■计算机识别、存储和加工处理的对象被称为( )。
A、数据　　B、数据元素　　C、数据结构　　D、数据类型
■组成数据的基本单位是( )。
A、数据变量　　B、数据项　　C、数据类型　　D、数据元素
■数据对象是指( )。
A、描述客观事物且由计算机处理的数值、字符等符号的总称　　B、数据的基本单位
C、性质相同的数据元素的集合　　D、相互之间存在一种或者多种特定关系的数据元素的集合
■在定义 ADT 时,除数据对象集外,还需说明( )。
A、数据元素　　B、算法　　C、数据项　　D、基本操作
■以下结构（　）是线性结构。
A、集合　　B、二叉树　　C、连通图　　D、栈
■从逻辑上可以把数据结构分为（ ）。
A、动态结构、静态结构　　B、集合结构、线性结构、树型结构、图结构
C、初等结构、构造型结构　　D、顺序结构、链式结构
■下列关于抽象数据类型的描述，不正确的是（　）。
A、用例驱动　　B、信息隐藏　　C、数据封装　　D、使用与实现分离
■计算机所处理的数据一般具有某种关系，这是指（　）。
A、数据与数据之间的某种关系。　　B、数据元素与数据元素之间存在的某种关系。
C、数据项之间的某种关系。　　D、数据文件内记录与记录之间的某种关系。
■在数据结构中,数据的逻辑结构包括(线性结构)和(非线性结构)。
数据的物理存储结构可以分为(顺序存储)、(链式存储)和散列存储、索引存储等四种。
■下列函数中,哪个函数具有最快的增长速度:
A、N(logN)^2　　B、N^2*logN　　C、N^3　　D、N*log(N^2)
■下面一段代码的时间复杂度是?

```
if ( A > B )                                    else
      for ( i=0; i <N;i++)          →                 for ( i=0; i <N;i++)
            for ( j=N*N; j>i; j-- )                       for ( j=N*2; j>i; j-- )
                  A += B;                                       A += B;
```

A、O(N)　　B、O(N^2)　　C、O(N^3)　　D、O(N^4)
■

```
int func ( int n )
{    int i = 0, sum = 0;
     while ( sum < n )   sum += ++i;//等差数列求和
     return i;
}
```

的时间复杂度是:(　　)。
A、O(logn)　　B、$O(\sqrt{n})$　　C、O(n)　　D、O(nlogn)
■

```
for(i=0; i<n; i++)
     for(j=i; j>0; j/=2)
       printf("%d\n", j);
```

的时间复杂度是: (　　)
A、O(N*i)　　B、O(N)　　C、O(N2)　　D、O(NlogN)
■下列程序段的时间复杂度是　　i=1;　　while (i<=n)　i*=10;
A、O(n)　　B、$O(\sqrt{n})$　　C、O(logn)　　D、O(n^10)
■下列程序段的时间复杂度是（　　）。

```
int fact(int n) {
    if   (n<=1)     return 1;
    else            return(n*fact(n-1));
}
```

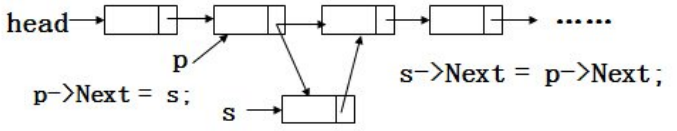
A、O(logn)　　B、O(n^2)　　C、O(1)　　D、O(n)
■某算法的时间复杂度为 O(nlogn)，表明该算法的（　　）。
A、执行时间等于 O(nlogn)　　B、执行时间与 O(nlogn)成正比
C、问题规模是 O(nlogn)　　D、问题规模与 O(nlogn)成正比
■计算机算法必须具备输入、输出和（　　）等五个特性。
A、可行性、可移植性和可扩充性　　B、可行性、确定性和有穷性
C、确定性、有穷性和稳定性　　D、易读性、稳定性和安全性

■下列关于算法的叙述不正确的是（　　）。
A、算法的优劣与算法描述语言无关，与所用计算机也无关。
B、算法的效率只与问题规模有关，而与数据的存储结构无关。
C、算法的有穷性是指算法必须能在有限时间和有限步骤内执行完。
D、健壮的算法不会因非法输入数据而出现莫名其妙的状态。
■某算法的时间复杂度为 O(n^2)。若该算法在规模为 n 的数据集上，运行时间为 10 秒；如果数据规模扩大为 2n，该算法大约需要运行（　　）。
A、100 秒　　B、10 秒　　C、6-7 分钟　　D、40 秒
■对于线性表,在顺序存储结构和链式存储结构中查找第 k 个元素,其时间复杂性分别是多少?
A、都是 O(1)　　B、都是 O(k)　　C、O(1)和 O(k)　　D、O(k)和 O(1)
■在顺序结构表示的线性表中,删除第 i 个元素(数组下标为 i-1),需要把后面的所有元素都往前挪一位,相应的语句是:
for (_____)PtrL->Data[j-1]=PtrL->Data[j];
A、j = i; j< = PtrL->Last; j++　　B、j =PtrL->Last; j>= i;　j--
C、j = i-1; j< = PtrL->Last; j++　　D、j =PtrL->Last; j>= i-1;　j—
■可以用带表头附加结点的链表表示线性表,也可以用不带头结点的链表表示线性表,前者最主要的好处是()。
A、可以加快对表的遍历　　B、使空表和非空表的处理统一，方便运算的实现
C、节省存储空间　　D、可以提高存取表元素的速度
■下列函数试图求链式存储的线性表的表长,是否正确? 不正确

```
int   Length ( List   *PtrL )
{     List   *p = PtrL;        int   j = 0;
      while ( p ) { p++; j++;}
      return   j;}
```

■在矩阵的多重链表表示中,第 i 行的 head 和第 i 列的 head 实际上是同一个结点。正确
■借助堆栈将中缀表达式 A-(B-C/D)*E 转换为后缀表达式,则该堆栈的大小至少为:
A、2　　B、3　　C、4　　D、5
■设 1、2、…、n–1、n 共 n 个数按顺序入栈,若第一个出栈的元素是 n,则第三个出栈的元素是:
A、3　　B、n-2　　C、n-3　　D、任何元素均可能
■若用单向链表实现一个堆栈,当前链表状态为:1->2->3。当对该堆栈执行 pop()、push(4)操作后,链表状态变成怎样?
(1)4->2->3　　(2)　1->2->4
A、只能是(1)　　B、只能是(2)　　C、(1)和(2)都有可能　　D、(1)和(2)都不可能
■如果一堆栈的输入序列是 aAbBc,输出为 abcBA,那么该堆栈所进行的操作序列是什么? 设 P 代表入栈,O 代表出栈。
A、PPPOOPOPOO　　B、POOPPPOPOO　　C、POPPOPPOOO　　D、PPOPPOOOPO
■输入序列为 ABC,变为 CBA 时经过的栈操作为( )。
A、push,pop,push,pop,push,pop　　B、push,push,push,pop,pop,pop
C、push,push,pop,pop,push,pop　　D、push,pop,push,push,pop,pop
■（　）不是栈的基本操作。A、判断栈是否为空　B、将栈置为空栈　　C、删除栈底元素　　D、删除栈顶元素
■3 个不同元素依次进栈，能得到（5）种不同的出栈序列。
■栈的进栈和出栈操作的算法时间复杂度为 O(n). 错误,O(1)
■在链式队列中，队头设在（链头）
■循环队列是队列的一种（顺序）存储结构。
■循环队列通常使用指针来实现队列的头尾相接。错误
■队列是一种插入和删除操作分别在表的两端进行的线性表，是一种先进后出的结构。错误
■设循环队列 Q 的存储容量为 maxSize=25,队头指针 fr10,队尾指针 rear=5，则队列中实际元素个数为（20）。
■假设利用一个数组 s[]顺序存储一个栈，用 top 作为栈顶指针，top=-1 表示栈空，当栈未满时，元素 x 进栈的操作
A、s[--top]=x;　　B、s[++top]=x;　　C、s[top--]=x;　　D、s[top++]=x;
■设用不带头结点的单链表作为链式栈，结点结构为（data, link），top 指向栈顶；若在链式栈插入一个由指针 s 所指向的结点，执行操作（ ）。
A、top->link=s;　　B、s->link=top->link; top->link=s;
C、s->link=top; top=s;　　D、s->link=top; top=top->link;
■在一个链表表示的队列中, f 和 r 分别指向队列的头和尾。下列哪个操作能正确地将 s 结点插入到队列中:
A、f->next=s; f=s;　　B、r->next=s; r=s;　　C、s->next=r; r=s;　　D、s->next=f; f=s;
■现采用大小为 10 的数组实现一个循环队列。设在某一时刻,队列为空且此时 front 和 rear 值均为 5。经过若干操作后,front 为 8,rear 为 2,问:此时队列中有多少个元素?　4

■若用不带头结点的单链表存储队列,其队头指针指向队头结点,其队尾指针指向队尾结点,则在进行删除操作时
A、仅修改队头指针　　B、仅修该队尾指针
C、队头、队尾指针都要修改　　D、队头、队尾指针都可能要修改
■如果循环队列用大小为 m 的数组表示,且用队头指针 front 和队列元素个数 size 代替一般循环队列中的 front 和 rear 指针来表示队列的范围,那么这样的循环队列可以容纳的元素个数最多为:m
■若用大小为 6 的数组来实现循环队列，且当前 front 和 rear 的值分别为 0 和 4。当从队列中删除两个元素，再加入两个元素后，front 和 rear 的值分别为：　2 和 0
■用顺序存储结构直接表示多项式 x+3x2000,数组大小需要多少?
A、2　　B、2000　　C、2001　　D、2002
■如果使用以下结构表示线性表,PtrL 为线性表结构的指针,那么该线性表"长度"怎么表示?
typedef struct LNode *List;
struct LNode{　　ElementType Data[MAXSIZE];　int Last;};　List PtrL;
A、Ptrl.Last　　B、Ptrl.Last+1　　C、Ptrl->Last　　D、Ptrl->Last+1
■如果把后移数组元素的循环
　　for (j = L->Last; j >= i-1; j--)　　改为 for (j = i-1; j <= L->Last; j++)
　　L->Data[j+1]=L->Data[j];　　L->Data[j+1]=L->Data[j];　　那会是什么后果?
A、效果一样,没区别　　B、分量 Data[i-1]到 Data[L->Last+1]都是同一个值,即移之前 Data[i-1]的值
C、分量 Data[i-1]到 Data[L->Last+1]都是同一个值,即移之前 Data[L->Last]的值　　D、说不清楚,要看具体数据

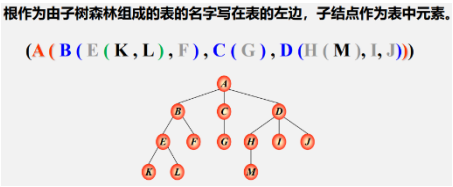


下题图
↓↓↓↓↓↓↓↓↓
↓↓↓↓↓↓↓↓↓

■如果语句执行顺序为:(1) p->Next=s;　(2) s->Next=p->Next;那么后果是什么?
A、也能正确插入 s 结点　B、s->Next 指向 s,从而不能正确完成插入　C、p->Next 指向 p,从而不能正确完成插入
■以下数据结构中，（　）是线性结构。
A、串　　B、家谱　　C、城市交通网　　D、文件管理
■以下（　）是一个线性表。
A、由 n 个实数组成的集合　　B、由 10 个字符组成的序列　　C、所有整数组成的序列　　D、邻接表
■在一个长度为 n 的顺序表中删除第 i 个元素时，需向前移动（ n-i ）个元素。
■线性表的顺序存储结构是一种（　）。
A、散列存取的存储结构　　B、链式存取的存储结构　C、索引存取的存储结构　D、随机存取的存储结构
■链表不具有的特点是（ ）。
A、插入、删除不需要移动元素　　B、可随机访问任意一个元素
C、不必事前估计存储空间　　D、所需空间与线性长度成正比
■设树 T 的度为 4，其中度为 1,2,3,4 的结点个数分别为 4,2,1,1,则 T 中的叶子数为（8 ）。
■假定一棵三叉树的结点数为 50，则它的最小高度为（5）
■假定一棵高度为 4 的完全二叉树中至少有（8）结点，至多有（15）结点。
■在顺序查找中,如果把下列程序中的循环条件"i>0"去掉，会发生什么后果?

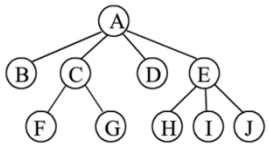
```
int SequentialSearch(List Tbl,ElementType K)
{   /*在表 Element[1]~Element[n]中查找关键字为 K 的数据元素*/
    int i;
    for(i = Tbl->Last; i>0 && Tbl->Data[i] != K;   i--);
    return i;    /*查找成功返回所在单元下标；不成功返回 0*/}
```

A、没有影响，结果一样　　B、要查找的元素存在时找不到
C、要查找的元素不存在时发生数组越界（i 指向小于 0 的位置）　　D、要查找的元素不存在时函数返回 0
■树的路径长度是从树根到每个结点的路径长度的（）
A、最小值　　B、最大值　　C、平均值　　D、总和



■假定一棵树的广义表表示为 (A(B(E),C(F(H,I,J),G), D))，则该树的度为（ 3 ），树的深度为（4），终端结点个数为（6），单分支结点的个数为（1），双分支结点个数为（1），三分支结点个数为（2），C 结点的双亲结点为（A），其孩子结点为（F）和（G）。
■在二分查找的程序实现中，如果 left 和 right 的更新不是取 mid+1 和 mid-1 而是都取 mid 程序也是正确的。错误

■写出下图所示的树的叶结点、非终端结点、每个结点的度及树深度。



叶节点：BDFGHIJ
非终端节点：ACE
每个结点的度：A:4 B:0 C:2 D:0 E:3 F:0 G:0 H:0 I:0 J:0
树的深度：3

■已知一棵树的边集合{<i,m>,<i,n>,<e,i>,<b,e>,<b,d>, <a,b>, <g,j>,<g,k>,<c,g>,<c,f>,<h,l>,<c,h><a,c> }

（1）根结点；　　a
（2）叶子结点；　　m,n,d,f,j,k,l
（3）g 结点的双亲结点；　　c
（4）g 结点的祖先结点；　　a,c
（5）g 结点的孩子结点；　　j,k
（6）e 结点的兄弟结点；　　d
（7）b 结点的层号；　　2
（8）树的深度；　　5
（9）以 c 结点为根的子树深度是多少？3

■如果一个完全二叉树最底下一层为第六层(根为第一层)且该层共有 8 个叶结点,那么该完全二叉树共有多少个结点?
A、31　　B、39　　C、63　　D、71

■若有一棵二叉树的总结点数为 98,只有一个儿子的结点数为 48,则该树的叶结点数是多少? 这样的树不存在

■有四个结点 A、B、C、D 的二叉树,其前序遍历序列为 ABCD,则下面哪个序列是不可能的中序遍历序列? DABC

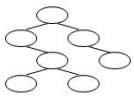
■对于二叉树,如果其中序遍历结果与前序遍历结果一样,那么可以断定该二叉树　　所有结点都没有左儿子

■设深度为 d(只有一个根结点时,d 为 1)的二叉树只有度为 0 和 2 的结点,则此类二叉树的结点数至少为 2d-1 正确

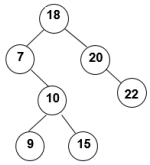
■已知一二叉树的后序和中序遍历的结果分别是 FDEBGCA 和 FDBEACG,那么前序遍历结果是 ABDFECG

■已知有棵 5 个结点的二叉树，其前序遍历序列是 A？？？？，中序遍历序列是 A？？？？，则可以断定
A、该树根结点是 A，且没有左子树　　B、该树根结点是 A，且没有右子树
C、该树最左边的结点是 A　　D、该树不存在

■对于二叉搜索树的最大元素结点,下面哪个说法是正确的?
A、一定是叶结点　　B、一定没有左儿子　　C、一定没有右儿子　　D、是后序遍历的最后一个结点



■已知一棵由 1、2、3、4、5、6、7 共 7 个结点组成的二叉搜索树(查找树),其结构如图所示,问:根结点是什么?
A、1　　B、4　　C、5　　D、不能确定



■在图中的搜索树中删除结点 7,那么删除后该搜索树的后序遍历结果是:
A、9,15,10,22,20,18　　B、15,10,9,22,20,18　　C、9,10,15,18,20,22　　D、22,20,18,15,10,9

■若一搜索树(查找树)是一个有 n 个结点的完全二叉树,则该树的最大值一定在叶结点上。 错误

■若一搜索树(查找树)是一个有 n 个结点的完全二叉树,则该树的最小值一定在叶结点上。 正确

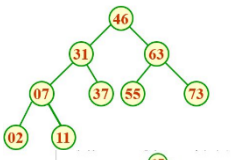
■假设一棵二叉树的先序遍历和后序遍历为 1,2,3,4 和 4,3,2,1，二叉树中序遍历不会是（3,2,4,1）。

■现有一棵无重复结点的平衡二叉树，对其中序遍历可以得到一个升序序列，下列描述正确的是(　)。
A、根结点的度一定为 2　　B、树中最小元素一定是叶结点
C、最后插入的元素一定是叶结点　　D、树中最大元素一定无右子树

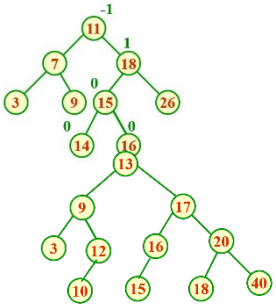
■具有 5 层结点的平衡二叉树至少有(12)结点。　　若平衡二叉树中插入一个结点后不平衡，设最低的不平衡点为 A，已知 A 的左孩子的平衡因子为 0，右孩子平衡因子为 1，则需做(RL)调整。

■一棵二叉树的先序遍历序列和其后序遍历序列正好相反，该二叉树一定是（）。
A、空树或只有根结点　　B、完全二叉树　　C、二叉排序树　　D、高度等于其结点数

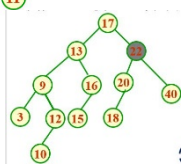
■依次输入关键字序列{55,31,11,37,46,73,63,02,07},构造平衡二叉树。



■输入关键字序列为 {16,3,7,11,9,26, 18, 14, 15}，构造平衡二叉树。



■给出如下平衡树，在该平衡二叉树中删除结点 22，请展示删除结点后的结果平衡二叉树。



■二叉树采用链式存储结构，设计算法实现左右子树的互换。

```
void Exchange(BitTree *bt)
{
    if(bt->lchild==NULL&&bt->rchild==NULL)
        ;
    else    //三种情况，1.都不为空，2.左为空，3.右为空；
    {
        //交换左右子树；
        BitTree *temp=bt->lchild;
        bt->lchild=bt->rchild;
        bt->rchild=temp;
    }

    //如果交换后的这个结点左子树不为空，则继续向下寻找可以交换的结点；
    if(bt->lchild)
        Exchange(bt->lchild);
    if(bt->rchild)
        Exchange(bt->rchild);
}
```

→

答案见右图→答案见右图→答案见右图→ 答案见右图 →

■下列序列中哪个是最小堆?

A、2, 55, 52, 72, 28, 98, 71　　B、2, 28, 71, 72, 55, 98, 52

C、2, 28, 52, 72, 55, 98, 71　　D、28, 2, 71, 72, 55, 98, 52

■在最大堆 {97,76,65,50,49,13,27}中插入 83 后,该最大堆为:

A、{97,76,65,83,49,13,27,50}　　B、{97,83,65,76,49,13,27,50}

C、{97,83,65,76,50,13,27,49}　　D、{97,83,65,76,49,50,13,27}

■对由同样的 n 个整数构成的二叉搜索树(查找树)和最小堆,下面哪个说法是不正确的:

A、二叉搜索树(查找树)高度大于等于最小堆高度

B、对该二叉搜索树(查找树)进行中序遍历可得到从小到大的序列

C、从最小堆根节点到其任何叶结点的路径上的结点值构成从小到大的序列

D、对该最小堆进行按层序(level order)遍历可得到从小到大的序列

■在将数据序列（ 6, 1, 5, 9, 8, 4, 7 ）建成大根堆时，正确的序列变化过程是:

A、6,1,7,9,8,4,5 → 6,9,7,1,8,4,5 → 9,6,7,1,8,4,5 → 9,8,7,1,6,4,5

B、6,9,5,1,8,4,7 → 9,6,5,1,8,4,7 → 9,6,7,1,8,4,5 → 9,8,7,1,6,4,5

C、6,9,5,1,8,4,7 → 6,9,7,1,8,4,5 → 9,6,7,1,8,4,5 → 9,8,7,1,6,4,5

D、6,1,7,9,8,4,5 → 7,1,6,9,8,4,5 → 7,9,6,1,8,4,5 → 9,7,6,1,8,4,5

■如果哈夫曼树有 67 个结点,则可知叶结点总数为:34

一段文本中包含对象{a,b,c,d,e},其出现次数相应为{3,2,4,2,1},则经过哈夫曼编码后,该文本所占总位数为:27

■为五个使用频率不同的字符设计哈夫曼编码,下列方案中哪个不可能是哈夫曼编码?

A、00,100,101,110,111　　B、000,001,01,10,11　　C、0000,0001,001,01,1　　D、000,001,010,011,1

■n 个权值不相同的字符构成哈夫曼树，关于该树的描述中错误的是（）。

A、该树一定是一棵完全二叉树　　B、树中一定没有度为 1 的结点

C、树中两个权值最小的结点一定是兄弟结点　　D、哈夫曼树的结点个数不能是偶数。

■设有一组记录的关键字为 {19,14,23,1,68,20,84,27,55,11,10,79},用分离链接法构造散列表,散列函数为 H(key)= key mod 13。问:散列地址为 1 的链中有几个记录?　4

■设一个散列表的大小是 11，散列函数是 H(key)=key mod 11. 若采用平方探测(di=$\pm i^2$)冲突解决方法,将 4 个元素{14,38,61,86}顺序插入散列表中。如果再插入元素 49,则该元素将被放在什么位置?　4

■假设一散列表的大小是 11,散列函数是 H(key)=key mod 11,用线性探测法解决冲突。先将 4 个元素{14,38,61,86}按顺序插入初始为空的散列表中。如果再插入元素 49,则该元素被插入到表中哪个位置(下标)?　7

■一个大小为 11 的散列表,散列函数为 H(key)=key mod 11,采用线性探测冲突解决策略。如果现有散列表中仅有的 5 个元素均位于下标为奇数的位置,问:该散列表的平均不成功查找次数是多少?　16/11

■在一个大小为 K 的空散列表中,按照线性探测冲突解决策略连续插入散列值相同的 N 个元素(N<K)。问：此时，该散列表的平均成功查找次数是多少?　(N+1)/2

■对于同一个元素集,如果以不同的顺序向散列表中插入各个元素,这些元素在散列表中的位置是一样的。　错误

■对于线性探测,如果当前装填因子值为 0.654321，此时不成功情况下的期望探测次数小于成功情况下的期望探测次数。错误　■当采用线性探测冲突解决策略时,非空且有空闲空间的散列表中无论有多少元素,不成功情况下的期望查找次数总是大于成功情况下的期望查找次数。正确

■有 N 个顶点的无向完全图有多少条边?　N(N-1)/2

■具有 N(>0)个顶点的无向图至多有多少个连通分量　N

■具有 N(>0)个顶点的无向图至少有多少个连通分量　1

■在一个有向图中,所有顶点的入度和等于所有结点的出度之和的(1)倍。

■若 n 个顶点的图是一个环,则它有(n)棵生成树。

■一个有 n 个顶点和 n 条边的无向图一定是(有环的)。

■给定有向图的邻接矩阵如下：顶点 2(编号从 0 开始)的出度和入度分别是: 0, 2

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

■如果从无向图的任一顶点出发进行一次深度优先搜索可访问所有顶点,则该图一定是

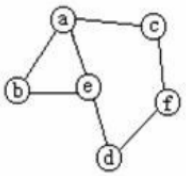
A、有回路的图　　B、完全图　　C、连通图　　D、一棵树

■下列命题正确的是 A、一个图的邻接矩阵表示是唯一的,邻接表表示也唯一

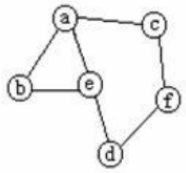
B、一个图的邻接矩阵表示是唯一的,邻接表表示不唯一　　C、一个图的邻接矩阵表示不唯一,邻接表表示是唯一的　　D、一个图的邻接矩阵表示不唯一,邻接表表示也不唯一

■用邻接表表示有 N 个顶点、E 条边的图,则遍历图中所有边的时间复杂度为: O(N+E)

■若图的邻接矩阵中主对角线上的元素均为 0,其余元素全为 1,则可以判断该图一定(完全图)
■用一维数组 G[ ]存储有 4 个顶点的无向图如下:G[ ] = {0,1,0,1,1,0,0,0,1,0},则顶点 2 和顶点 0 之间是有边的。√
■有向图的邻接矩阵一定是不对称的×
■已知一个图如下图所示,从顶点 a 出发按广度优先搜索法进行遍历,则可能得到的一种顶点序列为:



A、a,b,c,e,d,f　　B、a,b,c,e,f,d
C、a,e,b,c,f,d　　D、a,c,f,d,e,b
■已知一个图如下图所示,从顶点 a 出发按深度优先搜索法进行遍历,则可能得到的一种顶点序列为:



A、a,e,b,c,f,d　　B、a,b,e,c,d,f
C、a,c,f,e,b,d　　D、a,e,d,f,c,b
■图的广度优先遍历算法类似于树的(层次遍历)。
■任意一个无向连通图的最小生成树(有一棵或多棵)。
■下列关于最小生成树的论述中,正确的是()。
A、最小生成树的代价唯一　　B、所有权值最小的边一定会出现在所有的最小生成树中
C、使用 Prim 算法从不同顶点开始得到的最小生成树一定相同
D、使用 Prim 算法和 Kruskal 算法得到的最小生成树总不相同
■设无向图 G 和 G',若 G'是 G 的生成树,下列说法错误的是()。
A、G'是 G 的子图　　B、G'是 G 的连通分量
C、G'是 G 的极小连通子图且 V=V'　　D、G'是 G 的一个无环子图
■用 Prim 算法和 Kruskal 算法构造图的最小生成树,所得到的最小生成树(可能相同,可能不同)。
■Dijkstra 算法中的 dist 应该如何初始化?如果 s 到 w 有直接的边,则 dist[w]=的权重;否则 dist[w]定义为
A、正无穷　　B、负无穷　　C、-1　　D、这三种都可以
■Floyd 算法中的 D-1 矩阵应该初始化为什么?
A、带权的邻接矩阵,对角元是 0　　B、带权的邻接矩阵,对角元是无穷大
C、全是 0 的矩阵　　D、全是无穷大的矩阵
■Floyd 算法应该选择什么方法表示图?
A、邻接表　　B、邻接矩阵　　C、都一样　　D、都不选
■在 TopSort 函数中,如果外循环还没结束,就已经找不到"未输出的入度为 0 的顶点",则说明
A、图中必定存在回路　　B、图不连通　　C、图中可能有回路　　D、程序写错啦
■拓扑序一定是唯一的。(×)
■对 N 个记录进行堆排序,最坏的情况下时间复杂度是 O(NlogN)
■在堆排序中,元素下标从 0 开始。则对于下标为 i 的元素,其左、右孩子的下标分别为: 2i+1, 2i+2
■有一组记录(46,77,55,38,41,85),用堆排序建立的初始堆为 85,77,55,38,41,46
■堆排序是稳定的。×
■用直接插入排序方法对下面四个序列进行排序(从小到大),元素比较次数最少的是:
A、100,35,40,90,80,55,20,75　　B、35,40,20,55,70,100,90,80
C、20,35,55,40,75,80,90,100　　D、90,75,80,55,20,35,100,40
■希尔排序是稳定的算法。×
■对于 7 个数进行冒泡排序,最坏情况下需要进行的比较次数为 21
■对 n 个不同的元素利用冒泡法从小到大排序,_____的情况下元素交换的次数最多。
A、从大到小排列好　　B、从小到大排列好
C、元素无序　　D、元素基本有序
■什么是快速排序算法的最好情况?
A、初始就顺序　　B、初始就逆序
C、每次主元的位置都在数组的一端　　D、每次正好中分
■void Quick_Sort(ElementType A[], int N)
{/* 这里写什么?*/}
A、Quicksort(A, 0, N);　　B、Quicksort(A, 0, N-1);
C、Quicksort(A, 1, N);　　D、Quicksort(A, 1, N-1);
■快速排序是稳定的算法。×
■下列排序算法中, 插入排序可能出现:在最后一趟开始之前,所有的元素都不在其最终的位置上
■当待排序列已经基本有序时选择排序算法效率最差
■请选择下面四种排序算法中稳定的排序算法是: 归并排序
■数据序列(3,2,4,9,8,11,6,20)只能是下列哪种排序算法的两趟排序结果
A、冒泡排序　　B、插入排序　　C、选择排序　　D、快速排序