541,132,984,746,518,181,946,314,205,827）进行从小到大的排序时,采用冒泡排序和简单选择排序时,若先选出大元素,则第一次扫描结果分别是 （１） 采用快速排序（以中间元素518为基准）的第一次扫描结果是 （２） 。

设被排序的序列有n个元素，冒泡排序和简单选择排序的时间复杂度是　（３）　；快速排序的时间复杂度是　（４）　。

（１）

A．(181,132,314,205,541,518,946,827,746,984）和（541,132,827,746,518,181,946,314,205,984）

B．（132,541,746,518,181,946,314,205,827,984）和（541,132,827,746,518,181,946,314,205,984）

C．（205,132,314,181,518,746,946,984,541,827）和（132,541,746,518,181,946,314,205,827,984）

D．(541,132,984,746,827,181,946,314,205,518）和（132,541,746,518,181,946,314,205,827,984）

（２）A．（181,132,314,205,541,518,946,827,746,984）

B．（541,132,827,746,518,181,946,314,205,984）

C．（205,132,314,181,518,746,946,984,541,827）

D．(541,132,984,746,827,181,946,314,205,518）

（３）A．O(nlog2n) B．O(n) C．log2n D．O(n2)

（４）A．O(nlog2n) B．O(n2log2n) C．O(log2n) D．O(n2)

         **参考答案** （１）B　（２）C 　（３）D 　（４）A

7．结定结点的关键字序列（F、B、J、G、E、A、I、D、C、H），对它按字母的字典顺序进行排列，采用不同方法，其最终结果相同。但中间结果是不同的。

Shell排序的第一趟扫描（步长为5）结果应为 （１） 。

冒泡排序（大数下沉）的第一趟冒泡的效果是 （２） 。

快速排序的第一次扫描结果是 （３）

二路归并排序的第一趟结局是 （４） 。

若以层次序列来建立对应的完全二叉树后，采用筛选法建堆，其第一趟建的堆是 （５） 。

（１）Ａ．（Ｂ、Ｆ、Ｇ、Ｊ、Ａ、Ｄ、Ｉ、Ｅ、Ｈ、Ｃ）

Ｂ．（Ｂ、Ｆ、Ｇ、Ｊ、Ａ、Ｅ、Ｄ、Ｉ、Ｃ、Ｈ）

Ｃ．（Ａ、Ｂ、Ｄ、Ｃ、Ｅ、Ｆ、Ｉ、Ｊ、Ｇ、Ｈ）

Ｄ．（Ｃ、Ｂ、Ｄ、Ａ、Ｅ、Ｆ、Ｉ、Ｇ、Ｊ、Ｈ）

（２）Ａ．（Ａ、Ｂ、Ｄ、Ｃ、Ｆ、Ｅ、Ｉ、Ｊ、Ｈ、Ｇ）

Ｂ．（Ａ、Ｂ、Ｄ、Ｃ、Ｅ、Ｆ、Ｉ、Ｈ、Ｇ、Ｊ）

Ｃ．（Ｂ、Ｆ、Ｇ、Ｅ、Ａ、Ｉ、Ｄ、Ｃ、Ｈ、Ｊ）

Ｄ．（Ｂ、Ｆ、Ｇ、Ｊ、Ａ、Ｅ、Ｄ、Ｉ、Ｃ、Ｈ）

（３）Ａ．（Ｃ、Ｂ、Ｄ、Ａ、Ｆ、Ｅ、Ｉ、Ｊ、Ｇ、Ｈ）

Ｂ．（Ｃ、Ｂ、Ｄ、Ａ、Ｅ、Ｆ、Ｉ、Ｇ、Ｊ、Ｈ）

Ｃ．（Ｂ、Ａ、Ｄ、Ｅ、Ｆ、Ｇ、Ｉ、Ｊ、Ｈ、Ｃ）

Ｄ．（Ｂ、Ｃ、Ｄ、Ａ、Ｅ、Ｆ、Ｉ、Ｊ、Ｇ、Ｈ）

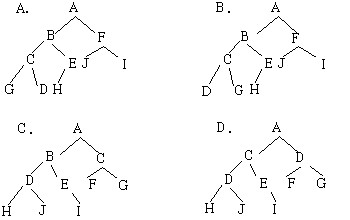
（４）Ａ．（Ｂ、Ｆ、Ｇ、Ｊ、Ａ、Ｅ、Ｄ、Ｉ、Ｃ、Ｈ）

Ｂ．（Ｂ、Ａ、Ｄ、Ｅ、Ｆ、Ｇ、Ｉ、Ｊ、Ｈ、Ｃ）

Ｃ．（Ａ、Ｂ、Ｄ、Ｃ、Ｅ、Ｆ、Ｉ、Ｊ、Ｇ、Ｈ）

Ｄ．（Ａ、Ｂ、Ｄ、Ｃ、Ｆ、Ｅ、Ｊ、Ｉ、Ｈ、Ｇ）

（５）



         **参考答案** （１）C （２）C　 （３）B　（４）A　　（５）B

8．二叉树 （1） 。在完全二叉树中，若一个结点没有 （2） ，则它必定是叶结点。每棵树都能唯一地转换成与它对应的二叉树。由树转换成的二叉树里，一个结点N的左子树是N在原树里对应结点的 （3） ，而N的右子树是它在原树里对应结点的 （4） 。二叉排序树的平均检索长度为 （5） 。

（1）A．是特殊的树 B．不是树的特殊形式

C．是两棵树的总称 D．是只有二个根结点的树形结构

（2）A．左子树 B．右子树 C．左子树或没有右子树 D．兄弟

（3）～(4)A．最左子树 B．最右子树 C．最邻近的右兄弟 D．最邻近的左兄弟

（5）A．O(n2) B．O(n) C．O(log2n) D．O(nlog2n)

         **参考答案** （1）B （2）A （3）A （4）C （5）C

9．哈希存储的基本思想是根据 （1） 来决定 （2） ，冲突（碰撞）指的是 （3） ， \_\_(4)\_\_\_越大，发生冲突的可能性也越大。处理冲突的两种主要方法是 （5） 。

（1）～（2）A．存储地址 B．元素的序号 C．元素个数 D．关键码值

（3） A．两个元素具有相同序号 B．两个元素的关键码值不同，而非码属性相同

C．不同关键码值对应到相同的存储地址 D．数据元素过多

（4） A．非码属性 B．平均检索长度 C．负载因子 D．哈希表空间

（5） A．线性探查法和双散列函数法 B．建溢出区法和不建溢出区法

C．除余法和折叠法 D．拉链法和开放地址法

         **参考答案** （1）D （2）A （3）C （4）C （5）D

10． 设二维数组F的行下标为1至5，列下标为0至8，F的每个数据元素均占4个字节。在按行存储的情况下，已知数据元素F[2，2]的第一个字节的地址是1044，则F[3，4]和F[4，3]的第一个字节的地址分别为 （1） 和 （2） ，而数组的第一个数据元素的第一个字节和数组最后一个元素的最后一个字节的地址分别为 （3） 和 （4） 。

对一般的二维数组G而言，当 （5） 时，其按行存储的G[I，J]的地址与按列存储的G[J，I]的地址相同。

（1）A．1088 B． 1084 C．1092 D．1120

（2）A．1092 B． 1088 C．1120 D．1124

（3）A．1004 B． 1044 C．1000 D．984

（4）A．1183 B． 1179 C．1164 D．1187

（5）A．G的列数与行数相同

B．G的列的上界与G的行的上界相同

C．G的列的上界与G的行的下界相同

D．G的列的上下界与G的行的上下界相同

         **参考答案** （1）A （2）C （3）C （4）B （5）D

11．某顺序存储的表格，其中有90,000个元素，已按关键字递增有序排列，现假定对各个元素进行查找的概率是相同的，并且各个元素的关键字皆不相同。

用顺序查找法查找时，平均比较次数约为 (1) ，最大比较次数为 (2) 。

现把90,000个元素按排列顺序划分成若干组，使每组有g个元素（最后一组可能不足g个）。查找时，先从第一组开始，通过比较各组的最后一个元素的关键字，找到欲查找的元素所在的组，然后再用顺序查找法找到欲查找的元素。在这种查找法中，使总的平均比较次数最小的g是 (3) ，此时的平均比较次数是 (4) 。当g的值大于等于90,000时，此方法的查找速度接近于 (5) 。

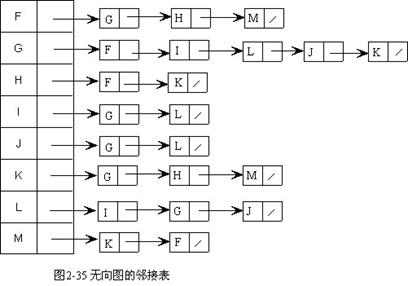
(1)～(2) A. 25,000 B. 30,000 C. 45,000 D. 90,000

(3)～(4) A. 100 B. 200 C. 300 D. 400

(5) A. 快速分类法 B. 斐波那契查找法 C. 二分法 D. 顺序查找法

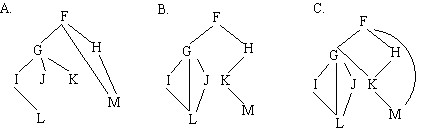
         **参考答案** （1）C （2）D （3）C （4）C （5）D

12．已知无向图的邻接表如图2-35所示：



此邻接表对应的无向图为 (1) 。此图从F开始的深度优先遍历为 （2） 。从F开始的广度优先遍历为 （3） 。从F开始的深度优先生成树为 （4） 。从F开始的广度优先生成树为 （5） 。

（1）



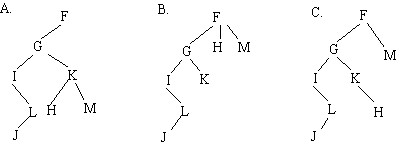
（2）A. F G I L J M K H B. F G I L J K H M

C. F G I L J K M H D. F G H M I L J K

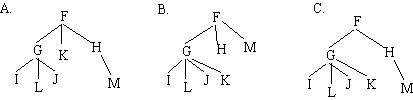
（3）A. F G I L J K M H B. F G H M I L J K

C. F G H I L J K M D. F G H M K I L J

（4）

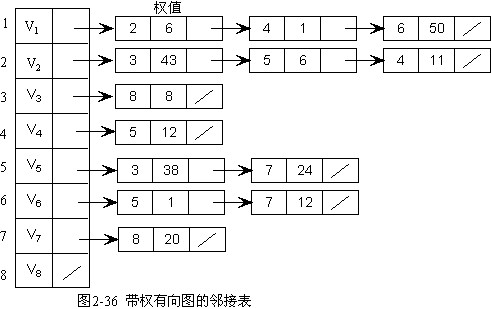


（5）



         **参考答案** （1）C （2）B （3）B （4）A （5）B

13．图2-36是带权的有向图G的邻接表。以结点V1出发深度遍历图G所得的结点序列为 （1） ；广度遍历图G所得的结点序列为 （2） ；G的一种拓扑序列是 （3） ；从结点V1到V8结点的最短路径是 （4） ；从结点V1到V8结点的关键路径是 （5） 。



（1）A. V1，V2，V3，V4，V5，V6，V7，V8 B. V1，V2，V3，V8，V4，V5，V6，V7

C. V1，V2，V3，V8，V4，V5，V7，V6 D. V1，V2，V3，V8，V5，V7，V4，V6

（2）A. V1，V2，V3，V4，V5，V6，V7，V8 B. V1，V2，V4，V6，V5，V3，V7，V8

C. V1，V2，V4，V6，V3，V5，V7，V8 D. V1，V2，V4，V6，V7，V3，V5，V8

（3）A. V1，V2，V3，V4，V5，V6，V7，V8 B. V1，V2，V4，V6，V5，V3，V7，V8

C. V1，V2，V4，V6，V3，V5，V7，V8 D. V1，V2，V4，V6，V7，V3，V5，V8

(4)～(5)A.( V1，V2，V4，V5，V3，V8) B. (V1，V6，V5，V3，V8)

C.( V1，V6，V7，V8) D. ( V1，V2，V5，V7，V8)

         **参考答案** （1）D （2）C （3）B （4）D （5）B

