第一章 概论

1. 名词解释

数据表示 2.数据处理 3.数据 4.数据元素 5.逻辑关系 6.逻辑结构7.结构

8.运算 9.基本运算 10.存储结构 11.顺序存储结构 12.链式存储结构

13.索引存储结构 14.散列存储结构 15.算法 16.运行终止的程序可执行部分

17.伪语言算法 18.非形式算法 19.时空性能 20.时间复杂性 21.数据结构

二、填空题

1.计算机专业人员必须完成的两项基本任务是：\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_。

2.数据在计算机存储器中的存在形式称为\_\_\_\_\_\_\_\_\_。

3.概括地说，数据结构课程的主要内容包括: 数据的\_\_\_\_\_\_\_\_\_\_、定义在\_\_\_\_\_\_\_\_\_、数据的\_\_\_\_\_\_\_\_\_\_的实现。此外，该课程还要考虑各种结构和实现方法的­­­\_\_\_\_\_\_\_\_\_\_。

4.由一种\_\_\_\_\_\_\_\_\_\_结构和一组\_\_\_\_\_\_\_\_\_\_构成的整体是实际问题的一种数学模型，这种数学模型的建立、选择和实现是数据结构的核心问题。

5.存储结构是逻辑结构的\_\_\_\_\_\_\_\_\_\_实现。

6.数据表示任务是逐步完成的，即数据表示形式的变化过程是\_\_\_\_\_\_\_\_\_\_->\_\_\_\_\_\_\_\_\_\_->\_\_\_\_\_\_\_\_\_\_。

7.数据处理任务也是逐步完成的，即转化过程是\_\_\_\_\_\_\_\_\_\_->\_\_\_\_\_\_\_\_\_\_->\_\_\_\_\_\_\_\_\_\_。

8.从数据结构的观点看，通常所说的"数据"应分成三个不同的层次，即\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_。

9.根据需要，数据元素又被称为\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_\_\_。

10.在有些场合下，数据项又称为\_\_\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_\_\_，它是数据的不可分割的最小标识单位。

11.从某种意义上说，数据、数据元素和数据项实际反映了数据组织的三个层次，数据

可由若干个\_\_\_\_\_\_\_\_\_\_构成，数据元素可由若干个\_\_\_\_\_\_\_\_\_\_构成。

12.根据数据元素之间关系的不同特性，通常有\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_四类基本逻辑结构，它们反映了四类基本的数据组织形式。

13.根据操作的效果，可将运算分成以下两种基本类型：

①\_\_\_\_\_\_\_\_\_\_型运算，其操作改变了原逻辑结构的“值”，如结点个数、某些结点的内容等；

②\_\_\_\_\_\_\_\_\_\_型运算，其操作不改变原逻辑结构，只从中提取某些信息作为运算的结果。

14.将以某种逻辑结构S为操作对象的运算称为“\_\_\_\_\_\_\_\_\_\_”，简称“\_\_\_\_\_\_\_\_\_\_”。

15.一般地，可能存在同一逻辑结构S上的两个运算A和B，A的实现需要或可以利用B，而B的实现不需要利用A。在这种情况下，称A可以“\_\_\_\_\_\_\_\_\_\_”为B。

16.存储实现的基本目标是建立数据的\_\_\_\_\_\_\_\_\_\_。

17.一般地，一个存储结构包括\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_三个主要部分。

18.通常，存储结点之间可以有\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_四种关联方式，称为四种基本存储方式。

19.可用任何一种存储方式所规定的存储结点之间的关联方式来间接表达给定逻辑

结构S中数据元素之间的逻辑关系。由此得到的存储结构，称为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_\_\_。

20.一个运算的实现是指一个完成该运算功能的\_\_\_\_\_\_\_\_\_\_。运算实现的核心是处

理步骤的规定，即\_\_\_\_\_\_\_\_\_\_\_。

21.任何算法都必须用某种语言加以描述。根据描述算法的语言的不同，可将算法分

为：\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_三类。

22.数据结构课程着重评论算法的\_\_\_\_\_\_\_\_\_\_\_，又称为“\_\_\_\_\_\_\_\_\_\_\_”。

23.通常从\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_等几方面评价算法的(包括程序)的质量。

24.一个算法的时空性能是指该算法的\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_，前者是算法包含的\_\_\_\_\_\_\_\_\_\_\_，后者是算法需要的\_\_\_\_\_\_\_\_\_\_\_。

25.通常采用下述办法来估算求解某类问题的各个算法在给定输入下的计算量：

1. 根据该类问题的特点合理地选择一种或几种操作作为“\_\_\_\_\_\_\_\_\_\_\_”；
2. 确定每个算法在给定输入下共执行了多少次\_\_\_\_\_\_\_\_\_\_\_，并将此次数规定为该算法在给定输入下的\_\_\_\_\_\_\_\_\_\_\_。

26.通常，一个算法在不同输入下的计算量是不同的。则可用以下两种方式来确定一个算法的计算量：

1. 以算法在所有输入下的计算量的最大值作为算法的计算量，这种计算量称为算法的\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_\_\_\_。
2. 以算法在所有输入下的计算量的加权平均值作为算法的计算量，这种计算量称为算法的\_\_\_\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_\_\_\_。

27.最坏情况时间复杂性和平均时间复杂性统称为\_\_\_\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_\_\_\_。

28.在一般情况下，一个算法的时间复杂性是\_\_\_\_\_\_\_\_\_\_\_的函数。

29.一个算法的输入规模或问题的规模是指\_\_\_\_\_\_\_\_\_\_\_。

30.常见时间复杂性的量级有：常数阶O(\_\_\_\_\_\_\_\_\_\_\_)、对数阶O(\_\_\_\_\_\_\_\_\_\_\_)、线性阶O (\_\_\_\_\_\_\_\_\_\_\_)、平方阶O(\_\_\_\_\_\_\_\_\_\_\_)、和指数阶O(\_\_\_\_\_\_\_\_\_\_\_)。通常认为，具有指数阶量级的算法是\_\_\_\_\_\_\_\_\_\_\_，而量级低于平方阶的算法是\_\_\_\_\_\_\_\_\_\_\_的。

31.数据结构的基本任务是数据结构的\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_。

32.数据结构的课程的主要内容可以概括为：\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_。

33.\_\_\_\_\_\_\_\_\_\_\_与数据元素本身的内容和形式无关。

34.从逻辑关系上讲，数据结构主要分为两大类，它们是\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_。

35.程序段“for(i=l;i<=n;i++){k++;for(j=1;j<=n;j++)l+=k;}”的时间复杂度T(n)= \_\_\_\_\_\_\_\_\_\_\_。

36.程序段“i=1;while(i<=n)i=i\*2;”的时间复杂度T(n)= \_\_\_\_\_\_\_\_\_\_\_。

三、单项选择题

1.以下说法错误的是

①用数字式计算机解决问题的实质是对数据的加工处理

②程序设计的实质是数据处理

③数据的逻辑结构是数据的组织形式，基本运算规定了数据的基本操作方式

④运算实现是完成运算功能的算法，或这些算法的设计

⑤数据处理方式总是与数据某种相应的表示形式相联系，反之亦然

2.根据数据元素之间关系的不同特性，以下四类基本的逻辑结构反映了四类基本的

数据组织形式。以下解释错误的是 ( )

①集合中任何两个结点之间都有逻辑关系但组织形式松散

②线性结构中结点按逻辑关系依次排列形成一条"锁链"

③树形结构具有分支、层次特性，其形态有点像自然界中的树

④图状结构中的各个结点按逻辑关系互相缠绕，任何两个结点都可以邻接

3.关于逻辑结构，以下说法错误的是 ( )

①逻辑结构与数据元素本身的形成、内容无关

②逻辑结构与数据元素的相对位置有关

③逻辑结构与所含结点个数无关

④一些表面上很不相同的数据可以有相同的逻辑结构

⑤逻辑结构是数据组织的某种"本质性"的东西

4.根据操作的效果，可将运算分成加工型运算、引用型运算两种基本类型。对于表格

处理中的五种功能以下解释错误的是 ( )

①查找引用型运算，功能是找出满足某种条件的结点在s(线形结构)中的位置

②读取引用型运算 功能是读出s(线形结构)中某指定位置结点的内容

③插入引用型运算，功能是在s(线形结构)的某指定位置上增加一个新结点

④删除加工型运算，功能是撤消s(线形结构)某指定位置上的结点

⑤更新加工型运算，功能是修改s(线形结构)中某指定结点的内容

5.一般地，一个存储结构包括以下三个主要部分。以下说法错误的是 ( )

①存储结点每个存储结点可以存放一个或一个以上的数据元素

②数据元素之间关联方式的表示 也就是逻辑结构的机内表示

③附加设施，如为便于运算实现而设置的“哑结点”等等

6.一般地，一个存储结构包括以下三个主要部分。以下说法错误的是

①每个存储结点只能存放一个数据元素 ( )

②数据元素之间的关联方式可由存储结点之间的关联方式直接表达

③一种存储结构可以在两个级别上讨论。其一是机器级，其二是语言级

④语言级描述可经编译自动转换成机器级 因此也可以看成是一种机内表示

7.通常从正确性、易读性、健壮性、高效性等四个方面评价算法(包括程序)的质

量。以下解释错误的是 ( )

①正确性 算法应能正确地实现预定的功能(即处理要求)

②易读性 算法应易于阅读和理解 以便于调试 修改和扩充

③健壮性 当环境发生变化时，算法能适当地做出反应或进行处理，不会产生不需要的运行结果

④高效性 即达到所需要的时间性能

8.对于数据结构课程的主要内容，以下解释正确的是 ( )

①数据结构的定义，包括逻辑结构、存储结构和基本运算集

②数据结构的实现，包括存储实现、运算实现和基本运算集

③数据结构的评价和选择，包括逻辑结构的选择、基本运算集的选择和存储

选择

9，与数据元素本身的形式、内容、相对位置、个数无关的是数据的 ( )

①存储结构 ②存储实现 ③逻辑结构 ④运算实现10顺序存储结构 ( )

①仅适合于静态查找表的存储

②仅适合于动态查找表的存储

③既适合静态又适合动态查找表的存储

④既不适合静态又不适合动态查找表的存储

11.算法的时间复杂度，都要以通过算法中执行频度最高的语句的执行次数来确定这种

观点 ( )

①正确 ②错误

12以下说法正确的是 ( )

①所谓数据的逻辑结构指的是数据元素之间的逻辑关系。

②逻辑结构与数据元素本身的内容和形式无关

③顺序文件只适合于存放在磁带上，索引文件只能存放在磁盘上

④基于某种逻辑结构之上的运算，其实现是惟一的

13以下说法正确的是 ( )

①数据元素是数据的最小单位

②数据项是数据的基本单位

③数据结构是带有结构的各数据项的集合

④数据结构是带有结构的数据元素的集合

14以下说法错误的是 ( )

①所谓数据的逻辑结构指的是数据元素之间的逻辑关系的整体

②数据的逻辑结构是指各数据元素之间的逻辑关系，是用户按使用需要而建立的③数据结构、数据元素、数据项在计算机中的映象分别称为存储结构、结点、数据域

④数据项是数据的基本单位

15通常要求同一逻辑结构中的所有数据元素具有相同的特性，这意味着 ( )

①数据元素具有同一特点

②不仅数据元素所包含的数据项的个数要相同，而且对应数据项的类型要一致

③每个数据元素都一样

④数据元素所包含的数据项的个数要相等

四、简答及应用

1数据与数据元素有何区别？

2·为什么说数据元素之间的逻辑关系是数据内部组织的主要方面?

3·逻辑结构与存储结构是什么关系?

4·运算与运算的实现是什么关系?有哪些相同点和不同点？

5，类C语言与标准C语言的主要区别是什么?

五、算法设计

1. 设计求解下列问题的类C语言算法，并分析其最坏情况时间复杂性及其量级。
   1. 在数组A[1..n]中查找值为K的元素,若找到则输出其位置i(1<=i<=n),否则输出0作为标志。
   2. 找出数组A[1..n]中元素的最大值和次最大值（本小题以数组元素的比较为标准操作）。

第二章 线性表

1. 名词解释
   1. 线性结构 2.数据结构的顺序实现 3.顺序表 4.链表 5.数据结构的链接实现

6. 建表 7.字符串 8.串 9.顺序串 10.链串

二、填空题

1.为了便于讨论，有时将含n(n>=0)个结点的线性结构表示成(a1，a2,……an)，其中每个ai代表一个\_\_\_\_\_\_。a1称为\_\_\_\_\_\_结点，an称为\_\_\_\_\_\_结点，i称为ai在线性表中的\_\_\_\_\_\_\_\_或\_\_\_\_\_\_。对任意一对相邻结点ai、ai┼1(1<=i<n),ai称为ai┼1的直接\_\_\_\_\_\_ai┼1称为ai的直接\_\_\_\_\_\_。

2.为了满足运算的封闭性，通常允许一种逻辑结构出现不含任何结点的情况。不含任何结点的线性结构记为\_\_\_\_\_\_或\_\_\_\_\_\_。

3.线性结构的基本特征是:若至少含有一个结点，则除起始结点没有直接\_\_\_\_\_\_外，其他结点有且仅有一个直接\_\_\_\_\_\_;除终端结点没有直接\_\_\_\_\_\_外，其它结点有且仅有一个直接\_\_\_\_\_\_.

4.所有结点按1对1的邻接关系构成的整体就是\_\_\_\_\_\_结构。

5.线性表的逻辑结构是\_\_\_\_\_\_结构。其所含结点的个数称为线性表的\_\_\_\_\_\_，简称\_\_\_\_\_\_.

6.表长为O的线性表称为\_\_\_\_\_\_

7.线性表典型的基本运算包括:\_\_\_\_\_\_、\_\_\_\_\_\_、\_\_\_\_\_\_、\_\_\_\_\_\_、\_\_\_\_\_\_、\_\_\_\_\_\_等六种。

8.顺序表的特点是\_\_\_\_\_\_。

9.顺序表的类型定义可经编译转换为机器级。假定每个datatype类型的变量占用k(k>=1)个内存单元，其中，b是顺序表的第一个存储结点的第一个单元的内存地址，那么，第i个结点ai的存储地址为\_\_\_\_\_\_。

10.以下为顺序表的插入运算，分析算法，请在\_\_\_\_\_\_处填上正确的语句。

Void insert\_sqlist(sqlist L，datatype x，int i)

/\*将X插入到顺序表L的第i-1个位置\*/

{ if( L.last == maxsize) error(“表满”)；

if((i<1)||(i>L.last+1))error(“非法位置”);

for(j=L.last;j>=i;j--)\_\_\_\_\_\_;

L.data[i-1]=x;

L.last=L.last+1;

}

11.对于顺序表的插入算法insert\_sqlist来说，若以结点移动为标准操作，则插入算法的最坏时间复杂性为\_\_\_\_\_\_\_\_，量级是\_\_\_\_\_\_\_\_。插入算法的平均时间复杂性为\_\_\_\_\_\_\_\_，平均时间复杂性量级是\_\_\_\_\_\_\_\_。

12.以下为顺序表的删除运算，分析算法，请在\_\_\_\_\_\_\_\_处填上正确的语句。

void delete\_sqlist(sqlist L,int i) /\*删除顺序表L中的第i-1个位置上的结点\*/

{if((i<1)||(i>L.last))error(“非法位置”)；

for(j=i+1;j=L.last;j++)\_\_\_\_\_\_\_\_;

L.last=L.last-1;

}

13.对于顺序表的删除算法delete\_sqlist来说，若以结点移动为标准操作，最坏情况时间复杂性及其量级分别是\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_,其平均时间复杂性及其量级分别为\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_。

14.以下为顺序表的定位运算，分析算法，请在\_\_\_\_\_\_\_\_处填上正确的语句。

int locate\_sqlist(sqlist L,datatype X)

/\*在顺序表L中查找第一值等于X的结点。若找到回传该结点序号；否则回传0\*/

{\_\_\_\_\_\_\_\_;

while((i≤L.last)&&(L.data[i-1]!=X))i++;

if(\_\_\_\_\_\_\_\_)return(i);

else return(0);

}

15.对于顺序表的定位算法，若以取结点值与参数X的比较为标准操作，平均时间复杂性量级为\_\_\_\_\_\_\_\_。求表长和读表元算法的时间复杂性为\_\_\_\_\_\_\_\_。

16.在顺序表上，求表长运算LENGTH（L）可通过输出\_\_\_\_\_\_\_\_实现，读表元运算

GET（L，i）可通过输出\_\_\_\_\_\_\_\_实现。

17.线性表的常见链式存储结构有\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_。

18.单链表表示法的基本思想是用\_\_\_\_\_\_\_\_表示结点间的逻辑关系。

19.所有结点通过指针的链接而组织成\_\_\_\_\_\_\_\_。

20.为了便于实现各种运算，通常在单链表的第一个结点之前增设一个类型相同的结点，称为\_\_\_\_\_\_\_\_，其它结点称为\_\_\_\_\_\_\_\_。

21.在单链表中，表结点中的第一个和最后一个分别称为\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_。头结点的数据域可以不存储\_\_\_\_\_\_\_\_，也可以存放一个\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_。

22.单链表INITIATE（L）的功能是建立一个空表。空表由一个\_\_\_\_\_\_\_\_和一个\_\_\_\_\_\_\_\_组成。

23.INITIATE()的功能是建立一个空表。请在\_\_\_\_\_\_\_\_处填上正确的语句。

lklist initiate\_lklist() /\*建立一个空表\*/

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return(t);

}

24.以下为求单链表表长的运算，分析算法，请在 \_\_\_\_\_\_\_\_处填上正确的语句。

int length\_lklist(lklist head) /\*求表head的长度\*/

{\_\_\_\_\_\_\_\_;

j=0;

while(p->next!=NULL)

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

j++;

}

return(j); /\*回传表长\*/

}

25.以下为单链表按序号查找的运算，分析算法，请在\_\_\_\_处填上正确的语句。

pointer find\_lklist(lklist head,int i)

{ p=head;j=0;

while(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{ p=p->next; j++; }

if(i==j) return(p);

else return(NULL);

}

26.以下为单链表的定位运算，分析算法，请在\_\_\_\_处填上正确的语句。

int locate\_lklist(lklist head,datatype x)

/\*求表head中第一个值等于x的结点的序号。不存在这种结点时结果为0\*/

{ p=head;j=0;

while(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_){p=p->next;j++;}

if (p->data==x) return(j);

else return(0);

}

27.以下为单链表的删除运算，分析算法，请在\_\_\_\_处填上正确的语句。

void delete\_lklist(lklist head,int i)

{ p=find\_lklist(head,i-1);

if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{ q=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

p->next=p->next;

free(q);

}

else error(“不存在第i个结点”)

}

28.以下为单链表的插入运算，分析算法，请在\_\_\_\_处填上正确的语句。

void insert\_lklist(lklist head,datatype x,int i)

/\*在表head的第i个位置上插入一个以x为值的新结点\*/

{ p=find\_lklist(head,i-1);

if(p==NULL)error(“不存在第i个位置”)；

else {s=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;s->data=x;

s->next=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

p->next=s;

}

}

29.以下为单链表的建表算法，分析算法，请在\_\_\_\_处填上正确的语句。

lklist create\_lklist1()

/\*通过调用initiate\_lklist和insert\_lklist算法实现的建表算法。假定$是结束标志\*/

{ ininiate\_lklist(head);

i=1;

scanf(“%f”,&x);

while(x!=’$’)

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

scanf(“%f”,&x);

}

return(head);

}

该建表算法的时间复杂性约等于\_\_\_\_\_\_\_\_\_\_\_\_，其量级为\_\_\_\_\_\_\_\_\_\_\_\_。

30.以下为单链表的建表算法，分析算法，请在\_\_\_\_处填上正确的语句。

lklist create\_lklist2() /\*直接实现的建表算法。\*/

{ head=malloc(size);

p=head;

scanf(“%f”,&x);

while(x!=’$’)

{ q=malloc(size);

q->data=x;

p->next=q;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

scanf(“%f”,&x);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return(head);

}

此算法的量级为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

31．除单链表之外，线性表的链式存储结构还有\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_等。

32．循环链表与单链表的区别仅仅在于其尾结点的链域值不是\_\_\_\_\_\_\_\_\_，而是一个指向\_\_\_\_\_\_\_\_\_的指针。

33．在单链表中若在每个结点中增加一个指针域,所含指针指向前驱结点，这样构成的链表中有两个方向不同的链，称为\_\_\_\_\_\_。

34．C语言规定，字符串常量按\_\_\_\_\_\_处理，它的值在程序的执行过程中是不能改变的。而串变量与其他变量不一样，不能由\_\_\_\_\_\_语句对其赋值。

35．含零个字符的串称为\_\_\_\_\_\_串，用\_\_\_\_\_\_表示。其他串称为\_\_\_\_\_\_串。任何串中所含\_\_\_\_\_\_的个数称为该串的长度。

36．当且仅当两个串的\_\_\_\_\_\_相等并且各个对应位置上的字符都\_\_\_\_\_\_时，这两个串相等。一个串中任意个连续字符组成的序列称为该串的\_\_\_\_\_\_串，该串称为它所有子串的\_\_\_\_\_\_串。

37．串的顺序存储有两种方法：一种是每个单元只存一个字符，称为\_\_\_\_\_\_格式，另一种是每个单元存放多个字符，称为\_\_\_\_\_\_格式。

38．通常将链串中每个存储结点所存储的字符个数称为\_\_\_\_\_\_。当结点大小大于1时，链串的最后一个结点的各个数据域不一定总能全被字符占满，此时，应在这些未用的数据域里补上\_\_\_\_\_\_。

三、单向选择题

1．对于线性表基本运算，以下结果是正确的是 ( )

①初始化INITIATE(L)，引用型运算，其作用是建立一个空表L=Ф

. ② 求表长LENGTH(L)，引用型运算，其结果是线性表L的长度

③读表元GET(L,i), 引用型运算。若1<=i<=LENGTH(L),其结果是线性表L的第i个结点；

否则，结果为0

④定位LOCATE(L,X), 引用型运算.若L中存在一个或多个值与X相等的结点，运算结果为这些结点的序号的最大值；否则运算结果为0

⑤插入INSERT(L,X,i),加工型运算。其作用是在线性表L的第i+1个位置上增加一个以X为值的新结点

⑥删除DELETE(L,i), 引用型运算.其作用是撤销线性表L的第i个结点Ai

2.线性结构中的一个结点代表一个 （ ）

① 数据元素 ② 数据项 ③ 数据 ④ 数据结构

3．顺序表的一个存储结点仅仅存储线性表的一个 （ ）

① 数据元素 ② 数据项 ③ 数据 ④ 数据结构

4．顺序表是线性表的 （ ）

①链式存储结构 ②顺序存储结构 ③ 索引存储结构 ④ 散列存储结构

5.对于顺序表，以下说法错误的是 （ ）

①顺序表是用一维数组实现的线性表，数组的下标可以看成是元素的绝对地址

②顺序表的所有存储结点按相应数据元素间的逻辑关系决定的次序依次排列

③顺序表的特点是:逻辑结构中相邻的结点在存储结构中仍相邻

④顺序表的特点是:逻辑上相邻的元素，存储在物理位置也相邻的单元中

6.对顺序表上的插入、删除算法的时间复杂性分析来说，通常以（ ）为标准操作

①条件判断 ②结点移动

③算术表达式 ④赋值语句

7.对于顺序表的优缺点，以下说法错误的是 （ ）

①无需为表示结点间的逻辑关系而增加额外的存储空间

②可以方便地随机存取表中的任一结点

③插人和删除运算较方便

④由于顺序表要求占用连续的空间，存储分配只能预先进行(静态分配)

⑤容易造成一部分空间长期闲置而得不到充分利用

8.指针的全部作用就是 （ ）

①指向某常量 ② 指向某变量

③指向某结点 ④存储某数据

9.除了( ) ，其它任何指针都不能在算法中作为常量出现，也无法显示。

①头指针 ②尾指针

③指针型变量 ④空指针

10.单链表表示法的基本思想是指针P表示结点间的逻辑关系，则以下说法错误的是（　　　）

①任何指针都不能用打印语句输出一个指针型变量的值

②如果要引用(如访问)p所指结点，只需写出p(以后跟域名)即可

③若想修改变量p的值(比如让P指向另一个结点)，则应直接对p赋值

④对于一个指针型变量P的值。只需知道它指的是哪个结点

⑤结点\*ｐ是由两个域组成的记录，p->data是一个数据元素，p->next的值是一个指针

11.单链表的一个存储结点包含　　　　　　　　　　　　　（　　　　）

①数据域或指针域

②指针域或链域

③指针域和链域

④数据域和链域

12.对于单链表表示法，以下说法错误的是　　　　　　　　　（　　　　）

①数据域用于存储线性表的一个数据元素

②指针域或链域用于存放一个指向本结点所含数据元素的直接后继所在结点的指针

③所有数据通过指针的链接而组织成单链表

④NULL称为空指针，它不指向任何结点，只起标志作用

13.对于单链表表示法，以下说法错误的是　　　　　　　　　　（　　　　　）

①指向链表的第一个结点的指针，称为头指针

②单链表的每一个结点都被一个指针所指

③任何结点只能通过指向它的指针才能引用

④终端结点的指针域就为NULL

⑤尾指针变量具标识单链表的作用，故常用尾指针变量来命名单链表

14．有时为了叙述方便，可以对一些概念进行简称，以下说法错误的是 （ ）

①将“指针型变量”简称为“指针”

②将“头指针变量”称为“头指针”

③将“修改某指针型变量的值”称为“修改某指针”

④将“p中指针所指结点”称为“P值”

15．设指针P指向双链表的某一结点，则双链表结构的对称性可用（ ）式来刻画

1. p->prior->next->==p->next->next
2. p->prior->prior->==p->next->prior
3. p->prior->next->==p->next->prior
4. p->next->next==p->prior->prior

16.以下说法错误的是 （ ）

①对循环链表来说，从表中任一结点出发都能通过前后操作而扫描整个循环链表

②对单链表来说，只有从头结点开始才能扫描表中全部结点

③双链表的特点是找结点的前趋和后继都很容易

④对双链表来说，结点\*P的存储位置既存放在其前趋结点的后继指针域中，也存放在它的后继结点的前趋指针域中。

17．在循环链表中，将头指针改设为尾指针（rear）后，其头结点和尾结点的存储位置分别是 （ ）

①real和rear->next->next

②rear->next 和real

③rear->next->next和rear

④rear和rear->next

18.以下说错误的是 ( )

①对于线性表来说，定位运算在顺序表和单链表上的量级均为O（n）

②读表元运算在顺序表上只需常数时间O（1）便可实现，因此顺序表是一种随机存取结构

③在链表上实现读表元运算的平均时间复杂性为O（1）

④链入、摘除操作在链表上的实现可在O（1）时间内完成

⑤链入、摘除操作在顺序表上的实现，平均时间复杂性为O（n）

19．在串的基本运算中，属于加工型运算的有 （ ）

①EQAL(S,T) ②LENGTH(S)

③CONCAT(S,T) ④REPLACE(S,T,R) ⑤INDEX(S,T)

20. 在串的基本运算中，属于引用型运算的有 （ ）

①ASSIGN(S,T) ②INSERT(S1,i,S2)

③DELETE(S,i,j) ④SUBSTR(S,i,j) ⑤REPLACE(S,T,R)

21．循环链表主要优点是 （ ）

①不再需要头指针了

②已知某个结点的位置后，能够容易找到它的直接前趋

③在进行插入、删除运算时，能更好地保证链表不断开

④从表中任一结点出发都能扫描到整个链表

22，每种数据结构都具备三个基本操作:插入、删除和查找，这种说法 （ ）

①正确 ②错误

23．以下说法错误的是 （ ）

①数据的物理结构是指数据在计算机内实际的存储形式

②算法和程序没有区别，所以在数据结构中二者是通用的

③对链表进行插人和删除操作时，不必移动结点

④双链表中至多只有一个结点的后继指针为空

24．以下说法正确的是

①线性结构的基本特征是：每个结点有且仅有一个直接前趋和一个直接后继

②线性表的各种基本运算在顺序存储结构上的实现均比在链式存储结构上的实现效率要低

③在线性表的顺序存储结构中，插人和删除元素时，移动元素的个数与该元素位置有关

④顺序存储的线性表的插人和删除操作不需要付出很大的代价，因为平均每次操只有近一半的元素需要移动

25．以下说法错误的是 （ ）

①求表长、定位这二种运算在采用顺序存储结构时实现的效率不比采用链式存储结构时实现的效率低

②顺序存储的线性表可以随机存取

③由于顺序存储要求连续的存储区域，所以在存储管理上不够灵活

④线性表的链式存储结构优于顺序存储结构

26．以下说法错误的是 （ ）

①线性表的元素可以是各种各样的，逻辑上相邻的元素在物理位置上不一定相邻

②在线性表的顺序存储结构中，逻辑上相邻的两个元素在物理位置上不一定相邻

③在线性表的链式存储结构中，逻辑上相邻的元素在物理位置上不一定相邻

④线性表的链式存储结构的特点是用一组任意的存储单元存储线性表的数据元素

27．以下说法正确的是（ ）

①在单链表中，任何两个元素的存储位置之间都有固定的联系，因为可以从头结点进行查找任何一个元素

②在单链表中，要取得某个元素，只要知道该元素的指针即可，因此，单链表是随机存取的存储结构

③顺序存储结构属于静态结构，链式结构属于动态结构

④顺序存储方式只能用于存储线性结构

28.以下说法正确的是（ ）

①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

②链表的每个结点中都恰好包含一个指针

③线性表的顺序存储结构优于链式存储结构

④顺序存储结构属于静态结构，链式结构属于动态结构

29.下面关于线性表的叙述正确的是( )

①线性表采用顺序存储，必须占用一片连续的存储单元

②线性表采用顺序存储，便于进行插人和删除操作

③线性表采用链接存储，不必占用一片连续的存储单元

④线性表采用链接存储，不便于插人和删除操作

30.线性表L=(a1,a2,...,ai,...,an),下列说法正确的是( )

①每个元素都有一个直接前驱和直接后继

②线性表中至少要有一个元素

③表中诸元素的排列顺序必须是由小到大或由大到小的

④除第一个元素和最后一个元素外其余每个元素都有一个且仅有一个直接前驱和直接后继

31.线性表的逻辑顺序与存储顺序总是一致的，这种说法( )

①正确 ②不正确

32.设p,q是指针,若p=q,则\*p=\*q ,这种说法( )

①正确 ②不正确

33.线性表若采用链表存储结构时，要求内存中可用存储单元的地址( )

①必需是联系的 ②部分地址必须是连续的

③一定是不连续的 ④连续不连续都可以

34.设REAR是指向非空带头结点的循环单链表的尾指针,则删除表首结点的操作可表示为( )

①p=rear; ②rear=rear->next;

rear=rear->next; free(rear);

free(p)

③rear=rear->next->next; ④ p=rear->next->next;

free(rear); rear->next->next=p->next;

free(p);

35. 单链表中，增加头结点的目的是为了 ( )

①使单链表至少有一个结点 ②标示表结点中首结点的位置

③方便运算的实现 ④说明单链表是线性表的链式存储实现

36线性结构中的一个结点代表一个数据元素，通常要求同一线性结构的所有结点所代表的数据元素具有相同的特性，这意味着

1. 每个结点所代表的数据元素都一样。
2. 每个结点所代表的数据元素包含的数据项的个数要相等
3. 不仅数据元素包含的数据项的个数要相同，而且对应数据项的类型要一致
4. 结点所代表的数据元素有同一特点

37．带头结点的单链表Head为空的判定条件是

①Head=Null ②Head->next=NULL ③Head->next=Head

38.非空的单循环链表L的尾结点\*P,满足

P->next=NULL P=NULL P->next=L P=L.

39.双向链表结点结构如下：

|  |  |  |
| --- | --- | --- |
| LLink | data | RLink |

其中：LLink是指向前驱结点的指针域：

data是存放数据元素的数据域；

Rlink是指向后继结点的指针域。

下面给出的算法段是要把一个新结点\*Q作为非空双向链表中的结点\*p的前驱，插入到此双向链表中。不能正确完成要求的算法段是

①Q->LLink=P->LLink; ②P->LLink=Q;

Q->Rlink=P; Q->Rlink=P;

P->LLink=Q; P->LLink->Rlink=Q;

P->LLink->Rlink=Q; Q->LLink=P->LLink;

③Q->LLink=P->LLink;

Q->Rlink=P;

P->LLink->Rlink=Q;

P->LLink=Q;

40.若某线性表中最常用的操作是取第i个元素和找第i个元素的前趋元素，则采用（ ）存储方式最节省时间。

①顺序表 ②单链表 ③双链表 ④单循环链表

41．串是任意有限个

①符号构成的集合 ②符号构成的序列

③字符构成的集合 ④字符构成的序列

四、简答及应用

1. 请用类C语言描述顺序表，并予以解释说明。
2. 请用类C语言描述单链表的类型定义，并予以解释说明。
3. 请用类C语言描述双链表的类型定义，并予以解释说明。
4. 请用类C语言描述顺序串的类型定义。
5. 请用类C语言描述链串的类型定义。

6.叙述以下概念的区别：头指针变量、头指针、头结点、首结点，并说明头指针变量和头结点的作用。

7．有哪些链表可仅由一个尾指针来惟一确定，即从尾指针出发能访问到链表上任何一个结点。

8．简述下列每对术语的区别：

空串和空格串;串变量和串常量;主串和子串;串变量的名字与串变量的值。

9．设有 A=” ”,B="mule"，C="old"，D="my",试计算下列运算的结果(注:A+B是CONCAT（A,B）的简写,A=" "的 " "含有两个空格)。

(a) A+B

(b) B+A

(c) D+C+B

(d) SUBSTR(B,3,2)

(e) SUBSTR(C,1,0)

(f) LENGTH(A)

(g) LENGTH(D)

(h) INDEX(B,D)

(i) INDEX(C,"d")

(j) INSERT(D,2,C)

(k) INSERT(B,1,A)

(l) DELETE(B,2,2)

(m) DELETE(B,2,0)

10.已知:S="(xyz)\*",T="(x+z)\*y"。试利用连接、求子串和置换等基本运算，将S转换为T。

五、算法设计

1． 设A=（a1,a2,a3,......an）和B=(b1,b2,.. .,bm)是两个线性表（假定所含数据元素均为整数）。若n=m且ai=bi(i=1,.. .,n),则称A=B;若ai=bi(i=1,.. .,j)且aj+1<bj+1（j<n<=m）, 则称A<B;在其他情况下均称A>B。是编写一个比较A和B的算法，当A<B,A=B或A>B是分别输出-1，0或者1。

2，试编写在无头结点的单链表上实现线性表基本运算LOCATE(L,X)、INSERT(L,X，i)和DELETE(L,i)的算法，并和在带头结点的单链表上实现相的算法进行比较。

3．试编写在不带头结点的单链表上实现线性表基本运算LENGTH(L)的算法。

4．假设有两个按数据元素值递增有序排列的线性表A和B，均以单链表作存储结构。编写算法将A表和B表归并成一个按元素值递减有序（即非递增有序，允许值相同）排列的线性表C，并要求利用原表(即A表和B表的)结点空间存放表C。

5．设有线性表A=(a1,a2,.. .,am)和B=(b1,b2,.. .,bn).试写合并A、B为线性表C的算法，使得：

C=

假设A、B均以单链表为存储结构(并且m、n显示保存)。要求C也以单链表为存储结构并利用单链表A、B的结点空间。

6，设线性表存放在向量A[arrsize]的前elenum分量中，且递增有序。试写一算法，将X插入到线性表的适当位置上，以保持线性表的有序性，并且分析算法的时间复杂性。

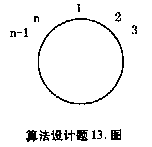
7．已知单链表L中的结点是按值非递减有序排列的，试写一算法将值为x的结点插入表L中，使得L仍然有序。

8，试分别以顺序表和单链表作存储结构，各写一个实现线性表的就地(即使用尽可能少的附加空间)逆置的算法，在原表的存储空间内将线性表(a1，a2,.. .,an)逆置为(an,.. .,a2,a1)。

9．假设分别以两个元素值递增有序的线性表A、B表示两个集合(即同一线性表中的元素各不相同)，现要求构成一个新的线性表C，C表示集合A与B的交，且C中元素也递增有序。试分别以顺序表和单链表为存储结构，填写实现上述运算的算法。

10，已知A、B和C为三个元素值递增有序的线性表，现要求对表A作如下运算： 删去那些既在表B中出现又在表C中出现的元素。试分别以两种存储结构(一处种顺序结构，一种链式的)编写实现上述运算的算法。

11．假设在长度大于1的循环链表中，既无头结点也无头指针。s为指向链表中某个结点的指针，试编写算法删除结点\*s的前趋结点。

12．已知一单链表中的数据元素含有三个字符(即:字母字符、数字字符和其它字符)。试编写算法，构造三个循环链表，使每个循环链表中只含同一类的字符，且利用原表中的结点空间作为这三个表的结点空间(头结点可另辟空间)。13.(Josephus环)任给正整数n、k，按下述方法可得排列1，2，……,n的一个置换:将数字1，2，.. .,n环形排列(如图算法设计题13.所示)，按顺时针方向从1开始 计数；计满K时输出该为之上的数字(并从环中删去该数字)，然后从下一个数字开始继续计数，直到环中所有数字均被输出为止。例如，n=10,k=3时，输出的置换是3，6，9，2，7，1，8，5，10，

4。试编写一算法，对输人的任意正整数n、k，输出相应的置换

14·在双链表上实现线性表的下列基本运算(a)初始化； (b)定位(c)插入(d)删除。

15·设有一双链表，每个结点中除有prior、data和next三个域外，还有一个访问频度域freq，在链表被起用之前，其值均初始化为零。每当在双链表上进行一次LOCATEL，X)运算时，令元素值为X的结点中freq域的值增1，并使此链表中结点保持按访问频度递减的顺序排列，以便使频繁访问的结点总是靠近表头。试编写实现符合上述要求的LOCATE运算的算法。

16·若X和Y是用结点大小为1单链表表示的串，设计一个算法找出X中第一个不在y中出现的字符。

17．在顺序串上实现串的判等运算EQUAL(S,T)。

18．在链串上实现判等运算EQUAL(S,T)。

19．若S和T是用结点大小为1的单链表存储的两个串（S、T为头指针），设计一个算法将串S中首次与串T匹配的子串逆值。

20．用串的其他运算构造串的子串定位运算index。

第三章 栈、队列和数组

1. 名词解释：

1.栈、栈顶、栈底、栈顶元素、空栈2.顺序栈3.链栈4.递归5.队列、队尾、队头6.顺序队7.循环队8.队满9.链队10.随机存储结构11.特殊矩阵12.稀疏矩阵13.对称方阵14.上（下）三角矩阵

1. 填空题：
   1. 栈修改的原则是\_\_\_\_\_\_\_\_\_或称\_\_\_\_\_\_\_\_，因此，栈又称为\_\_\_\_\_\_\_\_线性表。在栈顶进行插入运算，被称为\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_，在栈顶进行删除运算，被称为\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_。
   2. 栈的基本运算至少应包括\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_五种。
   3. 对于顺序栈，若栈顶下标值top=0,此时，如果作退栈运算，则产生“\_\_\_\_\_\_\_\_”。
   4. 对于顺序栈而言，在栈满状态下，如果此时在作进栈运算，则会发生“\_\_\_\_\_\_\_\_”。
   5. 一般地，栈和线性表类似有两种实现方法，即\_\_\_\_\_\_\_\_实现和\_\_\_\_\_\_\_\_实现。
   6. top=0表示\_\_\_\_\_\_\_\_，此时作退栈运算，则产生“\_\_\_\_\_\_\_\_”；top=sqstack\_maxsize-1表示\_\_\_\_\_\_\_\_，此时作进栈运算，则产生“\_\_\_\_\_\_\_\_”。
   7. 以下运算实现在顺序栈上的初始化，请在\_\_\_\_\_\_\_\_处用适当的句子予以填充。

int InitStack(SqStackTp \*sq)

{ \_\_\_\_\_\_\_\_;

return(1);}

* 1. 以下运算实现在顺序栈上的进栈，请在\_\_\_\_\_\_\_\_处用适当的语句予以填充。

Int Push(SqStackTp \*sq,DataType x)

{ if(sp->top==sqstack\_maxsize-1}{error(“栈满”);return(0);}

else{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=x;

return(1);}

}

* 1. 以下运算实现在顺序栈上的退栈，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_用适当句子予以填充。

Int Pop(SqStackTp \*sq,DataType \*x)

{if(sp->top==0){error(“下溢”);return(0);}

else{\*x=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return(1);}

}

10. 以下运算实现在顺序栈上判栈空，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Int EmptyStack(SqStackTp \*sq)

{if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) return(1);

else return(0);

}

11.以下运算实现在顺序栈上取栈顶元素，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Int GetTop(SqStackTp \*sq,DataType \*x)

{if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) return(0);

else{\*x=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return(1);}

}

12. 以下运算实现在链栈上的初始化，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用请适当句子予以填充。

Void InitStacl(LstackTp \*ls){ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;}

13.` 以下运算实现在链栈上的进栈，请在处用请适当句子予以填充。

Void Push（LStackTp \*ls,DataType x）

{ LstackTp \*p;p=malloc(sizeof(LstackTp));

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

p->next=ls;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

14．以下运算实现在链栈上的退栈，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用请适当句子予以填充。

Int Pop(LstackTp \*ls,DataType \*x)

{LstackTp \*p;

if(ls!=NULL)

{ p=ls;

\*x=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

ls=ls->next;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return(1)；

}else return(0);

}

15. 以下运算实现在链栈上读栈顶元素，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用请适当句子予以填充。

Int Get Top(LstackTp \*ls,DataType \*x)

{ if(ls!=NULL){ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;return(1);}

else return(0);

}

16.必须注意，递归定义不能是“循环定义”。为此要求任何递归定义必须同时满足如下条件：

①被定义项在定义中的应用（即作为定义项的出现）具有\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_；

②被定义项在最小“尺度”上的定义不是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_的。

17．队列简称\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。在队列中，新插入的结点只能添加到\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_，被删除的只能是排在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_的结点。

18．队列以线性表为逻辑结构，至少包括\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_、五种基本运算。

19．顺序队的出、入队操作会产生“\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”。

20．以下运算实现在循环队上的初始化，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Void InitCycQueue(CycqueueTp \*sq)

{ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;sq->rear=0;}

21. 以下运算实现在循环队上的入队列，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用请适当句子予以填充。

Int EnCycQueue(CycquereTp \*sq,DataType x)

{ if((sq->rear+1)%maxsize== \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{error(“队满”);return(0);

else{ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return(1);

}

22. 以下运算实现在循环队上的出队列，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Int OutCycQueue(CycquereTp \*sq,DataType \*x)

{if(sq->front== \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_){error(“队空”);return(0)；}

else{ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return(1);

}

}

23. 以下运算实现在循环队上判队空，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Int EmptyCycQueue(CycqueueTp sq)

{if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) return(1);

else return(0);

}

24. 以下运算实现在循环队上取队头，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Int GetHead(CycqueueTp sq,DataType \*x)

{ if(sq.rear== \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_return(0);

else{ \*x=sq.data[\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ];

return(1);

}

25.链队在一定范围内不会出现\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_的情况。当lq.front==lq.rear试，队中无元素，此时\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

26．以下运算实现在链队上的初始化，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

void InitQueue(QueptrTp \*lp)

{ LqueueTp \*p;

p=(LqueueTp \*)malloc(sizeof(LqueueTp))；

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_；

lq->rear=p;

(lq->front)->next=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

27. 以下运算实现在链队上的入队列，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Void EnQueue(QueptrTp \*lq,DataType x)

{ LqueueTp \*p;

p=(LqueueTp \*)malloc(sizeof(LqueueTp));

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=x;

p->next=NULL;

(lq->rear)->next=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

28. 以下运算实现在链队上的出队列，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

int OutQueue(QuetrTp \*lq,DataType \*x)

{ LqueueTp \*s;

if(lq->front==lq->rear){erroe(“队空”)；return(0);}

else { s=(lq->front)->next;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=s->data;

(lq->front)->next=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_；

if(s->next==NULL) lq->rear=lq->front;

free(s);

return(1);

}

}

29. 以下运算实现在链队上判队空，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充

int EmptyQueue(QueptrTp \*lq)

{ if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) return(1)；

else return(0);

}

30. 以下运算实现在链队上读队头元素，请在\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_处用适当句子予以填充。

Int GetHead(QueptrTp lq,DataType \*x)

{ LqueueTp \*p;

if(lq.rear==lq.front) return(0);

else{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ =p->data;

return(1);

}

}

31．一般地，一个n维数组可视为其数据元素为\_\_\_\_\_\_\_\_\_\_\_维数组的线性表。数组通常只有\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_两种基本运算。

32，通常采用\_\_\_\_\_\_\_\_\_\_\_存储结构来存放数组 。对二维数组可有两种存储方法：一种是以\_\_\_\_\_\_\_\_\_\_\_为主序的存储方式，另一种是以\_\_\_\_\_\_\_\_\_\_\_为主序的存储方式。C语言数组用的是以\_\_\_\_\_\_\_\_\_\_\_序为主序的存储方法；FORTRAN语言用的是以\_\_\_\_\_\_\_\_\_\_\_序为主序的存储方法

33．需要压缩存储的矩阵可分为\_\_\_\_\_\_\_\_\_\_\_矩阵和\_\_\_\_\_\_\_\_\_\_\_矩阵两种。

34．对称方阵中有近半的元素重复， 若为每一对元素只分配一个存储空间 ，则可将n2个元素压缩存储到\_\_\_\_\_\_\_\_\_\_\_个元素的存储空间中。

35．假设以一维数组M（1：n(n+1)/2）作为n阶对称矩阵A的存储结构，以行序为主序存储其下三角（包括对角线）中的元素，数组M和矩阵A间对应的关系为\_\_\_\_\_\_\_\_\_\_\_。

36．上三角矩阵中，主对角线上的第t行(1<=t<=n)有\_\_\_\_\_\_\_\_\_\_\_个元素，按行优先顺序存放上三角矩阵中的元素aij时，aij之前的前i-1行共有\_\_\_\_\_\_\_\_\_\_\_个元素，在第i行上， aij是该行的第\_\_\_\_\_\_\_\_\_\_\_个元素，M[k]和aij的对应关系是。

当i>j时，aij=c,c存放在M[\_\_\_\_\_\_\_\_\_\_\_]中。

37．下三角矩阵的存储和对称矩阵类似。M[K]和aij的对应关系是\_\_\_\_\_\_\_\_\_\_\_。

38．基于三元组的稀疏矩阵转置的处理方法有两种，以下运算按照矩阵A的列序来进行转置，请在\_\_\_\_\_\_\_\_\_\_\_处用适当的句子用以填充。

Trans\_Sparmat(SpMatrixTp a,SpMatrixTp \*b)

{ (\*b).mu=a.nu;(\*b).nu=a.mu;(\*b).tu=a.tu;

if(a.tu)

{ q=1;

for(col=1; \_\_\_\_\_\_\_\_\_\_\_;col++)

for(p=1;p<=a.tu;p++)

if(\_\_\_\_\_\_\_\_\_\_\_==col)

{(\*b).data[q].i=a.data[p].j;

(\*b).data[q].j=a.data[p].i;

(\*b).data[q].v=a.data[p].v;

\_\_\_\_\_\_\_\_\_\_\_;

}

}

39．基于三元组的稀疏矩阵转置的处理方法有两种，以下计算按照矩阵A的三元组a.data的次序进行转置，请在\_\_\_\_\_\_\_\_\_\_\_处用适当的句子用以填充。

Fast\_Trans\_Sparmat(SpMatrixTp a,SpMatrixTp \*b)

{ (\*b).mu=a.nu;(\*b).nu=a.mu;(\*b).tu=a.tu=a.tu;

if(a.tu)

{for(col=1;\_\_\_\_\_\_\_\_\_\_\_;col++) num[col]=0;

for(t=1;t<=a,tu;t++) num[a.data[t].j]++;

cpot[1]=1;

for(col=2;col<=a.nu;col++) cpot[col]=\_\_\_\_\_\_\_\_\_\_\_;

for(p=1;p<=a.tu;p++)

{ col=a.data[p].j;

q=cpot[col];

(\*b).data[q].i=a.data[p].j;

(\*b).data[q].j=a.data[p].i;

(\*b).data[q].v=a.data[p].v;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

}

}

40．栈称为\_\_\_\_\_\_\_\_\_\_\_线性表。 ;

41．队称为\_\_\_\_\_\_\_\_\_\_\_线性表。

42设一个链栈的栈顶指针为ls，栈中结点的格式为 info next,栈空的条件是\_\_\_\_\_\_\_\_\_\_\_；如果栈不为空，则退栈操作为p=ls; \_\_\_\_\_\_\_\_\_\_\_;ls=ls->next;free(p)。

43．设有二为数组int M[10][20](注：m为0...10,n为0...20),每个元素（整数）栈两个存储单元，数组的起始地址为2000，元素M[5][10]的存储位置为\_\_\_\_\_\_\_\_\_\_\_，M[8][19]的存储值为\_\_\_\_\_\_\_\_\_\_\_。

44．在具有n个单元的循环队列中，队满时共有\_\_\_\_\_\_\_\_\_\_\_个元素。

45．\_\_\_\_\_\_\_\_\_\_\_可以作为实现递归函数调用的一种数据结构。

46．数组M中每个元素的长度是3个字节，行下标i从1到8，列下标j从1到0，从首的址EA开始连续存放在存储其中。若按行方式存放，元素M[8][5]的起始地址为\_\_\_\_\_\_\_\_\_\_\_；若按列优先方式存放，元素M[8][5]的地址为\_\_\_\_\_\_\_\_\_\_\_。

47．对带有头结点的列队列lq,判定队列中只有一个数据元素的条件是\_\_\_\_\_\_\_\_\_\_\_。

48．二维数组M的成员是6个字符（每个字符栈一个存储单元）组成的串，行下标i的范围从0到8，列下标j的范围从1到10，则存放M至少需要\_\_\_\_\_\_\_\_\_\_\_个字节；M的第8列和第5行共占\_\_\_\_\_\_\_\_\_\_\_个字节；若M按行方式存储，元素M[8][5]的起始地址与当M按列优先方式存储时的\_\_\_\_\_\_\_\_\_\_\_元素的起始地址一致。

三、单项选择题

1．在以下栈的基本运算中，不是加工型运算的是 （ ）

①lnitStack(S) ②Push(S,X) ③Pop(S) ④empty(S)

2.以下说法正确的是 ( )

①因链栈本身没有容量限制,故在用户内存空间的范围内不会出现栈满情况

②因顺序栈本身没有容量限制,故在用户内存空间的范围内不会出现栈满情况

③对于链栈而言,在栈满状态下,如果此时再作进栈运算，则会发生“上溢”

④对于顺序栈而言在栈满状态下如果此时再作迸栈运算，则会发生“下溢”。

3．在以下队列的基本运算中，不是加工型运算的是 （ ）

①InitQueue(Q) ②EnQueue(Q,X) ③OutQueu(Q,X) ④GetHead(Q,x)

4.顺序队列的人队操作应为 ( )

①sq.rear=sq.rear+1 sq.data[sq.rear]=x

②sq.data[sq.rear]=x sq.rear=sq.rear+1

③sq.rear=(sq.rear+1)% maxsize; sq.data[sq.rear]=x

④sq.data[sqrear]=x sq.rear=(sq.rear+1)% maxsize

5.循环队列的人队操作应为 ( )

①sq.rear=sq.rear+1 sq.data[sq.rear]=x

②sq.data[sq.rear]=x sq.rear=sq.rear+1

③sq.rear=(sq.rear+1)% maxsize sq.data[sq.rear]=x

④sq.data[sq.rear]=x sq.rear=(sq.rear+1)% maxsize

6. 顺序队列的出队操作为 ( )

①sq.front=(sq.front+1)% maxsize

②sq.front=sq.front+1

③sq.rear=(sq.rear+1)% maxsize

④sq.rear=sq.rear+1

7. 循环队列的出队操作为 ( )

①sq.front=(sq.ftont+1)% maxsize

②sq.front=sq.front+1

③sq.rear=(sq.rear+)% maxsize

④sq.rear=sq.rear+1

8.循环队列的队满条件为 ( )

①(sq.rear+1) % mazsize ==(sq.front+1) % maxsize;

②(sq.rear+1 % maxsize ==sq.front+1

③sq.(rear+1) % maxsize ==sq.front

④sq.rear ==sq.front

9.循环队列的队空条件为 ( )

①(sq.rear+1) % maxsize ==(sq.front+1) % maxsize

②(sq.rear+) % maxsize ==sq.front+1

③(sp.rear+1) % maxsize ==sq.front

④sq.rear == sq.front

10.数组的数据元素类型DataType可根据实际需要而定义。以下说法完全正确的是 ( )

①数组的读运算可以读取一个数据元素整体,写运算只能修改一个数据元素的一部分

②数组的读、写运算可以读取或修改一个数据元素的一部分或一个整体

③数组的读、写运算只能读取或修改一个数据元素的一部分

④数组的读、写运算只能读取或修改一个数据元素整体

11．对于以行序为主序的存储结构来说，在数组A[c1···d1，c2···d2]中，c1和d1分别为

数组A的第一个下标的上、下界，c2…d2分别为第二各下标的上、下界，每个数据元素占K

个存储单元，二维数组中任一元素a[i,j]的存储位置可由( )式确定.

①Loc[i,j]=[( d2-c2+1)(i- c1)+(j- c2)]\*k

②Loc[i,j]=loc[c1, c2]+[( d2- c2+1)(i- c1)+(j- c2)]\*k

③Loc{i,j}=A[c1, c2]+[( d2- c2+1)(i- c1)+(j- c2)]\*k

④Loc[i,j]=loc[0,0]+[( d2- c2+1)(i- c1)=(j- c2)]\*k

12对于C语言的二维数组DataType A[m][n],每个数据元素占K个存储单元，二维数组中任意元素a[i,j] 的存储位置可由( )式确定.

①Loc[i,j]=A[m,n]+[(n+1)\*i+j]\*k

②Loc[i,j]=loc[0,0]+[(m+n)\*i+j]\*k

③Loc[i,j]=loc[0,0]+[(n+1)\*i+j]\*k

④Loc[i,j]=[(n+1)\*i+j]\*k

13.线性表的顺序存储结构是一种( )的存储结构,线性表的链式存储结构是一种（ ）的存储结构。

① 随机存取 ② 顺序存储

14．如果以链表作为栈的存储结构，则退栈操作是 ( )

①必须判别栈是否满 ②必须判别栈是否空

③判别栈元素的类型 ④对栈不做任何操作

15对于基于三元组的稀疏矩阵转置的处埋方法以下说法正确的是 ( )

①按照矩阵A的列序来进行转置，算法的时间复杂度为0(nu+tu)

②按照A的三元组a.data的次序进行转置，算法的时间复杂度为O(nu\*tu)

③按照矩阵A的列序来进行转置的方法称快速转置

④按照矩阵A的列序进行转置,对于tu<<mu x nu才有意义。

16．稀疏矩阵的压缩存储方法是只存储 ( )

①非零元素 ② 三元祖（i,j, aij） ③ aij ④ i,j

17．基于三元组的稀疏矩阵，对每个非零元素aij,可以用一个（ ）唯一确定。

①非零元素 ②三元组(i,j,aij) ③ aij ④ i,j

18如果以链表作为栈的存储结构,则退栈操作时 ( )

①必须判别栈是否满 ②判别栈元素的类型

③必须判别栈是否空 ④ 队栈不做任何判别

19．设C语言数组Data[m+1]作为循环队列SQ的存储空间， front为队头指针，rear为队为指针，则执行出队操作的语句为 （ ）

①front=front+1 ② front=（front+1）%m

③rear=(rear+1)%m ④ front=(front+1)%(m+1)

20.三角矩阵可压缩存储到数组（ ）中。

①M[1:n(n+1)/2+1] ② M[1:n(n+1)/2]

③M[n(n+1)/2+1] ④M[n(n+1)/2]

21.设有一顺序栈S，元素s1,s2,s3,s4,s5,s6依次进栈，如果6个元素出线的顺序是s2,s3,s4, s6 , s5,s1,则栈的容量至少应该是 （ ）

①2 ② 3 ③ 5 ④6

22．设有一顺序栈已含3个元素，如下图所示，元素a4正等待进栈。那么下列4个序列

中不可能出现的出栈序列是 （ ）

0 1 2 3 maxsize-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 |  |  |  |

sq

↑top

①a3,a1,a4,a2 ②a3,a2,a4,a1 ③ a3,a4,a2,a1 ④a4,a3,a2,a1

23.向一个栈顶指针为Top的链中插入一个s所指结点时，其操作步骤为 （ ）

①Top->next=s ② s->next=Top->next;Top->next=s

③s->next=Top;Top=s ④ s->next=Top;Top=Top->next

24.从栈顶指针为Top的链栈中删除一个结点，并将被删节点的值保存到x中，其操作步骤为（ ）

①x=Top->data;Top=Top->next ②Top=Top->next;x=Top->data

③x=Top;Top=Top->next ④ x=Top->data

25.在一个链队中，若f,r分别为队首、队尾指针，则插入s所指结点的操作为（ ）

①f->next=c;f=s ②r->next=s;r=s

③s->next=r;r=s ④ s->next=f;f=s

26常对数组进行的两种基本操作是 （ ）

①建立与删除 ② 索引与修改 ③ 查找与修改 ④ 查找与索引

27．链栈与顺序栈相比，有一个比较明显的优点即 （ ）

①插入操作更方便 ② 通常不会出现栈满的情况

③不会出现栈空的情况 ④ 删除操作更方便

28．若采用三元组压缩技术存储稀疏矩阵，只要把每个元素的行下标和列下标互换，就完成了对该矩阵的转置运算，这种观点 （ ）

①正确 ②错误

29。二为数组M[i,j]的元素是4个字符（每个字符占一个存储单元）组成的串，行下标i的范围从O到4，列下标j的范围从O到5。M按行存储时元素M[3,5] 的起始地址与M按列存储时元素( )的起始地址相同。

①M [2,4] ② M[3,4] ③M[3,5] ④M[4,4]

30.一个栈的入栈序列是a,b,c,d,e,则栈的不可能的输出序列是 （ ）

① e d c b a ②d e c b a ③d c e a b ④a b c d e

31.一个队列的人列序是1，2，3，4，则队列的输出系列是 （ ）

① 4，3，2，1 ② 1，2，3，4， ③1，4，3，2 ④ 3，2，4，1

32．设计一个判别表达式中左、右括号是否配对出线的算法，采用（ ）数据结构最佳。

①线性标的顺序存储结构 ②栈

③ 队列 ④ 线性表的链式存储结构

33．若已知一个栈的输入序列为1，2，3，...,n,其输出序列为P1、P2、...Pn。若p1=n,则P1为

①i ②n=i ③ n-i+1 ④ 不确定

34．以下说法正确的是

①顺序队和循环队的队满和队空判断条件是一样的

②栈可以作为实现过程调用的一种数据结构

③插人和删除操作是数据结构中最基本的两种操作，所以这两种操作在数组中也经常使用

④在循环队列中，front指向队列中第一个元素的前一位置，rear指向实际的队尾元素，队列为满的条件front=rear

35. 以下说法正确的是

①数组是同类型值的集合

②数组是一组相继的内存单元

③数组是一种复杂的数据结构，数组元素之间的关系，既不是线性的，也不是树形的

④使用三元组表表示稀疏矩阵的元素，有时并不能节省存储空间

四、简答及应用

1．简述顺序栈的类型定义。

2．简述链栈的类型定义。

3．简述顺序队列、循环队列的类型定义。

4．简述链队的类型定义。

5，写出基于三元组的稀疏矩阵的类型说明。

6．对于循环队列，试写出求队列长度的算法。

7．设有编号为t，2，3，4的四辆列车。顺序进入一个占世界共的展台，试写出这四两列车开出车站的所有可能的顺序。

8设有上三角矩阵(aij)n\*n,将其上三角元素逐行存于数组B(1:m)中（m充分大），使得B[k]=aij,且k=f1(i)+f2(j)+c。是推导出函数f1,f2和常数c(要求f1和f2中不含常数项)。

9．设有三对角矩阵（aij）n\*n,将其三条对角线上的元素逐行存于数组B(1:3n-2)中，使得B[k]=aij,求：

1. 用i、j表示k的下标变换公式；
2. 用k表示i、j的下标变换公式.

10.阅读下列程序，写出程序的运行结果。

# define sqstack\_maxsize 40

typedef struct sqstack

{ char data[sqstack\_maxsize];

int top;

} SqStackTp;

main()

{ SqStackTp sq;

int i;

char ch;

InitStack(&sq);

For(ch=’A’;ch<=’A’+12;ch++)

{ Push(&sq,ch);

printf(“%c”,ch);

}

printf(“\n”);

while(!EmptyStack(sq))

{ Pop(&sq,&ch);

printf(“&c”,ch);

} printf(“\n”);

}

11.阅读下列算法，写出其完整的功能。

Void reverse\_list( LinkedListTP \*head)

{ LstackTP ls,p;

DataType x;

InitStack(&ls);

p=head->next;

While(p!=NULL)

{ Push(&ls<p->data);

p=p->next;}

p=head->next;

While(! EmptyStack(&ls))

{ Pop(&l,&x); p->data=x;

p=p-next;}

}

12，对下列函数，按照《数据结构导论》课本的图3-5失利，画出调用f(5)是引起的工作栈状态变化情况。

Int f(int I)

{ if(n==1) return(10);

else return(f(I-1)+2);

}

五、算法设计

1.某汽车轮渡口，过江渡船每次能载10辆车过江。过江车辆分为客车类和货车类，

上渡船的有如下规定：同类车先到先上船；且每上4辆客车，才允许上一辆货车；若等待客车不足4辆，则以火车代替，若无货车等待允许客车都上船。试写一算法模拟渡口管理。

可以在一个数组中保存两个栈：一个栈以数组的第一个单元作为栈底，另一个栈以数组的最后一个单元作为栈底（如下图所示）。设S是其中一个占，是分别编写过程push(S,X)(元素X入栈)， 出栈pop(S)和取栈顶元素Top(S).题示：设其中一个栈为0，另一栈为1。

0 1 2 M-1 M M+1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | ······ |  |  |

栈0底 栈0顶 栈1顶 栈1底

3．假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾元素结点（注意不设头指针），试编写相应的初始化队列、入队列和出队列算法。

4．假设以数组cycque[m](假设数组范围在0..m)存放循环队列的元素，同时设变量rear和quelen分别指示循环队列中队尾元素位置和内含元素的个数。试给出此循环队列的队满条件，并写出相应的入队列和出队列的算法。

5．编写算法:按行优先存储顺序，将稀疏矩阵转换为三元组的表示形式。

6．稀疏矩阵用三元组的表示形式，试写一算法实现两个稀疏矩阵相加，结果仍用三元组表示。

7．假设一个算术表达式中可以包含三中括号：圆括号“（”和“）”，方括号“[”和“]”以及花括号与“{”和“}”，且这三种括号可按任意的次序嵌套试用，如（.. .[.. .{.. .}.. .[.. .].. .].. .( .. .[.. .].. .)。试利用栈的运算编写判断给定表达式中所含括号是否正确 配对出现的算法（可设表达式已存入字符型数组中）。

8．已知Ackerman函数定义如下：

akm(m,n)=

试写出递归算法。

9．设函数f(m,n)按下式定义（ m,n为）0的整数）

f(m,n)=

试写出计算函数的递归过程。

10．设有一个背包可以放入的物品重量为S，现有n件物品，重量分别为w1,w2,.. .，wn.问能否从这n件物品中选择若干件放入此背包，使得放入的重量之和正好为S.如果存在一种符合上述要求的选择，则称此背包问题有解（或承接为真），否则此问题无解（或结为假），试用递归和非递归两种方法设计此背包问题的算法。

11．借助栈(可用栈的基本运算)来实现单链表的逆置运算。

第六章 树

一．名词解释：

1 树 2。结点的度 3。叶子 4。分支点 5。树的度

6．父结点、子结点 7兄弟 8结点的层数 9树的高度 10 二叉树

11 空二叉树 12 左孩子、右孩子 13孩子数 14 满二叉树 15完全二叉树

16 先根遍历 17 中根遍历 18后根遍历 19二叉树的遍历 20 判定树

21 哈夫曼树

二、填空题

1. 树（及一切树形结构）是一种“\_\_\_\_\_\_\_\_”结构。在树上，\_\_\_\_\_\_\_\_结点没有直接前趋。对树上任一结点X来说，X是它的任一子树的根结点惟一的\_\_\_\_\_\_\_\_。
2. 一棵树上的任何结点（不包括根本身）称为根的\_\_\_\_\_\_\_\_。若B是A的子孙，则称A是B的\_\_\_\_\_\_\_\_
3. 一般的，二叉树有\_\_\_\_\_\_二叉树、\_\_\_\_\_\_的二叉树、只有\_\_\_\_\_\_的二叉树、只有\_\_\_\_\_\_

的二叉树、同时有\_\_\_\_\_\_的二叉树五种基本形态。

1. 二叉树第i(i>=1)层上至多有\_\_\_\_\_\_个结点。
2. 深度为k(k>=1)的二叉树至多有\_\_\_\_\_\_个结点。
3. 对任何二叉树，若度为2的节点数为n2，则叶子数n0=\_\_\_\_\_\_。
4. 满二叉树上各层的节点数已达到了二叉树可以容纳的\_\_\_\_\_\_。满二叉树也是\_\_\_\_\_\_二叉树，但反之不然。
5. 具有n个结点的完全二叉树的深度为\_\_\_\_\_\_。
6. 如果将一棵有n个结点的完全二叉树按层编号，则对任一编号为i(1<=i<=n)的结点X有：
   1. 若i=1,则结点X是\_\_\_\_\_\_；若i〉1,则X的双亲PARENT(X)的编号为\_\_\_\_\_\_。
   2. 若2i>n，则结点X无\_\_\_\_\_\_且无\_\_\_\_\_\_；否则，X的左孩子LCHILD(X)的编号为\_\_\_\_\_\_。
   3. 若2i+1>n，则结点X无\_\_\_\_\_\_；否则，X的右孩子RCHILD(X)的编号为\_\_\_\_\_\_。

10.二叉树通常有\_\_\_\_\_\_存储结构和\_\_\_\_\_\_存储结构两类存储结构。

11.每个二叉链表的访问只能从\_\_\_\_\_\_结点的指针，该指针具有标识二叉链表的作用。

12.对二叉链表的访问只能从\_\_\_\_\_\_指针开始，若二叉树为空，则\_\_\_\_\_\_=NULL.

13.二叉链表中每个存储结点的每个指针域必须有一个值，这个值或者是\_\_\_\_\_\_\_\_\_\_\_\_的指针，或者是\_\_\_\_\_\_。

14.具有n个结点的二叉树中,一共有\_\_\_\_\_\_\_\_个指针域,其中只有\_\_\_\_\_\_\_\_个用来指向结点的左右孩子，其余的\_\_\_\_\_\_\_\_个指针域为NULL。

15．二叉树有不同的链式存储结构，其中最常用的是\_\_\_\_\_\_\_\_与\_\_\_\_\_\_\_\_。

16．可通过在非完全二叉树的“残缺”位置上增设“\_\_\_\_\_\_\_\_”将其转化为完全二叉树。

17．以下程序段采用先根遍历方法求二叉树的叶子数，请在横线处填充适当的语句。

Void countleaf(bitreptr t,int \***count)/**\*根指针为t，假定叶子数count的初值为0\*/

{if(t!=NULL)

{if((t->lchild==NULL)&&(t->rchild==NULL))\_\_\_\_\_\_\_\_;

countleaf(t->lchild,&count);

\_\_\_\_\_\_\_\_

}

}

18．一棵二叉树由根、左子树和右子树三部分组成，因此对二叉树的遍历也可相应地分解成\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_三项“子任务”。

19．若以D、L、R分别表示二叉树的三项子任务，限定“先左后右”，这样可能的次序有：\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_三种，按这三种次序进行的遍历分别称为\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_。

20．树的主要遍历方法有\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_等三种。

21．判定树的每个\_\_\_\_\_\_\_\_包含一个条件，对应于一次比较或判断，每个\_\_\_\_\_\_\_\_对应一种分类结果。

22．设定T是一判定树，其终端结点为n1,……，nk。每个终端结点ni对应的百分为pi（通常将pi称为n1的权）。再假定ni的祖先数为li，这样，按T进行分类的平均比较次数为\_\_\_\_\_\_\_\_。

23．根据文字说明，请在以下横线处填充适当的语句。

采用静态链表作存储结构，设置一个大小为2n-1的数组,令数组每个元素由四个域组成：wt是结点的权值；lchild、rchild分别为结点的左、右孩子指针；parent是结点的双亲在数组中的下标。其数组元素类型定义如下：

typedef struct

{float wt /\*权值\*/

int parent,lchild rchild; /\*指针域\*/

}node;

typedef node hftree[2\*n-1];

在这种存储结构上的哈夫曼算法可描述如下：

void Huffman(int k,float W[k],hftree T) /\*求给定权值W的哈夫曼树T\*/

{int i,j,x,y;

float m,n;

for(i=0;i<2\*k-1;i++)

{ T[i].parent=-1;T[i].lchild=-1;T[i].rchild=-1;

if(\_\_\_\_\_\_\_\_)T[i].wt=W[i];

else T[i].wt=0

}

for(i=0;i<k-1;i++)

{ x=0;y=0;m=maxint;n=maxint;

for(j=0;j<k=i;j++)

if((T[j].wt<m)&&(T[j].parent==-1)){n=m;y=\_\_\_\_\_\_\_\_;m=\_\_\_\_\_\_\_\_;x=j;}

else if((T[j].wt<n)&&(T[j].parint==-1)){n=T[j].wt;y=j;}

T[x].parent=\_\_\_\_\_\_\_\_;T[y].parent=\_\_\_\_\_\_\_\_;

T[k+i].wt=\_\_\_\_\_\_\_\_;

T[k+i].lchild=\_\_\_\_\_\_\_\_;T[k+i].rchild=\_\_\_\_\_\_\_\_;

}

24．若二叉树的一个叶子是某子树的中根遍历序列中的第一个结点，则它必是该子树的后根遍历序列中的\_\_\_\_\_\_\_\_个结点。

25．在\_\_\_\_\_\_\_\_遍历二叉树的序列中，任何结点的子树上的所有结点，都是直接跟在该结点之后。

26．具有n个结点的完全二叉树，若按层次从上到下、从左到右对其编，号（根结点为1号），则编号最大的分支结点序号是\_\_\_\_\_\_\_\_，编号最小的分支结点序号是\_\_\_\_\_\_\_\_，编号最大的叶子结点序号是\_\_\_\_\_\_\_\_，编号最小的叶子结点序号是\_\_\_\_\_\_\_\_。

27．二叉树的先根遍历序列中，除根结点外，任一结点均处在其双亲结点的\_\_\_\_\_\_\_\_。

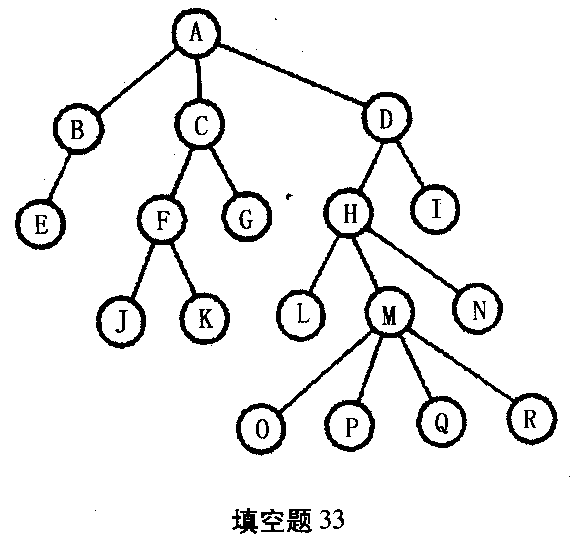
28．先根遍历树和先根遍历与该树对应的二叉树，其结果\_\_\_\_\_\_\_\_。

29．树与二叉树之间最主要的差别是：二叉树中各结点的子树要区分为\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_，即使在结点只有一棵子树的情况下，也要明确指出该子树是\_\_\_\_\_\_\_\_还是\_\_\_\_\_\_\_\_。

30．由\_\_\_\_\_\_\_\_转换成二叉树时，其根结点的右子树总是空的。

31．哈夫曼树是带权路径度\_\_\_\_\_\_\_\_的树，通常权值较大的结点离根\_\_\_\_\_\_\_\_。

32．有m个叶子结点的哈夫曼树，其结点总数为\_\_\_\_\_\_\_\_。



33．一棵树的形状如图填空题33所示，它的根结点是\_\_\_\_\_\_\_\_，叶子结点是\_\_\_\_\_\_\_\_，结点H的度是\_\_\_\_\_\_\_\_，这棵树的度是\_\_\_\_\_\_\_\_，这棵树的深度是\_\_\_\_\_\_\_\_，结点F的儿子结点是\_\_\_\_\_\_\_\_，结点G的父结点是\_\_\_\_\_\_\_\_。

34．树的结点数目至少为\_\_\_\_\_\_\_\_，二叉树的结点数目至少为\_\_\_\_\_\_\_\_。

35．\_\_\_\_\_\_\_\_中结点的最大度数允许大于2，而\_\_\_\_\_\_\_\_中结点的最大度数不允许大于2。

36．结点最少的树为\_\_\_\_\_\_\_\_，结点最少的二叉树为\_\_\_\_\_\_\_\_。

37．若一棵二叉树的叶子数为n，则该二叉树中，左、右子树皆非空的结点个数为\_\_\_\_\_\_\_\_。

38．任意一棵具有n个结点的二叉树，若它有m个叶子，则该二叉树上度数为1的结点为\_\_\_\_\_\_\_\_个。

39．现有按中序遍历二叉树的结果为ABC，问有\_\_\_\_\_\_\_\_种不同形态的二叉树可以得到这一遍历结果，这些二叉树分别是\_\_\_\_\_\_\_\_。

40．以数据集{4，5，6，7，10，12，18}为叶结点权值所构造的哈无曼树为\_\_\_\_\_\_\_\_，其带权路径长度为\_\_\_\_\_\_\_\_。

41．有m个叶子结点的哈夫曼树上的结点数是\_\_\_\_\_\_\_\_。

42．已知一棵度为3的树有2个度为1的结点，3个度过为2的结点，4个度为3的结点，则该树中有\_\_\_\_\_\_\_\_个叶子结点。

43．设树T的度为4，其中度为1、2、3和4的结点个数分别是4、2、1和1，则T中叶子结点的个数是\_\_\_\_\_\_\_\_。

三、单向选择题

1. 以下说法错误的是 ( )

①树形结构的特点是一个结点可以有多个直接前趋

②线性结构中的一个结点至多只有一个直接后继

③树形结构可以表达(组织)更复杂的数据

④树(及一切树形结构)是一种"分支层次"结构

⑤任何只含一个结点的集合是一棵树

2，以下说法错误的是 ( )

①二叉树可以是空集

②二叉树的任一结点都有两棵子树

③二叉树与树具有相同的树形结构

④二叉树中任一结点的两棵子树有次序之分

3、以下说法错误的是 ( )

①完全二叉树上结点之间的父子关系可由它们编号之间的关系来表达

②在三叉链表上，二叉树的求双亲运算很容易实现

③在二叉链表上，求根，求左、右孩子等很容易实现

④在二叉链表上，求双亲运算的时间性能很好

4、以下说法错误的是 ( )

①一般在哈夫曼树中，权值越大的叶子离根结点越近

②哈夫曼树中没有度数为1的分支结点

③若初始森林中共有n裸二叉树，最终求得的哈夫曼树共有2n-1个结点

④若初始森林中共有n裸二叉树，进行2n-1次合并后才能剩下一棵最终的哈夫曼树

5．深度为6的二叉树最多有( )个结点 ( )

①64 ②63 ③32 ④31

6．将含有83个结点的完全二叉树从根结点开始编号，根为1号，后面按从上到下、从左到右的顺序对结点编号，那么编号为41的双结点编号为 ( )

①42 ②40 ③21 ④20

7．任何一棵二叉树的叶结点在其先根、中根、后跟遍历序列中的相对位置 ( )

①肯定发生变化 ②有时发生变化

③肯定不发生变化 ④无法确定

8．设二叉树有n个结点，则其深度为 ( )

①n-1 ②n ③5floor(log2n) ④无法确定

9．设深度为k的二叉树上只有度为0和度为2的节点，则这类二叉树上所含结点总数最少（ ）个

①k+1 ②2k ③2k-1 ④2k+1

10.下列说法正确的是 ( )

①树的先根遍历序列与其对应的二叉树的先根遍历序列相同

②树的先根遍历序列与其对应的二叉树的后根遍历序列相同

③树的后根遍历序列与其对应的二叉树的先根遍历序列相同

④树的后根遍历序列与其对应的二叉树的后根遍历序列相同

11．下列说法中正确的是 ( )

①任何一棵二叉树中至少有一个结点的度为2

②任何一棵二叉树中每个结点的度都为2

③任何一棵二叉树中的度肯定等于2

④任何一棵二叉树中的度可以小于2

12．一棵二叉树满足下列条件：对任意结点，若存在左、右子树，则其值都小于它的左子树上所有结点的值，而大于右子树上所有结点的值。现采用（ ）遍历方式就可以得到这棵二叉树所有结点的递增序列。

①先根 ②中根 ③后根 ④层次

13．设森林T中有4棵树，第一、二、三、四棵树的结点个数分别是n1,n2,n3,n4,那么当把森林T转换成一棵二叉树后，且根结点的右子树上有（ ）个结点。

①n1-1 ②n1 ③n1+n2+n3 ④n2+n3+n4

14．森林T中有4棵树，第一、二、三、四棵树的结点个数分别是n1,n2,n3,n4,那么当把森林T转换成一棵二叉树后，且根结点的左孩子上有（ ）个结点。

①n1-1 ②n1 ③n1+n2+n3 ④n2+n3+n4

15.对含有（ ）个结点的非空二叉树，采用任何一种遍历方式，其结点访问序列均相同。

①0 ②1 ③2 ④不存在这样的二叉树

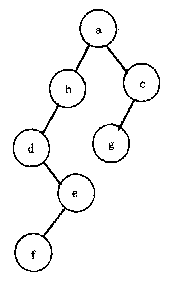
16．讨论树、森林和二叉树的关系，目的是为了（ ）

①借助二叉树上的运算方法去实现对树的一些运算

②将树、森林按二叉树的存储方式进行存储

③将树、森林转换成二叉树

④体现一种技巧，没有什么实际意义



17．如图选择题17所示二叉树的中序遍历序列是（ ）

①abcdgef ② dfebagc ③dbaefcg ④defbagc

18.已知某二叉树的后续遍历序列是dabec,中序遍历序列是deabc,它的前序遍历序列是（ ）

①acbed ②deabc ③decab ④cedba

19.如果T2是由有序树T转化而来的二叉树，那么T中结点的前序就是T2中结点的（ ）

①前序 ②中序 ③后序 ④层次序

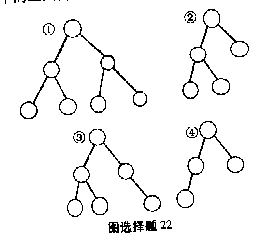
20．如果T2是由有序树T转化而来的二叉树，那么T中结点的后序就是T2中结点的（ ）

①前序 ②中序 ③后序 ④层次序

21．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 （ ）

①bdgcefha ②gdbecfha ③④ bdgechfa ④ gdbehfca

22.在图选择题22中的二叉树中，（ ）不是完全二叉树。



23、二叉树的前序遍历序列中，任意一个结点均处在其子女结点的前面，这种说法 ( )

①正确 ②错误

24、由于二叉树中每个结点的度最大为2，所以二叉树是一种特殊的树，这种说法 ( )

①五确 ②错误

25，二叉树是每个结点的度不超过2的有序树的特殊情况，这种说法 ( )

①正确 ②错误

26·树最适合用来表示 ( )

①有序数据元素 ②无序数据元素

③元素之间具有分支层次关系的数据 ④元素之间无联系的数据

27，深度为5的二叉树至多有( )个结点。

①16 ②32 ③31 ④10

28、在计算递归函数时，如不使用递归过程，则一般情况下必须借助于（ ）数据结构

①栈 ②树 ③双向队列 ④顺序表

29.堆(Heap)是 ( )

①完全二叉树 ②线性表 ③满二叉树

30、下列说法中正确的是 ( )

①二叉树中任何一个结点的度都为2 ②二叉树的度为2

③任何一棵二叉树中至少有一个结点的度为2 ④一棵二叉树的度可以小于2

31、设二叉树根结点的层次为0，一棵高度为h的满二叉树中的结点个数是（ ）

①2h ②2h-1 ③2h-1 ④2h+1-1

32、设二叉树结点的先根序列、中根序列和后根序列中，所有叶子结点的先后顺序( )

①都不相同 ②完全相同

③先序和中序相同，而与后序不同 ④中序和后序相同，而与先序不同

33·以下说法错误的是 ( )

①存在这样的二叉树，对它采用任何次序的遍历，其结点访问序列均相同

②二叉树是树的特殊情形

③由树转换成二叉树，其根结点的右子树总是空的

④在二叉树只有一棵子树的情况下也要明确指出该子树是左子树还是右子树

34、以下说法正确的是 ( )

①先根遍历树和前序遍历与该树对应的二叉树，其结果不同

②后根遍历树和前序遍历与该树对应的二叉树，其结果不同

③前序遍历森林和前序遍历与该森林对应的二叉树，其结果相同

④后序遍历森林和中序遍历与该森林对应的二叉树，其结果不同

35·以下说法正确的是 ( )

①一般来说，若深度为k的n个结点的二叉树具有最小路径长度，那么从根结点到第k-1层具有最多的结点数为2k-1-1余下的n-2k-1+1个结点在第k层的任一位置上

②若有一个结点是某二叉树子树的前序遍历序列中的最后一个结点，则它必是该子树的前序遍历序列中的最后一个结点。

③若一个结点是某二叉树子树的前序遍历序列中的最后一个结点，则它必是该子树的中序遍历序列中的最后一个结点。

④在二叉树中插人结点，该二叉树便不再是二叉树

36、以下说法正确的是 ( )

①若一个树叶是某二叉树子树的前序遍历序列中的最后一个结点，则它必是该子树的前序遍历序列中的最后一个结点。

②若一个树叶是某二叉树子树的前序遍历序列中的最后一个结点，则它必是该子树的中序离历序列中的最后一个结点

③二叉树中，具有两个子女的父结点，在中序遍历序列中，它的后继结点最多只能有一个子女结点。

④在二叉树中，具有一个子女结点，在中序遍历序列中，它没有后继子女结点。

37、以下说法错误的是 ( )

①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。

②若一个二叉树的树叶是某子树的中序遍历序列中的第一个结点，则它必是该子树的后序遍历序列中的第一个结点。

③已知二叉树的前序遍历和后序遍历序列并不能惟一地确定这棵树，因为不知道树的根结点是哪一个。

④在前序遍历二叉树的序列中，任何结点的子树的所有结点都是直接跟在该结点的之后。

四、简答及应用题

1．简述二叉链表的类型定义。

2．简述三叉链表的类型定义。

3．简述孩子链表表示法的类型定义。

4．分别就图简答题4.1中的二叉树和树回答下列问题

(1)哪个是根结点?

(2)哪些是叶结点?

(3)哪个是G的双亲?

(4)哪些是G的祖先?

(5)哪些是G的孩子?

(6)哪些是E的子孙?

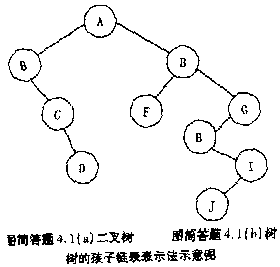
(7)哪些是E的兄弟? 哪些是C的兄弟?

(8)结点B和I的层数分别是多少?

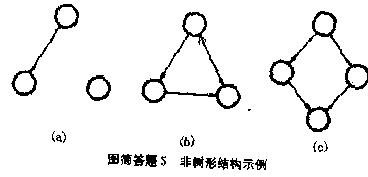
(9)树的深度是多少?

(10)以结点G为根的子树的深度是多少?

(11)树的度数是多少？

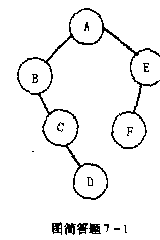


5．为什么图简答题5所示的结构都不是树形结构？详细说明理由。



6．分别画出含3个结点的树与二叉树的所有不同形态。

7．分别画出图简答题7-1所示二叉树的二叉链表、三叉链表和顺序存储结构。



8．分别写出图简答题4.1（a）所示二叉树的先根、中根和后根序列。

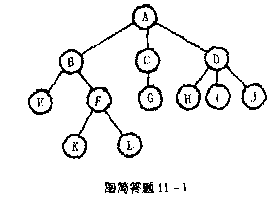
9.试找出分别满足下列条件的所有二叉树：

1. 先根序列和中根序列相同； （2）中根序列和后根序列相同；

（ 3 ）先根序列和后根序列相同。

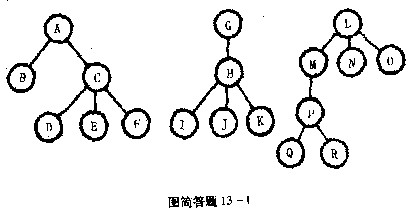
10．已知一棵二叉树的中根序列和后根序列分别为BDCEAFHG和EDCBHGEA，试画出这棵二叉树。

11． 试分别画出图简答题11-1所示树的孩子链表、孩子兄弟链表和静态双亲链表。

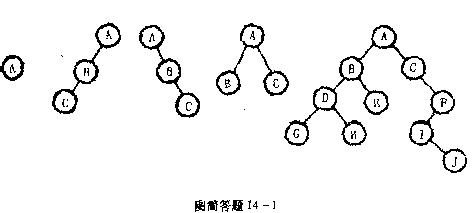


12．分别给出简答题11.1图中树的先根、后根和层次遍历的结点访问序列。

13．将图简答题13-1所示的林转换成二叉树。



14．分别画出图简答题14-1所示各二叉树对应的林。



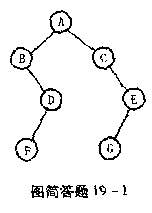
15．给定权值7，18，3，32，5，26，12，8，构造相应的哈夫曼树。

16．试以三种遍历为基础，分别写出在二叉树上查找直接前趋或直接后继的关键操作步骤。

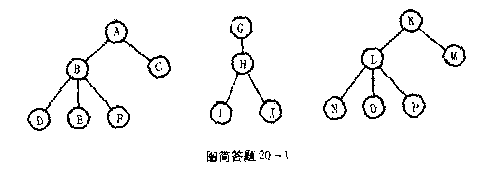
17．已知一棵二叉树的前根序列和中根序列分别为ABDGHECFIJ及GDHBEACIJF，请画出这棵二叉树。

18．设某密码电文由8个字母组成，每个字母在电文中的出现频率分别是7，19，2，6，32，3，21，10，试为这8个字母设计相应的哈夫曼编码。

19．有一二叉树如图19-1所示，试画出它的顺序存储结构示意图。



20．将图简答20-1所示森林转换为二叉树。



五，算法设计

1、以二叉链表为存储结构，分别实现二叉树的下列运算:

（1）PARENT (BT,X)；

（2）CREATE ( X, LBT.RBT)；

(3)DELLEFT(BT,X)。

2.以三叉链表作存储结构重做上题。

3．以二叉链表作存储结构，试编写求二叉树深度的算法。

4．一棵n个结点的完全二叉树存放在二叉树的顺序存储结构中，试编写非递归算法对该树进行先根遍历。

5．试编写算法判断两棵二叉树是否等价。若二叉树T1和T2等价，则T1和T2都是空的二叉树；或T1和T2的根结点的值相同，并且T1的左子树与T2的左子树是等价的，T1的右子树与T2的右子树是等价的。

6．试编写算法交换二叉树中所有结点的左、右子树（自选存储结构）。

7．试编写算法查找二叉链表中数据域值为X的结点（假定各结点的数据域值各不相同），并打印出X所有祖先的数据域值（提示：利用后根遍历非递归算法）。

8．试以孩子链表为存储结构，实现树数据结构的下列运算：

（1）PARENT（T，X）； （2） CHILD (T,X,i)； （3）DELETE(T,X，i)。

9．试分别以孩子兄弟链表和静态双亲链表为存储结构，重做上题。

10．试分别编写二叉树中根遍历在下列存储结构上的非递归算法：

1. 二叉链表；
2. 三叉链表（提示：考虑是否需要引入工作栈）；

11．试分别编写二叉树后根遍历在下列存储结构上的非递归算法：

1. 二叉链表（提示：可在指针进栈的同时将一个标志进栈）；
2. 在存储结点中增设了标志域的mark:0..2的三叉链表（要求不用工作栈）；

12．试编写一个将百分制转换成五分制的算法，要求其时间性能尽可能地高（即平均比较次数尽可能少）。假定学生成绩的分布情况如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 分数 | 0-59 | 60-69 | 70-79 | 80-89 | 90-100 |
| 比例 | 0．05 | 0．15 | 0．40 | 0．30 | 0．10 |

13．有一二叉链表，按后根遍历时输出的结点顺序为a1, a2,…a an。试编写一算法，要求输出后根序列的逆序an,an-1 …，a2 , a1 。

14．有一二叉链表，试编写按层次顺序遍历二叉树的算法。

15．以二叉链表作为存储结构，试编写求二叉树中叶子数的算法。

第七章 图

一、名词解释

1.图 2.无向完全图 3.有向完全图 4.子图 5.连通分量 6.图的遍历

7.图的最小生成树 8.拓扑排序

1. 填空题

1.设x,y是图G中的两顶点，则（x,y）与（y,x）被认为\_\_\_\_\_\_\_\_边，但<x,y>与<y,x>是\_\_\_\_\_\_\_\_的两条弧。

2.若顶点的偶对是有序的，此图为\_\_\_\_\_\_\_\_图，有序偶对用\_\_\_\_\_\_\_\_括号括起来；若顶点偶对是无序的，此图为\_\_\_\_\_\_\_\_图，无序偶对用\_\_\_\_\_\_\_\_括号括起来。

3.设x,y∈V，若<x,y>∈E,则<x,y>表示有向图G中从x到y的一条\_\_\_\_\_\_\_\_，x称为\_\_\_\_\_\_\_\_点，y称为\_\_\_\_\_\_\_\_点。若（x,y）∈E，则在无向图G中x和y间有一条\_\_\_\_\_\_\_\_。

4.在无向图中，若顶点x与y间有边（x,y），则x与y互称\_\_\_\_\_\_\_\_，边（x,y）称为与顶点x和y\_\_\_\_\_\_\_\_。

5.一个具有n个顶点的完全无向图的边数为\_\_\_\_\_\_\_\_。一个具有n个顶点的完全有向图的弧度数为\_\_\_\_\_\_\_\_。

6.图的边或弧附带的数值叫\_\_\_\_\_\_\_\_。每条边或弧都带权的图称为\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_。

7.无向图中的顶点V的度是\_\_\_\_\_\_\_\_，记为\_\_\_\_\_\_\_\_。若G是一个有向图，则把以顶点V为终点的弧的数目称为V的\_\_\_\_\_\_\_\_，记为\_\_\_\_\_\_\_\_；把以顶点V为始点的弧的数目称为V的\_\_\_\_\_\_\_\_，称为\_\_\_\_\_\_\_\_。有向图中顶点V的度定义为D（V）=\_\_\_\_\_\_\_\_。

8.路径长度定义为\_\_\_\_\_\_\_\_。第一个顶点和最后一个顶点相同的路径称为\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_。序列中顶点不重复出现的路径称为\_\_\_\_\_\_\_\_。除了第一个顶点和最后一个顶点外，其余顶点不重复的回路，称为\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_。

9.在无向图中，如果从顶点v到顶点v’有路径，则称v和v’是\_\_\_\_\_\_\_的。如果对于图中的任意两个顶点vi,vj∈V，且vi和vj都是连通的，则称G为\_\_\_\_\_\_\_\_。

10.连通分量是无向图中的\_\_\_\_\_\_\_\_连通子图。

11.一个连通图的生成树是含有该连通图的全部顶点的一个\_\_\_\_\_\_\_\_。

12.若连通图G的顶点个数为n，则G的生成树的边数为\_\_\_\_\_\_\_\_。如果G的一个子图G’的边数\_\_\_\_\_\_\_\_，则G’中一定有环。相反，如果G’的边数\_\_\_\_\_\_\_\_，则G’一定不连通。

13.无向图的邻接矩阵是一个\_\_\_\_\_\_\_\_矩阵，有向图的邻接矩阵不一定是\_\_\_\_\_\_\_\_矩阵。

14.对于无向图的邻接矩阵，顶点vi的度是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。对于有向图的邻接矩阵，顶点vi的出度OD（vi）为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_，顶点vi的入度ID(vi)是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

15.图的存储结构主要有\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_两种。

16.邻接表表示法是借助\_\_\_\_\_\_\_\_来反映顶点间的邻接关系，所以称这个单链表为邻接表。

17.对无向图，若它有n顶点e条边，则其邻接表中需要\_\_\_\_\_\_\_\_个结点。其中，\_\_\_\_\_\_\_\_个结点构成邻接表，\_\_\_\_\_\_\_\_个结点构成顶点表。

18.对有向图，若它有n顶点e条边，则其邻接表中需要\_\_\_\_\_\_\_\_个结点。其中，\_\_\_\_\_\_\_\_个结点构成邻接表，\_\_\_\_\_\_\_\_个结点构成顶点表。

19.在邻接表上，无向图中顶点vi的度恰为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。对有向图，顶点vi的出度是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。为了求入度，必须遍历整个邻接表，在所有单链表中，其邻接点域的值为\_\_\_\_\_\_\_\_的结点的个数是顶点vi的入度。

20.遍历图的基本方法有\_\_\_\_\_\_\_\_优先搜索和\_\_\_\_\_\_\_\_优先搜索两种。

21.以下是图的深度优先算法，请在\_\_\_\_\_\_\_\_处填充适当的语句。

Dfs(GraphTp g,int v)

{ ArcNodeTp \*p;

printf(“%d”,v);

visited[v]=1;

p=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

while(p!=NULL)

{if(!\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) Dfs(g,p->adjvex);

p=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

}

22.以下是图的广度优先搜索算法，请在\_\_\_\_\_\_\_\_处填充适当的语句。

Bfs(GraphTp g,int v)

{QueptrTp Q;

ArcNodeTp \*p;

InitQueue(&Q);

printf(“%d”,v);

visited[v]=1;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

while(!EmptyQueue(Q))

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

p=g.adjlist[v].firstarc;

while(p!=NULL)

{if(!visited[p->adjvex])

{printf(“%d”,p->>adjvex);

visited[[->adjvex]=1;

EnQueue(&Q,p->adjvex);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

}

}

23.深度优先搜索遍历类似于树的\_\_\_\_\_\_\_\_遍历，它所用到的数据结构是\_\_\_\_\_\_\_\_；广度优先搜索遍历类似于树的\_\_\_\_\_\_\_\_遍历，它所用到的数据结构是\_\_\_\_\_\_\_\_。

24.任何连通图的连通分量只有一个，即\_\_\_\_\_\_\_\_。

25.对具有n个顶点的图其生成树有且仅有\_\_\_\_\_\_\_\_条边，即生成树是图的边数\_\_\_\_\_\_\_\_的连通图。

26.一个图的\_\_\_\_\_\_\_\_的表示法是惟一的，而\_\_\_\_\_\_\_\_表示法是不惟一的。

27.对无向图，其邻接矩阵是一个关于\_\_\_\_\_\_\_\_对称的矩阵。

28.在有向图的邻接矩阵上，由第i行可得到第\_\_\_\_\_\_\_\_个结点的出度，而由第j列可得到第\_\_\_\_\_\_\_\_个结点的入度。

29.\_\_\_\_\_\_\_\_的有向图，其全部顶点有可能排成一个拓扑序列。

30.一个有向图G中若有弧,<Vi,Vj>、<Vj,Vk>和<Vi,Vk>,则在图G的拓扑序列中，顶点Vi、Vj和Vk的相对位置为\_\_\_\_\_\_\_\_。

1. 单项选择题

1.设有无向图G=（V，E）和G’=（V’,E’）,如G’为G的生成树，则下面不正确的说法是（ ）

①G’为G的子图 ②G’为G的连通分量

③G’为G的极小连通子图且V’=V ④G’是G的无环子图

2.任何一个带权的无向连通图的最小生成树（ ）

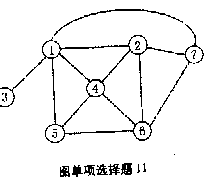
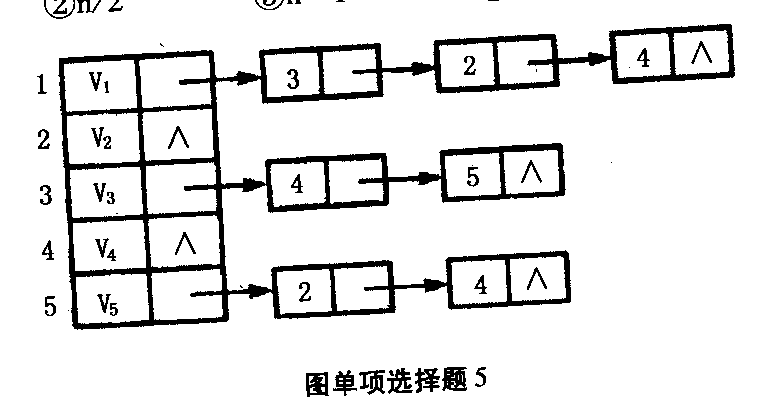
①只有一棵 ②有一棵或多棵 ③一定有多棵 ④可能不存在

3.设图G采用邻接表存储，则拓扑排序算法的时间复杂度为（ ）

①O(n) ②O(n+e) ③O(n\*n) ④O(n\*e)

4.含n个顶点的连通图中的任意一条简单路径，其长度不可能超过 （ ）

①1 ② n/2 ③ n-1 ④n



5．一有向图G的邻接表存储结构如图单项选择5所示。现按深度优先遍历算法，从顶点V1出发，所得到的顶点序列是 （ ）

①V1,V3, V2 ,V4, V5 ②V1, V3, V4, V2, V5 ③V1,V2, V3, V4, V5 ④V1, V3, V4, V5 , V2

6.在无向图中，所有顶点的度数之和是所有边数的（ ）倍。

①0.5 ②1 ③2 ④4

7.在有向图中，所有顶点的入度之和是所有顶点出度之和的（ ）倍。 （ ）

①0.5 ② 1 ③ 2 ④4

8．在图的邻接表存储结构上执行深度优先搜索遍历类似于二叉树上的 （ ）

①先根遍历 ② 中根遍历 ③ 后根遍历 ④按层次遍历

9．在图的邻接表存储结构上执行广度优先搜索遍历类似于二叉树上的 （ ）

①先根遍历 ②中根遍历 ③后根遍历 ④按层次遍历

10．判断一个有向图是否存在回路，除了可以利用拓扑排序方法，还可以利用 （ ）

①求关键路径方法②求最短路径的Dijkstra方法③广度优先遍历方法④深度优先遍历方法

11．在图单项选择中，从顶点V1出发，按广度优先遍历图的顶点序列是 （ ）

①V1 V3 V5 V4 V2 V6 V7 ②V1 V2 V4 V7 V6 V5 V3 ③V1 V5 V3 V4 V2 V7 V6 ④V1 V4 V7 V2 V6 V5 V3

12.在图单项选择中，从顶点V1出发,广度遍历图的顶点序列是 （ ）

①V1 V5 V3 V4 V2 V6 V7 ②V1 V5 V3 V4 V2 V7 V6 ③V1 V7 V2 V6 V4 V5 V3 ④V1 V2 V4 V7 V6 V5 V3

13.设有6个结点的无向图，该图至少应有（ ）条边能确保是一个连通图。 （ ）

① 5 ② 6 ③ 7 ④ 8

14.以下说法正确的是 （ ）

①连通图的生成树，是该连通图的一个极小连通子图。

②无向图的邻接矩阵是对称的，有向图的邻接矩阵一定是不对称的。

③任何一个有向图，其全部顶点可以排成一个拓扑序列。

④有回路的图不能进行拓扑排序。

15以下说法错误的是 （ ）

①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。

②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和无向图的存储都适用。

③存储无向图的相邻矩阵是对称的，因此只要存储相邻矩阵的下（或上）三角部分就可以了

③用相邻矩阵A表示图，判定任意两个结点Vi和Vj之间是否有长度为m的路径相连，则只要检查A的第 i行第j列的元素是否为0即可。

16．以下说法正确的是 （ ）

①连通分量是无向图中的极小连通子图。

②强连通分量是有向图中的极大强连通子图。

③在一个有向图的拓扑序列中，若顶点a在顶点b之前，则图中必有一条弧<a,b>。

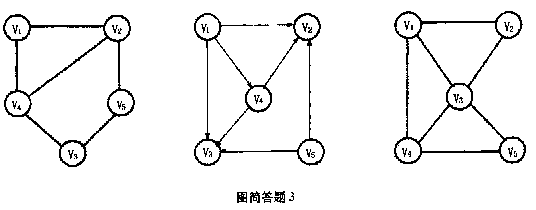
④对有向图G,如果从任意顶点出发进行一次深度优先或广度优先搜索能访问到每个顶点，则该图一定是完全图。

四、简答及应用

1． 简述图的邻接矩阵表示的类型定义

2． 简述图的邻接表的类型定义。

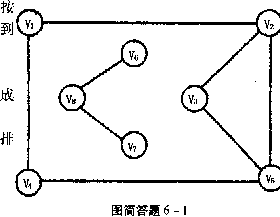
3． 给出无向图简答题3中g1的邻接矩阵和邻接表。



4．分别给出图简答题3中g2的邻接矩阵、邻接表和逆邻接表。

5．分别给出图简答题3中g3从v5出发按深度优先搜索和广度优先搜索算法遍历得到的顶点序列。

6．求出图简答题6-1的连通分量。

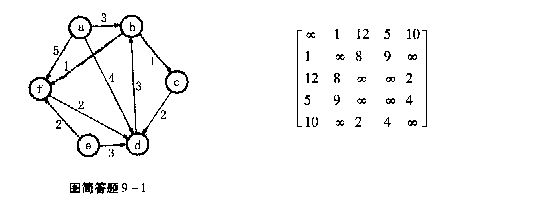


7．求出带权图简答题7-1的最小生成树

8．写出有向图简答题8-1的拓扑排序序列。

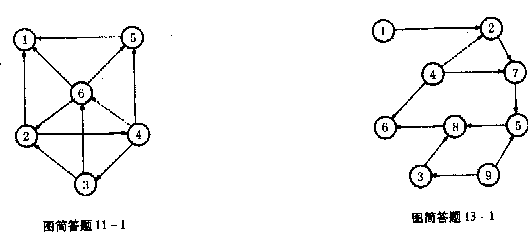
9．给出网图简答题9-1的邻接矩阵表示。

10．已知连通网的邻接矩阵如下，试画出它所表示的连通网及该连通网的最小生成树。



11．对于图简答题11-1从其邻接表中回答下列问题：

1. 图中有多少条弧？
2. 图中是否存在从i到j的弧？
3. 如何求顶点i的出度？



12．图简答题11-1所示为一有向图，请给出该图的下述要求：

1. 每个顶点的入/出度。
2. 邻接矩阵。
3. 邻接表。
4. 逆邻接表。

13．拓扑排序的结果不是唯一的，对图简答题13-1进行拓扑排序，试写出其中任意5个不同的拓扑排序列。

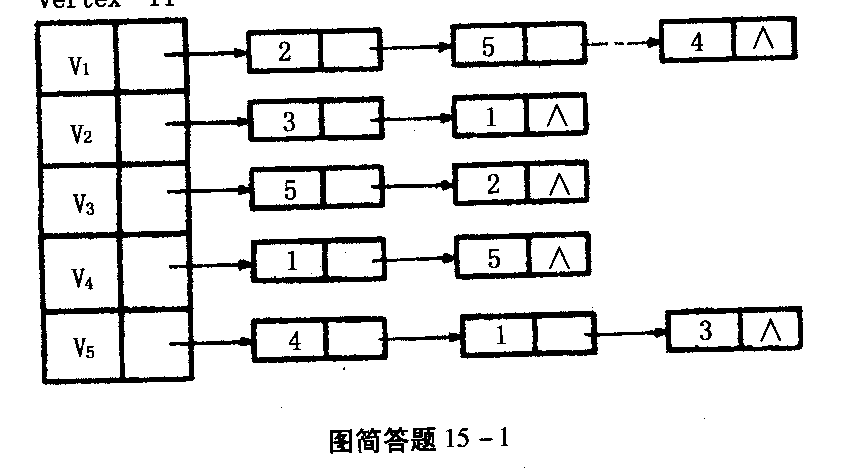
14．对m个顶点的无向图G，采用邻接矩阵，如何判别下列有关问题：

1. 图中有多少条边？
2. 任意两个顶点i和j是否有边相连？
3. 任意一个顶点的度是多少？

15.已知图G的邻接表如图简答题15-1所试，顶点V1为出发点，完成要求：

1. 深度优先搜索的顶点序列；
2. 广度优先搜索的顶点序列；
3. 由深度优先搜索得到的一棵生成树；
4. 由广度优先搜索得到的一棵生成树。

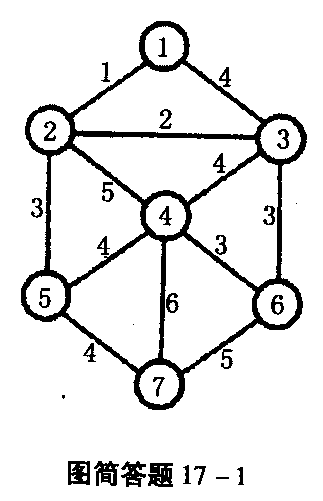
Vertex fi



16.设有一无向图G=(V,E),其中V={1,2,3,4,5,6},E={(1,2),(1,6),(2,6),(1,4),(6,4),(1,3),(3,4),(6,5),(4,5),(1,5),(3,5)}。

1. 按上述顺序输入后，画出其相应的邻接表；
2. 在该邻接表上，从顶点4开始，写出DFS序列和BFS序列。

17．图简答题17-1所示为一无向连通网络，先要求根据prim算法构造出它的最小生成树。



五、算法设计

* 1. 写出将一个无向图的邻接矩阵转换成邻接表的算法。
  2. 写出将一个无向图的邻接表转换成邻接矩阵的算法。
  3. 试以邻接矩阵为存储结构，分别写出连通图的深度优先和广度优先搜索算法。
  4. 写出建立一个有向图的逆邻接表的算法。
  5. G为一n个顶点的有向图，其存储结构分别为：

1. 邻接矩阵；
2. 邻接表。

请写出相应存储结构上的计算有向图G出度为0的顶点个数的算法。

6．以邻接表作存储结构，给出拓扑排序算法的实现。

第九章 查找表

一、名词解释

1.查找表2.集合3.查找长度4.有序表5.平衡化6.平衡二叉排序树7.平衡因子8.散列表

9.散列函数10.散列地址11.同义词12.冲突13.开散列表14.拉链法15.堆积

二、填空题

1.对任何集合A而言，A的每一个成员a称为A的一个\_\_\_\_\_\_\_\_，记为\_\_\_\_\_\_\_\_。

2.空集是\_\_\_\_\_\_\_\_的集合，记为\_\_\_\_\_\_\_\_。

3.在集合这种逻辑结构中，任何结点之间都不存在\_\_\_\_\_\_\_\_关系，这是集合这种逻辑结构的基本特点。

4.集合中没有相同的\_\_\_\_\_\_\_\_。作为一个数学概念，集合的元素没有\_\_\_\_\_\_\_\_。

5.数据元素之间的逻辑关系反映的是\_\_\_\_\_\_\_\_。

6.对于集合这逻辑结构来说，其中的数据元素之间也可以有各种各样的非逻辑关系，但任何一对数据元素之间没有\_\_\_\_\_\_\_\_关系，即没有\_\_\_\_\_\_\_\_关系。

7.查找表按其所包括的运算的不同分为\_\_\_\_\_\_\_\_查找表和\_\_\_\_\_\_\_\_查找表。

8.查找表中主关键字指的是\_\_\_\_\_\_\_\_，次关键字指的是\_\_\_\_\_\_\_\_。

9.静态查找表包括\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_三种基本运算。

10.动态查找表包括\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_五种基本运算。

11.假定key为主关键字，若顺序表中第n个元素的键值为K，则顺序查找算法的查找长度为1；若第1个元素的键值为K，则查找长度为\_\_\_\_\_\_\_\_；若表中无键值等于K的元素，则查找长度为\_\_\_\_\_\_\_\_。

12.以下算法在有序表R中用二分查找法查找键值等于K的元素，请分析程序，并在\_\_\_\_\_\_\_\_上填充合适的语句。

int binsearch(sqtable R,keytype K)

{low=1;hig=R.n;/\*置查找区间初值。low,hig分别标记查找区间的下、上界\*/

while(low<=hig)

{mid=（low+hig)/2;

switch

{case K==R.item[mid].key:return(mid);/\*找到，返回位置mid\*/

case K<R.item[mid].key:\_\_\_\_\_\_\_\_;break;/\*缩小区间\*/

case K>R.item[mid].kiy:\_\_\_\_\_\_\_\_;break;/\*缩小区间\*/

}

}

return(0);/\*若区间长度已为0但仍不成功，则返回0，表示查找不成功\*/

}

13.二分查找在查找成功时的查找长度不超过\_\_\_\_\_\_\_\_，其平均查找长度为\_\_\_\_\_\_\_\_，当n较大时约等于\_\_\_\_\_\_\_\_。

14.一个\_\_\_\_\_\_\_\_表由一个顺序表和一个索引表两部分组成。其中的顺序表在组织形式与普通的\_\_\_\_\_\_\_\_表完全相同；而索引表本身在组织形式上也是一个\_\_\_\_\_\_\_\_表。

15.索引顺序表的特点表现为以下两个方面：a.顺序表中的数据元素"\_\_\_\_\_\_\_\_"；b.索引表反映了这些"\_\_\_\_\_\_\_\_"的有关特性。

16.在索引顺序表上，对于顺序表中的每一块，索引表中有相应的一个"索引项"。每个索引项有两个域：块内最大\_\_\_\_\_\_\_\_值和块\_\_\_\_\_\_\_\_位置。

17.索引顺序表上的查找分两个阶段：一、\_\_\_\_\_\_\_\_；二、\_\_\_\_\_\_\_\_。该查找方法称为\_\_\_\_\_\_\_\_查找。

18.静态查找表的三种不同实现各有优缺点。其中，\_\_\_\_\_\_\_\_查找效率最低但限制最少。\_\_\_\_\_\_\_\_查找效率最高但限制最强。而\_\_\_\_\_\_\_\_查找则介于上述二者之间。

19.二叉排序树是一种特殊的、增加了限制条件的二叉树，其限制条件是任一结点的键值\_\_\_\_\_\_\_\_于其左孩子（及其子孙）的键值且\_\_\_\_\_\_\_\_于其右孩子（及其子孙）的键值。

20.在表示一棵二叉排序树的二叉链表上，要找键值比某结点X的键值\_\_\_\_\_\_\_\_的结点，只需通过结点X的左指针到它的左子树中去找。

21.中根遍历一棵二叉排序树所得的结点访问序列是键值的\_\_\_\_\_\_\_\_序列。

22.对于一个无序序列，可以通过构造一棵\_\_\_\_\_\_\_\_而使其成为一个有序序列。

23.以下算法在指针T所指的二叉排序树上查找键值等于K的结点。成功时回送指向该结点的指针；否则回送空指针。请分析程序，并在\_\_\_\_\_\_\_\_填充合适的语句。

bitreptr search\_bst(bitreptr T,keytype K)

{if(T==NULL)return(NULL);

else swith

{case T->key==K:\_\_\_\_\_\_\_\_;

case\_\_\_\_\_\_\_\_: rerutn(search\_bst(T->lchild,K));

case\_\_\_\_\_\_\_\_: rerutn(search\_bst(T->rchild,K));

}

}

24.二叉排序树上的查找长度不仅与\_\_\_\_\_\_\_\_有关，也与二叉排序树的\_\_\_\_\_\_\_\_有关。

25.在随机情况下，含有n个结点的二叉排序树的平均查找长度为\_\_\_\_\_\_\_\_，其时间效率很高。

26.二叉排序的查找效率与树的形态有关。当二叉排序树退化为一条单支时，查找算法退化为\_\_\_\_\_\_\_\_查找，平均查找长度上升为\_\_\_\_\_\_\_\_。

27.有n个结点的AVL树的深度与\_\_\_\_\_\_\_\_是同数量级的，因而在它上面进行查找的平均查找长度也与\_\_\_\_\_\_\_\_同数量级。

28.平衡二叉排序树上任一结点的平衡因子只可能是\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_。

29.采用散列技术时需要考虑的两个主要问题是：一、\_\_\_\_\_\_\_\_？二、\_\_\_\_\_\_\_\_？

30.按组织形式的不同，通常有\_\_\_\_\_\_\_\_与\_\_\_\_\_\_\_\_两类散列表。

31.以下算法在开散列表HP中查找键值等于K的结点，成功时返回指向该结点的指针，不成功时返回空指针。请分析程序，并在\_\_\_\_\_\_\_\_上填充合适的语句。

pointer research\_openhash(keytype K,openhash HP)

{i=H(K); /\*计算K的散列地址\*/

p=HP[i]; /\*i的同义词子表表头指针传给p\*/

while(\_\_\_\_\_\_\_\_)p=p->next;/\*未达表尾且未找到时，继续扫描\*/

\_\_\_\_\_\_\_\_;

}

32以下算法实现若开散列表HP中无键值为K结点，则插入一个这样的结点。请分析程序并在\_\_\_\_\_\_\_\_上填充合适的语句。

void insert\_openhash(keytype K,openhash HP)

{if(research\_openhash(K,HP)==NULL)

{i=H(K);

q=malloc(size);q->key=\_\_\_\_\_\_\_\_; /\*生成新结点\*/

\_\_\_\_\_\_\_\_=HP[i];HP[i]=\_\_\_\_\_\_\_\_; /\*前插法链入新结点\*/

}

}

33.以下算法实现若开散列表HP中存在键值为K结点，则将其删除。请分析程序并在\_\_\_\_\_\_\_\_上填充合适的语句。

void delete\_openhash(keytype K,openhash HP)

{i=H(K);

if(HP[i]==NULL)return;/\*空表则退出\*/

p=HP[i];

if(p->key==K){\_\_\_\_\_\_\_=p->next;free(p);return;} /\*表首结为待点时的删除\*/

while(p->next!=NULL) /\*其他情况下的删除\*/

{q=p;p=p=->next;

if(p->key==K){\_\_\_\_\_\_\_\_=p->next;delete(p);return;}

}

}

34.对闭散列表来说，\_\_\_\_\_\_\_\_的方法就是处理冲突的方法。

35.常见的构造后继散列地址序列的方法有\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_四种。

36.以下算法假定以线性探测法解决冲突，在闭散列表HL中查找键值为K的结点，成功时回送该位置；不成功时回送标志-1。请分析程序，并在\_\_\_\_\_\_\_\_上填充合适的语句。

int search\_cloxehash(keytype K,closehash HL)

{d=H(K); /\*计算散列地址\*/

i=d;

while(HL[i].key!=K&&(i!=d-1)i=\_\_\_\_\_\_;/\*未成功且未查遍整个HL时继续扫描\*/

if(\_\_\_\_\_\_\_\_)reurn(i); /\*查找成功\*/

else return(-1); /\*查找失败\*/

}

37.\_\_\_\_\_\_\_\_查找法的平均查找长度与元个数n个数无关。

38.在分块查找法中，首先查找\_\_\_\_\_\_\_\_，然后再查找找相应的\_\_\_\_\_\_\_\_。

39.在具有24个元素的有序表上进行二分查找，则比较一次查找成功的结点数为\_\_\_\_\_\_\_\_，比较二次查找成功的结点数为\_\_\_\_\_\_\_\_，比较五次查找成功的结点数为\_\_\_\_\_\_\_\_。总的平均查找长度为\_\_\_\_\_\_\_\_。

40.在散列存储中，装填因子x的值越大，则\_\_\_\_\_\_\_\_。

41.\_\_\_\_\_\_\_\_是散列表的一个重要参数，它反映出散列表的装满程度。

42.当所有结点的权值都相等时，用这些结点构造的二叉排序树上只有\_\_\_\_\_\_\_\_。

43.在二叉排序树上插入新结点时，不必移动其它结点，仅需使某结点的指针域由\_\_\_\_\_\_\_\_变为\_\_\_\_\_\_\_\_即可。

44．二分查找方法仅适用于这样的表：表中的记录必须\_\_\_\_\_\_\_\_，其存储结构必须是\_\_\_\_\_\_\_\_。

45.设线性表(a1,a2,…,a500)元素的值由小到大排列。对一个给定的k值，用二分法检索查找表中与k相等的元素，在检索不成功的情况下至多较\_\_\_\_\_\_\_\_次。

46.设线性表L=（a1,a2,…,an)(n>2),表中元素按值的递增顺序排列。对一个给定的值k，分别用顺序检索和二分法检索查找与k相等的元素，比较次数分别为s和b ，若检索不成功，则s和b的数量关系是\_\_\_\_\_\_\_\_。

47.设有两个散列函数H1(k)=k mod 13 和H2(k)=k mod 11+1，散列表为T[0…12]，用双重散解决冲突。函数H1用来计算散列地址，当发生冲突时，H2作为计算下一个探测地址的地址增量，假定在某一时刻T的状态为：

T: 0 1 2 3 4 5 6 7 8 9 10 11 12

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 80 | 85 |  |  |  |  | 34 |  |  |  |  |

下一个被插入的关键码是42，其插入的位置是:\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

48.设有一个已按各元素的值排好序的线性表，长度为125，对给定的k值，用二分法查找与k相等的元素，若查找成功，则至少需要比较\_\_\_\_\_\_\_\_次，至多需比较\_\_\_\_\_\_\_\_次。

三、单项选择题

1. 以下说法错误的是 （ ）

①数字分析法对健值的各位进行分析，选择分布较均匀的若干位组成散列地址。

②除余法选择一个适当的正整数p,以p除健值以所得的余数作为散列地址。

③平方取中法以健值平方的中间几位作为散列地址。

④基数转换法将健值看成另一种进制的数再转换成原来进制的数，然后选择其中几位作为散列地址。

⑤随机数法选择一个随机函数random,以健值在该函数下的值作为散列地址。

1. 以下所法正确的是 （ ）

①数字分析法需事先知道所有可能出现的健值及所有健值的每一位上各数字的分布情况，并且健值的位数比散列地址的位数多。

②除余法要求事先知道全部健值。

③平方取中法需要事先掌握健值的分布情况。

④随机数法适用于健值不相等的场合。

1. 除余法选择一正整数p，以健值除以p所得的余数作为散列地址。通常选p为（ ）

①小于或等于散列表长的素数。

②接近表长的且不与组成关键字的字符基数直接相关的质数。

③大于或等于散列表长的素数。

④接近表长的质数。

4．顺序查找法适合于（ ）存储结构的查找表。

①压缩 ② 散列 ③索引 ④顺序或链式

5．对采用二分查找法进行查找运算的查找表，要求按（ ）方式进行存储。

①顺序存储 ② 链式存储

③顺序存储且结点按关键字有序 ④ 链式存储且结点按关键字有序

6．设顺序表的长为n,则其每个元素的平均查找长度是 （ ）

①n ② (n-1)/2 ③ n/2 ④ (n+1)/2

7.设有序表的关键字序列为{1，4，6，10，18，35，42，53，67，71，78，84，92，99}，当用二分查找法查找健值为84的结点时，经（ ）次比较后查找成功。

①2 ② 3 ③ 4 ④ 12

8．静态查找表与动态查找表两者的根本差别在于 （ ）

①逻辑结构不同 ② 存储实现不同

③施加的操作不同 ④ 数据元素的类型不同

9．长度为12的按关键字有序的查找表采用顺序组织方式。若采用二分查找方法，则在等概率情况下，查找失败时的ASL值是 （ ）

①37/12 ② 62/13 ③ 39/12 ④49/13

10.在表长为n的顺序表中，实施顺序查找，在查找不成功时，与关键字比较的次数为（ ）

①n ②1 ③ n+1 ④n-1

11.二分查找法适用于存储结构为（ ）的，且按关键字排序的线性表。 （ ）

①顺序存储 ② 链接存储 ③ 顺序存储或链接存储 ④索引存储

12．用顺序查找法对具有n个结点的线性表查找的时间复杂性量级为

①O（n2） ② O(nlog2n) ③ O（n） ④ O(log2n)

13．用二分查找法对具有n个结点的线性表查找的时间复杂性量级为 （ ）

①O(n2) ② O(nlog2n) ③ O(n) ④O(log2n)

14．二叉排序树的查找和折半查找的时间性能相同。这种说法 （ ）

①正确 ② 错误

15．与其他查找方法相比，散列查找法的特点是 （ ）

①通过关键字比较进行查找

②通过关键字计算记录存储地址进行查找

③通过关键字计算记录存储地址，并进行一定的比较进行查找

16．在采用线性探测法处理冲突所构成的闭散列表上进行查找，可能要探测多个位置，在查找成功的情况下，所探测的这些位置上的健值 （ ）

①一定都是同义词 ② 一定都不是同义词

③都相同 ④ 健值不一定有序的顺序表

18．设散列函数为H(k)= k mod 7,一组关键码为23，14，9，6，30，12和18，散列表T的地址空间为0..6,用线性探测法解决冲突，依次将这组关键码插入T中，得到的散列表为 （ ）

①0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 14 | 6 | 23 | 9 | 18 | 30 | 12 |

② 0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 14 | 18 | 23 | 9 | 30 | 12 | 6 |

③0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 14 | 12 | 9 | 23 | 30 | 18 | 6 |

④0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 6 | 23 | 30 | 14 | 18 | 12 | 9 |

19．设有一个用线性探测法解决冲突得到的散列表：

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

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 13 | 25 | 80 | 16 | 17 | 6 | 14 |  |  |

散列函数为H(k)=k mod 11,若要查找元素14，探测的次数是 （ ）

① 18 ② 9 ③3 ④6

20．在散列函数H(k)=k MOD m 中，一般来讲，m应取 （ ）

①奇数 ②偶数 ③ 素数 ④ 充分大的数

21，分块查找的时间性能 （ ）

①低于二分查找 ② 高于顺序查找而低于二分查找

③高于顺序查找 ④ 低于顺序查找而高二分查找

22．以下说法正确的是 （ ）

①顺序查找效率最低但限制最强。

②二分查找效率最高但限制最少。

③分块查找效率介于顺序查找和二分查找之间。

23．以下说法正确的是 （ ）

①查找表中数据元素的任何数据项都可以作为关键字。

②采用二分查找法对有序表进行查找总比采用顺序查找法对其进行查找要快。

③二叉排序树的查找和二分查找时间的性能相同。

④最佳二叉排序树一定是平衡二叉树

⑤“顺序查找法”是指在顺序表上进行查找的方法。

24．．以下说法错误的是 （ ）

①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。

②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。

③装填因子是散列法的一个重要参数，它反映散列表的装填程度。

④散列表的查找效率主要取决于散列表造表时选取的散列函数和处理冲突的方法。

25．以下说法错误的是 （ ）

①当所有的结点的权之都相等时，用这些结点构造的二叉排序树的特点是只有右子树

②中序遍历二叉排序树的结点就可以得到排好序的结点序列。

③任一二叉排序树的平均检查时间都小于用顺序查找法查找同样结点的线性表的平均查找时间。

④对两棵具有相同关键字集合的而形状不同的二叉排序树，按中序遍历它们得到的序列的顺序是一样的。

⑤采用分块查找方法，既能实现线性表所希望的较快的查找速度，又能适应动态变化的需要。

26．以下说法正确的是 （ ）

①平衡树一定是丰满树。

②虽然信息项的序列的顺序不一样，但依次生成的二叉排序树确是一样的。

③在二叉排序树上插入新的结点时，不必移动其他结点，只需改动某个结点的指针，由空变为非空即可。

④在二叉排序树上删除一个结点时，不必移动其他结点，只要将该结点的父结点的相应的指针域置空即可。

27．长度为12的有序表：Apr,Aug,Dec,Feb,Jan,Jul,Jun,Mar,May,Nov,Oct,Sep,按折半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（ ）。

① 35/12 ②37/12 ③39/12 ④43/12

28．已知一采用开放地址法解决Hash表冲突，要从此Hash表中删除一个记录，正确的做法是 （ ）

①将该元素所在的存储单元清空。

②将该元素用一个特殊的元素代替

③将与该元素有相同Hash地址的后继元素顺次前移一个位置。

④用与该元素有相同Hash地址的最后插入表中的元素替代。

四、简答及应用

1. 写出作为静态查找表存储结构的顺序表的类型定义。
2. 何谓二叉排序树？
3. 简述开散列表的组织方式及类型定义。
4. 简述闭散列表的类型定义。
5. 简述闭散列表解决冲突的基本思想。
6. 简述二次探测法解决冲突的基本思想。
7. 简述多重散列法解决冲突的基本思想
8. 简述公共溢出区法解决冲突的基本思想。
9. 对长度为20的有序表进行二分查找，试画出它的一棵判定树，并求等概率情况下的平均查找长度。
10. 给定有序表D={006,087,155,188,220,465,505,508,511,586,656,670,700,766,897,908}，用二分查找法在D中查找586，试用图示法表示出查找过程。

11．给定表（19，14，22，01，66，21，83，27，56，13，10）。

（1）试按元素在表中的次序将它们依次插入一棵初始时为空的二叉排序树，画出插入完成之后的二叉排序树。

（2）按表中元素顺序构造一棵AVL树，并求其在等概率情况下查找成功的平均查找长度。

1. 给定表（Jan,Feb,Mar,Apr,May,Jun,Jun,Aug,Sep,Oct,Nov,Dec）.设取散列函数H(x)=|\_i/2\_|,其中i为健值中第一个字母在英语字母表中的序号，要求：

（1）画出相应的开散列表；

（2）画出闭散列表（以线性探测法处理）；

（3）分别求这两个散列表在等概率情况下查找成功与不成功时的平均查找长度。

1. 已知散列函数为H(K)=K mod 12,健值序列为25，37，52，43，84，99，120，15，26，11，70，82，采用拉链法处理冲突，试构造开散列表，并计算查找成功的平均查找长度。
2. 顺序查找时间为O(n),二分查找时间为O(log2n),散列查找时间为O(1),为什么有高效率的查找方法而不放弃低效率的方法？

五、算法设计

1. 假设线性表中结点是按健值递增的顺序存放的。试写一顺序查找法，将岗哨设在高下标端。然后分别求出等概率情况下查找成功和不成功时的平均查找长度。
2. 若线性表中各结点的查找概率不等，则可用如下策略提高顺序查找的效率：若找到指定的结点，则将该结点和其前驱（若存在）结点交换，使得经常被查找的结点尽量位于表的前端。试对线性表的顺序存储结构和链式存储结构写出实现上述策略的顺序查找算法（注意，查找时必须从表头开始向后扫描）。
3. 试写出二分查找的递归算法。
4. 试编写算法求出指定结点在给定的二叉排序树中所在的层数。
5. 试编写算法，在给定的二叉排序树上找出任意两个不同结点的最近公共祖先（若在两结点A、B中，A是B的祖先，则认为A、B的最近公共祖先就是A）。
6. 试写一个判别给定二叉树是否为二叉排序树的算法，设此二叉树以二叉链表作存储结构。
7. 试写出在二叉排序树上删除指定结点的算法。
8. 在以T为根指针的AVL树上插入健值K的新结点，返回值为新AVL树的根指针。

第十章 排序

一、名词解释

1.排序 2.内部排序 3.外部排序 4.堆 5.堆排序

二、填空

1.若待排序的序列中存在多个记录具有相同的键值，经过排序，这些记录的相对次序仍然保持不变，则称这种排序方法是\_\_\_\_\_\_\_\_的，否则称为\_\_\_\_\_\_\_\_的。

2.按照排序过程涉及的存储设备的不同，排序可分为\_\_\_\_\_\_\_\_排序和\_\_\_\_\_\_\_\_排序。

3.按排序过程中依据的不同原则对内部排序方法进行分类，主要有：\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_等四类。

4.在排序算法中，分析算法的时间复杂性时，通常以\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_为标准操作。评价排序的另一个主要标准是执行算法所需要的\_\_\_\_\_\_\_\_。

5.常用的插入排序方法有\_\_\_\_\_\_\_\_插入排序、\_\_\_\_\_\_\_\_插入排序、\_\_\_\_\_\_\_\_插入排序和\_\_\_\_\_\_\_\_插入排序。

6.以下为直接插入排序的算法。请分析算法，并在\_\_\_\_\_\_\_\_上填充适当的语句。

void straightsort(list r);

{for(i=\_\_\_\_\_\_\_\_\_\_\_;i<=n;i++)

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

while(r[0].key<r[j].key){r[j+1]=\_\_\_\_\_\_\_\_;j--;}

r[j+1]=\_\_\_\_\_\_\_;

}

}

7.直接插入排序是稳定的，它的时间复杂性为\_\_\_\_\_\_\_\_，空间复杂度为\_\_\_\_\_\_\_\_。

8.以下为冒泡排序的算法。请分析算法，并在\_\_\_\_\_\_\_\_上填充适当的语句。

void bulbblesort(int n,list r) /\*flag为特征位，定义为布尔型\*/

{for(i=1;i<=\_\_\_\_\_\_\_\_;i++)

{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

for(j=1;j<=\_\_\_\_\_\_\_\_\_;j++)

if(r[j+1].key<r[j].key){flag=0;p=r[j];r[j]=r[j+1];r[j+1]=p;}

if(flag) return;

}

}

9.对于n个记录的集合进行冒泡排序，其最坏情况下所需的时间复杂度是\_\_\_\_\_\_\_\_。

10.以下对r[h],r[h+1],……r[p]子序列进行一趟忆速排序。请分析算法，并在\_\_\_\_\_\_\_\_上填充适当的语句。

int quickpass(list r,int h,int p)

{i=h;j=p;x=r[i];/\*置初值，以第一个记录的键值为标准\*/

while(i<j)

{while((r[j].key>=x.key)&&(i<j))\_\_\_\_\_\_\_\_;/\*自尾端进行比较\*/

if(i<j)

{\_\_\_\_\_\_\_\_;i++;/\* 将r[j].kiy<x.key的记示移至i所指位置\*/

while((r[i].key<=x.key)&&(i<j))\_\_\_\_\_\_\_\_;/\*自首行端进行比较\*/

if(i<j){\_\_\_\_\_\_\_\_;j--;}/\* 将r[j].kiy<x.key的记示移至j所指位置\*/

}

}

r[i]=\_\_\_\_\_\_\_\_;return(i);/\*一趟快速 排序结束，将x移至正确的位置\*/

}

11.对快速排序来讲，其最好情况下的时间复杂度是\_\_\_\_\_\_\_\_，其最坏情况下的时间复杂度是\_\_\_\_\_\_\_\_。

12.以下是直接选择排序的算法。请分析算法，并在横线上填充适当的语句。

void select(list r,int n)

{for(i=1;i<=\_\_\_\_\_\_\_\_;i++)/\*每次循环，选择出一个最小键值\*/

{k=i;

for(j=i+1;j<=n;j++)if(r[j].key<r[k].key)\_\_\_\_\_\_\_\_;

if(\_\_\_\_\_\_)swap(r[k],r[i]）;/\*函数swap(r[k],r[i]）交换r[k]和r[i]的位置\*/

}

}

13.直接选择排序是不稳定的，其时间复杂性为\_\_\_\_\_\_\_\_。

14.树形选择排序要增加\_\_\_\_\_\_\_\_个结点以保存前面比较的结果。另外，排序的结果还需要另外开辟\_\_\_\_\_\_\_\_.

15.树形选择排序可用一棵树来表示这一排序过程，树中的n个叶子代表待排序记录的键值，叶子上面一层是\_\_\_\_\_\_\_\_两两比较的结果，依次类推。\_\_\_\_\_\_\_\_表示最后选择出来的最小关键字。在选择次最小键值时，只需将叶结点中最小键值改成\_\_\_\_\_\_\_\_，重新进行比较。依次类推。

16.若树形选择排序的叶子数为n，除第一次需执行\_\_\_\_\_\_\_\_次比较就选择出一个最小的键值外，以后的每次都只经过\_\_\_\_\_\_\_\_次比较就选择出一个最小的键值。所以树形选择排序总的时间开销为\_\_\_\_\_\_\_\_。

17.从一个无序序列建立一个堆的方法是：首先将要排序的所有键值分放到一棵\_\_\_\_\_\_\_\_的各个结点中，然后从i=\_\_\_\_\_\_\_\_的结点ki开始，逐步把以kn/2,kn/2-1,kn/2-2,……为根的子树排成堆，直到以k1为根的树排成堆，就完成了建堆的过程。

18.堆排序是不稳定，空间复杂度为\_\_\_\_\_\_\_\_。在最坏情况下，其时间复杂性也为\_\_\_\_\_\_\_\_。

19.以下将ah,…,am和am+1,…,an两个有序序列（它们相应的关键字值满足Kh<=…<=Km,Km+1<=…<=Kn）合并成一个有序序列Rh，…,Rn（使其关键字值满足Kh<=…<=Kn）。请分析算法，并在\_\_\_\_\_\_\_\_上填充适当的语句。

void merge(list a,list R,int h,int m,int n)

{i=h;k=h;j=m+1;

while((i<=m)&&(j<=n))

{if(a[i].key<=a[j].key){R[k]=\_\_\_\_\_\_\_\_;\_\_\_\_\_\_\_\_;}

else{R[k]=\_\_\_\_\_\_\_\_;\_\_\_\_\_\_\_\_;}

k++;

}

while(i<=\_\_\_\_\_\_\_\_){R[k]=a[i];i++;k++;}

while(j<=\_\_\_\_\_\_\_\_){R[k]=a[j];j++;k++;}

}

此算法的执行时间为\_\_\_\_\_\_\_\_.

20．归并排序要求待排序列由若干个\_\_\_\_\_\_\_\_\_\_\_的子序列组成。

21．二路归并排序的时间复杂度是\_\_\_\_\_\_\_\_\_\_\_。

22．对于n个记录的集合进行归并排序，所需的附加空间消耗是\_\_\_\_\_\_\_\_\_\_\_。

23.设表中元素的初始状态是按键值递增的，分别用堆排序、快速排序、冒泡排序和归并排序方法对其仍按递增顺序进行排序，则\_\_\_\_\_\_\_\_\_\_\_最省时间，\_\_\_\_\_\_\_\_\_\_\_最费时间。

24．分别采用堆排序、快速排序、插入排序和归并排序算法对初始状态为递增序列的表按递增顺序进行排序，最省时间的是\_\_\_\_\_\_\_\_\_\_\_算法，最费时间的是\_\_\_\_\_\_\_\_\_\_\_算法。

三、单项选择

1. 以下说法错误的是 （ ）

①直接插入排序的空间复杂度为O(1)。

②快速排序附加存储开销为O(log2n)。

③堆排序的空间复杂度为O(n)。

④二路归并排序的空间复杂度为O(n),需要附加两倍的存储开销。

1. 以下不稳定的排序方法是 （ ）

①直接插入排序 ②冒泡排序 ③直接选择排序 ④二路归并排序

1. 以下稳定的排序方法是 （ ）

①快速排序 ②冒泡排序 ③直接选择排序 ④ 堆排序

1. 以下时间复杂性不是O(n2)的排序方法是 （ ）

①直接插入排序 ②二路归并排序 ③冒泡排序 ④直接选择排序

1. 以下时间复杂性不是O(nlog2n)的排序方法是 （ ）

①堆排序 ② 直接插入排序 ③二路归并排序 ④快速排序

1. 在文件局部有序或文件长度较小的情况下，最佳的排序方法是 （ ）

①直接插入排序 ② 冒泡排序 ③ 直接选择排序 ④归并排序

1. 对于大文件的排序要研究在外设上的排序技术，即 （ ）

①快速排序法 ② 内排序法 ③外排序法 ④ 交叉排序法

8．排序的目的是为了以后对已排序的数据元数进行（ ）操作。

①打印输出 ②分类 ③ 合并 ④查找

9．当初始序列已按健值有序时，用直接插入算法进行排序，需要比较的次数为（ ）

①n-1 ②log2n ③ 2log2n ④n2

10.快速排序在最坏的情况下的时间复杂度是 （ ）

①O(log2n) ②O(nlog2n) ③ O(n2) ④O(n3)

11.具有24个记录的序列，采用冒泡排序至少的比较次数是 （ ）

①1 ②23 ③ 24 ④ 529

12．用某种排序方法对序列（25，84，21，47，15，27，68，35，20）进行排序，记录序列的变化情况如下：

25 84 21 47 15 27 68 35 20

15 20 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

则采取的排序方法是 （ ）

①直接选择排序 ②冒泡排序 ③快速排序 ④二路归并排序

13．在排序过程中，健值比较的次数与初始序列的排列顺序无关的是 （ ）

①直接插入排序和快速排序 ②直接插入排序和归并排序

③直接选择排序和归并排序 ④快速排序和归并排序

14．（ ）方法是从未排序序列中依次取出元素与已排序序列中的元素作比较，将其放入已排序序列的正确位置上。

①归并排序 ② 插入排序 ③快速排序 ④选择排序

15（ ）方法是从未排序序列中挑选元素，并将其依次放入已排序序列的一端。

①归并排序 ②插入排序 ③ 快速排序 ④选择排序

16．（ ）方法是对序列中的元素通过适当的位置交换将有关元素一次性地放置在其最终位置上。

①归并排序 ②插入排序 ③快速排序 ④选择排序

17．将上万个一组无序并且互不相等的正整数序列，存放于顺序存储结构中，采用（ ）方法能够最快地找出其中最大的正整数。

①快速排序 ②插入排序 ③ 选择排序 ④ 归并排序

18一般情况下，以下四种排序方法中，平均查找长度最小的是 （ ）

①归并排序 ②快速排序 ③选择排序 ④插入排序

19．以下四种排序方法中，要求附加的内存容量最大的是 （ ）

①插入排序 ②选择排序 ③快速排序 ④归并排序

20已知一个链表中有3000个结点，每个结点存放一个整数，（ ）可用于解决这3000个整数的排序问题且不需要对算法作大的变动。

①直接插入排序法 ②简单选择排序方法

③快速排序方法 ④堆排序方法

21．若用冒泡排序法对序列（18，14，6，27，8，12，16，52，10，26，47，29，41，24）从小到大进行排序，共要进行（ ）次比较。

①33 ②45 ③70 ④91

22．在任何情况下，快速排序方法的时间性能总是最优的。这种说法

①正确 ②错误

23．对一个由n个整数组成的序列，借助排序过程找出其中的最大值，希望比较次数和移动次数最少，应选用（ ）方法。

①归并排序 ②直接插入排序

③直接选择排序 ④快速排序。

四、简答及应用

1．对于给定的一组键值：83，40，63，13，84，35，96，57，39，79，61，15，分别画出应用直接插入排序、直接选择排序、快速排序、堆排序、归并排序对上述序列进行排序中各趟的结果。

2．举例说明本章介绍的各排序方法中那些是不稳定的？

3．相对于树形选择排序，直接选择排序和堆排序有何优点？

4．试比较直接插入排序、直接选择排序、快速排序、堆排序、归并排序的时、空性能。

5．判断下列两序列是否为堆？如不是，按照建堆的思想把它调整为堆，并用图表示建堆的过程。

（1）（3，10，12，22，36，18，28，40）；

（2）（5，8，11，15，23，20，32，7）。

6．对于下列一组关键字46，58，15，45，90，18，10，62，试写出快速排序每一趟的排序结果，并标出每一趟中各元素的移动方向。

7．已知数据序列为（12，5，9，20，6，31，24），对该数据序列进行排序，试写出插入排序和冒泡排序每趟的结果。

五、算法设计

1. 设计一个用链表表示的直接选择排序算法。
2. 写出非递归调用的快速排序算法。
3. 插入排序中找插入位置的操作可以通过二分法查找的方法来实现。试据此写一个改进后的插入排序方法。
4. 一个线性表中的元素为正整数或负整数。设计一个算法，将正整数和负整数分开，使线性表前一半为负整数，后一半为正整数。不要求对这些元素排序，但要求尽量减少交换次数。
5. 已知（k1,k2……，kn）是堆，试写一个算法将（k1,k2,……，kn,kn+1）调整为堆。按此思想写一个从空堆开始一个一个填入元素的建堆算法（题示：增加一个k n+1后应从叶子向根的方向调整）。
6. 设计一个用链表表示的直接插入排序算法。
7. **参考答案**
   1. 名词解释 （略）
   2. 填空题

1、数据表示 数据处理 2、机内表示

3、逻辑结构 逻辑结构上的基本运算 存储结构和运算 评价和选择

4、逻辑性 基本运算 5、存储

6、机外表示 逻辑结构 存储结构

7、处理要求 基本运算和运算 算法

8、数据 数据元素 数据项

9、元素 结点 顶点 记录

10、字段 域 11、数据元素 数据项

12、集合 线性结构 树形结构 图状结构

13、加工 引用 14、定义在S上的运算 S上运算

15、归纳 16、机内表示

17、存储结点 数据元素之间关联方式的表示 附加设施

18、顺序存储方式 链式存储方式 索引存储方式 散列存储方式

19、给定逻辑结构S的存储实现 存储映象

20、程序 算法设计

21、运行终止的程序可执行部分 伪语言算法 非形式算法

22、时空性能 算法分析 23、正确性能 易读性 健壮性 高效性

24、时间性能（或时间效率） 空间性能（或空间效率） 计算量 存储量

25、标准操作 标准操作 计算量

26、最坏情况时间复杂性 最坏情况时间复杂度 平均时间复杂性 平均时间复杂度

27、时间复杂性 时间复杂度 28、算法输入规模

29、作为该算法输入的数据所含数据元素的数目，或与此数目有关的其他参数

30、1 log2n n n2  2n 实际不可计算 高效

31、设计 实现

32、数据结构的定义 数据结构的实现 数据结构的评价 选择

33、数据的逻辑结构 34、线性结构 非线性结构

34、O（n2） 36、o(log2n)。

* 1. 单项选择题

1.② 2.① 3.② 4.③ 5.① 6.② 7.④ 8.③ 9.③

10.③ 11.② 12.② 13.④ 14.④ 15.②

* 1. 简答及应用
     1. 凡能被计算机存储、加工的对象通称为数据。

数据元素是数据的基本单位，在程序中作为一个整体而加以考虑和处理。换句话说，

数据元素被当作运算的基本单位，并且通常具有完整确定的实际意义。根据需要，数据

元素又被称为元素、结点、顶点或记录。

在很多情况下，数据元素又是由数据项组成的，但数据项通常不肯有完整确定的实际意义，或不被当作一个整体对待。在有些场合下，数据项又称为字段或域。它是数据的不可分割的最小标识单位。

从某种意义上说，数据，数据元素和数据实际反映了数据组织的三个层次，数据可由若干个数据元素构成，而数据元素又可由若干个数据项构成。

２、所谓逻辑关系是指数据元素之间的关联方式或称“邻接关系”。数据元素之间逻辑关系的整体称为逻辑结构。数据的逻辑结构就是数据的组织形式。关于逻辑结构的以下几点需特别注意：

（１）、逻辑结构与数据元素本身的形成、内容无关。

（２）、逻辑结构与数据元素的相对位置无关。

（３）、逻辑结构与所含结点个数无关。

由此可见，一些表面上很不相同的数据可以有相同的逻辑结构，因此，逻辑结构是数据组织的某种“本质性”的东西，是数据内部组织的主要方面。

　　　　３、逻辑结构反映数据元素之间的逻辑关系，而存储结构是数据结构在计算机中的表示，它包括数据元素的表示及其关系的表示。

　　　　４、一般地，运算是指在任何逻辑结构上施加的操作，即对逻辑结构的加工。一个运算的实现是指一个完成该运算功能的程序。

　　　　　　相同点：运算与运算的实现都能完成对数据的“处理”或某种特定的操作。

　　　　　　不同点：运算只描述处理功能，不包括处理步骤和方法，而运算实现的核心是处理步骤。

　　　　５、类Ｃ语言基本上是标准Ｃ语言的简化。类Ｃ语言与标准Ｃ语言的主要区别如下：

1. 局部量的说明可以省略（但形参表中及函数类型的说明需保留），重要的变量需在注解中用文字说明基类型和作用。
2. 分情形语句可以采用下述形式：

　switch

{ case 条件１：语句序列１；break;

case 条件2：语句序列2；break;

……

case 条件n：语句序列n；break;

default: 语句序列n+1;

}

其中“default: 语句序列n+1;”可以省略。

1. 不含goto语句，增加了一个出错处理语句error（字符串），其功能是终止它所在算法的执行并回送表示出错信息的字符串。
2. 输入输出语句有：

输入语句 scanf([格式串]，变量度，……，变量n);

输出语句 printf([格式串]，变量度，……，变量n);

(５) 类Ｃ语言的形参书写比标准Ｃ语言简单，如int abc (int a,int b,int c)可以简写为 int abc(int a,b,c)。

　　　五．算法设计

　　　　１．（１）int locate(dataytpe A[1..n],dateytpe k)

{ i=n;

while ((I<=n)&&(A[i]!=k)) I++;

if (I<=n) return(i);

else return(o);

}

当查找不成功时，总是比较n+1次，所以，最坏时间复杂性为n+1。其量

T（n）=O(n).

（２）Void CZ\_max(datatype A[n],x,y)

{ x=A[1]; y=A[1];

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

if(x<A[i]{y=x;x=A[i];} /\*替换最大值\*/

else if(y<A[i] y=A[i]; /\*替换次最大值\*/

}

若经条件判断语句为标准操作，则最坏情况时间复杂性为n-1。其量级为

T(n)=O(n).

第二章 参考答案

1. 名词解释 （略）
2. 填空题

1、结点 起始 终端 序号 位置 前趋 后趋

2、（） ф

3、前趋 前趋 后趋 后趋

4、线性 5、线性 长度 表长 6、空表

7、初始化INITLATE（L） 求表长LENGTH（L） 读表长GET（L，i） 定位LOCATE

（L，X） 插入INSERT（L，X，i） 删除DELETE（L，i）

8、逻辑结构中相邻的结点在存储结构中仍相邻

9、b+（i-1）x k

10、L．data[j]=L．data[j-1]

11、n O(n) n/2 O(n)

12、L.data[j-2]=l.data[j-1]

13、n-1 o(n) (n-1)/2 O(n)

14、i=1 i≦L.last

15、O(n) O(1)

16、L.last L.data[i-1]

17、单链表 循环链表 双链表

18、指针 19，单链表

20、头结点 表结点

21、首结点 尾结点 任何信息、特殊标志 表长

22、头结点 头结点

23、t=malloc(size) t->next=NULL

24、p=haed p=p->next

25、(p->next!=NULL)&&(j<I)

26、(p->next!=NULL)&&(p->data!=x)

27、(p!=NULL)&&(p->next!=NULL) p->next

28、mailloc(size) p->next

29、insert\_lklist(head,x,I) I++ n(n-1)/2 O(n2)

30、p=q p->next=NULL O(n)

31、单向循环链表（简称循环链表） 双向循环链表（简称双链表）

32、NULL 头结点 33、双链表

34、字符数组 赋值 35、空 ф 非空 字符

36、长度 相同 子 主 37、非紧缩 紧缩

38、结点大小 不属于字符集的特殊符号

1. 单项选择题

1、②2、①3、①4、②5、①6、②7、③8、③9、④

10、②11、④12、③13、⑤14、④15、③16、①17、②18、③

19、④20、④21、④22、223、②24、③25、④26、②27、③

28、④29、①30、④31、②32、②33、④34、④35、③36、③

37、②38、③39、②40、①41、④

1. 简答及应用

1、线性表的数据元素的类型为datatype，则在语言上可用下述类型定义来描述顺序表：

const maxsize=顺序表的容量；

typedef struct

{ datatype data[maxsize]

int last;

}sqlist;

sqlist L;

数据域data是一个一维数组，线性表的第1，2……，n个元素分别存放在此数组的第0，1，……，last-1个分量中，数据域last表示线性表当前的长度，而last-1是线性表的终端结点在顺序表中的位置。常数maxsize称为顺序表的容量，从last到maxsize-1为顺序表当前的空闲区（或称备用区）。

Sqlist类型完整地描述了顺序表的组织。L被说明为sqlist类型的变量，即为一顺序表，其表长应写为L.last，而它的终端结点则必须写为 L.data[L.last-1]。

2、假设数据元素的类型为datatype。单链表的类型定义如下：

typedef struct node \*pointer

struct node

{datatype data;

pointer next;

};

typedef pointer lklist;

其中，①ponter是指向struct node类型变量的指针类型;②struct node是结构体类型规定一个结点是由两个域data和next组成的记录，其中data的结点的数据域,next是结点的链域；③lklist与pointer相同类型，用来说明头指针变量的类型，因而lklist也就被用来作为单链表的类型。

3、typedef struct dnode \*dpointer;

struct dnode

{datatype data;

dpointer prior,next;

}

typedaf dpinter dlklist;

链域prior和next分别指向本结点数据域data所含数据元素的直接前趋和直接后继所在的结点。所有结点通过前趋和后继指针链接在一起，再加上起标识作用的头指针，就得到双向循环链表。

4、顺序串的类型定义与顺序表类似，可描述如下：

　const maxlen=串的最大长度

typedef struct

{char ch[maxlen]

int curlen;

}string

5、链串的类型定义为：

const nodesize=用户定义的结点大小;

typedef struct node \*pointer;

struct node

{char ch[nodesize]

poinernext;

}

typedef pointer strlist;

当结点大小为1时，可将ch域简单地定义为：char ch

6、head称为头指针变量，该变量的值是指向链表的第一个结点的指针，称为头指针。头指针变量是用于存放头指针的变量。

为了便于实现各种运算，通常在单链表的第一个结点之前增设一个类型相同的结点，称为头结点。其它结点称为表结点。表结点中和第一个和最后一个分别称为首结点和尾结点。

头指针变量的作用：对单链表中任一结点的访问，必须首先根据头指针变量中存放的头指针找到第一个结点，再依次按各结点链域存放的指针顺序往下找，直到找到或找不到。头指针变量具有标识单链表的作用，故常用头指针变量为命名单链表。

头结点的作用：头结点的数据域可以不存储任何信息，也可以存放一个特殊标志或表长。其作用是为了对链表进行操作时，将对第一个结点煌处理和对其它结点的处理统一起来。

7、循环单链表、循环双链表。

8、空串和空格串：含0个字符的串称为空串，其长度为0。空格串是含有一个或多处空格字符组成的串，其长度不为0。

串变量和串常量：串常量在程序的执行过程中只能引用不能改变而串变量的值在程序执行过程中是可以改变和重新赋值的。

主串和子串：一个串中任意个连续字符组成的子序列称为该串的子串，该串称为它的所以子串的主串。

串变量的名字与串变量的值：串变量的名字表示串的标识符；串变量的值表示串变量的字符序列。

9、(a)A+B “ mule”

(b)B+A “mule ”

(c)D+C+B “myoldmule”

(d)SUBSTR(B,3,2) “le”

(e)SUBSTR(C,1,0) “ ”

(f)LENGTH(A) 2

(g)LENGTH(D) 2

(h)INDEX(B,D) 0

(i)INDEX(C,”d”) 3

(j)INSERT(D,2,C) “myold”

(k)INSERT(B,1,A) “m ule”

(l)DELETE(B,2,2) “me”

(m)DELETE(B,2,0) “mule”

10、REPLACE(S,SUBSTR(T,7,1),SUBSTR(T,3,1));CONCAT(S,”y”);

1. 算法设计
   1. 分析：(1)当A、B表都不为空时，比较A，B表中各元素对应位置的内容的大小，进而判断A，B的大小。

(2)当A，B表都不为空时，且A，B表中各各元素对应位置的内容相同时，比较A，B的长度，进而判断A，B的大小或是否相等。

const maxsize=顺序表的容量;

typedef struct

{int data[maxsize]

int last;

}sqlist;

int CMP\_sqlist(sqlist A ,sqlist B)

{ for (i=0;(i<A.last)&&(I<B.last));i++}

{ if(A.data[i]<B.data[i])return(-1);

if(A.data[i]>B.data[i])return(1);

}

if(A.last= =B.last) return(0);

else if(A.last>B.last) return(1);

else return(-1);

}

2、(1)定位LOCATE(L，X)

在带头结点类单链表上实现的算法为：

int locate\_lklist(lklist head,datatype x)

/\*求表head中第一个值等于x的的序号，不存在这种结点时结果为0\*/

{p=head->next;j=1; /\*置初值\*/

while((p!=NULL)&&(p->data!=x))

{p=p->next;j++}/\*未达表结点又未找到值等于X的结点时经，继续扫描\*/

if (p->data = =x) return(j);

else return(0);

}

在无头结点的单链表上实现的算法为：

int Wlocate(lklist head,datatype X)

/\*求表head中第一个值等于x的结点的序号。不存在这种结点时结果为0\*/

{p=head; j=1; /\*置初值\*/

while((p!=NULL)&&(p->data!=x))

{p=p->next;j++}/\*未达表结点又未找到值等于X的结点时经，继续扫描\*/

if( p->data = =X) return(j);

else return(0);

}

(2)按序号查找find(L,i)

在带头结点的单链表上实现的算法为：

pointer find\_lklist(lklist head , int i);

{ j=1; p=head->next;

while((j<1)&&(p!=NULL)){p=p->next; j++}

if(i= = j) return(p);

else return(NULL);

}

在无头结点的单链表上实现的算法为：

pointer find\_lklist(lklist head , int i);

{ j=1; p=head;

while((j<1)&&(p!=NULL)){p=p->next; j++}

if(i= = j) return(p);

else return(NULL);

}

(3)、插入INSERT(L，X，i)

在带头结点单链表上实现的算法为：

void insert\_lklist(lklist head,datatype x,int I)

/\*在表haed的第i个位置上插入一人以x为值的新结点\*/

{p=find\_lklist(head,i-1); /\*先找第i-1个结点\*/

if(p= =NULL)reeor(“不存在第i个位置”)/\*若第i-1个结点不存在，退出\*/

else{s=malloc(size);s->data=x /\*否则生成新结点\*/

s->next=p->next /\*结点\*p在链域值传给结点\*s的链域\*/

p->next=s; /\*修改\*p的链域\*/

}

}

在无头结点的单链表上实现的算法为：

void Winsert(lklist head,dataytpe X,int i)

/\*在表haed的第i个位置上插入一人以x为值的新结点\*/

{if(i<=0) error(“i<=0”);

else{ s=malloc(size);s->data=X; /\*否则生成新结点\*/

if(i= =1){s->next=head;head=s;}

else{ p=wfind\_lklist(lklist head,i-1);

if(p= =NULL) error(“i>n+1”);

else{s->next=p->next;p->next=s;}

}

}

(4)删除DELDTE(L,i)

在带头结点的单链表上实现的算法为：

void delete\_lklist(lklist head,int i) /\*删除表head的第i个结点\*/

{p=find\_lklist(head,i-1) /\*先找待删结点的直接前驱\*/

if((p!==NULL)&&(p->next!=NULL))/\*若直接前趋存在且待结点存在\*/

(q=p->next; /\*q指向待删结点\*/

p->next=q->next/\*摘除待结点\*/;

free(q);/\*释放已摘除结点q\*/

}

else error(“不存在第i个结点”)/\*否则给出相关信息\*/

}

在无头结点的单链表上实现的算法为：

void Wdelete(lklist head,int i)

/\*　删除表head的第i个结点，若该链表仅有一个结点时，赋该结点指针NULL\*/

{if(i<=0) error(“I＜＝0”

else{if(i= =0){q=head;head=head->next;free(q);}

else{p=wfind\_lklist(head,i-1);/\*找链表head中第i-1结点指针\*/

if(p!=NULL)&&(p->next!=NULL)

｛q=p->next; p->next=q->next; free(q);｝

else error(“不存在第I个结点”)；

｝

｝

｝

1. 分析：从第一个结点开始访问，只要是非空计数一次。

Int wlength\_lklist(lklist head)　　　/\*求表head的长度\*/

{p=head;j=0;

while(p!=NULL){p=p->next;j++;}

return(j); /\*回传表长\*/

｝

1. 设A,B,C均为无头结点单链表

分析：（1）当有序表A，B均非空时，找出两表中元素最小的一个元素，然后将此结点插入到C表中，重复上述步骤。

（2）当A，B两表有一个为空表时，将另一表中元素顺序地插入到C表中。

（3）由于C按递减排序，因此在C表中插入元素时，应始终插入到C表表头。

Lklist Wlink\_lklist(lklist A,lklist B)

{ while((A!=NULL)&&(B!=NULL))

if(A->data<b->data){p=A;A=A->next;}

else

p->next=C;C=p;

}

if(A= =NULL) A=B;

while(A!=NULL)

{p=A;A=A->next;

p->next=C;C=p;}

return(C);

}

1. 分析：（1）当有序表A、B均非空时，依次分别从A、B表头部取下结点，插入C表中。

　　　　（2）当A、B两表有一个为空表时，将非空表插入到C表尾部。

设A，B，C均为带头结点的单链表

lklist HB\_lklist(lklist A,lklist B)

{ C=A;

A=A->next;B=B->next; //去除头结点

While((A!=NULL)&&(B!=NULL))

{p->next=A;p=p->next;A=A->next;

p->next=B;p=p->next;B=B->next;

}

if((B= =NULL)&&(A!=NULL)) p->next=A;

else if((A= =NULL)&&(B!=NULL)) p->next=B;

Return(c);

}

1. 分析：从有序表的尾部开始依次取元素与插入元素比较，若大于插入元素，此元素后移一位，再取它前面一个元素重复上述步骤；则将待插入元素插入。

Void CR(datatype A[],datatype X,int elenum)

{ i=elenum-1;

while((i>=0)&&X<A[i]){A[i+1]=A[i];I--}

A[i+1]=X;

Elenum=elenum+1;

}

算法的时间复杂度为O（n）。

1. 分析首先从表头开始查找待插入位置的直接前趋，找到后插入待插结点。

/\*设L表带头结点\*/

Void CR\_lklist(lklist L,datatype x)

{q=L;p=q->next;

while((p!=NULL)&&(p->data<x)){q=p;p=p->next;}/\*查X插入位置q\*/

s=malloc(size);s->data=s;

}

1. (1)顺序表

分析：将顺序表的第一个元素与最后一个元素互换，第二个元素与倒数第二个元素互换。

Void NZ\_sqlist(sqlist A)

{for{i=0;i<((A.last-1)/2);i++}

{x=A.data[i];

A.data[i]=A.data[A.last-i-1];

A.data[A.last-i-1]=x;

}

}

(2)单链表

分析：将原单链表的元素依次取出，再插入另一个单链表的头部。

设该单链表为无头结点，s为指向表的第一个结点的指针。

Void NZ\_lklist(lklist s)

{p=NULL; /\*p指向当前结点的前趋结点\*/

while(s!=NULL)

{ q=s;s=s->next; /\*将原单链表的元素依次取出到q\*/

q->next=p;p=q; /\*再插入另一个单链表p的头部\*/

}

s=p; /\*s指向新单链表的第一个结点\*/

}

1. 分析：A与B的交是指A与B的相同部分元素，即那些在A中出现又在B中出现的元素。由于A、B是有序表，故从表头开始依次比较当前指针所指元素的值是否相同，若相同，在C表中插入该元素，然后将两个表的指针后移，否则指向较小元素的指针后移。重复上述步骤，直到A，B表中有一个表到表尾。

(1)顺序表

sqlist HDZ\_sqlist(sqlist A,sqlist B)

{ t=0;j=0;k=0;

while((t<=A.last-1)&&(j<=B.last-1))

switch{ case A.data[i]<B.data[j];i++;break;

case A.data[i]>B.data[j];j++;break;

case A.data[i]= =B.data[j];C.data[k]=A.data[i];k++;i++;j++;break;

}

C.last=k;

Return(c );

}

(2)单链表（设带头结点）

lklist HDZ\_lklist(lklist A,lklist B)

{ C=initiate\_lklist();

r=C;p=A->next;q=B->next:;

While ((p!=null)&&(q!=null))

Switch

{ case p->data <q->data: p=p->next;break;

case p->data <q->data: q=q->next;break;

case p->data==q->data;

s=malloc(size);s->data=p->data;

s->next=r->next; r->next=s;

r=s; p=p->next;q=q->next;

}

return(c);

}

10.分析：①在有序表B、C中找出相同元素X；

②若X在表A中出现则删除，否则转①；

③重复①②直到B、C表有一个表查找完毕。

1. 顺序表

void ASBC\_sqlist(sqlist A, sqlist B, sqlist C)

{I=0;j=0;k=0;

while ((j<B.last)&&(k<C.last))

switch

{case B.data[j]<C.data[k]:j++;break;

case B.data[j]<C.data[k]:k++;break;

case B.data[j]==C.data[k]:/\*B、C中找到相同元素\*/

{while((I<A.last)&&( A.data[I]<B.data[j]))I++;/\*在A中查找B、C表共有元素\*/

if (I>= A.last) return;

if (A.data[I]==B.data[j])/\*在A中存在B、C表共有元素，删除\*/

{for(t=I+1;t< A.last;t++) A.data[t-1]= A.data[t] A.last--;}

j++;k++;

}

}

}

1. 单链表（设含头结点）

void ASBC\_lklist(lklist A,lklist B,lklist C)

{pa= A ->next;q= A;

pb= B ->next;pc= C ->next;

while ((pb!=null)&&(pc!=null))

switch

{

case pb->data<pc->data: pb=pb->next;break;

case pb->data<pc->data: pc=pc->next;break;

case pb->data= =pc->data:/\* B、C中找到相同元素\*/

if(pa= =null)return;

if (pa->data= =pb->data)/\*在 A中存在 B、C 表共有元素，删除\*/

{q->next=pa->next; free(pa);pa=q->next;}

pb=pb->next;pc=pc->next;

}

}

}

11．分析设置两个指针，分别指向\*S及其后继，然后按循环链表特性，顺序往下查找\*s的

直接前趋，找到后删除；

void DELETE\_Xlklist(lklist S)

{p=s; q=p->next;

while (q->nest!=s)

{ p=q; q=q->next;}

p->next=s; free(q);

}

12．分析：在链表L中依次取元素，若取出的元素是字母，把它插入到字母B中，然后在L中删除该元素；若取出的元素是数字，把它插入到数字链D中，然后在L中删除该元素。继续取下一个元素，直到链表的尾部。最后B、D、L中分别存放的是字母字符、数字字符和其它字符。

设原表有头结点、头指针L，新建数字字符链D，字母字符链B，其它字符链R。

void DISM\_lklist(lklist L,lklist D,lklist B,lklist R)

{ D =malloc(size of(int)); D ->next= D; /\*建D循环链表头结点\*/

B =malloc(sizeof(char)); B ->next= B; /\*建B循环链表头结点\*/

p= L;q=p->next;

while(q!=null)

{if((q->data<=’9’)&&(q->data>=’0’))

{p->next=q->next; /\*在表L 中摘除q结点\*/

q->next= D ->next; D ->next=q; /\*将q结点插入D中\*/

q=p->next; /\*移动q指针\*/

}

else if (q->data<=’z’)&&(q->data>=’a’)||(q->data<=’z’)&&(q->data>=’a’))

{p->next=q->next; /\*在表L中删除q 结点\*/

q->next= B ->next; B ->next=q; /\*将q结点插入B中\*/

q=p->next; /\*移动q指针\*/

}else {p=q;q=p->next;} /\*移动q指针\*/

}

p->next=L;R=L; /\*使R为循环表\*/

}

13．分析：使用一维数组，数组下标表示元素的序号，数组值为1表示元素存在，数组

值为0表示此元素不存在。若累加数组的下标大于N，再从1开始继续累加数组值，直到

所有元素都输出。

Void JSP( int n, k, a[n])

{for(I=0;I<n;I++) a[I ]=1;

j=0;

for(I=0;I<n;I++)

{s=0;

while (s<l){j=(j%n) +1;

s=s+a[j-1];

printf(“%d”,j);a[j-1]=0;

}

}

14．（1）始化

dlklist initiate\_dlklist()

{t=malloc(size);

t->next =null;

t->prior=null;

return(t);

}

(2)定位（设含头结点）

int locate\_dlklist (dlklist head, datatype x)

/\*求表head中第一个值等于x 的结点的序号。不存在这种结点时结果为0 \*/

{p=head->next;j=1; /\*置初值\*/

while((p!=null)&&(p->data!=x))

{p=p->next;j++;} /\*未达尾结点又未找到值等于x的结点时继续扫描\*/

if (p->data = = x) return (j);

else return (0);

}

1. 插入（设含头结点）

void insert\_dlklist (dlklist head, datatype x, int I)

/\*在表head的第I 个位置上插入一个以x为值的新结点\*/

{p=head->next; j=1; /\*先找第I-1个结点\*/

while ((p!=null) &&(j<I-1))

if ((I-1)!=j) error(“不存在第I个位置”)/\*若第I-1个结点不存在，退出\*/

else {s=malloc(size);s->data=x;/\*否则生成新结点\*/

s->prior=p;

s->next=p->next;

p->next->prior=s;

p->next=s;

}

}

1. 删除（设含头结点）

void deldtd\_dlklist(dlklist head, int I) /\*删除表head的第i个结点\*/

{p=head->next;j=1; /\*先找第i个结点\*/

while((p!=null)&&(j<I)) {p=p->next;j++;}

if(I!=j) error(“不存在第I个位置”)/\*若第i个结点不存在，退出\*/

if(p!=null) /\*若直接前趋存在且待删结点存在\*/

{p->prior->next=p->next;

p->next->prior=p->prior;

free(p);/\*释放已摘除结点p\*/

}

else error(“不在存在第I 结点”) /\*否则给出相关信息\*/

}

15．分析：首先在链表中查找元素值为X的结点，若找到则让freq域的值增1；然后

依次和它的前趋的freq域值比较，若比前freq域值大，和前趋结点位置交换，直到比

前趋结点的freq域值小为止。

Typedef struct dfnode \*dfpointer;

Struct dfnode

{datatype data;

int freq;

dfpointer prior,next;

}

typedef dfpointer dflklist;

设该双链表含头结点。

Int LOCATE\_dflklist(dflklist L,datatype X)

{/\*定位值等于X的结点\*/

p=L->next; I=1;

while ((p!=null)&&(p->data!=X))

{o=p->next; I++;}

if ((p->data!=X||(p= = null)) error(“不存在值为X的结点 ”);

else { p->freq++; /\*令元素值为X的结点中freq域的值增1\*/

q=p->prior;

while((q!= L)&&(q->freq<p->freq))

{I=I-1;

p->prior->next=p->next; /\*摘除p\*/

p->next->prior=p->prior;

q->prior-<next=p /\*在q前插入p\*/

p->prior=q->prior;

p->next=q;

q->prior=p;

q=p->prior; /\*q重新指向p的前趋\*/

}

return(i);

}

16．分析：将X串的结点依次与Y串中的结点值比较，若找到第一个不在Y中出现的结点，则查找成功，返回该值。

设单链表无头结点

char CZC\_lklist (lklist X,lklist Y)

{p=x;

while (p!=null)

{q=Y;

while (q!=null)&&(q->data!=p->data) q=q->next;

if (q!=null) return(p->data);

else p=p->next;

}

return(‘＄’);

/\*若X中的字符都在Y中出现，则返回’＄’\*/

}

17.const maxlen=串的最大长度;

typedef struct

{char ch [maxlen];

int curlen;

} string;

int EQUAL\_string(string s,string t )

{ if (s.curlen!=t.curlen) return(0);

for (t=0; t<s.curlen; t++)

if (s.ch[t]!=t.ch[t] ) return(0);

return(1);

}

18.设单链表无头结点

const nodesize =用户定义的结点大小；

typedef struct node \*pointer;

struct node

{char ch;

pointer next;

}

typedef pointer strlist;

int EQULA\_strlist(strlist s ,strlist t )

{ while ((s!= null )&&(t!=null)&&(s->ch==t->ch))

{s=s->next;t=t->next;}

return((t= = null)&&(s= =null));

}

19．分析：首先判断串T是否为串S的子串，若串T是串S的子串对S中该子串逆置。

Int NZ\_strlist (strlist s,strlist t)

{p=s->next;t=t->next;q=s;

while(p!=null)

{pp =p ; tt =t; /\*判断串T是否为串S的子串\*/

while ((tt!=null)&&(pp!=null)&&(pp->ch= =tt->ch))

{pp=pp->next;tt=tt->next;}

if (tt==null) /\*串T是串S的子串对S中的该子串的位置\*/

{qq=q->next; /\* q是子串的第一个结点前趋pp是子串最后一个结点后继\*/

while(qq!=pp)

{g=qq; qq= qq->next;

q->next =pp; pp=g;

}

q->next=pp; /\*将该子串的前趋与逆置后的子串的相连\*/

return(1);/\*找到并逆置返1 \*/

}else {q=p;p=p->next;}

}

return(0);/\*找不到匹配的串返0\*/

}

20．Int index(char \*s, char \*t)

{l=LENGTH(s);k=LENGTH(t);

for (j=1;j<=l;j++)

{if (( l-j+1)>=k)

{ASSIGN(m,SUBSTR(s,j,k));

if (EQUAL(m,t )) return(j);

}else return(0);

}

}

1. 参考答案

一、名词解释（略）

二、填空题

1. 先进后出、后进先出，后进先出，进栈，入栈，退栈，出栈
2. 初始化InitStack(S)、进栈Push(S,X), 退栈Pop(S)，读栈顶Top(S),判栈空Empty(S)
3. 下溢
4. 上溢
5. 顺序、链接
6. 栈空、下溢、栈满、上溢
7. sq->top=0
8. sq->top++,sq->data[sq->top]
9. sq->data[sq->top],sq->top—
10. sq->top= =0
11. sq->top= =0,sq->data[sq->top]
12. ls=NULL
13. p->data=x,ls=p
14. p->data,free(p)
15. \*x=ls->data
16. 更小的“尺度”、递归
17. 队、队尾、队头
18. 队列初始化InitQueue(Q)、入队列EnQueue(Q,X)、出队OutQueue(Q,X)、判队列空EmptyQueue(Q)、读队头ead(Q,x)
19. 假溢出
20. sq->front=0
21. sq->front,sq->rear=(sq->rear+1)%maxsize,sq->data[sq->rear]=x
22. sq->rear,sq->fornt=(sq->rear+1)%maxsize,\*x= sq->data[sq->rear]
23. sq.rear= sq.front
24. sq.front,(sq.front+1)%maxsize
25. 队满、队空
26. lq->front=p,NULL
27. p->data,p,lq->rear=p
28. \*x,s->next
29. lq.rear= =lq.front
30. p=lq.front->next,\*x
31. n-1,读、写
32. 顺序、列序、行序、行、列
33. 特殊、稀疏
34. n(n+1)/2

35、 i(i-1)/2+j 当i≧j

　　　k=

j(j-1)/2+i 当i<j

36、n-t+1,(i-1)(2n-i+2)/2,j-i+1

(i-1)(2n-i+2)/2+j-i+1 当i<=j

k=

n(n+1)/2+1 当i>j

n(n+1)/2+1

37、　　　(i-1)/2+j 当i≧j

　k=

n(n+1)/2+1 当i<j

38、col<=a.nu,a.data[p].j,q++

39、col<=a.nu,cpot[col-1]+num[col-1],cpot[col]++

40、先进后出（后进先出）

41、先进先出（后进后出）

42、ls= =NULL,\*x=p->info

43、2230,2374

44、n-1

45、栈

46、EA+222,EA+117

47、lq->front->next= =lq->rear

48、540,108,M[3][10]

三、单项选择题

1、④2、①3、④4、①5、③6、②7、①8、③9、④10②、

11、②12、③13、①②14、②15、④16、①17、②18、③19、④20、①

21、②22、①23、③24、①25、②26、③27、②28、②29、②30、③

31、②32、②33、③34、②35、④

四、简答及应用

1、顺序栈类型定义如下：

#define sqstack\_maxsize 顺序栈的容量

typedef struct sqstack

{DataType data[sqstack\_maxsize];

int top

}SqStackTp

它有两个域：data和top。Data为一个一维数组，用于存储栈中元素，DataType为栈元素的数据类型。Top为int型，它的实际取值范围为0~sqstack\_maxsize－1。

2、链栈的类型定义如下：

typedef stuct node

{ DataType data;

struct node \*next;

}LstackTp;

单链表的第一个结点就是链栈栈顶结点，链栈由栈顶指针惟一确定。栈中的其它结点通过它们的next域链接起来不。栈底结点的next域为NULL。

3、顺序队列的类型定义如下：

#define maxsize 顺序栈的容量

typedef struct sqqueue

{DataType data[maxsize];

int fornt,rear

}SqQueueTp

SqQueueTp sq;

该类型变量有三个域：data,front,rear。其中data存储队中元素的一维数组。队头指针front和队尾指针rear定义为整型变量，实际取值范围为0~maxsize－1。

循环队列的类型定义如下：

#define maxsize　　　循环队的容量

typedef struct cycqueue{

DataType data[maxsize]

Int front,rear

}CycqueueTp;

CycqueueTp sq;

4、typedef struct linked\_queue

{ DataType data;

struct linked\_queue \*next;

}LqueueTp;

typedef struct queueptr

{ LqueueTp \*front, \*rear;

}QueptrTp;

QueptrTp lq;

5、#define maxnum　　　非零元素的容量

typedef struct node

{ int i,j ;　　/\*非零元素所在的行号、列号\*/

DataType v;　　/\*非零元素的值\*/

}NODE;

typedef struct spmatrix

{ int mu,nu,tu;　　/\*行数、列数、非零元素的个数\*/

NODE data[maxnum+1];/\*这里假定三元组的下标的起始值为1\*/

}SpMatrixTp

6、int length(CycqueueTp sq)

{len=(sq.rear-sq.front+maxsize)%maxsize;

return(len);

}

7、1234、4321、2143、3421、3241、1324、1432、1342、1243、3214、2134、2314、2341、2431

8、 i(2n-i+1)/2 当i<=j

f1(i)=

1. 当i>j

j 当i<=j

f2 (j)=

1. 当i>j

-n 当i<=j

c=

n(n+1)/2+1 当i>j

9、(1)k=2i+j-2; (i,j=1,2,…..n)

(2)i=ceil((k+1)/3) j=floor(k/3)+k mod 3

10、运行结果：ABCDEFGHIJKLM

MLKJIHGFEDCBA

11、借助栈将一个带头结点的单链表倒置。

12-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Top-> |  |  | Top-> |  |  | Top-> |  |  | Top-> |  |  | Top-> |  |  | Top-> |  |  |
|  |  |  |  |  |  |  |  |  |  | 1 | r’’’’ |
|  |  |  |  |  |  |  |  | 2 | r’’’ | 2 | r’’’ |
|  |  |  |  |  |  | 3 | r’’ | 3 | r’’ | 3 | r’’ |
|  |  |  |  | 4 | r’ | 4 | r’ | 4 | r’ | 4 | r’ |
|  |  | 5 | r | 5 | r | 5 | r | 5 | r | 5 | r |
|  |  |  |  |  |  |  |  |  |  |  |  |

调用f(5)前　　调用f(5)前　　调用f(4)前 调用f(3)前 调用f(2)前 调用f(1)前

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Top-> |  |  | Top-> |  |  | Top-> |  |  | Top-> |  |  | Top-> |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 2 | r’’’ |  |  |  |  |  |  |  |  |
| 3 | r’’ | 3 | r’’ |  |  |  |  |  |  |
| 4 | r’ | 4 | r’ | 4 | r’ |  |  |  |  |
| 5 | r | 5 | r | 5 | r | 5 | r |  |  |
|  |  |  |  |  |  |  |  |  |  |

返回f(1)后　　　　　返回f(2)后　　　返回f(3)后　　　　返回f(4)后　　　返回f(5)后

**五、算法设计**

1．本程序中，将客车类定义一个队KE，货车类定义一个队HE，过江渡船定义成一个栈DC。栈采用顺序存储结构，队采用链式存储结构。

#define sqstack\_maxsize 10

typedef struct sqstack

{DataType data[sqstack\_maxsize];

int top;

}SqStackTp;

typedef struct linked\_queue

{DataType data;

struct linked\_queue \*next;

}LqueueTp;

typedef struct queueptr

{LqueueTp \*front,\*rear;

}QueptrTp;

int InitStack(SqStackTp \*sq) {sq->top=0; return(1);}

void InitQueue (QueptrTp \*lp)

{LqueueTp \*p;

p=(LqueueTp \* )malloc(sizeof(LqueueTp));

lq->front=p;

lq->rear=p;

( lq->front)->next=NULL;

}

int QutQueue(QueptrTp \*lp,Data Type \*x)

{LqueueTp \*s;

if (lq->front==lq->rear) {error(“队空”);return(0);}

else {s=(lq->front)->nest;

\*x=s->data;

(lq->front)->next=s->next;

if (s->next == NULL) lq->rear=llq->front;

free(s);

return(1);

}

}

int EmptyQueue(QueptrTp lq)

{if (lq.rear==lq.front) return(1);

return(0);

}

int Push (SqStackTp \*sq , DataType x)

{ if (sq ->top = =sqstack\_maxsize-q) {return(0);}

else {sq ->top++; sq->data[sq->top]=x;

return(1);}

}

void main()

{ SqStackTp DC; //DC表示渡船

QueptrtTp KE ,HE; // KE表示客车E、HE表示货车

Int t ,j=0;

Initstack(DC);

Initqueue(KE);

Initqueue(HE);

While(DC.top<sqstack\_maxsize)

{j=o;

for (I=I;j<=4;I++) //先上4辆客车

if (!emptyqueue(KE)&&(DC.top <sqstack\_maxsize))

{ outqueue (&KE, &t);Push (&DC, t ); j++:}

for (I=j;I<5;I++) //再上1辆货车或客车不足时用货车补足

if (!emptyqueue(HE)&& (DC.top < sqstack \_maxsize))

{outqueue(&HE,&t); Push(&DC, t);j++;}

if (j<5) for (I=j;I<5;I++) // 当货车不足时用客车补足

if (!emptyqueue(KE)&&(DC.top <sqstack\_maxsize))

{outqueue(&KE,&t);Push (&DC,t ) ; j++}

else printf (“客车、货车合计不足10辆！”);

}

}

2.typedef struct dustack

{DataTypeelem[1:M];

int top0 ,top1;

}dustktp

void initstack (dustktp \***S ) /\***初始化\*/

{S->top0=0; S->top1=M+1;}

void push (dustktp \***S,DataType X, int I ) /**\*I指示是栈0或栈1入栈\*/

{if (S->top0= =s->top1-1)error(“栈满！”)；

else {if (I==0){s->top0++;S->elem[S->top0]=X;}

else {S->top1--; S->elem[S->top1]=X}

}

}

void pop(dustktp \***S, DataType \*X, int I ) /**\*I指示是栈0或栈1入栈\*/

{if (I==0)

{if (S->top0==0) error (“栈空！”)

else {\*X =S->elem[S->top0];S->top0--;}

}

else {if (S->top1= =M+1) error(“栈空！”)

else {\*X =S->elem[S->top1];S->top1++;}

}

}

3.void initqueue(lklist \*lq) **/**\*初始化\*/

{lq=malloc(size);

lq->next=lq;

}

void EnCycQueue(lklist \*lq ,DataType \*X) **/**\*入队列\*/

{ p=malloc(size);p->data=X;

p->next=lq->next;

lq->next=p;

lq=p;

}

void outqueue(lklist \*lq ,DataType \*X)  **/**\*出队列\*/

{if (lq->next=lq )error(“队空！”);

else{p=lq->next;q=p->next;}

if (q= =lq ) lq=p ;

p->next=q->next;\*X=q->data;

free(q);

}

4．设cycque[m]\ rear\quelen皆为全局变量

viod Enqueue (DataType cycque[m], DataType X)

/\*入队列\*/

{ if (quelen= =m+1) error(“队满!”);

else {real=(real+1)%(m+1);

cycque[real]=X;quelen++;

}

}

void outqueue(DataType cycque[m], DataType \*X)

/\*出队列\*/

{if (quelen ==0) error (“队空!”);

else { frone =(real- quelen +1+(m+1))%(m+1); /\*计算对头下标\*/

\*X=cycque [frone]; quelen--;

}

}

5. void trans\_mat \_trix(DateType a[m][n],SpMatrixTp b )

{p=0;

for (I=1; I<=m ; I++)

for (j=1; j<=n;j++)

if (a[I][j]) /\*非零元素\*/

{p++; /\*给三元组赋值\*/

b.data[p].i=I;

b.data[p].j=j;

b.data[p].v=a[I][j];

}

b.mu=m; b.nu=n; b.tu=p; /\*赋行数、列数和非0元素数\*/

}

6．本题首先判断三元组A，B表示的矩阵是否行列相同，若相同才能进行矩阵的加法

运算。若三元组表示的矩阵能进行相加运算，其思路如下：

若a,b表的指针均没有到表尾，重复下列步骤：

1. 若a表元素的i 域值小于b表元素的i域值，将a表当前元素插入到c表表尾， a表指针后移。
2. 若a表元素的i 域值大于b表元素的i域值，将b表当前元素插入到c表表尾， b表指针后移。
3. 若a表元素的i域值等于b表元素的i域值，又分以下几种情况讨论：

①若a表元素的j域值小于b表元素的j域值，将a表当前元素插入到c表表尾，a表指针后移。

②若a表元素的j域值大于b表元素的j域值，将b表当前元素插入到c表表尾，b表指针后移。

③若a表元素的j域值等于b表元素的j域值，若a,b表当前元素的v域值和非零，则在c表表尾播入元素的I\j域值等于a表当前元素的I\j域 的值，域v的值等于a,b表的域值的和，将a,b表当前指针后移。

(4) 若a表的指针没到表尾，b表的指针到表尾，将a表剩余元素依次插入到c表表尾，否则，将b表剩余元素依次插入到c表表尾.

SpMaterixTp trsxsum(SpMaterixTp a, SpMaterixTp b , SpMaterixTp c)

{if ( (a .mu=b. mu)&&(a.nu=b.mu)) /\*a，b为同行同列矩阵\*/

{c.mu=a.mu ;c.nu=a.nu; I=1; j=1; p=0;

if (a.tu&&b.tu) /\*a，b为非空矩阵\*/

{cola=a.data[I].i; rowa = a.data[I] .j; /\*取三元组a的元素的行、列下标\*/

colb= b.data[j].i; rowb =b.data[j] .j; /\*取三元组b的元素的行、列下标\*/

while ((I<=a.tu)&&(j<=b.tu))

switch

{cola <colb: /\*在c中，插入三元组a的元素\*/

p++; c.data [p] .i=a.data [I] .i;

c.data [p].j= a.data [I].j;

c.data [p] .v= a.data[I].v;

I++; break; /\*a元素后移\*/

Cola>colb: /\*在c中，插入三元组b的元素\*/

P++;c.data[p] .i=b.data [j] .i;

c.data [p].j= b.data [j].j;

c.data [p] .v= b.data[j].v;

j++; break; /\*b元素后移\*/

Cola =colb :

P++; /\*三元组 a，b的元素I域相等\*/

If (rowa<rowb) /\*在c中插入三元组a的元素\*/

{c.data[p].j=a.data [I].j;

c.data[p].j=a.data[j].j;

c.data[p].v=a.data[I].v; I++;

}

else if (rowa>rowb) /\*在c中插入三元组b的元素\*/

{c.data[p].i=b.data[j].i;

c.data[p].j=b.data[j].j;

c.data[p].v= b.data[j].v; j++;

}

else if (a.data[I].v +b.data[j].v)

/\*a中的元素和b中的元素I，j的域都相等且v域的和非零\*/

{c.data [p].i =a.data[I].i ;/\*在c中元素\*/

c.data[p].j=a.data[j].j;

c.data[p].v=a.data[I].v+b.data[j].v;

I++; j++;

} else {p--;I++;j++}; /\*a中元素和b中元素的I，j域都相等且v 域的和为零\*/

break;

} /\*swith结束\*/

}

if (I<=a.tu) /\*b表到表尾，将a表中剩余元素插入到C表中\*/

{p++; c.data[p].i =a.data[p].i; c.data[p].j=a.data[p].j;

c.data[p].v=a.data[p].v; I++;

else{ /\*a表到表尾，将b表中剩余元素插入到C表中\*/

p++; c.data[p].i=b.data[j].i; c.data[p].j=b.data[j].j;

c.data[p].v=b.data[j].v; j++;}

}c.tu=p; /\*c表赋非零元素个数\*/

}

7.设表达式已存入字符数组A[n]中。

Void prool(A[n])

{Initstack (d); I=0; flag =true;

while ((I<n)&&(flag))

{if （(A[i]=‘(’)||(A[I]=‘[’||(A[I]=‘{’)Push (s,A[I]);

else if （(A[i]=‘)’ ）||（A[Push (s,A[I]);

if (emptystack(s)) flag= false;

else{x=GetTop(s);

switch(a[I])

{case’}‘:if (x=’(’)pop (s);

else flag =false;

break;

case’]’:if (x=’[]’) pop(s);

else flag=false;

break;

case’}’:if (x=’{}’) pop(s);

else flag=false;

}}

I++;

}

}

8.int akm(int m ,int n)

{if (m= =0) return(n+1);

else if (n= =0) return(akm (m-1,1));

else return(akm(m-1,akm(m,n-1)));

}

9. int f(int m, int n )

{if (m\*n= =0) return(m+n+1);

else return(f(m-1,f(m,n-1)));

}

10. 用Knap(S,n)表示背包问题的解，这是一个布尔函数，其参数应满足 S>0,n≥1。背包问题如果有解，其选择只有两种可能：一种是选择的一组物品中不包含Wn，这样Knap(S,n)的解就是Knap(S,n-1)的解，另一种是选择中包含Wn，这时Knap(S,n)的解就是Knap(S-Wn,n)的解。另外可以定义：当S=0时，背包问题总有解，即Knap(0,n)=true ，只要不选择任何物品放入背包即可：当S〈0时，背包问题无解，即Knap(S,n)=false，因为无论怎样选择总不能使重量之和为负值，当S>0但n<1时，背包问题也无解，即Knap(S,n)=false，因为不取任何东西就要使重量为正值总是办不到的。从而，背包问题可以递归定义如下：

╭

| true, 当S=0

| false, 当S<0

Knap(S,n)= < false, 当 s>0且 n<1

| Knap(S,n-1)或Knap(S- ,n-1)， 当s>0 且 n>=1

|

上述递归定义是确定的，因为每递归一次n都减1，S也可能减少 ，所以递归若干次以后，一定会出现S≤0或者 n=0，无论哪种情况都可由递归出口明确定值。

Int knap(int s ,int n)

{if (s==0) return(1);

else if (s<0||(s>0&&n<1)) return(0);

else if (knap(s-w[n],n-1)){printf(“%d”,w[n]);return(1);}

else return(knap(s,n-1));

}

11.方法是先依次让单链表上的元素进栈，然后再依次出栈。

Void invert (lklist head)

{LstackTp s;

initstack(s);

p= head;

while (p<>null)

{Push (s,p->data);p=p->next;}

p=head;

while(not emptystack(s))

{pop(s,p->data); p=p->next;}

}

第六章 参考答案

一、名词解释（略）

二、填空题

1. 分支层次、根、直接前趋
2. 子孙、祖先
3. 空、只含根、非空左子树、非空右子树、非空左右子树
4. 2i-1
5. 2k-1
6. n2+1
7. 最大值、完全
8. floor(log2n)+1
9. 根、floor(i/2)、左孩子、右孩子、2i、右孩子、2i+1
10. 顺序、链式
11. 根
12. 根、root
13. 指向该结点的一个孩子、空指针NULL
14. 2n、n-1、n+1
15. 二叉链表、三叉链表
16. 虚结点
17. \*count++,countleaf(l->rchile,count)
18. 访问根结点、遍历左子树、遍历右子树
19. DLR、LDR、LRD、先根（或前序）遍历、中根（或中序）遍历、后根（或后序）遍历
20. 先根遍历、后根遍历、层次遍历
21. 非终端结点、终端结点
22. WPL（T）＝
23. i<k,x,T[j].wt,k+i,k+i,m+n,x,y
24. 第一
25. 先根
26. floor(n/2),l,n,floor(n/2)+1
27. 后面
28. 相同
29. 左子树、右子树、左子树、右子树
30. 树
31. 最短、较近
32. 2m-1
33. （A），（E、G、I、J、K、L、N、O、P、Q、R），（3），（4），（5），（J、K）（C）
34. 1、0
35. 树、二叉树
36. 只有一个根结点的树、空二叉树
37. n-1
38. n-2m+1
39. 5、答案见图 A B A C C

B A C C B A

C B A B

1. WPL＝165

　　　　　　　　　　　　　　　　62

　　　　　　　　　　　37　　　　　　　　　　25

　　　　　　19　　　　　　　　18　　 13　　　　　　　12

　　　10　　　　　9　　　　　　7　　　　　　6

　　　　　　5　　　　　4

41、m-1

42、本题有一定的难度，其求解方法如下：设总结点数为n度为0、1、2、3的结点数分别为n0,n1,n3,则有下面两个等式成立：

n=n0+n1+n2+n3 /\*结点数\*/

n-1=n1+2n2+3n3 /\*分支数\*/

　两式相减得 n0=1+n2+2n3=1+3+2x4=12 (12)

三、单项选择

1、①2、③3、④4、④5、②6、④7、③8、④9、③10、①

11、④12、②13、④14、①15、②16、①17、②18、④19、①20、②

21、④22、③23、①24、②25、②26、③27、③28、①29、②30、④

31、④32、②33、②34、③35、①36、③37、③

四、简答及应用题

1. 二叉链表的类型定义如下：

typedef struct btnode \*bitreptr;

struct btnode{ datatype data;

bitreptr lchild,rehild;

}

bitreptr root;

其中，data域称为数据域，用于存储二叉树结点中的数据元素；lchild 域称为左孩子指针域，用于存放指向本结点左孩子的指针（简称左指针）。rchild域称为右孩子指针域，用于存放指向本结点右孩子的指针（简称右指针）。

2．三叉链表与二叉链表的主要区别在于，它的结点比二叉链表的结点多一个指针域，该域用于存储一指向本结点双亲的指针。三叉链表的类型定义如下：

typedef struct ttnode \*ttnodeptr;

struct ttnode

{datatype data;

ttnodeptr lchild, rchild, parent;

}

ttnodeptr root;

3．typedef struct tagnode /\*表结点类型\*/

{int child; /\*孩子结点在表头数据组中的序号\*/

struct tagnode \*next;  **/**\*表结点指针域\*/

}node,\*link;

typedef struct /\*头结点类型\*/

{datatype data; /\*结点数据元素\*/

link headptr; /\*头结点指针域\*/

}headnode;

typedef headnode childlink[axnode]; /\*表头结点数组\*/

带双亲的孩子链表表示法，其类型定义与孩子链表类似，只需将头结点的定义改为：

typedef struct

{datatype data;

int parent;

link headptr;

}headnodel;

4．就图简答题4.1（a）中的二叉树

⑴根结点是A； ⑵叶结点是：D，F，J；

⑶G的双亲是：E； ⑷G的祖先是：A；

⑸G的孩子是：H； ⑹E的子孙是：F，G，H，I，J；

⑺E的兄弟是：B；C的兄弟是：C无兄弟；⑻结点B和I的层数分别是2，5；

⑼树的深度是：6； ⑽以结点G为根的子树的深度是：4；

⑾树的度数是：2。

就图简答题4.1(b)中的树

⑴根结点是A； ⑵叶结点是：D，F，G，H，I，J；

⑶G的双亲是：C； ⑷G的祖先是：A；

⑸G的孩子是：G无孩子； ⑹E的子孙是：H，I，J；

⑺E的兄弟是：D；C的兄弟是：B； ⑻结点B和I的层数分别是2，4；

⑼树的深度是：4； ⑽以结点G为根的子树的深度是：1；

⑾树的度数是：3。

5．(a)因为该图所示结构，有两个结点没有直接前趋，即有两个根结点，而树只能有一个根结点。

(b)因为找不到树的根结点，所以不满足树的定义。

(c)因为最上面一个结点的后继结点分不出两个不相交的子集，不满足树的定义。

6．答案见图简答题6所示。

7．答案见图简答题7-2所示。

8．先根序列：A B C D E F G H I J;

中根序列：B C D A F E H J I G;

后根序列：D C B F J I H G E A。

9．⑴二叉树中任意一个结点都无左孩子；

⑵二叉树中任意一个结点都无右孩子；

⑶至多只有一个结点的二叉树。

10．由后根遍历序列得到二叉树的根结点A（后根序列中最后一个结点）；在中序序列中，A的左力是A的左子树上的结点，A的右边是A的右子树上的结点；再到后根序列中找左子树和右子树的根结点，依次类推，直到画出该二叉树，如图简答题10所示。

11．答案见图简答题11-2所示。

12．先根序列：A B E F K L C G D H I J；

后根序列：E K L F B G C H I J D A；

层次序列：A B C D E F G H I J K L。

13．答案见图简答题13-2所示。

14．答案见图简答题14-2所示。

(a)(b)(c)(d)(e)

15.答案见图简答题15-1所示。

16．利用先根遍历方法查找结点\*A的直接后继：

当A->lchild<>NULL时，A的先根后继结点是A->lchild；否则，当A->rchild<>NULL时，A的先根后继结点是A->rchild。

16．利用先根遍历方法查找结点\*A的直接后继：

当A->lchild<>NULL时，A的先根后继结点是A->lchild；否则，当A->rchild<>NULL时，A的先根后继结点是A->rchild。

利用中根遍历方法查找结点\*A的直接后继：

当A->lchild<>NULL时，\*A的中根前趋是其左子树的“最右下结点”； 当A->rchild<>NULL时，\*A的中根后继是其右子树的“最左下结点”。

利用后根遍历方法查找结点\*A的直接前趋：

当A->rchild<>NULL时，A的后根前趋结点是A->rchild；否则，A->rchild<>NULL时，A的后根前趋结点是A->lchild；

17．本题的解题过程如下：

①由前根序列各A是二叉树的根结点；由中根序列知GDHBE是A的左子树中的结点，CIJF是A的右子树中的结点。

②由前根序列知B是A的左子树的根结点；由中根序列知GDH是B的左子树中的结点，E是B的右子树中的结点。

③由前根序列知D是B的左子树的根结点；由中根序列知G是B的左子树中的结点，H是B的右子树中的结点

④由前根序列知C是A的右子树的根结点；由中根序列知IJF是C的右子树中的结点（C的左子树为空）。

⑤由前根序列知F是C的右子树的根结点；由中根序列知IJ是F的左子树中的结点，（F的右子树为空）。

⑥由前根序列知I是F的左子树的根结点；由中根序列知J是F的右子树中的结点，（F的左子树为空）。

完整的二叉树如图简答题17所示。

18．第一步，先以给定的权值构造出哈夫曼树，如图简答题18所示。

第二步，假没规定哈夫曼树上所有的左指针用0表示，所有的右指针用1表示。

第三步，从根开始沿每一条通向叶子的路径上的数字，这些数字就是对应叶子结点所代表的字母的哈夫曼编码。8个字母所应的哈夫曼编码为：

7---0010 19---10

2---00000 6---0001

32---01 3---00001

21---11 10---0011

**图简答题18**

19．任意一棵二叉树只有先转换成完全二叉树后，才能用顺序存储结构进行存储。转换后的二叉树如图19-2所示。任意二叉树的顺序存储结构示意图如图19-3所示。

**图简答题19-2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C |  | D |  | E |  |  | F |  |  |  | G | …… |

**图简答题19-3**

20．转换后的二叉树如图简答题20-2所示。

**图简答题20-2**

**五、算法设计**

1．⑴bitreptr PARENT(bitreptr BT, bitreptr p, datatype data) /\*调用前p为空指针\*/

{if(BT!=NULL)if(BT->data==X) return(p); /\*找到,返回其父结点\*/

else{p=BT;

PARENT(BT->lchild, p, X); /\*查找其左子树\*/

PARENT(BT->rchild, p, X); /\*查找其右子树\*/

}

}

⑵void CREATE(datatype X, bitreptr LBT, bitreptr RBT)

{ BT=malloc(size); /\*申请根结点\*/

　　BT->data=x;BT->lchild=LBT; BT->rchild=RBT;

}

(3)void DELLEFT(bitreptr BT,datatype X)

{if (BT!=null ) if (BT->data = X) {BT->lchild=null,BT->rchild=null;}

else{DELLEFT(BT->lchild,X); DELLEFT(BT->rchild,X);}

}

2.(1) ttnodeptr PARENT(ttndoeptr BT, datatype data)

{ if (BT!=null) if (BT->data = X) return (BT->patrnt); /\*找到，返回其父结点\*/

else {PARENT (BT->lchild,X);PARENT(BT->rchild,X);}

}

(2) void CREATE(datatype X, ttnodeptr LBT, ttnodeper RBT)

{BT=malloc(size); /\*申请根结点\*/

BT->data=x; BT->lchild=LBT; BT->rchild=RBT;

LBT->parent = BT; R BT->parent = BT;

}

(3)void DELLEFT(ttnodeptr BT,datatype X)

{if (BT!=null) if (BT->data ==X){ BT->lchild =null; BT->rchild=null;}

else {DELLEFT(BT->lchild ,X); DELLEFT(BT->rchild,X);}

}

3.采用递归方法，对每一个结点先求其左右子树的深度，取较大值加1作为此结点的深度。

Int depth (bitreptr BT)

{if (BT= =null)return（0);

else {l=hepth (BT->lchild);

r=hepth(BT->rchild);

return((l>r?l:r)+1);

}

4.按照题目要求，附加一个工作栈以完成对该树的非递归遍历，思路如下：

1. 每访问一个结点，将此结点压入栈，查看此结点是否有左子树，若有，访问左子树，转（1）执行。
2. 从栈弹出一结点，如果此结点有右子树，访问右子树并转第（1）步执行，否则转第（2）步执行。

Void preorder(datatype a[n],int n )

{ INISTACK(sd); /\*初始工作栈sd\*/

I=1; PUSH(sd,0);

If (i<=n)

{visite(a[I]); /\*访问此结点\*/

PUSH(sd,I);

J=2\*I; /\* 取左子树\*/

While(!EMPTY(sd)) /\*若栈sd 非空\*/

{while(j<=n) /\*若2I<=n,则该结点有左子树\*/

{PUSH(sd,j); /\*进栈\*/

I=j; j=2\*I; /\*取左子树\*/

Visite(a[I]); /\*访问此结点\*/

}

I=pop(sd); /\*出栈\*/

J=2\*I+1; /\*取右子树\*/

}

}

}

5.本题用递归方法，采用以下步骤：

1. 若t1和t2都是空树则等价；
2. 若t1和t2有一个为空树另一个非空树，则不等价。
3. 若t1和t2两个都是非空树且它们的值相等，比较它们的左、右子树。

Int same\_tree (bitreptr t1, t2)  
 {if ((t1= =null)&&(t2= =null)) return(1); /\*t1,t2都是空树\*/

Else if ((t1= =null)||(t2= =null)) return(0); /\*t1,t2只有一个空树\*/

Else if (t1->data = =t2->data) /\*t1和t2的值相等\*/

Return (same\_tree (t1->lchild, t2->lchild)&&

Same\_tree(t1->rchild,t2->rchild));

}

6.采用后根遍历递归访问的方法，交换每一个结点的左右子树。

Void exchg\_tree(bitreptr BT)

{if (BT!=null) /\*非空\*/

{exchg\_tree(BT->lchild); /\*交换左子树所有结点指针\*/

exchg\_tree(BT->rchild); /\*交换右子树所有结点指针\*/

p=BT->lchild; /\*交换根结点左右指针\*/

BT->lchild=BT->rchild; ->rchild =p;

}

}

7.设置一个栈用于装入查找结点的所有祖先。

栈的元素结构说明如下：

typedef struct

{bitreptr p;

int tag;

}snode;

int search(bitreptr T,datatype X)

{top =0; /\*栈s初置为0\*/

while ((T!=null)&&(T->data!=X)||(top!=0))

{while (((T!=null)&&(T->data!=X))

{top ++;

s[top ].p=T; s[top].tag=0; /\*结点入栈，置标志0\*/

T=T->lchild; /\*找左子树\*/

}

if (((T!=null)&&(T->data==X)) /\*找到\*/

{for (I=1;I<=top;I++)

printf (“%d\n”,s[I].p->data); /\*输出\*/

return(1);

}

else while ((top>0)&&(s[top].tag==1)) top --; /\*退出右子树已访问过的结点\*/

if (top>0){s[top].tag=1;T=s[top]; T=T->rchild;/\*置访问标志为1，访问右子树\*/

}

}

return(0);

}

8. （1） PARENT(T,X)的功能是查找值为X的结点的双亲。查找的方法是，在表头数组中依次顺序查看每个头结点的链域，查找是否含水量有值为X的结点，如有，则返回其结点的值。

Headnode parent (headnode T,datatype X ) /\*n表示树的结点数\*／

{k=0;

if (T[k].data= =X) return(null);　　／\*根结点值是Ｘ返回空值\*／

while (k<n)

{p=T[k].headptr; /\*取子树指针\*/

while (p!=null)

{j=p->child; /\*取孩子序号\*/

if (T[j].data= =X) return(T[k]); /\*查找到X，返回父结点\*/

p=p->next; /\*取下一个孩子\*/

}

k++;

}

return(null); /\*没有找到，返回空值\*/

}

（2）HILD（T，X，I）的功能是在树T中查找值为X的结点的第I个孩子。算法思路是：先在表头结点中查找值为X的结点，然后再由此结点的链表找第I个孩子，如找到则返回这个孩子的指针值。

Headnode child(headnode T,datatype X,int I)

{k=0;

while ((k<n)&&(T[k].data!=X)) k++; /\*在头结点中查找元素\*/

if (t[k].data= =X) /\*查找到元素X\*/

{j=0; p=T[k].headptr ; /\*取X结点第一个孩子\*/

while ((p!=null )&&(j<I)){j++; p=p->next;};/\*查找第I个孩子\*/

if ((p!=null)&&(j= =I)) return(p); /\*找到第I个孩子\*/

else return(null); /\*没找到，返回定指针\*/

}

else return(null); /\*没有找到结点X\*/

}

(3) DELETE(T,X,I)的功能是删除值为X的结点的第I个孩子。算法思路是：先在表头数组中查找值为X的结点。找到后再沿此结点的链表查找第I个孩子，如存在将其删除。

Void delete (headnode T,datatype X,int I)

{k=0;

while ((k<n)&&(T[k].data!=X))k++; /\*在头结点中查找元素\*/

if (k<=n) /\*查找到结点X\*/

{j=0; p =T[k].headptr;

q=p; /\*q为p的前趋\*/

while ((p!=null)&&(j<I){j++;q=p p=p->next;} /\*查第I个孩子\*/

if ((p!null)&&(j==I)) /\*找到第I个孩子\*/

{t=p->child; q->next=p->next;free(p); /\*删除第I个孩子结点\*/

while(t<n){T[t]=T[t+1];t++;}

}

else printf(“X没有第I棵子树”)；

}

else printf(“T中不存在结点X”)；

}

9．（1）用孩子兄弟链表为存储结构，实现PARENT（T，X）运算。

本题算法的思路是：①当根结点非空进栈；②当栈非空p=pop(s)(退栈操作)，取元素p的左子树T；③当T非空且值为X，返回双亲P。否则，若T非空，让T进栈，T取它的右子树，转③执行；④转②执行。

本题用到栈S存放查找到的每一个结点，栈中元素结构说明如下：

typedef struct {bitreptr p;}s[max\_size];

bitreptr PARENT(bitreptr T,datatype X)

{

if (t!=null)

{top =1; s[top].p=T; /\*若头结点非空，进栈\*/

while ((T!=null)||(top>0))

{p=s[top].p;topj--; T=p->rchild；/\*退栈\*/

while ((T->data!=X)&&(T->rchild!=null))

{top++;s[top].p=T; T=T->rchild;}

/\*若此结点值不为X，查找其兄弟域\*/

if ((T!=null)&&(T->data= =X))return(p) ;/\*找到返回其双亲结点\*/

}

}

return(null);

}

(2)用孩子兄弟链表为存储结构，实现CHILD（T，X，I）运算。

算法CHILD（T，X，I）的功能是查找结点X的第I个孩子。本算法在查找结点X时，同（1）基本思路相仿，只在找到结点X时，再查找它是否有第I个孩子。

Bitreptr CHILD(bitreptr T,datatype X,int I ) /\*用的栈S同（1）\*/

{if (T!=null)

{top =1; s[top].p =T; /\*若头结点非空，进栈\*/

while ((T!=null )||(top >0))

{whhile ((T!=null )&&(T->data==X))

{top ++; s[top].p=T; T=T->lchild } /\*若此结点值不为X，查找其左子树\*/

if ((T!=null)&&(T->data= =X))

{j=0; p=T->lchild ; /\*找到结点X，查找第I个孩子\*/

while ((p!=null)&&(j<I)){j++; p=p->rchild;}

if (j= =I) return(p); /\*找到返回结点\*p\*/

else return(null); /\*没有找到返回空指针\*/

}

while ((top>0)&&(s[top].p->rchild= =null)) top--;

if (top>0){T=s[top].p->rchild ;top--;}

}

}return(null);

}

(3) 用孩子兄弟链表为存储结构，现实DELETE（T，X，I）的运算。

Void DELETE(bitreptr T,datatype X,int I)

{if (T!=null)

{top=1; s[top].p=T; /\*若头结点非空，进栈\*/

while((T!=null)||(top>0))

{while ((T!=null)&&(T->data!=X)){top++;s[top].p=T; T=T->lchild;}

if (t->data= =X) /\*找到结点\*/

{j=o; p=T->lchild; /\*查找第I个孩子\*/

while ((p!=null)&&(j<I)){j++; T=p;p=p->rchild;}

if ((p!=null)&&(j= =I)

{T->rchild =p->rchild ;free(p);} /\*找到第I个孩子删除\*/

else printf(“ 没有找到第I个孩子”)；

return;

}

while((top>0) &&(T->rchild!=null))top--;

if (top>0){T=s[top].rchild;top--;}

}

printf(“没有找到结点X返回”)；

}

}

（4）.用静态双亲链表为存储结构，实现PARENT（T，X）运算。

静态双亲链表的类型定义如下：

#define size

typedef struct

{ datatype data; /\*数据域\*/

int parent; /\*双亲域（静态指针域）\*/

}node;

typedef node T[size]; /\*静态双亲链表\*/

node PARENT(node T[],datatype X) /\*n表示树的结点数\*/

{k=0;

while (k<n)

if (T[k].data= =X) if (T[k.parent ]>=0) return (null);

else {printf(“X在根结点”); return(null);}

else k++;

printf (“找不到Z结点”!); return(null);

}

(5)用静态双亲链表为存储结构，实现CHILD（T，X，I）运算。

算法CHILD（T，X，I）的功能是查找结点X的第I个孩子。本算法在查找结点X时，同（4）基本思路相仿，只在找到结点X时，再查找它是否有第I个孩子。

Node CHILD（node T[]，datatypeX，int I）

{k=0;

while (k<n)

if (T[k].data= =X)

{j=0;h=k;

while (++k<n)

if (T[k].parent= =h) if ((j= =I) return(T[k]);

else j++;

printf (“找不到X结点的第I个孩子！”); return(null);

}else k++;

printf(“找不到X结点！”);return (null);

}

(6)用静态双亲链表为存储结构，现实DELETE（T，X，I）运算。

Int DELETE（bitreptr T，datatypeX，int I）

{k=0;

while (k<n)

if (T[k].data= =X)

{j=0;h=k;

while (++k<n)

if (T[k].parent= =h)

If ((j= =I)

{while (k<(n-1)){T[k]=T[k+1]; k++;}

free(T(n-1)); return(!);}

else j++;

printf (“找不到X结点的第I个孩子！”); return(0);

}else k++;

printf(“找不到X结点！”);return (0);

}

10.(1) 需要设置一个栈，用于存放查找到的每一个结点。本题的思路是：当前的根

结点不空时，查找最左下的孩子，并将查找过的结点依次进栈，找到最左下孩子后，

输出该结点，然后转其右子树上，重复前面查找过程。

本题用到栈S存放查长到的每一个结点，栈中元素结构说明如下：

typedef struct{bitreptr p;}s[max\_size];

void inorder1(bitreptr T)

{ initstack(s);

while ((T!=null )||(not enpty(s)))

{while(T!=null){push(s,T);T:=T->lchild;} /\*向左下找下一个结点\*/

if (not empty(s))

{T=pop(s); /\*退出下一结点\*/

printf(“%datatype”,T->data); /\*访问\*/

T =T->rchild; /\*遍历右子树\*/

}

}

}

(2) 本题需用到栈，因此解法同（1）。

Viod inorder2(ttnodeptr T) /\*栈同（1）所设\*/

{initstack(s);

while ((T!=null)||(not empty(s)))

{while (t!=null){push(s,T);T:=T->lchild;} /\*向左下找下一个结点\*/

if (not empty(s))

{T=pop(s); /\*退出下一结点\*/

printf(“%datatype”,T->data); /\*访问\*/

T=T->rchild; /\*遍历右子树\*/

}

}

}

11.(1)本题用到栈的结构说明如下：

typedef struct

{ bitreptr p; //tag的说明：tag=0表示访问过左子树；

int tag; tag=1表示访问过右子树。

}snode;

因为本题要求对二叉树的后根序非递归启遍历，解决此问题的算法思路是：

1. 对任一非空结点，让此结点的标志tag=0,并进栈，到其左子树，转①执行。
2. 若左子树为空且栈非空，让栈顶点元素的标志tag=1，然后到其右子树，

转①执行。

1. 若栈非空且顶点的访问标志tag=1,访问此元素，退栈，转③执行；否则，栈非空，

转②执行。

Void postorderl(bitreptr T)

{top=0; /\*栈S初值为空栈\*/

while((T!=null)||(top!=0))

{while(T!=null)

{top++;s[top].p=T; /\*结点及访问标志入栈\*/

s[top].tag=0;T=T->lchild;} /\*找下一个结点\*/

while((top>0)&&(s[top].tag= =1)) /\*输出右子树已遍历结点\*/

{T=T[top].p;top--; /\*取一个结点并退栈\*/

printf(“%datatype”,T->data);} /\*访问\*/

if (top>0){s[top].tag=1;T=T->rchild;} /\*置访问右子树并赋指针\*/

}

}

⑵本题所用三叉结点结构如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mark | data | lchild | parent | rchild |

其中：mark=0表示已访问过该结点的左子树；

mark=1表示已访问过该结点的右子树；

mark=2表示已访问过该结点

void postorder2 ( ttnoderptr T) /\*T为三叉链表的二叉树 \*/

{ p=NULL; /\*p为T的双亲结点 \*/

do{

while(T!=NULL) /\*遍历左子树 \*/

{T->mark=0;p=T;T=T->lchild; /\*置访问标志，取左子树 \*/

T=p; /\*取其双亲结点 \*/

while((T!=NULL)&&(T->mark==1)) /\*访问所有右子树已访问结点 \*/

{ printf(“datatype”, T->data); /\*访问 \*/

T->mark=2;T=T->parent; /\*置已访问标志并取其双亲结点\*/

if(T==NULL) return; /\*根结点被访问，结束 \*/

else if(T->mark==0) /\*遍历其右子树 \*/

{T->mark=1;p=T;T=T->rchild;} /\*置标志，取右子树 \*/

}while(1); /\*继续循环 \*/

}

12．此问题所构造的一棵比较次数最小的判定树见图算法设计12．。

void tran(float score)

{if(score<=70)

if(score<=80) grade=’C’; /\*70~79 \*/

else if(score<=90) grade=’B’; /\*80~89 \*/

else grade=’A’; /\*90~100 \*/

else if(score>=60) grade=’D’; /\*60~69 \*/

else grade=’E’; /\*0~59 \*/

}

**图算法设计题12**

13．void preorder(bitreptr T)

{if (t!=NULL)

{printf(“%f”, T->data);

preorder(T->rchild);

preorder(T->lchild);

}

}

14．本算法要借用队列来完成，其基本思想是，只要队列不为空，就出队列，然后判断该结点是否有左孩子和右孩子，如有就依次输出左、右孩子的值，然后让左、右孩子进队列。

void layorder (bitreptr T)

{initqueue (q) /\*队列初始化\*/

if(T!=NULL)

{printf(“%f”, T->data);

enqueue (q, T); /\*入队列\*/

while (not emptyqueue (q) ) /\*若队列非空\*/

{outqueue (q, p) ; /\*出队\*/

if (p->lchild!=NULL)

{printf(“%f”, p->lchild->data);

enqueue (q, p->lchild); /\*入队列\*/

}

if (p->rchild!=NULL)

{printf(“%”, p->rchild->data);

enqueue (q, p->rchild); /\*入队列\*/

}

}

}

}

15．本算法的基本思想是：先求左子树的叶子数，再求右子树的叶子数，两都相加就是根结点的叶子数，也就是对应二叉树的叶子数。

int leafcount (bitreptr T) /\*求二叉树 T的叶子数\*/

{ if(T==NULL)leaf=0; /\*当二叉树为空时，叶子数等于0\*/

else if(T->lchild==NULL)&&(T->rchild==NULL)leaf=1

/\*当二叉树仅含一个根结点时，叶子数为1\*/

else{L=leafcount(T->lchild); /\*求左子树的叶子数\*/ （1）

R=leafcount(T->lchild); /\*求右子树的叶子数\*/ （2）

leaf=L+R; /\*左、右子树叶子数之和等于二叉树的叶子数\*/ （3）

}

return(leaf);

}

1. 参考答案

四、简答及应用

　1．作为静态查找表存储结构的顺序表的类型定义如下：

#define maxsize 静态查找表的表长 ;

typedef struct

{ keytype key ; /\* 关键字 \*/

…… /\* 其他域 \*/

} rec ;

typedef struct

{ rec item [ maxsize + 1 ] ;

int n ; /\* 最后一个数据元素的下标 \*/

} sqtable ;

静态查找表中的数据元素存放在上述数组的第1到第 n 个单元中，第 n+1　到最后一个单元为备用区，0或 n+1 单元被用于设置“岗哨”，以便简化查找运算的实现。

2．一棵二叉排序树（又称二叉查找树）或者是一棵空树，或者是一棵同时满足下列条件的二叉树：

1. 若它的左子树不空，则左子树上所有结点的键值均小于它根结点键值。;
2. 若它的右子树不空，则右子树上所有结点的键值均大于它根结点键值。;
3. 它的左、右子树也分别为二叉排序树。

3．开散列表的组织方式如下：设选定的散列函数为 H ，H的值域（即散列地址的集合）为0．．n – 1。设置一个“地址向量”pointer HP[ n ],其中的每个指针HP[ i ]指向个单链表用于存储所有散列地址为i 的数据元素,即所有散列地址为ii的同义词。每一个样的单链表称为一个同义词子表。由地址向量以及向量中的每个指针所指的同义词子表构成存储结构称为开散列表。这种开散列解决冲突的方法又称“拉链法”。开散列表类型定义如下：

typedef struct tagnode

{ keytype key ;

struct tagnode \*next ;

．．．．．．　　　　　　　　　　　　/\*其他其他域\*/

}\*pinter,nod ;

tyedef poiner openhah[n] ;

4．闭散列表是一个一维数组，其元素的类型与动态查找表中数据元素的类型一致：

　　#define maxsize　　闭散列表的容量

typedef struct

{ keytype key ;

．．．．．． /\*其他域\*/

} element ;

typedef element closehash[ maxsize ] ;

5．闭散列表是一个大小固定的一维数组，解决冲突的基本思想是在需要时为每个键值K生成一个散列地址序列d0，d1，…，di，…，dm-1。其中d0=H(K)是K的散列地址；所有di(0<i<m)是后继散列地址。当插入K时，若位置d0=H(K)上的结点已被别的数据元素占用，则按上述地址序列依次探测，将找到的第一个空闲位置di作为K的存储位置。若所有后继散列地址都不空闲，说明该闭散列表已满（溢出）。相应地，查找或删除K时将按同样的后继地址序列依次查找。查找成功时回送该位置或删除该位置上的数据元素（实际上是对该结点加以标记）；查找不成功时回送一个特殊标志。

6.对地址单元d = H ( K ) ,如发生冲突，以d为中心在左右两边交替进行探测。按照二次探测法，键值K的散列地址序列为：

d0 = H ( K ) ,

d1 = (d0 + 12 ) mod m ,

d2 = (d0-12 ) mod m ,

d3 = (d0 + 22 ) mod m ,

d4 = (d0 - 22 ) mod m ,

……

7．此法要求设立多个散列函数Hi, i= 1 , …, k 。当给定值K与闭散列表中的某个键值是相对于某个散列函数Hi的同义词因而发生冲突时，继续计算该给定值K在下一个散列函数Hi+1下的散列地址，直到不再产生冲突为止。

8．散列表由两个一维数组组成。一个称为基本表，另一个称为溢出表。插入首先在基本表上进行；假如发生冲突，则将同义词存入溢出表。

9．假设长度为20的有序序列为（ a1 , a2 , … , a20 ）,按二分查找法得到的判定树如图简答题9所示。

成功平均查找长度为：（ 1 + 2 \* 2 + 3 \* 4 + 4 \* 8 + 5 \* 5 ）/ 20= 74 /20 = 3.7

10．

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

①006 087 155 188 220 465 505 508 511 586 656 670 700 766 897 908

↑ ↑ ↑

low mid hig

②006 087 155 188 220 465 505 508 511 586 656 670 700 766 897 908

↑ ↑ ↑

low mid hig

③006 087 155 188 220 465 505 508 511 586 656 670 700 766 897 908

↑ ↑ ↑

low mid hig

（成功）

11．分析：平衡二叉树AVL要求二叉树上任何一个结点的平衡因子为-1，0，1，如有一个元素的平衡因子不是-1，0，1，此二叉树就不是平衡二叉树，必须通过调整把此二叉树变成平衡二叉树。平衡二叉树是二叉树排序树（对结点数结点相同的二叉树）平均查找次数最小的一种二叉树。

答案见图简答题 11（1）（2）所示。成功平均查找长度为：ASL=( 1 +2 \* 2 + 3 \* 4 + 4 \* 4 ) / 11 = 33 /11 = 3 .

12．分析：开散列表上查找失败的平均查找次数的计算方法，是对每一种情况（即一个下标）计算查找失败次数的总和除以表的长度，即（3 + 1 + 2 + 2 + 1 + 4 + 3 + 3 + 1 + 2 + 1 +1 + 1 + 1 ）/ 14 ≈2。

闭散列表查找失败的平均查找次数的计算方法，是对每一种情况（即每一个下标）计算查找失败次数的总和除以表的长度，即（5+4+3+2+1+9+8+7+6+5+4+3+2+1）/14 = 60/14≈4。

开散列表见图简答题12-1。

闭散列表见图简答题12-2。

0 1 2 3 4 5 6 7 8 9 10 11 12 13

Apr | Aug | Dec | Feb | | Jan |Mar | May | Jun | Jul | Sep | Oct | Nov |

1 2 1 1 1 1 2 4 5 2 5 6

图简答题12-2闭散列表

开散列表上查找成功的平均查找长度：

ASL =（1+2+1+1+1+2+3+1+2+1+2+1）/12 = 18/12 = 1.5

开散列表上查找不成功的平均查找长度：

ASL =（3+1+2+2+1+4+3+3+1+2+1+1+1+1）/14 = 26/14≈2

闭散列表上查找成功的平均查找长度：

ASL =（1+2+1+1+1+1+2+4+5+2+5+6）/12 = 31/12≈2.6

闭散列表上查找不成功的平均查找长度：

ASL =（5+4+3+2+1+9+8+7+6+5+4+3+2+1）/14 = 60/14≈4

13．开散列表见图简答题13。

查找成功的平均查找长度为：（4\*2+8）/ 12 = 4/3。

14．衡量算法的标准有很多，时间复杂度只是其中之一。尽管有些算法时间性能很好，但是其他方面可能就存在着不足。比如散列查找的时间性能很优越，但是需要关注如何合理地构造散列函数问题，而且总存在着冲突等现象，为了解决冲突，还得采用其他方法。

二分查找也是有代价的，因为事先必须对整个查找区间进行排序，而排序也是费时的，所以常应用于频繁查找的场合。对于顺序查找，尽管效率不高，但却比较简单，常用于查找范围较小或偶而进行查找的情况。

五、算法设计

1．分析：将岗哨设置在高下标端，表示从线性表低端查找时，在不成功的情况下，算法自动在岗哨处终止。另外，在等概率前提下，查找成功的平均查找长度等于每一个元素查找次数之和除以结点总数。

Int sqsearch0 (sqtable A,keytype X) /\*数组有元素n个\*/

{ i=1; A.item[n+1].key=X; /\*设置哨兵\*/

while (A.item[n+1].key!=X) i++;

return (i% (n+1)); /\*找不到返回0，找到返回其下标\*/

}

查找成功平均查找长度为：（1+2+3+…+n）/ n =（1+n）/ 2。

查找不成功平均查找长度为：n+1。

2．（1）int sqsearch1 (sqtable A,keytype X) /\*数组有元素n个\*/

{ i=1; a.item[n+1].key!=X; /\*设置哨兵\*/

while (A.item[n+1].key!=X) i++;

if ( (i!=1)&&(i<n+1))

{ temp=A.item[i].key;

A.item[i].key=A.item[i-1].key; /\*结点i-1和i交换\*/

A.item[i-1].key=temp;

i--;

}

return (i% (n+1));

}

（2）lklist lksearch2 (lklist head,dataytpe X)

{ p=head->next;

q=head;

r=NULL; /\*r是q的前趋，q是p的前趋\*/

while ((p!=NULLl)&&(p->data!=X)) {r=q;q=p;p=p->next;} /\*搜索结点X\*/

if ((p!=head->next)&&(p!=NULL))

{q->next=p->next;p->next=q;r->next=p;} /\*交换q，p结点顺序\*/

return (p);

3．分析：在等查区间的上、下界处设两个指针，由此计算出中间元素的序号，当中间元素大于给定值X时，接下来到其低端区间去查找；当中间元素小于给定值X时，接下来到其高端区间去查找；当中间元素等于给定值X时，表示查找成功，输出其序号。

Int binlist (sqtable A,int s,t,keytype X) /\*t、s分别为查找区间的上、下界\*/

{ if(s<t) return(0); /\*查找失败\*/

else

{ mid=(s+t)/2;

switch(mid)

{ case x<A.item[mid].key: return(binlist(A,s,mid-1,X)); /\*在低端区间上递归\*/

case x==A.item[mid].key: return(mid); /\*查找成功\*/

case x>A.item[mid].key: return(a,mid+1,t,X)); /\*在高端区间上递归\*/

}

}

}

4．分析：采用递归方法，从根结点开始查找结点p，若查询结点是所要找的结点，返回其深度h0。否则，到左、右子树上去找，查找深度加1。

int find1 ( birtreptr T , p , int h0 )

/\*在二叉树排序树T中查找结点p的层次，若不在时返回空值。h0为根结点T的层次\*/

{ if ( p == NULL ) return ( 0 ) ; /\* 没找到，返回0 \*/

if( p ==T) return( h0 ) ; /\* 找到 \*/

else if (p -> data < T -> data ) return( find1( T ->lchild,p,h0) +1) ; /\* 到左子树去找 \*/

else return ( find1 ( T -> rchild,p,h0) + 1); /\* 到右子树去找 \*/

}

int find2( birtrptr T, p ) { return ( find1 ( T, p,1 ) ) ; }

5．分析：根据题目要求分下面情况进行讨论：

1. 若A为根结点，则A为公共祖先；
2. 若A->data<T->tata且T->data<B->data,T为公共祖先；
3. 若A->data<T->data且B->data<T->data或A->data>T->data且B->data>T->data,则到T的左右子树去查找。

bitreptr ANS (bitreptr T,A,B) /\*T为二叉排序树的根结点\*/

{ if (T==NULL) return (NULL); /\*不存在\*/

else if ((A->data<T-V>data)&&(T->data<B->data)|| /\*T为公共祖先\*/

(A->data==T->data)) return (T);

else if ((A->data>T->data) return (ANS(T->rchild,A,B)); /\*到右子树上查找\*/

else return (ANS(T->lchild,A,B)); /\*到左子树上查找\*/

}

6．分析：对二叉排序树来讲，其中根遍历序列为一个递增有序序列。因此，对给定的二叉树进行中根遍历，如果始终能保证前一个值比后一个值小，则说明该二叉树是二叉排序树。

int bsbtr (bitreptr T) /\*predt记录当前结点前趋值，初值为-∞\*/

{ if (T==NULL) return(1);

else {b1=bsbtr(T->lchild); /\*判断左子树\*/

if (!b1 || (predt>=T->data)) return(0); /\*当前结点和前趋比较\*/

predt=T->data; /\*修改当前结点的前趋值\*/

return(bsbtr(T->rchild)); /\*判断右子树并返回最终结果\*/

}

}

7．分析：在二叉排序树上删除指定结点p，首先要找到p的双亲结点Pre。删除结点p又分三种情况（下面仅以p为其双亲左孩子举例）：

1. 若p结点为叶子结点，只要令pre->lchild=null即可。
2. 若p结点只有一棵子树树，令其子树代替结点，即pre->lchild=p->lchild或Pre->lchild=p->rchild。
3. 若结点p的左子树和右子树均不空，做法是p的直接前趋（或直接后继）替代p，然后再从二叉排序树中删除它的直接前趋（或直接后继）。  
   bitreptr srch\_bstree (bitreptr pre, bitreptr bst, keytype k, int found)  
   /\*pre为bst结点的双亲，k为删除结点的关键字，found为查找标志，若找到关键字为k的结点，则置found值为1，找不到found值为0\*/  
    { if (bst==NULL) {found=0; return(p); }  
    else if (bst->key==k) {found=1; return(pre); }  
    else if (k<bst->key) {pre=bst; return(srch\_bstre(pre,bst->lchild,k,found)); };  
    else {pre=bst;return(srch\_bstree(pre,bst->rchild,k,found)) };  
    }  
    void del\_bstree(bitreptr pre,bst,p)  
    /\*bst为二叉树根结点，p为删除结点，pre为p的双亲结点，若p为根结点，则Pre为Null\*/  
   { k=p->key; found=0;  
    pre=srch\_bstree( pre,bst,k,found)  
    if (found)  
    switch{  
    case (p->lchild==NULL)&&(p->rchild==NULL): /\*p为叶结点\*/  
    pre->lchild=NULL; free(p); break;  
    case (p->lchild==NULL)&&(p->rchild!=NULL): /\*p有右孩子\*/  
    pre->lchild=p->rchild; free(p); break;  
    case (p->lchild!=NULL)&&(p->rchild==NULL): /\*p有左孩子\*/  
    pre->lchild=p->lchild; free(p); break;  
    case (p->lchild!=NULL)&&(p->rchild!=NULL): /\*p有左右孩子\*/  
    { q=p; s=p->lchild;  
    while (s->rchild!=NULL) {q=s;s=s->rchild;} /\*找p的直接前趋\*/  
    p->data=s->data /\*p直接前趋的值传给p\*/  
    if (q!=p) q->rchild=s->lchild; /\*删除p直接前趋结点\*/  
    else q->lchild=s->lchild; /\*q=p时，即为p的左孩子\*/  
    free (s);  
    }  
    } /\*end switch\*/  
    else {/\*p为双亲右孩子，类似p为双亲左孩子，略。\*/}  
   else printf (“没有找到结点P，删除结点有错误。”);

}

8．分析：显然，AVL树的插入算法应在二叉排序树插入算法的基础上扩充以下功能：①判断插入结点之后是否平衡；②若是，寻找最小失衡子树并转③；③判断失衡类型并做相应调整。

失衡的判断可以与寻找最小失衡子树结合起来，最小失衡子树的根结点一定是离插入结点最近，且插入之前平衡因子的绝对值为1的结点。AVL树插入算法的基本步骤如下：

1. 在寻找新结点的插入位置的过程中，记下离该位置最近，且平衡因子不等于零的结点a（此结点即为可能出现的最小失衡树的根）；
2. 修改自该结点至插入位置路径上所有结点的平衡因子（注意：树上其他结点的平衡因子均不受插入的影响）；
3. 判断实施插入操作之后，结a点的平衡因子的绝对值是否大于1（即判断是否失衡）。若是，进一步判断失衡类型并做相应调整；否则插入过程结束。

仍以二叉链表作为AVL树的存储结构。但每个结点需增加一个域用于存储平衡因子。类型定义如下：

typedef struct taganode

{ keytype key; /\*键值\*/

int bf; /\*平衡因子域\*/

struct taganode \*lchild, \*rchild; /\*孩子指针域\*/

…… /\*其他域\*/

} anode, \*avlpt;

在此存储结构上，AVL树的插入算法如下：

avlpt insert\_avltree (keytype K, avlpt T)

{s=malloc(size); s->key=K; s->lchild=NULL; s->rchild=NULL;

/\*生成以K为键值的新结点\*/

if (T==NULL) return(s); /\*查找s的插入位置，并记录a\*/

else

{ f=NULL; a=T; q=NULL;

while (p!=NULL)

{ if (p->bf!=0) {a=p; f=q}

/\*a记录bf!=0的结点，其终值指向离插入位置最近的bf!=0的结点\*/

q=p;

if (s->key<p->key) q->lchild=s;

else p=p->rchild;

} /\*与while对应\*/

if (s->key<q->key) q->lchild=s;

else q->rchild=s; /\*插入s\*/

if(s->key<a->key){p=a->lchild; b=p; d=1;}/\*s插入在a的左子树上，增量d为1\*/

else {p=a->rchild; b=p; d=-1;} /\*s插入在a的右子树上，增量d为-1\*/

while (p!=s) /\*修改自a的孩子到s路径上结点的平衡因子\*/

if (s->key<p->key) /\*若s在p的左子树上\*/

{p->bf=1; p=p->lchild;}/\*p的平衡因子加1。原来为0，因为p是a的子孙\*/

else {p->bf=-1; P=P->rchild;} /\*p的平衡因子减1，原来为0\*/

switch /\*判断是否失衡并分类调整\*/

{ case a->bf==0: a->bf=d; break;

/\*找插入位置过程中未遇到bf!=0的结点，a指树根\*/

case a->bf+d==0: a->bf=0; break; /\*插入不导致以a为根的子树失衡\*/

default: /\*其他情况均失衡，判别失衡类型并调整\*/

{ if(d==-1) switch

{ case b->bf==1: LL\_rotation; break; /\*LL调整\*/

case b->bf==-1: LR\_rotation; break;

/\*LR调整，结束时令b指向新子树的根\*/

}

else switch

{ case b->bf==-1: RR\_rotation; break;

case b->bf==1: RL\_rotation; break; /\*结束时令B指向新子树的根\*/

}

switch /\*将新子树链接到原a双亲f上\*/

{ case f==NULL: return(b); /\*原a为树根\*/

case f->lchild==a: f->lchild=b; return(T);

case f->rchild==a: f->rchild=b; return(T);

}

}

}

}

第七章 参考答案

四、简答及应用

1.用邻接矩阵表示法来表示一个具有n个顶点的图时，除了用邻接矩阵中的n×n个元素存储顶点问相邻关系外，往往还需要另设一个数组存储n个顶点的信息。类型定义如下：

＃define vnum 20

typedef struct graph

{ VertexType vexs[ vnum ] ; /\* 顶点信息 \*/

int arcs[vnum ] [vnum ] ; /\* 邻接矩阵 \*/

int vexnum , arcnum ; /\* 顶点数，弧（边）数 \*/

} GraphTp ;

若图中每个顶点只含有一个编号( 1 <= i <= vnum ),则只需一个二维数组表示图的邻接矩阵。此时存储结构可简单说明如下：

# define vnum 20

typedef int GraphTp [vnum] [vnum ] ;

2．单链表中每一个结点称为表结点，应包括两个域：邻接点域，用以存放与vi相邻接的顶点序号；链域，用以指向同vi邻接的下一个结点。另外，每一个单链表设一个表头结点。每一个表头结点有两个域，一个用来存放顶点vi的信息；另一个域用来指向邻接表中的第一个结点。为了便于管理和随机访问任一顶点的单链表，将所有单链表的头结点组织成一个一维数组。邻接表的类型定义如下：

#define vnum 顶点个数

typedef struct arcnode

{ int adjvex ; /\* 下一条弧（边）的始点编号 \*/

struct arcnode nextarc; **/**\* 指向下一条弧（边）的指针 \*/

} ArcNodeTp ;

typedef struct vexnode

{ int vertex; /\* 顶点编号 \*/

AarcNodeTp firstarc ;  **/**\* 指向第一条弧（边）的指针 \*/

}AdjList[vnum] ;

typedef struct graph

{ AdjList adjlist ;

int vexnum ,arcnum ; /\* 顶点和弧（边）的个数 \*/

}GraphTp ;

3．g1的邻接矩阵如下：

g1的邻接表如图简答题3-1所示。

4．Gg2的邻接矩阵如下：

g2的邻接表如图简答题4-1所示。

5．深度优先搜索序列 V5 V4  V3 V2 V1.

广度优先搜索序列 V5 V4  V3  V2 V1.

图的深度优先搜索和广度优先搜索的顶点序列不是惟一的。

6．答案见图简答题6-2

7．答案见图简答题 7-2

8．图简答题8-1的拓扑排序序列：V3 ,V1, V4, V5, V2,V6.

9.网图简答题9-1的邻接矩阵如下：

10.答案见图简答题10-2

11.⑴有11条弧；

⑵顶点i,j之间有弧的条件是A[ i,j ] <> 0 ;

　⑶顶点i的出度是以顶点ii为起点的弧的个数或A[i,k] <> 0的个数(k=0,1,2,…，9)。

12.⑴图简答题11-1每个顶点的入、出度

顶点 入度 出度

V1 3 0

V2 2 2

V3 1 2

V4 1 3

V5 2 1

V6 2 3

⑵邻接矩阵如下：

⑶邻接表和逆邻接表见图简答题12-1

⑷逆邻接表如下：

13．根据图简答题13-1进行拓扑排序，得出5个拓扑序列为：

V1, V4, V2, V7, V9, V3, V3, V8, V6

V1, V9, V3, V4, V9, V2, V7, V5, V6

V1, V4, V2, V9, V3, V7, V5, V8, V6

V9, V3, V1, V4, V2, V7, V5, V8, V6

V9, V1, V4, V2, V3, V7, V5, V8, V6

14．⑴邻接矩阵非零元素个数的总和除以2.

⑵当A[ i,j ] <> 0或A[ i,j ] <> 0时，表示两顶点i,j之间有边相连。

⑶计算邻接矩阵上任一行非零元素的个数。

15．⑴深度优先搜索顶点序列：V1, V2, V3, V5, V4 .

⑵广度优先搜索顶点序列：V1, V2, V5, V4, V3 .

⑶由深度优先搜索得到的一棵生成树如图简答题15-2所示。

⑷由广度优先搜索得到的一棵生成树如图简答题15-2所示。

16．⑴按题中顺序输入各条边后建立起来的邻接表如图简答题16所示。

⑵在邻接表上进行深度优先遍历所得到的DFS序列为4，5，3，1，6，2；在邻接表上进行广度优先遍历所得到的BFS序列为4，5，3，6，1，2.

17．假设无向连接通网络为G=( V, E ) ,而构造出的最小生成树为T(U,TE)。初始时，U中只有一个顶点（即出发点），而TE=￠。应用prim算法的操作步骤是：

1. 将出发点用实线画出（表示出发点属于U集合）；
2. 查找所有虚线顶点连接到实线顶点上的边的权值（若其顶点到出发点无边，则认为它们之间有一条权值为∞的边），找出权值最小的那条边；
3. 将找出的权值最小的边改为用实线画出（表示将该条边加入TE中），将该条边的另一顶点也用实线画出（表示将该顶点加入到U中）；
4. 调整所有虚线顶点到实线顶点边上的权值，原则是用权值较小的边取代权值较大的边；
5. 如果V<>U，重复执行第②步，否则退出。

按照上述操作步骤，最小生成树的构造过程如图简答题17-2所示。

五、算法设计

1．本题算法思路是：先设置一个空的邻接表，然后在邻接矩阵上查找值不为空的元素，找到后在邻表的对应单链表中插入相应的边的表结点。

void mattolist ( int a[][], AdjList b[], int n) /\*n为图的结点个数\*/

{ for (i=0; i<n; i++) b[i].firstarc=NULL; /\*邻接表置空\*/

for (i=0; i<n; i++) /\*逐行进行\*/

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

if (a[i][j]!=0)

{ p=(ArcNodeTp\*) molloe (sizeof (ArcNodeTp)); /\*产生邻接点\*/

p->adjvex=j; /\*插入到表头\*/

p->nextare=b[i].firstare;

b[i].firstarc=p;

}

}

2．本题的算法思路是：先建一个空的邻接矩阵，然后在邻接表上顺序地取每个单链表中的表结点，如果表结点不为空，则将邻接矩阵中对应单元的值置为1。

void list\_to\_mat (AdjList b[], int n, int a[][])

{ for (i=0; i<n; i++) for (j=0; j<n; j++) a[i][j]=0;

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

{ p=b[i].firstarc;

while (p!=NULL) {a[i, p->adjvex]=1; p=p->nextare;}

}

}

3．①连通图的深度优先搜索算法

本题的算法思路是：先访问并标记出发点v，然后在邻接矩阵中与v对应的行查找v的邻接点，找到后，再以该邻接点为新的出发点，进行深度优先搜索遍历。

void dfs (int a[][], int v, int n) /\*n为图的结点个数，visited数组初值为0\*/

{ printf (“%d”, v); visited[v]=1 /\*访问项点v\*/

w=1;

while ((w<=n)&&(a[v][w]==0)) w=w+1; /\*找顶点v的第一个邻接点\*/

while (w<=n)

{ if (!visited[w]) dfs(a, w, n); w++; } /\*找顶点v处于w之后的下一个邻接点\*/

}

②连通图的广度优先搜索算法

本题借助一个队列Q来存放已访问过的顶点。基本算法思路是，首先输出出发点，并标记已访问过，让其进入队列。接下来只要队列非空，则出队列，然后在与顶点对应的行上查找其邻接点。如果该邻接点未被访问过，则输出该顶点并让它进入队列。重复上述步骤直到队列为空队列为止。

void bfs (int a[][], int v, int n) /\*n为图结点个数，visited数组初值为0\*/

{ InitQueue(Q); /\*置队列Q为空队列\*/

printf(“%d”, v); visited[v]=1; /\*访问顶点\*/

EnQueue (Q, v); /\*v入队列\*/

while (!EmptyQueue (Q))

{ v=OutQueue (Q); /\*出队\*/

w=1;

while (w<=n) /\*找顶点v的邻接点\*/

{ if (!visited[w])

{ printf (“%d”, w); visited[w]=1; /\*访问顶点w\*/

EnQueue (Q, w); /\*入队\*/

}

w++; /\*w后退\*/

} /\*end if\*/

} /\*end while\*/

}

4．在有向图中，若邻接表中顶点Vi有邻接点Vj，在逆邻接表中顶点Vj一定有邻接点Vi，由此得出本题的算法思路是：首先，将逆邻接表表头结点的firstarc域置空，然后逐行将表头结点的邻接点进行转化。

Void list (AdjList b1[], AdjList b2[], int n) /\*n为图的结点个数\*/

{ for (i=0; i<n; i++) b2[i].firstarc=NULL; /\*逐行进行转换\*/

for (i=0; i<n; i++) /\*指向第一个邻接点\*/

{ p1=b1[i].firstarc;

while (p1!=NULL)

{ j=p1->adjvex; /\*j为邻接点\*/

p2=(ArcNodeTp \*) molloe (sizeof (ArcNodeTp)); /\*产生逆邻接点\*/

p2->adjvex=i;

p2->nextarc=b2[j].firstarc; /\*插入到表头\*/

b2[j].firstarc=p2;

p1=p1->nextarc; /\*指向下一个邻接点\*/

}

}

}

5．（1）在邻接矩阵上，一行对应于一个顶点，而且每行的非零元素的个数等于对应顶点的出度。因此，当某行非零元素的个数为零时，则对应顶点的出度为零。据此，从第一行开始，查找每行是否有非零元素，如果没有则计数器加1。

Void sum\_zero1 (int a[][],int n, int count) /\*n为结点个数，count计度数为0的结点数 \*/

{ for (I=0;I<n;I++)

{ tag=0; /\*tag为标志\*/

for (j=0; j<n; j++) if (a[I][j]>=1) tag=1; /\*有边\*/

if (tag==0) count++; /\*I出度为0\*/

}

}

(2)邻接表结构中的边表恰好就是出边表。因此，其表头数组中firstarc域为空的个数等于出度为零的元素个数。

Void sum\_zero2 (AdjList a[], int count) /\* count的初值为0，a为有向图的邻接表\*/

{ for (I=0; I<n; I++)

if (a[I].firstarc==NULL) count++;

}

6．在邻接表的表头中增加一个入度域in，数据结构修改如下：

typedef struct vexnode

{ VertexType vertex;

int in; /\*增加一个入度域\*/

ArcNodeTp \*firstarc;

} AdjList[vnum];

typedef struct graph

{ AdjList adjlist;

int vexnum, arcnum;

} GraphTp;

拓扑排序算法如下：

Top\_Sort (GraphTp g)

{ LstackTp \*S; /\*建立入度为0的顶点栈S\*/

ArcNodeTp \*p;

int m, I, v;

InitStack (S);

for (I=0; I<g.vexnum; I++)

if (g.adjlist[I].in==0) /\*if (w的入度==0)\*/

push(S,&v) /\*w入S栈\*/

m=0;

while (!EmptyStack(S))

{ Pop(S,&v); /\*S出栈->v\*/

printf(“%d”,v); /\*输出v\*/

m++;

p=g.adjlist[I].firstarc; /\*p=图g中顶点v的第一个邻接点\*/

while (p!=NULL) /\*p存在\*/

{ (g.adjlist[p->adjvex].in)--; /\*p的入度--\*/

if (g.adjlist[p->adjvex].in==0) /\*if (p的入度==0)\*/

Push (S, p->adjvex); /\*p入S栈\*/

p=p->nextarc; /\*p=图g中顶点v的下一个邻接点\*/

}

}

if (m<g.vexnum) return 0; /\*图含有环\*/

else return 1; /\*拓扑排序完成\*/

}

第十章 参考答案

二、填空

1．稳定、不稳定　　　　　　　　　　　　　　 2.内部、外部

3．插入排序、交换排序、选择排序、归并排序　 4.键值比较、记录移动、附加空间

5．直接、折半、表、希尔　　　　　　　　　　 6.2、r[j]、r[0]

7.O（n2）、O（1） 8.n-1、flag=1、n-i

9.O(n2) 10.j--、r[i]=r[j]、j++、r[j]=r[i]、x

11.O(nlog2n)、O(n2) 12.n-1、k=j、k!=I

13.O(n2) 14.n-1、存储区

15.叶子、树根、＋∝　　　　　　　　　　　　　16.n-1、log2n、O(nlog2n)

17.完全二叉树、n/2 18.O(1)、O(nlog2n)

19.a[ii]、ii++、a[j]、j++、m、n、O(n-h+1) 20.有序

21．O(log2n) 22.O(n2)

23.冒泡排序、快速排序　　　　　　　　　　　　24.插入排序、快速排序

三、单项选择

1. ③ 2. ③ 3. ② 4. ② 5. ② 6. ②

7. ③ 8. ④ 9. ① 10. ③ 11. ② 12. ③

13. ③ 14. ④ 15. ④ 16. ③ 17. ③ 18. ②

19. ④ 20. ④ 21. ④ 22. ② 23. ③

四、简答及应用

1．①直接插入排序

序号　　1　　2　　3　　4　　5　　6　　7　　8　　9　　10　　11　　12

　 关键字　83 40　 63 13 84 35 96 57 39 79 61 15

i = 2 40 83 [63 13 84 35 96 57 39 79 61 15]

i = 3 40 63 83 [13 84 35 96 57 39 79 61 15]

i = 4 13 40 63 83 [84 35 96 57 39 79 61 15]

i = 5 13 40 63 83 84 [35 96 57 39 79 61 15]

i = 6 13 35 40 63 83 84 [96 57 39 79 61 15]

i = 7 13 35 40 63 83 84 96 [57 39 79 61 15]

i = 8 13 35 40 57 63 83 84 96 [39 79 61 15]

i = 9 13 35 39 40 57 63 83 84 96 [79 61 15]

i = 10 13 35 39 40 57 63 79 83 84 96 [61 15]

i = 11 13 35 39 40 57 61 63 79 83 84 96 [15]

i = 12 13 15 35 39 40 57 61 63 79 83 84 96

　②直接选择排序

序号　　1　　2　　3　　4　　5　　6　　7　　8　　9　　10　　11　　12

　 关键字　83 40　 63 13 84 35 96 57 39 79 61 15

i = 1 13 [40 63 83 84 35 96 57 39 79 61 15]

i = 2 13 15 [63 83 84 35 96 57 39 79 61 40]

i = 3 13 15 35 [83 84 63 96 57 39 79 61 40]

i = 4 13 15 35 39 [84 63 96 57 83 79 61 40]

i = 5 13 15 35 39 40 [63 96 57 83 79 61 84]

i = 6 13 15 35 39 40 57 [96 63 83 79 61 84]

i = 7 13 15 35 39 40 57 61 [63 83 79 96 84]

i = 8 13 15 35 39 40 57 61 63 [83 79 96 84]

i = 9 13 15 35 39 40 57 61 63 79 [83 96 84]

i = 10 13 15 35 39 40 57 61 63 79 83 [96 84]

i = 11 13 15 35 39 40 57 61 63 79 83 84 [96]

　③快速排序

　 关键字　　　83 40　 63 13 84 35 96 57 39 79 61 15

第一趟排序后 [15 40 63 13 61 35 79 57 39] 83 [96 84]

第二趟排序后 [13] 15 [63 40 61 35 79 57 39] 83 　 84 [96]

第三趟排序后 13 15 [39 40 61 35 57] 63 [79] 83 84 96

第四趟排序后 13 15 [35] 39 [61 40 57] 63 79 83 84 96

第五趟排序后 13 15 35 39 [57 40] 61 63 79 83 84 96

第六趟排序后 13 15 35 39 40 [57] 61 63 79 83 84 96

第七趟排序后 13 15 35 39 40 57 61 63 79 83 84 96

④堆排序

关键字：　83　40　63　13　84　35　96　57　39　79　61　15

排序成功的序列：96　84　83　79　63　61　57　40　39　35　15　13

　　　　　排序过程如图简答题8－1.1、8－1.2、8－1.3所示。

⑤归并排序

　 关键字　　　83 40　 63 13 84 35 96 57 39 79 61 15

第一趟排序后 [40 83] [13 63] [35 84] [57 96] [39 79] [15 61]

第二趟排序后 [13 40 63 83] [35 57 84 96] [15 39 　 61 79]

第三趟排序后 [13 35 40 57 63 83 84 96] [15 39 61 79]

第四趟排序后 13 15 35 39 40 57 61 63 79 83 84 96

2．稳定排序有直接插入、冒泡排序、归并排序。

　不稳定排序有快速排序、直接选择排序、堆排序。举例如下：

①快速排序

初始状态　　40　　65　　38　　49　　97　　65　　13　　60

排序后 　　13　　38　　40　　49　　60　　65　　65　　97

（65表示记录初始位置在65记录位置之后）

②堆排序

初始状态　　65　　38　　75　　97　　80　　13　　27　　65

排序后 　　13　　27　　38　　65　　65　　75　　80　　97

③直接排序

初始状态　　40　　65　　38　　49　　97　　65　　13　　60

排序后 　　13　　38　　40　　49　　60　　65　　65　　97

3．直接选择相对于树形选择排序的优点：算法易懂，容易实现，基本上不需要附加空间。而树形排序需要附加n – 1附加空间。

直接选择排序相对于树形选择排序的缺点：直接选择排序每一趟都不保留比较结果比较次数较多，而树形排序则能利用大部分上一次的比较结果，因此时间性能比直接选择排序优越。

堆排序相对于树形排序的优点：堆排序除建第一个堆费时外，其余元素进行堆比较次数小于等于h（h表示n元素构成的完全二叉树的深度），而树形排序每次比较都是h次。因此，堆排序时间复杂度小于树形排序。堆排序基本也不需要附加空间。

4．直接插入排序、直接选择排序、快速排序、堆排序和归并排序时空性如表：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 平均时间 | 最坏情况 | 辅助空间 |
| 直接插入 | O(n2) | O(n2) | O(n1) |
| 直接选择 | O(n2) | O(n2) | O(1) |
| 快速排序 | O(nlg2n) | O(n2) | O(nlog2n) |
| 堆排序 | O(nlog2n) | O(nlog2) | O(1) |
| 归并排序 | O(nlog2n) | O(nlog2n) | O(n) |

从上表可以得出：就平均时间性能而言，快速排序最佳，其所需时间最省，但快速排序在最坏情况下，时间性能不如堆排序和归并排序。　而后两者相比的结果是，当n较大时，归并排序所需时间优于堆排序，但它所需辅助空间最多。

注意，在所有排序方法中，没有哪一种是绝对最优的，有的适用于n较大的情况，有适用于n较小的情况，还有的与关键字的分布和初始位置有关……因此，在实际使用时，需要根据不同情况适当选用排序方法，甚至可将多种方法结合使用。

5．分析：先按序列画出对应的完全二叉树，再看其是否满足堆的定义。按这一思路，画出的完全二叉树见图简答8-5.1.

显然序列⑴是堆，而序列⑵不是堆，值为7的元素应该上调，调整过程见图简答题8-5.2.

6.第一趟：　　 [46, 58, 15, 45, 90, 18, 10, 62]

[46, 58, 15, 45, 90, 18, 10, 62] .

一次交换之后　[10, 58, 15, 45, 90, 18, 46, 62]

二次交换之后　[10, 46, 15, 45, 90, 18, 58, 62]

三次交换之后　[10, 18, 15, 45, 90, 46, 58, 62]

　[10, 18, 15, 45, 90, 46, 58, 62]

　　　　　　　[10, 18, 15, 45, 90, 46, 58, 62]

四次交换之后　[10, 18, 15, 45, 46, 90, 58, 62]

以上“－”表示当前经比较不交换位置的元素。“　”表示当前经比较交换位置的元素。

　　第一趟：　[10 18 15 45] 46 [90 58 62]

第二趟：　10 [18 15 45] 46 [62 58] 90

第三趟：　10 [15] 18 45] 46 [58] 62 90

结果：　　10 15　 18 45 46 58 62 90

7．

⑴插入排序的每趟的结果

　　　初始值健值序列　[12] 5 9 20 6 31 24

i=2 [5 12] 9 20 6 31 24

i=3 [5 9 12] 20 6 31 24

i=4 [5 9 12 20] 6 31 24

i=5 [5 6 9 12 20] 31 24

i=6 [5 6 9 12 20 31] 24

i=7 [5 6 9 12 20 24 31]

⑵冒泡排序的每趟的结果：

　　　初始键值序列　　[12 5 9 20 6 31 24]

第一趟之后　　[5 9 12 6 20 24] 31

第二趟之后　　[5 9 6 12 20] 24 31

第三趟之后　　[5 6 9 12] 20 24 31

第四趟之后　　 5 6 9 12 20 24 31

五、算法设计

1.分析：每趟从单链表头部开始，顺序查找当前链值最小的结点。找到后，插入到当前的有序表区的最后。

Void selesort ( lklist L ) /\* 设链表L带头结点 \*/

{ q=L; /\* 指向第一数据前趋 \*/

while ( q-> next !=NULL )

{ pl = q -> next ;

minp =pl; /\* minp指向当前已知的最小数 \*/

while ( pl -> next != NULL )

{ if ( pl -> next -> data < minp -> data )

minp = pl -> next ; /\* 找到了更小数 \*/

pl = pl -> next ; /\* 继续往下找 \*/

}

if ( minp != q -> next) /\* 将最小数交换到第一个位置上 \*/

{ r1 = minp -> next ;

minp -> next = r1 -> next ; /\* 删除最小数 \*/

r2 = q -> next ;

q -> next = r2 -> next ; /\* 删除当前表中第一个数 \*/

r1 -> next = q -> next ;

q -> next = r1 ; /\* 将最小数插入到第一个位置上 \*/

r2 -> next = minp -> next ;

minp -> next = r2 ; /\* 将原第一个数放到最小数原位置上 \*/

}

q = q > next ; /\* 选择下一个最小数 \*/

}

}

2.分析：先调用划分函数 quickpass () ,以确定中间元素的位置，然后再借助栈分别对中间元素左、右两边的区域进行快速排序。

　Void qksort ( list A ) /\* n为元素个数 \*/

{ InitStack ( S ) ; /\* 设置一个栈保存有关参数和变量 \*/

　　　j = 1 ; h = n ; /\* j , h 分别指向表头和表尾 \*/

while ( ( j < h ) ‖( !empty ( S ) ) )

{ while ( j < h )

{ quickpass ( A,j,h,i ) ; /\* 划分函数 quickpass () 参照教材 \*/

push ( S,j,h,i ) ; /\* 保存变量值 \*/

h = i – 1 ; /\* 设置对左边进行划分的参数 \*/

}

if ( !empty ( S ) )

{ pop ( S,j,h,i ) ; /\* 取出变量值 \*/

j = i + 1 ; /\* 设置对右边进行划分的参数 \*/

}

}

}

3．分析：插入排序的基本思想是：每趟从无序区间中取出一个元素，再按键值大小插入到前面的有序区间中。对于有序区，当然可以用二分查找来确定插入位置。

　　Void straightsort ( list A ) /\* n为元素个数，数组下标从1开始，到n结束。 \*/

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

{ low = 1 ; high = i – 1 ; /\* low ,high 分为当前元素上、下界 \*/

A[0] . key = A[i] . key ;

While ( low <= high )

{ mid = ( low + high ) /2 ;

switch

{ case : A[0] . key <= A[mid] . key : high = mid – 1 ; /\* 修改上界 \*/

case : A[0] . key > A[mid] . key : low = mid + 1 ; /\* 修改下界 \*/

}

for ( j = i-1 ; j >= mid ; j - -) A [j +1] = A [ j ] ; /\* 移动数据 \*/

A[ mid ] = A[ i ] ;

}

}

}

4．分析：本题的算法思想是：先设置好上、下界，然后分别从线性表两端查找正数和负数，找到后进行交换，直到上、下界相遇。

Void example ( datatype A [n] )

{ i = 1, j = n ; /\* i , j为左右边界 \*/

while ( i < j )

{ while ( ( i < j ) && ( A[ i ] < 0 ) ) i++ ; /\* 在左边界找正数 \*/

while ( (i < j ) && ( A[ j ] > 0 )) j - - ; /\* 在右边界找负数 \*/

if ( i < j)

{ temp = A[ i ]; A[ i ] = A[ j ]; A[ j ] = A [temp ] ; /\*交换两个元素的值 \*/

i++ ; j - -;

}

}

}

5．分析：此问题分为两个算法，第一个算法 shift 假设 a[ 1] , a[ 2 ],…,a[ k]为堆，增加一个无素a[ k + 1 ],把数组a[ 1 ],a[ 2 ], …,a[ k + 1 ]调整为堆。第二个算法heep 从1开始调用算法sift将整个数组调整为堆。

Void sift (datatype A[ n ] , int K) /\* n > = k + 1 \*/

{ x = A[ K+1] ;

i = K +1 ;

while ( ( i/2 > 0 )&&( A[i/2]>x) ) { A[i]= A[i./2]; i = i/2;} /\*从下往上插入位置 \*/

A[i] = x ;

}

Void heap ( datatype A[ n ] ) ; /\* 从1开始调用算法sift ，将整个数组调整为堆 \*/

{ for ( k = 1 ; k <= n-1; k++ ) sift ( A,k ) ; }

6．分析：本算法采用的存储结构是带头结点的双循环链表。首先找元素插入位置，然后把元素从原链表中删除，插入到相应的位置处。请注意双链表上插入和删除操作的步骤。

Void sort ( dlklist H ) /\* 链表H采用带头结点的双循环链表 \*/

{ pre = H -> next ;

while ( p != H )

{p = pre -> next ; /\* p是有序表的末尾 \*/

q = p -> next ; /\* 保存下一个插入元素 \*/

while((pre!=H)&&(p->data<pre ->data))pre=pre–prior; /\*从后往前找插入位置 \*/

if ( pre ! =p -> prior )

{ p-> prior->next = p -> next ; /\* 删除p \*/

p-> next->prior = p -> prior;

p->next = pre ->next ; /\* 插入到pre之后 \*/

pre -> next -> prior = p ;

p ->prior= pre ; pre ->next = p ;

}

p=q;

}

}

全真模拟试题（一）

1. **单项选择题（在每小题的4个备选答案中，选出正确的答案，并将其号码填在**

**题干的括号内。每小题2分，共24分）**

* 1. 若某线性表中最常用的操作是取第i 个元素和找第i个元素的前趋元素，则采用（ ）存储方式最节省时间。

①单链表 ②双链表 ③单向循环 ④顺序表

* 1. 串是任意有限个（ ）

①符号构成的序列 ②符号构成的集合

③字符构成的序列 ④字符构成的集合

* 1. 设矩阵A（aij ，l≤i,j≤ 10）的元素满足：

aij≠0(i≥j, l≤i, j≤ 10)

aij=0 (i<j, l≤i, j≤ 10)

现将A的所有非0元素以行序为主序存放在首地址为2000的存储区域中，每个元素占有4个单元，则元素A[9][5]的首址为

①2340 ②2336 ③2164 ④2160

* 1. 如果以链表作为栈的存储结构，则退栈操作时（ ）
     1. 必须判别栈是否满
     2. 对栈不作任何判别
     3. 必须判别栈是否空
     4. 判别栈元素的类型
  2. 设数组Data[0..m]作为循环队列SQ的存储空间，front为队头指针，rear为队尾指针，则执行出队操作的语句为（ ）

①front=front+1 ②front=(front+1)% m

③rear=(rear+1)%m ④front=(front+1)%(m+1)

* 1. 深度为6（根的层次为1）的二叉树至多有（ ）结点。
     1. 64 ②32 ③31 ④63
  2. 将含100个结点的完全二叉树从根这一层开始，每层上从左到右依次对结点编号，根结点的编号为1。编号为49的结点X的双亲编号为（ ）

①24 ②25 ③23 ④无法确定

* 1. 设有一个无向图G=（V，E）和G’=（V’，E’）如果G’为G的生成树，则下面不正确的说法是（ ）

①G’为G 的子图 ②G’为G 的边通分量

③G’为G的极小连通子图且V’=V ④G’为G的一个无环子图

* 1. 用线性探测法查找闭散列表，可能要探测多个散列地址，这些位置上的键值（ ）
     1. 一定都是同义词 ②一定都不是同义词

③都相同 ④不一定都是同义词

* 1. 二分查找要求被查找的表是（ ）
     1. 键值有序的链接表 ②链接表但键值不一定有序

③ 键值有序的顺序表 ④顺序表但键值不一定有序

* 1. 当初始序列已经按键值有序，用直接插入算法对其进行排序，需要循环的次数为（ ）

①n2 ②nlog2n ③log2n ④n-1

* 1. 堆是一个键值序列{k1,k2,…, kn},对i=1,2,…,|\_n/2\_|,满足( )

①ki≤k2i≤k2i+1 ②ki<k2i+1<k2i

③ki≤k2i且ki≤k2i+1(2i+1≤n) ④ki≤k2i 或ki≤k2i+1(2i+1≤n)

1. **判断题（判断下列各题是否正确，正确在括号内打“V”，错的找“X”。每小题1分，共10分）**
   1. 双链表中至多只有一个结点的后继指针为空。（ ）
   2. 在循环队列中，**front**指向队列中第一个元素的前一位置，rear指向实际的队尾元素，队列为满的条件是**front=**rear。（ ）
   3. 对链表进行插入和删除操作时，不必移动结点。（ ）
   4. 栈可以作为实现程序设计语言过程调用时的一种数据结构。（ ）
   5. 在一个有向图的拓朴序列中，若顶点a在顶点b之前，则图中必有一条弧<a,b>。( )i
   6. 对有向图G，如果从任一顶点出发进行一次深度优先或广度优先搜索就能访问每个顶点，则该图一定是完全图。( )
   7. “顺序查找法”是指在顺序表上进行查找的方法。（ ）
   8. 向二叉排序树插入一个新结点时，新结点一定成为二叉排序树的一个叶子结点。（）
   9. 键值序列{A，C，D，E，F，E，F}是一个堆。
   10. 二路归并时，被归并的两个子序列中的关键字个数一定要相等。（）
2. **填空题（每空2 分，共24分）**
   1. 设r指向单链表的最后一个结点，要在最后一个结点之后插入s所指的结点，需执行的三条语句是\_\_\_\_\_\_\_\_\_\_\_；r=s; r->next=null;。
   2. 在单链表中，指针p 所指结点为最后一个结点的条件是\_\_\_\_\_\_\_\_\_\_\_。
   3. 设一个链栈的栈顶指针是ls，栈中结点格式为info | link ,栈空的条件是\_\_\_\_\_\_\_\_\_\_.如果栈不为空，则退栈操作为p=ls；\_\_\_\_\_\_\_\_\_\_\_；free(p);。
   4. 已知一棵度为3的树有2个度为1的结点，3个度为2的结点，4个度为3的结点，则该树中有\_\_\_\_\_\_\_\_\_\_\_\_ 个叶子的结点。
   5. 树有三种常用的存储结构，即孩子链表法、孩子兄弟链表法和\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ .
   6. N个顶点的连通图的生成树有\_\_\_\_\_\_\_\_\_\_\_条边。
   7. 一个有向图G中若有弧<vi,vj>、<vj,vk>和<vi,vk>, 则在图G的拓扑序列中，顶点vi,vj和vk的相对位置为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
   8. 设表中元素的初始状态是按键值递增的，分别用堆排序、快速排序、冒泡排序和归并排序方法对其进行（按递增排序），　　　　　最省时间，　　　　　最费时间。
   9. 下面是将键值为x 的结点插入到二叉排序树中的算法，请在划线处填上适当的内容。

typedef struct pnode

{int key;

struct pnode  **\*le**ft, \*right;

}pnode;

void searchinsert(int x, pnode t ) /\*t为二叉排序树根结点的指针\*／

{if ( )

{p=malloc(size);

p->key=x;p->lchild=null;

p->rchild=null;t=p;

} else if (x<t->key) searchinsert(x,t->lchild)

else\_\_\_\_\_\_\_\_\_;

}

1. **应用题（本题共２８分）**

1．树的后根遍历方法是：若树非空则（４分）

（1）依据次后根遍历根的各个子树Ｔ１，Ｔ２，……Ｔm;

（2）访问根结点。

对下图所示的树，用后根遍历方法进行遍历，请写出遍历所得到的结点访问序列。

A

BA

CA

DA

EA

FA

GA

HA

I

J

KA

1. 将下图的森林转换为二叉树。（４分）

A

BA

CA

DA

EA

FA

GA

I

J

1. 下图表示一个地区的交通网，顶点表示城市，边表示连结城市间的公路，边上的权表示修建公路花费的代价。怎样选择能够沟通每个城市且总造价最省的n-1条公路，画出所有可能的方案。（４分）

v2

v4

v1

v5

v3

v6

16

21

11

14

33

6

19

18

6

5

∧

∧

1

0

0

1

1

2

2

3

5

7A

∧

∧

∧

∧

0 v1

1 v2

2 v3

3 v4

4 v5

5 v6

6 v7

7 v8

2

3

7

7

7

4

5

6

4

6

∧

∧

图7.11 遍历无向图 (a) 无向图G6 (b) 深度优先搜索示例 (c) G6的邻接表表示

(c)

表头结点

4．已知一个无向图的邻接表如下图所示。（本题４分，每小题２分）

V5

V1

V2

V3

V4

1

1

2

3

4

5

2

5

4

3

3

4

4

5

2

2

1

Λ

Λ

Λ

Λ

Λ

1. 画出这个图。

(2) 以v1为出发点，对图进行广度优先搜索，写出所有可能的访问序列。

5.设n个元素的有序表为Ｒ，Ｋ为一个给定的值，二分查找算法如下：

int binsearch(sqlist R, keytype K)

{j=1;h=n ;suc=0;

while((j<=h)&&(!suc))

{mid =(j+h)/2;

switch

{case K=R[mid].key: suc=1; break;

case K<R[mid].key: h=mid-1; break;

case K>R[mid].key: j=mid+1

}

}

if (suc) return(mid); else return(0);

}

将上述算法中划线语句改为：K<R[mid].key: h=mid.

1. 改动后，算法能否正常工作？请说明原因。
2. 若算法不能正常工作，给出一个查找序列和一个出错情况的查找键值；若能正常工作，请给出一个查找序列和查找某个键值的比较次数。（本题６分，每小题３分）

6.有一组键值25，84，21，47，15，27，68，35，24，采用快速排序方法由小到大进行排序，请写出每趟的结果，并标明在第一趟排序过程中键值的移动情况。（本题６分）

**五、设计题（共14分）**

1．设棵二叉树以二叉链表为存储结构，结点结构为　lchild |data |rchild 。设计一个算法，求在前根序列中处于第k个位置的结点。（本题６分）

1. 设某单链表Ｌ的结点结构为data |next,试画出该链表的结构图，并用类Ｃ语言编写算法判断该链表的元素是否是递增的。（本题８分）

**全真模拟试题（二）**

**一、单项选择题（在每个小题的4个备选答案中，选出正确的答案，并将其号码填在题后的括号内。每小题2分，共24分）**

1．一个具有n个顶点的无向完全图的边数为（　 ）

①n(n+1)/2 ②n(n-1)/2 ③n(n-1) ④n(n+1)

2．在索引顺序表中查找一个元素，可用的且最快的方法是（　 ）

①用顺序查找法确定元素所在块，再用顺序查找法在相应块中查找

②用顺序查找法确定元素所在块，再用二分查找法在相应块中查找

③用二分查找法确定元素所在块，再用顺序查找法在相应块中查找

④用二分查找法确定元素所在块，再用二分查找法在相应块中查找

3．若某线性表中最常用的操作是在最后一个元素之后插入一个元素和删除最后一个元素，则采用（ 　）存储方式最节省运算时间。

1. 单链表　 ②双链表

③带头结点的双循环链表　④容量足够大的顺序表

4．串是（　 ）

①一些符号构成的序列　 ②有限个字母构成的序列

③一个以上的字符构成的序列　④有限个字符构成的序列

5．堆排序在最坏情况下，其时间复杂性为（ 　）

1. O(nlog2n) 　②O(n2)

③O(log2n2) ④O(log2n)

6．快速排序的记录移动次数（　 ）比较次数，其总执行时间为O(nlog2n)。

1. 大于 　②大于等于 　③小于等于 　④小于

　　7．一棵二叉树有n个结点，要按某顺序对该二叉树中的结点编号，（号码为1-n），编号须具有如下性质：二叉树中任一结点V，其编号等于其左子树中结点的最大编号加1。而其右子树中结点的最小编号等于V的编号加1。试问应按（　）遍历顺序编号。

1. 前根 　②中根 　③后根 　④层次

　　8.3个结点可构成（ 　）个不同形态的二叉树。

1. 2 　②3 　③4 　④5

　　9．对有n个记录的有序表采用二分查找，其平均查找长度的量级为（　）

1. O(log2n)　②O(nlog2n)　③O(n)　④O(n2)

10．对有n个记录的表按记录键值有序的顺序建立二叉树，在这种情况下，其平均查找长度的量级为（　）

1. O(n)　②O(nlog2n)　③O(1)　④(log2n)

11．栈操作的原则是（　）

1. 先进先出　②后进先出　③栈顶插入　④栈顶删除

12．设矩阵A是一对称矩阵(aij=aji,1<=i,j<=8)，若每个矩阵元素占3个单元，将其上三角部分（包括对角线）按行序为主序存放在数组B中，B的首地址为1000，则矩阵元素a67的地址为（　）

1. 1031　 ②1093 　③1096 　④1032

**二、判断题（判断下列各题是否正确，正确在括号内打“√”，错的打“×”。每小题1分，共10分）**

1．如果两个串含有相同的字符，则这两个串相等。　　（　）

2．数组可以看成线性结构的一种推广，因此可以对它进行插入、删除等运算。　（　）

3．在索引顺序表上实现分块查找，在等概率查找情况下，其平均查找长度不仅与表中元素个数有关，而且与每一块中元素个数有关。（　）

4．在顺序表中取出第i个元素所花费的时间与i成正比。　　（　）

5．在栈满情况下不能作进栈运算，否则产生“上溢”。　　　　(　）

6．二路归并排序的核心操作是将两上有序序列归并为一个有序序列。　（　）

7．对任意一个图，从它的某个顶点出发，进行一次深度优先或广度优先搜索，即可访问图的每个顶点.（　）

8．二叉排序树或者是一棵空二叉树，或者是具有下列性质的二叉树：若它的左子树非空，则根结点的值大于其左孩子的值;若它的右子树非空，则根结点的值小于其右孩子的值。（　）

9．在执行某个排序算法过程中，出现了排序码朝着最终排序序列位置相反方向移动，则该算法是不稳定的。（　）

10．一个有向图的邻接表和逆邻接表中表结点的个数一定相等。　（　）

**三、填空题（每空2分，共26分）**

1．在带有头结点的单链表L中，若要删除第一个结点，则需执行下列三条语句：＿＿＿＿＿＿＿＿；L->next=U->next；free(U)；

2．有一个长度为20的有序表采用二分查找方法进行查找，共有＿＿＿＿＿＿个元素的查找长度为3。

3．采用冒泡排序对有n个记录的表A按键值递增排序，若L的初始状态是按键值递增，则排序过程中记录的比较次数为＿＿＿＿＿。若A的初始状态为递减排列，则记录的交换次数为＿＿＿＿＿＿＿。

4．在无头结点的双链表中，指针P所指结点是第一个结点的条件是＿＿＿＿＿＿。

5．G为无向图，如果从G的某个顶点出发，进行一次广度优先搜索，即可访问图的每个顶点，则该图一定是＿＿＿＿＿图。

6．如果一个有向图中没有＿＿＿＿＿＿，则该图的全部顶点可能排成一个拓扑序列。

7．深度为8（根的层次号为1）的满二叉树有＿＿＿＿＿＿个叶子结点。

8．将一棵有100个结点的完全二叉树按层编号，则编号为49的结点X，其双亲PARENT（X）的编号为＿＿＿＿＿＿＿。

9．设某闭散列表HT未满，散列函数H（KEY）为键值第一字母在字母表中的序号，处理冲突方法为线性探测法，请在下列算法划线处填上适当内容，以实现按键值第一字母的顺序输出闭散列表中所有键值的算法。

void printword(keytype HT[m])

{ for(i=1;i<=26;i++)

{ j=i;

while(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{ if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) printf(“datatype”,HT[j]);

j=(j+1)% m;

}

}

}

10．设有一个链队，结点结构为data|next，front为队头指针，rear为队尾指针，当执行入队操作时需执行下列语句：

malloc(p)；p->data=x; p->next=NULL；

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿；

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿；

**四、应用题（共26分）**

1．有向图G的邻接表如下图所示，若删去图G中的边＜V3，V6＞和＜V4，V5＞，试画出修改后图的邻接表。（4分）

顶点　入度

|  |  |  |
| --- | --- | --- |
| V1 | 0 |  |
| V2 | 1 |  |
| V3 | 4 |  |
| V4 | 0 |  |
| V5 | 1 |  |
| V6 | 1 | ∧ |

|  |  |
| --- | --- |
| 2 |  |
| 3 | ∧ |
| 6 |  |
| 3 |  |
| 3 |  |

|  |  |
| --- | --- |
| 1 | ∧ |

|  |  |
| --- | --- |
| 5 | ∧ |

2．有向图如下图所示，写出以V1为出发点对图进行深度优先搜索所得到的所有可能的访问序列。（4分）

3

1

2

5

4

6

3．对于键值序列（49,38,65,97,76,13,27,50），使用堆排序算法完成排序过程。要求：

⑴画出初始堆（用二叉树表示）。

⑵画出分别输出13,27后重建的两个堆。（5分）

4.一个深度为d（根的层次号为1）的满K叉树有如下性质：第d层上的结点都是叶子结点，其余各层上的每个结点都有K棵非空子树。如果从根这一层开始从左到右顺序逐层对全部结点编号，且根结点的编号为1，问编号为n的结点有右兄弟的条件是什么？其右兄弟的编号是多少？（3分）

5．给定权值｛5,10,12,15,30,40｝，构造相应的哈夫曼树，要求写构造歩骤。（4分）

6．已知一表为（Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec）,按表中顺序依次插入初始为空的二叉排序树，要求：

⑴画出建立的二叉排序树。（4分）

⑵求出在等概率情况下查找成功的平均查找长度。（2分）

**五、设计题（共14分）**

1．设有一单链表L，结点结构为data|next，结点个数至少3个，试画出链表L的结构图，并编写算法判断该单链表L中的元素是否成等差关系，即：设各元素值次为a1,a2,a3,…,an，判断ai+1-ai=ai-ai-1是否成立，其中i满足2<=i<=n-1.(8分)

2．设有一棵二叉树以二叉链表作为存储结构，结点结构为lchild|data|rchild,其中data域中存放一个字符，设计一个算法按前根遍历顺序仅打印出data域为数字的字符（即‘0’<=data<=‘9’）（6分）

全真模拟试题(一)参考答案

1. 单项选择题

1④ 2③

3④ 分析：按题意，矩阵A是个三角矩阵，A[ I,j]的首地址可用下列公式计算：

LOC（aij）=LOC(a11)+(k-1)\*L

其中K为A[I,j]在A中的序号k=I\*(I-1)/2+j，L为每个元素所占的单元数。

所以有：LOC（aij）=2000+（9\*（9-1）/2+5-1）\*4=2000+160=2160。故为答案④

4．③

5．④

6．④

7．①

8．② 分析：如果G’为G的生成树，那么G’是G的子图，也是G的无环子图，并且还是G的极小连通子图，且V’=V，而连通分量则是指无向图的极大连通子图。故答案②是错误的。

9．④ 10。③ 11。④ 12。③

1. 判断题

1.v 2.x 3.v 4.v 5.x 6.x 7.x 8.v 9.v 10.x

1. 填空题
   1. R->next =s.
   2. P->next= = NULL
   3. Ls= =NULL 、ls=ls->link.
   4. 12 分析： 设n1=2,n2=3,n3=4,

树的总结点数为n=n0+ n1+n2+n3

树的分支数为n-1= n1+2n2+3n3

②-①得：-1= n2+2n3-n0

有n0=n2+2n3+1=3+2\*4+1=12

* 1. 双亲表示法。
  2. N-1
  3. I,j,k.
  4. 冒泡排序、快速排序
  5. T= =NULL、searchinsert(x,t->rchild).

1. 应用题
   1. EBFGCKHIJDA。
   2. 答案如图应用题I 9. 2.2 所示。
   3. 3．分析：本题实际上是求最小生成树问题。由于衅中有两条权值为6的边，故可以得到两种方案。答案如图应用题I 9. 3.2 所示。
   4. 答案：

（1）答案如图应用题I 9. 4.2 所示。

（2）v1 v2 v4 v5 v3 和 v1 v4 v2 v3 v5。

5．（1）经过改动以后，有可能出现死循环，比如当查找的键值K小于有序表中的最小键值时，就会出现死循环。故算法不能正常进行。

（2）假设有序表的查找序列为（2，3，4，5，6），当待查的键值K=1时，出现死循环。

6．答案：

25 84 21 47 15 27 68 35 24

第一趟 [24 15 21] 25 [47 27 68 35 84]

第二趟 [21 15] 24 25 [35 27] 47 [68 84]

第三趟 [15] 21 24 25 [27] 35 47 68 [84]

得到 15 21 24 25 27 35 47 68 84

第一趟排序过程中键值的移动情况如下：

第一趟： [25 84 21 47 15 27 68 35 24 ]

一次交换之后 [24 84 21 47 15 27 68 35 25]

二次交换之后 [24 25 21 47 15 27 68 35 84]

[24 25 21 47 15 27 68 35 84]

[24 25 21 47 15 27 68 35 84]

[24 25 21 47 15 27 68 35 84]

三次交换之后 [24 15 21 47 25 27 68 35 84]

[24 15 21 47 25 27 68 35 84]

四次交换之后 [24 15 21 25 47 27 68 35 84]

以上“-”表示当前经比较不交换位置的元素。“ ”表示当前经比较交换位置的元素。

1. 设计题
   1. Bitreptr search(bitreptr t ,int k)

{if (t!=null)

{count++;

if (count= =k)return (t);

else {search(t->lchild,k);

search(t->rchild,k);

}

}

}

2. 单链表L的结构如图设计题I 9. 2.2所示。

Int isviser(lklist L)

{p=L;

while(p->next!=null)

if (p->data <p->next->data) p=p->next;

else return(0);

return(1);

}

　　　　　　　　　　　　全真模拟二参考答案

一、单项选择题

1.②　　2.③　　3.④　　4. ④ 5.① 6. .③ 7. ② 8. ④ 9. ①

10. ①　对键值有序的、具有n个记录的表来讲，当所建立的二叉排序树是一棵深度为n的单支树时，在它上面的查找操作已经退化为顺序查找，所以其平均查找长度的量级为O（n）.

11．②

12．②　按题意要求，将对称矩阵A的上三角部分按行优先进行存放数组了B中，那么B[k]与aij的对应关系为：

当i<=j时，k=(i-1)/2\*(2\*n-i+2)+j-i+1

因此有：k=(6-1)/2\*(2\*8-6+2)+7-6+1=32

故　　　　LOC（a67）=LOC(a11)+(k-1)\*l=1000+(32-1)\*3=1093

二、判断题

　　1. ×　　2. × 3. √ 4. × 5. √ 6. √ 7. × 8. × 9. × 10.

三、填空题

1. U=L - > next
2. 4。　分析：二分查找的过程可以用一棵有序树来表示，该树第三层上有4个结点，表示经过三次比较查找成功的元素个数为4。
3. n-1、n(n-1)/2。　分析：采用冒泡排序时，若初始时已经自然有序，那么经过一趟n-1次比较后，算法就自动终止了。若初始状态为递减排列，希望排序成递增排列，则排序过程中比较一次，交换一次，总的比较、交换次数为n(n-1)/2，其中n-1为趟数，n/2为平均每趟的比较交换次数。
4. p - > prior = NULL。
5. 连通
6. 回路或环
7. 28-1 = 27 = 128
8. 24
9. HT[j]!=NULL或HT[j]不为空、H(HT[j])=I
10. rear - > next = p、rear = p

四、应用题

1. 修改后的有向图G的邻接表如图应用题Ⅱ 9.1.2所示。

顶点　入度

|  |  |  |
| --- | --- | --- |
| V1 | 0 |  |
| V2 | 1 |  |
| V3 | 4 | ∧ |
| V4 | 0 |  |
| V5 | 0 |  |
| V6 | 0 | ∧ |

|  |  |
| --- | --- |
| 2 |  |
| 3 | ∧ |

|  |  |
| --- | --- |
| 1 | ∧ |

|  |  |
| --- | --- |
| 3 | ∧ |
| 3 | ∧ |

　图应用题Ⅱ 9.1.2

1. 1，2，5，4，3，6

　　　　1，3，6，4，5，2

　　　　1，3，5，4，6，2

3．⑴初始堆如图应用题Ⅱ 9.3.2所示。

　 ⑵输出13后重建的堆如图应用题Ⅱ 9.3.3所示。

　 ⑶输出27后重建的堆如图应用题Ⅱ 9.3.4所示。

4．分析：在满k叉树中，除编号为1的根结点外，其余结点依次为每k个结点拥有一个共同的双亲。比如：

第二号－第k+1号结点的双亲是第1号结点；

第k+2号－第2k+1号结点的双亲是第2号结点；

第2k+1号－第3k+1号结点的双亲是第3号结点；

　　　　　　．．．．．．

从中可以看出，若编号为n，那么当(n-1)%k = 0时，它一定是某个结点的最右边的孩子，即它的右边不会再有兄弟了。反之，当(n-1)%k≠0，它的右边一定还有兄弟。

答案：编号为n的结点有兄弟的条件是(n-1)%k≠0，该点的右兄弟的编号是n+1。

　　　　5．哈夫曼树的构造过程如图应用题Ⅱ 9.5.2所示。

　　　　6．⑴建立的二叉排序树如图应用题Ⅱ 9.6.2所示。

　　　　　 ⑵在等概率情况下，查找成功的平均查找长度为

1/2(1\*1+2\*2+3\*3+4\*3+5\*2+6\*1) = 42/12 = 7/2 = 3.5。

　　　　五、设计题

　　　　1．单链表的结构图如图设计题Ⅱ 9.1.2所示。

．．．

|  |  |
| --- | --- |
| a1 |  |

|  |  |
| --- | --- |
| a2 |  |

|  |  |
| --- | --- |
| an |  |

算法：int isrise (lklist L)

{ p = L -> next; b = p -> data – L -> data;

while (p -> next != NULL)

{ q =p -> next;

if (q -> data – p -> data !=b) return(0)

else p = q;

}

return(1);

}

2．Void Nchar (bitreptr t)

{ if (t != Null)

{ if (t -> data >= ’0’ ) && (t -> data <= ‘9’) printf (“%d” , t -> data );

Nchar (t -> lchild);

Nchar (t -> rchild);

}

}

**全真试题（一）**

**本试卷分两部分，第一部分为选择题，1页至3页，第二部分为非选择题，4页至10页，共10页，选择题30分，非选择题70分，满分100分。考试时间的150分钟。**

**第一部分 选择题 （共30分）**

**一、单项选择题（本大题共15小题，每小题2分，共30分）。在每小题列出的四个选项中只有一个是符合题目要求的，请将其代码填在题后的括号内。错选或未选均无分。**

1. 算法指的是 （ ）

A.计算机程序 B. 解决问题的计算方法 C. 排序算法 D. 解决问题的有限运算序列

2. 线性表采用链式存储时，结点的存储地址 （ ）

A.必须是不连续的 B.连续与否均可 C.必须是连续的 D.和头结点的存储地址相连续

3. 将长度为n的单链表链接在长度为m的单链表之后的算法的时间复杂度为 （ ）

A. O(1) B. O(n) C. O(m) D. O(m+n)

4. 由有两个栈共享一个向量空间的好处是: （ ）

A. 减少存取时间，降低下溢发生的机率 B.节省存储空间，降低上溢发生的机率

C. 减少存取时间，降低上溢发生的机率 D.节省存储空间，降低下溢发生的机率

5．设数组data[m]作为循环队列SQ的存储空间，front为头指针，rear为尾指针，则执行出队操作后其头指针的值为 （ ）

A. front=front+1 B. front=(front+1)%(m-1) C. front=(front-1)%m D. front=(front+1)%m

6. 如下陈述中正确的是 （ ）

A．串是一种特殊的线性表 B. 串的长度必须大于零 C.串中元素只能是字母 D.空串就是空白串

7．设目标串的长度是n，模式串的长度为=⎣n/3⎦，则执行模式匹配算法时，在最坏情况下的时间复杂度是（ ）

A.O() B.O(n) C.O(n2) D.O(n3)

8．一个非空广义表的表头 （ ）

A．不可能是子表 B. 只能是子表 C.只能是原子 D.可以是子表或原子

9．假设以带行表的三元组表表示稀疏矩阵，则和下列行表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 2 | 3 | 3 | 5 |

对应的稀疏矩阵是

10.在一棵度为3的树中，度为3的结点个数为2，度为2的结点个数为1，则度为0的结点个数为 （ ）

A. 4 B. 5 C. 6 D. 7

11. 在含n个顶点和e条边的无向图的邻接矩阵中，零元素的个数为 （ ）

A. e B.2e C. n2-e D. n2-2e

12．假设一个有n个顶点和e条弧的有向图用邻接表表示，则删除与某个顶点vi相关的所有弧的时间复杂度是（ ）

A. O(n) B.O(e) C.O(n+e) D.O(n\*e)

13．用某种排序方法对关键字序列（25，84，21，47，15，27，68，35，20）进行排序时，序列的变化情况如下：

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

则所采用的排序方法是 ( )

A.选择排序 B.希尔排序 C.归并排序 D.快速排序

14．适于对动态查找表进行高效率查找的组织结构是 ( )

A．有序表 B.分块有序表 C.二叉排序树 D. 线性链表

15．不定长文件是指 ( )

A．文件的长度不固定 B. 记录的长度不固定 C.字段的长度不固定 D.关键字项的长度不固定

**第二部分 非选择题 （共70分）**

**二、填空题（本大题共10小题，每小题2分，若有两个空格，每个空格1分，共20分）。**

**不写解答过程，将正确的答案写在每小题的空格内。错填或不填均无分。**

16．数据的逻辑结构是从逻辑关系上描述数据，它与数据的 无关，是独立于计算机的。

17．在一个带头结点的单向循环链表中，p指向尾结点的直接前驱，则指向头结点的指针head可用p表示为head＝ 。

18．栈顶的位置是随着 操作而变化的。

19．在串S=”structure”中，以t为首字符的子串有 个。

20．假设一个9阶的上三角矩阵A按列优先顺序压缩存储在一维数组B中，其中B[0]存储矩阵中第1个元素B1,1,则B[31]中存放的元素是 。

21．已知一棵完全二叉树中共有768个结点，则该树中共有 个叶子结点。

22．已知一个图的广度优先生成树如右图所示，则与此相应

a

c

b

e

f

g

d

的广度优先遍历序列为 。

23.在单链表上难以实现的排序方法有 和 。

24．在有序表（12，24，36，48，60，72，84）中二分查找关键字72时所需进行的关键字比较次数为 。

25．多重表文件和倒排文件都属于 文件。

**三、解答题（本大题共4小题，每小题5分，共20分）。**

26．画出下列广义表的共享结构图形表示

P=(((z),(x,y)),((x,y),x),(z))

27.请画出与下列二叉树对应的森林。

A

B

C

D

E

F

G

I

J

K

28.已知一个无向图的顶点集为{a,b,c,d,e}，其邻接矩阵如下所示



1. 画出该图的图形；
2. 根据邻接矩阵从顶点a出发进行深度优先遍历和广度优先遍历，写出相应的遍历序列。

29．已知一个散列表如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 35 |  | 20 |  |  | 33 |  | 48 |  |  | 59 |
| 0 1 2 3 4 5 6 7 8 9 10 11 12 | | | | | | | | | | | | |

其散列函数为h(key)=key%13，处理冲突的方法是双重散列法，其探查序列为：

hi=(h(key)+i\*hl(key))%m i=0,1,…,m-1

其中

hl(key)=key%11+1

回答下列问题：

1. 对表中关键字35，20，33，和48进行查找时，所需进行的比较次数各为多少？
2. 该散列表在等概率查找时查找成功的平均查找长度为多少？

**四、算法阅读题（本大题共4小题，每小题5分，共20分）。**

30．下列算法的功能是比较两个链串的大小，其返回值为：



请在空白处填入适当的内容。

int comstr(LinkString s1,LinkString s2)

{//s1和s2是两个链串的头指针

while (s1 && s2)

{if(s1->data<s2->data) return -1;

if(s1->data>s2->data) return 1;

① ;

② ;

}

if( ③ ) return -1;

if( ④ ) return 1;

⑤ ;

}

31. 阅读下面的算法

LinkList mynote(LinkList L)

{//L是不带头结点的单链表的头指针

if(L && L->next)

{q=L; L=L->next; p=L;

S1: while(p->next) p=p->next;

S2: p->next=q; q->next=NULL;

}

return L

}

请回答下列问题：

1. 说明语句S1的功能；
2. 说明语句组S2的功能；
3. 设链表表示的线性表为（a1, a2,…, an），写出算法执行后的返回值所表示的线性表。

rear[0]

……

……

rear[1]]

front[1]]

front[0]]

32. 假设两个队列共享一个循环向量空间（参见右下图）

其类型Queue2定义如下：

typedef struct

{DataType data[MaxSize];

int front[2],rear[2];

}Queue2;

对于i=0或1，front[i]和rear[i]分别为第

i个队列的头指针和尾指针。请对以下算法填

空，实现第i个队列的入队操作。

int InQueue(Queue2 \*Q,int i,DataType x)

{//若第i个队列不满。则元素x入队列，

//并返回1；否则返回0

if(i<0 || i>1) return 0;

if(Q.>rear[i]==Q->front[ ① ]return 0;

Q->data[ ② ]=x;

Q->rear[i]= ③ ;

return 1;

}

33. 已知二叉树的存储结构为二叉链表，阅读下面算法。

typedef struct node

{DataType data;

struct node \*next;

}ListNode;

typedef ListNode \*LinkList;

LinkList Leafhead=NULL;

void Inorder(BinTree T)

{LinkList s;

if (T)

{Inorder(T->lchild);

if((!T->lchild) && (T->rchild))

{s=(ListNode\*)malloc(sizeof(ListNode));

s->data=T->data;

s->next=Leafhead;

Leafhead=s;

}

Inorder(T->rchild);

}

}

对于如下所示的二叉树

D

A

B

C

H

E

F

G

1. 画出执行上述算法后所建立的结构；
2. 说明该算法的功能。

**五、算法设计题（本题共10分）**

34．阅读下列函数

int arrange(int a[],int l,int h,int x)

{ //l和h分别为数据区的下界和上界

int i,j,t;

i=l; j=h;

while(i<j)

{while(i<j &&a[j]>=x) j--;

while(i<j &&a[i]<x) i++;

if(i<j)

{t=a[j]; a[j]=a[i]; a[i]=t;}

}

if (a[i]<x) return i;

else return i-1;

}

1. 写出该函数的功能；
2. 写一个调用上述函数实现下列功能的算法：对一整数数组b[n]中的元素进行重新排列，将所有负数均调整到数组的低下标端，将所有正数均调整到数组的高下标端，若有零值，则置于两者之间，并返回数组中零元素的个数。

**全真试题（二）**

**本试卷分两部分，第一部分为选择题，1页至2页，第二部分为非选择题，3页至10页，共10页；选择题30分，非选择题70分，满分100分。考试时间150分钟。**

**第一部分 选择题（共30分）**

**一、单项选择题（本大题共15小题，每小题2分，共30分）在每小题列出的四个选项中只有一个是符合题目要求的，请将其代码填在题后的括号内。错选或未选均无分。**

1、若结点的存储地址与其关键字之间存在的某种映射关系，则称这种存储结构为（ ）

A.顺序存储结构 B.链式存储结构

C.索引存储结构 D.散列存储结构

2. 在长度为n的顺序表的第i（1≤i≤n+1）个位置上插入一个元素，元素的移动次数为（ ）

A.n-i+1 B.n-i C.i D.i-1

3. 对于只在表的首.尾两端进行插入操作的线性表，宜采用的存储结构为（ ）

A.顺序表 B.用头指针表示的单循环链表

C.用尾指针表示的单循环链表 D.单链表

4. 若进栈序列为a，b，c，则通过入出栈操作可能得到的a，b，c的不同排列个数为（ ）

A.4 B.5 C.6 D.7

5. 为查找某一特定单词在文本中出现的位置，可应用的串运算是（ ）

A.插入 B.删除 C.串联接 D.子串定位

6. 已知函数Sub(s,i,j)的功能是返回串s中从第i个字符起长度为j的子串，函数Scopy(s,t)的功能为复制串t到s。若字符串S=“SCIENCESTUDY”，则调用函数Scopy(P,Sub(S,1,7)后得到（ ）

A.P=“SCIENCE” B.P=“STUDY” C.S=“SCIENCE ” D.S=“STUDY”

7. 三维数组h[4][5][6]按行优先存储方法存储在内存中，若每个元素占2个存储单元，且数组中第一个元素的存储地址为120，则元素A[3][4][5]的存储地址为（ ）

A.356 B.385 C.360 D.362

p

8. 如右图所示广义表是一种（ ）

A.线性表 B.纯表 C.结点共享表 D.递归表

9. 下列陈述中正确的是（ ）

A.二叉树是度为2的有序树 B.二叉树中结点只有一个孩子时无左右之分

C.二叉树中必有度为2的结点 D.二叉树中最多只有两棵子树，并且有左右之分

10. n个顶点的有向完全图中含有有向边的数目最多为（ ）

b

a

d

c

e

f

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

11. 已知一个有向图如右所示，

则从顶点a出发进行深度优先遍历，

不可能得到的DFS序列为（ ）

A.a d b e f c B.a d c e f b

C.a d c b f e D.a d e f c b

12. 在最好和最坏情况下的时间复杂度均为O(nlogn)且稳定的排序方法是（ ）

A.快速排序 B.堆排序 C.归并排序 D.基数排序

1

2

3

5

4

13. 不可能生成右图所示二叉排序树的关键字序列是（ ）

A.4 5 3 1 2 B.4 2 5 3 1 C.4 5 2 1 3 D.4 2 3 1 5

14. AVL树是一种平衡的二叉排序树，树中任一结点的（ ）

A.左.右子树的高度均相同 B.左.右子树高度差的绝对值不超过1

C.左子树的高度均大于右子树的高度 D.左子树的高度均小于右子树的高度

15. 在VSAM文件的控制区间中，记录的存储方式为（ ）

A.无序顺序 B.有序顺序 C.无序链接 D.有序链接

**第二部分 非选择题（共70分）**

**二.填空题（本大题共10小题，每小题2分，若有两个空格，每个空格1分，共20分）不写解答过程，将正确的答案写在每小题的空格内。错填或不填均无分。**

16. 若一个算法中的语句频度之和为T(n)=3720n+4nlogn，则算法的时间复杂度为 。

17. 在如图所示的链表中，若在指针p所指的结点之后插入数据域值相继为a和b的两个结点，则可用下列两个语句实现该操作，它们依次是 和 。

b

a

p

s

Λ

题17图

18. 假设以S和X分别表示进栈和退栈操作，则对输入序列a,b,c,d,e进行一系列栈操作SSXSXSSXXX之后，得到的输出序列为 。

19. 串S=“I am a worker”的长度是 。

20. 假设一个10阶的下三角矩阵A按列优先顺序压缩存储在一维数组C中，则C数组的大小应为 。

21. 在n个结点的线索二叉链表中，有 个线索指针。

22. 若采用邻接矩阵结构存储具有n个顶点的图，则对该图进行广度优先遍历的算法时间复杂度为 。

23. 对关键字序列（52，80，63，44，48，91）进行一趟快速排序之后得到的结果为 。

24. 由10000个结点构成的二叉排序树，在等概率查找的假设下，查找成功时的平均查找长度的最大值可能达到 。

25. 若要找出所有工资低于1500元，职称是副教授，及所有工资低于2000元，职称是教授的记录，则查询条件是 。

**三.解答题（本大题共4小题，每小题5分，共20分）**

26. 已知一个6行5列的稀疏矩阵中非零元的值分别为：-90，41，-76，28，-54，65和-8，它们在矩阵中的列号依次为：1，4，5，1，2，4和5。当以带行表的三元组表作存储结构时，其行表RowTab中的值依次为0，0，2，2，3和5。请写出该稀疏矩阵（注：矩阵元素的行列下标均从1开始）。

27. 已知树T的先序遍历序列为ABCDEFGHIJKL后序遍历序列为CBEFDJIKLHGA。请画出树T。

28. 对关键字序列（72，87，61，23，94，16，05，58）进行堆排序，使之按关键字递减次序排列。请写出排序过程中得到的初始堆和前三趟的序列状态。

初始堆：

第1趟：

第2趟：

第3趟：

29. 在关键字序列（07，12，15，18，27，32，41，92）中用二分查找法查找和给定值92相等的关键字，请写出查找过程中依次和给定值“92”比较的关键字。

**四.算法阅读题（本大题共4小题，每小题5分，共20分**）

30. 以下函数中，h是带头结点的双向循环链表的头指针。

（1）说明程序的功能 ；

（2）当链表中结点数分别为1和6（不包括头结点）时，请写出程序中while循环体的执行次数。

int f (DlistNode \* h)

{

DlistNode \* p, \* q;

int j=1;

p=h-＞nest;

q=h-＞prior;

while(p != q && p-＞prior != q)

if (p-＞data==q-＞data)

{

p=p-＞next;

q=q-＞prior;

}

else j =0;

return j;

}

31. 设栈S=（1，2，3，4，5，6，7），其中7为栈顶元素。请写出调用algo（&S）后栈S的状态。

void algo (Stack \*S)

{

int i =0;

Queue Q; Stack T;

InitQueue(&Q); InitStack(&T);

while (!StackEmpty(S))

{

if ((i=!i) !=0) Push(&T,Pop(&S));

else EnQueue(&Q, Pop(&S));

}

while (! Queue Empty(Q))

Push(&S,DeQueue(&Q));

while (!StackEmpty(T))

Push (&S,Pop(&T));

}

32. 已知带权图的邻接矩阵表示和邻接表表示的形式说明分别如下：

# define MaxNum 50 //图的最大顶点数

# define INFINITY INT\_MAX //INT\_MAX为最大一整数，表示∞

typedef struce{

char vexs[MaxnNum]; //字符类型的顶点表

int edges[MaxNum][MaxNum]; //权值为整型的邻接矩阵

int n, e; //图中当前的顶点数和边数

} MGraph; //邻接矩阵结构描述

typedef struct node{

int adjvex; //邻接点域

int weight; //边的权值

struct node \*next; //链指针域

} EdgeNode \*next; //边表结点结构描述

typedef struct{

char vertex; //顶点域

EdgeNode\* firstedge; //边表头指针

} VertexNode; //顶点表结点结构描述

typedef struct{

VertexNode adjlist[MaxNum]; //邻接表

int n, e; //图中当前的顶点数和边数

} ALGraph; //邻接表结构描述

下列算法是根据一个带权图的邻接矩阵存储结构G1建立该图的邻接表存储结构G2，请填入合适的内容，使其成为一个完整的算法。

void convertM(MGraph \* G1, ALGraph \*G2)

{

int i, j;

EdgeNode \*p;

G2-＞n=G1-＞n;

G2-＞e=G1-＞e;

for (i=0; i＜G1-＞n; i++ )

{

G2-＞adjlist[i].vertex=G1-＞vexs[i];

G2-＞adjlist[i].firstedge= (1) ;

}

for (i=0; i＜G1-＞n; i++ )

for (j=0; j＜G1-＞n; j++ )

if (G1-＞deges[i][j]＜INFINITY

{

p=(Edge Node \*)malloc (sizeof (EdgeNode));

p-＞weight= (2) ;

p-＞adjvex=j;

p-＞next=G2-＞adjlist[i].firstedge;

(3) ;

}

}

33.阅读下列算法，并回答下列问题：

（1）该算法采用何种策略进行排序？

（2）算法中R[n+1]的作用是什么？

typedef struct {  
 KeyType key;

infoType otherinfo;

}nodeType;

typedef nodeType SqList [MAXLEN];

void sort (SqList R, int n)

{

// n 小于MAXLEN-1

int k, i;

for (k=n-1; k＞=1; k--)

if (R[k].key＞R[k+1].key)

{

R[n+1]=R[k];

for (i=k+1; R[i].key＜R[n+1].key; i++)

R[i-1]=R[i];

R[i-1]=R[n+1];

}

}

**三.算法设计题（本题共10分）**

34. 假设二叉树T采用如下定义的存储结构：

typedef struct node{

DateType data;

Struct node \* lchild, \*rchild, \*parent;

} PBinTree;

其中，结点的lchild域和rchild域已分别填有指向其左.右孩子结点的指针，而parent域中的值为空指针（拟作为指向双亲结点的指针域）。请编写一个递归算法，将该存储结构中各结点的parent域的值修改成指向其双亲结点的指针。

**全真试题（三）**

**本试卷分两部分，第一部分为选择题，1页至3页，第二部分为非选择题，4页至12页，共12页；选择题30分，非选择题70分，满分100分。考试时间150分钟。**

**第一部分 选择题（共30分）**

**一、单项选择题（本大题共15小题，每小题2分，共30分）在每小题列出的四个选项中只有一个是符合题目要求的，请将其代码填在题后的括号内。错选或未选均无分。**

1.下面程序段的时间复杂度是（ ）

for (i=0; i＜n; i++)

for (j=1; j＜m; j++)

A[i][j]=0;

A.O (n) B.O(m+n-1) C.O(m+n) D.O(m\*n)

2.在单链表中，指针p指向元素为x的结点，实现“删除x的后继”的语句是（ ）

A.p=p-＞next; B.p-＞next =p-＞next -＞next ;

C.p-＞next =p; D.p=p-＞next-＞next;

3.在头指针为head且表长大于1的单循环链表中，指针p 指向表中某个结点，若p-＞next-＞next=head，则（ ）

A.p指向头结点 B.p指向尾结点

C.\*p的直接后继是头结点 D.\*p的直接后继是尾结点

4.判定“带头结点的链队列为空”的条件是（ ）

A.Q.front ==NULL B.Q.REAR==NULL

C.Q.front==Q.rear D.Q.front!=Q.rear

5.设有两个串T和P，求P在T中首次出现的位置的串运算称作（ ）

A.联接 B.求子串 C.字符定位 D.子串定位

6.广义表A=(a,(b),(),(c,d,e))的长度为（ ）

A.4 B.5 C.6 D.7

7.一棵含18个结点的二叉树的高度至少为（ ）

A.3 B.4 C.5 D.6

8.已知二叉树的先序序列为ABDECF，中序序列为DBEAFC，则后序序列为（ ）

A.DEBAFC B.DEFBCA C.DEBCFA D.DEBFCA

9.无向图中一个顶点的度是指图中（ ）

A.通过该项点的简单路径数 B.与该顶点相邻接的顶点数

C.通过该顶点的回路数 D.与该顶点连通的顶点数

10.已知一个图如下所示，从顶点a出发

f

a

b

c

d

e

进行广度优先遍历可能得到的序列为（ ）

A.a c e f b d

B.a c b d f e

C.a c b d e f

D.a c d b f e

11.在下列排序方法中，平均时间性能为O(nologn)且空间性能最好的是（ ）

A.快速排序 B.堆排序 C.归并排序 D.基数排序

12.已知一组关键字为{25，48，36，72，79，82，23，40，16，35}，其中每相邻两个为有序子序列。对这些子序列进行一趟两两归并的结果是（ ）

A.{25，36，48，72，23，40，79，82，16，35}

B.{25，36，48，72，16，23，40，79，82，35}

C.{25，36，48，72，16，23，35，40，79，82}

D.{16，23，25，35，36，40，48，72，79，82}

13.设顺序存储的线性表共有123个元素，按分块查找的要求等分成3块。若对索引表采用顺序查找来确定块，并在确定的块中进行顺序查找，则在查找概率相等的情况下，分块查找成功时的平均查找长度为（ ）

A.21 B.23 C.41 D.62

14.索引非顺序文件的特点是（ ）

A.主文件无序，索引表有序 B.主文件有序，索引表无序

C.主文件有序，索引表有序 D.主文件无序，索引表无序

15.倒排文件的主要优点是（ ）

A.便于进行插入和删除运算 B.便于进行文件的恢复

C.便于进行多关键字查询 D.节省存储空间

**第二部分 非选择题（共70分）**

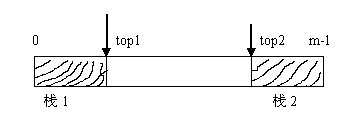
**二、填空题（本大题共10小题，每小题2分，若有两个空格，每个空格1分，共20分）不写解答过程，将正确的答案写在每小题的空格内。错填或不填均无分。**

16.抽象数据类型的特点是将 和 封装在一起，从而实现信息隐藏。

17.从顺序表中删除一个元素时，表中所有在被删元素之后的元素均需 一个位置。

18.在队列中，允许进行插入操作的一端称为 ，允许进行删除操作的一端称为 。

19.如图两个栈共享一个向量空间，top1和top2分别为指向两个栈顶元素的指针，则“栈满”的判定条件是 。



20.设S1=“good”，S2=“ ”，S3=“book”，则S1，S2和S3依次联接后的结果是 。

21.假设三维数组A[10][9][8]按行优先顺序存储，若每个元素占3个存储单元，且首地址为100，则元素A[9][8][7]的存储地址是 。

22.已知在一棵含有n个结点的树中，只有度为k的分支结点和度为0的叶子结点，则该树中含有的叶子结点的数目为 。

23.能够成功完成拓扑排序的图一定是一个 。

24.如果在排序前，关键字序列已接近正序或逆序，则在堆排序和快速排序两者之中，选用

较为适当。

25.假设哈希表的表长 m，哈希函数为H(key)，若用线性探查法解决冲突，则探查地址序列的形式表达为 。

**三、解答题（本大题共4小题，每小题5分，共20分）**

26.假设通信电文使用的字符集为{a,b,c,d,e,f}，各字符的在电文中出现的频度分别为：34，5，12，23，8，18，试为这6个字符设计哈夫曼编码。请先画出你所构造的哈夫曼树（要求树中左孩子结点的权值小于右孩子结点的权值），然后分别写出每个字符对应的编码。

27.已知一个图如下所示，其顶点按a.b.c.d.e.f顺序

f

b

a

c

d

e

存放在邻接表的顶点表中，请画出该图的邻接表，

使得按此邻接表进行深度优先遍历时得到的顶点

序列为a.c.b.e.f.d，进行广度优先遍历时得到的

顶点序列为a c b d f e。

28.已知两个4×5的稀疏矩阵的三元组表分别如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 4 | 16 |
| 1 | 2 | 2 | 18 |
| 2 | 3 | 4 | -25 |
| 3 | 4 | 2 | 28 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 32 |
| 1 | 2 | 2 | -22 |
| 2 | 2 | 5 | 69 |
| 3 | 3 | 4 | 25 |
| 4 | 4 | 2 | 51 |

请画出这两个稀疏矩阵之和的三元组表。

29.从空树起，依次插入关键字40，8，90，15，62，95，12，23，56，32，构造一棵二叉排序树。

（1）画出该二叉排序树

（2）画出删去该树中元素值为90的结点之后的二叉排序树。

**四、算法阅读题（本大题共4小题，每小题5分，共20分）**

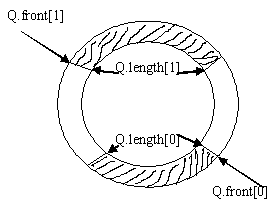
30.如图所示，利用同一循环向量空间实现两个队列，其类型Queue2定义如下：

typedef sturct{

DataType data[MaxSize];

int front [2],length[2];

} Queue2;

对于i=0或1， front[i]和length[i]分别为第i个队列的头指针和长度域。请在空缺处填入合适的内容， 

实现第i个循环队列的入队操作。

int EnQueue (Queue2 \*Q, int i, DataType x)

{ // 若第i个队列不满，则元素x入队列，并返回1，否则返回0

if (i＜0 || i＞1)return 0;

if ( (1) )

return 0;

Q-＞data[ (2) ]=x;

Q-＞length[ (3) ]++;

return 1;

}

31.某二叉树的线索链表存储结构如图（b）所示，其中 p为指向根结点的指针，图(a)为结点结构。阅读下列算法，并回答问题：

lchild

rtag

rchild

data

(a)

BA

A

P

DA

FA

CA

EA

GA

HA

Λ

Λ

Λ

Λ

Λ

Λ

1Λ

1Λ

0Λ

0Λ

0Λ

0Λ

10Λ

10Λ

（1）写出执行函数调用f(p)的输出结果；

（2）简述函数f 的功能。

void f (BinThrTree t)

{

while (t)

{

printf(t-＞date);

if (t-＞lchild)

t=t-＞lchild;

else

t=t-＞rchild;

}

}

（1）

（2）

32.下列函数FindCycle (G, i)的功能是，对一个采用邻接表作存储结构的有向图G，利用深度优先搜索策略寻找一条经过顶点Vi的简单回路。数据cycle\_path用于保存搜索过程中形成的回路，cycle\_path[k]=j(j≥0)表示在回路中顶点Vk的下一个顶点Vj。请在空缺处填入合适的内容，使其成为一个完整的算法。

已知邻接表的顶点表结点结构为：

|  |  |
| --- | --- |
| vertex | firstedge |

边表结点EdgeNode结构为：

|  |  |
| --- | --- |
| adjvex | next |

int cycle\_ path[MaxNurm];

int FindCycle (ALGraph \* G, int i)

{ //若回路存在，则返回1， 否则返回0

int j;

for (j=0; j＜G-＞n; j++) cycle\_path[j]=-1;

return DFSPath (G, i, i );

}

int DFSPath (ALGraph \*G, int j, int i )

{

EdgeNode \* p;

Int cycled=0;

for (p=G-＞adjlist[j].firstedge; p &&! cycled; p=p-＞next)

{

cycle\_path[j]=p-＞adjvex;

if ( (1) )cycled=1; //已找到回路

else

if (cycle\_path[p-＞adjvex]==-1) cycled= (2) ;

}

return (3) ;

}

33.阅读下列函数algo，并回答问题。

（1）假设整型数组A[1..8]中的元素依次为（3，8，9，1，7，4，2，6）。执行函数调用algo（A，8）时，外层while的循环体执行多少次？函数的返回值是多少？

（2）简述函数algo（L，n）的功能。

int algo (int L[ ], int n)

{

int i=0, j, s=1, t=n;

while (i! =(n+1)/2)

{

int x =L[s];

i=s; j=t;

while (i＜j)

{

while (i＜j &&L[j]＞=x)j--;

L[i]=L[j];

while (i＜j &&L[i]＜=x)i++;

L[j]=L[i];

}

L[i]=x;

if (x＜(n+1)/2) s=i+1;

else t =i-1;

}

if (i ==0) return 0;

else return L[i];

}

(1)

(2)

**五、算法设计题（本题共10分）**

34.假设以带头结点的单循环链表作非递减有序线性表的存储结构。请设计一个时间复杂度为0（n）的算法，删除表中所有数值相同的多余元素，并释放结点空间。例如：

（7，10，10，21，30，42，42，42，51，70）

经算法操作后变为（7，10，21，30，42，51，70）

**全真试题（四）**

**本试卷分两部分，第一部分为选择题，1页至2页，第二部分为非选择题，3页至12页，共12页；选择题30分，非选择题70分，满分100分。考试时间150分钟。**

**第一部分 选择题（共30分）**

**一、单项选择题（本大题共15小题，每小题2分，共30分）在每小题列出的四个选项中只有一个是符合题目要求的，请将其代码填在题后的括号内。错选或未选均无分。**

1.下列各式中，按增长率由小至大的顺序正确排列的是（ ）

A.,n!,2n,n3/2 B.n3/2,2n,nlogn,2100 C.2n, logn, nlogn, n3/2 D.2100, logn, 2n, nn

2.若要在单链表中的结点\*p之后插入一个结点\*s，则应执行的语句是（ ）

A.s-＞next=p-＞next; p-＞next=s; B.p-＞next=s; s-＞next=p-＞next

C.p-＞next=s-＞next; s-＞next=p; D.s-＞next=p;p-＞next=s-＞next;

3.若要在O（1）的时间复杂度上实现两个循环链表头尾相接，则应对两个循环链表各设置一个指针，分别指向（ ）

A.各自的头结点 B.各自的尾结点

C.各自的第一个元素结点 D.一个表的头结点，另一个表的尾结点

4.栈的两种常用存储结构分别为（ ）

A.顺序存储结构和链式存储结构 B.顺序存储结构和散列存储结构

C.链式存储结构和索引存储结构 D.链式存储结构和散列存储结构

5.已知循环队列的存储空间为数组data[21]，且当前队列的头指针和尾指针的值分别为8和3，则该队的当前长度为（ ）

A.5 B.6 C.16 D.17

6.已知在如下定义的链串结点中，每个字符占1个字节，指针占4个字节，则该链串的存储密度为（ ）

typedef struct node{

char date[8];

struct node \* next;

} LinkStrNode;

A.1/4 B.1/2 C.2/3 D.3/4

7.应用简单的匹配算法对主串s=“BDBABDABDAB”与子串t=“BDA”进行模式匹配，在匹配成功时，进行的字符比较总次数为（ ）

A.7 B.9 C.10 D.12

8.二维数组A[20][10]采用列优先的存储方法，若每个元素占2个存储单元，且第1个元素的首地址为200，则元素A[8][9]的存储地址为（ ）

A.574 B.576 C.578 D.580

9.对广义表L=((a,b),c,d)进行操作tail (head (L))的结果是（ ）

A.(c,d ) B.(d ) C.b D.(b)

10.已知一棵树的前序序列为ABCDEF，后序序列为CEDFBA，则对该树进行层次遍历得到的序列为（ ）

A.ABCDEF B.ABCEFD C.ABFCDE D.ABCDFE

11.一个含n个顶点和e条弧的有向图以邻接矩阵表示法为存储结构，则计算该有向图中某个顶点出度的时间复杂度为（ ）

A.O(n) B.O(e) C.O(n+e) D.O(n2)

12.在关键字序列（12，23，34，45，56，67，78，89，91）中二分查找关键字为45，89和12的结点时，所需进行的比较次数分别为（ ）

A.4，4，3 B.4， 3， 3 C.3，4，4 D.3，3，4

13.下列排序方法中，最好与最坏时间复杂度不相同的排序方法是（ ）

A.冒泡排序 B.直接选择排序 C.堆排序 D.归并排序

14.已知含10个结点的二叉排序树是一棵完全二叉树，则该二叉排序树在等概率情况下查找成功的平均查找长度等于（ ）

A.1.0 B.2.9 C.3.4 D.5.5

15.在下列各种文件中，不能进行顺序查找的文件是（ ）

A.顺序文件 B.索引文件 C.散列文件 D.多重表文件

**第二部分 非选择题（共70分）**

**二、填空题（本大题共10小题，每小题2分，共20分）请在每小题的空格中填上正确答案。错填、不填均无分。**

16.抽象数据类型是指数据逻辑结构及与之相关的 。

17.已知在结点个数大于1的单循环链表中，指针p指向表中某个结点，则下列程序段执行结束时，指针q指向结点\*p的 结点。

q=p;

while (q-＞next!=p)q=p-＞next;

18.假设S和X分别表示进栈和出栈操作，由输入序列“ABC”得到输出序列“BCA”的操作序列为SSXSXX，则由“a\*b+c/d”得到“ab\*cd/+”的操作序列为 。

19.在文本编辑程序中查找某一特定单词在文本中出现的位置，可以利用串的 运算。

20.假设以行优先顺序将一个n阶的5对角矩阵压缩存储到一维数组Q中，则数组Q的大小至少为 。

21.在含100个结点的完全二叉树中，叶子结点的个数为 。

22.在无向图中，若从顶点a到顶点b存在 ，则称a与b之间是连通的。

23.如果排序过程不改变 之间的相对次序，则称该排序方法是稳定的。

24.索引顺序查找适宜对 的顺序表进行查找。

25.文件的检索操作可按检索条件不同分为下列四种询问，它们是简单询问、范围询问、函数询问及 。

F

A

B

C

D

E

K

**三、解答题（本大题共4小题，每小题5分，共20分）**

26.画出下图所示二叉树的中序线索链表的存储表示。

27.已知图G=（V，E），其中：

V={a,b,c,d,e},

E={(a,b),(b,d),(c,b),(c,d),(d,e),(e,a),(e,c)}。

（1）画出图G；

（2）画出图G的邻接表。

28.已知自顶向下的二路归并排序的算法如下所示，按此算法对关键字序列（55，28，73，91，37，64，19，82，46）进行排序，列出算法执行过程中前5次调用Merge函数进行归并之后的关键字序列。

void MergeSortDC(SeqList R, int low, int high)

{//用分治法对R[low.. high]进行二路归并排序

int mid;

if (low＜high＝{ //区间长度大于1

mid=(low+high)/2 //分解

MergeSortDC(R,low,mid); //递归地对R[low..mid]排序

MergeSortDC(R,mid+1,high); //递归地对R[mid+1..high]排序

Merge(R,low,mid,high); //组合，将两个有序区归并为一个有序区

}

}// MergeSortDC

29.由于元素的插入先后次序不同，所构成的二叉排序树可能有多种形态。请画出4棵含1，2，3，4，5，6六个元素且以1为根、深度为4的二叉排序树。

**四、算法阅读题（本大题共4小题，每小题5分，共20分）**

30.L为一个带头结点的循环链表。函数f30的功能是删除L中数据域data的值大于c的所有结点，并由这些结点组建成一个新的带头结点的循环链表，其头指针作为函数的返回值。请在空缺处填入合适的内容，使其成为一个完整的算法。

LinkList f30(LinkList L, intc)

{

LinkList Lc,p,pre;

pre=L;

p= (1) ;

Lc=(LinList)malloc(sizeof(ListNode));

Lc-＞next=Lc;

while (p!=L)

if (p-＞data＞c)

{

pre-＞next=p-＞next;

(2) ;

Lc-＞next=p;

p=pre-＞next;

}

else

{

pre=p;

(3) ;

}

return Lc;

}

31.设栈S=（1，2，3，4，5，6，7）其中7为栈顶元素。

（1）写出调用f31(&S)后的S；

（2）简述函数f31中第1个循环语句的功能。

void f31 (Stack \* S)

{

Queue Q;

Stack T;

int i=0;

InitQueue(&Q);

InitStack(&T);

while (!StackEmpty(S))

if ((i=!i)!=0) Push(&T,Pop(S));

else EnQueue(&Q,Pop(S));

while (!StackEmpty(&T))

Push(S,Pop(&T));

while(!QueueEmpty(&Q))

Push(S,DeQueue(&Q));

}

32.图的邻接矩阵表示描述如下：

#define MaxNum 20 //图的最大顶点数

typedef struct{

char vexs[MaxNum]; //字符类型的顶点表

int edges[MaxNum][MaxNum]; //邻接矩阵

int n, e; //图的顶点数和边数

}MGraph; //图的邻接矩阵结构描述

阅读下列算法，并回答问题：

（1）对于下列图G的邻接矩阵，写出函数调用F32（&G,3）的返回值；



（2）简述函数f32的功能；

（3）写出函数f32的时间复杂度。

int f32(MGraph \*G, int i)

{

int d=0, j;

for (j=0; j＜G-＞n; j++)

{

if (G-＞edges[i][j])d++;

if( G-＞edges[j][i])d++;

}

return d;

}

33.阅读下列算法并回答问题：

（1）设数组L[1..8]的初值为（4，-3，7，-1，-2，2，5，-8），写出执行函数调用f33（L，8）之后的L[1..8]中的元素值；

（2）简述函数f33的功能。

void f33(intR[ ],int n)

{

int x =R [1];

int low =1, high=n;

while (low＜high)

{

while (low＜high && R[high]＞0)

high--;

if (low＜high＝

{

R[low++]=R[high];

while (low＜high && R[low] ＜0)

low++;

R[high--]=R[low];

}

}

R[low]=x;

}

**五、算法设计题（本题共10分）**

34.假设二叉链表作为二叉树的存储结构，其特点结构为：

|  |  |  |
| --- | --- | --- |
| lchild | data | Rchild |

依照如下给定的函数f34的原型，编写求二叉树T中叶子结点所在的最小层次与最大层次的函数。其中，参数level为函数执行过程中T当前所指结点的层次，其初值为1；\*lmin与\*lmax分别为叶子结点的最小层次与最大层次，它们的初值均为0。

void f34(BinTree T, int level, int \*lmin, int \*lmax);

**全真试题（五）**

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

1. **填空题（每空1分，共15分）**

1．数据类型是一个 集合和定义在这个集合上的 的总称。

2．算法有五个特征它们是 、 、可行性、输入和输出。

3．一个线性表是n个元素的 。

4．在双向链表的结点中，除了含数据元素域外，还含有 个指针域。

5．栈是限定仅在 进行插入和删除的线性表。不含元素的空表称为 。

6．对于数组来说，一旦确定了它的 和 ，便可以为它分配内存空间。

7．树的 包含一个数据元素及若干指向其子树的分支。

8．满二叉树是一棵深度为k且含有 个结点的二叉树。

9．在有向图G中，如果对于每一对顶点Vi和Vj（Vi≠Vj），从Vi到Vj 和从Vj到Vi都存在路径，则称G为 。

10．根据排序过程中涉及到的存储器不同，可将排序方法分为 和 。

**二．单项选择题（在每个小题的四个备选答案中，选出一个正确的，并将正确的答案的号码添在题干好的括号内。每小题1分，共10分。）**

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

1．单链表结点的数据元素只能是哪一种？ （ ）

A．整型 B．字符串 C．任何数据类型 D．实型

2．若已有一个栈，输入序列为A，B，C，D，E，那么下面哪种序列不可能得到？（ ）

A．ABCDE B．EDCBA C．BAEDC D．ECDBA

3．二叉树后序遍历的次序是什么？ （ ）

A 根、左子树、右子树 B左子树、根、右子树

C 左子树、右子树、根 D根、右子树、左子树

4．已给如图二叉树，它的中序遍历是哪一项？ （ ）

F

E

D

B

A

C

A．ABECDF B．ABCDEF C．CBDAEF D．CDBFEA

5．在串的静态存储结构中，下面的哪一项不是紧缩格式的特点？ （ ）

A．节省空间 B．存取速度慢

C．存取的速度快 D．一个存取单元可能放多个字符

6．已知完全二叉树有26个结点，则整棵二叉树有多少个度为1的结点？ （ ）

A．1 B．0 C．2 D．不确定

7．已给下图，哪一项是该图的拓扑排序？ （ ）

5

1

2

3

4

A．1，2，3，4，5 B．1，3，2，4，5

C．1，2，4，3，5 D．1，2，3，5，4

8．在常用的哈希表处理冲突的方法中，哪一种方法容易产生“二次聚集” （ ）

A．开放定址法 B．再哈希法 C．链地址法 E．都不会产生

9．直接插入排序的算法复杂性是多少？ （ ）

A．O（n2） B. O（nlogn） C. O(n) D. O(logn)

10.串联文件的记录之间有何关系?

A 相继的两个物理记录的存储位置相邻

B．物理记录之间的次序由指针相联

C．两个逻辑相邻的记录物理上相邻

D．都不是

**三、简释名词（每小题3分，共15分）**

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

1. 线索二叉树
2. 拓扑排序
3. 关键路径
4. 堆排序
5. 直接存取文件

**四、简答题（每小题5分，共30分）**

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

1. 已给一个栈S，写出对S的所有操作。
2. 以数据集{3，4，5，8，12，18}为叶结点的权值，构造一棵哈夫曼树。

5

1

3

2

4

6

12

2

3

4

6

7

5

1. 已给右图

写出其邻接矩阵，并画出从顶点①

开始的最小生成树。

1. 已给输入序列{17，60，29，38，42，50}，按除留余数法，填写如下哈希表。其中除留余数法的公式如下：

H（key）=key % 11

当出现冲突时，按Hi=（H（key）+1）MOD 11 的线性探测再散列的方法进行。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 关键字 |  |  |  |  |  |  |  |  |  |  |  |

1. 已给有8个的无序序列：{49，38，65，97，76，13，27，49}，画出建立初始堆的过程的示例图。
2. 已给无向图如下：

5

1

4

2

3

6

要求：写出该图从顶点①

开始的广度优先和深度优先搜索

序列，并画出该图的生成树

**五、综合题（用类C语言写出下列各题的算法。每小题10分共30分）**

|  |  |
| --- | --- |
| 得分 | 评卷人 |
|  |  |

1. 设栈用顺序结构实现，写出表示栈的数据结构，并实现如下操作：
   1. void push\_stak(stack s, elemtp x)
   2. void pop\_stack(stack s)

2．.设用邻接矩阵表示一个有向图，编写一个算法，求一个结点的入度和出度的和。

3．设thrt是指向中序双向线索二叉树头结点的指针，写出从thrt起沿后继点进行遍历的算法。

**全真试题（六）**

**一、填空题（每空1分，共15分）**

1 数据元素之间有四种基本结构，它们是集合、 结构、 结构和网状结构。

2 算法的时间度量用 表示，算法的存储空间度量用 表示。

3 每个单线性链表的存取必须从 开始进行。

4 在单链表中，若P和S是两个指针，且满足P->next与S相同，则语句P->next=S->next作用是 S指向的结点。

5 栈的实现有两种存储结构，它们是 结构和 结构。

6 在PASCAL语言中，多维数组在内存中按 为主排列；而在FORTRAN语言中，多维数组按 为主排列。

7 树中结点的 称为树的深度。

8 二叉树的第k层上至多有 个结点。

9 在图的邻接表存储结构中，对图的每一个顶点都建立一个 。

10 在n个元素的有序表中，折半查找给定值至多比较 次，平均查找长度为 。

**二、单项选择题（在每小题的四个备选答案中，选出一个正确的答案，并将其号码填在题干后的括号内，每小题的1分，共10分）**

1. 若线性表最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素，那么采用哪种存储方式最省时间？ ( )

A. 单链表 　　　　 B. 仅有头指针的单循环链表

C. 仅有尾指针的单循环链表　　D. 双链表

2. 利用两个队列，设输入序列为A，B，C，D，E，则下面哪种排列不可能得到？ ( )

A． EDCBA B. ABCDE C. EABCD D. CABDE

3. 哈夫曼树的带权路径长度是什么? ( )

A. 所有结点权值之和　　　　 B. 所有叶结点带权路径长度之和

C. 权结点的值　 　　　　　 D. 除根以外所有结点权值之和

4. 已给如下二叉树，按先序遍历的结果是什么? ( )

1

2

3

4

5

A． 12354

B. 12345

C. 21435

D. 24531

5. 队列和栈都是什么结构? ( )

A. 顺序存储的线性结构 　　　B. 链式存储的线性结构

C. 限制存储结点的线性结构　　D. 限制存储结点的非线性结构

6. 设有一棵22个结点的完全二叉树，那么整棵二叉树有多少个度为0的结点? ( )

A．6 B.7 C.8 D.11

7. 在有向图的邻接表表示中，下面哪一种操作最费时间? ( )

A. 求某顶点的出度　　　　B. 求某顶点的入度

C. 求图中顶点的个数　　　D. 求从某顶点出发的弧

８. 设在二叉排序树上要删除P指向的节点，且设f指向P的父结点，P为f的左孩子,P结点只有左子树,无右子树,那么应做的操作是什么？ ( )

A f->lchild=null 　　　　　B. f->lchild=p->lchild

C.f->lchild=p->rchild D. 都不是

９. 在下列排序方法中，在最好情况下，时间复杂度为O(n)的算法是哪一个？ ( )

A. 简单选择排序　 B.直接插入排序 　 C.冒泡排序　 D.都不是

10 在下列方法中，排序所花时间不受初始数据排列特征影响的算法是哪一个？ ( )

A.直接插入排序　 B. 简单选择排序 　 C. 快速排序　 D.都不是

**三、简释名词（小题3分，共15分）**

1.串

2.广义表

3.连通图的生成树

4.哈希表中的二次聚集

5.记录的逻辑结构

**四、简答题（每小题的5分，共30分）**

1.设有一个三维数组a[C1：d1，C2：d2，C3：d3]，其中Ci：di是第i维的界偶，若该数组在内存中按行排列，且a[C1，C2，C3]在内存中的地址为a0，写出任一下标变量a[i1，i2，i3]的地址。设数组的每个元素占ℓ个单元。

2.设通信中出现5种字符，它们是A，B，C，D，E，出现的频率分别为0.2,0.1,0.3,0.15,0.25。构造其哈夫曼树，并对字符编码。

3.已给一个有向图的邻接表如下：

1

1

2

3

4

5

6

4

3

5

2

2

6

5

5

Λ

Λ

Λ

Λ

Λ

Λ

要求:画出该有向图，并写出3种拓扑排序。

4．已给关键字序列为{13,24,37,90,53}，画出每插入一个关键后所建立的二叉平衡树。

5．已给如下一个序列:

{49,38,65, 97,76,13,27,49,55,4},用希尔排序(shell)，写出每趟排序的结果（增量依次取5，3，1）。

6．已给有向图

V5

V0

V1

V2

V4

V3

10

5

50

20

30

60

100

利用迪杰特拉算法（Dijkstra），,求V0到各顶点的最短距离和路线，即填写如下表格。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 终点 | 从V0到各终点的dist值和最短距离 | | | | |
| V1 |  |  |  |  |  |
| V2 |  |  |  |  |  |
| V3 |  |  |  |  |  |
| V4 |  |  |  |  |  |
| V5 |  |  |  |  |  |

**五、综合题（用类C语言写出下列各题的算法，每小题10分，共30分）**

1.已给单链表的头指针为H，每个节点的数据域data为整型，指针域为next，设计一个算法inserta(H,d)，实现:

若D已在链表H中，则输出“已存在”，然后返回；否则将d插在链表的最后。

data

lc

rc

2. 设二叉树用二叉链表表示，各结点的结构为 ，其中data为整型字

段。设计一个算法，打印出其中data值为正偶数的结点值。要求每个结点不能比孩子结点先打印。

1. 已给有向图，用邻接表表示。写出这种表示的数据结构，并设计一个算法，打印某一个顶点的入度。

rc

lc

lc