2017 ～ 2018学年度第2学期

《数据结构》 复习提纲

一、单项选择题

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 答案 | C | A | D | C | A | B | A | B | C | D |
| 题号 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 答案 | A | A | D | A | A | D | C | B | A | B |

1．在数据结构中，从逻辑上可以把数据结构分为\_\_\_\_\_\_\_\_\_两类。

A．动态结构和静态结构 B．紧凑结构和非紧凑结构

C．线性结构和非线性结构 D．内部结构和外部结构

2．链表不具有的特点是\_\_\_\_\_\_\_\_\_。

A．可随机访问任一元素 B．插入、删除不需要移动的元素

C．不必事先估计存储空间 D．所需空间与线性表长度成正比

3．若线性表最常用的运算是存取第i个元素及其前驱元素，则采用\_\_\_\_\_\_\_\_\_存储方式节省时间。

A．单链表 B．双链表 C．循环单链表 D．顺序表

4．算法分析的目的是\_\_\_\_\_\_\_\_\_。

A．找出数据结构的合理性 B．研究算法中的输入和输出关系

C．分析算法的效率以求改进 D．分析算法的易读性和文档性

5．若一个栈用数组data[1..n]存储，初始栈顶指针top为0，则以下元素x进栈的操作正确的是\_\_\_\_\_\_\_\_\_。

A．top++; data[top]=x; B．data[top]=x; top++;

C．top--; data[top]=x; D．data[top]=x; top--;

6．表达式a\*(b+c)-d的后缀表达式是\_\_\_\_\_\_\_\_\_。

A．abcd\*+- B．abc+\*d- C．abc\*+d- D．-+\*abcd

7．递归函数f(1)=1，f(n)=f(n-1)+n(n＞1)的递归出口是\_\_\_\_\_\_\_\_\_。

A．f(1)=1 B．f(1)=0 C．f(0)=0 D．f(n)=n

8．将递归算法转换成对应的非递归算法时，通常需要使用\_\_\_\_\_\_\_\_\_保存中间结果。

A．队列 B．栈 C．链表 D．树

9．对稀疏矩阵采用压缩存储，其缺点之一是\_\_\_\_\_\_\_\_\_。

A．无法判断矩阵有多少行、多少列

B．无法根据行、列号查找某个矩阵元素

C．无法根据行、列号直接计算矩阵元素的存储地址

D．使矩阵元素之间的逻辑关系更加复杂

10．一个n阶上三角矩阵a按行优先顺序压缩存放在一维数组b中，则b中的元素个数是\_\_\_\_\_\_\_\_\_。

A．n B．n2 C．n(n+1)/2 D．n(n+1)/2+1

11．度为4，高度为h的树\_\_\_\_\_\_\_\_\_。

A．至少有h+3个结点 B．最多有4h-1个结点

C．最多有4h个结点 D．至少有h+4个结点

12．用双亲存储结构表示树，其优点之一是比较方便\_\_\_\_\_\_\_\_\_。

A．找指定结点的双亲结点 B．找指定结点的孩子结点

C．找指定结点的兄弟结点 D．判断某结点是不是叶子结点

13．设有13个值，用它们组成一棵哈夫曼树，则该哈夫曼树共有\_\_\_\_\_\_\_\_\_个结点。

A．13 B．12 C．26 D．25

14．无向图的邻接矩阵是一个\_\_\_\_\_\_\_\_\_。

A．对称矩阵 B．零矩阵 C．上三角矩阵 D．对角矩阵

15．在图的广度优先遍历算法中用到一个队列，每个顶点最多进队\_\_\_\_\_\_\_\_\_次。

A．1 B．2 C．3 D．不确定

16．在用Prim和Kruskal算法构造最小生成树时，前者更适合于\_\_\_\_\_\_\_\_\_。

A．有向图 B．无向图 C．稀疏图 D．稠密图

17．有一个有序表R[1..13]={1, 3, 9, 12, 32, 41, 45, 62, 75, 77, 82, 95, 100}，当用二分查找法查找值为82的节点时，经过\_\_\_\_\_\_\_\_\_次比较后查找成功。

A．1 B．2 C．4 D．8

18．在采用分块查找时，若线性表中共有625个元素，查找每个元素的概率相同，假设采用顺序查找来确定结点所在的块，则每块分为\_\_\_\_\_\_\_\_\_个结点最佳。

A．9 B．25 C．6 D．625

19．若R中有10000个元素，如果仅要求求出其中最大的10个元素，则采用\_\_\_\_\_\_\_\_\_方法最节省时间。

A．堆排序 B．希尔排序 C．快速排序 D．基数排序

20．有一组序列（48, 36, 68, 99, 75, 24, 28, 52）进行快速排序，要求结果从小到大排序，则进行一次划分之后的结果为\_\_\_\_\_\_\_\_\_。

A．[24 28 36] 48 [52 68 75 99] B．[28 36 24] 48 [75 99 68 52]

C．[36 68 99] 48 [75 24 28 52] D．[28 36 24] 48 [99 75 68 52]

二、填空题

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题号 | 答案 | 题号 | 答案 | 题号 | 答案 |
| 1 | 问题规模 | 2 | O(n) | 3 | 假溢出 |
| 4 | 285 | 5 | 4 | 6 | 2h-1 |
| 7 | 关键活动 | 8 | 无环图 | 9 | RL |
| 10 | 基数排序 |  |  |  |  |

1．在分析算法的时间复杂度时，通常认为算法的执行时间是\_\_\_\_\_\_\_\_\_的函数。

2．求一个双链表长度的算法的时间复杂度为\_\_\_\_\_\_\_\_\_。

3．在实现顺序队的时候，通常将数组看成是一个首尾相连的环，这样做的目的是为了避免产生\_\_\_\_\_现象。

4．有如下递归过程：

void reverse( int m )

{ printf("%d", n%10);

if( n/10!=0 )

reverse(n/10);

}

调用语句reverse(582)的结果是\_\_\_\_\_\_\_\_\_。

5．广义表(((a, b,( ), c), d), e, ((f), g))的深度是\_\_\_\_\_\_\_\_\_。

6．在高度为h（h≥0）的二叉树中最多有\_\_\_\_\_\_\_\_\_个结点。

7．AOE网中从源点到汇点长度最长的路径称为关键路径，该路径上的活动称为\_\_\_\_\_\_\_\_\_。

8．可以进行拓扑排序的有向图一定是\_\_\_\_\_\_\_\_\_。

9．输入序列为（20， 35， 30，……），构造一棵平衡二叉树，其中的第一次调整为\_\_\_\_\_\_\_\_\_型调整。

10．在排序过程中，任何情况下都不比较关键字大小的排序方法是\_\_\_\_\_\_\_\_\_。

三、判断题

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 答案 | F | T | F | F | F | T | F | T | F | F |

1．如果数据元素值发生改变，则数据的逻辑结构也随之改变。 （ ）

2．在循环单链表中没有为空的指针域。 （ ）

3．顺序栈中元素值的大小是有序的。 （ ）

4．任何递归算法都是尾递归。 （ ）

5．稀疏矩阵的特点是矩阵中元素较少。 （ ）

6．哈夫曼树中不存在度为1的结点。 （ ）

7．完全二叉树中的每个结点或者没有孩子或者有两个孩子。 （ ）

8．强连通分量是有向图中的极大强连通子图。 （ ）

9．哈希冲突是指同一个关键字的记录对应多个不同的哈希地址。 （ ）

10．任何情况下折半插入排序都优于直接插入排序。 （ ）

四、问答题

1．设计一个算法，删除一个单链表L中元素值最大的结点（假设这样的结点唯一）。

void delmaxnode(LinkList \*&L)

{ LinkList \*p,\*pre, \*maxp, \*maxpre;

p=L->next; pre=L;

maxp=p; maxpre=pre;

while (p!=NULL)

{

if (maxp->data<p->data)

{

maxp=p;

maxpre=pre;

}

pre=p;

p=p->next;

}

maxpre->next=maxp->next;

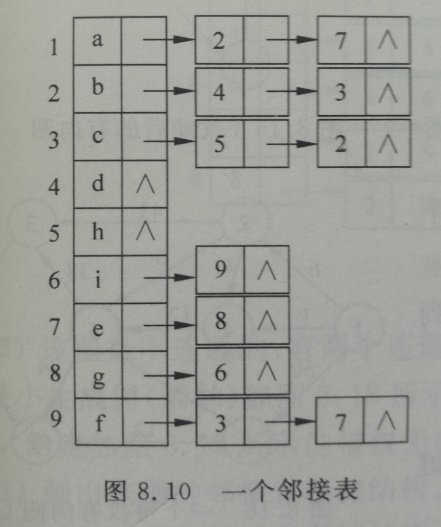
free(maxp);

}

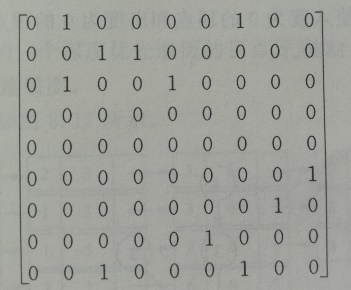
2．已知一棵完全二叉树的第6层（设根为第1层）有8个叶子结点，则该完全二叉树的结点个数最多是多少。

完全二叉树的叶子结点只能在最下两层，对于本题，结点最多的情况是第6层为倒数第二层，即1~6层构成一个满二叉树，其结点总数为26-1=63。第6层有25=32个结点，其中含8个叶子结点，另外有32-8=24个非叶子结点，它们中的每个结点都有两个孩子结点（均为第7层的叶子结点），计48个叶子结点，这样最多的结点个数=63+48=111。

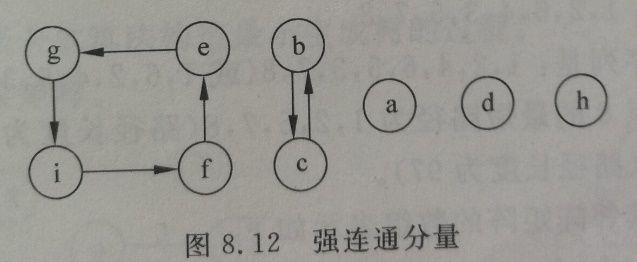
3．一个有向图G的邻接表存储如下图所示，要求：



（1）给出该图的邻接矩阵存储结构；



（2）给出该图的所有强连通分量。



4．什么是平衡二叉树？输入关键字序列（16，3，7，11），给出构造一棵平衡二叉树的过程。

若一棵二叉排序树中每个结点的左右子树的高度最多相差1，则称此二叉树为平衡二叉树。

0

1

0

0

0

0

1

0

0

-1

5．线性表有顺序表和链表两种存储方式，不同的排序方法适合不同的存储结构。对于常见的内部排序方法，说明哪些更适合于顺序表，哪些更适合于链表？哪些两者都适合？

更适合于顺序表的排序方法有希尔排序、折半插入排序、快速排序、堆排序和归并排序。

更适合于链表的排序方法是基数排序

两者都适合的排序方法有直接插入排序、冒泡排序和简单排序。

五、算法设计题

用Floyd算法计算图中任意两个顶点之间的最短路径。

void Floyd(MGraph g)

{

int A[MAXV][MAXV],path[MAXV][MAXV];

int i,j,k;

for (i=0; i<g.n; i++)

for (j=0; j<g.n; j++)

{

A[i][j]=g.edges[i][j];

path[i][j]=-1;

}

for (k=0; k<g.n; k++)

{

for (i=0; i<g.n; i++)

for (j=0; j<g.n; j++)

if (A[i][j]>A[i][k]+A[k][j])

{

A[i][j]=A[i][k]+A[k][j];

path[i][j]=k; //修改最短路径

}

}

Dispath(A,path,g.n); //输出最短路径

}

**一、选择题**

1. 研究数据结构就是研究（ ）。

|  |  |
| --- | --- |
| A 数据的逻辑结构 | B 数据的存储结构 |
| C 数据的逻辑和存储结构 | D 数据逻辑结构、存储结构及其数据在运算上的实现 |

1. 链表不具有的特点是（ ）。

|  |  |
| --- | --- |
| A 可随机访问任一元素 | B 插入删除不需要移动元素 |
| C 不必事先估计存储空间 | D 所需空间与线性表长度成正比 |

1. 设一个栈的输入序列为1，2，3，4，则借助一个栈所得到的输出序列不可能是（ ）。

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

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

|  |  |
| --- | --- |
| A 线性表的顺序存储结构 | B 栈 |
| C 线性表的链式存储结构 | D 队列 |

1. 表达式a\*(b+c)-d的后缀表达式是（ ）。

|  |  |
| --- | --- |
| A abcd\*+- | B abc+\*d- |
| C abc\*+d- | D -+\*abcd |

1. 队列的操作原则是（ ）。

|  |  |
| --- | --- |
| A 先进先出 | B 只能进行插入 |
| C 后进先出 | D 只能进行删除 |

1. 判断带头结点的单链表llist为空的条件是（ ）。

|  |  |
| --- | --- |
| A llist->link==llist | B llist==NULL |
| C llist->link==NULL | D llist!=NULL |

1. 一个具有20个结点的完全二叉树，其叶结点个数为（ ）。

|  |  |
| --- | --- |
| A 9 | B 10 |
| C 11 | D 12 |

1. 下列不属于内部排序的算法是（ ）。

|  |  |
| --- | --- |
| A 归并排序 | B 拓扑排序 |
| C 选择排序 | D 插入排序 |

1. 在一棵度为3的树中，度为3的结点数为2，度为2的结点数为1，则叶结点数为（ ）。

|  |  |
| --- | --- |
| A 4 | B 5 |
| C 6 | D 7 |

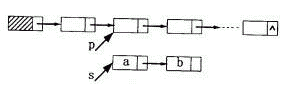
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| D | A | C | B | B | A | C | B | B | C |

二、填空题

1. 已知二叉树有50个叶子结点，则该二叉树的总结点数至少是 99 。
2. 在一个长度为n的线性表的第i个元素（1≦i≦n）之前插入一个元素时，插入函数需向后移动 n-i+1 个元素。
3. 若频繁地对线性表进行插入与删除操作，该线性表应采用 链式 存储结构。
4. 若某线性表采用顺序存储结构，每个元素占2个存储单元，首地址为是100，则第5个元素的地址是 108 。
5. 若结点y是结点x的一棵子树的根，则x称作y的 父结点 。
6. 一个串中包括的字符个数称作这个串的 长度 。
7. 一棵树高为h的完全二叉树至少有2h-1个结点，至多有 2h-1 个结点。
8. 算法的复杂性分析主要从算法的时间复杂性和 空间 复杂性进行考虑。
9. 数据结构主要根据 逻辑 结构和存储结构进行分类。
10. 图的遍历分为两种类型：深度优先遍历和 广度 优先遍历。

三、算法分析与设计题

1. 在链表中，若已知指针p，要在指针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，已知指针s指向结点a。写出实现上述插入操作的关键语句。



s->next->next=p->next;

p->next=s;

1. 如图所示的一棵二叉树，写出对此树的先根序列、中根序列和后根序列。

先根序列：ABDEFC 中根序列：DEFBAC 后根序列：FEDBCA

1. 以{5，6，7，8，9，10，15，18，22}作为叶结点的权值来构造一棵Huffman树。

22

9

10

6

5

8

7

18

15

1. 请设计一个算法，求出循环表中结点的个数。

int Countnode(ClinkList clist) {

int n；

PNode p；

p=clist->link；

n=0；

while(p!=clist) {

n++；

p=p->link； }

return(++n)；}

一、填空题

1．设有一个8阶的对称矩阵A, 采用压缩存储方式, 按行优先顺序存储方式。设a11为第一个元素,其存储地址为1,假设每个元素占用1个存储单元,则a64的存储地址为 19 。

2．若频繁地对线性表进行插入与删除操作，该线性表应采用 链式 存储结构。

3．算法的5个重要特征是 有穷性 、确定性、可行性、输入和输出。

4. 每次从无序子表中取出一个元素，然后把它插入到有序子表的适当位置，则此种排序方法叫做 直接插入 排序。

5．设二维数组A[1..M，1..N]（即M行N列）按行优先顺序存储在一维数组B[1..M\*N]中，则二维数组元素A[i,j]在一维数组B中的下标为 (i-1)\*N+j 。

6．在序列(2,5,8,11,15,16,22,24,27,35,50)中采用折半查找(二分查找)方法查找元素24，需要进行 4 次元素之间的比较。

7．一棵树高为h的完全二叉树至少有 2h-1 个结点，至多有 2h-1 个结点。

8．由权值分别为11,8,6,2,5的叶结点生成一棵Huffman树，它的带权路径长度为 71 。

9．对关键字序列(52,80,63,44,48,91)一趟快速排序后的结果 (48，44，52，63，80，91) 。

二、单项选择题

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 答案 | D | A | A | A | C | B | D | B | A | C |
| 题号 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 答案 | C | A | A | D | D | C | D | A | A | C |

1. 数据结构是指 。

A. 一种数据类型

B. 数据的存储结构

C. 一组性质相同的数据元素的集合

D. 相互之间存在一种或多种特定关系的数据元素的集合

2. 以下算法的时间复杂度为 。

void fun(int n)

{ int i=1;

while (i<=n)

i++;

}

A. O(n) B. O()

C. O(nlog2n) D. O(log2n)

3. 在一个长度为n的有序顺序表中删除元素值为x的元素时，在查找元素x时采用二分查找，此时的时间复杂度为 。

A. O(n) B. O(nlog2n)

C. O(n2) D. O()

4. 在一个带头结点的循环单链表L中，删除元素值为x的结点，算法的时间复杂度为 。

A. O(n) B. O()

C. O(nlog2n) D. O(n2)

5. 若一个栈采用数组s[0..n-1]存放其元素，初始时栈顶指针为n，则以下元素x进栈的正确操作是 。

A.top++;s[top]=x; B.s[top]=x;top++;

C.top--;s[top]=x; B.s[top]=x;top--;

6. 中缀表达式“2\*(3+4)-1”的后缀表达式是 ，其中#表示一个数值的结束。

A. 2#3#4#1#\*+- B. 2#3#4#+\*1#-

C. 2#3#4#\*+1#- D. -+\*2#3#4#1#

7. 设环形队列中数组的下标为0～N-1，其队头、队尾指针分别为front和rear（front指向队列中队头元素的前一个位置，rear指向队尾元素的位置），则其元素个数为 。

A. rear-front B. rear-front-1

C. (rear-front)％N+1 D. (rear-front+N)％N

8. 若用一个大小为6的数组来实现环形队列，队头指针front指向队列中队头元素的前一个位置，队尾指针rear指向队尾元素的位置。若当前rear和front的值分别为0和3，当从队列中删除一个元素，再加入两个元素后，rear和front的值分别为 。

A. 1和5 B. 2和4

C. 4和2 D. 5和1

9. 一棵深度为h（h≥1）的完全二叉树至少有 个结点。

A. 2h-1 B. 2h

C. 2h+1 D. 2h-1+1

10. 一棵含有n个结点的线索二叉树中，其线索个数为 。

A. 2n B. n-1

C. n+1 D. n

11. 设一棵哈夫曼树中有1999个结点，该哈夫曼树用于对 个字符进行编码。

A. 999 B. 998

C. 1000 D. 1001

12. 一个含有n个顶点的无向连通图采用邻接矩阵存储，则该矩阵一定是 。

A. 对称矩阵 B. 非对称矩阵

C. 稀疏矩阵 D. 稠密矩阵

13. 设无向连通图有n个顶点e条边，若满足 ，则图中一定有回路。

A. e≥n B. e<n

C. e=n-1 D. 2e≥n

14. 对于AOE网的关键路径，以下叙述 是正确的。

A. 任何一个关键活动提前完成，则整个工程一定会提前完成

B. 完成整个工程的最短时间是从源点到汇点的最短路径长度

C. 一个AOE网的关键路径一定是唯一的

D. 任何一个活动持续时间的改变可能会影响关键路径的改变

15. 设有100个元素的有序表，用折半查找时，不成功时最大的比较次数是 。

A. 25 B. 50

C. 10 D. 7

16. 在一棵m阶B-树中删除一个关键字会引起合并，则该结点原有 个关键字。

A. 1 B. ⎡m/2⎤

C. ⎡m/2⎤-1 D. ⎡m/2⎤+1

17. 哈希查找方法一般适用于 情况下的查找。

A. 查找表为链表

B. 查找表为有序表

C. 关键字集合比地址集合大得多

D. 关键字集合与地址集合之间存在着某种对应关系。

18. 对含有n个元素的顺序表采用直接插入排序方法进行排序，在最好情况下算法的时间复杂度为 。

A. O(n) B. O(nlog2n)

C. O(n2) D. O()

19. 用某种排序方法对数据序列{24,88,21,48,15,27,69,35,20}进行递增排序，元素序列的变化情况如下：

（1）{24,88,21,48,15,27,69,35,20}

（2）{20,15,21,24,48,27,69,35,88}

（3）{15,20,21,24,35,27,48,69,88}

（4）{15,20,21,24,27,35,48,69,88}

则所采用的排序方法是 。

A. 快速排序 B. 简单选择排序

C. 直接插入排序 D. 归并排序

20. 以下序列是堆的是 。

A. {75,65,30,15,25,45,20,10} B. {75,65,45,10,30,25,20,15}

C. {75,45,65,30,15,25,20,10} D. {75,45,65,10,25,30,20,15}

三、问答题

1. 有一棵二叉排序树按先序遍历得到的序列为：(12,5,2,8,6,10,16,15,18,20)。回答以下问题：

（1）画出该二叉排序树。（2）给出该二叉排序树的中序遍历序列。（3）求在等概率下的查找成功和不成功情况下的平均查找长度。

答：（1）先序遍历得到的序列为：(12,5,2,8,6,10,16,15,18,20)，中序序列是一个有序序列，所以为：(2,5,6,8,10,12,15,16,18,20)，由先序序列和中序序列可以构造出对应的二叉树，如图所示。

（2）中序遍历序列为：2,5,6,8,10,12,15,16,18,20。

（3）ASL成功=(1×1+2×2+4×3+3×4)/10=29/10。

ASL不成功=(5×3+6×4)/11=39/11。



2. 有人提出这样的一种从图G中顶点u开始构造最小生成树的方法：

假设G=(V，E)是一个具有n个顶点的带权连通无向图，T=(U，TE)是G的最小生成树，其中U是T的顶点集，TE是T的边集，则由G构造从起始顶点u出发的最小生成树T的步骤如下：

（1）初始化U={u}。以u到其他顶点的所有边为候选边。

（2）重复以下步骤n-1次，使得其他n-1个顶点被加入到U中。

从候选边中挑选权值最小的边加入到TE，设该边在V-U中的顶点是v，将v加入U中。考查顶点v，将v与V-U顶点集中的所有边作为新的候选边。若此方法求得的T是最小生成树，请予以证明。若不能求得最小边，请举出反例。

答：此方法不能求得最小生成树。例如，对于如图（a）所示的带权连通无向图，按照上述方法从顶点0开始求得的结果为（b）所示的树，显然它不是最小生成树，正确的最小生成树如图（c）所示。

在有些情况下，上述方法无法求得结果，例如对于如图（d）所示的带权连通无向图，从顶点0出发，找到顶点1（边（0,1）），从顶点1出发，找到顶点3（边（1,3）），再从顶点3出发，找到顶点0（边（3,0）），这样构成回路，就不能求得最小生成树了。



求最小生成树的反例图

3. 如果对含有n（n>1）个元素的线性表的运算只有4种：删除第一个元素；删除最后一个元素；在第一个元素前面插入新元素；在最后一个元素的后面插入新元素，则最好使用以下哪种存储结构，并简要说明理由。

（1）只有尾结点指针没有头结点指针的循环单链表（2）只有尾结点指针没有头结点指针的非循环双链表（3）只有头结点指针没有尾结点指针的循环双链表（4）既有头结点指针也有尾结点指针的循环单链表

答：本题答案为（3），因为实现上述4种运算的时间复杂度均为O(1)。

4. 已知一棵度为4的树中，其度为0、1、2、3的结点数分别为14、4、3、2，求该树的结点总数*n*和度为4的结点个数，并给出推导过程。

答：结点总数n=n0+n1+n2+n3+n4，即n=23+n4，又有：度之和=n-1=0×n0+1×n1+2×n2 +3×n3+4×n4，即n=17+4n4，综合两式得：n4=2，n=25。所以，该树的结点总数为25，度为4的结点个数为2。

四、算法设计题

1. 假设二叉树b采用二叉链存储结构，设计一个算法void findparent(BTNode \*b, ElemType x, BTNode \*&p)求指定值为x的结点的双亲结点p。提示：根结点的双亲为NULL；若在b中未找到值为x的结点，p亦为NULL。

void findparent(BTNode \*b,ElemType x,BTNode \*&p)

{ if (b!=NULL)

{ if (b->data==x) p=NULL;

else if (b->lchild!=NULL && b->lchild->data==x)

p=b;

else if (b->rchild!=NULL && b->rchild->data==x)

p=b;

else

{ findparent(b->lchild,x,p);

if (p==NULL)

findparent(b->rchild,x,p);

}

}

else p=NULL;

}

2. 设A和B是两个结点个数分别为m和n的单链表（带头结点），其中元素递增有序。设计一个尽可能高效的算法求A和B的交集，要求不破坏A、B的结点，将交集存放在单链表C中。给出所设计的算法的时间复杂度和空间复杂度。

void insertion(LinkList \*A,LinkList \*B,LinkList \*&C)

{ LinkList \*p=A->next,\*q=B->next,\*s,\*t;

C=(LinkList \*)malloc(sizeof(LinkList));

t=C;

while (p!=NULL && q!=NULL)

{ if (p->data==q->data)

{ s=(LinkList \*)malloc(sizeof(LinkList));

s->data=p->data;

t->next=s;

t=s;

p=p->next;

q=q->next;

}

else if (p->data<q->data)

p=p->next;

else

q=q->next;

}

t->next=NULL;

}

算法的时间复杂度为O(m+n)，空间复杂度为O(MIN(m, n))。