更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

即 k 为模式串的下次比较位置。k 值可能有多个，为了不使向右移动丢失可能的匹配，右移距离应该取最小，由于 j-k 表示右移的距离，所以取 max{k}。k 的最大值为 j-1。

3）除上面两种情况外，发生失配时，主串指针 i 不回溯，在最坏情况下，模式串从第 1 个字符开始与主串的第 i 个字符比较。

02. 【解答】

1）P='aabaac'，按照 next 数组生成算法，对于 P 有：

① 设 next[1]=0，next[2]=1。

编号	1	2	3	4	5	6
S	a	a	b	a	a	c
next	0	1				

② j=3 时 k=next[j-1]=next[2]=1，观察 S[j-1](S[2])与 S[k](S[1])是否相等，S[2]=a，S[1]=a，S[2]=S[1]，所以 next[j]=k+1=2。

```
      ↓j-1=2
a   a   b   a   a   c
    a   a   b   a   a   c
    ↑k=1
```

③ j=4 时 k=next[j-1]=next[3]=2，观察 S[j-1](S[3])与 S[k](S[2])是否相等，S[3]=b，S[2]=a，S[3]!=S[2]。

```
          ↓j-1=3
a   a   b   a   a   c
    a   a   b   a   a   c
        ↑k=2
```

此时 k=next[k]=1，观察 S[3]与 S[k](S[1])是否相等，S[3]=b，S[1]=a，S[3]!=S[1]。k=next[k]=0，因为 k=0，所以 next[j]=1。

```
          ↓j-1=3
a   a   b   a   a   c
        a   a   b   a   a   c
    ↑k=0
```

④ j=5 时 k=next[j-1]=next[4]=1，观察 S[j-1](S[4])与 S[k](S[1])是否相等，S[4]=a，S[1]=a，S[4]=S[1]，所以 next[j]=k+1=2。

```
              ↓j-1=4
a   a   b   a   a   c
            a   a   b   a   a   c
            ↑k=1
```

⑤ j=6 时 k=next[j-1]=next[5]=2，观察 S[j-1](S[5])与 S[k](S[2])是否相等，S[5]=a，S[2]=a，S[5]=S[2]，所以 next[j]=k+1=3。

```
                ↓j-1=5
a   a   b   a   a   c
            a   a   b   a   a   c
                ↑k=2
```

最后的结果为

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

编号	1	2	3	4	5	6
S	a	a	b	a	a	c
next	0	1	2	1	2	3

也可以通过求部分匹配值表的方法来求 next 数组。

2）利用 KMP 算法的匹配过程如下。

第一趟：从主串和模式串的第一个字符开始比较，失配时 i=6，j=6。

主串	a	a	b	a	a	b	a	a	b	a	a	c
	a	a	b	a	a	c						

第二趟：next[6]=3，主串当前位置和模式串的第 3 个字符继续比较，失配时 i=9，j=6。

主串	a	a	b	a	a	b	a	a	b	a	a	c
				a	a	b	a	a	c			

第三趟：next[6]=3，主串当前位置和模式串的第 3 个字符继续比较，匹配成功。

主串	a	a	b	a	a	b	a	a	b	a	a	c
							a	a	b	a	a	c

归纳总结

在学习 KMP 算法时，应从分析暴力法的弊端入手，思考通过何种方式来优化。实际上已匹配相等的序列就是模式串的某个前缀，因此每次回溯就相当于是模式串与模式串某个前缀在比较，这种频繁的重复比较是效率低的原因所在，可以从分析模式串本身的结构入手，然后就能得知当匹配到某个字符不等时，应向后滑动到什么位置，即已匹配相等的前缀和模式串若有首尾重合，对齐它们，对齐部分显然无需再比较，下一步直接从主串的当前位置继续比较。

思维拓展

编程实现：模式串在主串中有多少个完全匹配的子串？（注：统考不会考 KMP 算法题）

关注公众号【乘龙考研】
一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

第 5 章 树与二叉树

【考纲内容】

（一）树的基本概念

（二）二叉树

二叉树的定义及其主要特征；二叉树的顺序存储结构和链式存储结构；二叉树的遍历；线索二叉树的基本概念和构造

（三）树、森林

树的存储结构；森林与二叉树的转换；树和森林的遍历

（四）树与二叉树的应用

哈夫曼（Huffman）树和哈夫曼编码；并查集及其应用

兑换后

看视频讲解

【知识框架】

- 树形结构
  - 二叉树
    - 概念：定义、存储结构
    - 操作
      - 三种遍历
      - 线索二叉树
    - 应用
      - 并查集
      - 哈夫曼树
  - 树和森林
    - 概念：定义、存储结构
    - 操作
      - 与二叉树的转换
      - 遍历
    - 应用：并查集

【复习提示】

本章内容多以选择题的形式考查，但也会出涉及树遍历相关的算法题。树和二叉树的性质、遍历操作、转换、存储结构和操作特性等，满二叉树、完全二叉树、线索二叉树、哈夫曼树的定义和性质，都是选择题必然会涉及的内容。

5.1 树的基本概念

5.1.1 树的定义

树是 n（$n\geqslant 0$）个结点的有限集。当 $n=0$ 时，称为空树。在任意一棵非空树中应满足：

1）有且仅有一个特定的称为根的结点。

2）当 $n>1$ 时，其余结点可分为 m（$m>0$）个互不相交的有限集 $T_1, T_2, \cdots, T_m$，其中每个集

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

合本身又是一棵树，并且称为根的子树。

显然，树的定义是递归的，即在树的定义中又用到了其自身，树是一种递归的数据结构。树作为一种逻辑结构，同时也是一种分层结构，具有以下两个特点：

1）树的根结点没有前驱，除根结点外的所有结点有且只有一个前驱。

2）树中所有结点都可以有零个或多个后继。

树适合于表示具有层次结构的数据。树中的某个结点（除根结点外）最多只和上一层的一个结点（即其父结点）有直接关系，根结点没有直接上层结点，因此在 n 个结点的树中有 $n-1$ 条边。而树中每个结点与其下一层的零个或多个结点（即其子女结点）都有直接关系。

5.1.2　基本术语

下面结合图 5.1 中的树来说明一些基本术语和概念。

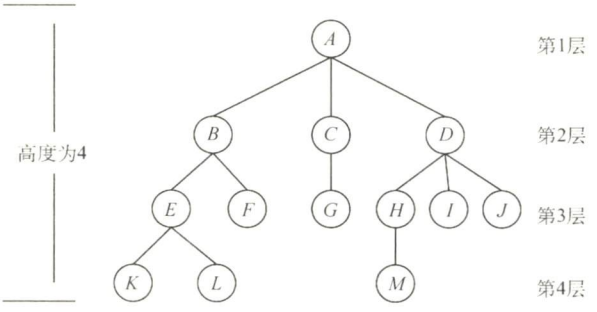


图 5.1　树的树形表示

1）考虑结点 K。根 A 到结点 K 的唯一路径上的任意结点，称为结点 K 的祖先。如结点 B 是结点 K 的祖先，而结点 K 是结点 B 的子孙。路径上最接近结点 K 的结点 E 称为 K 的双亲，而 K 为结点 E 的孩子。根 A 是树中唯一没有双亲的结点。有相同双亲的结点称为兄弟，如结点 K 和结点 L 有相同的双亲 E，即 K 和 L 为兄弟。

2）树中一个结点的孩子个数称为该结点的度，树中结点的最大度数称为树的度。如结点 B 的度为 2，结点 D 的度为 3，树的度为 3。

3）度大于 0 的结点称为分支结点（又称非终端结点）；度为 0（没有子女结点）的结点称为叶结点（又称终端结点）。在分支结点中，每个结点的分支数就是该结点的度。

4）结点的深度、高度和层次。

结点的层次从树根开始定义，根结点为第 1 层，它的子结点为第 2 层，以此类推。双亲在同一层的结点互为堂兄弟，图 5.1 中结点 G 与 E, F, H, I, J 互为堂兄弟。

结点的深度是从根结点开始自顶向下逐层累加的。

结点的高度是从叶结点开始自底向上逐层累加的。

树的高度（或深度）是树中结点的最大层数。图 5.1 中树的高度为 4。

5）有序树和无序树。树中结点的各子树从左到右是有次序的，不能互换，称该树为有序树，否则称为无序树。假设图 5.1 为有序树，若将子结点位置互换，则变成一棵不同的树。

6）路径和路径长度。树中两个结点之间的路径是由这两个结点之间所经过的结点序列构成的，而路径长度是路径上所经过的边的个数。

注意：由于树中的分支是有向的，即从双亲指向孩子，所以树中的路径是从上向下的，同一双亲的两个孩子之间不存在路径。

关注公众号【乘龙考研】一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

7）森林。森林是 m（$m \geqslant 0$）棵互不相交的树的集合。森林的概念与树的概念十分相近，因为只要把树的根结点删去就成了森林。反之，只要给 m 棵独立的树加上一个结点，并把这 m 棵树作为该结点的子树，则森林就变成了树。

注意：上述概念无须刻意记忆，根据实例理解即可。考研时不大可能直接考查概念，而都是结合具体的题目考查。做题时，遇到不熟悉的概念可以翻书，练习得多自然就记住了。

5.1.3 树的性质

树具有如下最基本的性质：

1）树中的结点数等于所有结点的度数之和加1。

2）度为 m 的树中第 i 层上至多有 m^{i-1} 个结点（$i \geqslant 1$）。

3）高度为 h 的 m 叉树至多有 $(m^h-1)/(m-1)$ 个结点[①]。

4）具有 n 个结点的 m 叉树的最小高度为 $\lceil \log_m(n(m-1)+1) \rceil$。

5.1.4 本节试题精选

一、单项选择题

01. 树最适合用来表示（ ）的数据。

A. 有序　　B. 无序

C. 任意元素之间具有多种联系　　D. 元素之间具有分支层次关系

02. 一棵有 n 个结点的树的所有结点的度数之和为（ ）。

A. $n-1$　　B. n　　C. $n+1$　　D. $2n$

03. 树的路径长度是从树根到每个结点的路径长度的（ ）。

A. 总和　　B. 最小值　　C. 最大值　　D. 平均值

04. 对于一棵具有 n 个结点、度为4的树来说，（ ）。

A. 树的高度至多是 $n-3$　　B. 树的高度至多是 $n-4$

C. 第 i 层上至多有 $4(i-1)$ 个结点　　D. 至少在某一层上正好有4个结点

05. 度为4、高度为 h 的树，（ ）。

A. 至少有 $h+3$ 个结点　　B. 至多有 $4h-1$ 个结点

C. 至多有 $4h$ 个结点　　D. 至少有 $h+4$ 个结点

06. 假定一棵度为3的树中，结点数为50，则其最小高度为（ ）。

A. 3　　B. 4　　C. 5　　D. 6

07. 【2010统考真题】在一棵度为4的树 T 中，若有20个度为4的结点，10个度为3的结点，1个度为2的结点，10个度为1的结点，则树 T 的叶结点个数是（ ）。

A. 41　　B. 82　　C. 113　　D. 122

二、综合应用题

01. 含有 n 个结点的三叉树的最小高度是多少？

02. 已知一棵度为4的树中，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。

03. 已知一棵度为 m 的树中，有 n_1 个度为1的结点，有 n_2 个度为2的结点……有 n_m 个度为 m 的结点，问该树有多少个叶结点？

---

① 推导公式 $S=m^{h-1}+m^{h-2}+m^{h-3}+\cdots+m+1=(m^h-1)/(m-1)$。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

关注公众号【乘龙考研】
一手更新 稳定有保障

5.1.5　答案与解析

一、单项选择题

01．D

树是一种分层结构，它特别适合组织那些具有分支层次关系的数据。

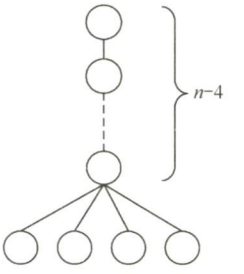
02．A

除根结点外，其他每个结点都是某个结点的孩子，因此树中所有结点的度数加 1 等于结点数，也即所有结点的度数之和等于总结点数减 1。这是一个重要的结论，做题时经常用到。

03．A

树的路径长度是指树根到每个结点的路径长的总和，根到每个结点的路径长度的最大值应是树的高度减 1。注意与哈夫曼树的带权路径长度相区别。

04．A



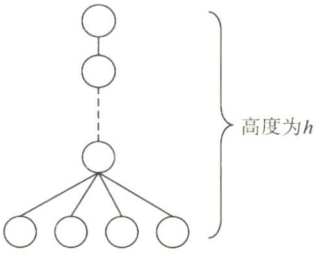
要使得具有 n 个结点、度为 4 的树的高度最大，就要使得每层的结点数尽可能少，类似右图所示的树，除最后一层外，每层的结点数是 1，最终该树的高度为 $n-3$。树的度为 4 只能说明存在某结点正好（也最多）有 4 个孩子结点，选项 D 错误。

05．A

要使得度为 4、高度为 h 的树的总结点数最少，需要满足以下两个条件：

① 至少有一个结点有 4 个分支。

② 每层的结点数目尽可能少。



情况类似右图所示的树，结点个数为 $h+3$。

要使得度为 4、高度为 h 的树的总结点数最多，应使每个非叶结点的度均为4，即为满树，总结点个数最多为 $1+4+4^2+\cdots+4^{h-1}$。

对于上面的两题，应画出草图来求解，就能一目了然。

06．C

要求满足条件的树，那么该树是一棵完全三叉树。在度为 3 的完全三叉树中，第 1 层有 1 个结点，第 2 层有 $3^1=3$ 个结点，第 3 层有 $3^2=9$ 个结点，第 4 层有 $3^3=27$ 个结点，因此结点数之和为 $1+3+9+27=40$，第 5 层的结点数 $=50-40=10$ 个，因此最小高度为 5。

07．B

设树中度为 i（$i=0, 1, 2, 3, 4$）的结点数分别为 n_i，树中结点总数为 n，则 $n=$ 分支数 $+1$，而分支数又等于树中各结点的度之和，即 $n=1+n_1+2n_2+3n_3+4n_4=n_0+n_1+n_2+n_3+n_4$。依题意，$n_1+2n_2+3n_3+4n_4=10+2+30+80=122$，$n_1+n_2+n_3+n_4=10+1+10+20=41$，可得出 $n_0=82$，即树 T 的叶结点的个数是 82。

二、综合应用题

01．【解答】

要求含有 n 个结点的三叉树的最小高度，那么满足条件的一定是一棵完全三叉树，设含有 n 个结点的完全三叉树的高度为 h，第 h 层至少有 1 个结点，至多有 3^{h-1} 个结点。则有

$$1+3^1+3^2+\cdots+3^{h-2}<n\leqslant 1+3^1+3^2+\cdots+3^{h-2}+3^{h-1}$$

即 $(3^{h-1}-1)/2<n\leqslant(3^h-1)/2$，得 $3^{h-1}<2n+1\leqslant 3^h$，也即 $h<\log_3(2n+1)+1$，$h\geqslant\log_3(2n+1)$。

由于 h 只能为正整数，$h=\lceil\log_3(2n+1)\rceil$，故这样的三叉树的最小高度是 $\lceil\log_3(2n+1)\rceil$。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

02.【解答】

设树中度为 i（$i=0,1,2,3,4$）的结点数为 n_i，那么结点总数 $n=n_0+n_1+n_2+n_3+n_4$，即 $n=23+n_4$，根据“树中所有结点的度数加1等于结点数”的结论，有 $n=0+n_1+2n_2+3n_3+4n_4+1$，即有 $n=17+4n_4$。

综合两式得 $n_4=2$，$n=25$。所以该树的结点总数为25，度为4的结点个数为2。

03.【解答】

树中的结点数等于所有结点的度数加1，因此有 $n=\sum_{i=0}^{m} in_i+1=n_1+2n_2+3n_3+\cdots+mn_m+1$。

又有 $n=n_0+n_1+n_2+\cdots+n_m$，所以

$$\begin{aligned} n_0 &= (n_1+2n_2+3n_3+\cdots+mn_m+1)-(n_1+n_2+\cdots+n_m) \\ &= n_2+2n_3+\cdots+(m-1)n_m+1=1+\sum_{i=2}^{m}(i-1)n_i \end{aligned}$$

注意：综合以上几题，常用于求解树结点与度之间关系的有：

① 总结点数 $=n_0+n_1+n_2+\cdots+n_m$。

② 总分支数 $=1n_1+2n_2+\cdots+mn_m$（度为 m 的结点引出 m 条分支）。

③ 总结点数 = 总分支数 + 1。

这类题目经常在选择题中出现，读者对于以上关系应当熟练掌握并灵活应用。

5.2 二叉树的概念

关注公众号【乘龙考研】
一手更新 稳定有保障

5.2.1 二叉树的定义及其主要特性

1. 二叉树的定义

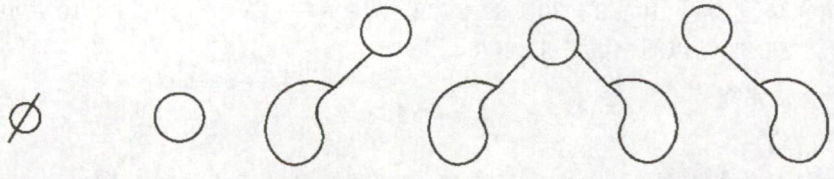
二叉树是一种特殊的树形结构，其特点是每个结点至多只有两棵子树（即二叉树中不存在度大于2的结点），并且二叉树的子树有左右之分，其次序不能任意颠倒。

与树相似，二叉树也以递归的形式定义。二叉树是 n（$n\geqslant 0$）个结点的有限集合：

① 或者为空二叉树，即 $n=0$。

② 或者由一个根结点和两个互不相交的被称为根的左子树和右子树组成。左子树和右子树又分别是一棵二叉树。

二叉树是有序树，若将其左、右子树颠倒，则成为另一棵不同的二叉树。即使树中结点只有一棵子树，也要区分它是左子树还是右子树。二叉树的5种基本形态如图5.2所示。



(a) 空二叉树　(b) 只有根结点　(c) 只有左子树　(d) 左右子树都有　(e) 只有右子树

图5.2　二叉树的5种基本形态

二叉树与度为2的有序树的区别：

① 度为2的树至少有3个结点，而二叉树可以为空。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

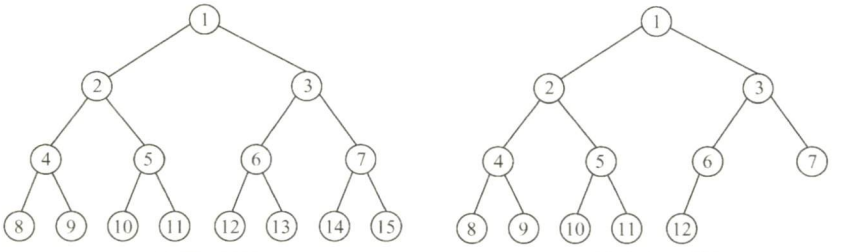
② 度为 2 的有序树的孩子的左右次序是相对于另一孩子而言的，若某个结点只有一个孩子，则这个孩子就无须区分其左右次序，而二叉树无论其孩子数是否为 2，均需确定其左右次序，即二叉树的结点次序不是相对于另一结点而言的，而是确定的。

2．几个特殊的二叉树

1）满二叉树。一棵高度为 h，且含有 2^h-1 个结点的二叉树称为满二叉树，即树中的每层都含有最多的结点，如图 5.3(a)所示。满二叉树的叶结点都集中在二叉树的最下一层，并且除叶结点之外的每个结点度数均为 2。

可以对满二叉树按层序编号：约定编号从根结点（根结点编号为 1）起，自上而下，自左向右。这样，每个结点对应一个编号，对于编号为 i 的结点，若有双亲，则其双亲为$\lfloor i/2 \rfloor$，若有左孩子，则左孩子为 $2i$；若有右孩子，则右孩子为 $2i+1$。

2）完全二叉树。高度为 h、有 n 个结点的二叉树，当且仅当其每个结点都与高度为 h 的满二叉树中编号为 1～n 的结点一一对应时，称为完全二叉树，如图 5.3(b)所示。其特点如下：



(a) 满二叉树　　(a) 完全二叉树

图 5.3　两种特殊形态的二叉树[①]

① 若 $i \leqslant \lfloor n/2 \rfloor$，则结点 i 为分支结点，否则为叶结点。

② 叶结点只可能在层次最大的两层上出现。对于最大层次中的叶结点，都依次排列在该层最左边的位置上。

③ 若有度为 1 的结点，则只可能有一个，且该结点只有左孩子而无右孩子（重要特征）。

④ 按层序编号后，一旦出现某结点（编号为 i）为叶结点或只有左孩子，则编号大于 i 的结点均为叶结点。

⑤ 若 n 为奇数，则每个分支结点都有左孩子和右孩子；若 n 为偶数，则编号最大的分支结点（编号为 $n/2$）只有左孩子，没有右孩子，其余分支结点左、右孩子都有。

3）二叉排序树。左子树上所有结点的关键字均小于根结点的关键字；右子树上的所有结点的关键字均大于根结点的关键字；左子树和右子树又各是一棵二叉排序树。

4）平衡二叉树。树上任意一个结点的左子树和右子树的深度之差不超过 1。

3．二叉树的性质

1）非空二叉树上的叶结点数等于度为 2 的结点数加 1，即 $n_0=n_2+1$。

证明：设度为 0, 1 和 2 的结点个数分别为 n_0, n_1 和 n_2，结点总数 $n=n_0+n_1+n_2$。

再看二叉树中的分支数，除根结点外，其余结点都有一个分支进入，设 B 为分支总数，则 $n=B+1$。由于这些分支是由度为 1 或 2 的结点射出的，所以又有 $B=n_1+2n_2$。

于是得 $n_0+n_1+n_2=n_1+2n_2+1$，则 $n_0=n_2+1$。

① 完全二叉树就是对应相同高度的满二叉树缺失最下层最右边的一些连续叶结点。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

注意：该性质经常在选择题中用到，希望考生牢记并灵活应用。拓展到任意一棵树，若结点数量为 n，则边的数量为 $n-1$。

2）非空二叉树上第 k 层上至多有 2^{k-1} 个结点（$k\geqslant 1$）。

第 1 层至多有 $2^{1-1}=1$ 个结点（根），第 2 层至多有 $2^{2-1}=2$ 个结点，以此类推，可以证明其为一个公比为 2 的等比数列 2^{k-1}。

3）高度为 h 的二叉树至多有 2^h-1 个结点（$h\geqslant 1$）。

该性质利用性质 2 求前 h 项的和，即等比数列求和的结果。

4）对完全二叉树按从上到下、从左到右的顺序依次编号 $1,2,\cdots,n$，则有以下关系：

① 当 $i>1$ 时，结点 i 的双亲的编号为 $\lfloor i/2\rfloor$，即当 i 为偶数时，其双亲的编号为 $i/2$，它是双亲的左孩子；当 i 为奇数时，其双亲的编号为 $(i-1)/2$，它是双亲的右孩子。

② 当 $2i\leqslant n$ 时，结点 i 的左孩子编号为 $2i$，否则无左孩子。

③ 当 $2i+1\leqslant n$ 时，结点 i 的右孩子编号为 $2i+1$，否则无右孩子。

④ 结点 i 所在层次（深度）为 $\lfloor \log_2 i\rfloor+1$。

5）具有 n 个（$n>0$）结点的完全二叉树的高度为 $\lceil \log_2(n+1)\rceil$ 或 $\lfloor \log_2 n\rfloor+1$。

设高度为 h，根据性质 3 和完全二叉树的定义有

$$2^{h-1}-1<n\leqslant 2^h-1 \quad 或 \quad 2^{h-1}\leqslant n<2^h$$

得 $2^{h-1}<n+1\leqslant 2^h$，即 $h-1<\log_2(n+1)\leqslant h$，因为 h 为正整数，所以 $h=\lceil \log_2(n+1)\rceil$；或者得 $h-1\leqslant \log_2 n<h$，所以 $h=\lfloor \log_2 n\rfloor+1$。

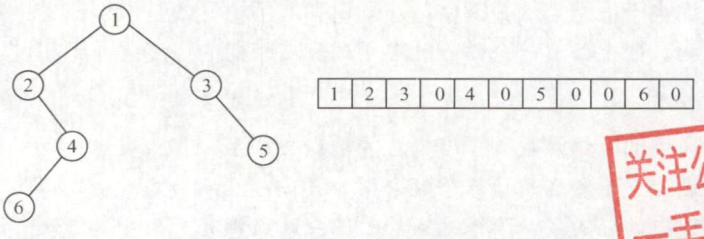
5.2.2 二叉树的存储结构

1. 顺序存储结构

二叉树的顺序存储是指用一组地址连续的存储单元依次自上而下、自左至右存储完全二叉树上的结点元素，即将完全二叉树上编号为 i 的结点元素存储在一维数组下标为 $i-1$ 的分量中。

依据二叉树的性质，完全二叉树和满二叉树采用顺序存储比较合适，树中结点的序号可以唯一地反映结点之间的逻辑关系，这样既能最大可能地节省存储空间，又能利用数组元素的下标值确定结点在二叉树中的位置，以及结点之间的关系。

但对于一般的二叉树，为了让数组下标能反映二叉树中结点之间的逻辑关系，只能添加一些并不存在的空结点，让其每个结点与完全二叉树上的结点相对照，再存储到一维数组的相应分量中。然而，在最坏情况下，一个高度为 h 且只有 h 个结点的单支树却需要占据近 2^h-1 个存储单元。二叉树的顺序存储结构如图 5.4 所示，其中 0 表示并不存在的空结点。



关注公众号【乘龙考研】
一手更新 稳定有保障

图 5.4 二叉树的顺序存储结构

注意：这种存储结构建议从数组下标 1 开始存储树中的结点，若从数组下标 0 开始存储，则不满足性质 4 的描述（比如结点 A 存储在 0 下标位置上时，无法根据性质 4 来计算出其孩子结点在数组中的位置），这是考生在书写程序时容易忽略的。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

2．链式存储结构

由于顺序存储的空间利用率较低，因此二叉树一般都采用链式存储结构，用链表结点来存储二叉树中的每个结点。在二叉树中，结点结构通常包括若干数据域和若干指针域，二叉链表至少包含 3 个域：数据域 data、左指针域 lchild 和右指针域 rchild，如图 5.5 所示。

lchild	data	rchild

图 5.5　二叉树链式存储的结点结构

图 5.6 所示为常用的二叉链表的存储结构。而实际上在不同的应用中，还可以增加某些指针域，如增加指向父结点的指针后，变为三叉链表的存储结构。

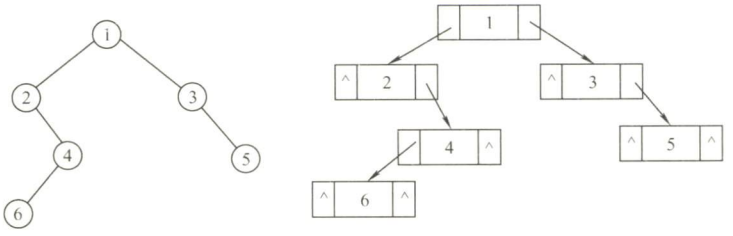


图 5.6　二叉链表的存储结构

关注公众号【乘龙考研】一手更新 稳定有保障

二叉树的链式存储结构描述如下：

```
typedef struct BiTNode{
    ElemType data;                          //数据域
    struct BiTNode *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

使用不同的存储结构时，实现二叉树操作的算法也会不同，因此要根据实际应用场合（二叉树的形态和需要进行的运算）来选择合适的存储结构。

容易验证，在含有 n 个结点的二叉链表中，含有 $n+1$ 个空链域（重要结论，经常出现在选择题中）。在下一节中，我们将利用这些空链域来组成另一种链表结构——线索链表。

5.2.3　本节试题精选

一、单项选择题

01. 下列关于二叉树的说法中，正确的是（　）。

A. 度为 2 的有序树就是二叉树

B. 含有 n 个结点的二叉树的高度为 $\lfloor \log_2 n \rfloor + 1$

C. 在完全二叉树中，若一个结点没有左孩子，则它必是叶结点

D. 含有 n 个结点的完全二叉树的高度为 $\lceil \log_2 n \rceil$

02. 以下说法中，正确的是（　）。

A. 在完全二叉树中，叶结点的双亲的左兄弟（若存在）一定不是叶结点

B. 任何一棵二叉树中，叶结点数为度为 2 的结点数减 1，即 $n_0 = n_2 - 1$

C. 完全二叉树不适合顺序存储结构，只有满二叉树适合顺序存储结构

D. 结点按完全二叉树层序编号的二叉树中，第 i 个结点的左孩子的编号为 $2i$

03. 具有 10 个叶结点的二叉树中有（　）个度为 2 的结点。

A. 8　　B. 9　　C. 10　　D. 11

04. 设高度为 h 的二叉树上只有度为 0 和度为 2 的结点，则此类二叉树中所包含的结点数至少为（　）。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

A. h　　B. $2h-1$　　C. $2h+1$　　D. $h+1$

05. 假设一棵二叉树的结点个数为50，则它的最小高度是（ ）。

A. 4　　B. 5　　C. 6　　D. 7

06. 设二叉树有$2n$个结点，且$m<n$，则不可能存在（ ）的结点。

A. n个度为0　　B. $2m$个度为0

C. $2m$个度为1　　D. $2m$个度为2

07. 一个具有1025个结点的二叉树的高h为（ ）。

A. 11　　B. 10　　C. 11～1025　　D. 10～1024

08. 设二叉树只有度为0和2的结点，其结点个数为15，则该二叉树的最大深度为（ ）。

A. 4　　B. 5　　C. 8　　D. 9

09. 高度为h的完全二叉树最少有（ ）个结点。

A. 2^h　　B. 2^h+1　　C. 2^{h-1}　　D. 2^h-1

10. 已知一棵完全二叉树的第6层（设根为第1层）有8个叶结点，则完全二叉树的结点个数最少是（ ）。

A. 39　　B. 52　　C. 111　　D. 119

11. 若一棵深度为6的完全二叉树的第6层有3个叶结点，则该二叉树共有（ ）个叶结点。

A. 17　　B. 18　　C. 19　　D. 20

12. 一棵完全二叉树上有1001个结点，其中叶结点的个数是（ ）。

A. 250　　B. 500　　C. 254　　D. 501

13. 若一棵二叉树有126个结点，在第7层（根结点在第1层）至多有（ ）个结点。

A. 32　　B. 64　　C. 63　　D. 不存在第7层

14. 一棵有124个叶结点的完全二叉树，最多有（ ）个结点。

A. 247　　B. 248　　C. 249　　D. 250

15. 一棵有n个结点的二叉树采用二叉链存储结点，其中空指针数为（ ）。

A. n　　B. $n+1$　　C. $n-1$　　D. $2n$

16. 在一棵完全二叉树中，其根的序号为1，（ ）可判定序号为p和q的两个结点是否在同一层。

A. $\lfloor \log_2 p \rfloor = \lfloor \log_2 q \rfloor$　　B. $\log_2 p = \log_2 q$

C. $\lfloor \log_2 p \rfloor + 1 = \lfloor \log_2 q \rfloor$　　D. $\lfloor \log_2 p \rfloor = \lfloor \log_2 q \rfloor + 1$

17. 假定一棵三叉树的结点数为50，则它的最小高度为（ ）。

A. 3　　B. 4　　C. 5　　D. 6

18. 对于一棵满二叉树，共有n个结点和m个叶结点，高度为h，则（ ）。

A. $n=h+m$　　B. $n+m=2h$　　C. $m=h-1$　　D. $n=2^h-1$

19. 【2009统考真题】已知一棵完全二叉树的第6层（设根为第1层）有8个叶结点，则该完全二叉树的结点个数最多是（ ）。

A. 39　　B. 52　　C. 111　　D. 119

20. 【2011统考真题】若一棵完全二叉树有768个结点，则该二叉树中叶结点的个数是（ ）。

A. 257　　B. 258　　C. 384　　D. 385

21. 【2018统考真题】设一棵非空完全二叉树T的所有叶结点均位于同一层，且每个非叶结点都有2个子结点。若T有k个叶结点，则T的结点总数是（ ）。

A. $2k-1$　　B. $2k$　　C. k^2　　D. 2^k-1

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

22.【2020 统考真题】对于任意一棵高度为 5 且有 10 个结点的二叉树，若采用顺序存储结构保存，每个结点占 1 个存储单元（仅存放结点的数据信息），则存放该二叉树需要的存储单元数量至少是（　）。

A. 31　　B. 16　　C. 15　　D. 10

二、综合应用题

01. 在一棵完全二叉树中，含有 n_0 个叶结点，当度为 1 的结点数为 1 时，该树的高度是多少？当度为 1 的结点数为 0 时，该树的高度是多少？

02. 一棵有 n 个结点的满二叉树有多少个分支结点和多少个叶结点？该满二叉树的高度是多少？

03. 已知完全二叉树的第 9 层有 240 个结点，则整个完全二叉树有多少个结点？有多少个叶结点？

04. 一棵高度为 h 的满 m 叉树有如下性质：根结点所在层次为第 1 层，第 h 层上的结点都是叶结点，其余各层上每个结点都有 m 棵非空子树，若按层次自顶向下，同一层自左向右，顺序从 1 开始对全部结点进行编号，试问：

1）各层的结点个数是多少？

2）编号为 i 的结点的双亲结点（若存在）的编号是多少？

3）编号为 i 的结点的第 k 个孩子结点（若存在）的编号是多少？

4）编号为 i 的结点有右兄弟的条件是什么？其右兄弟结点的编号是多少？

05. 已知一棵二叉树按顺序存储结构进行存储，设计一个算法，求编号分别为 i 和 j 的两个结点的最近的公共祖先结点的值。

5.2.4　答案与解析

关注公众号【乘龙考研】
一手更新 稳定有保障

一、单项选择题

01. C

在二叉树中，若某个结点只有一个孩子，则这个孩子的左右次序是确定的；而在度为 2 的有序树中，若某个结点只有一个孩子，则这个孩子就无须区分其左右次序，选项 A 错误。选项 B 仅当是完全二叉树时才有意义，对于任意一棵二叉树，高度可能为 $\lfloor \log_2 n \rfloor + 1 \sim n$。在完全二叉树中，若有度为 1 的结点，则只可能有一个，且该结点只有左孩子而无右孩子，选项 C 正确。完全二叉树的高度为 $\lceil \log_2(n+1) \rceil$ 或 $\lfloor \log_2 n \rfloor + 1$，也可以通过举例 $n = 4$ 来排除，选项 D 错误。

02. A

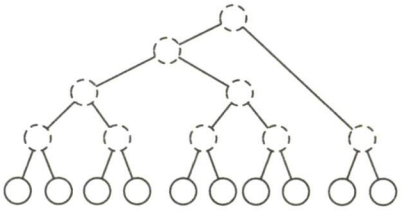
在完全二叉树中，叶结点的双亲的左兄弟的孩子一定在其前面（且一定存在），故双亲的左兄弟（若存在）一定不是叶结点，选项 A 正确。n_0 应等于 $n_2 + 1$，选项 B 错误。完全二叉树和满二叉树均可以采用顺序存储结构，选项 C 错误。第 i 个结点的左孩子不一定存在，选项 D 错误。

选项 B 的这种通用公式适用于所有二叉树，我们应能立即联想到采用特殊值代入法验证，如画一个只含 3 个结点的满二叉树的草图来验证是否满足条件。

03. B

由二叉树的性质 $n_0 = n_2 + 1$，得 $n_2 = n_0 - 1 = 10 - 1 = 9$。

【另解】画出草图，如右图所示。首先画出 10 个叶结点，然后每 2 个结点向上合并，构造一个新的度为 2 的分支结点，直到构成如右图所示的二叉树，其中度为 2 的分支结点数为 9。

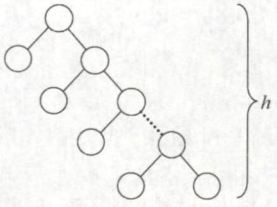


04. B

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

结点最少的情况如下图所示。除根结点层只有 1 个结点外，其他 $h-1$ 层均有两个结点，结点总数 $=2(h-1)+1=2h-1$。



关注公众号【乘龙考研】
一手更新 稳定有保障

05. C

要求满足条件的树，分析可知当这 50 个结点构成一棵完全二叉树时高度最小，$h=\lfloor \log_2 n \rfloor+1=\lfloor \log_2 50 \rfloor+1=6$。

【另解】第 1 层最多有 1 个结点，第 2 层最多有 2^1 个结点，第 3 层最多有 2^2 个结点，第 4 层最多有 2^3 个结点，以此类推，可以得到 h 最少为 6。

06. C

由二叉树的性质 1 可知 $n_0=n_2+1$，结点总数 $=2n=n_0+n_1+n_2=n_1+2n_2+1$，则 $n_1=2(n-n_2)-1$，所以 n_1 为奇数，说明该二叉树中不可能有 $2m$ 个度为 1 的结点。

07. C

当二叉树为单支树时具有最大高度，即每层上只有一个结点，最大高度为 1025。而当树为完全二叉树时，其高度最小，最小高度为 $\lfloor \log_2 n \rfloor+1=11$。

08. C

解题思路同第 4 题，第一层有一个结点，其余 $h-1$ 层上各有两个结点，总结点数 $=1+2(h-1)=15$，$h=8$。建议画出草图。

09. C

高度为 h 的完全二叉树中，第 1 层～第 $h-1$ 层构成一个高度为 $h-1$ 的满二叉树，结点个数为 $2^{h-1}-1$。第 h 层至少有一个结点，所以最少的结点个数 $=(2^{h-1}-1)+1=2^{h-1}$。

10. A

第 6 层有叶结点说明完全二叉树的高度可能为 6 或 7，显然树高为 6 时结点最少。若第 6 层上有 8 个叶结点，则前 5 层为满二叉树，故完全二叉树的结点个数最少为 $2^5-1+8=39$ 个结点。

11. A

深度为 6 的完全二叉树，第 5 层共有 $2^4=16$ 个结点。第 6 层最左边有 3 个叶结点，其对应的双亲结点为第 5 层最左边的两个结点，所以第 5 层剩余的结点均为叶结点，共有 $16-2=14$ 个，加上第 6 层的 3 个叶结点，共有 17 个叶结点。

12. D

由完全二叉树的性质，最后一个分支结点的序号为 $\lfloor 1001/2 \rfloor=500$，故叶结点个数为 501。

【另解】$n=n_0+n_1+n_2=n_0+n_1+(n_0-1)=2n_0+n_1-1$，因为 $n=1001$，而在完全二叉树中，n_1 只能取 0 或 1。当 $n_1=1$ 时，n_0 为小数，不符合题意。所以 $n_1=0$，故 $n_0=501$。

13. C

要使二叉树第 7 层的结点数最多，只考虑树高为 7 层的情况，7 层满二叉树有 127 个结点，126 仅比 127 少 1 个结点，只能少在第 7 层，故第 7 层最多有 $2^6-1=63$ 个结点。

14. B

在非空的二叉树当中，由度为 0 和 2 的结点数的关系 $n_0=n_2+1$ 可知 $n_2=123$；总结点数 $n=$

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

$n_0+n_1+n_2=247+n_1$，其最大值为 248（n_1 的取值为 1 或 0，当 $n_1=1$ 时结点最多）。注意，由完全二叉树总结点数的奇偶性可以确定 n_1 的值，但不能根据 n_0 来确定 n_1 的值。

【另解】$124<2^7=128$，故第 8 层没满，前 7 层为完全二叉树，由此可推算第 8 层可能有 120 个叶结点，第 7 层的最右 4 个为叶结点，考虑最多的情况，这 4 个叶结点中的最左边可以有 1 个左孩子（不改变叶结点数），因此结点总数 $=2^7-1+120+1=248$。

15．B

非空指针数＝总分支数＝$n-1$，空指针数＝2×结点总数－非空指针数＝$2n-(n-1)=n+1$。

【另解】在树中，1 个指针对应 1 个分支，n 个结点的树共有 $n-1$ 个分支，即 $n-1$ 个非空指针，每个结点都有 2 个指针域，故空指针数＝$2n-(n-1)=n+1$。

16．A

由完全二叉树的性质，编号为 i（$i\geqslant1$）的结点所在的层次为$\lfloor\log_2 i\rfloor+1$，若两个结点位于同一层，则一定有 $\lfloor\log_2 p\rfloor+1=\lfloor\log_2 q\rfloor+1$，因此有 $\lfloor\log_2 p\rfloor=\lfloor\log_2 q\rfloor$ 成立。

17．C

分析可知，满足条件的三叉树可以是完全三叉树，这棵树的第 i（$i\geqslant1$）层最多有 3^{i-1} 个结点。设高度为 h，则 $3^0+3^1+\cdots+3^{h-1}=(3^h-1)/2$ 是结点数的上限，问题是求解 $50\leqslant(3^h-1)/2$ 的最小 h 值，即 $h\geqslant\log_3 101$，有 $h=\lceil\log_3 101\rceil=5$。

18．D

对于高度为 h 的满二叉树，$n=2^0+2^1+\cdots+2^{h-1}=2^h-1, m=2^{h-1}$，故选 D。

19．C

第 6 层有叶结点，完全二叉树的高度可能为 6 或 7，显然树高为 7 时结点最多。完全二叉树与满二叉树相比，只是在最下一层的右边缺少了部分叶结点，而最后一层之上是个满二叉树，并且只有最后两层上有叶结点。若第 6 层上有 8 个叶结点，则前 6 层为满二叉树，而第 7 层缺失了 $8\times2=16$ 个叶结点，故完全二叉树的结点个数最多为 $2^7-1-16=111$。

20．C

求解过程与第 12 题类似。由完全二叉树的性质，最后一个分支结点的序号为$\lfloor768/2\rfloor=384$，故叶结点的个数为 $768-384=384$。

【另解】$n=n_0+n_1+n_2=n_0+n_1+(n_0-1)=2n_0+n_1-1$，其中 $n=768$，而在完全二叉树中，n_1 只能取 0 或 1，当 $n_1=0$ 时，n_0 为小数，不符合题意。所以 $n_1=1$，故 $n_0=384$。

21．A

非叶结点的度均为 2，且所有叶结点都位于同一层的完全二叉树就是满二叉树。对于一棵高度为 h 的满二叉树（空树 $h=0$），其最后一层全部是叶结点，数目为 2^{h-1}；总结点数为 2^h-1。因此当 $2^{h-1}=k$ 时，可以得到 $2^h-1=2k-1$。

22．A

二叉树采用顺序存储时，用数组下标来表示结点之间的父子关系。对于一棵高度为 5 的二叉树，为了满足任意性，其 1～5 层的所有结点都要被存储起来，即考虑为一棵高度为 5 的满二叉树，共需要存储单元的数量为 $1+2+4+8+16=31$。

二、综合应用题

01．【解答】

在非空的二叉树中，由度为 0 和度为 2 的结点之间的关系 $n_0=n_2+1$，可知 $n_2=n_0-1$。因此总结点数 $n=n_0+n_1+n_2=2n_0+n_1-1$。

①当 $n_1=1$ 时，$n=2n_0$，$h=\lceil\log_2(n+1)\rceil=\lceil\log_2(2n_0+1)\rceil$。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

②当 $n_1=0$ 时，$n=2n_0-1$，$h=\lceil\log_2(n+1)\rceil=\lceil\log_2(2n_0)\rceil=\lceil\log_2(n_0)\rceil+1$。

02.【解答】

满二叉树中 $n_1=0$，由二叉树的性质 1 可知 $n_0=n_2+1$，即 $n_2=n_0-1$，$n=n_0+n_1+n_2=2n_0-1$，则 $n_0=(n+1)/2$。分支结点个数 $n_2=n-(n+1)/2=(n-1)/2$。高度为 h 的满二叉树的结点数 $n=1+2^1+2^2+\cdots+2^{h-1}=2^h-1$，即高度 $h=\log_2(n+1)$。

03.【解答】

在完全二叉树中，若第 9 层是满的，则结点数 $=2^{9-1}=256$，而现在第 9 层只有 240 个结点，说明第 9 层未满，是最后一层。1～8 层是满的，所以总结点数 $=2^8-1+240=495$。

因为第 9 层是最后一层，所以第 9 层的结点都是叶结点。且第 9 层的 240 个结点的双亲在第 8 层中，其双亲个数为 120，即第 8 层有 120 个分支结点，其余为叶结点，所以第 8 层的叶结点个数为 $2^{8-1}-120=8$。因此，总的叶结点个数 $=8+240=248$。

【另解】总结点数 $n=n_0+n_1+n_2$，$n_2=n_0-1$，$n=n_0+n_1+n_2=2n_0+n_1-1$。若 $n_1=1$，则 $2n_0+n_1-1=2n_0=495$，不符合；若 $n_1=0$，则 $2n_0+n_1-1=2n_0-1=495$，则 $n_0=248$。

注意：对于本题，应理解完全二叉树中只有最底层的结点是不满的，其他各层的结点是满的（即第 i 层有 2^{i-1} 个结点）。

04.【解答】

1）第 1 层有 $m^0=1$ 个结点，第 2 层有 m^1 个结点，第 3 层有 m^2 个结点……一般地，第 i 层有 m^{i-1} 个结点（$1\leqslant i\leqslant h$）。

2）在 m 叉树的情形下，结点 i 的第 1 个子女编号为 $j=(i-1)\times m+2$，反过来，结点 i 的双亲的编号是 $\lfloor(i-2)/m\rfloor+1$，根结点没有双亲，所以要求 $i>1$。

3）因为结点 i 的第 1 个子女编号为 $(i-1)m+2$，若设该结点子女的序号为 $k=1,2,\cdots,m$，则第 k 个子女结点的编号为 $(i-1)m+k+1$（$1\leqslant k\leqslant m$）。

4）结点 i 不是其双亲的第 m 个子女时才有右兄弟。设其双亲编号为 j，可得 $j=\lfloor(i+m-2)/m\rfloor$，结点 j 的第 m 个子女的编号为 $(j-1)m+m+1=jm+1=\lfloor(i+m-2)/m\rfloor\times m+1$，所以当结点的编号 $i\leqslant\lfloor(i+m-2)/m\rfloor\times m$ 时才有右兄弟，右兄弟的编号为 $i+1$。或者，对于任意一个双亲结点 j，其第 m 个子女结点的编号是 $jm+1$，故若不为第 m 的子女结点，则 $(i-1)\%m!=0$。

05.【解答】

首先，必须明确二叉树中任意两个结点必然存在最近的公共祖先结点，最坏的情况下是根结点（两个结点分别在根结点的左右分支中），而且从最近的公共祖先结点到根结点的全部祖先结点都是公共的。由二叉树顺序存储的性质可知，任意一个结点 i 的双亲结点的编号为 $i/2$。求解 i 和 j 最近公共祖先结点的算法步骤如下（设从数组下标 1 开始存储）：

1）若 $i>j$，则结点 i 所在层次大于或等于结点 j 所在层次。结点 i 的双亲结点为结点 $i/2$，若 $i/2=j$，则结点 $i/2$ 是原结点 i 和结点 j 的最近公共祖先结点，若 $i/2\neq j$，则令 $i=i/2$，即以该结点 i 的双亲结点为起点，采用递归的方法继续查找。

2）若 $j>i$，则结点 j 所在层次大于或等于结点 i 所在层次。结点 j 的双亲结点为结点 $j/2$，若 $j/2=i$，则结点 $j/2$ 是原结点 i 和结点 j 的最近公共祖先结点，若 $j/2\neq i$，则令 $j=j/2$。

重复上述过程，直到找到它们最近的公共祖先结点为止。

本题代码如下：

```
ElemType Comm_Ancestor(SqTree T,int i,int j){
//本算法在二叉树中查找结点 i 和结点 j 的最近公共祖先结点
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        if(T[i]!='#'&&T[j]!='#'){          //结点存在
            while(i!=j){                   //两个编号不同时循环
                if(i>j)
                    i=i/2;                 //向上找 i 的祖先
                else
                    j=j/2;                 //向上找 j 的祖先
            }
            return  T[i];
        }
    }
```

由解题中算法的步骤描述可知，本题也很容易地联想到采用递归的方法求解。

关注公众号【乘龙考研】
一手更新 稳定有保障

5.3　二叉树的遍历和线索二叉树

5.3.1　二叉树的遍历

二叉树的遍历是指按某条搜索路径访问树中每个结点，使得每个结点均被访问一次，而且仅被访问一次。由于二叉树是一种非线性结构，每个结点都可能有两棵子树，因而需要寻找一种规律，以便使二叉树上的结点能排列在一个线性队列上，进而便于遍历。

由二叉树的递归定义可知，遍历一棵二叉树便要决定对根结点 N、左子树 L 和右子树 R 的访问顺序。按照先遍历左子树再遍历右子树的原则，常见的遍历次序有先序（NLR）、中序（LNR）和后序（LRN）三种遍历算法，其中“序”指的是根结点在何时被访问。

1．先序遍历

先序遍历（PreOrder）的操作过程如下。

若二叉树为空，则什么也不做；否则，

1）访问根结点；

2）先序遍历左子树；

3）先序遍历右子树。

对应的递归算法如下：

```
void PreOrder(BiTree T){
    if(T!=NULL){
        visit(T);                      //访问根结点
        PreOrder(T->lchild);           //递归遍历左子树
        PreOrder(T->rchild);           //递归遍历右子树
    }
}
```

对于图 5.4 所示的二叉树，先序遍历所得到的结点序列为 1 2 4 6 3 5。

2．中序遍历

中序遍历（InOrder）的操作过程如下。

若二叉树为空，则什么也不做；否则，

1）中序遍历左子树；

2）访问根结点；

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

3）中序遍历右子树。

对应的递归算法如下：

```
void InOrder(BiTree T){
    if(T!=NULL){
        InOrder(T->lchild);      //递归遍历左子树
        visit(T);                //访问根结点
        InOrder(T->rchild);      //递归遍历右子树
    }
}
```

对于图 5.4 所示的二叉树，中序遍历所得到的结点序列为 2 6 4 1 3 5。

3．后序遍历

后序遍历（PostOrder）的操作过程如下。

若二叉树为空，则什么也不做；否则，

1）后序遍历左子树；

2）后序遍历右子树；

3）访问根结点。

对应的递归算法如下：

```
void PostOrder(BiTree T){
    if(T!=NULL){
        PostOrder(T->lchild);   //递归遍历左子树
        PostOrder(T->rchild);   //递归遍历右子树
        visit(T);               //访问根结点
    }
}
```

对于图 5.4 所示的二叉树，后序遍历所得到的结点序列为 6 4 2 5 3 1。

三种遍历算法中，递归遍历左、右子树的顺序都是固定的，只是访问根结点的顺序不同。不管采用哪种遍历算法，每个结点都访问一次且仅访问一次，故时间复杂度都是 $O(n)$。在递归遍历中，递归工作栈的栈深恰好为树的深度，所以在最坏情况下，二叉树是有 n 个结点且深度为 n 的单支树，遍历算法的空间复杂度为 $O(n)$。

注意：以上三种遍历方式及算法描述是简单易懂的，读者需要将它们作为模板来记忆，考研中的很多题目都是基于这三个模板延伸出来的。

4．递归算法和非递归算法的转换

在上节介绍的三种遍历算法中，暂时抹去和递归无关的 visit()语句，则 3 个遍历算法完全相同，因此，从递归执行过程的角度看先序、中序和后序遍历也是完全相同的。

图 5.7 用带箭头的虚线表示了这三种遍历算法的递归执行过程。其中，向下的箭头表示更深一层的递归调用，向上的箭头表示从递归调用退出返回；虚线旁的三角形、圆形和方形内的字符分别表示在先序、中序和后序遍历的过程中访问根结点时输出的信息。例如，由于中序遍历中访问结点是在遍历左子树之后、遍历右子树之前进行的，则带圆形的字符标在向左递归返回和向右递归调用之间。由此，只要沿虚线从 1 出发到 2 结束，将沿途所见的三角形（或圆形或方形）内的字符记下，便得到遍历二叉树的先序（或中序或后序）序列。例如，在图 5.7 中，沿虚线游走可以分别得到先序序列为 $ABDEC$、中序序列为 $DBEAC$、后序序列为 $DEBCA$。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

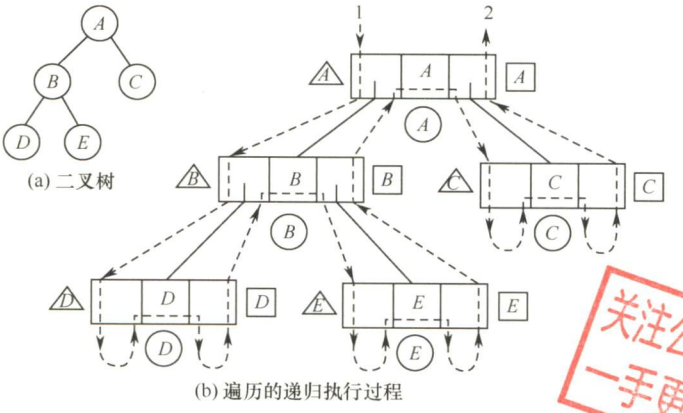


图 5.7　三种遍历过程示意图

关注公众号【乘龙考研】
一手更新 稳定有保障

借助栈的思路，我们来分析中序遍历的访问过程：

①沿着根的左孩子，依次入栈，直到左孩子为空，说明已找到可以输出的结点，此时栈内元素依次为 *A B D*。②栈顶元素出栈并访问：若其右孩子为空，继续执行②；若其右孩子不空，将右子树转执行①。栈顶 *D* 出栈并访问，它是中序序列的第一个结点；*D* 右孩子为空，栈顶 *B* 出栈并访问；*B* 右孩子不空，将其右孩子 *E* 入栈，*E* 左孩子为空，栈顶 *E* 出栈并访问；*E* 右孩子为空，栈顶 *A* 出栈并访问；*A* 右孩子不空，将其右孩子 *C* 入栈，*C* 左孩子为空，栈顶 *C* 出栈并访问。由此得到中序序列 *D B E A C*。读者可根据上述分析画出遍历过程的出入栈示意图。

根据分析可以写出中序遍历的非递归算法如下：

```
void InOrder2(BiTree T){
InitStack(S); BiTree p=T;          //初始化栈 S；p 是遍历指针
    while(p||!IsEmpty(S)){         //栈不空或 p 不空时循环
        if(p){                     //一路向左
            Push(S,p);             //当前结点入栈
            p=p->lchild;           //左孩子不空，一直向左走
        }
        else{                      //出栈，并转向出栈结点的右子树
            Pop(S,p);visit(p);     //栈顶元素出栈，访问出栈结点
            p=p->rchild;           //向右子树走，p 赋值为当前结点的右孩子
        }                          //返回 while 循环继续进入 if-else 语句
    }
}
```

先序遍历和中序遍历的基本思想是类似的，只需把访问结点操作放在入栈操作的前面，读者可以参考中序遍历的过程说明自行模拟出入栈示意图。先序遍历的非递归算法如下：

```
void PreOrder2(BiTree T){
InitStack(S); BiTree p=T;          //初始化栈 S；p 是遍历指针
    while(p||!IsEmpty(S)){         //栈不空或 p 不空时循环
        if(p){                     //一路向左
            visit(p);Push(S,p);    //访问当前结点，并入栈
            p=p->lchild;           //左孩子不空，一直向左走
        }
        else{                      //出栈，并转向出栈结点的右子树
            Pop(S,p);              //栈顶元素出栈
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
                p=p->rchild;          //向右子树走，p 赋值为当前结点的右孩子
            }                         //返回 while 循环继续进入 if-else 语句
        }
    }
```

后序遍历的非递归实现是三种遍历方法中最难的。因为在后序遍历中，要保证左孩子和右孩子都已被访问并且左孩子在右孩子前访问才能访问根结点，这就为流程的控制带来了难题。

后序非递归遍历算法的思路分析：从根结点开始，将其入栈，然后沿其左子树一直往下搜索，直到搜索到没有左孩子的结点，但是此时不能出栈并访问，因为如果其有右子树，还需按相同的规则对其右子树进行处理。直至上述操作进行不下去，若栈顶元素想要出栈被访问，要么右子树为空，要么右子树刚被访问完（此时左子树早已访问完），这样就保证了正确的访问顺序。

后序遍历的非递归算法见本节综合应用题 03 的解析部分。

按后序非递归算法遍历图 5.7(a)中的二叉树，当访问到 E 时，A, B, D 都已入过栈，对于后序非递归遍历，当一个结点的左右子树都被访问后才会出栈，图中 D 已出栈，此时栈内还有 A 和 B，这是 E 的全部祖先。实际上，访问一个结点 p 时，栈中结点恰好是 p 结点的所有祖先，从栈底到栈顶结点再加上 p 结点，刚好构成从根结点到 p 结点的一条路径。在很多算法设计中都可以利用这一思路来求解，如求根到某结点的路径、求两个结点的最近公共祖先等。

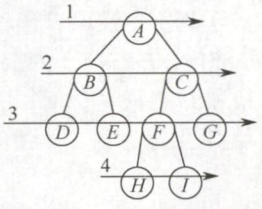


图 5.8　二叉树的层次遍历

5. 层次遍历

图 5.8 所示为二叉树的层次遍历，即按照箭头所指方向，按照 1, 2, 3, 4 的层次顺序，对二叉树中的各个结点进行访问。

要进行层次遍历，需要借助一个队列。首先将二叉树根结点入队，然后出队，访问出队结点，若它有左子树，则将左子树根结点入队；若它有右子树，则将右子树根结点入队。完成入队后出队，访问出队结点……如此反复，直至队列为空。

二叉树的层次遍历算法如下：

```
void LevelOrder(BiTree T){
    InitQueue(Q);                     //初始化辅助队列
    BiTree p;
    EnQueue(Q,T);                     //将根结点入队
    while(!IsEmpty(Q)){               //队列不空则循环
        DeQueue(Q, p);                //队头结点出队
        visit(p);                     //访问出队结点
        if(p->lchild!=NULL)
            EnQueue(Q,p->lchild);     //左子树不空，则左子树根结点入队
        if(p->rchild!=NULL)
            EnQueue(Q,p->rchild);     //右子树不空，则右子树根结点入队
    }
}
```

上述二叉树层次遍历的算法，读者在复习过程中应将其作为一个模板，在熟练掌握其执行过程的基础上来记忆，并达到熟练手写的程度。这样才能将层次遍历模板应用于各种题目之中。

注意：遍历是二叉树各种操作的基础，可以在遍历的过程中对结点进行各种操作。例如，对于一棵已知树求结点的双亲、求结点的孩子结点、求二叉树的深度、求二叉树的叶结点个数、判断两棵二叉树是否相同等。所有这些操作都建立在二叉树遍历的基础上，因此必须掌握二叉树的

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

各种遍历过程，并能灵活运用以解决各种问题。

6．由遍历序列构造二叉树

由二叉树的先序序列和中序序列可以唯一地确定一棵二叉树。

在先序遍历序列中，第一个结点一定是二叉树的根结点；而在中序遍历中，根结点必然将中序序列分割成两个子序列，前一个子序列是根结点的左子树的中序序列，后一个子序列是根结点的右子树的中序序列。根据这两个子序列，在先序序列中找到对应的左子序列和右子序列。在先序序列中，左子序列的第一个结点是左子树的根结点，右子序列的第一个结点是右子树的根结点。如此递归地进行下去，便能唯一地确定这棵二叉树。

同理，由二叉树的后序序列和中序序列也可以唯一地确定一棵二叉树。

因为后序序列的最后一个结点就如同先序序列的第一个结点，可以将中序序列分割成两个子序列，然后采用类似的方法递归地进行划分，进而得到一棵二叉树。

由二叉树的层序序列和中序序列也可以唯一地确定一棵二叉树，实现方法留给读者思考。需要注意的是，若只知道二叉树的先序序列和后序序列，则无法唯一确定一棵二叉树。

例如，求先序序列（*ABCDEFGHI*）和中序序列（*BCAEDGHFI*）所确定的二叉树。

首先，由先序序列可知 *A* 为二叉树的根结点。中序序列中 *A* 之前的 *BC* 为左子树的中序序列，*EDGHFI* 为右子树的中序序列。然后，由先序序列可知 *B* 是左子树的根结点，*D* 是右子树的根结点。以此类推，就能将剩下的结点继续分解下去，最后得到的二叉树如图 5.9(c)所示。

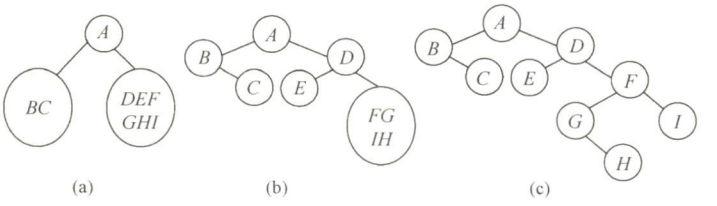


图 5.9　一棵二叉树的构造过程

关注公众号【乘龙考研】
一手更新 稳定有保障

5.3.2　线索二叉树

1．线索二叉树的基本概念

遍历二叉树是以一定的规则将二叉树中的结点排列成一个线性序列，从而得到几种遍历序列，使得该序列中的每个结点（第一个和最后一个结点除外）都有一个直接前驱和直接后继。

传统的二叉链表存储仅能体现一种父子关系，不能直接得到结点在遍历中的前驱或后继。前面提到，在含 n 个结点的二叉树中，有 $n+1$ 个空指针。这是因为每个叶结点都有 2 个空指针，每个度为 1 的结点都有 1 个空指针，空指针总数为 $2n_0+n_1$，又 $n_0=n_2+1$，所以空指针总数为 $n_0+n_1+n_2+1=n+1$。由此设想能否利用这些空指针来存放指向其前驱或后继的指针？这样就可以像遍历单链表那样方便地遍历二叉树。引入线索二叉树正是为了加快查找结点前驱和后继的速度。

规定：若无左子树，令 `lchild` 指向其前驱结点；若无右子树，令 `rchild` 指向其后继结点。如图 5.10 所示，还需增加两个标志域标识指针域，以指向左（右）孩子或前驱（后继）。

lchild	ltag	data	rtag	rchild

图 5.10　线索二叉树的结点结构

其中，标志域的含义如下：

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

$$ltag=\begin{cases}0, & \text{lchild域指示结点的左孩子}\\ 1, & \text{lchild域指示结点的前驱}\end{cases}$$

$$rtag=\begin{cases}0, & \text{rchild域指示结点的右孩子}\\ 1, & \text{rchild域指示结点的后继}\end{cases}$$

线索二叉树的存储结构描述如下：

```
typedef struct ThreadNode{
    ElemType data;                          //数据元素
    struct ThreadNode *lchild,*rchild;      //左、右孩子指针
    int ltag,rtag;                          //左、右线索标志
}ThreadNode,*ThreadTree;
```

以这种结点结构构成的二叉链表作为二叉树的存储结构，称为*线索链表*，其中指向结点前驱和后继的指针称为*线索*。加上线索的二叉树称为*线索二叉树*。

2. 中序线索二叉树的构造

二叉树的*线索化*是将二叉链表中的空指针改为指向前驱或后继的线索。而前驱或后继的信息只有在遍历时才能得到，因此线索化的实质就是遍历一次二叉树。

以中序线索二叉树的建立为例。附设指针 pre 指向刚刚访问过的结点，指针 p 指向正在访问的结点，即 pre 指向 p 的前驱。在中序遍历的过程中，检查 p 的左指针是否为空，若为空就将它指向 pre；检查 pre 的右指针是否为空，若为空就将它指向 p，如图 5.11 所示。

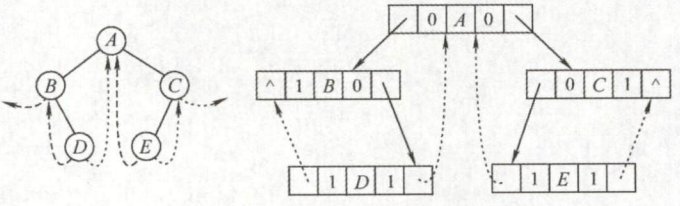


图 5.11 中序线索二叉树及其二叉链表示

通过中序遍历对二叉树线索化的递归算法如下：

```
void InThread(ThreadTree &p,ThreadTree &pre){
    if(p!=NULL){
        InThread(p->lchild,pre);    //递归，线索化左子树
        if(p->lchild==NULL){        //左子树为空，建立前驱线索
            p->lchild=pre;
            p->ltag=1;
        }
        if(pre!=NULL&&pre->rchild==NULL){
            pre->rchild=p;          //建立前驱结点的后继线索
            pre->rtag=1;
        }
        pre=p;                      //标记当前结点成为刚刚访问过的结点
        InThread(p->rchild,pre);    //递归，线索化右子树
    }//if(p!=NULL)
}
```

通过中序遍历建立中序线索二叉树的主过程算法如下：

```
void CreateInThread(ThreadTree T){
    ThreadTree pre=NULL;
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        if(T!=NULL){                    //非空二叉树，线索化
            InThread(T,pre);            //线索化二叉树
            pre->rchild=NULL;           //处理遍历的最后一个结点
            pre->rtag=1;
        }
    }
```

为了方便，可以在二叉树的线索链表上也添加一个头结点，令其 lchild 域的指针指向二叉树的根结点，其 rchild 域的指针指向中序遍历时访问的最后一个结点；令二叉树中序序列中的第一个结点的 lchild 域指针和最后一个结点的 rchild 域指针均指向头结点。这好比为二叉树建立了一个双向线索链表，方便从前往后或从后往前对线索二叉树进行遍历，如图 5.12 所示。

关注公众号【乘龙考研】
一手更新 稳定有保障

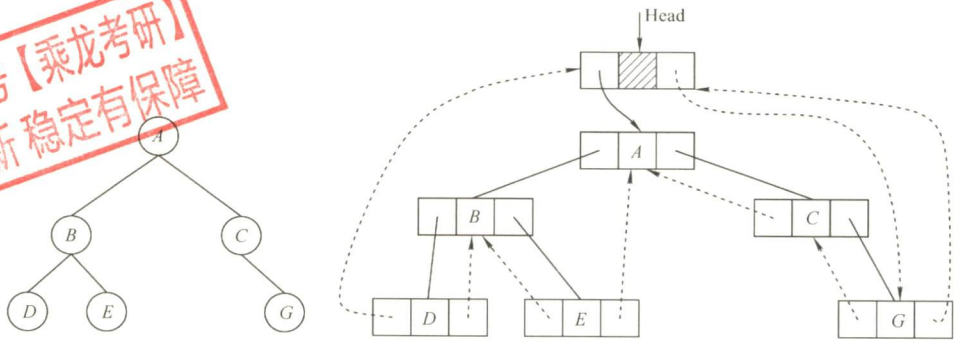


图 5.12　带头结点的中序线索二叉树

3. 中序线索二叉树的遍历

中序线索二叉树的结点中隐含了线索二叉树的前驱和后继信息。在对其进行遍历时，只要先找到序列中的第一个结点，然后依次找结点的后继，直至其后继为空。在中序线索二叉树中找结点后继的规律是：若其右标志为“1”，则右链为线索，指示其后继，否则遍历右子树中第一个访问的结点（右子树中最左下的结点）为其后继。不含头结点的线索二叉树的遍历算法如下。

1）求中序线索二叉树中中序序列下的第一个结点：

```
ThreadNode *Firstnode(ThreadNode *p){
    while(p->ltag==0) p=p->lchild;  //最左下结点(不一定是叶结点)
    return p;
}
```

2）求中序线索二叉树中结点 p 在中序序列下的后继：

```
ThreadNode *Nextnode(ThreadNode *p){
    if(p->rtag==0) return Firstnode(p->rchild);
    else return p->rchild;   //rtag==1 直接返回后继线索
}
```

请读者自行分析并完成求中序线索二叉树的最后一个结点和结点 p 前驱的运算[①]。

3）利用上面两个算法，可以写出不含头结点的中序线索二叉树的中序遍历的算法：

```
void Inorder(ThreadNode *T){
    for(ThreadNode *p=Firstnode(T);p!=NULL; p=Nextnode(p))
        visit(p);
}
```

① 将函数 1 中的 ltag 和 lchild 换成 rtag 和 rchild，即为求中序线索二叉树的最后一个结点。将函数 2 中的 rtag 和 rchild 换成 ltag 和 lchild，此外调用函数改为求中序线索二叉树的最后一个结点，即为求中序线索二叉树中结点 p 的前驱。

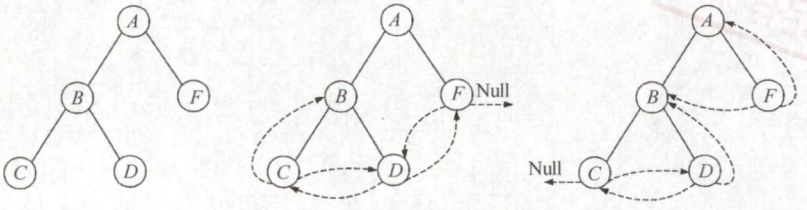
有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

4. 先序线索二叉树和后序线索二叉树

上面给出了建立中序线索二叉树的代码，建立先序线索二叉树和建立后序线索二叉树的代码类似，只需变动线索化改造的代码段与调用线索化左右子树递归函数的位置。

以图 5.13(a)的二叉树为例给出手动求先序线索二叉树的过程：先序序列为 *A B C D F*，然后依次判断每个结点的左右链域，如果为空则将其改造为线索。结点 *A*, *B* 均有左右孩子；结点 *C* 无左孩子，将左链域指向前驱 *B*，无右孩子，将右链域指向后继 *D*；结点 *D* 无左孩子，将左链域指向前驱 *C*，无右孩子，将右链域指向后继 *F*；结点 *F* 无左孩子，将左链域指向前驱 *D*，无右孩子，也无后继，故置空，得到的先序线索二叉树如图 5.13(b)所示。求后序线索二叉树的过程：后序序列为 *C D B F A*，结点 *C* 无左孩子，也无前驱故置空，无右孩子，将右链域指向后继 *D*；结点 *D* 无左孩子，将左链域指向前驱 *C*，无右孩子，将右链域指向后继 *B*；结点 *F* 无左孩子，将左链域指向前驱 *B*，无右孩子，将右链域指向后继 *A*，得到的后序线索二叉树如图 5.13(c)所示。



(a) 一棵二叉树　　(b) 先序线索二叉树　　(c) 后序线索二叉树

图 5.13　先序线索二叉树和后序线索二叉树

如何在先序线索二叉树中找结点的后继？如果有左孩子，则左孩子就是其后继；如果无左孩子但有右孩子，则右孩子就是其后继；如果为叶结点，则右链域直接指示了结点的后继。

在后序线索二叉树中找结点的后继较为复杂，可分三种情况：①若结点 *x* 是二叉树的根，则其后继为空；②若结点 *x* 是其双亲的右孩子，或是其双亲的左孩子且其双亲没有右子树，则其后继即为双亲；③若结点 *x* 是其双亲的左孩子，且其双亲有右子树，则其后继为双亲的右子树上按后序遍历列出的第一个结点。图 5.13(c)中找结点 *B* 的后继无法通过链域找到，可见在后序线索二叉树上找后继时需知道结点双亲，即需采用带标志域的三叉链表作为存储结构。

5.3.3 本节试题精选

一、单项选择题

01. 在下列关于二叉树遍历的说法中，正确的是（　）。

A. 若有一个结点是二叉树中某个子树的中序遍历结果序列的最后一个结点，则它一定是该子树的前序遍历结果序列的最后一个结点

B. 若有一个结点是二叉树中某个子树的前序遍历结果序列的最后一个结点，则它一定是该子树的中序遍历结果序列的最后一个结点

C. 若有一个叶结点是二叉树中某个子树的中序遍历结果序列的最后一个结点，则它一定是该子树的前序遍历结果序列的最后一个结点

D. 若有一个叶结点是二叉树中某个子树的前序遍历结果序列的最后一个结点，则它一定是该子树的中序遍历结果序列的最后一个结点

02. 在任何一棵二叉树中，若结点 *a* 有左孩子 *b*、右孩子 *c*，则在结点的先序序列、中序序列、后序序列中，（　）。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

A. 结点 b 一定在结点 a 的前面　　B. 结点 a 一定在结点 c 的前面
C. 结点 b 一定在结点 c 的前面　　D. 结点 a 一定在结点 b 的前面

03. 设 n, m 为一棵二叉树上的两个结点，在中序遍历时，n 在 m 前的条件是（ ）。
A. n 在 m 右方　B. n 是 m 祖先　C. n 在 m 左方　D. n 是 m 子孙

04. 设 n, m 为一棵二叉树上的两个结点，在后序遍历时，n 在 m 前的条件是（ ）。
A. n 在 m 右方　B. n 是 m 祖先　C. n 在 m 左方　D. n 是 m 子孙

05. 在二叉树中有两个结点 m 和 n，若 m 是 n 的祖先，则使用（ ）可以找到从 m 到 n 的路径。
A. 先序遍历　B. 中序遍历　C. 后序遍历　D. 层次遍历

06. 在二叉树的前序序列、中序序列和后序序列中，所有叶结点的先后顺序（ ）。
A. 都不相同　　B. 完全相同
C. 前序和中序相同，而与后序不同　　D. 中序和后序相同，而与前序不同

07. 对二叉树的结点从 1 开始进行连续编号，要求每个结点的编号大于其左、右孩子的编号，同一结点的左、右孩子中，其左孩子的编号小于其右孩子的编号，可采用（ ）次序的遍历实现编号。
A. 先序遍历　B. 中序遍历　C. 后序遍历　D. 层次遍历

08. 前序为 A, B, C，后序为 C, B, A 的二叉树共有（ ）。
A. 1 棵　B. 2 棵　C. 3 棵　D. 4 棵

09. 一棵非空的二叉树的先序遍历序列与后序遍历序列正好相反，则该二叉树一定满足（ ）。
A. 所有的结点均无左孩子　　B. 所有的结点均无右孩子
C. 只有一个叶结点　　D. 是任意一棵二叉树

10. 设结点 X 和 Y 是二叉树中任意的两个结点。在该二叉树的先序遍历序列中 X 在 Y 之前，而在其后序遍历序列中 X 在 Y 之后，则 X 和 Y 的关系是（ ）。
A. X 是 Y 的左兄弟　　B. X 是 Y 的右兄弟
C. X 是 Y 的祖先　　D. X 是 Y 的后裔

11. 若二叉树中结点的先序序列是 $\cdots a \cdots b \cdots$，中序序列是 $\cdots b \cdots a \cdots$，则（ ）。
A. 结点 a 和结点 b 分别在某结点的左子树和右子树中
B. 结点 b 在结点 a 的右子树中
C. 结点 b 在结点 a 的左子树中
D. 结点 a 和结点 b 分别在某结点的两棵非空子树中

关注公众号【乘龙考研】
一手更新 稳定有保障

12. 一棵二叉树的前序遍历序列为 1234567，它的中序遍历序列可能是（ ）。
A. 3124567　B. 1234567　C. 4135627　D. 1463572

13. 下列序列中，不能唯一地确定一棵二叉树的是（ ）。
A. 层次序列和中序序列　　B. 先序序列和中序序列
C. 后序序列和中序序列　　D. 先序序列和后序序列

14. 已知一棵二叉树的后序序列为 $DABEC$，中序序列为 $DEBAC$，则先序序列为（ ）。
A. $ACBED$　B. $DECAB$　C. $DEABC$　D. $CEDBA$

15. 已知一棵二叉树的先序遍历结果为 $ABCDEF$，中序遍历结果为 $CBAEDF$，则后序遍历的结果为（ ）。
A. $CBEFDA$　B. $FEDCBA$　C. $CBEDFA$　D. 不确定

16. 已知一棵二叉树的层次序列为 $ABCDEF$，中序序列为 $BADCFE$，则先序序列为（ ）。
A. $ACBEDF$　B. $ABCDEF$　C. $BDFECA$　D. $FCEDBA$

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

17. 引入线索二叉树的目的是（ ）。

A. 加快查找结点的前驱或后继的速度　B. 为了能在二叉树中方便插入和删除

C. 为了能方便找到双亲　D. 使二叉树的遍历结果唯一

18. 线索二叉树是一种（ ）结构。

A. 逻辑　B. 逻辑和存储　C. 物理　D. 线性

19. n 个结点的线索二叉树上含有的线索数为（ ）。

A. $2n$　B. $n-1$　C. $n+1$　D. n

20. 判断线索二叉树中*p结点有右孩子结点的条件是（ ）。

A. `p!=NULL`　B. `p->rchild!=NULL`

C. `p->rtag==0`　D. `p->rtag==1`

21. 一棵左子树为空的二叉树在先序线索化后，其中空的链域的个数是（ ）。

A. 不确定　B. 0个　C. 1个　D. 2个

22. 在线索二叉树中，下列说法不正确的是（ ）。

A. 在中序线索树中，若某结点有右孩子，则其后继结点是它的右子树的最左下结点

B. 在中序线索树中，若某结点有左孩子，则其前驱结点是它的左子树的最右下结点

C. 线索二叉树是利用二叉树的 $n+1$ 个空指针来存放结点的前驱和后继信息的

D. 每个结点通过线索都可以直接找到它的前驱和后继

23. 二叉树在线索化后，仍不能有效求解的问题是（ ）。

A. 先序线索二叉树中求先序后继　B. 中序线索二叉树中求中序后继

C. 中序线索二叉树中求中序前驱　D. 后序线索二叉树中求后序后继

24. 若 X 是二叉中序线索树中一个有左孩子的结点，且 X 不为根，则 X 的前驱为（ ）。

A. X 的双亲　B. X 的右子树中最左的结点

C. X 的左子树中最右的结点　D. X 的左子树中最右的叶结点

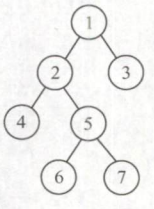
25. （ ）的遍历仍需要栈的支持。

A. 前序线索树　B. 中序线索树　C. 后序线索树　D. 所有线索树

26. 某二叉树的先序序列和后序序列正好相反，则该二叉树一定是（ ）。

A. 空或只有一个结点　B. 高度等于其结点数

C. 任意一个结点无左孩子　D. 任意一个结点无右孩子

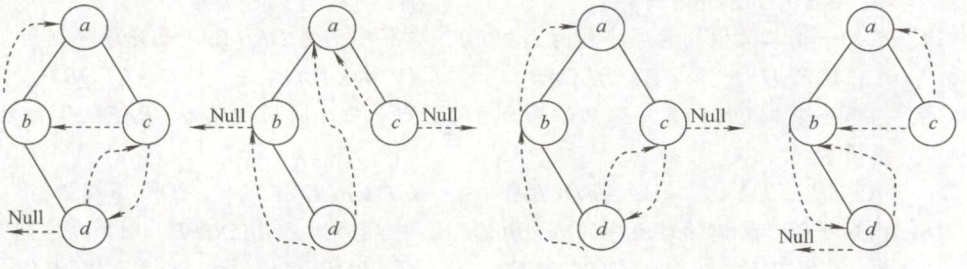


27.【2009统考真题】给定二叉树如右图所示。设N代表二叉树的根，L代表根结点的左子树，R代表根结点的右子树。若遍历后的结点序列是3175624，则其遍历方式是（ ）。

A. LRN　B. NRL

C. RLN　D. RNL

28.【2010统考真题】下列线索二叉树中（用虚线表示线索），符合后序线索树定义的是（ ）。



有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

29. 【2011 统考真题】一棵二叉树的前序遍历序列和后序遍历序列分别为 1, 2, 3, 4 和 4, 3, 2, 1，该二叉树的中序遍历序列不会是（　）。

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

30. 【2012 统考真题】若一棵二叉树的前序遍历序列为 a, e, b, d, c，后序遍历序列为 b, c, d, e, a，则根结点的孩子结点（　）。

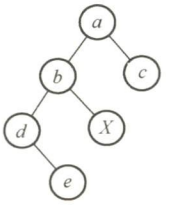
A. 只有 e　　B. 有 e、b　　C. 有 e、c　　D. 无法确定

31. 【2013 统考真题】若 X 是后序线索二叉树中的叶结点，且 X 存在左兄弟结点 Y，则 X 的右线索指向的是（　）。

A. X 的父结点　　B. 以 Y 为根的子树的最左下结点

C. X 的左兄弟结点 Y　　D. 以 Y 为根的子树的最右下结点

32. 【2014 统考真题】若对右图所示的二叉树进行中序线索化，则结点 X 的左、右线索指向的结点分别是（　）。



A. e, c　　B. e, a

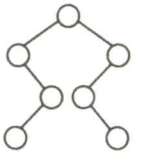
C. d, c　　D. b, a

33. 【2015 统考真题】先序序列为 a, b, c, d 的不同二叉树的个数是（　）。

A. 13　　B. 14

C. 15　　D. 16

34. 【2017 统考真题】某二叉树的树形如右图所示，其后序序列为 e, a, c, b, d, g, f，树中与结点 a 同层的结点是（　）。



A. c　　B. d

C. f　　D. g

35. 【2017 统考真题】要使一棵非空二叉树的先序序列与中序序列相同，其所有非叶结点须满足的条件是（　）。

A. 只有左子树　　B. 只有右子树

C. 结点的度均为 1　　D. 结点的度均为 2

36. 【2022 统考真题】若结点 p 与 q 在二叉树 T 的中序遍历序列中相邻，且 p 在 q 之前，则下列 p 与 q 的关系中，不可能的是（　）。

I. q 是 p 的双亲　　II. q 是 p 的右孩子

III. q 是 p 的右兄弟　　IV. q 是 p 的双亲的双亲

A. 仅 I　　B. 仅 III　　C. 仅 II、III　　D. 仅 II、IV

关注公众号【乘龙考研】
一手更新 稳定有保障

二、综合应用题

01. 若某非空二叉树的先序序列和后序序列正好相反，则该二叉树的形态是什么？

02. 若某非空二叉树的先序序列和后序序列正好相同，则该二叉树的形态是什么？

03. 编写后序遍历二叉树的非递归算法。

04. 试给出二叉树的自下而上、从右到左的层次遍历算法。

05. 假设二叉树采用二叉链表存储结构，设计一个非递归算法求二叉树的高度。

06. 设一棵二叉树中各结点的值互不相同，其先序遍历序列和中序遍历序列分别存于两个一维数组 A[1...n] 和 B[1...n] 中，试编写算法建立该二叉树的二叉链表。

07. 二叉树按二叉链表形式存储，试编写一个判别给定二叉树是否是完全二叉树的算法。

08. 假设二叉树采用二叉链表存储结构存储，试设计一个算法，计算一棵给定二叉树的所有双分支结点个数。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

09. 设树 B 是一棵采用链式结构存储的二叉树，编写一个把树 B 中所有结点的左、右子树进行交换的函数。

10. 假设二叉树采用二叉链存储结构存储，设计一个算法，求先序遍历序列中第 k（$1 \leqslant k \leqslant$ 二叉树中结点个数）个结点的值。

11. 已知二叉树以二叉链表存储，编写算法完成：对于树中每个元素值为 x 的结点，删除以它为根的子树，并释放相应的空间。

12. 在二叉树中查找值为 x 的结点，试编写算法（用 C 语言）打印值为 x 的结点的所有祖先，假设值为 x 的结点不多于一个。

13. 设一棵二叉树的结点结构为(LLINK,INFO,RLINK)，ROOT 为指向该二叉树根结点的指针，p和q分别为指向该二叉树中任意两个结点的指针，试编写算法ANCESTOR(ROOT,p,q,r)，找到 p 和 q 的最近公共祖先结点 r。

14. 假设二叉树采用二叉链表存储结构，设计一个算法，求非空二叉树 b 的宽度（即具有结点数最多的那一层的结点个数）。

15. 设有一棵满二叉树（所有结点值均不同），已知其先序序列为 *pre*，设计一个算法求其后序序列 *post*。

16. 设计一个算法将二叉树的叶结点按从左到右的顺序连成一个单链表，表头指针为 head。二叉树按二叉链表方式存储，链接时用叶结点的右指针域来存放单链表指针。

17. 试设计判断两棵二叉树是否相似的算法。所谓二叉树 T_1 和 T_2 相似，指的是 T_1 和 T_2 都是空的二叉树或都只有一个根结点；或者 T_1 的左子树和 T_2 的左子树是相似的，且 T_1 的右子树和 T_2 的右子树是相似的。

18. 写出在中序线索二叉树里查找指定结点在后序的前驱结点的算法。

19. 【2014 统考真题】二叉树的带权路径长度（WPL）是二叉树中所有叶结点的带权路径长度之和。给定一棵二叉树 T，采用二叉链表存储，结点结构为

left	weight	right

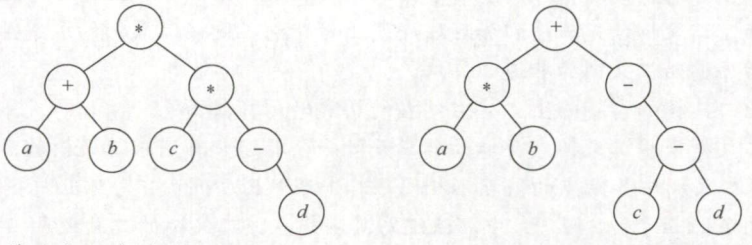
其中叶结点的 weight 域保存该结点的非负权值。设 root 为指向 T 的根结点的指针，请设计求 T 的 WPL 的算法，要求：

1）给出算法的基本设计思想。

2）使用 C 或 C++语言，给出二叉树结点的数据类型定义。

3）根据设计思想，采用 C 或 C++语言描述算法，关键之处给出注释。

20. 【2017 统考真题】请设计一个算法，将给定的表达式树（二叉树）转换为等价的中缀表达式（通过括号反映操作符的计算次序）并输出。例如，当下列两棵表达式树作为算法的输入时：



输出的等价中缀表达式分别为(a+b)*(c*(-d))和(a*b)+(-(c-d))。

二叉树结点定义如下：

```
typedef struct node{
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
    char data[10];                  //存储操作数或操作符
    struct node *left, *right;
}BTree;
```

要求：

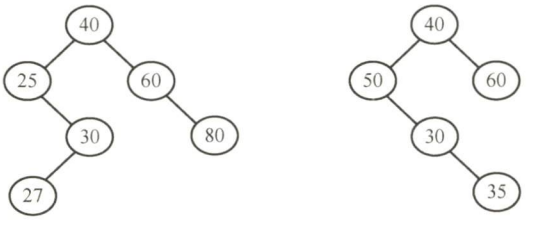
1）给出算法的基本设计思想。

2）根据设计思想，采用 C 或 C++语言描述算法，关键之处给出注释。

21.【2022 统考真题】已知非空二叉树 T 的结点值均为正整数，采用顺序存储方式保存，数据结构定义如下：

```
typedef struct {                    // MAX_SIZE 为已定义常量
     int SqBiTNode[MAX_SIZE];       // 保存二叉树结点值的数组
     int ElemNum;                   // 实际占用的数组元素个数
} SqBiTree;
```

T 中不存在的结点在数组 SqBiTNode 中用-1 表示。例如，对于下图所示的两棵非空二叉树 T_1 和 T_2，



二叉树 T_1　　二叉树 T_2

关注公众号【乘龙考研】一手更新 稳定有保障

T_1 的存储结果如下：

T1.SqBiTNode	40	25	60	-1	30	-1	80	-1	-1	27		

T1.ElemNum = 10

T_2 的存储结果如下：

T2.SqBiTNode	40	50	60	-1	30	-1	-1	-1	-1	-1	35	

T2.ElemNum = 11

请设计一个尽可能高效的算法，判定一棵采用这种方式存储的二叉树是否为二叉搜索树，若是，则返回 true，否则，返回 false。要求：

1）给出算法的基本设计思想。

2）根据设计思想，采用 C 或 C++语言描述算法，关键之处给出注释。

5.3.4　答案与解析

一、单项选择题

01. C

二叉树中序遍历的最后一个结点一定是从根开始沿右子女指针链走到底的结点，设用 p 指示。若结点 p 不是叶结点（其左子树非空），则前序遍历的最后一个结点在它的左子树中，选项 A、B 错；若结点 p 是叶结点，则前序与中序遍历的最后一个结点就是它，选项 C 正确。若中序遍历的最后一个结点 p 不是叶结点，它还有一个左子女 q，结点 q 是叶结点，那么结点 q 是前序遍历的最后一个结点，但不是中序遍历的最后一个结点，选项 D 错。

02. C

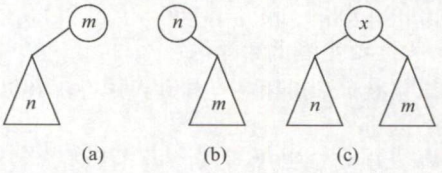
有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

这三种遍历方式中，无论哪种遍历方式，都先遍历左子树，再遍历右子树，所以结点 b 一定在结点 c 的前面访问。

03．C

中序遍历时，先访问左子树，再访问根结点，后访问右子树。n 在 m 前的 3 种可能性如左图所示，从中看出 n 总是在 m 的左方。



【另解】设 n 和 m 的最近公共祖先 p，则有以下可能：

情形 1，m 和 n 分别在 p 的左、右（右、左）分支上；情形 2，m 或 n 为 p 结点，另一结点在 p 的分支上。只有 n 和 m 分别处于 p 的左、右分支上，m 为祖先结点且 n 位于 m 的左分支，n 为祖先结点且 m 位于 n 的右分支，符合题意。

04．D

后序遍历的顺序是 LRN，若 n 在 N 的左子树，m 在 N 的右子树，则在后序遍历的过程中 n 在 m 之前访问；若 n 是 m 的子孙，设 m 在 N 的位置，则 n 无论是在 m 的左子树还是在右子树，在后序遍历的过程中 n 都在 m 之前访问。其他都不可以。选项 C 要成立，就要加上两个结点位于同一层这个条件。

05．C

在后序遍历退回时访问根结点，就可以从下向上把从 n 到 m 的路径上的结点输出，若采用非递归的算法，则当后序遍历访问到 n 时，栈中把从根到 n 的父指针的路径上的结点都记忆下来，也可以找到从 m 到 n 的路径。

06．B

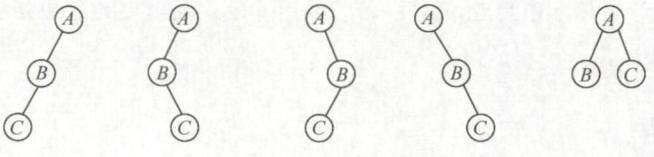
在三种遍历方式中，访问左右子树的先后顺序是不变的，只是访问根结点的顺序不同，因此叶结点的先后顺序完全相同。此外，读者可以采用特殊值法，画一个结点数为 3 的满二叉树，采用三种遍历方式来验证答案的正确性。

07．C

对每个顶点从 1 开始按序编号，要求结点编号大于其左、右孩子编号，并且左孩子编号小于右孩子编号。编号越大说明遍历顺序越靠后，因此，三者遍历顺序为先左子树、再右子树、后根结点。4 个选项中仅后序遍历满足要求。

08．D

前序为 A、B、C 的不同二叉树共有 5 种，其中后序为 C、B、A 的有 4 种（前 4 种），都是单支树，如下图所示。



09．C

非空树的先序序列和后序序列相反，即“根左右”与“左右根”顺序相反，因此树只有根结点，或者根结点只有左子树或右子树，以此类推，其子树有同样的性质。因此，树中所有非叶结点的度均为 1，即二叉树仅有一个叶结点。

10．C

设二叉树的前序遍历顺序为 NLR，后序遍历顺序为 LRN。根据题意，在前序遍历序列中 X 在 Y 之前，在后序遍历序列中 X 在 Y 之后，若设 X 在根结点的位置，Y 在其左子树或右子树中，

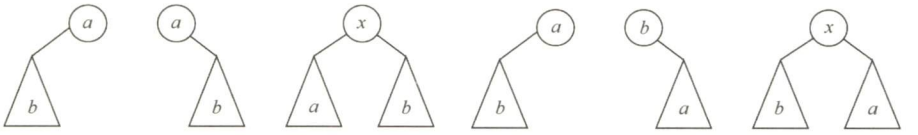
有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类目更网盘群&笔记类资料，请关注微信公众号【乘龙考研】
关注公众号【乘龙考研】
一手更新 稳定有保障

即满足要求。

11．C

先序序列是$\cdots a\cdots b\cdots$，因此 a 和 b 结点的 3 种情况如下图(a)～(c)所示。中序序列是$\cdots b\cdots a\cdots$，因此 a 和 b 结点的 3 种情况如下图(d)～(f)所示，相同部分是 b 在 a 的左子树中。



(a)　(b)　(c)　(d)　(e)　(f)

12．B

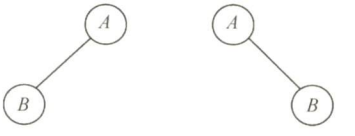
解法 1：由题可知，1 为根结点，2 为 1 的孩子。对于选项 A，3 应为 1 的左孩子，前序序列应为 13…，不符。类似的选项 C 也是错误的。对于选项 B，2 为 1 的右孩子，3 为 2 的右孩子……满足题意。对于选项 D，463572 应为 1 的右子树，2 为 1 的右孩子，46357 为 2 的左子树，3 为 2 的左孩子，46 为 3 的左子树，57 为 3 的右子树，前序序列 4、6 应相连，5、7 应相连，与题目不符。

解法 2：前序遍历时需要借助栈。二叉树的前序序列和中序序列的关系相当于以前序序列作为入栈次序，以中序序列作为出栈次序。题中以 1234567 入栈；选项 A，第一个出栈的是 3，故 1 不可能在 2 之前出栈，错误。选项 C，1 不可能在 3 之前出栈，错误。选项 D，6 第三个出栈，此时栈顶元素是 5，不是 3，错误。故选择选项 B。

解法 3：因前序序列和中序序列可以确定一棵二叉树，所以可试着用题目中的序列构造出相应的二叉树，即可得知，只有选项 B 的序列可以构造出二叉树。

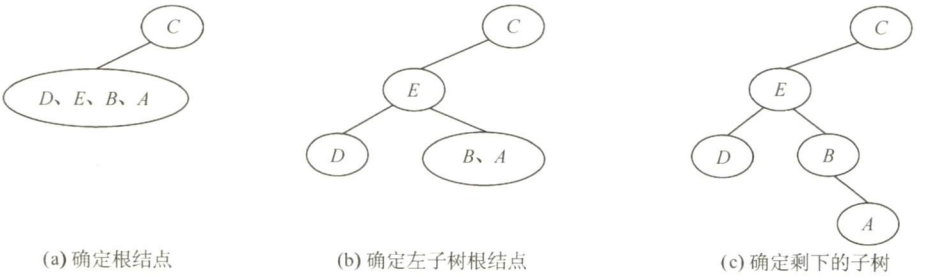
13．D

先序序列为 NLR，后序序列为 LRN，虽然可以唯一确定树的根结点，但无法划分左、右子树。例如，先序序列为 AB，后序序列为 BA，则其对应的二叉树如右图所示。



14．D

根据后序序列与中序序列可构造出二叉树，如下图所示。由图可知先序序列为 $CEDBA$。



(a) 确定根结点　(b) 确定左子树根结点　(c) 确定剩下的子树

15．A

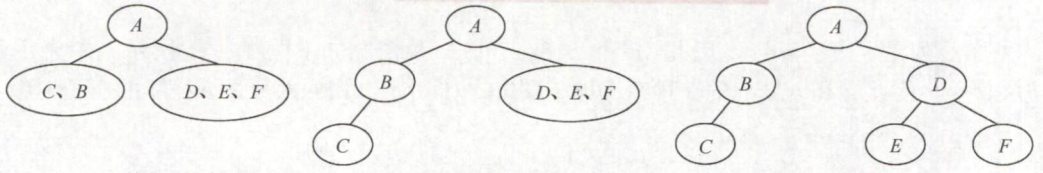
对于这种遍历序列问题，先根据遍历的性质排除若干项，若还无法确定答案，则再根据遍历结果得到二叉树，找到对应遍历序列。例如，在本题中，已知先序和中序遍历结果，可知本树的根结点为 A，左子树有 C 和 B，其余为右子树，则后序遍历结果中，A 一定在最后，并且 C 和 B 一定在前面，排除答案 B 和 D。又因先序中有 DEF，中序中有 EDF，则 D 为这个子树的根，所以 D 在后序中排在 EF 之后，故答案为选项 A。

根据二叉树的递归定义，要确定二叉树，就要分别找到根结点和左、右子树。因此，根据遍历结果，必定要确定根结点位置和如何划分左、右子树，才可以确定最终的二叉树。故仅有先序和后序遍历不能唯一确定一棵二叉树，而二者之一加上中序遍历都可以唯一确定一棵二叉树。如

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

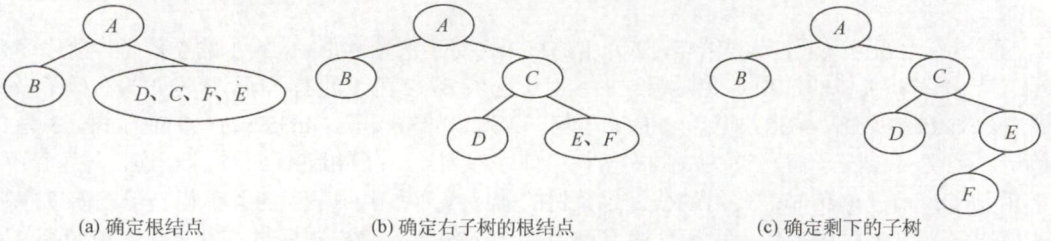
在本题中，根据先序和中序遍历的结果确定二叉树的过程如下图所示。



(a) 确定根结点　(b) 确定左子树　(c) 确定右子树

16．B

可构造出二叉树如下图所示。因此，先序序列为 *ABCDEF*。



(a) 确定根结点　(b) 确定右子树的根结点　(c) 确定剩下的子树

17．A

线索是前驱结点和后继结点的指针，引入线索的目的是加快对二叉树的遍历。

18．C

二叉树是一种逻辑结构，但线索二叉树是加上线索后的链表结构，即它是二叉树在计算机内部的一种存储结构，所以是一种物理结构。

19．C

n 个结点共有链域指针 $2n$ 个，其中，除根结点外，每个结点都被一个指针指向。剩余的链域建立线索，共 $2n-(n-1)=n+1$ 个线索。

20．C

线索二叉树中用 `ltag/rtag` 标识结点的左/右指针域是否为线索，其值为 1 时，对应指针域为线索，其值为 0 时，对应指针域为左/右孩子。

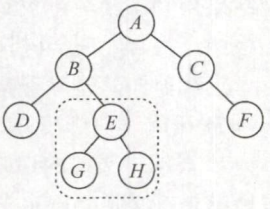
21．D

对左子树为空的二叉树进行先序线索化，根结点的左子树为空并且也没有前驱结点（先遍历根结点），先序遍历的最后一个元素为叶结点，左、右子树均为空且有前驱无后继结点，故线索化后，树中空链域有 2 个。

22．D

不是每个结点通过线索都可以直接找到它的前驱和后继。在先序线索二叉树中查找一个结点的先序后继很简单，而查找先序前驱必须知道该结点的双亲结点。同样，在后序线索二叉树中查找一个结点的后序前驱也很简单，而查找后序后继也必须知道该结点的双亲结点，二叉链表中没有存放双亲的指针。

23．D



后序线索二叉树不能有效解决求后序后继的问题。如右图所示，结点 E 的右指针指向右孩子，而在后序序列中 E 的后继结点为 B，在查找 E 的后继时后序线索不能起到任何作用，只能按常规方法来查找。

24．C

在二叉中序线索树中，某结点若有左孩子，则按照中序“左根右”的顺序，该结点的前驱结

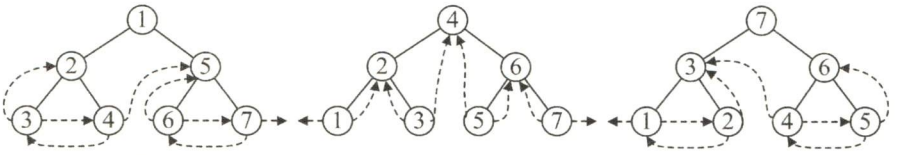
有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

点为左子树中最右的一个结点（注意，并不一定是最右叶结点）。

25．C

后序线索树遍历时，最后访问根结点，若从右孩子 x 返回访问父结点，则由于结点 x 的右孩子不一定为空（右指针无法指向其后继），因此通过指针可能无法遍历整棵树。如下图所示，结点中的数字表示遍历的顺序，图(c)中结点 6 的右指针指向其右孩子 5，而不指向其后序后继结点 7，因此后序遍历还需要栈的支持，而图(a)和图(b)均可遍历。



(a) 先序线索树　　(b) 中序线索树　　(c) 后序线索树

26．B

非空二叉树的先序序列和后序序列相反，即“根左右”与“左右根”顺序相反，因此树只有根结点，或根结点只有左子树或右子树，以此类推，其子树具有同样的性质，任意结点只有一个孩子，才能满足先序序列和后序序列正好相反。树形应为一个长链，因此选择选项 B。

27．D

分析遍历后的结点序列，可以看出根结点是在中间被访问的，而且右子树结点在左子树之前，则遍历的方法是 RNL。本题考查的遍历方法并不是二叉树遍历的 3 种基本遍历方法，对于考生而言，重要的是掌握遍历的思想。

28．D

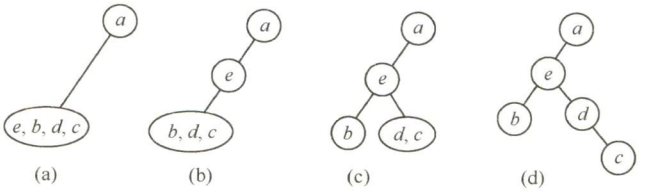
题中所给二叉树的后序序列为 $dbca$。结点 d 无前驱和左子树，左链域空，无右子树，右链域指向其后继结点 b；结点 b 无左子树，左链域指向其前驱结点 d；结点 c 无左子树，左链域指向其前驱结点 b，无右子树，右链域指向其后继结点 a。正确选项为 D。

29．C

前序序列为 NLR，后序序列为 LRN，由于前序序列和后序序列刚好相反，故不可能存在一个结点同时有左右孩子，即二叉树的高度为 4。1 为根结点，由于根结点只能有左孩子（或右孩子），因此在中序序列中，1 或在序列首或在序列尾，选项 A, B, C, D 皆满足要求。仅考虑以 1 的孩子结点 2 为根结点的子树，它也只能有左孩子（或右孩子），因此在中序序列中，2 或在序列首或在序列尾，选项 A, B, D 皆满足要求，故选选项 C。

30．A

前序序列和后序序列不能唯一确定一棵二叉树，但可以确定二叉树中结点的祖先关系：当两个结点的前序序列为 XY、后序序列为 YX 时，则 X 为 Y 的祖先。考虑前序序列 a, e, b, d, c、后序序列 b, c, d, e, a，可知 a 为根结点，e 为 a 的孩子结点；此外，由 a 的孩子结点的前序序列 e, b, d, c 和后序序列 b, c, d, e，可知 e 是 bcd 的祖先，故根结点的孩子结点只有 e。故选选项 A。



(a)　　(b)　　(c)　　(d)

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

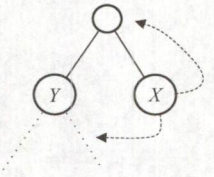
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

排除法：显然 a 为根结点，且确定 e 为 a 的孩子结点，排除选项 D。各种遍历算法中左右子树的遍历次序是固定的，若 b 也为 a 的孩子结点，则在前序序列和后序序列中 e、b 的相对次序应是不变的，故排除选项 B，同理排除选项 C。

特殊法：前序序列和后序序列对应多棵不同的二叉树树形，我们只需画出满足该条件的任意一棵二叉树即可，任意一棵二叉树必定满足正确选项的要求。

显然选择选项 A，最终得到的二叉树满足题设中前序序列和后序序列的要求。

31． A。



根据后序线索二叉树的定义，X 结点为叶结点且有左兄弟，因此这个结点为右孩子结点，利用后序遍历的方式可知 X 结点的后序后继是其父结点，即其右线索指向的是父结点。为了更加形象，在解题的过程中可以画出如右所示的草图。

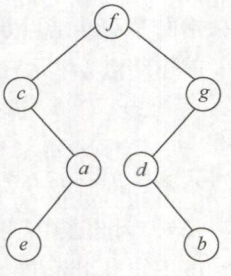
32． D

线索二叉树的线索实际上指向的是相应遍历序列特定结点的前驱结点和后继结点，所以先写出二叉树的中序遍历序列 *debxac*，中序遍历中在 x 左边和右边的字符，就是它在中序线索化的左、右线索，即 b, a，选择选项 D。

33． B

根据二叉树前序遍历和中序遍历的递归算法中递归工作栈的状态变化得出：前序序列和中序序列的关系相当于以前序序列为入栈次序，以中序序列为出栈次序。因为前序序列和中序序列可以唯一地确定一棵二叉树，所以题意相当于“以序列 a, b, c, d 为入栈次序，则出栈序列的个数为？”，对于 n 个不同元素进栈，出栈序列的个数为 $\frac{1}{n+1}C_{2n}^{n}=14$。

34． B



后序序列是先左子树，接着右子树，最后父结点，递归进行。根结点左子树的叶结点首先被访问，它是 e。接下来是它的父结点 a，然后是 a 的父结点 c。接着访问根结点的右子树。它的叶结点 b 首先被访问，然后是 b 的父结点 d，再后是 d 的父结点 g，最后是根结点 f，如右图所示。因此 d 与 a 同层，选项 B 正确。

35． B

先序序列先父结点，接着左子树，然后右子树。中序序列先左子树，接着父结点，然后右子树，递归进行。若所有非叶结点只有右子树，则先序序列和中序序列都是先父结点，然后右子树，递归进行，因此选项 B 正确。

36． B

对于此类题，每种情况只需举出一个反例即可。如图 1 所示，q 是 p 的双亲，中序遍历序列为$\{p, q\}$，选项 I 可能。如图 2 所示，q 是 p 的右孩子，中序遍历序列为$\{p, q\}$，选项 II 可能。如图 4 所示，q 是 p 的双亲的双亲，中序遍历序列为$\{x, p, q\}$，选项 IV 可能。如图 3 所示，q 是 p 的右兄弟，F 是 q 和 p 的父结点，中序遍历要求先遍历左子树，再访问根结点，最后遍历右子树，因此一定先访问 p，再访问 F，最后访问 q，p 和 q 不可能相邻出现，选项 III 不可能。

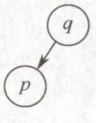


图 1

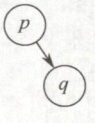


图 2

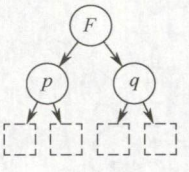


图 3

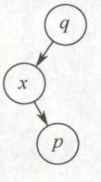


图 4

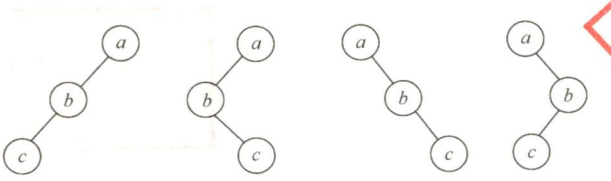
有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

二、综合应用题

01.【解答】

二叉树的先序序列是 NLR，后序序列是 LRN。要使 NLR = NRL（后序序列反序）成立，L 或 R 应为空，这样的二叉树每层只有一个结点，即二叉树的形态是其高度等于结点个数。以 3 个结点 a, b, c 为例，其形态如下图所示。



关注公众号【乘龙考研】
一手更新 稳定有保障

02.【解答】

二叉树的先序序列是 NLR，后序序列是 LRN。要使 NLR = LRN 成立，L 和 R 应均为空，所以满足条件的二叉树只有一个根结点。

03.【解答】

算法思想：后序非递归遍历二叉树是先访问左子树，再访问右子树，最后访问根结点。结合图 5.7 来分析：①沿着根的左孩子，依次入栈，直到左孩子为空。此时栈内元素依次为 $A\ B\ D$。②读栈顶元素：若其右孩子不空且未被访问过，将右子树转执行①；否则，栈顶元素出栈并访问。栈顶 D 的右孩子为空，出栈并访问，它是后序序列的第一个结点；栈顶 B 的右孩子不空且未被访问过，E 入栈，栈顶 E 的左右孩子均为空，出栈并访问；栈顶 B 的右孩子不空但已被访问，B 出栈并访问；栈顶 A 的右孩子不空且未被访问过，C 入栈，栈顶 C 的左右孩子均为空，出栈并访问；栈顶 A 的右孩子不空但已被访问，A 出栈并访问。由此得到后序序列 $D\ E\ B\ C\ A$。

在上述思想的第②步中，必须分清返回时是从左子树返回的还是从右子树返回的，因此设定一个辅助指针 r，用于指向最近访问过的结点。也可在结点中增加一个标志域，记录是否已被访问。

```
void PostOrder(BiTree T){
    InitStack(S);
    BiTNode *p=T;
    BiTNode *r=NULL;
    while(p||!IsEmpty(S)){
        if(p){                          //走到最左边
            push(S,p);
            p=p->lchild;
        }
        else{                           //向右
            GetTop(S,p);                //读栈顶结点（非出栈）
            if(p->rchild&&p->rchild!=r)//若右子树存在，且未被访问过
                p=p->rchild;            //转向右
            else{                       //否则，弹出结点并访问
                pop(S,p);               //将结点弹出
                visit(p->data);         //访问该结点
                r=p;                    //记录最近访问过的结点
                p=NULL;                 //结点访问完后，重置 p 指针
            }
        }//else
    }//while
}
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

注意：每次出栈访问完一个结点就相当于遍历完以该结点为根的子树，需将 p 置 NULL。

04.【解答】

一般的二叉树层次遍历是自上而下、从左到右，这里的遍历顺序恰好相反。算法思想：利用原有的层次遍历算法，出队的同时将各结点指针入栈，在所有结点入栈后再从栈顶开始依次访问即为所求的算法。具体实现如下：

1）把根结点入队列。

2）把一个元素出队列，遍历这个元素。

3）依次把这个元素的左孩子、右孩子入队列。

4）若队列不空，则跳到 2），否则结束。

关注公众号【乘龙考研】
一手更新 稳定有保障

算法实现如下：

```
void InvertLevel(BiTree bt){
    Stack s; Queue Q;
    if(bt!=NULL){
        InitStack(s);                    //栈初始化，栈中存放二叉树结点的指针
        InitQueue(Q);                    //队列初始化，队列中存放二叉树结点的指针
        EnQueue(Q,bt);
        while(IsEmpty(Q)==false){        //从上而下层次遍历
            DeQueue(Q,p);
            Push(s,p);                   //出队，入栈
            if(p->lchild)
                EnQueue(Q,p->lchild); //若左子女不空，则入队列
            if(p->rchild)
                EnQueue(Q,p->rchild); //若右子女不空，则入队列
        }
        while(IsEmpty(s)==false){
            Pop(s,p);
            visit(p->data);
        }                                //自下而上、从右到左的层次遍历
    }//if 结束
}
```

05.【解答】

采用层次遍历的算法，设置变量 level 记录当前结点所在的层数，设置变量 last 指向当前层的最右结点，每次层次遍历出队时与 last 指针比较，若两者相等，则层数加 1，并让 last 指向下一层的最右结点，直到遍历完成。level 的值即为二叉树的高度。

算法实现如下：

```
int Btdepth(BiTree T){
//采用层次遍历的非递归方法求解二叉树的高度
    if(!T)
        return  0;                       //树空，高度为 0
    int front=-1,rear=-1;
    int last=0,level=0;                  //last 指向当前层的最右结点
    BiTree Q[MaxSize];                   //设置队列 Q，元素是二叉树结点指针且容量足够
    Q[++rear]=T;                         //将根结点入队
    BiTree p;
    while(front<rear){                   //队不空，则循环
            p=Q[++front];                //队列元素出队，即正在访问的结点
        if(p->lchild)
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
            Q[++rear]=p->lchild;      //左孩子入队
        if(p->rchild)
            Q[++rear]=p->rchild;      //右孩子入队
        if(front==last){              //处理该层的最右结点
                level++;              //层数增 1
            last=rear;                //last 指向下层
        }
    }
    return  level;
}
```

求某层的结点个数、每层的结点个数、树的最大宽度等，都可采用与此题类似的思想。当然，此题可编写为递归算法，其实现如下：

```
int Btdepth2(BiTree T){
    if(T==NULL)
        return  0;                    //空树，高度为 0
    ldep=Btdepth2(T->lchild);         //左子树高度
    rdep=Btdepth2(T->rchild);         //右子树高度
    if(ldep>rdep)
        return ldep+1;                //树的高度为子树最大高度加根结点
    else
        return rdep+1;
}
```

关注公众号【乘龙考研】一手更新 稳定有保障

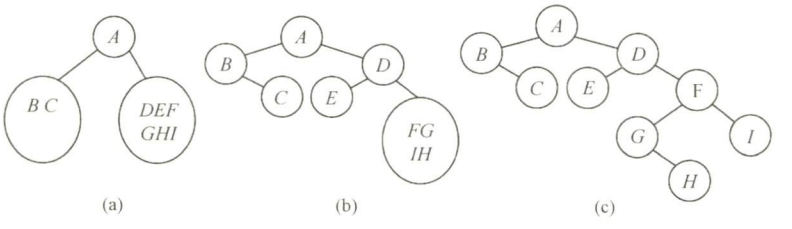
06.【解答】

由先序序列和中序序列可以唯一确定一棵二叉树。算法的实现步骤如下。

1）根据先序序列确定树的根结点。

2）根据根结点在中序序列中划分出二叉树的左、右子树包含哪些结点，然后根据左、右子树结点在先序序列中的次序确定子树的根结点，即回到步骤 1）。

如此重复上述步骤，直到每棵子树仅有一个结点（该子树的根结点）为止，如下图所示。



算法实现如下：

```
BiTree PreInCreat(ElemType A[],ElemType B[],int l1,int h1,int l2,int h2){
//l1,h1 为先序的第一和最后一个结点下标，l2,h2 为中序的第一和最后一个结点下标
//初始调用时，l1=l2=1，h1=h2=n
    root=(BiTNode*)malloc(sizeof(BiTNode));     //建根结点
    root->data=A[l1];                           //根结点
    for(i=l2;B[i]!=root->data;i++);             //根结点在中序序列中的划分
    llen=i-l2;                                  //左子树长度
    rlen=h2-i;                                  //右子树长度
    if(llen)                                    //递归建立左子树
        root->lchild=PreInCreat(A,B,l1+1,l1+llen,l2,l2+llen-1);
    else                                        //左子树为空
        root->lchild=NULL;
    if(rlen)                                    //递归建立右子树
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        root->rchild=PreInCreat(A,B,h1-rlen+1,h1,h2-rlen+1,h2);
    else                                        //右子树为空
        root->rchild=NULL;
    return  root;                               //返回根结点指针
}
```

07.【解答】

根据完全二叉树的定义，具有 n 个结点的完全二叉树与满二叉树中编号从 1～n 的结点一一对应。算法思想：采用层次遍历算法，将所有结点加入队列（包括空结点）。遇到空结点时，查看其后是否有非空结点。若有，则二叉树不是完全二叉树。

算法实现如下：

关注公众号【乘龙考研】
一手更新 稳定有保障

```
bool IsComplete(BiTree T){
//本算法判断给定二叉树是否为完全二叉树
    InitQueue(Q);
    if(!T)
        return 1;                   //空树为满二叉树
    EnQueue(Q,T);
    while(!IsEmpty(Q)){
        DeQueue(Q,p);
        if(p){                      //结点非空，将其左、右子树入队列
            EnQueue(Q,p->lchild);
            EnQueue(Q,p->rchild);
        }
        else                        //结点为空，检查其后是否有非空结点
            while(!IsEmpty(Q)){
                DeQueue(Q,p);
                if(p)               //结点非空，则二叉树为非完全二叉树
                    return  0;
            }
    }
    return 1;
}
```

08.【解答】

计算一棵二叉树 b 中所有双分支结点个数的递归模型 f(b) 如下：

f(b)=0　　若 b=NULL

f(b)=f(b->lchild)+f(b->rchild)+1　　若*b 为双分支结点

f(b)=f(b->lchild)+f(b->rchild)　　其他情况（*b 为单分支结点或叶结点）

具体算法实现如下：

```
int DsonNodes(BiTree b){
    if(b==NULL)
        return 0;
    else if(b->lchild!=NULL&&b->rchild!=NULL)  //双分支结点
        return  DSonNodes(b->lchild)+DsonNodes(b->rchild)+1;
    else
        return  DSonNodes(b->lchild)+DsonNodes(b->rchild);
}
```

当然，本题也可以设置一个全局变量 Num，每遍历到一个结点时，判断每个结点是否为分支结点（左、右结点都不为空，注意是双分支），若是则 Num++。

09.【解答】

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

采用递归算法实现交换二叉树的左、右子树，首先交换 b 结点的左孩子的左、右子树，然后交换 b 结点的右孩子的左、右子树，最后交换 b 结点的左、右孩子，当结点为空时递归结束（后序遍历的思想）。算法实现如下：

```
void swap(BiTree b){
//本算法递归地交换二叉树的左、右子树
    if(b){
        swap(b->lchild);          //递归地交换左子树
        swap(b->rchild);          //递归地交换右子树
        temp=b->lchild;           //交换左、右孩子结点
        b->lchild=b->rchild;
        b->rchild=temp;
    }
}
```

关注公众号【乘龙考研】一手更新 稳定有保障

10.【解答】

设置一个全局变量 i（初值为 1）来表示进行先序遍历时，当前访问的是第几个结点。然后可以借用先序遍历的代码模型，先序遍历二叉树。当二叉树 b 为空时，返回特殊字符'#'；当 k==i 时，该结点即为要找的结点，返回 b->data；当 k≠i 时，递归地在左子树中查找，若找到则返回该值，否则继续递归地在右子树中查找，并返回其结果。对应的递归模型如下：

```
f(b,k)='#'                                                  当 b=NULL 时
f(b,k)=b->data                                              当 i=k 时
f(b,k)=((ch=f(b->lchild,k))=='#'?f(b->rchild,k):ch)         其他情况
```

算法的实现如下：

```
int i=1;                            //遍历序号的全局变量
ElemType PreNode(BiTree b,int k){
//本算法查找二叉树先序遍历序列中第 k 个结点的值
    if(b==NULL)                     //空结点，则返回特殊字符
        return '#';
    if(i==k)                        //相等，则当前结点即为第 k 个结点
        return  b->data;
    i++;                            //下一个结点
    ch=PreNode(b->lchild,k);        //左子树中递归寻找
    if(ch!='#')                     //在左子树中，则返回该值
        return ch;
    ch=PreNode(b->rchild,k);        //在右子树中递归寻找
        return ch;
}
```

本题实质上就是一个遍历算法的实现，只不过用一个全局变量来记录访问的序号，求其他遍历序列的第 k 个结点也采用相似的方法。二叉树的遍历算法可以引申出大量的算法题，因此考生务必要熟练掌握二叉树的遍历算法。

11.【解答】

删除以元素值 x 为根的子树，只要能删除其左、右子树，就可以释放值为 x 的根结点，因此宜采用后序遍历。算法思想：删除值为 x 的结点，意味着应将其父结点的左（右）子女指针置空，用层次遍历易于找到某结点的父结点。本题要求删除树中每个元素值为 x 的结点的子树，因此要遍历完整棵二叉树。算法实现如下：

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
void DeleteXTree(BiTree &bt){          //删除以 bt 为根的子树
    if(bt){
        DeleteXTree(bt->lchild);
        DeleteXTree(bt->rchild);       //删除 bt 的左子树、右子树
        free(bt);                      //释放被删结点所占的存储空间
    }
}
//在二叉树上查找所有以 x 为元素值的结点，并删除以其为根的子树
void Search(BiTree bt,ElemType x){
    BiTree Q[];                        //Q 是存放二叉树结点指针的队列，容量足够大
    if(bt){
        if(bt->data==x){               //若根结点值为 x，则删除整棵树
            DeleteXTree(bt);
            exit(0);
        }
        InitQueue(Q);
        EnQueue(Q,bt);
        while(!IsEmpty(Q)){
            DeQueue(Q,p);
            if(p->lchild)                      //若左子女非空
                if(p->lchild->data==x){        //左子树符合则删除左子树
                    DeleteXTree(p->lchild);
                    p->lchild=NULL;
                }                              //父结点的左子女置空
                else
                    EnQueue(Q,p->lchild);      //左子树入队列
            if(p->rchild)                      //若右子女非空
                if(p->rchild->data==x){        //右子女符合则删除右子树
                    DeleteXTree(p->rchild);
                    p->rchild=NULL;
                }                              //父结点的右子女置空
                else
                    EnQueue(Q,p->rchild);      //右子女入队列
        }
    }
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

12.【解答】

算法思想：采用非递归后序遍历，最后访问根结点，访问到值为 x 的结点时，栈中所有元素均为该结点的祖先，依次出栈打印即可。算法实现如下：

```
typedef struct{
    BiTree  t;
    int     tag;
}stack;  //tag=0 表示左子女被访问，tag=1 表示右子女被访问
void  Search(BiTree bt,ElemType x){
//在二叉树 bt 中，查找值为 x 的结点，并打印其所有祖先
    stack s[];                                  //栈容量足够大
    top=0;
    while(bt!=NULL||top>0){
        while(bt!=NULL&&bt->data!=x){           //结点入栈
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
            s[++top].t=bt;
            s[top].tag=0;
            bt=bt->lchild;                          //沿左分支向下
        }
        if(bt!=NULL&&bt->data==x){
            printf("所查结点的所有祖先结点的值为:\n");  //找到 x
            for(i=1;i<=top;i++)
            printf("%d",s[i].t->data);              //输出祖先值后结束
            exit(1);
        }
        while(top!=0&&s[top].tag==1)
            top--;                                  //退栈（空遍历）
        if(top!=0){
            s[top].tag=1;
            bt=s[top].t->rchild;                    //沿右分支向下遍历
        }
    } //while(bt!=NULL||top>0)
}
```

因为查找的过程就是后序遍历的过程，所以使用的栈的深度不超过树的深度。

13.【解答】

后序遍历最后访问根结点，即在递归算法中，根是压在栈底的。本题要找 p 和 q 的最近公共祖先结点 r，不失一般性，设 p 在 q 的左边。算法思想：采用后序非递归算法，栈中存放二叉树结点的指针，当访问到某结点时，栈中所有元素均为该结点的祖先。后序遍历必然先遍历到结点 p，栈中元素均为 p 的祖先。先将栈复制到另一辅助栈中。继续遍历到结点 q 时，将栈中元素从栈顶开始逐个到辅助栈中去匹配，第一个匹配（即相等）的元素就是结点 p 和 q 的最近公共祖先。算法实现如下：

```
typedef struct{
    BiTree t;
    int tag; //tag=0 表示左子女已被访问，tag=1 表示右子女已被访问
}stack;
stack s[],s1[];                                     //栈，容量足够大
BiTree Ancestor(BiTree ROOT,BiTNode *p,BiTNode *q){
//本算法求二叉树中 p 和 q 指向结点的最近公共结点
    top=0;bt=ROOT;
    while(bt!=NULL||top>0){
        while(bt!=NULL){
            s[++top].t=bt;
            s[top].tag=0;
            bt=bt->lchild;
            }                                       //沿左分支向下
        while(top!=0&&s[top].tag==1){
        //假定 p 在 q 的左侧，遇到 p 时，栈中元素均为 p 的祖先
            if(s[top].t==p){
                for(i=1;i<=top;i++)
                    s1[i]=s[i];
                    top1=top;
            }                                       //将栈 s 的元素转入辅助栈 s1 保存
            if(s[top].t==q)                         //找到 q 结点
                for(i=top;i>0;i--){//将栈中元素的树结点到 s1 中去匹配
```

关注公众号【乘龙考研】一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
                    for(j=top1;j>0;j--)
                        if(s1[j].t==s[i].t)
                            return  s[i].t; //p 和 q 的最近公共祖先已找到
                }
                top--;                          //退栈
            } //while
            if(top!=0){
                s[top].tag=1;
                bt=s[top].t->rchild;
            }                                   //沿右分支向下遍历
        } //while
        return  NULL;                           //p 和 q 无公共祖先
    }
```

14.【解答】

采用层次遍历的方法求出所有结点的层次，并将所有结点和对应的层次放在一个队列中。然后通过扫描队列求出各层的结点总数，最大的层结点总数即为二叉树的宽度。算法实现如下：

```
typedef struct{
    BiTree data[MaxSize];       //保存队列中的结点指针
    int level[MaxSize];         //保存 data 中相同下标结点的层次
    int front,rear;
}Qu;
int BTWidth(BiTree b){
    BiTree p;
    int k,max,i,n;
    Qu.front=Qu.rear=-1;        //队列为空
    Qu.rear++;
    Qu.data[Qu.rear]=b;         //根结点指针入队
    Qu.level[Qu.rear]=1;        //根结点层次为 1
    while(Qu.front<Qu.rear){
        Qu.front++;             //出队
        p=Qu.data[Qu.front];    //出队结点
        k=Qu.level[Qu.front];   //出队结点的层次
        if(p->lchild!=NULL){    //左孩子进队列
            Qu.rear++;
            Qu.data[Qu.rear]=p->lchild;
            Qu.level[Qu.rear]=k+1;
        }
        if(p->rchild!=NULL){    //右孩子进队列
            Qu.rear++;
            Qu.data[Qu.rear]=p->rchild;
            Qu.level[Qu.rear]=k+1;
        }
    }//while
    max=0;i=0;                  //max 保存同一层最多的结点个数
    k=1;                        //k 表示从第一层开始查找
    while(i<=Qu.rear){          //i 扫描队中所有元素
        n=0;                    //n 统计第 k 层的结点个数
        while(i<=Qu.rear&&Qu.level[i]==k){
            n++;
            i++;
        }
```

关注公众号【乘龙考研】一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
            k=Qu.level[i];
            if(n>max)    max=n;        //保存最大的 n
        }
        return max;
    }
```

注意：本题队列中的结点，在出队后仍需要保留在队列中，以便求二叉树的宽度，所以设置的队列采用非环形队列，否则在出队后可能被其他结点覆盖，无法再求二叉树的宽度。

15.【解答】

对一般二叉树，仅根据先序或后序序列，不能确定另一个遍历序列。但对满二叉树，任意一个结点的左、右子树均含有相等的结点数，同时，先序序列的第一个结点作为后序序列的最后一个结点，由此得到将先序序列 pre[l1..h1]转换为后序序列 post[l2..h2]的递归模型如下：

f(pre,l1,h1,post,l2,h2)=不做任何事情　　h1<l1 时

f(pre,l1,h1,post,l2, h2)= post[h2]=pre[l1]　　其他情况

取中间位置 half=(h1-l1)/2；

将 pre[l1+1,l1+half]左子树转换为 post[l2,l2+half-1]，

即 f(pre,l1+1,l1+half,post,l2,l2+half-1)；

将 pre[l1+half+1,h1]右子树转换为 post[l2+half,h2-1]，

即 f(pre,l1+half+1,h1,post,l2+half,h2-1)。

其中，post[h2]=pre[l1]表示后序序列的最后一个结点（根结点）等于先序序列的第一个结点（根结点）。相应的算法实现如下：

```
void PreToPost(ElemType pre[],int l1,int h1,ElemType post[],int l2,int h2){
    int half;
    if(h1>=l1){
        post[h2]=pre[l1];
        half=(h1-l1)/2;
        PreToPost(pre,l1+1,l1+half,post,l2,l2+half-1);      //转换左子树
        PreToPost(pre,l1+half+1,h1,post,l2+half,h2-1);      //转换右子树
    }
}
```

例如，有以下代码：

```
ElemType *pre="ABCDEFG";
ElemType post[MaxSize];
PreToPost(pre,0,6,post,0,6);
printf("后序序列: ");
for(int i=0;i<=6;i++)
    printf("%c",post[i]);
printf("\n");
```

关注公众号【乘龙考研】
一手更新 稳定有保障

执行结果如下：

```
后序序列: C D B F G E A
```

16.【解答】

通常使用的先序、中序和后序遍历对于叶结点的访问顺序都是从左到右，这里选择中序递归遍历。算法思想：设置前驱结点指针 pre，初始为空。第一个叶结点由指针 head 指向，遍历到叶结点时，就将它前驱的 rchild 指针指向它，最后一个叶结点的 rchild 为空。算法实现如下：

```
LinkedList head, pre=NULL;                    //全局变量
LinkedList InOrder(BiTree bt){
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        if(bt){
            InOrder(bt->lchild);                        //中序遍历左子树
            if(bt->lchild==NULL&&bt->rchild==NULL)      //叶结点
                if(pre==NULL){
                    head=bt;
                    pre=bt;
                }                                       //处理第一个叶结点
                else{
                    pre->rchild=bt;
                    pre=bt;
                }                                       //将叶结点链入链表
            InOrder(bt->rchild);                        //中序遍历右子树
            pre->rchild=NULL;                           //设置链表尾
        }
        return head;
    }
```

上述算法的时间复杂度为 $O(n)$，辅助变量使用 head 和 pre，栈空间复杂度为 $O(n)$。

17.【解答】

本题采用递归的思想求解，若 T_1 和 T_2 都是空树，则相似；若有一个为空另一个不空，则必然不相似；否则递归地比较它们的左、右子树是否相似。递归函数的定义如下：

1）f(T1,T2)=1；若 T1==T2==NULL。

2）f(T1,T2)=0；若 T1 和 T2 之一为 NULL，另一个不为 NULL。

3）f(T1,T2)=f(T1->lchild,T2->lchild)&&f(T1->rchild,T2->rchild)；若 T1 和 T2 均不为 NULL。

因此，算法实现如下：

```
    int similar(BiTree T1,BiTree T2){
    //采用递归的算法判断两棵二叉树是否相似
        int leftS,rightS;
        if(T1==NULL&&T2==NULL)          //两棵树皆空
            return  1;
        else if(T1==NULL||T2==NULL)     //只有一棵树为空
            return 0;
        else{                           //递归判断
            leftS=similar(T1->lchild,T2->lchild);
            rightS=similar(T1->rchild,T2->rchild);
            return leftS&&rightS;
        }
    }
```

关注公众号【乘龙考研】一手更新 稳定有保障

18.【解答】

算法思想：在后序序列中，若结点 p 有右子女，则右子女是其前驱，若无右子女而有左子女，则左子女是其前驱。若结点 p 左、右子女均无，设其中序左线索指向某祖先结点 f（p 是 f 右子树中按中序遍历的第一个结点），若 f 有左子女，则其左子女是结点 p 在后序下的前驱；若 f 无左子女，则顺其前驱找双亲的双亲，一直找到双亲有左子女（这时左子女是 p 的前驱）。还有一种情况，若 p 是中序遍历的第一个结点，则结点 p 在中序和后序下均无前驱。

算法代码如下：

```
    BiThrTree  InPostPre(BiThrTree t,BiThrTree p){
    //在中序线索二叉树 t 中，求指定结点 p 在后序下的前驱结点 q
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
    BiThrTree q;
    if(p->rtag==0)          //若 p 有右子女，则右子女是其后序前驱
        q=p->rchild;
    else if(p->ltag==0)     //若 p 只有左子女，则左子女是其后序前驱
        q=p->lchild;
    else if(p->lchild==NULL)
        q=NULL;             //p 是中序序列第一结点，无后序前驱
    else{   //顺左线索向上找 p 的祖先，若存在，再找祖先的左子女
        while(p->ltag==1&&p->lchild!=NULL)
            p=p->lchild;
        if(p->ltag==0)
            q=p->lchild;    //p 结点的祖先的左子女是其后序前驱
        else
            q=NULL;         //仅有单支树（p 是叶子），已到根结点，p 无后序前驱
  }
    return  q;
}
```

19.【解答】

考查二叉树的带权路径长度，二叉树的带权路径长度为每个叶结点的深度与权值之积的总和，可以使用先序遍历或层次遍历解决问题（考生只需采用一种思路）。

1）算法的基本设计思想。

① 基于先序递归遍历的算法思想是用一个 static 变量记录 wpl，把每个结点的深度作为递归函数的一个参数传递，算法步骤如下：

若该结点是叶结点，则变量 wpl 加上该结点的深度与权值之积。

若该结点是非叶结点，则左子树不为空时，对左子树调用递归算法，右子树不为空，对右子树调用递归算法，深度参数均为本结点的深度参数加 1。

最后返回计算出的 wpl 即可。

② 基于层次遍历的算法思想是使用队列进行层次遍历，并记录当前的层数：

当遍历到叶结点时，累计 wpl。

当遍历到非叶结点时，把该结点的子树加入队列。

当某结点为该层的最后一个结点时，层数自增 1。

队列空时遍历结束，返回 wpl。

关注公众号【乘龙考研】
一手更新 稳定有保障

2）二叉树结点的数据类型定义如下。

```
typedef struct BiTNode{
    int weight;
    struct BiTNode *lchild,*rchild;
}BiTNode,*BiTree;
```

3）算法的代码如下。

①基于先序遍历的算法：

```
int WPL(BiTree root){
    return wpl_PreOrder(root,0);
}
int wpl_PreOrder(BiTree root,int deep){
    static int wpl=0;                    //定义变量存储 wpl
    if(root->lchild==NULL&&root->rchild==NULL)  //若为叶结点，则累积 wpl
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        wpl+=deep*root->weight;
    if(root->lchild!=NULL)              //若左子树不空，则对左子树递归遍历
        wpl_PreOrder(root->lchild,deep+1);
    if(root->rchild!=NULL)              //若右子树不空，则对右子树递归遍历
        wpl_PreOrder(root->rchild,deep+1);
    return wpl;
}
```

②基于层次遍历的算法：

```
#define MaxSize 100          //设置队列的最大容量
int wpl_LevelOrder(BiTree root){
BiTree q[MaxSize];           //声明队列，end1 为头指针，end2 为尾指针
int end1,end2;               //队列最多容纳 MaxSize-1 个元素
end1=end2=0;                 //头指针指向队头元素，尾指针指向队尾的后一个元素
int wpl=0,deep=0;            //初始化 wpl 和深度
BiTree lastNode;             //lastNode 用来记录当前层的最后一个结点
BiTree newlastNode;          //newlastNode 用来记录下一层的最后一个结点
lastNode=root;               //lastNode 初始化为根结点
newlastNode=NULL;            //newlastNode 初始化为空
q[end2++]=root;              //根结点入队
while(end1!=end2){           //层次遍历，若队列不空则循环
    BiTree t=q[end1++];      //拿出队列中的头一个元素
    if(t->lchild==NULL&&t->rchild==NULL){
        wpl+=deep*t->weight;
    }                        //若为叶结点，则统计 wpl
    if(t->lchild!=NULL){     //若非叶结点，则让左结点入队
        q[end2++]=t->lchild;
        newlastNode=t->lchild;
    }                        //并设下一层的最后一个结点为该结点的左结点
    if(t->rchild!=NULL){     //处理叶结点
        q[end2++]=t->rchild;
        newlastNode=t->rchild;
    }
    if(t==lastNode){         //若该结点为本层最后一个结点，则更新 lastNode
        lastNode=newlastNode;
        deep+=1;             //层数加 1
    }
}
return wpl;                  //返回 wpl
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

注意：当 static 关键字用于代码块内部的变量的声明时，用于修改变量的存储类型，即从自动变量修改为静态变量，但变量的链接属性和作用域不受影响。用这种方式声明的变量在程序执行之前创建，并在程序的整个执行期间一直存在，而不是每次在代码块开始执行时创建，在代码块执行完毕后销毁。也就是说，它保持局部变量内容的持久。静态局部变量的生存期虽然为整个源程序，但其作用域仍与局部变量相同，即只能在定义该变量的函数内使用该变量。退出该函数后，尽管该变量还继续存在，但不能使用它。有兴趣的读者可以查阅相关资料。

20.【解答】

1）算法的基本设计思想。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

表达式树的中序序列加上必要的括号即为等价的中缀表达式。可以基于二叉树的中序遍历策略得到所需的表达式。

表达式树中分支结点所对应的子表达式的计算次序，由该分支结点所处的位置决定。为得到正确的中缀表达式，需要在生成遍历序列的同时，在适当位置增加必要的括号。显然，表达式的最外层（对应根结点）和操作数（对应叶结点）不需要添加括号。

2）算法实现。

将二叉树的中序遍历递归算法稍加改造即可得本题的答案。除根结点和叶结点外，遍历到其他结点时在遍历其左子树之前加上左括号，遍历完右子树后加上右括号。

```
void BtreeToE(BTree *root){
    BtreeToExp(root,1);                    //根的高度为 1
}
void BtreeToExp( BTree *root,int deep)
{
    if(root==NULL) return;                 //空结点返回
    else if(root->left==NULL&&root->right==NULL)   //若为叶结点
        printf("%s", root->data);          //输出操作数，不加括号
    else{
        if(deep>1) printf("(");            //若有子表达式则加 1 层括号
        BtreeToExp(root->left, deep+1);
        printf("%s", root->data);          //输出操作符
        BtreeToExp(root->right, deep+1);
        if(deep>1) printf(")");            //若有子表达式则加 1 层括号
    }
}
```

关注公众号【乘龙考研】一手更新 稳定高质量

21.【解答 1】

1）算法的基本设计思想。

对于采用顺序存储方式保存的二叉树，根结点保存在 SqBiTNode[0]中；当某结点保存在 SqBiTNode[i]中时，若有左孩子，则其值保存在 SqBiTNode[2i+1]中；若有右孩子，则其值保存在 SqBiTNode[2i+2]中；若有双亲结点，则其值保存在 SqBiTNode[(i-1)/2]中。

二叉搜索树需要满足的条件是：任意一个结点值大于其左子树中的全部结点值，小于其右子树中的全部结点值。中序遍历二叉搜索树得到一个升序序列。

使用整型变量 val 记录中序遍历过程中已遍历结点的最大值，初值为一个负整数。若当前遍历的结点值小于或等于 val，则算法返回 false，否则，将 val 的值更新为当前结点的值。

2）算法实现。

```
#define false 0
#define true 1
typedef int bool;
bool judgeInOrderBST(SqBiTree bt,int k,int *val){//初始调用时 k 的值是 0
    if(k<bt.ElemNum&&bt.SqBiTNode[k]!=-1){
        if(!judgeInOrderBST(bt,2*k+1,val)) return false;
        if(bt.SqBiTNode[k]<=*val) return false;
        *val=bt.SqBiTNode[k];
        if(!judgeInOrderBST(bt,2*k+2,val)) return false;
    }
    return true;
}
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

【解答2】

1）算法的基本设计思想。

对于采用顺序存储方式保存的二叉树，根结点保存在 SqBiTNode[0]中；当某结点保存在 SqBiTNode[i]中时，若有左孩子，则其值保存在 SqBiTNode[2i+1]中；若有右孩子，则其值保存在 SqBiTNode[2i+2]中；若有双亲结点，则其值保存在 SqBiTNode[(i-1)/2]中。

二叉搜索树需要满足的条件是：任意一个结点值大于其左子树中的全部结点值，小于其右子树中的全部结点值。设置两个数组 pmax 和 pmin。根据二叉搜索树的定义，SqBiTNode[i]中的值应该大于以 SqBiTNode[2i+1]为根的子树中的最大值（保存在 pmax[2i+1]中），小于以 SqBiTNode[2i+2]为根的子树中的最小值（保存在 pmin[2i+1]中）。初始时，用数组 SqBiTNode 中前 ElemNum 个元素的值对数组 pmax 和 pmin 初始化。

在数组 SqBiTNode 中从后向前扫描，扫描过程中逐一验证结点与子树之间是否满足上述的大小关系。

2）算法实现。

```
#define false 0
#define true 1
typedef int bool;
bool judgeBST(SqBiTree bt){
    int k,m,*pmin,*pmax;
    pmin=(int *)malloc(sizeof(int)*(bt.ElemNum));
    pmax=(int *)malloc(sizeof(int)*(bt.ElemNum));
    for(k=0;k<bt.ElemNum;k++)    //辅助数组初始化
        pmin[k]=pmax[k]=bt.SqBiTNode[k];
    for(k=bt.ElemNum-1;k>0;k--){ //从最后一个叶结点向根遍历
        if(bt.SqBiTNode[k]!=-1){
            m=(k-1)/2;              //双亲
            if(k%2==1&&bt.SqBiTNode[m]>pmax[k])          //其为左孩子
                pmin[m]=pmin[k];
            else if(k%2==0&&bt.SqBiTNode[m]<pmin[k])     //其为右孩子
                pmax[m]=pmax[k];
            else  return false;
        }
    }
    return true;
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

5.4 树、森林

5.4.1 树的存储结构

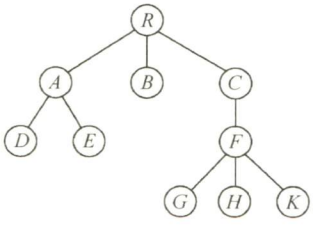
树的存储方式有多种，既可采用顺序存储结构，又可采用链式存储结构，但无论采用何种存储方式，都要求能唯一地反映树中各结点之间的逻辑关系，这里介绍3种常用的存储结构。

1. 双亲表示法

这种存储结构采用一组连续空间来存储每个结点，同时在每个结点中增设一个伪指针，指示其双亲结点在数组中的位置。如图5.14所示，根结点下标为0，其伪指针域为-1。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

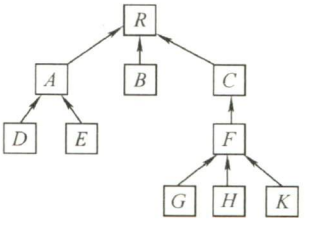
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】



(a) 一棵树

	data	parent
0	R	−1
1	A	0
2	B	0
3	C	0
4	D	1
5	E	1
6	F	3
7	G	6
8	H	6
9	K	6

(b) 双亲表示



(c) 双亲指针图示

图 5.14　树的双亲表示法

双亲表示法的存储结构描述如下：

```
#define MAX_TREE_SIZE 100            //树中最多结点数
typedef struct{                      //树的结点定义
    ElemType data;                   //数据元素
    int parent;                      //双亲位置域
}PTNode;
typedef struct{                      //树的类型定义
    PTNode nodes[MAX_TREE_SIZE];     //双亲表示
    int n;                           //结点数
}PTree;
```

关注公众号【乘龙考研】一手更新 稳定有保障

该存储结构利用了每个结点（根结点除外）只有唯一双亲的性质，可以很快地得到每个结点的双亲结点，但求结点的孩子时则需要遍历整个结构。

注意：区别树的顺序存储结构与二叉树的顺序存储结构。在树的顺序存储结构中，数组下标代表结点的编号，下标中所存的内容指示了结点之间的关系。而在二叉树的顺序存储结构中，数组下标既代表了结点的编号，又指示了二叉树中各结点之间的关系。当然，二叉树属于树，因此二叉树都可以用树的存储结构来存储，但树却不都能用二叉树的存储结构来存储。

2．孩子表示法

孩子表示法是将每个结点的孩子结点都用单链表链接起来形成一个线性结构，此时 n 个结点就有 n 个孩子链表（叶结点的孩子链表为空表），如图 5.15(a)所示。

这种存储结构寻找子女的操作非常直接，而寻找双亲的操作需要遍历 n 个结点中孩子链表指针域所指向的 n 个孩子链表。

3．孩子兄弟表示法

孩子兄弟表示法又称二叉树表示法，即以二叉链表作为树的存储结构。孩子兄弟表示法使每个结点包括三部分内容：结点值、指向结点第一个孩子结点的指针，以及指向结点下一个兄弟结点的指针（沿此域可以找到结点的所有兄弟结点），如图 5.15(b)所示。

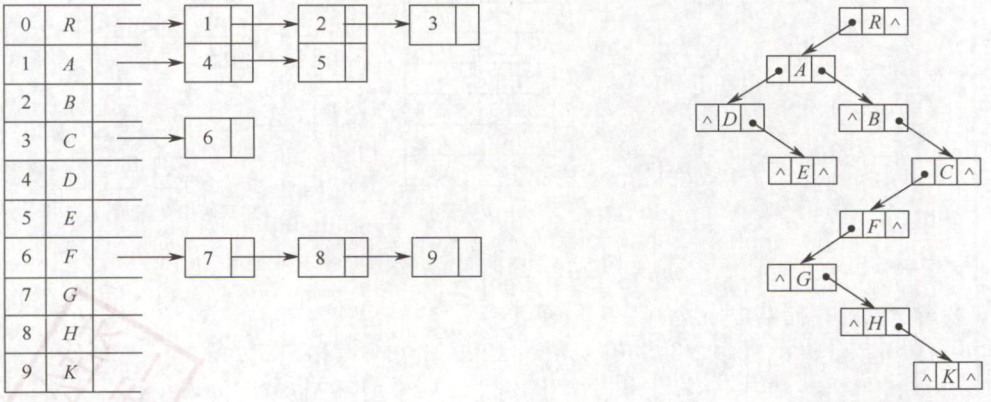
孩子兄弟表示法的存储结构描述如下：

```
typedef struct CSNode{
    ElemType data;                          //数据域
    struct CSNode *firstchild,*nextsibling;  //第一个孩子和右兄弟指针
}CSNode,*CSTree;
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

这种存储表示法比较灵活，其最大的优点是可以方便地实现树转换为二叉树的操作，易于查找结点的孩子等，但缺点是从当前结点查找其双亲结点比较麻烦。若为每个结点增设一个 parent 域指向其父结点，则查找结点的父结点也很方便。



(a) 孩子表示法　　(b) 孩子兄弟表示法

图 5.15　树的孩子表示法和孩子兄弟表示法

5.4.2　树、森林与二叉树的转换

由于二叉树和树都可以用二叉链表作为存储结构，因此以二叉链表作为媒介可以导出树与二叉树的一个对应关系，即给定一棵树，可以找到唯一的一棵二叉树与之对应。从物理结构上看，它们的二叉链表是相同的，只是解释不同而已。

树转换为二叉树的规则：每个结点左指针指向它的第一个孩子，右指针指向它在树中的相邻右兄弟，这个规则又称“左孩子右兄弟”。由于根结点没有兄弟，所以对应的二叉树没有右子树，如图 5.16 所示。

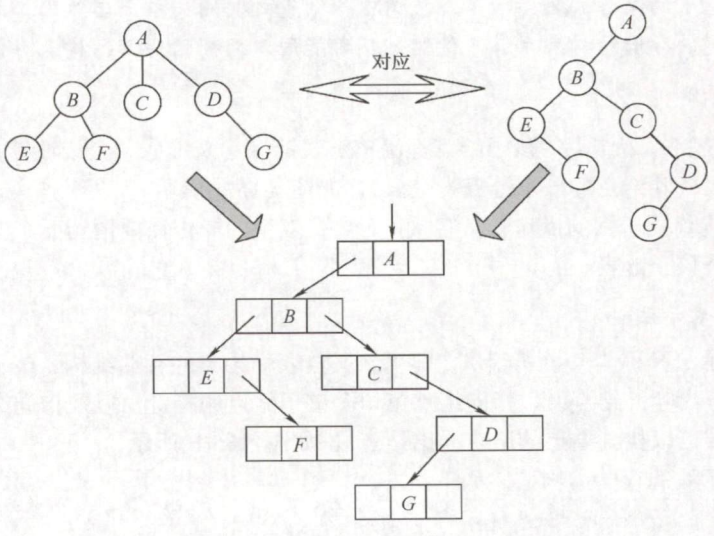


图 5.16　树与二叉树的对应关系

树转换成二叉树的画法：①在兄弟结点之间加一连线；②对每个结点，只保留它与第一个孩

关注公众号【乘龙考研】一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

子的连线，而与其他孩子的连线全部抹掉；③以树根为轴心，顺时针旋转 45°。

将森林转换为二叉树的规则与树类似。先将森林中的每棵树转换为二叉树，由于任意一棵和树对应的二叉树的右子树必空，若把森林中第二棵树根视为第一棵树根的右兄弟，即将第二棵树对应的二叉树当作第一棵二叉树根的右子树，将第三棵树对应的二叉树当作第二棵二叉树根的右子树……以此类推，就可以将森林转换为二叉树。

森林转换成二叉树的画法：①将森林中的每棵树转换成相应的二叉树；②每棵树的根也可视为兄弟关系，在每棵树的根之间加一根连线；③以第一棵树的根为轴心顺时针旋转 45°。

二叉树转换为森林的规则：若二叉树非空，则二叉树的根及其左子树为第一棵树的二叉树形式，故将根的右链断开。二叉树根的右子树又可视为一个由除第一棵树外的森林转换后的二叉树，应用同样的方法，直到最后只剩一棵没有右子树的二叉树为止，最后再将每棵二叉树依次转换成树，就得到了森林，如图 5.17 所示。二叉树转换为树或森林是唯一的。

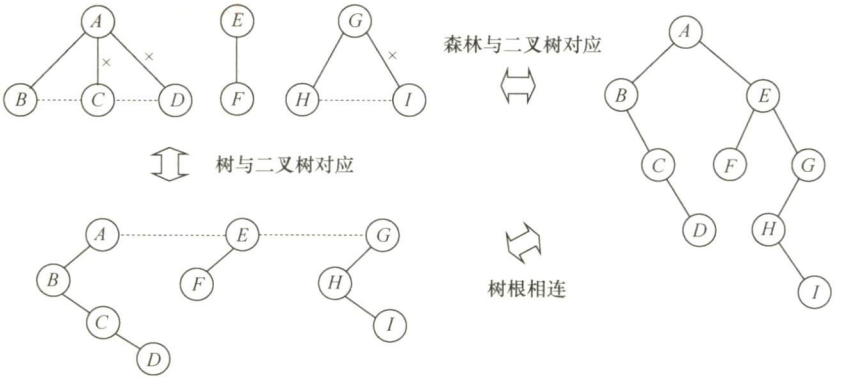


图 5.17　森林与二叉树的对应关系

5.4.3　树和森林的遍历

树的遍历是指用某种方式访问树中的每个结点，且仅访问一次。主要有两种方式：

1）*先根遍历*。若树非空，先访问根结点，再依次遍历根结点的每棵子树，遍历子树时仍遵循先根后子树的规则。其遍历序列与这棵树相应二叉树的先序序列相同。

2）*后根遍历*。若树非空，先依次遍历根结点的每棵子树，再访问根结点，遍历子树时仍遵循先子树后根的规则。其遍历序列与这棵树相应二叉树的中序序列相同。

图 5.16 的树的先根遍历序列为 *A B E F C D G*，后根遍历序列为 *E F B C G D A*。

另外，树也有层次遍历，与二叉树的层次遍历思想基本相同，即按层序依次访问各结点。

按照森林和树相互递归的定义，可得到森林的两种遍历方法。

1）*先序遍历森林*。若森林为非空，则按如下规则进行遍历：

- 访问森林中第一棵树的根结点。
- 先序遍历第一棵树中根结点的子树森林。
- 先序遍历除去第一棵树之后剩余的树构成的森林。

2）*中序遍历森林*。森林为非空时，按如下规则进行遍历：

- 中序遍历森林中第一棵树的根结点的子树森林。
- 访问第一棵树的根结点。
- 中序遍历除去第一棵树之后剩余的树构成的森林。

关注公众号【乘龙考研】一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

图 5.17 的森林的先序遍历序列为 $ABCDEFGHI$，中序遍历序列为 $BCDAFEHIG$。

当森林转换成二叉树时，其第一棵树的子树森林转换成左子树，剩余树的森林转换成右子树，可知森林的先序和中序遍历即为其对应二叉树的先序和中序遍历。

树和森林的遍历与二叉树的遍历关系见表 5.1。

表 5.1 树和森林的遍历与二叉树遍历的对应关系

树	森林	二叉树
先根遍历	先序遍历	先序遍历
后根遍历	中序遍历	中序遍历

注意：部分教材也将森林的中序遍历称为后序遍历，称中序遍历是相对其二叉树而言的，称后序遍历是因为根确实是最后才访问的，如遇到这两种称谓，那么都可以理解为同一种遍历方法。

关注公众号【乘龙考研】
一手更新 稳定有保障

5.4.4 本节试题精选

一、单项选择题

01. 下列关于树的说法中，正确的是（ ）。

I. 对于有 n 个结点的二叉树，其高度为 $\log_2 n$

II. 完全二叉树中，若一个结点没有左孩子，则它必是叶结点

III. 高度为 h（$h>0$）的完全二叉树对应的森林所含的树的个数一定是 h

IV. 一棵树中的叶子数一定等于与其对应的二叉树的叶子数

A. I 和 III　　B. IV　　C. I 和 II　　D. II

02. 利用二叉链表存储森林时，根结点的右指针是（ ）。

A. 指向最左兄弟　B. 指向最右兄弟　C. 一定为空　D. 不一定为空

03. 设森林 F 中有 3 棵树，第一、第二、第三棵树的结点个数分别为 M_1, M_2 和 M_3。与森林 F 对应的二叉树根结点的右子树上的结点个数是（ ）。

A. M_1　　B. M_1+M_2　　C. M_3　　D. M_2+M_3

04. 设森林 F 对应的二叉树为 B，它有 m 个结点，B 的根为 p，p 的右子树结点个数为 n，森林 F 中第一棵树的结点个数是（ ）。

A. $m-n$　　B. $m-n-1$　　C. $n+1$　　D. 条件不足，无法确定

05. 森林 $T=(T_1, T_2, \cdots, T_m)$ 转化为二叉树 BT 的过程为：若 $m=0$，则 BT 为空，若 $m\neq 0$，则（ ）。

A. 将中间子树 T_{mid}（$\text{mid}=(1+m)/2$）的根作为 BT 的根；将$(T_1, T_2, \cdots, T_{\text{mid}-1})$转换为 BT 的左子树；将$(T_{\text{mid}+1}, \cdots, T_m)$转换为 BT 的右子树

B. 将子树 T_1 的根作为 BT 的根；将 T_1 的子树森林转换成 BT 的左子树；将$(T_2, T_3, \cdots, T_m)$转换成 BT 的右子树

C. 将子树 T_1 的根作为 BT 的根；将 T_1 的左子树森林转换成 BT 的左子树；将 T_1 的右子树森林转换为 BT 的右子树；其他以此类推

D. 将森林 T 的根作为 BT 的根；将$(T_1, T_2, \cdots, T_m)$转化为该根下的结点，得到一棵树，然后将这棵树再转化为二叉树 BT

06. 设 F 是一个森林，B 是由 F 变换来的二叉树。若 F 中有 n 个非终端结点，则 B 中右指针域为空的结点有（ ）个。

A. $n-1$　　B. n　　C. $n+1$　　D. $n+2$

07. 若 T_1 是由有序树 T 转换而来的二叉树，则 T 中结点的后根序列就是 T_1 中结点的（ ）序列。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

A. 先序　　B. 中序　　C. 后序　　D. 层序

08. 某二叉树结点的中序序列为 *BDAECF*，后序序列为 *DBEFCA*，则该二叉树对应的森林包括（ ）棵树。

A. 1　　B. 2　　C. 3　　D. 4

09. 设 X 是树 T 中的一个非根结点，B 是 T 所对应的二叉树。在 B 中，X 是其双亲结点的右孩子，下列结论中正确的是（ ）。

A. 在树 T 中，X 是其双亲结点的第一个孩子

B. 在树 T 中，X 一定无右边兄弟

C. 在树 T 中，X 一定是叶结点

D. 在树 T 中，X 一定有左边兄弟

10. 在森林的二叉树表示中，结点 M 和结点 N 是同一父结点的左儿子和右儿子，则在该森林中（ ）。

A. M 和 N 有同一双亲　　B. M 和 N 可能无公共祖先

C. M 是 N 的儿子　　D. M 是 N 的左兄弟

11.【2009 统考真题】将森林转换为对应的二叉树，若在二叉树中，结点 u 是结点 v 的父结点的父结点，则在原来的森林中，u 和 v 可能具有的关系是（ ）。

I. 父子关系　II. 兄弟关系　III. u 的父结点与 v 的父结点是兄弟关系

A. 只有 II　　B. I 和 II　　C. I 和 III　　D. I、II 和 III

12.【2011 统考真题】已知一棵有 2011 个结点的树，其叶结点个数为 116，该树对应的二叉树中无右孩子的结点个数是（ ）。

A. 115　　B. 116　　C. 1895　　D. 1896

13.【2014 统考真题】将森林 F 转换为对应的二叉树 T，F 中叶结点的个数等于（ ）。

A. T 中叶结点的个数　　B. T 中度为 1 的结点个数

C. T 中左孩子指针为空的结点个数　　D. T 中右孩子指针为空的结点个数

14.【2016 统考真题】若森林 F 有 15 条边、25 个结点，则 F 包含树的个数是（ ）。

A. 8　　B. 9　　C. 10　　D. 11

15.【2019 统考真题】若将一棵树 T 转化为对应的二叉树 BT，则下列对 BT 的遍历中，其遍历序列与 T 的后根遍历序列相同的是（ ）。

A. 先序遍历　　B. 中序遍历

C. 后序遍历　　D. 按层遍历

16.【2020 统考真题】已知森林 F 及与之对应的二叉树 T，若 F 的先根遍历序列是 a, b, c, d, e, f，中根遍历序列是 b, a, d, f, e, c，则 T 的后根遍历序列是（ ）。

A. b, a, d, f, e, c　　B. b, d, f, e, c, a

C. b, f, e, d, c, a　　D. f, e, d, c, b, a

17.【2021 统考真题】某森林 F 对应的二叉树为 T，若 T 的先序遍历序列是 a, b, d, c, e, g, f，中序遍历序列是 b, d, a, e, g, c, f，则 F 中树的棵数是（ ）。

A. 1　　B. 2　　C. 3　　D. 4

18.【2022 统考真题】如果三叉树 T 中有 244 个结点（叶结点的高度为 1），那么 T 的高度至少是（ ）。

A. 8　　B. 7　　C. 6　　D. 5

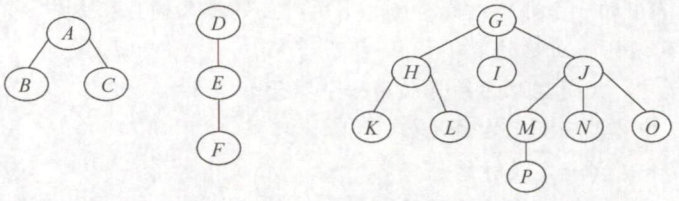
有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

二、综合应用题

01. 给定一棵树的先根遍历序列和后根遍历序列，能否唯一确定一棵树？若能，请举例说明；若不能，请给出反例。

02. 将下面一个由 3 棵树组成的森林转换为二叉树。



03. 已知某二叉树的先序序列和中序序列分别为 *ABDEHCFIMGJKL* 和 *DBHEAIM FCGKLJ*，请画出这棵二叉树，并画出二叉树对应的森林。

04. 编程求以孩子兄弟表示法存储的森林的叶结点数。

05. 以孩子兄弟链表为存储结构，请设计递归算法求树的深度。

06. 已知一棵树的层次序列及每个结点的度，编写算法构造此树的孩子-兄弟链表。

07. 【2016 统考真题】若一棵非空 k（$k\geqslant 2$）叉树 T 中的每个非叶结点都有 k 个孩子，则称 T 为正则 k 叉树。请回答下列问题并给出推导过程。

1）若 T 有 m 个非叶结点，则 T 中的叶结点有多少个？

2）若 T 的高度为 h（单结点的树 $h=1$），则 T 的结点数最多为多少个？最少为多少个？

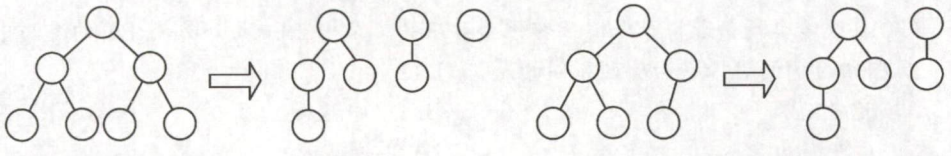
5.4.5 答案与解析

关注公众号【乘龙考研】
一手更新 稳定有保障

一、单项选择题

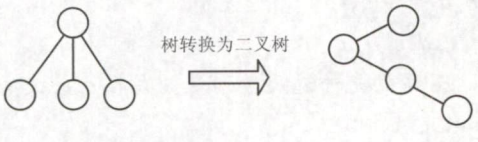
01. D

若 n 个结点的二叉树是一棵单支树，则其高度为 n。完全二叉树中最多存在一个度为 1 的结点且只有左孩子，若不存在左孩子，则一定也不存在右孩子，因此必是叶结点，选项 II 正确。只有满二叉树才具有性质 III，如下图所示。



(a) 满二叉树转化为对应的森林　　(b) 非满二叉树转化为对应的森林

在树转换为二叉树时，若有几个叶结点具有共同的双亲，则转换成二叉树后只有一个叶结点（最右边的叶结点），如左图所示，选项 IV 错误。注意，若树中的任意两个叶结点都不存在相同的双亲，则树中的叶子数才有可能与其对应的二叉树中的叶子数相等。



02. D

森林与二叉树具有对应关系，因此，我们存储森林时应先将森林转换成二叉树，转换的方法就是“左孩子右兄弟”，与树不同的是，若存在第二棵树，则二叉链表的根结点的右指针指向的是森林中的第二棵树的根结点。若此森林只有一棵树，则根结点的右指针为空。因此，右指针可

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

能为空也可能不为空。

03．D

森林与二叉树的转换规则同样是“左孩子右兄弟”，不过与普通树不同，森林中的每棵树是独立的，因此先要将每棵树的根结点全部视为兄弟结点的关系。所以，题中森林转换后，树 2 作为树 1 的根结点的右子树，树 3 作为树 2 的根结点的右子树。所以，森林 F 对应的二叉树根结点的右子树上的结点个数是 M_2+M_3。

04．A

森林转换成二叉树时采用孩子兄弟表示法，根结点及其左子树为森林中的第一棵树。右子树为其他剩余的树。所以，第一棵树的结点个数为 $m-n$。

05．B

将森林中每棵树的根结点视为兄弟结点的关系，再按照“左孩子右兄弟”的规则来进行转化。

06．C

根据森林与二叉树转换规则“左孩子右兄弟”。二叉树 B 中右指针域为空代表该结点没有兄弟结点。森林中每棵树的根结点从第二个开始依次连接到前一棵树的根的右孩子，因此最后一棵树的根结点的右指针为空。另外，每个非终端结点，其所有孩子结点在转换之后，最后一个孩子的右指针也为空，故树 B 中右指针域为空的结点有 $n+1$ 个。

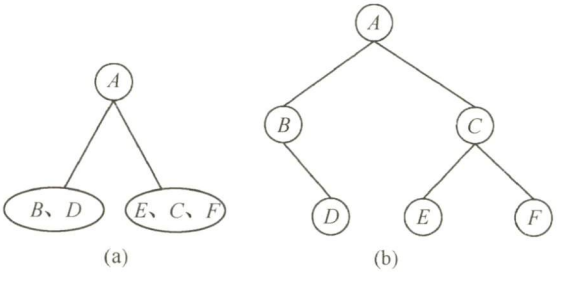
07．B

有序树 T 转换成二叉树 T_1 时，T 的后根序列是对应 T_1 的中序序列还是后序序列呢（显然树的后根序列不可能对应二叉树的先序序列和层序序列）？看右图所示的例子，在树 T 中，叶结点 B 应最先访问，在 T_1 中，B 的右兄弟 C 转换为它的右孩子，若对应 T_1 的后序序列，则 C 应在 B 的前面访问，所以 T 的后根序列不可能对应 T_1 的后序序列。

08．C

根据二叉树的前序遍历序列和中序遍历序列可以唯一确定一棵二叉树。根据后序遍历序列，A 是二叉树的根结点。根据中序遍历序列，二叉树的形态一定如下图中的(a)所示。

对于 A 的左子树，由后序遍历序列可知，因为 B 比 D 后被访问，因此 B 必为 D 的父结点，又由中序遍历序列可知，D 是 B 的右儿子。对于 A 的右子树，同理可确定结点 E、C、F 的关系。此二叉树的形态如右图中的(b)所示。



再根据二叉树与森林的对应关系，森林中树的棵数即为其对应二叉树（向右上旋转 45°后）的根结点 A 及其右兄弟数，或解释为：对应二叉树从根结点 A 开始不断往右孩子访问，所访问到的结点数。可知此森林中有 3 棵树，根结点分别为 A, C 和 F。

09．D

在二叉树 B 中，X 是其双亲的右孩子，因此在树 T 中，X 必是其双亲结点的右兄弟，换句话说，X 在树中必有左兄弟。

10．B

在森林的二叉树表示中，当 M 和 N 的父结点是二叉树根结点时，M 和 N 在不同的树上。因此 M 和 N 可能无公共祖先。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

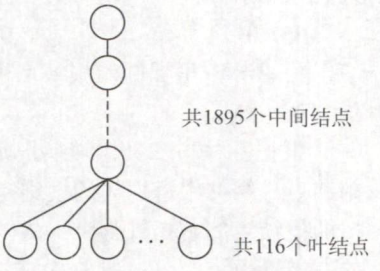
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

11．B

森林与二叉树的转换规则为“左孩子右兄弟”。在最后生成的二叉树中，父子关系在对应的森林关系中可能是兄弟关系或原本就是父子关系。情形 I：若结点 v 是结点 u 的第二个孩子结点，在转换时，结点 v 就变成结点 u 的第一个孩子的右孩子，符合要求。情形 II：结点 u 和 v 是兄弟结点的关系，但二者之中还有一个兄弟结点 k，转换后结点 v 就变为结点 k 的右孩子，而结点 k 则是结点 u 的右孩子，符合要求。情形 III：结点 v 的父结点要么是原先的父结点，要么是兄弟结点。若结点 u 的父结点与 v 的父结点是兄弟关系，则转换之后不可能出现结点 u 是结点 v 的父结点的父结点的情形。

12．D

树转换为二叉树时，树的每个分支结点的所有子结点中的最右子结点无右孩子，根结点转换后也没有右孩子，因此，对应二叉树中无右孩子的结点个数 = 分支结点数 + 1 = 2011 − 116 + 1 = 1896。



通常本题应采用特殊法求解，设题意中的树是如右图所示的结构，则对应的二叉树中仅有前 115 个叶结点有右孩子，故无右孩子的结点个数 = 2011 − 115 = 1896。

13．C

将森林转化为二叉树相当于用孩子兄弟表示法来表示森林。在变化过程中，原森林某结点的第一个孩子结点作为它的左子树，它的兄弟作为它的右子树。森林中的叶结点由于没有孩子结点，转化为二叉树时，该结点就没有左结点，所以 F 中叶结点的个数等于 T 中左孩子指针为空的结点个数，选择选项 C。此题还可通过一些特例来排除选项 A、B 和 D。

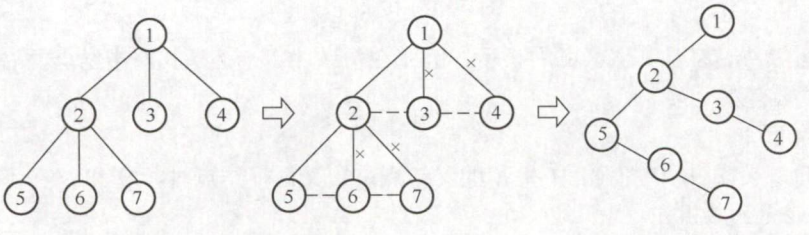
14．C

解法 1：树有一个很重要的性质，即在 n 个结点的树中有 $n-1$ 条边，“那么对于每棵树，其结点数比边数多 1”。本题森林中的结点数比边数多 10（即 25 − 15 = 10），显然共有 10 棵树。

解法 2：仔细分析后发现，此题考查的也是图的某些方面的性质：生成树和生成森林。此时对于图的生成树有一个重要的性质，即图中顶点数若为 n，则其生成树含有 $n-1$ 条边。对比解法一中树的性质，不难发现两种解法都用到了性质“树中结点数比边数多 1”，接下来的分析如解法一。

15．B

后根遍历树可分为两步：①从左到右访问双亲结点的每个孩子（转化为二叉树后就是先访问根结点再访问右子树）；②访问完所有孩子后再访问它们的双亲结点（转化为二叉树后就是先访问左子树再访问根结点），因此树 T 的后根遍历序列与其相应二叉树 BT 的中序遍历序列相同。对于此类题，采用特殊值法求解通常会更便捷，左下图树 T 转换为二叉树 BT 的过程如下图所示，树 T 的后序遍历序列显然和其相应二叉树 BT 的中序遍历序列相同，均为 5, 6, 7, 2, 3, 4, 1。

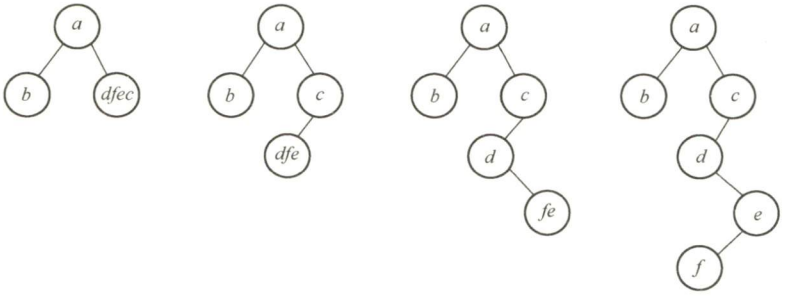


有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

16．C

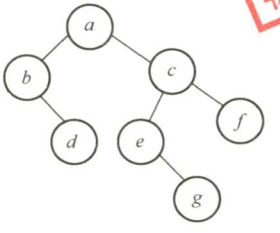
森林 F 的先根遍历序列对应于其二叉树 T 的先序遍历序列，森林 F 的中根遍历序列对应于其二叉树 T 的中序遍历序列。即 T 的先序遍历序列为 a, b, c, d, e, f，中序遍历序列为 b, a, d, f, e, c。根据二叉树 T 的先序序列和中序序列可以唯一确定它的结构，构造过程如下：



关注公众号【乘龙考研】一手更新 稳定有保障

可以得到二叉树 T 的后序序列为 b, f, e, d, c, a。

17．C



由二叉树 T 的先序序列和中序序列可以构造出 T，如下图所示。由森林转化成二叉树的规则可知，森林中每棵树的根结点以右子树的方式相连，所以 T 中的结点 a、c、f 为 F 中树的根结点，森林 F 中有 3 棵树，故选 C。

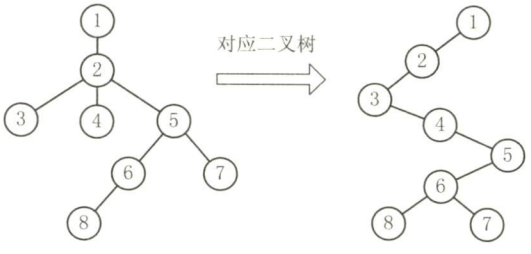
18．C

高度一定的三叉树中结点数最多的情况是满三叉树。高度为 5 的满三叉树的结点数 $= 3^0 + 3^1 + 3^2 + 3^3 + 3^4 = 121$，高度为 6 的满三叉树的结点数 $= 3^0 + 3^1 + 3^2 + 3^3 + 3^4 + 3^5 = 364$。由于三叉树 T 的结点数为 244，$121 < 244 < 364$，因此 T 的高度至少为 6。

二、综合应用题

01．【解答】

一棵树的先根遍历结果与其对应二叉树的先序遍历结果相同，树的后根遍历结果与其对应二叉树表示的中序遍历结果相同。由于二叉树的先序序列和中序序列能够唯一地确定这棵二叉树，因此，根据题目给出的条件，利用树的先根遍历序列和后根遍历序列能够唯一地确定这棵树。例如，对于右图所示的树，对应二叉树的先序序列为 1, 2, 3, 4, 5, 6, 8, 7，中序序列为 3, 4, 8, 6, 7, 5, 2, 1。原树的先根遍历序列为 1, 2, 3, 4, 5, 6, 8, 7，后根遍历序列为 3, 4, 8, 6, 7, 5, 2, 1。



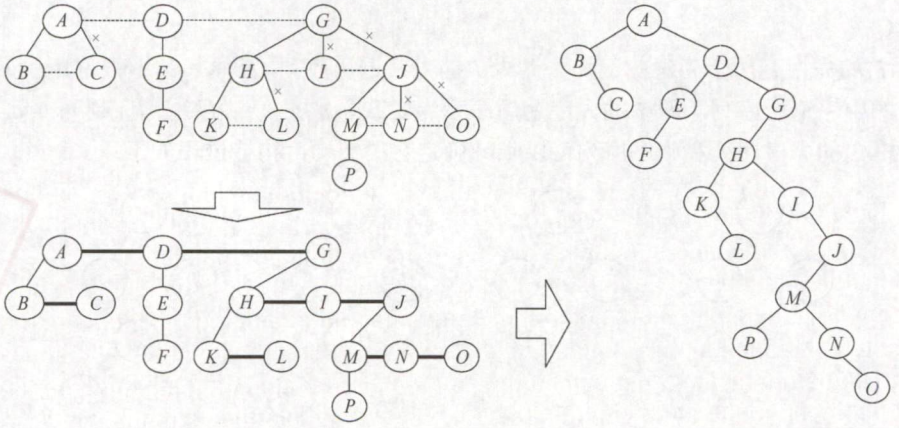
注意：树的先根遍历、后根遍历与对应二叉树的前序遍历、中序遍历对应。

02．【解答】

根据树与二叉树“左孩子右兄弟”的转换规则，将森林转换为二叉树的过程如下：①将每棵树的根结点也视为兄弟关系，在兄弟结点之间加一连线。②对每个结点，只保留它与第一个子结点的连线，与其他子结点的连线全部抹掉。③以树根为轴心，顺时针旋转 45°。结果如下图所示。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

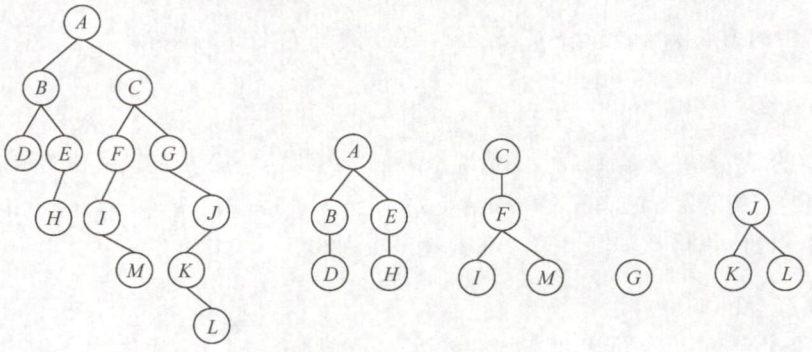
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】



03.【解答】

知道二叉树的先序和中序遍历后，可以唯一确定这棵树的结构。然后把二叉树转换到树和森林的方式是，若结点 x 是双亲 y 的左孩子，则把 x 的右孩子、右孩子的右孩子……都与 y 用连线连起来，最后去掉所有双亲到右孩子的连线。

最后得到的二叉树及对应的森林如下图所示。



04.【解答】

当森林（树）以孩子兄弟表示法存储时，若结点没有孩子（fch=NULL），则它必是叶子，总的叶结点个数是孩子子树（fch）上的叶子数和兄弟子树（nsib）上的叶结点个数之和。

算法代码如下：

```
typedef struct node
{
    ElemType    data;                   //数据域
    struct node *fch,*nsib;             //孩子与兄弟域
}*Tree;
int  Leaves(Tree t){                    //计算以孩子兄弟表示法存储的森林的叶子数
    if(t==NULL)
        return  0;                      //树空返回 0
    if(t->fch==NULL)                    //若结点无孩子，则该结点必是叶子
        return  1+Leaves(t->nsib);      //返回叶结点和其兄弟子树中的叶结点数
    else                                //孩子子树和兄弟子树中叶子数之和
        return  Leaves(t->fch)+Leaves(t->nsib);
}
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

05.【解答】

由孩子兄弟链表表示的树，求高度的算法思想如下：采用递归算法，若树为空，高度为零；否则，高度为第一子女树高度加 1 和兄弟子树高度的大者。其非递归算法使用队列，逐层遍历树，取得树的高度。算法代码如下：

```
int  Height(CSTree bt){
//递归求以孩子兄弟链表表示的树的深度
    int hc,hs;
    if(bt==NULL)
        return  0;
    else{//否则，高度取子女高度+1 和兄弟子树高度的大者
        hc=Height(bt->firstchild);        //第一子女树高
        hs=Height(bt->nextsibling);       //兄弟树高
        if(hc+1>hs)
            return  hc+1;
        else
            return  hs;
    }
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

06.【解答】

本题与树的层次序列有关。可设立一个辅助数组 pointer[]存储新建树的各结点的地址，再根据层次序列与每个结点的度，逐个链接结点。算法描述如下：

```
#define maxNodes 15
void createCSTree_Degree(CSTree&T,DataType e[],int degree[],int n){
//根据树结点的层次序列 e[]和各结点的度 degree[]构造树的孩子-兄弟链表
//参数 n 是树结点个数
    CSNode *pointer=new CSNode[maxNodes];//判断 pointer[i]为空的语句未写
    int i,j,d,k=0;
    for(i=0;i<n;i++){//初始化
        pointer[i]->data=e[i];
        pointer[i]->lchild=pointer[i]->rsibling=NULL;
    }
    for(i=0;i<n;i++){
        d=degree[i];                          //结点 i 的度数
        if(d){
            k++;                              //k 为子女结点序号
            pointer[i]->lchild=pointer[k];    //建立 i 与子女 k 间的链接
            for(j=2;j<=d;j++){
                k++;
                pointer[k-1]->rsibling=pointer[k];
            }
        }
    }
    T=pointer[0];
    delete [] pointer;
}
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

07.【解答】

1）正则 k 叉树中仅含有两类结点：叶结点（个数记为 n_0）和度为 k 的分支结点（个数记为 n_k）。树 T 中的结点总数 $n=n_0+n_k=n_0+m$。树中所含的边数 $e=n-1$，这些边均是从 m 个度为 k 的结点发出的，即 $e=mk$。整理得 $n_0+m=mk+1$，故 $n_0=(k-1)m+1$。

2）高度为 h 的正则 k 叉树 T 中，含最多结点的树形为：除第 h 层外，第 1 到第 $h-1$ 层的结点都是度为 k 的分支结点；而第 h 层均为叶结点，即树是“满”树。此时第 j（$1\leqslant j\leqslant h$）层的结点数为 k^{j-1}，结点总数 M_1 为

$$M_1=\sum_{j=1}^{h}k^{j-1}=\frac{k^h-1}{k-1}$$

含最少结点的正则 k 叉树的树形为：第 1 层只有根结点，第 2 到第 $h-1$ 层仅含 1 个分支结点和 $k-1$ 个叶结点，第 h 层有 k 个叶结点。也就是说，除根外，第 2 到第 h 层中每层的结点数均为 k，故 T 中所含结点总数 M_2 为

$$M_2=1+(h-1)k$$

5.5 树与二叉树的应用

5.5.1 哈夫曼树和哈夫曼编码

1. 哈夫曼树的定义

在许多应用中，树中结点常常被赋予一个表示某种意义的数值，称为该结点的权。从树的根到任意结点的路径长度（经过的边数）与该结点上权值的乘积，称为该结点的带权路径长度。树中所有叶结点的带权路径长度之和称为该树的带权路径长度，记为

$$\mathrm{WPL}=\sum_{i=1}^{n}w_il_i$$

式中，w_i 是第 i 个叶结点所带的权值，l_i 是该叶结点到根结点的路径长度。

在含有 n 个带权叶结点的二叉树中，其中带权路径长度（WPL）最小的二叉树称为哈夫曼树，也称最优二叉树。例如，图 5.18 中的 3 棵二叉树都有 4 个叶结点 a, b, c, d，分别带权 7, 5, 2, 4，它们的带权路径长度分别为

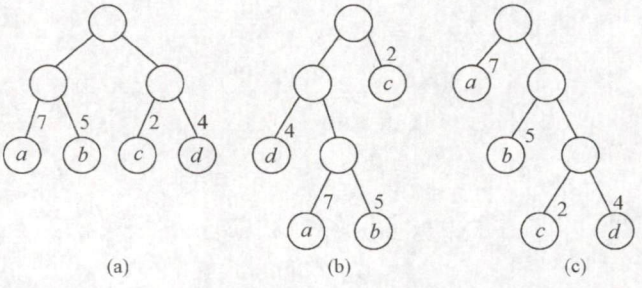


图 5.18 具有不同带权长度的二叉树

(a) WPL = 7×2 + 5×2 + 2×2 + 4×2 = 36。

(b) WPL = 4×2 + 7×3 + 5×3 + 2×1 = 46。

(c) WPL = 7×1 + 5×2 + 2×3 + 4×3 = 35。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

其中，图 5.18(c)树的 WPL 最小。可以验证，它恰好为哈夫曼树。

2．哈夫曼树的构造

给定 n 个权值分别为 $w_1, w_2, \cdots, w_n$ 的结点，构造哈夫曼树的算法描述如下：

1）将这 n 个结点分别作为 n 棵仅含一个结点的二叉树，构成森林 F。

2）构造一个新结点，从 F 中选取两棵根结点权值最小的树作为新结点的左、右子树，并且将新结点的权值置为左、右子树上根结点的权值之和。

3）从 F 中删除刚才选出的两棵树，同时将新得到的树加入 F 中。

4）重复步骤 2）和 3），直至 F 中只剩下一棵树为止。

从上述构造过程中可以看出哈夫曼树具有如下特点：

1）每个初始结点最终都成为叶结点，且权值越小的结点到根结点的路径长度越大。

2）构造过程中共新建了 $n-1$ 个结点（双分支结点），因此哈夫曼树的结点总数为 $2n-1$。

3）每次构造都选择 2 棵树作为新结点的孩子，因此哈夫曼树中不存在度为 1 的结点。

例如，权值{7, 5, 2, 4}的哈夫曼树的构造过程如图 5.19 所示。

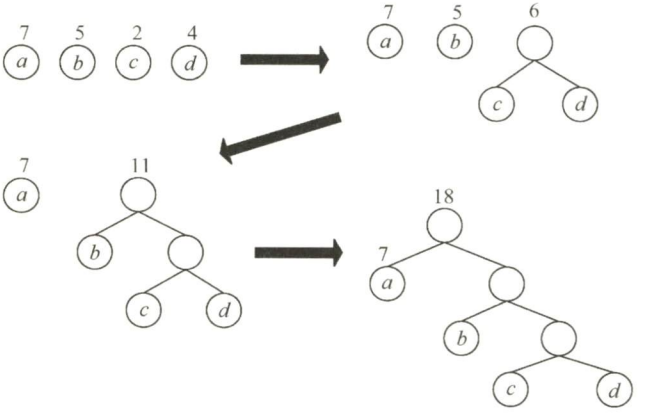


图 5.19　哈夫曼树的构造过程

关注公众号【乘龙考研】一手更新 稳定有保障

3．哈夫曼编码

在数据通信中，若对每个字符用相等长度的二进制位表示，称这种编码方式为*固定长度编码*。若允许对不同字符用不等长的二进制位表示，则这种编码方式称为*可变长度编码*。可变长度编码比固定长度编码要好得多，其特点是对频率高的字符赋以短编码，而对频率较低的字符则赋以较长一些的编码，从而可以使字符的平均编码长度减短，起到压缩数据的效果。哈夫曼编码是一种被广泛应用而且非常有效的数据压缩编码。

若没有一个编码是另一个编码的前缀，则称这样的编码为*前缀编码*。举例：设计字符 A, B 和 C 对应的编码 0, 101 和 100 是前缀编码。对前缀编码的解码很简单，因为没有一个编码是其他编码的前缀。所以识别出第一个编码，将它翻译为原码，再对余下的编码文件重复同样的解码操作。例如，码串 00101100 可被唯一地翻译为 0, 0, 101 和 100。另举反例：如果再将字符 D 的编码设计为 00，此时 0 是 00 的前缀，那么这样的码串的前两位就无法唯一翻译。

由哈夫曼树得到哈夫曼编码是很自然的过程。首先，将每个出现的字符当作一个独立的结点，其权值为它出现的频度（或次数），构造出对应的哈夫曼树。显然，所有字符结点都出现在叶结点中。我们可将字符的编码解释为从根至该字符的路径上边标记的序列，其中边标记为 0 表示“转向左孩子”，标记为 1 表示“转向右孩子”。图 5.20 所示为一个由哈夫曼树构造哈夫曼编码的示例，矩形方块表示字符及其出现的次数。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

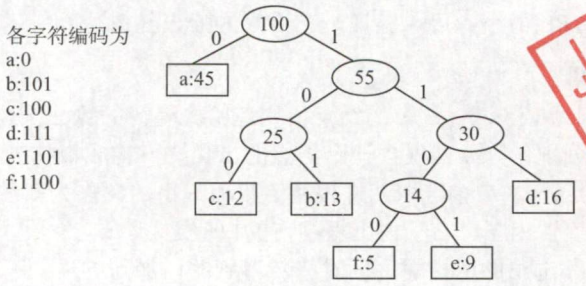


图 5.20 由哈夫曼树构造哈夫曼编码

关注公众号【乘龙考研】一手更新 稳定直保障

这棵哈夫曼树的 WPL 为

$$\mathrm{WPL} = 1\times45 + 3\times(13 + 12 + 16) + 4\times(5 + 9) = 224$$

此处的 WPL 可视为最终编码得到二进制编码的长度，共 224 位。若采用 3 位固定长度编码，则得到的二进制编码长度为 300 位，因此哈夫曼编码共压缩了 25%的数据。利用哈夫曼树可以设计出总长度最短的二进制前缀编码。

注意：0 和 1 究竟是表示左子树还是右子树没有明确规定。左、右孩子结点的顺序是任意的，所以构造出的哈夫曼树并不唯一，但各哈夫曼树的带权路径长度 WPL 相同且为最优。此外，如有若干权值相同的结点，则构造出的哈夫曼树更可能不同，但 WPL 必然相同且是最优的。

5.5.2 并查集

并查集是一种简单的集合表示，它支持以下 3 种操作：

1）`Initial(S)`：将集合 S 中的每个元素都初始化为只有一个单元素的子集合。

2）`Union(S,Root1,Root2)`：把集合 S 中的子集合 `Root2` 并入子集合 `Root1`。要求 `Root1` 和 `Root2` 互不相交，否则不执行合并。

3）`Find(S,x)`：查找集合 S 中单元素 x 所在的子集合，并返回该子集合的根结点。

通常用树（森林）的双亲表示作为并查集的存储结构，每个子集合以一棵树表示。所有表示子集合的树，构成表示全集合的森林，存放在双亲表示数组内。通常用数组元素的下标代表元素名，用根结点的下标代表子集合名，根结点的双亲结点为负数。

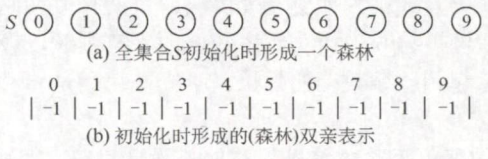


图 5.21 并查集的初始化

例如，若设有一个全集合为 $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$，初始化时每个元素自成一个单元素子集合，每个子集合的数组值为-1，如图 5.21 所示。

经过一段时间的计算，这些子集合合并为 3 个更大的子集合 $S_1 = \{0, 6, 7, 8\}$，$S_2 = \{1, 4, 9\}$，$S_3 = \{2, 3, 5\}$，此时并查集的树形表示和存储结构如图 5.22 所示。

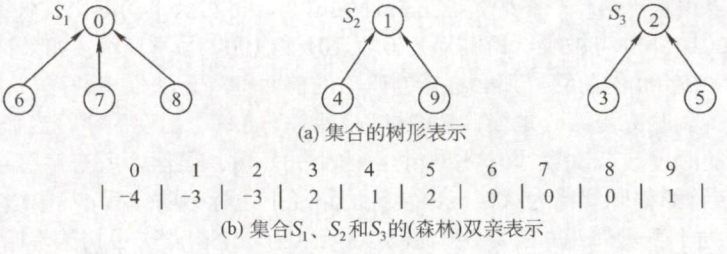
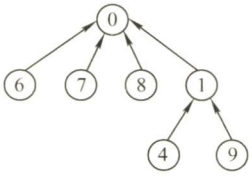


图 5.22 用树表示并查集

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

为了得到两个子集合的并，只需将其中一个子集合根结点的双亲指针指向另一个集合的根结点。因此，$S_1 \cup S_2$ 可以具有如图 5.23 所示的表示。



0	1	2	3	4	5	6	7	8	9
−7	0	−3	2	1	2	0	0	0	1

图 5.23　$S_1 \cup S_2$ 可能的表示方法

在采用树的双亲指针数组表示作为并查集的存储表示时，集合元素的编号从 0 到 SIZE-1。其中 SIZE 是最大元素的个数。下面是并查集主要运算的实现。

并查集的结构定义如下：

```
#define SIZE 100
int     UFSets[SIZE];                    //集合元素数组（双亲指针数组）
```

并查集的初始化操作（S 即为并查集）：

```
void Initial(int S[]){
    for(int i=0;i<SIZE;i++)              //每个自成单元素集合
        S[i]=-1;
}
```

Find 操作（函数在并查集 S 中查找并返回包含元素 x 的树的根）：

```
int Find(int S[],int x){
    while(S[x]>=0)                       //循环寻找 x 的根
        x=S[x];
    return  x;                           //根的 S[]小于 0
}
```

判断两个元素是否属于同一集合，只需分别找到它们的根结点，比较根结点是否相同即可。

Union 操作（函数求两个不相交子集合的并集）：

```
void Union(int S[],int Root1,int Root2){
    if(Root1==Root2) return;             //要求 Root1 与 Root2 是不同的集合
    S[Root2]=Root1;                      //将根 Root2 连接到另一根 Root1 下面
}
```

如果将两个元素所在的集合合并为一个集合，那么就需要先找到两个元素的根结点。

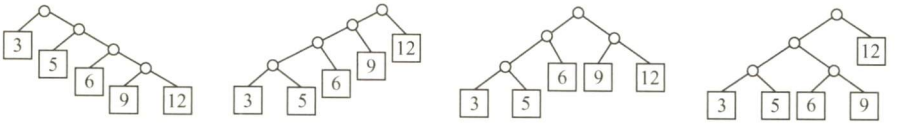
5.5.3　本节试题精选

一、单项选择题

01. 在有 n 个叶结点的哈夫曼树中，非叶结点的总数是（　）。

A. $n-1$　　B. n　　C. $2n-1$　　D. $2n$

02. 给定整数集合{3, 5, 6, 9, 12}，与之对应的哈夫曼树是（　）。



A.　　B.　　C.　　D.

03. 下列编码中，（　）不是前缀码。

A. {00, 01, 10, 11}　　B. {0, 1, 00, 11}

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

C. {0, 10, 110, 111}　　D. {10, 110, 1110, 1111}

04. 设哈夫曼编码的长度不超过4，若已对两个字符编码为1和01，则还最多可对（ ）个字符编码。

A. 2　　B. 3　　C. 4　　D. 5

05. 一棵哈夫曼树共有215个结点，对其进行哈夫曼编码，共能得到（ ）个不同的码字。

A. 107　　B. 108　　C. 214　　D. 215

06. 以下对于哈夫曼树的说法中，错误的是（ ）

A. 对应一组权值构造出来的哈夫曼树一般不是唯一的

B. 哈夫曼树具有最小的带权路径长度

C. 哈夫曼树中没有度为1的结点

D. 哈夫曼树中除了度为1的结点，还有度为2的结点和叶结点

07. 若度为 m 的哈夫曼树中，叶结点个数为 n，则非叶结点的个数为（ ）。

A. $n-1$　　B. $\lfloor n/m \rfloor - 1$

C. $\lceil (n-1)/(m-1) \rceil$　　D. $\lceil n/(m-1) \rceil - 1$

08. 并查集的结构是一种（ ）。

A. 二叉链表存储的二叉树　　B. 双亲表示法存储的树

C. 顺序存储的二叉树　　D. 孩子表示法存储的树

09. 并查集中最核心的两个操作是：①查找，查找两个元素是否属于同一个集合；②合并，如果两个元素不属于同一个集合，且所在的两个集合互不相交，则合并这两个集合。假设初始长度为10（0~9）的并查集，按1-2、3-4、5-6、7-8、8-9、1-8、0-5、1-9的顺序进行查找和合并操作，最终并查集共有（ ）个集合。

A. 1　　B. 2　　C. 3　　D. 4

10. 下列关于并查集的叙述中，（ ）是错误的（注意，本题涉及图的考点）。

A. 并查集是用双亲表示法存储的树

B. 并查集可用于实现克鲁斯卡尔算法

C. 并查集可用于判断无向图的连通性

D. 在长度为 n 的并查集中进行查找操作的时间复杂度为 $O(\log_2 n)$

11.【2010统考真题】n（$n \geq 2$）个权值均不相同的字符构成哈夫曼树，关于该树的叙述中，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. 树中任意一个非叶结点的权值一定不小于下一层任意一个结点的权值

12.【2014统考真题】5个字符有如下4种编码方案，不是前缀编码的是（ ）。

A. 01,0000,0001,001,1　　B. 011,000,001,010,1

C. 000,001,010,011,100　　D. 0,100,110,1110,1100

13.【2015统考真题】下列选项给出的是从根分别到达两个叶结点路径上的权值序列，能属于同一棵哈夫曼树的是（ ）。

A. 24, 10, 5和24, 10, 7　　B. 24,10, 5和24, 12, 7

C. 24, 10, 10和24, 14, 11　　D. 24, 10, 5和24, 14, 6

14.【2017统考真题】已知字符集{a, b, c, d, e, f, g, h}，若各字符的哈夫曼编码依次是0100, 10,

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

0000, 0101, 001, 011, 11, 0001，则编码序列 0100011001001011110101 的译码结果是（　）。

A. acgabfh　　B. adbagbb　　C. afbeagd　　D. afeefgd

15.【2018 统考真题】已知字符集{a, b, c, d, e, f}，若各字符出现的次数分别为 6, 3, 8, 2, 10, 4，则对应字符集中各字符的哈夫曼编码可能是（　）。

A. 00, 1011, 01, 1010, 11, 100　　B. 00, 100, 110, 000, 0010, 01

C. 10, 1011, 11, 0011, 00, 010　　D. 0011, 10, 11, 0010, 01, 000

16.【2019 统考真题】对 n 个互不相同的符号进行哈夫曼编码。若生成的哈夫曼树共有 115 个结点，则 n 的值是（　）。

A. 56　　B. 57　　C. 58　　D. 60

17.【2021 统考真题】若某二叉树有 5 个叶结点，其权值分别为 10, 12, 16, 21, 30，则其最小的带权路径长度（WPL）是（　）。

A. 89　　B. 200　　C. 208　　D. 289

18.【2022 统考真题】对任意给定的含 n（$n>2$）个字符的有限集 S，用二叉树表示 S 的哈夫曼编码集和定长编码集，分别得到二叉树 T_1 和 T_2。下列叙述中，正确的是（　）。

A. T_1 与 T_2 的结点数相同　　B. T_1 的高度大于 T_2 的高度

C. 出现频次不同的字符在 T_1 中处于不同的层

D. 出现频次不同的字符在 T_2 中处于相同的层

二、综合应用题

01. 设给定权集 $w=\{5, 7, 2, 3, 6, 8, 9\}$，试构造关于 w 的一棵哈夫曼树，并求其加权路径长度 WPL。

02.【2012 统考真题】设有 6 个有序表 A, B, C, D, E, F，分别含有 10, 35, 40, 50, 60 和 200 个数据元素，各表中的元素按升序排列。要求通过 5 次两两合并，将 6 个表最终合并为 1 个升序表，并使最坏情况下比较的总次数达到最小。请回答下列问题：

1）给出完整的合并过程，并求出最坏情况下比较的总次数。

2）根据你的合并过程，描述 n（$n\geqslant 2$）个不等长升序表的合并策略，并说明理由。

03.【2020 统考真题】若任意一个字符的编码都不是其他字符编码的前缀，则称这种编码具有前缀特性。现有某字符集（字符个数≥2）的不等长编码，每个字符的编码均为二进制的 0、1 序列，最长为 L 位，且具有前缀特性。请回答下列问题：

1）哪种数据结构适宜保存上述具有前缀特性的不等长编码？

2）基于你所设计的数据结构，简述从 0/1 串到字符串的译码过程。

3）简述判定某字符集的不等长编码是否具有前缀特性的过程。

5.5.4　答案与解析

关注公众号【乘龙考研】
一手更新 稳定有保障

一、单项选择题

01. A

由哈夫曼树的构造过程可知，哈夫曼树中只有度为 0 和 2 的结点。在非空二叉树中，有 $n_0=n_2+1$，故 $n_2=n-1$。

【另解】n 个结点构造哈夫曼树需要 $n-1$ 次合并过程，每次合并新建一个分支结点，故选择选项 A。

02. C

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

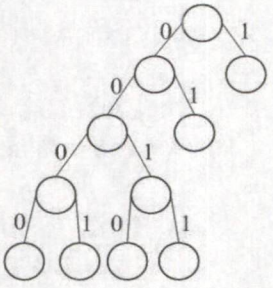
首先，3 和 5 构造为一棵子树，其根权值为 8，然后该子树与 6 构造为一棵新子树，根权值为 14，再后 9 与 12 构造为一棵子树，最后两棵子树共同构造为一棵哈夫曼树。

03．B

若没有一个编码是另一个编码的前缀，则称这样的编码为前缀编码。在选项 B 中，0 是 00 的前缀，1 是 11 的前缀。

04．C

在哈夫曼编码中，一个编码不能是任何其他编码的前缀。3 位编码可能是 001，对应的 4 位编码只能是 0000 和 0001。3 位编码也可能是 000，对应的 4 位编码只能是 0010 和 0011。若全采用 4 位编码，则可以为 0000, 0001, 0010 和 0011。题中问的是最多，故选择选项 C。



【另解】若哈夫曼编码的长度只允许小于或等于 4，则哈夫曼树的高度最高是 5，已知一个字符编码为 1，另一个字符编码是 01，这说明第二层和第三层各有一个叶结点，为使得该树从第 3 层起能够对尽可能多的字符编码，余下的二叉树应该是满二叉树，如右图所示，底层可以有 4 个叶结点，最多可以再对 4 个字符编码。

05．B

根据上题的结论，叶结点数为$(215+1)/2=108$，所以共有 108 个不同的码字。

【另解】在哈夫曼树中只有度为 0 和 2 的结点，结点总数 $n=n_0+n_2$，且 $n_0=n_2+1$，由题知 $n=215$，$n_0=108$。

06．D

哈夫曼树通常是指带权路径长度达到最小的扩充二叉树，在其构造过程中每次选根的权值最小的两棵树，一棵作为左子树，一棵作为右子树，生成新的二叉树，新的二叉树根的权值应为其左右两棵子树根结点权值的和。至于谁做左子树，谁做右子树，没有限制，所以构造的哈夫曼树是不唯一的。哈夫曼树只有度为 0 和 2 的结点，度为 0 的结点是外结点，带有权值，没有度为 1 的结点。

07．C

一棵度为 m 的哈夫曼树应只有度为 0 和 m 的结点，设度为 m 的结点有 n_m 个，度为 0 的结点有 n_0 个，又设结点总数为 N，$N=n_0+n_m$。因有 N 个结点的哈夫曼树有 $N-1$ 条分支，则 $mn_m=N-1=n_m+n_0-1$，整理得$(m-1)n_m=n_0-1$，$n_m=(n_0-1)/(m-1)$。

08．B

并查集的存储结构是用双亲表示法存储的树，主要是为了方便两个重要的操作。

09．C

初始时，0～9 各自成一个集合。查找 1-2 时，合并{1}和{2}；查找 3-4 时，合并{3}和{4}；查找 5-6 时，合并{5}和{6}；查找 7-8 时，合并{7}和{8}；查找 8-9 时，合并{7, 8}和{9}；查找 1-8 时，合并{1, 2}和{7, 8, 9}；查找 0-5 时，合并{0}和{5, 6}；查找 1-9 时，它们属于同一个集合。最终的集合为{0, 5, 6}、{1, 2, 7, 8, 9}和{3, 4}，因此答案选择选项 C。

10．D

在用并查集实现 Kruskal 算法求图的最小生成树时：判断是否加入一条边之前，先查找这条边关联的两个顶点是否属于同一个集合（即判断加入这条边之后是否形成回路），若形成回路，则继续判断下一条边；若不形成回路，则将该边和边对应的顶点加入最小生成树 T，并继续判断下一条边，直到所有顶点都已加入最小生成树 T。选项 B 正确。用并查集判断无向图连通性的方

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

法：遍历无向图的边，每遍历到一条边，就把这条边连接的两个顶点合并到同一个集合中，处理完所有边后，只要是相互连通的顶点都会被合并到同一个子集合中，相互不连通的顶点一定在不同的子集合中。选项 C 正确。未做路径优化的并查集在最坏情况下的高度为 n，此时查找操作的时间复杂度为 $O(n)$，时间复杂度通常指最坏情况下的时间复杂度。选项 D 错误。

11．A

哈夫曼树为带权路径长度最小的二叉树，不一定是完全二叉树。哈夫曼树中没有度为 1 的结点，选项 B 正确。构造哈夫曼树时，最先选取两个权值最小的结点作为左、右子树构造一棵新的二叉树，选项 C 正确。哈夫曼树中任意一个非叶结点 P 的权值为其左、右子树根结点的权值之和，其权值不小于其左、右子树根结点的权值，可知哈夫曼树中任意一个非叶结点的权值一定不小于下一层任意一个结点的权值，选项 D 正确。

12．D

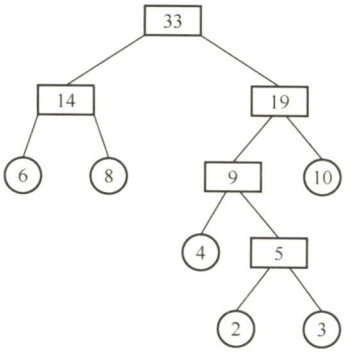
前缀编码的定义是在一个字符集中，任何一个字符的编码都不是另一个字符编码的前缀。选项 D 中的编码 110 是编码 1100 的前缀，违反了前缀编码的规则，所以选项 D 不是前缀编码。

13．D

在哈夫曼树中，左右孩子权值之和为父结点权值。仅以分析选项 A 为例：若两个 10 分别属于两棵不同的子树，则根的权值不等于其孩子的权值和，不符；若两个 10 属同棵子树，则其权值不等于其两个孩子（叶结点）的权值和，不符。选项 B、C 选项的排除方法一样。

14．D

哈夫曼编码是前缀编码，各个编码的前缀不同，因此直接拿编码序列与哈夫曼编码一一比对即可。序列可分割为 0100 011 001 001 011 11 0101，译码结果是 a f e e f g d。选项 D 正确。



15．A

构造一棵符合题意的哈夫曼树，如右图所示。

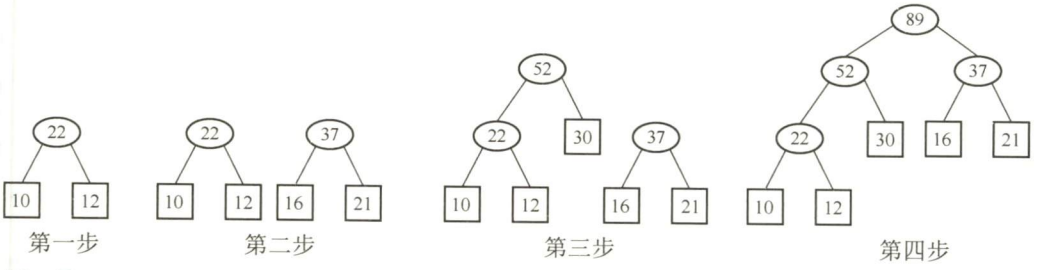
以左子树为 0，右子树为 1，可知答案为选项 A。

16．C

n 个符号构造成哈夫曼树的过程中，共新建了 $n-1$ 个结点（双分支结点），因此哈夫曼树的结点总数为 $2n-1=115$，n 的值为 58，答案选择选项 C。

17．B

对于带权值的结点，构造出哈夫曼树的带权路径长度（WPL）最小，哈夫曼树的构造过程如下图所示。求得其 WPL = (10 + 12)×3 + (30 + 16 + 21)×2 = 200，故选择选项 B。



18．D

可以画一个简单的特例来证明。图 1 是满足条件的二叉树 T_1，图 2 是满足条件的二叉树 T_2，结点中有值表示这个结点是编码字符。T_1 和 T_2 的结点数不同，选项 A 错误。T_1 的高度等于 T_2 的

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

高度，选项 B 错误。出现频次不同的字符在 T_1 中也可能处于相同的层，选项 C 错误。对于定长编码集，所有字符一定都在 T_2 中处于相同的层，而且都是叶结点。

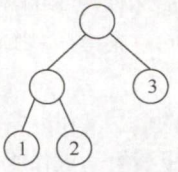


图 1

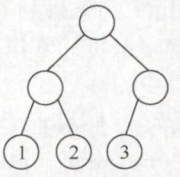
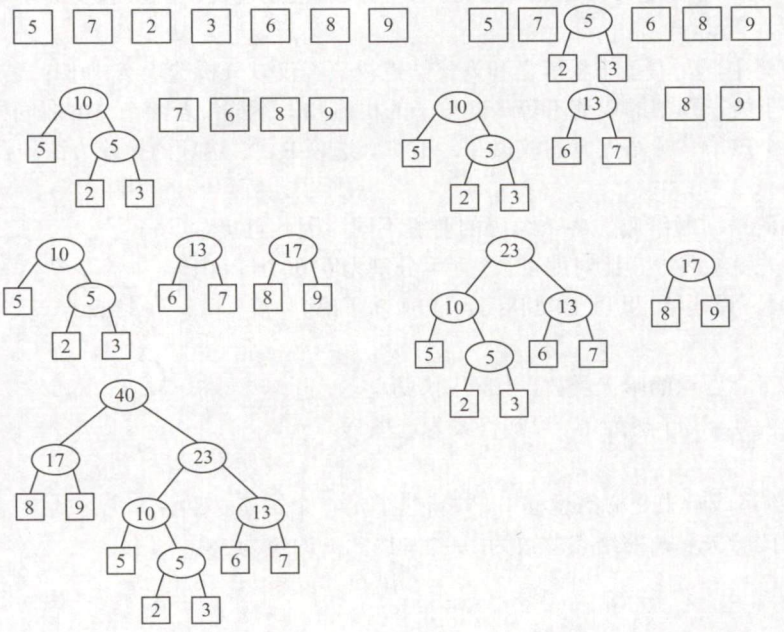


图 2

二、综合应用题

01.【解答】

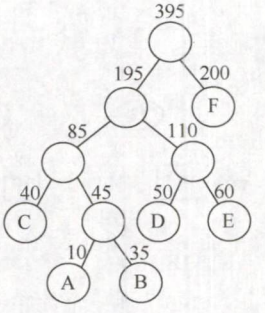
根据哈夫曼树的构造方法，每次从森林中选取两个根结点值最小的树合并成一棵树，将原先的两棵树作为左、右子树，且新根结点的值为左、右孩子关键字之和。构造过程如下图所示。



由构造出的哈夫曼树可得 WPL = (2 + 3)×4 + (5 + 6 + 7)×3 + (8 + 9)×2 = 108。

注意：哈夫曼树并不唯一，但带权路径长度一定是相同的。

02.【解答】



1）由于最先合并的表中的元素在后续的每次合并中都会再次参与比较，因此求最小合并次数类似于求最小带权路径长度，此时可立即想到哈夫曼树。根据哈夫曼树的构造过程，每次选择表集合中长度最小的两个表进行合并。6 个表的合并顺序如右图所示。

根据图中的哈夫曼树，6 个序列的合并过程如下：

① 在表集合{10, 35, 40, 50, 60, 200}中，选择表 A 与表 B 合并，生成含 45 个元素的表 AB。

② 在表集合{40, 45, 50, 60, 200}中，将表 AB 与表 C 合并，生成含 85 个元素的表 ABC。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

③ 在表集合{50, 60, 85, 200}中，表 D 与表 E 合并，生成含 110 个元素的表 DE。

④ 在表集合{85, 110, 200}中，表 ABC 与表 DE 合并，生成含 195 个元素的表 ABCDE。

⑤ 当前表集合为{195, 200}，表 ABCDE 与表 F 合并，生成含 395 个元素的表 ABCDEF。

由于合并两个长度分别为 m 和 n 的有序表，最坏情况下需要比较 $m+n-1$ 次，故最坏情况下比较的总次数计算如下：

第 1 次合并：最多比较次数 = 10 + 35 – 1 = 44。

第 2 次合并：最多比较次数 = 40 + 45 – 1 = 84。

第 3 次合并：最多比较次数 = 50 + 60 – 1 = 109。

第 4 次合并：最多比较次数 = 85 + 110 – 1 = 194。

第 5 次合并：最多比较次数 = 195 + 200 – 1 = 394。

比较的总次数最多为 44 + 84 + 109 + 194 + 394 = 825。

关注公众号【乘龙考研】
一手更新 稳定有保障

2）各表的合并策略是：对多个有序表进行两两合并时，若表长不同，则最坏情况下总的比较次数依赖于表的合并次序。可以借助于哈夫曼树的构造思想，依次选择最短的两个表进行合并，此时可以获得最坏情况下的最佳合并效率。

03.【解答】

1）使用一棵二叉树保存字符集中各字符的编码，每个编码对应于从根开始到达某叶结点的一条路径，路径长度等于编码位数，路径到达的叶结点中保存该编码对应的字符。

2）从左至右依次扫描 0/1 串中的各位。从根开始，根据串中当前位沿当前结点的左子指针或右子指针下移，直到移动到叶结点时为止。输出叶结点中保存的字符。然后从根开始重复这个过程，直到扫描到 0/1 串结束，译码完成。

3）二叉树既可用于保存各字符的编码，又可用于检测编码是否具有前缀特性。判定编码是否具有前缀特性的过程，也是构建二叉树的过程。初始时，二叉树中仅含有根结点，其左子指针和右子指针均为空。

依次读入每个编码 C，建立/寻找从根开始对应于该编码的一条路径，过程如下：

对每个编码，从左至右扫描 C 的各位，根据 C 的当前位（0 或 1）沿结点的指针（左子指针或右子指针）向下移动。当遇到空指针时，创建新结点，让空指针指向该新结点并继续移动。沿指针移动的过程中，可能遇到三种情况：

① 若遇到了叶结点（非根），则表明不具有前缀特性，返回。

② 若在处理 C 的所有位的过程中，均没有创建新结点，则表明不具有前缀特性，返回。

③ 若在处理 C 的最后一个编码位时创建了新结点，则继续验证下一个编码。

若所有编码均通过验证，则编码具有前缀特性。

归纳总结

本章的内容较多，其中二叉树是极其重要的考查点。关于二叉树的有关操作，在 2014 年的统考中第一次打破传统，出现了树的算法设计题，需要引起读者的注意。

遍历是二叉树的各种操作的基础，统考时会考查遍历过程中对结点的各种其他操作，而且容易结合递归算法和利用栈或队列的非递归算法。读者需重点掌握各种遍历方法的代码书写，并学会在遍历的基础上，进行一些其他的相关操作。其中递归算法短小精悍，出现的概率较大，请读者不要掉以轻心，要做到对几种遍历方式的程序模板烂熟于心，并结合一定数量的习题，才可以

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

在考试中快速地写出漂亮的代码。

二叉树遍历算法的递归程序：

```
void Track(BiTree *p){
    if(p!=NULL){
        //(1)
        Track(p->lchild);
        //(2)
        Track(p->rchild);
        //(3)
    }
}
```

访问函数 visit()位于（1）、（2）、（3）的位置，分别对应于先序、中序、后序遍历。但对于具体题目来说，设计算法时要灵活应用。请读者认真练习下面的例题。

例题：设二叉树的存储结构为二叉链表，编写有关二叉树的递归算法。

1）统计二叉树中度为 1 的结点个数。

2）统计二叉树中度为 2 的结点个数。

3）统计二叉树中度为 0 的结点个数。

4）统计二叉树的高度。

5）统计二叉树的宽度。

6）从二叉树中删去所有叶结点。

7）计算指定结点*p 所在的层次。

8）计算二叉树中各结点中的最大元素的值。

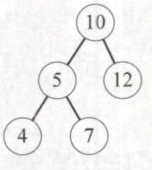
9）交换二叉树中每个结点的两个子女。

10）以先序次序输出一颗二叉树中所有结点的数据值及结点所在的层次。

关注公众号【乘龙考研】
一手更新 稳定有保障

思维拓展

输入一个整数 data 和一棵二元树。从树的根结点开始往下访问一直到叶结点，所经过的所有结点形成一条路径。打印出路径及与 data 相等的所有路径。例如，输入整数 22 和右图所示的二元树，则打印出两条路径 10, 12 和 10, 5, 7。



（**提示**：使用数组或栈保存访问的路径，并记录当前路径上所有元素的和 sum。若当前结点为叶结点，且当前结点值与 sum 的和等于 data，则满足条件，打印当前路径。然后递归返回到父结点，注意在递归返回之前要先减去当前结点元素的值。使用前序遍历操作的递归算法模板可以简化程序。）

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

关注公众号【乘龙考研】
一手更新 稳定有保障

第 6 章 图

【考纲内容】

（一）图的基本概念

（二）图的存储及基本操作

邻接矩阵；邻接表；邻接多重表；十字链表

（三）图的遍历

深度优先搜索；广度优先搜索

（四）图的基本应用

最小（代价）生成树；最短路径；拓扑排序；关键路径

兑换后

看视频讲解

【知识框架】

- 图
  - 图的定义
  - 图结构的存储
    - 邻接矩阵法、邻接表法
    - 邻接多重表、十字链表
  - 图的遍历
    - 深度优先遍历
    - 广度优先遍历
  - 图的相关应用
    - 最小生成树：Prim 算法、Kruskal 算法
    - 最短路径：Dijkstra 算法、Floyd 算法
    - 拓扑排序：AOV 网
    - 关键路径：AOE 网

【复习提示】

图算法的难度较大，主要掌握深度优先搜索与广度优先搜索。掌握图的基本概念及基本性质、图的存储结构（邻接矩阵、邻接表、邻接多重表和十字链表）及其特性、存储结构之间的转化、基于存储结构上的遍历操作和各种应用（拓扑排序、最小生成树、最短路径和关键路径）等。图的相关算法较多、易混，通常只要求掌握其基本思想和实现步骤，而算法的具体实现不是重点。

6.1 图的基本概念

6.1.1 图的定义

图 G 由顶点集 V 和边集 E 组成，记为 $G=(V,E)$，其中 $V(G)$表示图 G 中顶点的有限非空集；$E(G)$表示图 G 中顶点之间的关系（边）集合。若 $V=\{v_1,v_2,\cdots,v_n\}$，则用$|V|$表示图 G 中顶点的个数，$E=\{(u,v)\mid u\in V,v\in V\}$，用$|E|$表示图 G 中边的条数。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

注意：线性表可以是空表，树可以是空树，但图不可以是空图。就是说，图中不能一个顶点也没有，图的顶点集 V 一定非空，但边集 E 可以为空，此时图中只有顶点而没有边。

下面是图的一些基本概念及术语。

1．有向图

若 E 是有向边（也称弧）的有限集合时，则图 G 为有向图。弧是顶点的有序对，记为$<v, w>$，其中 v, w 是顶点，v 称为弧尾，w 称为弧头，$<v, w>$称为从 v 到 w 的弧，也称 v 邻接到 w。

图 6.1(a)所示的有向图 G_1 可表示为

$$G_1 = (V_1, E_1)$$
$$V_1 = \{1, 2, 3\}$$
$$E_1 = \{<1, 2>, <2, 1>, <2, 3>\}$$

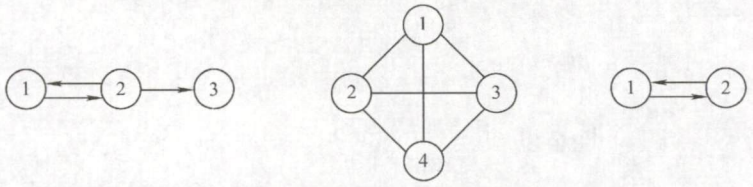
关注公众号【乘龙考研】
一手更新 稳定有保障

2．无向图

若 E 是无向边（简称边）的有限集合时，则图 G 为无向图。边是顶点的无序对，记为(v, w)或(w, v)。可以说 w 和 v 互为邻接点。边(v, w)依附于 w 和 v，或称边(v, w)和 v, w 相关联。

图 6.1(b)所示的无向图 G_2 可表示为

$$G_2 = (V_2, E_2)$$
$$V_2 = \{1, 2, 3, 4\}$$
$$E_2 = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$$



(a) 有向图G_1　　(b) 无向图G_2　　(c) 有向完全图G_3

图 6.1　图的示例

3．简单图、多重图

一个图 G 如果满足：①不存在重复边；②不存在顶点到自身的边，那么称图 G 为简单图。图 6.1 中 G_1 和 G_2 均为简单图。若图 G 中某两个顶点之间的边数大于 1 条，又允许顶点通过一条边和自身关联，则称图 G 为多重图。多重图和简单图的定义是相对的。数据结构中仅讨论简单图。

4．完全图（也称简单完全图）

对于无向图，$|E|$的取值范围为 0 到 $n(n-1)/2$，有 $n(n-1)/2$ 条边的无向图称为完全图，在完全图中任意两个顶点之间都存在边。对于有向图，$|E|$的取值范围为 0 到 $n(n-1)$，有 $n(n-1)$条弧的有向图称为有向完全图，在有向完全图中任意两个顶点之间都存在方向相反的两条弧。图 6.1 中 G_2 为无向完全图，而 G_3 为有向完全图。

5．子图

设有两个图 $G = (V, E)$和 $G' = (V', E')$，若 V'是 V 的子集，且 E'是 E 的子集，则称 G'是 G 的子图。若有满足 $V(G') = V(G)$的子图 G'，则称其为 G 的生成子图。图 6.1 中 G_3 为 G_1 的子图。

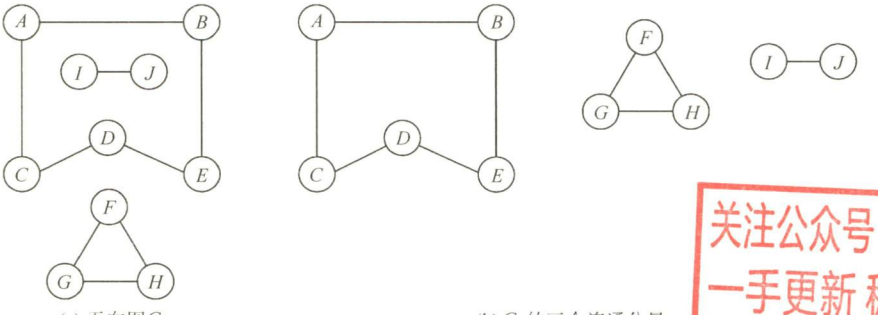
注意：并非 V 和 E 的任何子集都能构成 G 的子图，因为这样的子集可能不是图，即 E 的子集中的某些边关联的顶点可能不在这个 V 的子集中。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

6．连通、连通图和连通分量

在无向图中，若从顶点 v 到顶点 w 有路径存在，则称 v 和 w 是连通的。若图 G 中任意两个顶点都是连通的，则称图 G 为连通图，否则称为非连通图。无向图中的极大连通子图称为连通分量，在图 6.2(a)中，图 G_4 有 3 个连通分量如图 6.2(b)所示。假设一个图有 n 个顶点，如果边数小于 $n-1$，那么此图必是非连通图；思考，如果图是非连通图，那么最多可以有多少条边？[①]



(a) 无向图G_4　　(b) G_4的三个连通分量

关注公众号【乘龙考研】
一手更新 稳定有保障

图 6.2　无向图及其连通分量

7．强连通图、强连通分量

在有向图中，如果有一对顶点 v 和 w，从 v 到 w 和从 w 到 v 之间都有路径，则称这两个顶点是强连通的。若图中任何一对顶点都是强连通的，则称此图为强连通图。有向图中的极大强连通子图称为有向图的强连通分量，图 G_1 的强连通分量如图 6.3 所示。思考，假设一个有向图有 n 个顶点，如果是强连通图，那么最少需要有多少条边？[②]

注意：在无向图中讨论连通性，在有向图中讨论强连通性。

8．生成树、生成森林

连通图的生成树是包含图中全部顶点的一个极小连通子图。若图中顶点数为 n，则它的生成树含有 $n-1$ 条边。包含图中全部顶点的极小连通子图，只有生成树满足这个极小条件，对生成树而言，若砍去它的一条边，则会变成非连通图，若加上一条边则会形成一个回路。在非连通图中，连通分量的生成树构成了非连通图的生成森林。图 G_2 的一个生成树如图 6.4 所示。

图 6.3　图 G_1 的强连通分量

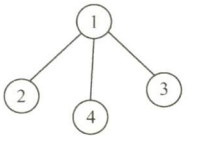


图 6.4　图 G_2 的一个生成树

注意：区分极大连通子图和极小连通子图。极大连通子图是无向图的连通分量，极大即要求该连通子图包含其所有的边；极小连通子图是既要保持图连通又要使得边数最少的子图。

9．顶点的度、入度和出度

在无向图中，顶点 v 的度是指依附于顶点 v 的边的条数，记为 TD(v)。在图 6.1(b)中，每个顶

① 非连通情况下边最多的情况：由 $n-1$ 个顶点构成一个完全图，此时再加入一个顶点则变成非连通图。
② 有向图强连通情况下边最少的情况：至少需要 n 条边，构成一个环路。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

点的度均为3。对于具有 n 个顶点、e 条边的无向图，$\sum_{i=1}^{n}\mathrm{TD}(v_i)=2e$，即无向图的全部顶点的度的和等于边数的2倍，因为每条边和两个顶点相关联。

在有向图中，顶点 v 的度分为入度和出度，入度是以顶点 v 为终点的有向边的数目，记为 $\mathrm{ID}(v)$；而出度是以顶点 v 为起点的有向边的数目，记为 $\mathrm{OD}(v)$。在图6.1(a)中，顶点2的出度为2、入度为1。顶点 v 的度等于其入度与出度之和，即 $\mathrm{TD}(v)=\mathrm{ID}(v)+\mathrm{OD}(v)$。对于具有 n 个顶点、e 条边的有向图，$\sum_{i=1}^{n}\mathrm{ID}(v_i)=\sum_{i=1}^{n}\mathrm{OD}(v_i)=e$，即有向图的全部顶点的入度之和与出度之和相等，并且等于边数，这是因为每条有向边都有一个起点和终点。

10．边的权和网

在一个图中，每条边都可以标上具有某种含义的数值，该数值称为该边的权值。这种边上带有权值的图称为带权图，也称网。

11．稠密图、稀疏图

边数很少的图称为稀疏图，反之称为稠密图。稀疏和稠密本身是模糊的概念，稀疏图和稠密图常常是相对而言的。一般当图 G 满足 $|E|<|V|\log|V|$ 时，可以将 G 视为稀疏图。

12．路径、路径长度和回路

顶点 v_p 到顶点 v_q 之间的一条路径是指顶点序列 $v_p,v_{i_1},v_{i_2},\cdots,v_{i_m},v_q$，当然关联的边也可理解为路径的构成要素。路径上边的数目称为路径长度。第一个顶点和最后一个顶点相同的路径称为回路或环。若一个图有 n 个顶点，并且有大于 $n-1$ 条边，则此图一定有环。

13．简单路径、简单回路

在路径序列中，顶点不重复出现的路径称为简单路径。除第一个顶点和最后一个顶点外，其余顶点不重复出现的回路称为简单回路。

14．距离

从顶点 u 出发到顶点 v 的最短路径若存在，则此路径的长度称为从 u 到 v 的距离。若从 u 到 v 根本不存在路径，则记该距离为无穷（∞）。

15．有向树

一个顶点的入度为0、其余顶点的入度均为1的有向图，称为有向树。

6.1.2 本节试题精选

一、单项选择题

关注公众号【乘龙考研】
一手更新 稳定有保障

01. 图中有关路径的定义是（ ）。

A. 由顶点和相邻顶点序偶构成的边所形成的序列

B. 由不同顶点所形成的序列

C. 由不同边所形成的序列

D. 上述定义都不是

02. 一个有 n 个顶点和 n 条边的无向图一定是（ ）。

A. 连通的　　B. 不连通的　　C. 无环的　　D. 有环的

03. 若从无向图的任意顶点出发进行一次深度优先搜索即可访问所有顶点，则该图一定是（ ）。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

A. 强连通图　　B. 连通图　　C. 有回路　　D. 一棵树

04. 以下关于图的叙述中，正确的是（ ）。

A. 图与树的区别在于图的边数大于或等于顶点数

B. 假设有图 $G=\{V,\{E\}\}$，顶点集 $V'\subseteq V$，$E'\subseteq E$，则 V'和$\{E'\}$构成 G 的子图

C. 无向图的连通分量是指无向图中的极大连通子图

D. 图的遍历就是从图中某一顶点出发访遍图中其余顶点

关注公众号【乘龙考研】一手更新 稳定有保障

05. 以下关于图的叙述中，正确的是（ ）。

A. 强连通有向图的任何顶点到其他所有顶点都有弧

B. 图的任意顶点的入度等于出度

C. 有向完全图一定是强连通有向图

D. 有向图的边集的子集和顶点集的子集都构成原有向图的子图

06. 一个有 28 条边的非连通无向图至少有（ ）个顶点。

A. 7　　B. 8　　C. 9　　D. 10

07. 对于一个有 n 个顶点的图：若是连通无向图，其边的个数至少为（ ）；若是强连通有向图，其边的个数至少为（ ）。

A. $n-1, n$　　B. $n-1, n(n-1)$　　C. n, n　　D. $n, n(n-1)$

08. 无向图 G 有 23 条边，度为 4 的顶点有 5 个，度为 3 的顶点有 4 个，其余都是度为 2 的顶点，则图 G 有（ ）个顶点。

A. 11　　B. 12　　C. 15　　D. 16

09. 在有 n 个顶点的有向图中，顶点的度最大可达（ ）。

A. n　　B. $n-1$　　C. $2n$　　D. $2n-2$

10. 具有 6 个顶点的无向图，当有（ ）条边时能确保是一个连通图。

A. 8　　B. 9　　C. 10　　D. 11

11. 设有无向图 $G=(V,E)$和 $G'=(V',E')$，若 G'是 G 的生成树，则下列不正确的是（ ）。

I. G'为 G 的连通分量

II. G'为 G 的无环子图

III. G'为 G 的极小连通子图且 $V'=V$

A. I、II　　B. 只有 III　　C. II、III　　D. 只有 I

12. 若具有 n 个顶点的图是一个环，则它有（ ）棵生成树。

A. n^2　　B. n　　C. $n-1$　　D. 1

13. 若一个具有 n 个顶点、e 条边的无向图是一个森林，则该森林中必有（ ）棵树。

A. n　　B. e　　C. $n-e$　　D. 1

14. 【2009 统考真题】下列关于无向连通图特性的叙述中，正确的是（ ）。

I. 所有顶点的度之和为偶数

II. 边数大于顶点个数减 1

III. 至少有一个顶点的度为 1

A. 只有 I　　B. 只有 II　　C. I 和 II　　D. I 和 III

15. 【2010 统考真题】若无向图 $G=(V,E)$中含有 7 个顶点，要保证图 G 在任何情况下都是连通的，则需要的边数最少是（ ）。

A. 6　　B. 15　　C. 16　　D. 21

16. 【2011 统考真题】下列关于图的叙述中，正确的是（ ）。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

I. 回路是简单路径

II. 存储稀疏图，用邻接矩阵比邻接表更省空间

III. 若有向图中存在拓扑序列，则该图不存在回路

A. 仅II　　B. 仅I、II　　C. 仅III　　D. 仅I、III

17.【2013统考真题】设图的邻接矩阵 A 如下所示，各顶点的度依次是（　）。

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

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

18.【2017统考真题】已知无向图 G 含有16条边，其中度为4的顶点个数为3，度为3的顶点个数为4，其他顶点的度均小于3。图 G 所含的顶点个数至少是（　）。

A. 10　　B. 11　　C. 13　　D. 15

19.【2022统考真题】对于无向图 $G=(V,E)$，下列选项中，正确的是（　）。

A. 当 $|V|>|E|$ 时，G 一定是连通的

B. 当 $|V|<|E|$ 时，G 一定是连通的

C. 当 $|V|=|E|-1$ 时，G 一定是不连通的

D. 当 $|V|>|E|+1$ 时，G 一定是不连通的

关注公众号【乘龙考研】
一手更新 稳定有保障

二、综合应用题

01. 图 G 是一个非连通无向图，共有28条边，该图至少有多少个顶点？

02. 如何对无环有向图中的顶点号重新安排可使得该图的邻接矩阵中所有的1都集中到对角线以上？

6.1.3 答案与解析

一、单项选择题

01. A

本题是北交大考研真题，不同教材对路径的定义可能略有不同，顶点之间关联的边也可理解为路径的构成要素。对于B选项，路径的定义中并没有要求是不同顶点，比如简单回路的第一个顶点和最后一个顶点是可以相同的，此外B选项也没有说明这些顶点之间有边相联。

02. D

若一个无向图有 n 个顶点和 $n-1$ 条边，可以使它连通但没有环（即生成树），但若再加一条边，在不考虑重边的情形下，则必然会构成环。

03. B

强连通图是有向图，与题意矛盾，选项A错误；对无向连通图做一次深度优先搜索，可以访问到该连通图的所有顶点，选项B正确；有回路的无向图不一定是连通图，因为回路不一定包含图的所有结点，选项C错误；连通图可能是树，也可能存在环，选项D错误。

04. C

图与树的区别是逻辑上的区别，而不是边数的区别，图的边数也可能小于树的边数，故选项A错；若 E' 中的边对应的顶点不是 V' 的元素，V' 和 $\{E'\}$ 无法构成图，故选项B错；无向图的极大连通子图称为连通分量，选项C正确；图的遍历要求每个结点只能被访问一次，且若图非连通，

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

则从某一顶点出发无法访问到其他全部顶点，选项 D 的说法不准确。

05. C

强连通有向图的任何顶点到其他所有顶点都有路径，但未必有弧；无向图任意顶点的入度等于出度，但有向图未必满足；若边集中的某条边对应的某个顶点不在对应的顶点集中，则有向图的边集的子集和顶点集的子集无法构成子图。

06. C

考查至少有多少个顶点的情形，我们考虑该非连通图最极端的情况，即它由一个完全图加一个独立的顶点构成，此时若再加一条边，则必然使图变成连通图。在 $28 = n(n-1)/2 = 8\times7/2$ 条边的完全无向图中，总共有 8 个顶点，再加上 1 个不连通的顶点，共 9 个顶点。

07. A

对于连通无向图，边最少即构成一棵树的情形；对于强连通有向图，边最少即构成一个有向环的情形。

08. D

由于在具有 n 个顶点、e 条边的无向图中，有 $\sum_{i=1}^{n}\mathrm{TD}(v_i)=2e$，故可求得度为 2 的顶点数为 7，从而共有 16 个顶点。

09. D

在有向图中，顶点的度等于入度与出度之和。n 个顶点的有向图中，任意一个顶点最多还可以与其他 $n-1$ 个顶点有一对指向相反的边相连。注意数据结构中仅讨论简单图。

10. D

5 个顶点构成一个完全无向图，需要 $n(n-1)/2 = 10$ 条边；再加上 1 条边后，能保证第 6 个顶点必然与此完全无向图构成一个连通图，故共需 11 条边。

11. D

一个连通图的生成树是一个极小连通子图，显然它是无环的，故选项 II、III 正确。极大连通子图称为连通分量，G' 连通但非连通分量。这里再补充一下“极大连通子图”：如果图本来就不是连通的，那么每个子部分若包含它本身的所有顶点和边，则它就是极大连通子图。

12. B

n 个顶点的生成树是具有 $n-1$ 条边的极小连通子图，因为 n 个顶点构成的环共有 n 条边，去掉任意一条边就是一棵生成树，所以共有 n 种情况，所以可以有 n 棵不同的生成树。

13. C

n 个结点的树有 $n-1$ 条边，假设森林中有 x 棵树，将每棵树的根连到一个添加的结点，则成为一棵树，结点数是 $n+1$，边数是 $e+x$，从而可知 $x=n-e$。

另解：设森林中有 x 棵树，则再用 $x-1$ 条边就能把所有的树连接成一棵树，此时，边数 +1 = 顶点数，即 $e+(x-1)+1=n$，故 $x=n-e$。

14. A

无向连通图对应的生成树也是无向连通图，但此时边数等于顶点数减 1，故选项 II 错误。考虑 2 个或以上顶点恰好构成一个环的情况，此时每个顶点的度都是 2，故选项 III 错误。在无向图中，所有顶点的度之和为边数的 2 倍，故选项 I 正确。

15. C

题干要求在“任何情况”下都是连通的，考虑最极端的情形，即图 G 的 6 个顶点构成一个完全无向图，再加上一条边后，第 7 个顶点必然与此完全无向图构成一个连通图，所以最少边数 = $6\times5/2+1=16$。若边数 n 小于或等于 15，可以使这 n 条边仅连接图 G 中的某 6 个顶点，从而导致

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

第 7 个顶点无法与这 6 个顶点构成连通图（不满足“任何情况”）。

16. C

回路对应于路径，简单回路对应于简单路径，故选项 I 错误；稀疏图是边比较少的情况，此时用邻接矩阵必将浪费大量的空间，应选用邻接表，故选项 II 错误。存在回路的图不存在拓扑序列，选项 III 正确。选项 II 和 III 中所涉知识点请参阅后面的内容。

17. C

邻接矩阵 $\boldsymbol{A}$ 为非对称矩阵，说明图是有向图，度为入度与出度之和。各顶点的度是矩阵中此结点对应的行（对应出度）和列（对应入度）的非零元素之和。

18. B

无向图边数的 2 倍等于各顶点度数的总和。为求至少的顶点数，应使每个顶点的度取最大，由于其他顶点的度均小于 3，可以设它们的度都为 2，设它们的数量是 x，列出方程 $4\times3+3\times4+2x=16\times2$，解得 $x=4$。因此至少包含 $4+4+3=11$ 个顶点。

19. D

注意，$|V|$ 是图的顶点数，$|E|$ 是图的边数。仅当 $|E|\geqslant(|V|-1)(|V|-2)/2+1$ 时（此时 $|V|-1$ 个顶点构成一个完全图，若再加入一条边，则一定变成连通图），图才一定是连通的。当 $|E|>|V|$ 时，图有可能是连通的，也可能是不连通的，选项 B 错误。当 $|E|<|V|$ 时，仅当 $|E|=|V|-1$ 时，图才可能是连通的（此时图是生成树），选项 A 错误。当 $|E|=|V|+1$ 时，图有可能是连通的，也可能是不连通的，选项 C 错误。当 $|E|<|V|-1$ 时，图一定是不连通的，选项 D 正确。

二、综合应用题

01.【解答】

由于图 G 是一个非连通无向图，在边数固定时，顶点数最少的情况是该图由两个连通子图构成，且其中之一只含一个顶点，另一个为完全图。其中只含一个顶点的子图没有边，另一个完全图的边数为 $n(n-1)/2=28$，得 $n=8$。所以该图至少有 $1+8=9$ 个顶点。

02.【解答】

按各顶点的出度进行排序。n 个顶点的有向图，其顶点的最大出度是 $n-1$，最小出度为 0。这样排序后，出度最大的顶点编号为 1，出度最小的顶点编号为 n。之后，进行调整，即只要存在弧 $<i,j>$，就不管顶点 j 的出度是否大于顶点 i 的出度，都把 i 编号在顶点 j 的编号之前，因为只有 $i\leqslant j$，弧 $<i,j>$ 对应的 1 才能出现在邻接矩阵的上三角。

通过后面小节的学习，会发现采用拓扑排序并依次编号是一种更为简便的方法。

6.2 图的存储及基本操作

图的存储必须要完整、准确地反映顶点集和边集的信息。根据不同图的结构和算法，采用不同的存储方式将对程序的效率产生相当大的影响，因此所选的存储结构应适合于待求解的问题。

6.2.1 邻接矩阵法

所谓邻接矩阵存储，是指用一个一维数组存储图中顶点的信息，用一个二维数组存储图中边的信息（即各顶点之间的邻接关系），存储顶点之间邻接关系的二维数组称为邻接矩阵。

结点数为 n 的图 $G=(V,E)$ 的邻接矩阵 $\boldsymbol{A}$ 是 $n\times n$ 的。将 G 的顶点编号为 $v_1,v_2,\cdots,v_n$。若 $(v_i,v_j)\in E$，

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

则 $A[i][j]=1$，否则 $A[i][j]=0$。

$$A[i][j]=\begin{cases}1, & 若(v_i,v_j)或\langle v_i,v_j\rangle是E(G)中的边\\ 0, & 若(v_i,v_j)或\langle v_i,v_j\rangle不是E(G)中的边\end{cases}$$

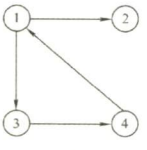
对于带权图而言，若顶点 v_i 和 v_j 之间有边相连，则邻接矩阵中对应项存放着该边对应的权值，若顶点 V_i 和 V_j 不相连，则通常用∞来代表这两个顶点之间不存在边：

$$A[i][j]=\begin{cases}w_{ij}, & 若(v_i,v_j)或\langle v_i,v_j\rangle是E(G)中的边\\ 0或\infty, & 若(v_i,v_j)或\langle v_i,v_j\rangle不是E(G)中的边\end{cases}$$

有向图、无向图和网对应的邻接矩阵示例如图 6.5 所示。

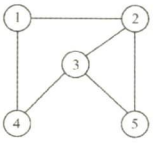
图的邻接矩阵存储结构定义如下：

```
#define MaxVertexNum 100                          //顶点数目的最大值
typedef char VertexType;                          //顶点的数据类型
typedef int EdgeType;                             //带权图中边上权值的数据类型
typedef struct{
    VertexType Vex[MaxVertexNum];                 //顶点表
    EdgeType Edge[MaxVertexNum][MaxVertexNum]; //邻接矩阵，边表
    int vexnum,arcnum;                            //图的当前顶点数和弧数
}MGraph;
```



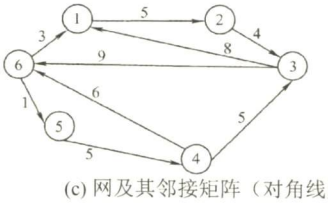
$$A_1=\begin{bmatrix}0&1&1&0\\0&0&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

(a) 有向图 G_1 及其邻接矩阵



$$A_2=\begin{bmatrix}0&1&0&1&0\\1&0&1&0&1\\0&1&0&1&1\\1&0&1&0&0\\0&1&1&0&0\end{bmatrix}$$

(b) 无向图 G_2 及其邻接矩阵



$$A_3=\begin{bmatrix}\infty&5&\infty&\infty&\infty&\infty\\\infty&\infty&4&\infty&\infty&\infty\\8&\infty&\infty&\infty&\infty&9\\\infty&\infty&5&\infty&\infty&6\\\infty&\infty&\infty&5&\infty&\infty\\3&\infty&\infty&\infty&1&\infty\end{bmatrix}$$

(c) 网及其邻接矩阵（对角线元素也经常用0表示）

关注公众号【乘龙考研】
一手更新 稳定有保障

图 6.5　有向图、无向图及网的邻接矩阵

注意：

①在简单应用中，可直接用二维数组作为图的邻接矩阵（顶点信息等均可省略）。

②当邻接矩阵的元素仅表示相应边是否存在时，EdgeType 可采用值为 0 和 1 的枚举类型。

③无向图的邻接矩阵是对称矩阵，对规模特大的邻接矩阵可采用压缩存储。

④邻接矩阵表示法的空间复杂度为 $O(n^2)$，其中 n 为图的顶点数$|V|$。

图的邻接矩阵存储表示法具有以下特点：

① 无向图的邻接矩阵一定是一个对称矩阵（并且唯一）。因此，在实际存储邻接矩阵时只需存储上（或下）三角矩阵的元素。

② 对于无向图，邻接矩阵的第 i 行（或第 i 列）非零元素（或非∞元素）的个数正好是顶点 i 的度 $TD(v_i)$。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

③ 对于有向图，邻接矩阵的第 i 行非零元素（或非∞元素）的个数正好是顶点 i 的出度 $OD(v_i)$；第 i 列非零元素（或非∞元素）的个数正好是顶点 i 的入度 $ID(v_i)$。

④ 用邻接矩阵存储图，很容易确定图中任意两个顶点之间是否有边相连。但是，要确定图中有多少条边，则必须按行、按列对每个元素进行检测，所花费的时间代价很大。

⑤ 稠密图适合使用邻接矩阵的存储表示。

⑥ 设图 G 的邻接矩阵为 $\boldsymbol{A}$，$\boldsymbol{A}^n$ 的元素 $A^n[i][j]$ 等于由顶点 i 到顶点 j 的长度为 n 的路径的数目。该结论了解即可，证明方法请参考离散数学教材。

6.2.2 邻接表法

当一个图为稀疏图时，使用邻接矩阵法显然要浪费大量的存储空间，而图的邻接表法结合了顺序存储和链式存储方法，大大减少了这种不必要的浪费。

所谓邻接表，是指对图 G 中的每个顶点 v_i 建立一个单链表，第 i 个单链表中的结点表示依附于顶点 v_i 的边（对于有向图则是以顶点 v_i 为尾的弧），这个单链表就称为顶点 v_i 的*边表*（对于有向图则称为*出边表*）。边表的头指针和顶点的数据信息采用顺序存储（称为*顶点表*），所以在邻接表中存在两种结点：顶点表结点和边表结点，如图 6.6 所示。

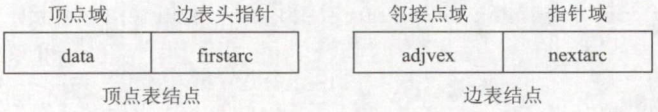


图 6.6 顶点表和边表结点结构

顶点表结点由顶点域（data）和指向第一条邻接边的指针（firstarc）构成，边表（邻接表）结点由邻接点域（adjvex）和指向下一条邻接边的指针域（nextarc）构成。

无向图和有向图的邻接表的实例分别如图 6.7 和图 6.8 所示。

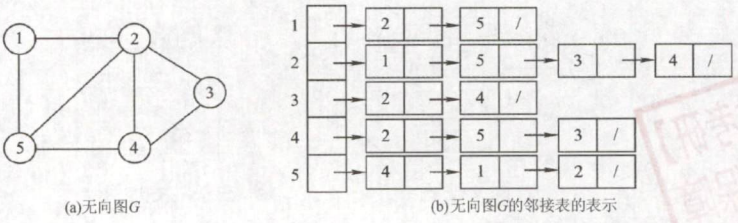


图 6.7 无向图邻接表表示法实例

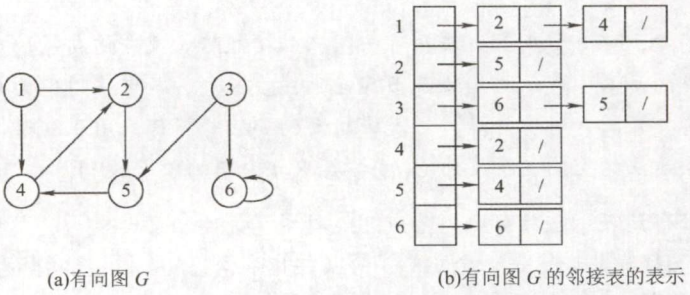


图 6.8 有向图邻接表表示法实例

图的邻接表存储结构定义如下：

关注公众号【乘龙考研】一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
#define MaxVertexNum 100          //图中顶点数目的最大值
typedef struct ArcNode{           //边表结点
    int adjvex;                   //该弧所指向的顶点的位置
    struct ArcNode *next;         //指向下一条弧的指针
    //InfoType info;              //网的边权值
}ArcNode;
typedef struct VNode{             //顶点表结点
    VertexType data;              //顶点信息
    ArcNode *first;               //指向第一条依附该顶点的弧的指针
}VNode,AdjList[MaxVertexNum];
typedef struct{
    AdjList vertices;             //邻接表
    int vexnum,arcnum;            //图的顶点数和弧数
} ALGraph;                        //ALGraph 是以邻接表存储的图类型
```

关注公众号【乘龙考研】一手更新 稳定有保障

图的邻接表存储方法具有以下特点：

① 若 G 为无向图，则所需的存储空间为 $O(|V|+2|E|)$；若 G 为有向图，则所需的存储空间为 $O(|V|+|E|)$。前者的倍数 2 是由于无向图中，每条边在邻接表中出现了两次。

② 对于稀疏图，采用邻接表表示将极大地节省存储空间。

③ 在邻接表中，给定一顶点，能很容易地找出它的所有邻边，因为只需要读取它的邻接表。在邻接矩阵中，相同的操作则需要扫描一行，花费的时间为 $O(n)$。但是，若要确定给定的两个顶点间是否存在边，则在邻接矩阵中可以立刻查到，而在邻接表中则需要在相应结点对应的边表中查找另一结点，效率较低。

④ 在有向图的邻接表表示中，求一个给定顶点的出度只需计算其邻接表中的结点个数；但求其顶点的入度则需要遍历全部的邻接表。因此，也有人采用逆邻接表的存储方式来加速求解给定顶点的入度。当然，这实际上与邻接表存储方式是类似的。

⑤ 图的邻接表表示并不唯一，因为在每个顶点对应的单链表中，各边结点的链接次序可以是任意的，它取决于建立邻接表的算法及边的输入次序。

6.2.3 十字链表

十字链表是有向图的一种链式存储结构。在十字链表中，对应于有向图中的每条弧有一个结点，对应于每个顶点也有一个结点。这些结点的结构如下图所示。

弧结点

tailvex	headvex	hlink	tlink	(info)

顶点结点

data	firstin	firstout

弧结点中有 5 个域：tailvex 和 headvex 两个域分别指示弧尾和弧头这两个顶点的编号；hlink 域指向弧头相同的下一个弧结点；tlink 域指向弧尾相同的下一个弧结点；info 域存放该弧的相关信息。这样，弧头相同的弧就在同一个链表上，弧尾相同的弧也在同一个链表上。

顶点结点中有 3 个域：data 域存放该顶点的数据信息，如顶点名称；firstin 域指向以该顶点为弧头的第一个弧结点；firstout 域指向以该顶点为弧尾的第一个弧结点。

图 6.9 为有向图的十字链表表示法。注意，顶点结点之间是顺序存储的。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

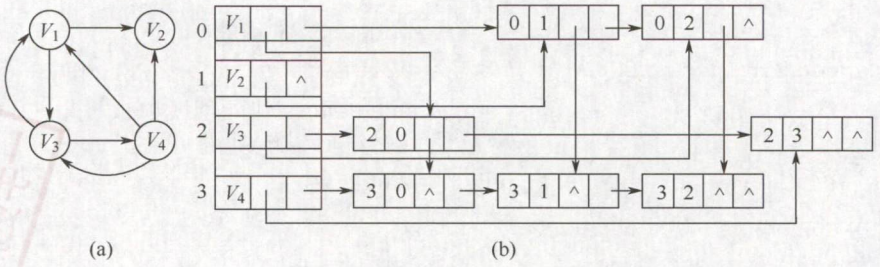


图 6.9 有向图的十字链表表示（弧结点省略 info 域）

在十字链表中，既容易找到 V_i 为尾的弧，也容易找到 V_i 为头的弧，因而容易求得顶点的出度和入度。图的十字链表表示是不唯一的，但一个十字链表表示确定一个图。

6.2.4 邻接多重表

邻接多重表是无向图的另一种链式存储结构。

在邻接表中，容易求得顶点和边的各种信息，但在邻接表中求两个顶点之间是否存在边而对边执行删除等操作时，需要分别在两个顶点的边表中遍历，效率较低。

与十字链表类似，在邻接多重表中，每条边用一个结点表示，其结构如下所示。

ivex	ilink	jvex	jlink	(info)

其中，ivex 和 jvex 这两个域指示该边依附的两个顶点的编号；ilink 域指向下一条依附于顶点 ivex 的边；jlink 域指向下一条依附于顶点 jvex 的边，info 域存放该边的相关信息。

每个顶点也用一个结点表示，它由如下所示的两个域组成。

data	firstedge

其中，data 域存放该顶点的相关信息，firstedge 域指向第一条依附于该顶点的边。

在邻接多重表中，所有依附于同一顶点的边串联在同一链表中，由于每条边依附于两个顶点，因此每个边结点同时链接在两个链表中。对无向图而言，其邻接多重表和邻接表的差别仅在于，同一条边在邻接表中用两个结点表示，而在邻接多重表中只有一个结点。

图 6.10 为无向图的邻接多重表表示法。邻接多重表的各种基本操作的实现和邻接表类似。

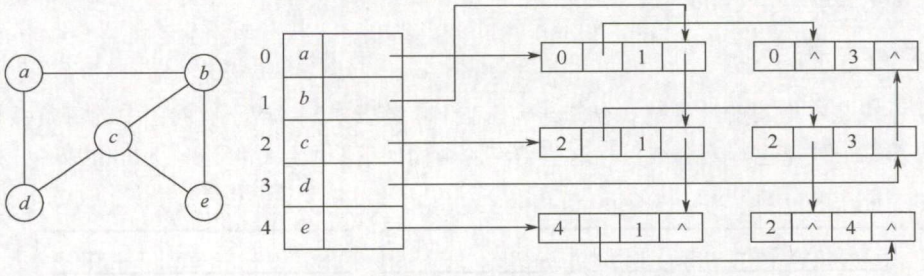


图 6.10 无向图的邻接多重表表示（边结点省略 info 域）

6.2.5 图的基本操作

图的基本操作是独立于图的存储结构的。而对于不同的存储方式，操作算法的具体实现会有着不同的性能。在设计具体算法的实现时，应考虑采用何种存储方式的算法效率会更高。

图的基本操作主要包括（仅抽象地考虑，故忽略掉各变量的类型）：

- Adjacent(G,x,y)：判断图 G 是否存在边$<x, y>$或(x, y)。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

- `Neighbors(G,x)`：列出图 G 中与结点 x 邻接的边。
- `InsertVertex(G,x)`：在图 G 中插入顶点 x。
- `DeleteVertex(G,x)`：从图 G 中删除顶点 x。
- `AddEdge(G,x,y)`：若无向边(x, y)或有向边$<x, y>$不存在，则向图 G 中添加该边。
- `RemoveEdge(G,x,y)`：若无向边(x, y)或有向边$<x, y>$存在，则从图 G 中删除该边。
- `FirstNeighbor(G,x)`：求图 G 中顶点 x 的第一个邻接点，若有则返回顶点号。若 x 没有邻接点或图中不存在 x，则返回 -1。
- `NextNeighbor(G,x,y)`：假设图 G 中顶点 y 是顶点 x 的一个邻接点，返回除 y 外顶点 x 的下一个邻接点的顶点号，若 y 是 x 的最后一个邻接点，则返回 -1。
- `Get_edge_value(G,x,y)`：获取图 G 中边(x, y)或$<x, y>$对应的权值。
- `Set_edge_value(G,x,y,v)`：设置图 G 中边(x, y)或$<x, y>$对应的权值为 v。

此外，还有图的遍历算法：按照某种方式访问图中的每个顶点且仅访问一次。图的遍历算法包括深度优先遍历和广度优先遍历，具体见下一节的内容。

6.2.6　本节试题精选

关注公众号【乘龙考研】
一手更新 稳定有保障

一、单项选择题

01. 关于图的存储结构，（　）是错误的。

A. 使用邻接矩阵存储一个图时，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的顶点数有关，与边数无关

B. 邻接表只用于有向图的存储，邻接矩阵适用于有向图和无向图

C. 若一个有向图的邻接矩阵的对角线以下的元素为 0，则该图的拓扑序列必定存在

D. 存储无向图的邻接矩阵是对称的，故只需存储邻接矩阵的下（或上）三角部分

02. 若图的邻接矩阵中主对角线上的元素皆为 0，其余元素全为 1，则可以断定该图一定（　）。

A. 是无向图　　B. 是有向图　　C. 是完全图　　D. 不是带权图

03. 在含有 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. n^2-e　　D. n^2-2e

04. 带权有向图 G 用邻接矩阵存储，则 v_i 的入度等于邻接矩阵中（　）。

A. 第 i 行非∞的元素个数　　B. 第 i 列非∞的元素个数

C. 第 i 行非∞且非 0 的元素个数　　D. 第 i 列非∞且非 0 的元素个数

05. 一个有 n 个顶点的图用邻接矩阵 $\boldsymbol{A}$ 表示，若图为有向图，顶点 v_i 的入度是（　）；若图为无向图，顶点 v_i 的度是（　）。

A. $\sum_{i=1}^{n} A[i][j]$　　B. $\sum_{j=1}^{n} A[j][i]$

C. $\sum_{i=1}^{n} A[j][i]$　　D. $\sum_{j=1}^{n} A[j][i]$ 或 $\sum_{j=1}^{n} A[i][j]$

06. 下列哪种图的邻接矩阵是对称矩阵？（　）

A. 有向网　　B. 无向图　　C. AOV 网　　D. AOE 网

07. 从邻接矩阵 $\boldsymbol{A}=\begin{bmatrix}0&1&0\\1&0&1\\0&1&0\end{bmatrix}$ 可以看出，该图共有（①）个顶点；若是有向图，则该图共有（②）条弧；若是无向图，则共有（③）条边。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

①A. 9　B. 3　C. 6　D. 1　E. 以上答案均不正确
②A. 5　B. 4　C. 3　D. 2　E. 以上答案均不正确
③A. 5　B. 4　C. 3　D. 2　E. 以上答案均不正确

08. 以下关于图的存储结构的叙述中，正确的是（ ）。
A. 一个图的邻接矩阵表示唯一，邻接表表示唯一
B. 一个图的邻接矩阵表示唯一，邻接表表示不唯一
C. 一个图的邻接矩阵表示不唯一，邻接表表示唯一
D. 一个图的邻接矩阵表示不唯一，邻接表表示不唯一

09. 用邻接表法存储图所用的空间大小（ ）。
A. 与图的顶点数和边数有关　B. 只与图的边数有关
C. 只与图的顶点数有关　D. 与边数的平方有关

10. 若邻接表中有奇数个边表结点，则（ ）。
A. 图中有奇数个结点　B. 图中有偶数个结点
C. 图为无向图　D. 图为有向图

11. 在有向图的邻接表存储结构中，顶点 v 在边表中出现的次数是（ ）。
A. 顶点 v 的度　B. 顶点 v 的出度
C. 顶点 v 的入度　D. 依附于顶点 v 的边数

12. n 个顶点的无向图的邻接表最多有（ ）个边表结点。
A. n^2　B. $n(n-1)$　C. $n(n+1)$　D. $n(n-1)/2$

13. 假设有 n 个顶点、e 条边的有向图用邻接表表示，则删除与某个顶点 v 相关的所有边的时间复杂度为（ ）。
A. $O(n)$　B. $O(e)$　C. $O(n+e)$　D. $O(ne)$

14. 对邻接表的叙述中，（ ）是正确的。
A. 无向图的邻接表中，第 i 个顶点的度为第 i 个链表中结点数的两倍
B. 邻接表比邻接矩阵的操作更简便
C. 邻接矩阵比邻接表的操作更简便
D. 求有向图结点的度，必须遍历整个邻接表

15. 邻接多重表是（ ）的存储结构。
A. 无向图　B. 有向图
C. 无向图和有向图　D. 都不是

16. 十字链表是（ ）的存储结构。
A. 无向图　B. 有向图
C. 无向图和有向图　D. 都不是

二、综合应用题

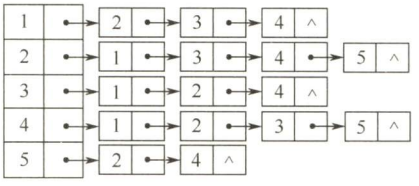
01. 已知带权有向图 G 的邻接矩阵如下图所示，请画出该带权有向图 G。

0	15	2	12	∞	∞	∞
∞	0	∞	∞	6	∞	∞
∞	∞	0	∞	8	4	∞
∞	∞	∞	0	∞	∞	3
∞	∞	∞	∞	0	∞	9
∞	∞	∞	5	∞	0	10
∞	4	∞	∞	∞	∞	0

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

02. 设图 $G=(V,E)$以邻接表存储，如下图所示。画出其邻接矩阵存储及图 G。



03. 对 n 个顶点的无向图和有向图，分别采用邻接矩阵和邻接表表示时，试问：

1）如何判别图中有多少条边？

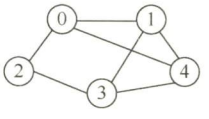
2）如何判别任意两个顶点 i 和 j 是否有边相连？

3）任意一个顶点的度是多少？

04. 写出从图的邻接表表示转换成邻接矩阵表示的算法。

05. 【2015 统考真题】已知含有 5 个顶点的图 G 如下图所示。

关注公众号【乘龙考研】
一手更新 稳定有保障



请回答下列问题：

1）写出图 G 的邻接矩阵 $\boldsymbol{A}$（行、列下标从 0 开始）。

2）求 $\boldsymbol{A}^2$，矩阵 $\boldsymbol{A}^2$ 中位于 0 行 3 列元素值的含义是什么？

3）若已知具有 n（$n\geqslant 2$）个顶点的图的邻接矩阵为 $\boldsymbol{B}$，则 $\boldsymbol{B}^m$（$2\leqslant m\leqslant n$）中非零元素的含义是什么？

06. 【2021 统考真题】已知无向连通图 G 由顶点集 V 和边集 E 组成，$|E|>0$，当 G 中度为奇数的顶点个数为不大于 2 的偶数时，G 存在包含所有边且长度为$|E|$的路径（称为 EL 路径）。设图 G 采用邻接矩阵存储，类型定义如下：

```
typedef struct{                          //图的定义
    int  numVertices,numEdges;           //图中实际的顶点数和边数
    char VerticesList[MAXV];             //顶点表。MAXV 为已定义常量
    int Edge[MAXV][MAXV];                //邻接矩阵
  }MGraph;
```

请设计算法 int IsExistEL(MGraph G)，判断 G 是否存在 EL 路径，若存在，则返回 1，否则返回 0。要求：

1）给出算法的基本设计思想。

2）根据设计思想，采用 C 或 C++语言描述算法，关键之处给出注释。

3）说明你所设计算法的时间复杂度和空间复杂度。

6.2.7　答案与解析

一、单项选择题

01. B

n 个顶点的图，若采用邻接矩阵表示，不考虑压缩存储，则存储空间大小为 $O(n^2)$，选项 A 正确。邻接表可用于存储无向图，只是把每条边都视为两条方向相反的有向边，因此需要存储两次，选项 B 错误。由于邻接矩阵中对角线以下的元素全为 0，若存在$<i,j>$，则必然有 $i<j$，由传递性可知图中路径的顶点编号是依次递增的，假设存在环 $k\to\cdots\to j\to k$，由题设可知 $k<j<k$，矛

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

盾，故不存在环，拓扑序列必定存在，选项C正确。选项D显然正确。

注意：若邻接矩阵对角线以下（或以上）的元素全为0，则图中必然不存在环，即拓扑序列一定存在，但这并不能说明拓扑序列是唯一的。

02．C

除主对角线上的元素外，其余元素全为1，说明任意两个顶点之间都有边相连，因此该图一定是完全图。

03．D

无向图的邻接矩阵中，矩阵大小为n^2，非零元素的个数为$2e$，故零元素的个数为n^2-2e。读者应掌握此题的变体，即当无向图变为有向图时，能够求出零的个数和非零的个数。

04．D

带权有向图的邻接矩阵中，0和∞表示的都不是有向边，而入度是由邻接矩阵的列中元素计算出来的；出度是由邻接矩阵的行中元素计算出来的。

05．B、D

有向图的入度是其第i列的非0元素之和，无向图的度是第i行或第i列的非0元素之和。

06．B

无向图的邻接矩阵存储中，每条边存储两次，且$A[i][j]=A[j][i]$。

07．B、B、D

邻接矩阵的顶点数等于矩阵的行（列）数，有向图的边数等于矩阵中非零元素的个数，无向图的边数等于矩阵中非零元素个数的一半。

注意：本题中所给的矩阵为对称矩阵，若不是对称矩阵，则必然不可能是无向图。

08．B

邻接矩阵表示唯一是因为图中边的信息在矩阵中有确定的位置，邻接表不唯一是因为邻接表的建立取决于读入边的顺序和边表中的插入算法。

09．A

邻接表存储时，顶点数n决定了顶点表的大小，边数e决定了边表结点的个数，且无向图的每条边存储两次，总存储空间为$O(n+2e)$，故选择选项A。而邻接矩阵只与图的顶点数有关，为$O(n^2)$。

10．D

无向图采用邻接表表示时，每条边存储两次，所以其边表结点的个数为偶数。题中边表结点为奇数个，故必然是有向图，且有奇数条边。

11．C

题中的边表是不包括顶点表的。因为任何顶点u对应的边表中存放的都是以u为起点的边所对应的另一个顶点v。从而v在边表中出现的次数也就是它的入度。

12．B

n个顶点的无向图最多有$n(n-1)/2$条边，每条边在邻接表中存储两次，所以边表结点最多为$n(n-1)$个。

13．C

与顶点v相关的边包括出边和入边，对于出边，只需遍历v的顶点表结点和其指向的边表；对于入边，则需遍历整个边表。先删除出边：删除v的顶点表结点的单链表，出边数最多为$n-1$，时间复杂度为$O(n)$；再删除入边：扫描整个边表（即扫描剩余全部顶点表结点及其指向的边表），

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

删除所有的顶点 v 的入边，时间复杂度为 $O(n+e)$。故总的时间复杂度为 $O(n+e)$。

14. D

无向图的邻接表中，第 i 个顶点的度为第 i 个链表中的结点数，故选项 A 错。邻接表和邻接矩阵对于不同的操作各有优势，选项 B 和 C 都不准确。有向图结点的度包括出度和入度，对于出度，需要遍历顶点表结点所对应的边表；对于入度，则需要遍历剩下的全部边表，故选项 D 正确。

15. A

邻接多重表是无向图的存储结构，选择选项 A。

16. B

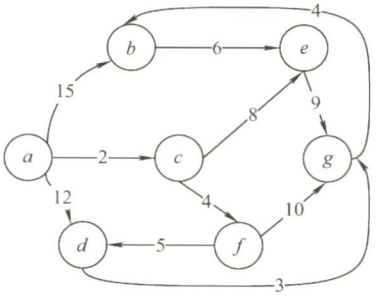
十字链表是有向图的存储结构，选择选项 B。

关注公众号【乘龙考研】
一手更新 稳定有保障

二、综合应用题

01.【解答】

带权有向图 G 如下图所示。

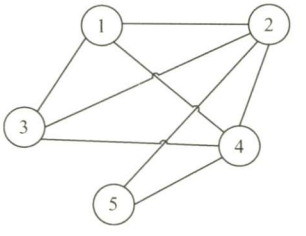


02.【解答】

其邻接矩阵存储如下所示。

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

在邻接表中，每条边存储了 2 次，在没有特殊说明时，通常默认其为无向图（当然，无向图也可视为具有对边的有向图）。该邻接表对应的图 G 如右图所示。



03.【解答】

1）对于邻接矩阵表示的无向图，边数等于矩阵中 1 的个数除以 2；对于邻接表表示的无向图，边数等于边结点的个数除以 2。对于邻接矩阵表示的有向图，边数等于矩阵中 1 的个数；对于邻接表表示的有向图，边数等于边结点的个数。

2）在邻接矩阵表示的无向图或有向图中，对于任意两个顶点 i 和 j，邻接矩阵中 arcs[i][j]或 arcs[j][i]为 1 表示有边相连，否则表示无边相连。在邻接表表示的无向图或有向图中，对于任意两个顶点 i 和 j，若从顶点表结点 i 出发找到编号为 j 的边表结点或从顶点表结点 j 出发找到编号为 i 的边表结点，表示有边相连；否则为无边相连。

3）对于邻接矩阵表示的无向图，顶点 i 的度等于第 i 行中 1 的个数；对于邻接矩阵表示的有

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

向图，顶点 i 的出度等于第 i 行中 1 的个数；入度等于第 i 列中 1 的个数；度数等于它们的和。对于邻接表表示的无向图，顶点 i 的度等于顶点表结点 i 的单链表中边表结点的个数；对于邻接表表示的有向图，顶点 i 的出度等于顶点表结点 i 的单链表中边表结点的个数，顶点 i 的入度等于邻接表中所有编号为 i 的边表结点数；度数等于入度与出度之和。

04.【解答】

算法的基本思想：设图的顶点分别存储在数组 v[n] 中。首先初始化邻接矩阵。遍历邻接表，在依次遍历顶点 v[i] 的边链表时，修改邻接矩阵的第 i 行的元素值。若链表边结点的值为 j，则置 arcs[i][j]=1。遍历完邻接表时，整个转换过程结束。此算法对于无向图、有向图均适用。

算法实现如下：

```
void Convert(ALGraph &G,int arcs[M][N]){
//此算法将邻接表方式表示的图 G 转换为邻接矩阵 arcs
    for(i=0;i<n;i++){              //依次遍历各顶点表结点为头的边链表
        p=(G->v[i]).firstarc;      //取出顶点 i 的第一条出边
        while(p!=NULL){            //遍历边链表
            arcs[i][p->adjvex]=1;
            p=p->nextarc;          //取下一条出边
        }                          //while
    }                              //for
}
```

05.【解答】

考查图的邻接矩阵的性质。

1）图 G 的邻接矩阵 $\boldsymbol{A}$ 如下：

$$\boldsymbol{A}=\begin{bmatrix}0&1&1&0&1\\1&0&0&1&1\\1&0&0&1&0\\0&1&1&0&1\\1&1&0&1&0\end{bmatrix}$$

2）$\boldsymbol{A}^2$ 如下：

$$\boldsymbol{A}^2=\begin{bmatrix}3&1&0&3&1\\1&3&2&1&2\\0&2&2&0&2\\3&1&0&3&1\\1&2&2&1&3\end{bmatrix}$$

0 行 3 列的元素值 3 表示从顶点 0 到顶点 3 之间长度为 2 的路径共有 3 条。

3）$\boldsymbol{B}^m$（$2\leqslant m\leqslant n$）中位于 i 行 j 列（$0\leqslant i, j\leqslant n-1$）的非零元素的含义是，图中从顶点 i 到顶点 j 的长度为 m 的路径条数。

06.【解答】

1）算法的基本设计思想

本算法题属于送分题，题干已经告诉我们算法的思想。对于采用邻接矩阵存储的无向图，在邻接矩阵的每一行（列）中，非零元素的个数为本行（列）对应顶点的度。可以依次计算连通图

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

G 中各顶点的度，并记录度为奇数的顶点个数，若个数为 0 或 2，则返回 1，否则返回 0。

2）算法实现

```
int IsExistEL(MGraph G){
    //采用邻接矩阵存储，判断图是否存在 EL 路径
    int degree,i,j,count=0;
    for(i=0;i<G.numVertices;i++){
        degree=0;
        for(j=0;j<G.numVertices;j++)
            degree+=G.Edge[i][j];       //依次计算各个顶点的度
        if(degree%2!=0)
            count++;        //对度为奇数的顶点计数
    }
    if(count==0||count==2)
        return 1;           //存在 EL 路径，返回 1
    else
        return 0;           //不存在 EL 路径，返回 0
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

3）时间复杂度和空间复杂度

算法需要遍历整个邻接矩阵，所以时间复杂度是 $O(n^2)$，空间复杂度是 $O(1)$。

6.3 图的遍历

图的遍历是指从图中的某一顶点出发，按照某种搜索方法沿着图中的边对图中的所有顶点访问一次且仅访问一次。注意到树是一种特殊的图，所以树的遍历实际上也可视为一种特殊的图的遍历。图的遍历算法是求解图的连通性问题、拓扑排序和求关键路径等算法的基础。

图的遍历比树的遍历要复杂得多，因为图的任意一个顶点都可能和其余的顶点相邻接，所以在访问某个顶点后，可能沿着某条路径搜索又回到该顶点上。为避免同一顶点被访问多次，在遍历图的过程中，必须记下每个已访问过的顶点，为此可以设一个辅助数组 visited[]来标记顶点是否被访问过。图的遍历算法主要有两种：广度优先搜索和深度优先搜索。

6.3.1 广度优先搜索

广度优先搜索（Breadth-First-Search, BFS）类似于二叉树的层序遍历算法。基本思想是：首先访问起始顶点 v，接着由 v 出发，依次访问 v 的各个未访问过的邻接顶点 $w_1, w_2, \cdots, w_i$，然后依次访问 $w_1, w_2, \cdots, w_i$ 的所有未被访问过的邻接顶点；再从这些访问过的顶点出发，访问它们所有未被访问过的邻接顶点，直至图中所有顶点都被访问过为止。若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为始点，重复上述过程，直至图中所有顶点都被访问到为止。Dijkstra 单源最短路径算法和 Prim 最小生成树算法也应用了类似的思想。

换句话说，广度优先搜索遍历图的过程是以 v 为起始点，由近至远依次访问和 v 有路径相通且路径长度为 $1, 2, \cdots$ 的顶点。广度优先搜索是一种分层的查找过程，每向前走一步可能访问一批顶点，不像深度优先搜索那样有往回退的情况，因此它不是一个递归的算法。为了实现逐层的访问，算法必须借助一个辅助队列，以记忆正在访问的顶点的下一层顶点。

广度优先搜索算法的伪代码如下：

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
bool visited[MAX_VERTEX_NUM];   //访问标记数组
void BFSTraverse(Graph G){      //对图 G 进行广度优先遍历
    for(i=0;i<G.vexnum;++i)
        visited[i]=FALSE;       //访问标记数组初始化
    InitQueue(Q);               //初始化辅助队列 Q
    for(i=0;i<G.vexnum;++i)     //从 0 号顶点开始遍历
        if(!visited[i])         //对每个连通分量调用一次 BFS
            BFS(G,i);           //vi 未访问过，从 vi 开始 BFS
}
void BFS(Graph G,int v){        //从顶点 v 出发，广度优先遍历图 G
    visit(v);                   //访问初始顶点 v
    visited[v]=TRUE;            //对 v 做已访问标记
    Enqueue(Q,v);               //顶点 v 入队列 Q
    while(!isEmpty(Q)){
        DeQueue(Q,v);           //顶点 v 出队列
        for(w=FirstNeighbor(G,v);w>=0;w=NextNeighbor(G,v,w))
                                //检测 v 所有邻接点
            if(!visited[w]){    //w 为 v 的尚未访问的邻接顶点
                visit(w);       //访问顶点 w
                visited[w]=TRUE;//对 w 做已访问标记
                EnQueue(Q,w);   //顶点 w 入队列
            }//if
    }//while
}
```

辅助数组 visited[]标志顶点是否被访问过，其初始状态为 FALSE。在图的遍历过程中，一旦某个顶点 v_i 被访问，就立即置 visited[i]为 TRUE，防止它被多次访问。

下面通过实例演示广度优先搜索的过程，给定图 G 如图 6.11 所示。

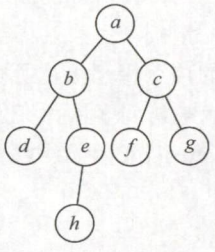


图 6.11 一个无向图 G

假设从 a 结点开始访问，a 先入队。此时队列非空，取出队头元素 a，由于 b, c 与 a 邻接且未被访问过，于是依次访问 b, c，并将 b, c 依次入队。队列非空，取出队头元素 b，依次访问与 b 邻接且未被访问的顶点 d, e，并将 d, e 入队（注意：a 与 b 也邻接，但 a 已置访问标记，故不再重复访问）。此时队列非空，取出队头元素 c，访问与 c 邻接且未被访问的顶点 f, g，并将 f, g 入队。此时，取出队头元素 d，但与 d 邻接且未被访问的顶点为空，故不进行任何操作。继续取出队头元素 e，将 h 入队列……最终取出队头元素 h 后，队列为空，从而循环自动跳出。遍历结果为 $abcdefgh$。

从上例不难看出，图的广度优先搜索的过程与二叉树的层序遍历是完全一致的，这也说明了图的广度优先搜索遍历算法是二叉树的层次遍历算法的扩展。

图的广度优先遍历还可用于求一些问题的最优解，但初试方面很难涉及。

1．BFS 算法的性能分析

无论是邻接表还是邻接矩阵的存储方式，BFS 算法都需要借助一个辅助队列 Q，n 个顶点均需入队一次，在最坏的情况下，空间复杂度为 $O(|V|)$。

采用邻接表存储方式时，每个顶点均需搜索一次（或入队一次），故时间复杂度为 $O(|V|)$，在搜索任意一个顶点的邻接点时，每条边至少访问一次，故时间复杂度为 $O(|E|)$，算法总的时间复杂度为 $O(|V|+|E|)$。采用邻接矩阵存储方式时，查找每个顶点的邻接点所需的时间为 $O(|V|)$，故算法总的时间复杂度为 $O(|V|^2)$。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

2．BFS 算法求解单源最短路径问题

若图 $G=(V, E)$为非带权图，定义从顶点 u 到顶点 v 的最短路径 $d(u, v)$为从 u 到 v 的任何路径中最少的边数；若从 u 到 v 没有通路，则 $d(u, v)=\infty$。

使用 BFS，我们可以求解一个满足上述定义的非带权图的单源最短路径问题，这是由广度优先搜索总是按照距离由近到远来遍历图中每个顶点的性质决定的。

BFS 算法求解单源最短路径问题的算法如下：

```
void BFS_MIN_Distance(Graph G,int u){
//d[i]表示从 u 到 i 结点的最短路径
    for(i=0;i<G.vexnum;++i)
        d[i]=∞;                         //初始化路径长度
    visited[u]=TRUE; d[u]=0;
    EnQueue(Q,u);
    while(!isEmpty(Q)){                 //BFS 算法主过程
        DeQueue(Q,u);                   //队头元素 u 出队
        for(w=FirstNeighbor(G,u);w>=0;w=NextNeighbor(G,u,w))
            if(!visited[w]){            //w 为 u 的尚未访问的邻接顶点
                visited[w]=TRUE;        //设已访问标记
                d[w]=d[u]+1;            //路径长度加 1
                EnQueue(Q,w);           //顶点 w 入队
            }//if
    }//while
}
```

关注公众号【乘龙考研】一手更新 稳定有保障

3．广度优先生成树

在广度遍历的过程中，我们可以得到一棵遍历树，称为广度优先生成树，如图 6.12 所示。需要注意的是，同一个图的邻接矩阵存储表示是唯一的，故其广度优先生成树也是唯一的，但由于邻接表存储表示不唯一，故其广度优先生成树也是不唯一的。

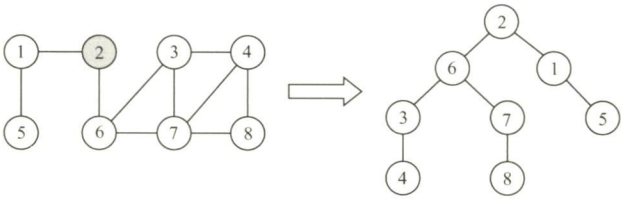


图 6.12　图的广度优先生成树

6.3.2　深度优先搜索

与广度优先搜索不同，深度优先搜索（Depth-First-Search，DFS）类似于树的先序遍历。如其名称中所暗含的意思一样，这种搜索算法所遵循的搜索策略是尽可能“深”地搜索一个图。

它的基本思想如下：首先访问图中某一起始顶点 v，然后由 v 出发，访问与 v 邻接且未被访问的任意一个顶点 w_1，再访问与 w_1 邻接且未被访问的任意一个顶点 w_2……重复上述过程。当不能再继续向下访问时，依次退回到最近被访问的顶点，若它还有邻接顶点未被访问过，则从该点开始继续上述搜索过程，直至图中所有顶点均被访问过为止。

一般情况下，其递归形式的算法十分简洁，算法过程如下：

```
bool visited[MAX_VERTEX_NUM];   //访问标记数组
void DFSTraverse(Graph G){      //对图 G 进行深度优先遍历
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        for(v=0;v<G.vexnum;++v)
            visited[v]=FALSE;          //初始化已访问标记数组
        for(v=0;v<G.vexnum;++v)        //本代码中是从 v=0 开始遍历
            if(!visited[v])
                DFS(G,v);
    }
    void DFS(Graph G,int v){           //从顶点 v 出发，深度优先遍历图 G
        visit(v);                      //访问顶点 v
        visited[v]=TRUE;               //设已访问标记
        for(w=FirstNeighbor(G,v);w>=0;w=NextNeighbor(G,v,w))
            if(!visited[w]){           //w 为 v 的尚未访问的邻接顶点
                DFS(G,w);
            }//if
    }
```

以图 6.11 的无向图为例，深度优先搜索的过程：首先访问 a，并置 a 访问标记；然后访问与 a 邻接且未被访问的顶点 b，置 b 访问标记；然后访问与 b 邻接且未被访问的顶点 d，置 d 访问标记。此时 d 已没有未被访问过的邻接点，故返回上一个访问过的顶点 b，访问与其邻接且未被访问的顶点 e，置 e 访问标记……以此类推，直至图中所有的顶点都被访问一次。遍历结果为 *abdehcfg*。

注意：图的邻接矩阵表示是唯一的，但对于邻接表来说，若边的输入次序不同，生成的邻接表也不同。因此，对于同样一个图，基于邻接矩阵的遍历所得到的 DFS 序列和 BFS 序列是唯一的，基于邻接表的遍历所得到的 DFS 序列和 BFS 序列是不唯一的。

1．DFS 算法的性能分析

DFS 算法是一个递归算法，需要借助一个递归工作栈，故其空间复杂度为 $O(|V|)$。

遍历图的过程实质上是对每个顶点查找其邻接点的过程，其耗费的时间取决于所用的存储结构。以邻接矩阵表示时，查找每个顶点的邻接点所需的时间为 $O(|V|)$，故总的时间复杂度为 $O(|V|^2)$。以邻接表表示时，查找所有顶点的邻接点所需的时间为 $O(|E|)$，访问顶点所需的时间为 $O(|V|)$，此时，总的时间复杂度为 $O(|V|+|E|)$。

2．深度优先的生成树和生成森林

与广度优先搜索一样，深度优先搜索也会产生一棵深度优先生成树。当然，这是有条件的，即对连通图调用 DFS 才能产生深度优先生成树，否则产生的将是深度优先生成森林，如图 6.13 所示。与 BFS 类似，基于邻接表存储的深度优先生成树是不唯一的。

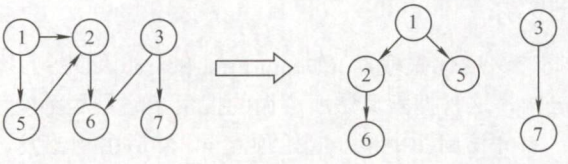


图 6.13　图的深度优先生成森林

6.3.3　图的遍历与图的连通性

图的遍历算法可以用来判断图的连通性。

对于无向图来说，若无向图是连通的，则从任意一个结点出发，仅需一次遍历就能够访问

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

图中的所有顶点；若无向图是非连通的，则从某一个顶点出发，一次遍历只能访问到该顶点所在连通分量的所有顶点，而对于图中其他连通分量的顶点，则无法通过这次遍历访问。对于有向图来说，若从初始点到图中的每个顶点都有路径，则能够访问到图中的所有顶点，否则不能访问到所有顶点。

故在 BFSTraverse() 或 DFSTraverse() 中添加了第二个 for 循环，再选取初始点，继续进行遍历，以防止一次无法遍历图的所有顶点。对于无向图，上述两个函数调用 BFS(G,i) 或 DFS(G,i) 的次数等于该图的连通分量数；而对于有向图则不是这样，因为一个连通的有向图分为强连通的和非强连通的，它的连通子图也分为强连通分量和非强连通分量，非强连通分量一次调用 BFS(G,i) 或 DFS(G,i) 无法访问到该连通分量的所有顶点，如图 6.14 所示。

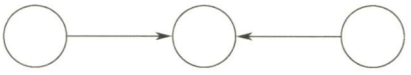


图 6.14 有向图的非强连通分量

关注公众号【乘龙考研】一手更新 稳定有保障

6.3.4 本节试题精选

一、单项选择题

01. 下列关于广度优先算法的说法中，正确的是（ ）。

I. 当各边的权值相等时，广度优先算法可以解决单源最短路径问题

II. 当各边的权值不等时，广度优先算法可用来解决单源最短路径问题

III. 广度优先遍历算法类似于树中的后序遍历算法

IV. 实现图的广度优先算法时，使用的数据结构是队列

A. I、IV　　B. II、III、IV　　C. II、IV　　D. I、III、IV

02. 对于一个非连通无向图 G，采用深度优先遍历访问所有顶点，在 DFSTraverse 函数（见考点讲解 DFS 部分）中调用 DFS 的次数正好等于（ ）。

A. 顶点数　　B. 边数　　C. 连通分量数　　D. 不确定

03. 对一个有 n 个顶点、e 条边的图采用邻接表表示时，进行 DFS 遍历的时间复杂度为（ ），空间复杂度为（ ）；进行 BFS 遍历的时间复杂度为（ ），空间复杂度为（ ）。

A. $O(n)$　　B. $O(e)$　　C. $O(n+e)$　　D. $O(1)$

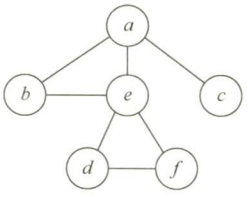
04. 对有 n 个顶点、e 条边的图采用邻接矩阵表示时，进行 DFS 遍历的时间复杂度为（ ），进行 BFS 遍历的时间复杂度为（ ）。

A. $O(n^2)$　　B. $O(e)$　　C. $O(n+e)$　　D. $O(e^2)$

05. 无向图 $G=(V,E)$，其中 $V=\{a,b,c,d,e,f\}$，$E=\{(a,b),(a,e),(a,c),(b,e),(c,f),(f,d),(e,d)\}$，对该图从 a 开始进行深度优先遍历，得到的顶点序列正确的是（ ）。

A. a,b,e,c,d,f　　B. a,c,f,e,b,d　　C. a,e,b,c,f,d　　D. a,e,d,f,c,b

06. 如下图所示，在下面的 5 个序列中，符合深度优先遍历的序列个数是（ ）。



1. $aebfdc$　2. $acfdeb$　3. $aedfcb$　4. $aefdbc$　5. $aecfdb$

A. 5　　B. 4　　C. 3　　D. 2

07. 用邻接表存储的图的深度优先遍历算法类似于树的（ ），而其广度优先遍历算法类似于

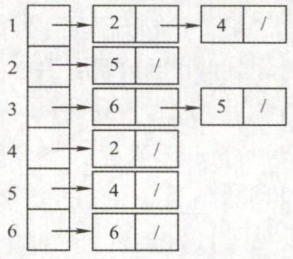
有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

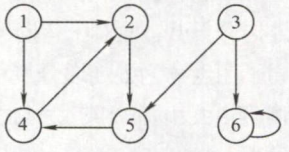
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

树的（ ）。

A. 中序遍历　　B. 先序遍历　　C. 后序遍历　　D. 按层次遍历

08. 一个有向图 G 的邻接表存储如下图所示，从顶点 1 出发，对图 G 调用深度优先遍历所得顶点序列是（ ）；按广度优先遍历所得顶点序列是（ ）。





A. 125436　　B. 124536　　C. 124563　　D. 362514

09. 无向图 $G=(V,E)$，其中 $V=\{a,b,c,d,e,f\}$，$E=\{(a,b),(a,e),(a,c),(b,e),(c,f),(f,d),(e,d)\}$。对该图进行深度优先遍历，不能得到的序列是（ ）。

A. *acfdeb*　　B. *aebdfc*　　C. *aedfcb*　　D. *abecdf*

10. 判断有向图中是否存在回路，除可以利用拓扑排序外，还可以利用（ ）。

A. 求关键路径的方法　　B. 求最短路径的 Dijkstra 算法

C. 深度优先遍历算法　　D. 广度优先遍历算法

11. 使用 DFS 算法递归地遍历一个无环有向图，并在退出递归时输出相应顶点，这样得到的顶点序列是（ ）。

A. 逆拓扑有序　　B. 拓扑有序　　C. 无序的　　D. 都不是

12. 设无向图 $G=(V,E)$和 $G'=(V',E')$，若 G'是 G 的生成树，则下列说法中错误的是（ ）。

A. G'为 G 的子图　　B. G'为 G 的连通分量

C. G'为 G 的极小连通子图且 $V=V'$　　D. G'是 G 的一个无环子图

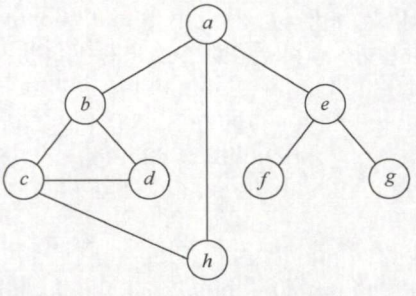
13. 图的广度优先生成树的树高比深度优先生成树的树高（ ）。

A. 小或相等　　B. 小　　C. 大或相等　　D. 大

14. 对有 n 个顶点、e 条边且使用邻接表存储的有向图进行广度优先遍历，其算法的时间复杂度是（ ）。

A. $O(n)$　　B. $O(e)$　　C. $O(n+e)$　　D. $O(ne)$

15. 【2013 统考真题】若对如下无向图进行遍历，则下列选项中，不是广度优先遍历序列的是（ ）。



A. h,c,a,b,d,e,g,f　　B. e,a,f,g,b,h,c,d

C. d,b,c,a,h,e,f,g　　D. a,b,c,d,h,e,f,g

16. 【2015 统考真题】设有向图 $G=(V,E)$，顶点集 $V=\{V_0,V_1,V_2,V_3\}$，边集 $E=\{<v_0,v_1>,<v_0,v_2>,$

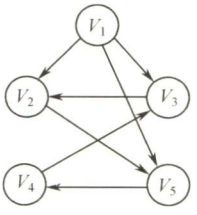
有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

$<v_0,v_3>$, $<v_1,v_3>$}。若从顶点 V_0 开始对图进行深度优先遍历，则可能得到的不同遍历序列个数是（　）。

A. 2　　B. 3　　C. 4　　D. 5

17. 【2016 统考真题】下列选项中，不是下图深度优先搜索序列的是（　）。



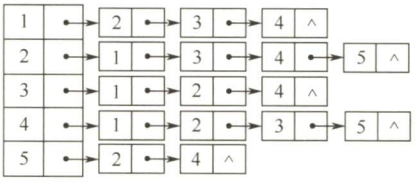
关注公众号【乘龙考研】
一手更新 稳定有保障

A. V_1, V_5, V_4, V_3, V_2　　B. V_1, V_3, V_2, V_5, V_4

C. V_1, V_2, V_5, V_4, V_3　　D. V_1, V_2, V_3, V_4, V_5

二、综合应用题

01. 图 $G=(V, E)$以邻接表存储，如下图所示，试画出图 G 的深度优先生成树和广度优先生成树（假设从结点 1 开始遍历）。



02. 试设计一个算法，判断一个无向图 G 是否为一棵树。若是一棵树，则算法返回 `true`，否则返回 `false`。

03. 写出图的深度优先搜索 DFS 算法的非递归算法（图采用邻接表形式）。

04. 分别采用基于深度优先遍历和广度优先遍历算法判别以邻接表方式存储的有向图中是否存在由顶点 v_i 到顶点 v_j 的路径（$i \neq j$）。注意，算法中涉及的图的基本操作必须在此存储结构上实现。

05. 假设图用邻接表表示，设计一个算法，输出从顶点 V_i 到顶点 V_j 的所有简单路径。

6.3.5　答案与解析

一、单项选择题

01. A

广度优先搜索以起始结点为中心，一层一层地向外层扩展遍历图的顶点，因此无法考虑到边权值，只适合求边权值相等的图的单源最短路径。广度优先搜索相当于树的层序遍历，选项 III 错误。广度优先搜索需要用到队列，深度优先搜索需要用到栈，选项 IV 正确。

02. C

DFS（或 BFS）可用来计算图的连通分量数，因为一次遍历必然会将一个连通图中的所有顶点都访问到，而对于已被访问的顶点将不再调用 DFS，故计算图的连通分量数正好是 `DFSTraverse()` 中 DFS 被调用的次数。

03. C、A、C、A

深度优先遍历时，每个顶点表结点和每个边表结点均查找一次，每个顶点递归调用一次，需要借助一个递归工作栈；而广度优先遍历时，也是每个顶点表结点和每个边表结点均查找一次，

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

每个顶点进入队列一次。故都是选 C, A。

04. A、A

采用邻接矩阵表示时，查找一个顶点所有出边的时间复杂度为 $O(n)$，共有 n 个顶点，故时间复杂度均为 $O(n^2)$。

05. D

画出草图后，此类题可以根据边的邻接关系快速排除错误选项。以 A 为例，在遍历到 e 之后，应该访问与 e 邻接但未被访问的结点，(e, c)显然不在边集中。

06. D

仅 1 和 4 正确。以 2 为例，遍历到 c 之后，与 c 邻接且未被访问的结点为空集，所以应为 a 的邻接点 b 或 e 入栈。以 3 为例，因为遍历要按栈退回，所以是先 b 后 c，而不能先 c 后 b。

07. B、D

图的深度优先搜索类似于树的先根遍历，即先访问结点，再递归向外层结点遍历，都采用回溯算法。图的广度优先搜索类似于树的层序遍历，即一层一层向外层扩展遍历，都需要采用队列来辅助算法的实现。

08. A、B

DFS 序列产生的路径为<1, 2>, <2, 5>, <5, 4>, <3, 6>; BFS 序列产生的路径为<1, 2>, <1, 4>, <2, 5>, <3, 6>。

09. D

画出 V 和 E 对应的图 G，然后根据搜索算法求解。

这里应注意：为什么本题序列是不唯一的，而上题却是唯一的呢？

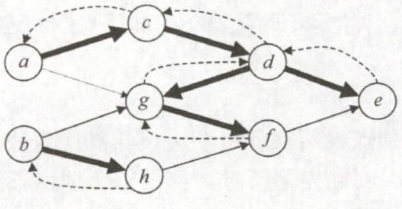
因为上题给出了具体的存储结构，此时就必须按照算法的过程来执行，每个顶点的邻接点的顺序已固定，但本题中每个顶点的邻接点的顺序是非固定的。

10. C

利用深度优先遍历可以判断图 G 中是否存在回路。对于无向图来说，若深度优先遍历过程中遇到了回边，则必定存在环；对于有向图来说，这条回边可能是指向深度优先森林中另一棵生成树上的顶点的弧；但是，从有向图的某个顶点 v 出发进行深度优先遍历时，若在 DFS(v)结束之前出现一条从顶点 u 到顶点 v 的回边，且 u 在生成树上是 v 的子孙，则有向图必定存在包含顶点 v 和顶点 u 的环。

11. A

对一个有向图做深度优先遍历，并未专门判断有向图是否有环（有向回路）存在，无论图中是否有环，都得到一个顶点序列。若无环，在退出递归过程中输出的应是逆拓扑有序序列。对有向无环图利用深度优先搜索进行拓扑排序的例子如下：如下图所示，退出 DFS 栈的顺序为 *efgdcahb*，此图的一个拓扑序列为 *bhacdgfe*。该方法的每一步均是先输出当前无后继的结点，即对每个结点 v，先递归地求出 v 的每个后继的拓扑序列。对于一个 AOV 网，按此方法输出的序列是一个逆拓扑序列。



有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

12．B

连通分量是无向图的极大连通子图，其中极大的含义是将依附于连通分量中顶点的所有边都加上，所以连通分量中可能存在回路，这样就不是生成树了。

注意：极大连通子图是无向图（不一定连通）的连通分量，极小连通子图是连通无向图的生成树。极小和极大是在满足连通的前提下，针对边的数目而言的。极大连通子图包含连通分量的全部边；极小连通子图（生成树）包含连通图的全部顶点，且使其连通的边数最少。

13．A

对于无向图的广度优先搜索生成树，起点到其他顶点的路径是图中对应的最短路径，即所有生成树中树高最小。此外，深度优先总是尽可能“深”地搜索图，因此其路径也尽可能长，故深度优先生成树的树高总是大于或等于广度优先生成树的树高。

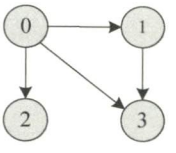
14．C

广度优先遍历需要借助队列实现。采用邻接表存储方式对图进行广度优先遍历时，每个顶点均需入队一次（顶点表遍历），故时间复杂度为 $O(n)$，在搜索所有顶点的邻接点的过程中，每条边至少访问一次（出边表遍历），所以时间复杂度为 $O(e)$，算法总的时间复杂度为 $O(n+e)$。

15．D

只要掌握 DFS 和 BFS 的遍历过程，便能轻易解决。逐个代入，手工模拟，选项 D 是深度优先遍历，而不是广度优先遍历。

16．D



画出该有向图的图形，如右图所示。采用图的深度优先遍历，共有 5 种可能：$<v_0, v_1, v_3, v_2>$，$<v_0, v_2, v_3, v_1>$，$<v_0, v_2, v_1, v_3>$，$<v_0, v_3, v_2, v_1>$，$<v_0, v_3, v_1, v_2>$，选 D。

17．D

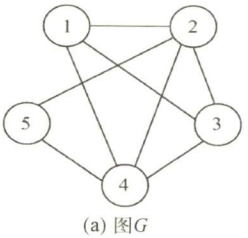
对于本题，只需按深度优先遍历的策略进行遍历。对于 A：先访问 V_1，然后访问与 V_1 邻接且未被访问的任意一个顶点（满足的有 V_2，V_3 和 V_5），此时访问 V_5，然后从 V_5 出发，访问与 V_5 邻接且未被访问的任意一个顶点（满足的只有 V_4），然后从 V_4 出发，访问与 V_4 邻接且未被访问的任意一个顶点（满足的只有 V_3），然后从 V_3 出发，访问与 V_3 邻接且未被访问的任意一个顶点（满足的只有 V_2），结束遍历。B 和 C 的分析方法与 A 相同，不再赘述。对于 D，首先访问 V_1，然后从 V_1 出发，访问与 V_1 邻接且未被访问的任意一个顶点（满足的有 V_2，V_3 和 V_5），然后从 V_2 出发，访问与 V_2 邻接且未被访问的任意一个顶点（满足的只有 V_5），按规则本应该访问 V_5，但 D 却访问了 V_3，错误。

关注公众号【乘龙考研】
一手更新 稳定有保障

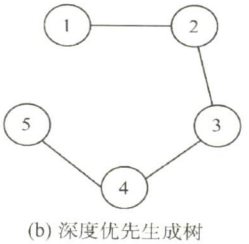
二、综合应用题

01．【解答】

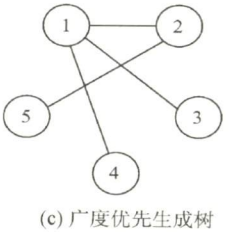
根据 G 的邻接表不难画出图(a)。



(a) 图G



(b) 深度优先生成树



(c) 广度优先生成树

1）采用深度优先遍历。深度优先搜索总是尽可能“深”地搜索图，根据存储结构可知深度

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

优先搜索的路径次序为(1, 2), (2, 3), (3, 4), (4, 5)，深度优先生成树如图(b)所示。需要注意的是，当存储结构固定时，生成树的树形也就固定了，比如不能先搜索(1, 3)。

2）采用广度优先遍历。广度优先搜索总是尽可能“广”地搜索图，一层一层地向外扩展，根据存储结构可知广度优先搜索的路径次序为(1, 2), (1, 3), (1, 4), (2, 5)，广度优先生成树如图(c)所示。

02.【解答】

一个无向图 G 是一棵树的条件是，G 必须是无回路的连通图或有 $n-1$ 条边的连通图。这里采用后者作为判断条件。对连通的判定，可以用能否一次遍历全部顶点来实现。可以采用深度优先搜索算法在遍历图的过程中统计可能访问到的顶点个数和边的条数，若一次遍历就能访问到 n 个顶点和 $n-1$ 条边，则可断定此图是一棵树。算法实现如下：

```
bool isTree(Graph& G){
    for(i=1;i<=G.vexnum;i++)
        visited[i]=FALSE;           //访问标记 visited[]初始化
    int Vnum=0,Enum=0;              //记录顶点数和边数
    DFS(G,1,Vnum,Enum,visited);
    if(Vnum==G.vexnum&&Enum==2*(G.vexnum-1))
        return  true;               //符合树的条件
    else
        return  false;              //不符合树的条件
}
void DFS(Graph& G,int v,int& Vnum,int& Enum,int visited[]){
//深度优先遍历图 G，统计访问过的顶点数和边数，通过 Vnum 和 Enum 返回
    visited[v]=TRUE;Vnum++;         //作访问标记，顶点计数
    int w=FirstNeighbor(G,v);       //取 v 的第一个邻接顶点
    while(w!=-1){                   //当邻接顶点存在
        Enum++;                     //边存在，边计数
        if(!visited[w])             //当该邻接顶点未访问过
            DFS(G,w,Vnum,Enum,visited);
        w=NextNeighbor(G,v,w);
    }
}
```

03.【解答】

在深度优先搜索的非递归算法中使用了一个栈 S 来记忆下一步可能访问的顶点，同时使用了一个访问标记数组 visited[i]来记忆第 i 个顶点是否在栈内或曾经在栈内，若是则它以后不能再进栈。图采用邻接表形式，算法的实现如下：

```
void DFS_Non_RC(AGraph& G,int v){
//从顶点 v 开始进行深度优先搜索，一次遍历一个连通分量的所有顶点
    int w;                          //顶点序号
    InitStack(S);                   //初始化栈 S
    for(i=0;i<G.vexnum;i++)
        visited[i]=FALSE;           //初始化 visited
    Push(S,v);visited[v]=TRUE;      //v 入栈并置 visited[v]
    while(!IsEmpty(S)){
        k=Pop(S);                   //栈中退出一个顶点
        visit(k);                   //先访问，再将其子结点入栈
        for(w=FirstNeighbor(G,k);w>=0;w=NextNeighor(G,k,w))
                                    //k 所有邻接点
            if(!visited[w]){        //未进过栈的顶点进栈
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
                Push(S,w);
                visited[w]=true;     //作标记，以免再次入栈
            }//if
        }//while
    }//DFS_Non_RC
```

注意：由于使用了栈，使得遍历的方式从右端到左端进行，不同于常规的从左端到右端，但仍然是深度优先遍历，读者可以用实例模拟验证。

04.【解答】

两个不同的遍历算法都采用从顶点 v_i 出发，依次遍历图中每个顶点，直到搜索到顶点 v_j，若能够搜索到 v_j，则说明存在由顶点 v_i 到顶点 v_j 的路径。

深度优先遍历算法的实现如下：

```
int visited[MAXSIZE]={0};          //访问标记数组
void DFS(ALGraph G,int i,int j,bool &can_reach){
//深度优先判断有向图 G 中顶点 vi 到顶点 vj 是否有路径,用 can_reach 来标识
    if(i==j){
        can_reach=true;
        return;                    //i 就是 j
    }
    visited[i]=1;                  //置访问标记
    for(int p=FirstNeighbor(G,i);p>=0;p=NextNeighbor(G,i,p))
        if(!visited[p]&&!can_reach)     //递归检测邻接点
            DFS(G,p,j,can_reach);
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

广度优先遍历算法的实现如下：

```
int visited[MAXSIZE]={0};          //访问标记数组
int BFS(ALGraph G,int i,int j){
//广度优先判断有向图 G 中顶点 vi 到顶点 vj 是否有路径，是则返回 1，否则返回 0
    InitQueue(Q); EnQueue(Q,i);    //顶点 i 入队
    while(!isEmpty(Q)){            //非空循环
        DeQueue(Q,u);              //队头顶点出队
        visited[u]=1;              //置访问标记
        if(u==j)  return 1;
        for(int p=FirstNeighbor(G,u);p;p=NextNeighbor(G,u,p)){
                                   //检查所有邻接点
            if(p==j)               //若 p==j，则查找成功
                return 1;
            if(!visited[p]){       //否则，顶点 p 入队
                EnQueue(Q,p);
                visited[p]=1;
            }
        }//for
    }//while
  return  0;
}
```

本题也可以这样解答：调用以 i 为参数的 DFS(G,i) 或 BFS(G,i)，执行结束后判断 visited[j] 是否为 TRUE，若是则说明 v_j 已被遍历，图中必存在由 v_i 到 v_j 的路径。但此种解法每次都耗费最坏时间复杂度对应的时间，需要遍历与 v_i 连通的所有顶点。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

05.【解答】

本题采用基于递归的深度优先遍历算法，从结点 u 出发，递归深度优先遍历图中结点，若访问到结点 v，则输出该搜索路径上的结点。为此，设置一个 path 数组来存放路径上的结点（初始为空），d 表示路径长度（初始为-1）。查找从顶点 u 到 v 的简单路径过程说明如下（假设查找函数名为 FindPath()）：

1）FindPath(G,u,v,path,d):d++;path[d]=u；若找到 u 的未访问过的相邻结点 u_1，则继续下去，否则置 visited[u]=0 并返回。

2）FindPath(G,u1,v,path,d):d++;path[d]=u_1；若找到 u_1 的未访问过的相邻结点 u_2，则继续下去，否则置 visited[u_1]=0。

3）以此类推，继续上述递归过程，直到 u_i=v，输出 path。

算法实现如下：

```
void FindPath(AGraph *G,int u,int v,int path[],int d){
    int w;
    ArcNode *p;
    d++;                          //路径长度增 1
    path[d]=u;                    //将当前顶点添加到路径中
    visited[u]=1;                 //置已访问标记
    if(u==v)                      //找到一条路径则输出
        print(path[]);            //输出路径上的结点
    p=G->adjlist[u]. firstarc;    //p 指向 u 的第一个相邻点
    while(p!=NULL){
        w=p->adjvex;              //若顶点 w 未访问，递归访问它
        if(visited[w]==0)
            FindPath(G,w,v,path,d);
        p=p->nextarc;             //p 指向 u 的下一个相邻点
    }
    visited[u]=0;                 //恢复环境，使该顶点可重新使用
}
```

6.4 图的应用

本节是历年考查的重点。图的应用主要包括：最小生成（代价）树、最短路径、拓扑排序和关键路径。一般而言，这部分内容直接以算法设计题形式考查的可能性很小，而更多的是结合图的实例来考查算法的具体操作过程，读者必须学会手工模拟给定图的各个算法的执行过程。此外，还需掌握对给定模型建立相应的图去解决问题的方法。

6.4.1 最小生成树

一个连通图的生成树包含图的所有顶点，并且只含尽可能少的边。对于生成树来说，若砍去它的一条边，则会使生成树变成非连通图；若给它增加一条边，则会形成图中的一条回路。

对于一个带权连通无向图 $G=(V, E)$，生成树不同，每棵树的权（即树中所有边上的权值之和）也可能不同。设$\mathfrak{R}$为 G 的所有生成树的集合，若 T 为$\mathfrak{R}$中边的权值之和最小的那棵生成树，则 T 称为 G 的最小生成树（Minimum-Spanning-Tree, MST）。

不难看出，最小生成树具有如下性质：

1）最小生成树不是唯一的，即最小生成树的树形不唯一，$\mathfrak{R}$中可能有多个最小生成树。当图

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

G 中的各边权值互不相等时，G 的最小生成树是唯一的；若无向连通图 G 的边数比顶点数少 1，即 G 本身是一棵树时，则 G 的最小生成树就是它本身。

2）最小生成树的边的权值之和总是唯一的，虽然最小生成树不唯一，但其对应的边的权值之和总是唯一的，而且是最小的。

3）最小生成树的边数为顶点数减 1。

构造最小生成树有多种算法，但大多数算法都利用了最小生成树的下列性质：假设 $G=(V, E)$ 是一个带权连通无向图，U 是顶点集 V 的一个非空子集。若(u, v)是一条具有最小权值的边，其中 $u\in U$，$v\in V-U$，则必存在一棵包含边(u, v)的最小生成树。

基于该性质的最小生成树算法主要有 Prim 算法和 Kruskal 算法，它们都基于贪心算法的策略。对这两种算法应主要掌握算法的本质含义和基本思想，并能够手工模拟算法的实现步骤。

下面介绍一个通用的最小生成树算法：

```
GENERIC_MST(G){
    T=NULL;
    while T 未形成一棵生成树;
        do 找到一条最小代价边(u,v)并且加入T后不会产生回路;
            T=T∪(u,v);
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

通用算法每次加入一条边以逐渐形成一棵生成树，下面介绍两种实现上述通用算法的途径。

1．Prim 算法

Prim（普里姆）算法的执行非常类似于寻找图的最短路径的 Dijkstra 算法（见下一节）。

Prim 算法构造最小生成树的过程如图 6.15 所示。初始时从图中任取一顶点（如顶点 1）加入树 T，此时树中只含有一个顶点，之后选择一个与当前 T 中顶点集合距离最近的顶点，并将该顶点和相应的边加入 T，每次操作后 T 中的顶点数和边数都增 1。以此类推，直至图中所有的顶点都并入 T，得到的 T 就是最小生成树。此时 T 中必然有 $n-1$ 条边。

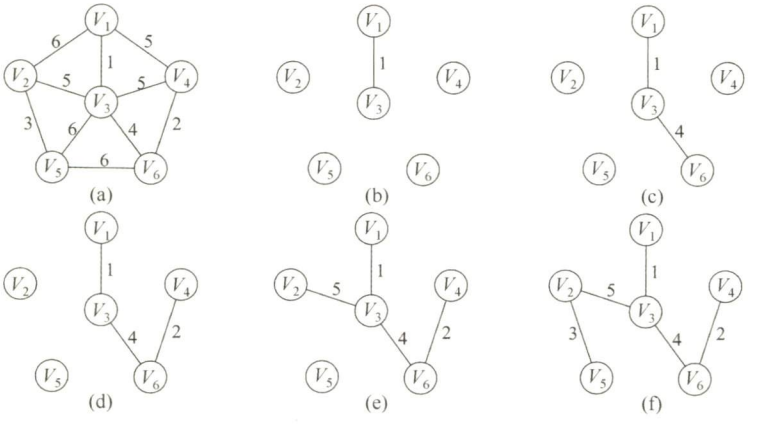


图 6.15　Prim 算法构造最小生成树的过程

Prim 算法的步骤如下：

假设 $G=\{V, E\}$是连通图，其最小生成树 $T=(U, E_T)$，E_T是最小生成树中边的集合。

初始化：向空树 $T=(U, E_T)$中添加图 $G=(V, E)$的任意一个顶点 u_0，使 $U=\{u_0\}$，$E_T=\varnothing$。

循环（重复下列操作直至 $U=V$）：从图 G 中选择满足$\{(u, v)|u\in U, v\in V-U\}$且具有最小权值的边$(u, v)$，加入树 T，置 $U=U\cup\{v\}$，$E_T=E_T\cup\{(u, v)\}$。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

Prim 算法的简单实现如下：

```
void Prim(G,T){
    T=∅;                        //初始化空树
    U={w};                      //添加任意一个顶点 w
    while((V-U)!=∅){            //若树中不含全部顶点
        设(u,v)是使 u∈U 与 v∈(V-U),且权值最小的边;
        T=T∪{(u,v)};            //边归入树
        U=U∪{v};                //顶点归入树
    }
}
```

Prim 算法的时间复杂度为 $O(|V|^2)$，不依赖于$|E|$，因此它适用于求解边稠密的图的最小生成树。虽然采用其他方法能改进 Prim 算法的时间复杂度，但增加了实现的复杂性。

2．Kruskal 算法

与 Prim 算法从顶点开始扩展最小生成树不同，Kruskal（克鲁斯卡尔）算法是一种按权值的递增次序选择合适的边来构造最小生成树的方法。

Kruskal 算法构造最小生成树的过程如图 6.16 所示。初始时为只有 n 个顶点而无边的非连通图 $T=\{V, \{\}\}$，每个顶点自成一个连通分量，然后按照边的权值由小到大的顺序，不断选取当前未被选取过且权值最小的边，若该边依附的顶点落在 T 中不同的连通分量上，则将此边加入 T，否则舍弃此边而选择下一条权值最小的边。以此类推，直至 T 中所有顶点都在一个连通分量上。

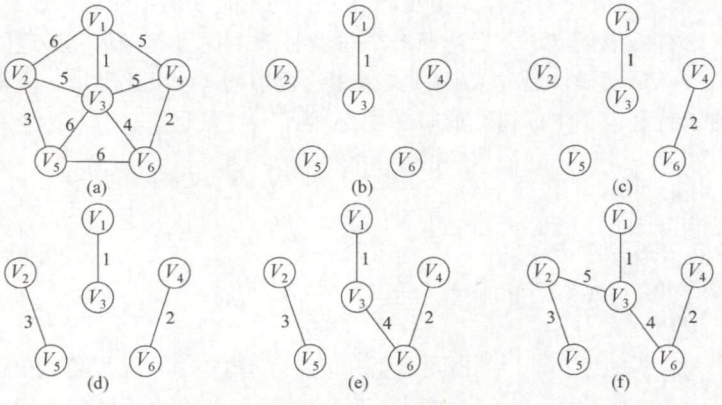


图 6.16 Kruskal 算法构造最小生成树的过程

Kruskal 算法的步骤如下：

假设 $G=(V, E)$是连通图，其最小生成树 $T=(U, E_T)$。

初始化：$U=V, E_T=\varnothing$。即每个顶点构成一棵独立的树，T 此时是一个仅含$|V|$个顶点的森林。

循环（重复直至 T 是一棵树）：按 G 的边的权值递增顺序依次从 $E-E_T$ 中选择一条边，若这条边加入 T 后不构成回路，则将其加入 E_T，否则舍弃，直到 E_T 中含有 $n-1$ 条边。

Kruskal 算法的简单实现如下：

```
void Kruskal(V,T){
    T=V;                        //初始化树 T，仅含顶点
    numS=n;                     //连通分量数
    while(numS>1){              //若连通分量数大于 1
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
            从 E 中取出权值最小的边(v,u);
            if(v 和 u 属于 T 中不同的连通分量){
            T=T∪{(v,u)};          //将此边加入生成树中
            numS--;               //连通分量数减 1
            }
        }
    }
```

关注公众号【乘龙考研】
一手更新 稳定有保障

根据图的相关性质，若一条边连接了两棵不同树中的顶点，则对这两棵树来说，它必定是连通的，将这条边加入森林中，完成两棵树的合并，直到整个森林合并成一棵树。

通常在 Kruskal 算法中，采用堆（见第 7 章）来存放边的集合，因此每次选择最小权值的边只需 $O(\log_2|E|)$的时间。此外，由于生成树 T 中的所有边可视为一个等价类，因此每次添加新的边的过程类似于求解等价类的过程，由此可以采用并查集的数据结构来描述 T，从而构造 T 的时间复杂度为 $O(|E|\log_2|E|)$。因此，Kruskal 算法适合于边稀疏而顶点较多的图。

6.4.2 最短路径

6.3 节所述的广度优先搜索查找最短路径只是对无权图而言的。当图是带权图时，把从一个顶点 v_0 到图中其余任意一个顶点 v_i 的一条路径（可能不止一条）所经过边上的权值之和，定义为该路径的带权路径长度，把带权路径长度最短的那条路径称为*最短路径*。

求解最短路径的算法通常都依赖于一种性质，即两点之间的最短路径也包含了路径上其他顶点间的最短路径。带权有向图 G 的最短路径问题一般可分为两类：一是单源最短路径，即求图中某一顶点到其他各顶点的最短路径，可通过经典的 Dijkstra（迪杰斯特拉）算法求解；二是求每对顶点间的最短路径，可通过 Floyd（弗洛伊德）算法来求解。

1. Dijkstra 算法求单源最短路径问题

Dijkstra 算法设置一个集合 S 记录已求得的最短路径的顶点，初始时把源点 v_0 放入 S，集合 S 每并入一个新顶点 v_i，都要修改源点 v_0 到集合 $V-S$ 中顶点当前的最短路径长度值（这里可能不太好理解？没关系，等下就会清楚）。

在构造的过程中还设置了两个辅助数组：

- `dist[]`：记录从源点 v_0 到其他各顶点当前的最短路径长度，它的初态为：若从 v_0 到 v_i 有弧，则 `dist[i]`为弧上的权值；否则置 `dist[i]`为∞。
- `path[]`：`path[i]`表示从源点到顶点 i 之间的最短路径的前驱结点。在算法结束时，可根据其值追溯得到源点 v_0 到顶点 v_i 的最短路径。

假设从顶点 0 出发，即 $v_0=0$，集合 S 最初只包含顶点 0，邻接矩阵 `arcs` 表示带权有向图，`arcs[i][j]`表示有向边`<i,j>`的权值，若不存在有向边`<i,j>`，则 `arcs[i][j]`为∞。

Dijkstra 算法的步骤如下（不考虑对 path[]的操作）：

1）初始化：集合 S 初始为{0}，`dist[]`的初始值 `dist[i]=arcs[0][i], i=1,2,…,n-1`。

2）从顶点集合 $V-S$ 中选出 v_j，满足 `dist[j]=Min{dist[i]|vi∈V-S}`，v_j 就是当前求得的一条从 v_0 出发的最短路径的终点，令 $S=S\cup\{j\}$。

3）修改从 v_0 出发到集合 $V-S$ 上任意一个顶点 v_k 可达的最短路径长度：若 `dist[j]+arcs[j][k]<dist[k]`，则更新 `dist[k]=dist[j]+arcs[j][k]`。

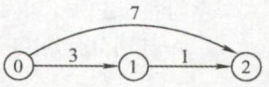
4）重复 2）～3）操作共 $n-1$ 次，直到所有的顶点都包含在 S 中。

步骤 3）也就是开头留下的疑问，每当一个顶点加入 S 后，可能需要修改源点 v_0 到集合 $V-S$

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

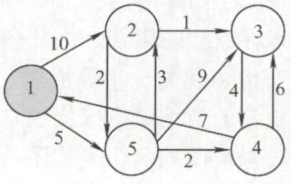
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

中可达顶点当前的最短路径长度，下面举一简单例子证明。如下图所示，源点为 v_0，初始时 $S=\{v_0\}$，dist[1]=3，dist[2]=7，当将 v_1 并入集合 S 后，dist[2]需要更新为 4。



思考：Dijkstra 算法与 Prim 算法有何相似之处？

例如，对图 6.17 中的图应用 Dijkstra 算法求从顶点 1 出发至其余顶点的最短路径的过程，如表 6.1 所示。算法执行过程的说明如下。



每轮得到的最短路径如下：
第1轮：1→5，路径距离为5
第2轮：1→5→4，路径距离为7
第3轮：1→5→2，路径距离为8
第4轮：1→5→2→3，路径距离为9

图 6.17 应用 Dijkstra 算法图

表 6.1 从 v_1 到各终点的 dist 值和最短路径的求解过程

顶点	第 1 轮	第 2 轮	第 3 轮	第 4 轮
2	10 $v_1\to v_2$	8 $v_1\to v_5\to v_2$	**8** $v_1\to v_5\to v_2$	
3	∞	14 $v_1\to v_5\to v_3$	13 $v_1\to v_5\to v_4\to v_3$	**9** $v_1\to v_5\to v_2\to v_3$
4	∞	**7** $v_1\to v_5\to v_4$		
5	**5** $v_1\to v_5$			
集合 S	**{1, 5}**	**{1, 5, 4}**	**{1, 5, 4, 2}**	**{1, 5, 4, 2, 3}**

初始化：集合 S 初始为 $\{v_1\}$，v_1 可达 v_2 和 v_5，v_1 不可达 v_3 和 v_4，因此 dist[]数组各元素的初值依次设置为 dist[2]=10，dist[3]=∞，dist[4]=∞，dist[5]=5。

第一轮：选出最小值 dist[5]，将顶点 v_5 并入集合 S，即此时已找到 v_1 到 v_5 的最短路径。当 v_5 加入 S 后，从 v_1 到集合 $V-S$ 中可达顶点的最短路径长度可能会产生变化。因此需要更新 dist[]数组。v_5 可达 v_2，因 $v_1\to v_5\to v_2$ 的距离 8 比 dist[2]=10 小，更新 dist[2]=8；v_5 可达 v_3，$v_1\to v_5\to v_3$ 的距离 14，更新 dist[3]=14；v_5 可达 v_4，$v_1\to v_5\to v_4$ 的距离 7，更新 dist[4]=7。

第二轮：选出最小值 dist[4]，将顶点 v_4 并入集合 S。继续更新 dist[]数组。v_4 不可达 v_2，dist[2]不变；v_4 可达 v_3，$v_1\to v_5\to v_4\to v_3$ 的距离 13 比 dist[3]小，故更新 dist[3]=13。

第三轮：选出最小值 dist[2]，将顶点 v_2 并入集合 S。继续更新 dist[]数组。v_2 可达 v_3，$v_1\to v_5\to v_2\to v_3$ 的距离 9 比 dist[3]小，更新 dist[3]=9。

第四轮：选出唯一最小值 dist[3]，将顶点 v_3 并入集合 S，此时全部顶点都已包含在 S 中。

显然，Dijkstra 算法也是基于贪心策略的。

使用邻接矩阵表示时，时间复杂度为 $O(|V|^2)$。使用带权的邻接表表示时，虽然修改 dist[]的时间可以减少，但由于在 dist[]中选择最小分量的时间不变，时间复杂度仍为 $O(|V|^2)$。

人们可能只希望找到从源点到某个特定顶点的最短路径，但这个问题和求解源点到其他所有顶点的最短路径一样复杂，时间复杂度也为 $O(|V|^2)$。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

值得注意的是，边上带有负权值时，Dijkstra 算法并不适用。若允许边上带有负权值，则在与 S（已求得最短路径的顶点集，归入 S 内的结点的最短路径不再变更）内某点（记为 a）以负边相连的点（记为 b）确定其最短路径时，其最短路径长度加上这条负边的权值结果可能小于 a 原先确定的最短路径长度，而此时 a 在 Dijkstra 算法下是无法更新的。例如，对于图 6.18 所示的带权有向图，利用 Dijkstra 算法不一定能得到正确的结果。

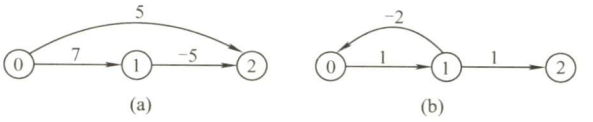


图 6.18 边上带有负权值的有向带权图

2．Floyd 算法求各顶点之间最短路径问题

求所有顶点之间的最短路径问题描述如下：已知一个各边权值均大于 0 的带权有向图，对任意两个顶点 $v_i \neq v_j$，要求求出 v_i 与 v_j 之间的最短路径和最短路径长度。

Floyd 算法的基本思想是：递推产生一个 n 阶方阵序列 $A^{(-1)}, A^{(0)}, \cdots, A^{(k)}, \cdots, A^{(n-1)}$，其中 $A^{(k)}[i][j]$ 表示从顶点 v_i 到顶点 v_j 的路径长度，k 表示绕行第 k 个顶点的运算步骤。初始时，对于任意两个顶点 v_i 和 v_j，若它们之间存在边，则以此边上的权值作为它们之间的最短路径长度；若它们之间不存在有向边，则以∞作为它们之间的最短路径长度。以后逐步尝试在原路径中加入顶点 k（$k=0, 1, \cdots, n-1$）作为中间顶点。若增加中间顶点后，得到的路径比原来的路径长度减少了，则以此新路径代替原路径。算法描述如下：

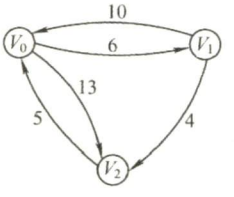
定义一个 n 阶方阵序列 $A^{(-1)}, A^{(0)}, \cdots, A^{(n-1)}$，其中，

$$A^{(-1)}[i][j] = \text{arcs}[i][j]$$

$$A^{(k)}[i][j] = \text{Min}\{A^{(k-1)}[i][j],\ A^{(k-1)}[i][k] + A^{(k-1)}[k][j]\},\quad k = 0, 1, \cdots, n-1$$

式中，$A^{(0)}[i][j]$是从顶点 v_i 到 v_j、中间顶点是 v_0 的最短路径的长度，$A^{(k)}[i][j]$是从顶点 v_i 到 v_j、中间顶点的序号不大于 k 的最短路径的长度。Floyd 算法是一个迭代的过程，每迭代一次，在从 v_i 到 v_j 的最短路径上就多考虑了一个顶点；经过 n 次迭代后，所得到的 $A^{(n-1)}[i][j]$就是 v_i 到 v_j 的最短路径长度，即方阵 $A^{(n-1)}$中就保存了任意一对顶点之间的最短路径长度。

图 6.19 所示为带权有向图 G 及其邻接矩阵。应用 Floyd 算法求所有顶点之间的最短路径长度的过程如表 6.2 所示。算法执行过程的说明如下。



(a) 有向图G

$$\begin{bmatrix} 0 & 6 & 13 \\ 10 & 0 & 4 \\ 5 & \infty & 0 \end{bmatrix}$$

(b) G的邻接矩阵

图 6.19 带权有向图 G 及其邻接矩阵

关注公众号【乘龙考研】
一手更新 稳定有保障

初始化：方阵 $A^{(-1)}[i][j] = \text{arcs}[i][j]$。

第一轮：将 v_0 作为中间顶点，对于所有顶点对$\{i, j\}$，如果有 $A^{-1}[i][j] > A^{-1}[i][0] + A^{-1}[0][j]$，则将 $A^{-1}[i][j]$更新为 $A^{-1}[i][0] + A^{-1}[0][j]$。有 $A^{-1}[2][1] > A^{-1}[2][0] + A^{-1}[0][1] = 11$，更新 $A^{-1}[2][1] = 11$，更新后的方阵标记为 A^0。

第二轮：将 v_1 作为中间顶点，继续检测全部顶点对$\{i, j\}$。有 $A^0[0][2] > A^0[0][1] + A^0[1][2] = 10$，更新 $A^0[0][2] = 10$，更新后的方阵标记为 A^1。

第三轮：将 v_2 作为中间顶点，继续检测全部顶点对$\{i, j\}$。有 $A^1[1][0] > A^1[1][2] + A^1[2][0] = 9$，

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

更新 $A^1[1][0]=9$，更新后的方阵标记为 A^2。此时 A^2 中保存的就是任意顶点对的最短路径长度。

表 6.2 Floyd 算法的执行过程

A	$A^{(-1)}$			$A^{(0)}$			$A^{(1)}$			$A^{(2)}$		
	V_0	V_1	V_2	V_0	V_1	V_2	V_0	V_1	V_2	V_0	V_1	V_2
V_0	0	6	13	0	6	13	0	6	**10**	0	6	10
V_1	10	0	4	10	0	4	10	0	4	**9**	0	4
V_2	5	∞	0	5	**11**	0	5	11	0	5	11	0

Floyd 算法的时间复杂度为 $O(|V|^3)$。不过由于其代码很紧凑，且并不包含其他复杂的数据结构，因此隐含的常数系数是很小的，即使对于中等规模的输入来说，它仍然是相当有效的。

Floyd 算法允许图中有带负权值的边，但不允许有包含带负权值的边组成的回路。Floyd 算法同样适用于带权无向图，因为带权无向图可视为权值相同往返二重边的有向图。

也可以用单源最短路径算法来解决每对顶点之间的最短路径问题。轮流将每个顶点作为源点，并且在所有边权值均非负时，运行一次 Dijkstra 算法，其时间复杂度为 $O(|V|^2)\cdot|V|=O(|V|^3)$。

6.4.3 有向无环图描述表达式

有向无环图：若一个有向图中不存在环，则称为有向无环图，简称 DAG 图。

有向无环图是描述含有公共子式的表达式的有效工具。例如表达式

$$((a+b)*(b*(c+d))+(c+d)*e)*((c+d)*e)$$

可以用上一章描述的二叉树来表示，如图 6.20 所示。仔细观察该表达式，可发现有一些相同的子表达式$(c+d)$和$(c+d)*e$，而在二叉树中，它们也重复出现。若利用有向无环图，则可实现对相同子式的共享，从而节省存储空间，图 6.21 所示为该表达式的有向无环图表示。

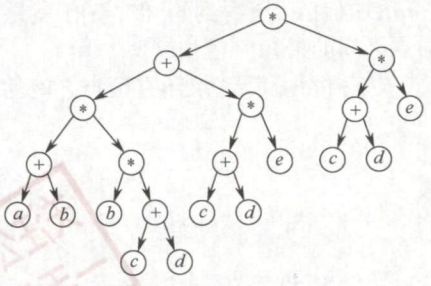


图 6.20 二叉树描述表达式

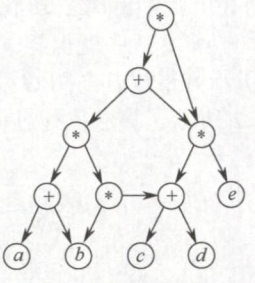


图 6.21 有向无环图描述表达式

6.4.4 拓扑排序

AOV 网：若用 DAG 图表示一个工程，其顶点表示活动，用有向边$<V_i, V_j>$表示活动 V_i 必须先于活动 V_j 进行的这样一种关系，则将这种有向图称为顶点表示活动的网络，记为 AOV 网。在 AOV 网中，活动 V_i 是活动 V_j 的直接前驱，活动 V_j 是活动 V_i 的直接后继，这种前驱和后继关系具有传递性，且任何活动 V_i 不能以它自己作为自己的前驱或后继。

拓扑排序：在图论中，由一个有向无环图的顶点组成的序列，当且仅当满足下列条件时，称为该图的一个拓扑排序：

① 每个顶点出现且只出现一次。

② 若顶点 A 在序列中排在顶点 B 的前面，则在图中不存在从顶点 B 到顶点 A 的路径。

或定义为：拓扑排序是对有向无环图的顶点的一种排序，它使得若存在一条从顶点 A 到顶点

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

B 的路径，则在排序中顶点 *B* 出现在顶点 *A* 的后面。每个 AOV 网都有一个或多个拓扑排序序列。

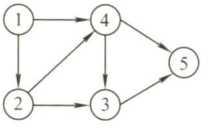
对一个 AOV 网进行拓扑排序的算法有很多，下面介绍比较常用的一种方法的步骤：

① 从 AOV 网中选择一个没有前驱的顶点并输出。

② 从网中删除该顶点和所有以它为起点的有向边。

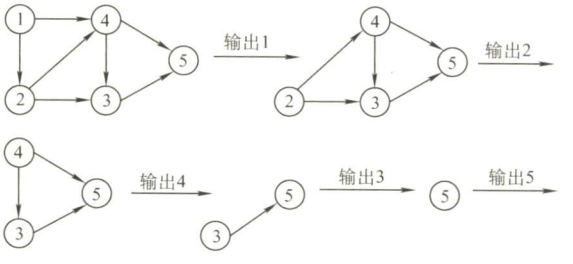
③ 重复①和②直到当前的 AOV 网为空或当前网中不存在无前驱的顶点为止。后一种情况说明有向图中必然存在环。

图 6.22 所示为拓扑排序过程的示例。每轮选择一个入度为 0 的顶点并输出，然后删除该顶点和所有以它为起点的有向边，最后得到拓扑排序的结果为{1, 2, 4, 3, 5}。



结点号	1	2	3	4	5
初始入度	0	1	2	2	2
第一轮		0	2	1	2
第二轮			1	0	2
第三轮			0		1
第四轮					0
第五轮					

(a)



(b)

图 6.22 有向无环图的拓扑排序过程

关注公众号【乘龙考研】一手更新 稳定有保障

拓扑排序算法的实现如下：

```
bool TopologicalSort(Graph G){
    InitStack(S);                   //初始化栈，存储入度为 0 的顶点
    int i;
    for(i=0;i<G.vexnum;i++)
        if(indegree[i]==0)
            Push(S,i);              //将所有入度为 0 的顶点进栈
    int count=0;                    //计数，记录当前已经输出的顶点数
    while(!IsEmpty(S)){             //栈不空，则存在入度为 0 的顶点
        Pop(S,i);                   //栈顶元素出栈
        print[count++]=i;           //输出顶点 i
        for(p=G.vertices[i].firstarc;p;p=p->nextarc){
        //将所有 i 指向的顶点的入度减 1，并且将入度减为 0 的顶点压入栈 S
            v=p->adjvex;
            if(!(--indegree[v]))
                Push(S,v);          //入度为 0，则入栈
        }
    }//while
    if(count<G.vexnum)
        return  false;              //排序失败，有向图中有回路
    else
```

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        return  true;            //拓扑排序成功
    }
```

由于输出每个顶点的同时还要删除以它为起点的边，故采用邻接表存储时拓扑排序的时间复杂度为 $O(|V| + |E|)$，采用邻接矩阵存储时拓扑排序的时间复杂度为 $O(|V|^2)$。此外，利用上一节的深度优先遍历也可实现拓扑排序，请读者仔细思考其原因及实现方法（见综合应用题6）。

对一个AOV网，如果采用下列步骤进行排序，则称之为*逆拓扑排序*：

① 从AOV网中选择一个没有后继（出度为0）的顶点并输出。

② 从网中删除该顶点和所有以它为终点的有向边。

③ 重复①和②直到当前的AOV网为空。

用拓扑排序算法处理AOV网时，应注意以下问题：

① 入度为零的顶点，即没有前驱活动的或前驱活动都已经完成的顶点，工程可以从这个顶点所代表的活动开始或继续。

② 若一个顶点有多个直接后继，则拓扑排序的结果通常不唯一；但若各个顶点已经排在一个线性有序的序列中，每个顶点有唯一的前驱后继关系，则拓扑排序的结果是唯一的。

③ 由于AOV网中各顶点的地位平等，每个顶点编号是人为的，因此可以按拓扑排序的结果重新编号，生成AOV网的新的邻接存储矩阵，这种邻接矩阵可以是三角矩阵；但对于一般的图来说，若其邻接矩阵是三角矩阵，则存在拓扑序列；反之则不一定成立。

6.4.5 关键路径

在带权有向图中，以顶点表示事件，以有向边表示活动，以边上的权值表示完成该活动的开销（如完成活动所需的时间），称之为*用边表示活动的网络*，简称AOE网。AOE网和AOV网都是有向无环图，不同之处在于它们的边和顶点所代表的含义是不同的，AOE网中的边有权值；而AOV网中的边无权值，仅表示顶点之间的前后关系。

AOE网具有以下两个性质：

① *只有在某顶点所代表的事件发生后，从该顶点出发的各有向边所代表的活动才能开始；*

② *只有在进入某顶点的各有向边所代表的活动都已结束时，该顶点所代表的事件才能发生。*

在AOE网中仅有一个入度为0的顶点，称为*开始顶点*（源点），它表示整个工程的开始；网中也仅存在一个出度为0的顶点，称为*结束顶点*（汇点），它表示整个工程的结束。

在AOE网中，有些活动是可以并行进行的。从源点到汇点的有向路径可能有多条，并且这些路径长度可能不同。完成不同路径上的活动所需的时间虽然不同，但是只有所有路径上的活动都已完成，整个工程才能算结束。因此，从源点到汇点的所有路径中，具有最大路径长度的路径称为*关键路径*，而把关键路径上的活动称为*关键活动*。

完成整个工程的最短时间就是关键路径的长度，即关键路径上各活动花费开销的总和。这是因为关键活动影响了整个工程的时间，即若关键活动不能按时完成，则整个工程的完成时间就会延长。因此，只要找到了关键活动，就找到了关键路径，也就可以得出最短完成时间。

下面给出在寻找关键活动时所用到的几个参量的定义。

1. 事件 v_k 的最早发生时间 $ve(k)$

它是指从源点 v_1 到顶点 v_k 的最长路径长度。事件 v_k 的最早发生时间决定了所有从 v_k 开始的活动能够开工的最早时间。可用下面的递推公式来计算：

$ve(\text{源点}) = 0$

$ve(k) = \text{Max}\{ve(j) + \text{Weight}(v_j, v_k)\}$，$v_k$ 为 v_j 的任意后继，Weight (v_j, v_k)表示$<v_j, v_k>$上的权值

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

计算 $ve()$值时，按从前往后的顺序进行，可以在拓扑排序的基础上计算：

① 初始时，令 $ve[1...n]=0$。

② 输出一个入度为 0 的顶点 v_j 时，计算它所有直接后继顶点 v_k 的最早发生时间，若 $ve[j]+\text{Weight}(v_j, v_k)>ve[k]$，则 $ve[k]=ve[j]+\text{Weight}(v_j, v_k)$。以此类推，直至输出全部顶点。

2．事件 v_k 的最迟发生时间 $vl(k)$

它是指在不推迟整个工程完成的前提下，即保证它的后继事件 v_j 在其最迟发生时间 $vl(j)$能够发生时，该事件最迟必须发生的时间。可用下面的递推公式来计算：

$vl(\text{汇点})=ve(\text{汇点})$

$vl(k)=\text{Min}\{vl(j)-\text{Weight}(v_k, v_j)\}$，$v_k$ 为 v_j 的任意前驱

关注公众号【乘龙考研】
一手更新 稳定有保障

注意：在计算 $vl(k)$时，按从后往前的顺序进行，可以在逆拓扑排序的基础上计算。

计算 $vl()$值时，按从后往前的顺序进行，在上述拓扑排序中，增设一个栈以记录拓扑序列，拓扑排序结束后从栈顶至栈底便为逆拓扑有序序列。过程如下：

① 初始时，令 $vl[1...n]=ve[n]$。

② 栈顶顶点 v_j 出栈，计算其所有直接前驱顶点 v_k 的最迟发生时间，若 $vl[j]-\text{Weight}(v_k, v_j)<vl[k]$，则 $vl[k]=vl[j]-\text{Weight}(v_k, v_j)$。以此类推，直至输出全部栈中顶点。

3．活动 a_i 的最早开始时间 $e(i)$

它是指该活动弧的起点所表示的事件的最早发生时间。若边$<v_k, v_j>$表示活动 a_i，则有 $e(i)=ve(k)$。

4．活动 a_i 的最迟开始时间 $l(i)$

它是指该活动弧的终点所表示事件的最迟发生时间与该活动所需时间之差。若边$<v_k, v_j>$表示活动 a_i，则有 $l(i)=vl(j)-\text{Weight}(v_k, v_j)$。

5．一个活动 a_i 的最迟开始时间 $l(i)$和其最早开始时间 $e(i)$的差额 $d(i)=l(i)-e(i)$

它是指该活动完成的时间余量，即在不增加完成整个工程所需总时间的情况下，活动 a_i 可以拖延的时间。若一个活动的时间余量为零，则说明该活动必须要如期完成，否则就会拖延整个工程的进度，所以称 $l(i)-e(i)=0$ 即 $l(i)=e(i)$的活动 a_i 是关键活动。

求关键路径的算法步骤如下：

1）从源点出发，令 $ve(\text{源点})=0$，按拓扑有序求其余顶点的最早发生时间 $ve()$。

2）从汇点出发，令 $vl(\text{汇点})=ve(\text{汇点})$，按逆拓扑有序求其余顶点的最迟发生时间 $vl()$。

3）根据各顶点的 $ve()$值求所有弧的最早开始时间 $e()$。

4）根据各顶点的 $vl()$值求所有弧的最迟开始时间 $l()$。

5）求 AOE 网中所有活动的差额 $d()$，找出所有 $d()=0$ 的活动构成关键路径。

图 6.23 所示为求解关键路径的过程，简单说明如下：

1）求 $ve()$：初始 $ve(1)=0$，在拓扑排序输出顶点过程中，求得 $ve(2)=3$，$ve(3)=2$，$ve(4)=\max\{ve(2)+2, ve(3)+4\}=\max\{5, 6\}=6$，$ve(5)=6$，$ve(6)=\max\{ve(5)+1, ve(4)+2, ve(3)+3\}=\max\{7, 8, 5\}=8$。

如果这是一道选择题，根据上述求 $ve()$的过程就已经能知道关键路径。

2）求 $vl()$：初始 $vl(6)=8$，在逆拓扑排序出栈过程中，求得 $vl(5)=7$，$vl(4)=6$，$vl(3)=\min\{vl(4)-4, vl(6)-3\}=\min\{2, 5\}=2$，$vl(2)=\min\{vl(5)-3, vl(4)-2\}=\min\{4, 4\}=4$，$vl(1)$必然为 0 而无须再求。

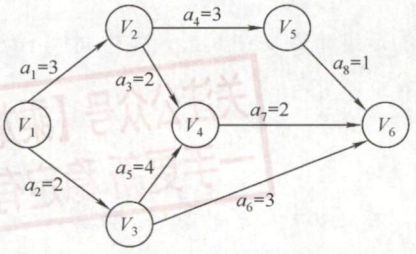
有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

3）弧的最早开始时间 $e()$等于该弧的起点的顶点的 $ve()$，求得结果如下表所示。

4）弧的最迟开始时间 $l(i)$等于该弧的终点的顶点的 $vl()$减去该弧持续的时间，求得结果如下表所示。

5）根据 $l(i)-e(i)=0$ 的关键活动，得到的关键路径为(v_1, v_3, v_4, v_6)。



	v_1	v_2	v_3	v_4	v_5	v_6
$ve(i)$	0	3	2	6	6	8
$vl(i)$	0	4	2	6	7	8

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
$e(i)$	0	0	3	3	2	2	6	6
$l(i)$	1	0	4	4	2	5	6	7
$l(i)-e(i)$	1	0	1	1	0	3	0	1

图 6.23　求解关键路径的过程

对于关键路径，需要注意以下几点：

1）关键路径上的所有活动都是关键活动，它是决定整个工程的关键因素，因此可以通过加快关键活动来缩短整个工程的工期。但也不能任意缩短关键活动，因为一旦缩短到一定的程度，该关键活动就可能会变成非关键活动。

2）网中的关键路径并不唯一，且对于有几条关键路径的网，只提高一条关键路径上的关键活动速度并不能缩短整个工程的工期，只有加快那些包括在所有关键路径上的关键活动才能达到缩短工期的目的。

6.4.6　本节试题精选

一、单项选择题

01. 任何一个无向连通图的最小生成树（　）。

A. 有一棵或多棵　　B. 只有一棵

C. 一定有多棵　　D. 可能不存在

02. 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（　）。

A. 相同　　B. 不相同

C. 可能相同，可能不同　　D. 无法比较

03. 以下叙述中，正确的是（　）。

A. 只要无向连通图中没有权值相同的边，则其最小生成树唯一

B. 只要无向图中有权值相同的边，则其最小生成树一定不唯一

C. 从 n 个顶点的连通图中选取 $n-1$ 条权值最小的边，即可构成最小生成树

D. 设连通图 G 含有 n 个顶点，则含有 n 个顶点、$n-1$ 条边的子图一定是 G 的生成树

04. 以下叙述中，正确的是（　）。

A. 最短路径一定是简单路径

B. Dijkstra 算法不适合求有回路的带权图的最短路径

C. Dijkstra 算法不适合求任意两个顶点的最短路径

D. Floyd 算法求两个顶点的最短路径时，$path_{k-1}$ 一定是 $path_k$ 的子集

05. 已知带权连通无向图 $G=(V, E)$，其中 $V=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$，$E=\{(v_1, v_2)10, (v_1, v_3)2,$

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

$(v_3, v_4)2, (v_3, v_6)11, (v_2, v_5)1, (v_4, v_5)4, (v_4, v_6)6, (v_5, v_7)7, (v_6, v_7)3\}$（注：顶点偶对括号外的数据表示边上的权值），从源点 v_1 到顶点 v_7 的最短路径上经过的顶点序列是（ ）。

A. v_1, v_2, v_5, v_7　　B. v_1, v_3, v_4, v_6, v_7　　C. v_1, v_3, v_4, v_5, v_7　　D. $v_1, v_2, v_5, v_4, v_6, v_7$

06. 下面的（ ）方法可以判断出一个有向图是否有环（回路）。

I. 深度优先遍历　II. 拓扑排序　III. 求最短路径　IV. 求关键路径

A. I、II、IV　　B. I、III、IV　　C. I、II、III　　D. 全部可以

07. 在有向图 G 的拓扑序列中，若顶点 v_i 在顶点 v_j 之前，则下列情形不可能出现的是（ ）。

A. G 中有弧 $<v_i, v_j>$　　B. G 中有一条从 v_i 到 v_j 的路径

C. G 中没有弧 $<v_i, v_j>$　　D. G 中有一条从 v_j 到 v_i 的路径

08. 以下关于拓扑排序的说法中，错误的是（ ）。

I. 若某有向图存在环路，则该有向图一定不存在拓扑排序

II. 在拓扑排序算法中为暂存入度为零的顶点，可以使用栈，也可以使用队列

III. 若有向图的拓扑有序序列唯一，则图中每个顶点的入度和出度最多为 1

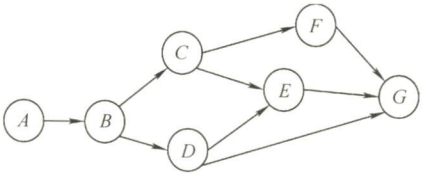
A. I、III　　B. II、III　　C. II　　D. III

09. 若一个有向图的顶点不能排成一个拓扑序列，则判定该有向图（ ）。

A. 含有多个出度为 0 的顶点　　B. 是个强连通图

C. 含有多个入度为 0 的顶点　　D. 含有顶点数大于 1 的强连通分量

10. 下图所示有向图的所有拓扑序列共有（ ）个。



A. 4　　B. 6　　C. 5　　D. 7

关注公众号【乘龙考研】一手更新 稳定有保障

11. 若一个有向图具有有序的拓扑排序序列，则它的邻接矩阵必定为（ ）。

A. 对称　　B. 稀疏　　C. 三角　　D. 一般

12. 下列关于图的说法中，正确的是（ ）。

I. 有向图中顶点 V 的度等于其邻接矩阵中第 V 行中 1 的个数

II. 无向图的邻接矩阵一定是对称矩阵，有向图的邻接矩阵一定是非对称矩阵

III. 在带权图 G 的最小生成树 G_1 中，某条边的权值可能会超过未选边的权值

IV. 若有向无环图的拓扑序列唯一，则可以唯一确定该图

A. I、II 和 III　　B. III 和 IV　　C. III　　D. IV

13. 若某带权图为 $G=(V, E)$，其中 $V=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$，$E=\{<v_1, v_2>5, <v_1, v_3>6, <v_2, v_5>3, <v_3, v_5>6, <v_3, v_4>3, <v_4, v_5>3, <v_4, v_7>1, <v_4, v_8>4, <v_5, v_6>4, <v_5, v_7>2, <v_6, v_{10}>4, <v_7, v_9>5, <v_8, v_9>2, <v_9, v_{10}>2\}$（注：边括号外的数据表示边上的权值），则 G 的关键路径的长度为（ ）。

A. 19　　B. 20　　C. 21　　D. 22

14. 下面关于求关键路径的说法中，不正确的是（ ）。

A. 求关键路径是以拓扑排序为基础的

B. 一个事件的最早发生时间与以该事件为始的弧的活动的最早开始时间相同

C. 一个事件的最迟发生时间是以该事件为尾的弧的活动的最迟开始时间与该活动的持

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

续时间的差

D. 关键活动一定位于关键路径上

15. 下列关于关键路径的说法中，正确的是（ ）。

I. 改变网上某一关键路径上的任意一个关键活动后，必将产生不同的关键路径

II. 在AOE图中，关键路径上活动的时间延长多少，整个工程的时间也就随之延长多少

III. 缩短关键路径上任意一个关键活动的持续时间可缩短关键路径长度

IV. 缩短所有关键路径上共有的任意一个关键活动的持续时间可缩短关键路径长度

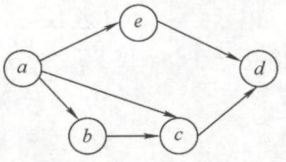
V. 缩短多条关键路径上共有的任意一个关键活动的持续时间可缩短关键路径长度

A. II和V　　B. I、II和IV　　C. II和IV　　D. I和IV

16. 用DFS遍历一个无环有向图，并在DFS算法退栈返回时打印相应的顶点，则输出的顶点序列是（ ）。

A. 逆拓扑有序　　B. 拓扑有序　　C. 无序的　　D. 无法确定

17. 【2010统考真题】对下图进行拓扑排序，可得不同拓扑序列的个数是（ ）。



A. 4　　B. 3　　C. 2　　D. 1

18. 【2012统考真题】下列关于最小生成树的叙述中，正确的是（ ）。

I. 最小生成树的代价唯一

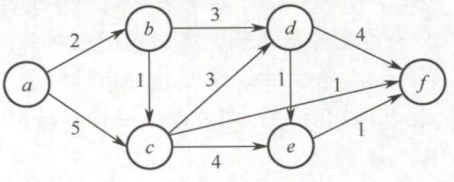
II. 所有权值最小的边一定会出现在所有的最小生成树中

III. 使用Prim算法从不同顶点开始得到的最小生成树一定相同

IV. 使用Prim算法和Kruskal算法得到的最小生成树总不相同

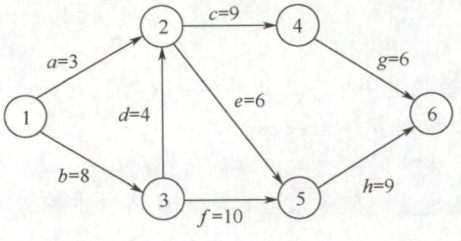
A. 仅I　　B. 仅II　　C. 仅I、III　　D. 仅II、IV

19. 【2012统考真题】对下图所示的有向带权图，若采用Dijkstra算法求从源点 a 到其他各顶点的最短路径，则得到的第一条最短路径的目标顶点是 b，第二条最短路径的目标顶点是 c，后续得到的其余各最短路径的目标顶点依次是（ ）。



A. d, e, f　　B. e, d, f　　C. f, d, e　　D. f, e, d

20. 【2013统考真题】下列AOE网表示一项包含8个活动的工程。通过同时加快若干活动的进度可以缩短整个工程的工期。下列选项中，加快其进度就可以缩短工程工期的是（ ）。



有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

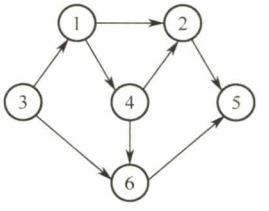
A. c 和 e　　B. d 和 c　　C. f 和 d　　D. f 和 h

21. 【2012 统考真题】若用邻接矩阵存储有向图，矩阵中主对角线以下的元素均为零，则关于该图拓扑序列的结论是（ ）。

A. 存在，且唯一　　B. 存在，且不唯一

C. 存在，可能不唯一　　D. 无法确定是否存在

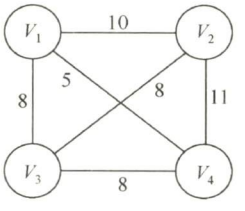
22. 【2014 统考真题】对下图所示的有向图进行拓扑排序，得到的拓扑序列可能是（ ）。



关注公众号【乘龙考研】
一手更新 稳定有保障

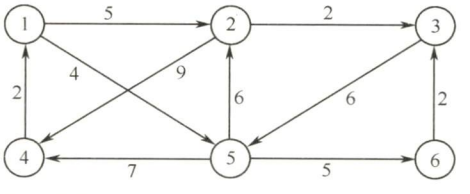
A. 3, 1, 2, 4, 5, 6　　B. 3, 1, 2, 4, 6, 5　　C. 3, 1, 4, 2, 5, 6　　D. 3, 1, 4, 2, 6, 5

23. 【2015 统考真题】求下面的带权图的最小（代价）生成树时，可能是 Kruskal 算法第 2 次选中但不是 Prim 算法（从 V_4 开始）第 2 次选中的边是（ ）。



A. (V_1, V_3)　　B. (V_1, V_4)　　C. (V_2, V_3)　　D. (V_3, V_4)

24. 【2016 统考真题】使用 Dijkstra 算法求下图中从顶点 1 到其他各顶点的最短路径，依次得到的各最短路径的目标顶点是（ ）。

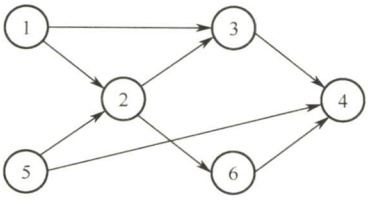


A. 5, 2, 3, 4, 6　　B. 5, 2, 3, 6, 4　　C. 5, 2, 4, 3, 6　　D. 5, 2, 6, 3, 4

25. 【2016 统考真题】若对 n 个顶点、e 条弧的有向图采用邻接表存储，则拓扑排序算法的时间复杂度是（ ）。

A. $O(n)$　　B. $O(n+e)$　　C. $O(n^2)$　　D. $O(ne)$

26. 【2018 统考真题】下列选项中，不是如下有向图的拓扑序列的是（ ）。



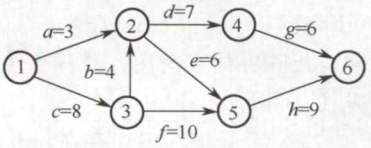
A. 1, 5, 2, 3, 6, 4　　B. 5, 1, 2, 6, 3, 4

C. 5, 1, 2, 3, 6, 4　　D. 5, 2, 1, 6, 3, 4

27. 【2019 统考真题】下图所示的 AOE 网表示一项包含 8 个活动的工程。活动 d 的最早开始时间和最迟开始时间分别是（ ）。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

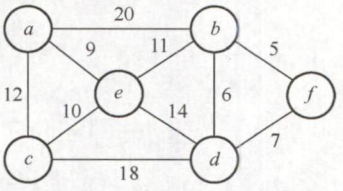


A. 3 和 7　　B. 12 和 12　　C. 12 和 14　　D. 15 和 15

28.【2019 统考真题】用有向无环图描述表达式 $(x+y)((x+y)/x)$，需要的顶点个数至少是（ ）。

A. 5　　B. 6　　C. 8　　D. 9

29.【2020 统考真题】已知无向图 G 如下所示，使用克鲁斯卡尔（Kruskal）算法求图 G 的最小生成树，加到最小生成树中的边依次是（ ）。



A. $(b,f),(b,d),(a,e),(c,e),(b,e)$

B. $(b,f),(b,d),(b,e),(a,e),(c,e)$

C. $(a,e),(b,e),(c,e),(b,d),(b,f)$

D. $(a,e),(c,e),(b,e),(b,f),(b,d)$

30.【2020 统考真题】修改递归方式实现的图的深度优先搜索（DFS）算法，将输出（访问）顶点信息的语句移到退出递归前（即执行输出语句后立刻退出递归）。采用修改后的算法遍历有向无环图 G，若输出结果中包含 G 中的全部顶点，则输出的顶点序列是 G 的（ ）。

A. 拓扑有序序列　　B. 逆拓扑有序序列

C. 广度优先搜索序列　　D. 深度优先搜索序列

31.【2020 统考真题】若使用 AOE 网估算工程进度，则下列叙述中正确的是（ ）。

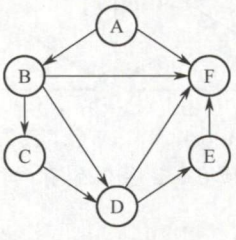
A. 关键路径是从源点到汇点边数最多的一条路径

B. 关键路径是从源点到汇点路径长度最长的路径

C. 增加任意一个关键活动的时间不会延长工程的工期

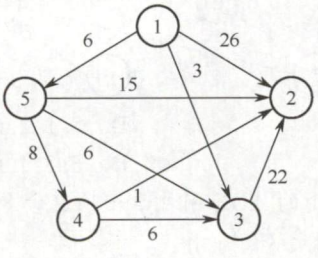
D. 缩短任意一个关键活动的时间将会缩短工程的工期

32.【2021 统考真题】给定如下有向图，该图的拓扑有序序列的个数是（ ）。



A. 1　　B. 2　　C. 3　　D. 4

33.【2021 统考真题】使用 Dijkstra 算法求下图中从顶点 1 到其余各顶点的最短路径，将当前找到的从顶点 1 到顶点 2, 3, 4, 5 的最短路径长度保存在数组 dist 中，求出第二条最短路径后，dist 中的内容更新为（ ）。

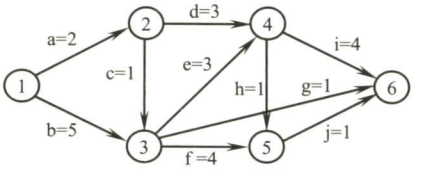


有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

A. 26, 3, 14, 6　　B. 25, 3, 14, 6　　C. 21, 3, 14, 6　　D. 15, 3, 14, 6

34. 【2022 统考真题】下图是一个有 10 个活动的 AOE 网，时间余量最大的活动是（ ）。



A. c　　B. g　　C. h　　D. j

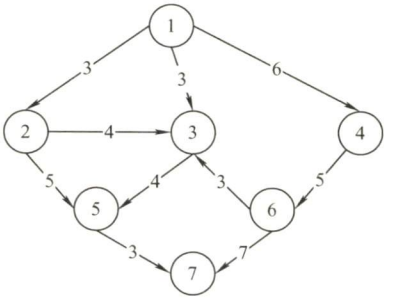
二、综合应用题

01. 下面是一种称为“破圈法”的求解最小生成树的方法：

所谓“破圈法”，是指“任取一圈，去掉圈上权最大的边”，反复执行这一步骤，直到没有圈为止。

试判断这种方法是否正确。若正确，说明理由；若不正确，举出反例（注：圈就是回路）。

02. 已知有向图如下图所示。



关注公众号【乘龙考研】
一手更新 稳定有保障

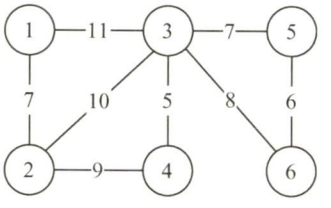
1）写出该图的邻接矩阵表示并据此给出从顶点 1 出发的深度优先遍历序列。

2）求该有向图的强连通分量的数目。

3）给出该图的任意两个拓扑序列。

4）若将该图视为无向图，分别用 Prim 算法和 Kruskal 算法求最小生成树。

03. 对下图所示的无向图，按照 Dijkstra 算法，写出从顶点 1 到其他各个顶点的最短路径和最短路径长度（顺序不能颠倒）。



04. 下图所示为一个用 AOE 网表示的工程。

1）画出此图的邻接表表示。

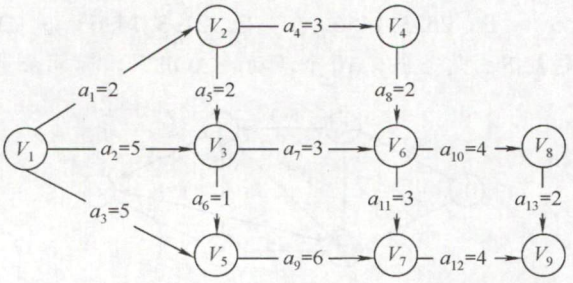
2）完成此工程至少需要多少时间？

3）指出关键路径。

4）哪些活动加速可以缩短完成工程所需的时间？

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】



05. 下表给出了某工程各工序之间的优先关系和各工序所需的时间（其中“—”表示无先驱工序），请完成以下各题：

1）画出相应的 AOE 网。

2）列出各事件的最早发生时间和最迟发生时间。

3）求出关键路径并指明完成该工程所需的最短时间。

工序代号	A	B	C	D	E	F	G	H
所需时间	3	2	2	3	4	3	2	1
先驱工序	—	—	A	A	B	A	C、E	D

06. 试说明利用 DFS 如何实现有向无环图拓扑排序。

07. 一连通无向图，边非负权值，问用 Dijkstra 最短路径算法能否给出一棵生成树，该树是否一定是最小生成树？说明理由。

08. 【2009 统考真题】带权图（权值非负，表示边连接的两顶点间的距离）的最短路径问题是找出从初始顶点到目标顶点之间的一条最短路径。假设从初始顶点到目标顶点之间存在路径，现有一种解决该问题的方法：

① 设最短路径初始时仅包含初始顶点，令当前顶点 u 为初始顶点。

② 选择离 u 最近且尚未在最短路径中的一个顶点 v，加入最短路径，修改当前顶点 $u=v$。

③ 重复步骤②，直到 u 是目标顶点时为止。

请问上述方法能否求得最短路径？若该方法可行，请证明；否则，请举例说明。

09. 【2011 统考真题】已知有 6 个顶点（顶点编号为 0～5）的有向带权图 G，其邻接矩阵 $\boldsymbol{A}$ 为上三角矩阵，按行为主序（行优先）保存在如下的一维数组中。

4	6	∞	∞	∞	5	∞	∞	∞	4	3	∞	∞	3	3

要求：

1）写出图 G 的邻接矩阵 $\boldsymbol{A}$。

2）画出有向带权图 G。

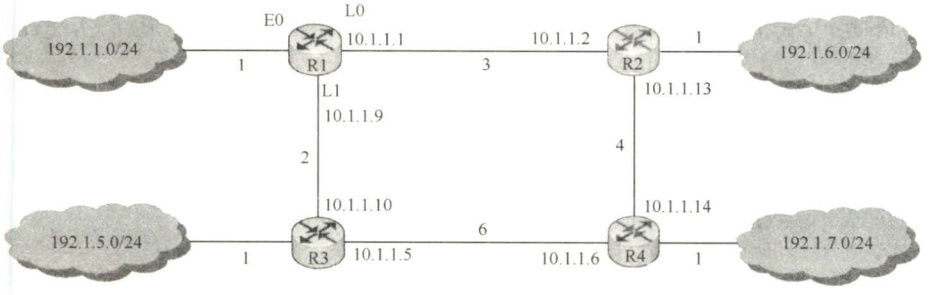
3）求图 G 的关键路径，并计算该关键路径的长度。

10. 【2014 统考真题】某网络中的路由器运行 OSPF 路由协议，下表是路由器 R1 维护的主要链路状态信息（LSI），R1 构造的网络拓扑图（见下图）是根据题下表及 R1 的接口名构造出来的网络拓扑。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

		R1 的 LSI	R2 的 LSI	R3 的 LSI	R4 的 LSI	备 注
Router ID		10.1.1.1	10.1.1.2	10.1.1.5	10.1.1.6	标识路由器的 IP 地址
Link1	ID	10.1.1.2	10.1.1.1	10.1.1.6	10.1.1.5	所连路由器的 Router ID
	IP	10.1.1.1	10.1.1.2	10.1.1.5	10.1.1.6	Link1 的本地 IP 地址
	Metric	3	3	6	6	Link1 的费用
Link2	ID	10.1.1.5	10.1.1.6	10.1.1.1	10.1.1.2	所连路由器的 Router ID
	IP	10.1.1.9	10.1.1.13	10.1.1.10	10.1.1.14	Link2 的本地 IP 地址
	Metric	2	4	2	4	Link2 的费用
Net1	Prefix	192.1.1.0/24	192.1.6.0/24	192.1.5.0/24	192.1.7.0/24	直连网络 Net1 的网络前缀
	Metric	1	1	1	1	到达直连网络 Net1 的费用



请回答下列问题。

1）本题中的网络可抽象为数据结构中的哪种逻辑结构？

2）针对表中的内容，设计合理的链式存储结构，以保存表中的链路状态信息（LSI）。要求给出链式存储结构的数据类型定义，并画出对应表的链式存储结构示意图（示意图中可仅以 ID 标识结点）。

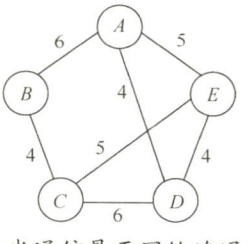
3）按照 Dijkstra 算法的策略，依次给出 R1 到达子网 192.1.x.x 的最短路径及费用。

11. 【2017 统考真题】使用 Prim 算法求带权连通图的最小（代价）生成树（MST）。请回答下列问题：

1）对下列图 G，从顶点 A 开始求 G 的 MST，依次给出按算法选出的边。

2）图 G 的 MST 是唯一的吗？

3）对任意的带权连通图，满足什么条件时，其 MST 是唯一的？



关注公众号【乘龙考研】
一手更新 稳定有保障

12. 【2018 统考真题】拟建设一个光通信骨干网络连通 BJ, CS, XA, QD, JN, NJ, TL 和 WH 等 8 个城市，下图中无向边上的权值表示两个城市之间备选光缆的铺设费用。

请回答下列问题：

1）仅从铺设费用角度出发，给出所有可能的最经济的光缆铺设方案（用带权图表示），并计算相应方案的总费用。

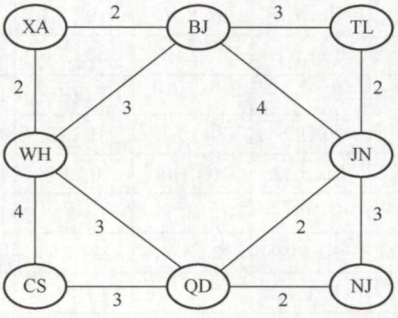
2）该图可采用图的哪种存储结构？给出求解问题 1）所用的算法名称。

3）假设每个城市采用一个路由器按 1）中得到的最经济方案组网，主机 H1 直接连接在

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

TL 的路由器上，主机 H2 直接连接在 BJ 的路由器上。若 H1 向 H2 发送一个 TTL = 5 的 IP 分组，则 H2 是否可以收到该 IP 分组？



6.4.7 答案与解析

一、单项选择题

01. A

当无向连通图存在权值相同的多条边时，最小生成树可能是不唯一的；另外，由于这是一个无向连通图，故最小生成树必定存在，从而选 A。

02. C

由于无向连通图的最小生成树不一定唯一，所以用不同算法生成的最小生成树可能不同，但当无向连通图的最小生成树唯一时，不同算法生成的最小生成树必定是相同的。

03. A

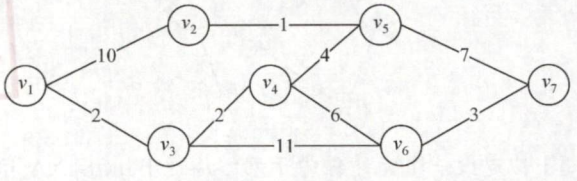
选项 A 显然正确，最小生成树算法是基于贪心策略的，每次总是选取权值最小且满足条件的边，如果各边权值不同，则每次选择的新顶点也是唯一的，因此最小生成树唯一。选项 B，若无向图本身就是一棵树，则最小生成树就是它本身，这时就是唯一的。选项 C，选取的 $n-1$ 条边可能构成回路。选项 D，含有 n 个顶点、$n-1$ 条边的子图可能构成回路，也可能不连通。

04. A

选项 A 正确，见严蔚敏的教材《数据结构》。Dijkstra 算法适合求解有回路的带权图的最短路径，也可以求任意两个顶点的最短路径，不适合求带负权值的最短路径问题。在用 Floyd 算法求两个顶点的最短路径时，当最短路径发生更改时，$path_{k-1}$ 就不是 $path_k$ 的子集。

05. B

题干内容所述的图 G 如下图所示。A, B, C, D 对应的路径长度分别为 18, 13, 15, 24。应用 Dijkstra 算法不难求出最短路径为 $v_1 \to v_3 \to v_4 \to v_6 \to v_7$。



06. A

使用深度优先遍历，若从有向图上的某个顶点 u 出发，在 DFS(u)结束之前出现一条从顶点 v 到 u 的边，由于 v 在生成树上是 u 的子孙，则图中必定存在包含 u 和 v 的环，因此深度优先遍历可以检测一个有向图是否有环。拓扑排序时，当某顶点不为任何边的头时才能加入序列，存在环时环中的顶点一直是某条边的头，不能加入拓扑序列。也就是说，还存在无法找到下一个可以加入拓扑序列的顶点，则说明此图存在回路。求最短路径是允许图有环的。至于关键路径能否判断一个图有

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

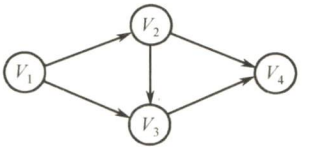
环，则存在一些争议。关键路径本身虽然不允许有环，但求关键路径的算法本身无法判断是否有环，判断是否有环是求关键路径的第一步——拓扑排序。所以这个问题的答案主要取决于你从哪个角度出发看问题，考生需要理解问题本身，真正统考时是不会涉及一些模棱两可的问题的。

07. D

若图 G 中存在一条从 v_j 到 v_i 的路径，说明 V_j 是 V_i 的前驱，则要把 V_j 消去以后才能消去 V_i，从而拓扑序列中必然先输出 v_j，再输出 v_i，这显然与题意矛盾。

08. D

I 中，对于一个存在环路的有向图，使用拓扑排序算法运行后，肯定会出现有环的子图，在此环中无法再找到入度为 0 的结点，拓扑排序也就进行不下去。II 中，注意，若两个结点之间不存在祖先或子孙关系，则它们在拓扑序列中的关系是任意的（即前后关系任意），因此使用栈和队列都可以，因为进栈或队列的都是入度为 0 的结点，此时入度为 0 的所有结点是没有关系的。III 是难点，若拓扑有序序列唯一，则很自然地让人联想到一个线性的有向图（错误），下图的拓扑序列也是唯一的，但度却不满足条件。



关注公众号【乘龙考研】
一手更新 稳定有保障

09. D

一个有向图中的顶点不能排成一个拓扑序列，表明其中存在一个顶点数目大于 1 的回路（环），该回路构成一个强连通分量，从而答案选 D。

10. C

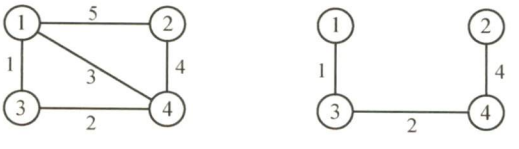
本图的拓扑排序序列有 *ABCFDEG*, *ABCDFEG*, *ABCDEFG*, *ABDCFEG* 和 *ABDCEFG*。读者应能把这一类经典习题的拓扑序列全部写出来。

11. C

此题一直以来争议较大，因为有些书中漏掉了“有序”二字。可以证明，对有向图中的顶点适当地编号，使其邻接矩阵为三角矩阵且主对角元全为零的充分必要条件是，该有向图可以进行拓扑排序。若这个题目把“有序”二字去掉，显然应该选 D。但此题题干中已经指出是“有序的拓扑序列”，因此应该选 C。需要注意的是，若一个有向图的邻接矩阵为三角矩阵（对角线以上或以下的元素为 0），则图中必不存在环，因此其拓扑序列必然存在。

12. C

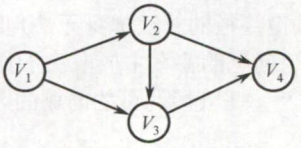
有向图邻接矩阵的第 V 行中 1 的个数是顶点 V 的出度，而有向图中顶点的度为入度与出度之和，I 错。无向图的邻接矩阵一定是对称矩阵，但当有向图中任意两个顶点之间有边相连，且是两条方向相反的有向边时，有向图的邻接矩阵也是一个对称矩阵，II 错。最小生成树中的 $n-1$ 条边并不能保证是图中权值最小的 $n-1$ 条边，因为权值最小的 $n-1$ 条边并不一定能使图连通。在下图中，左图的最小生成树如下图所示，权值为 3 的边并不在其最小生成树中。



有向无环图的拓扑序列唯一并不能唯一确定该图。在下图所示的两个有向无环图中，拓扑序列都为 V_1, V_2, V_3, V_4，IV 错。注意，很多辅导书对该命题的判断是错误的。

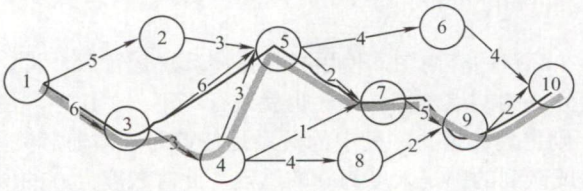
有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】



13．C

画出题目所表示的图如下，可得到关键路径的长度为 21。图中所示的两条路径都是关键路径。



14．C

一个事件的最迟发生时间 = min{以该事件为尾的弧的活动的最迟开始时间}，或 min{以该事件为尾的弧所指事件的最迟发生时间与该弧的活动的持续时间之差}。

15．C

若改变的是所有关键路径上的公共活动，则不一定会产生不同的关键路径（延长必然不会导致，只有缩短才有可能导致）。根据关键路径的定义，可知选项 II 正确。关键路径是源点到终点的最长路径，只有所有关键路径的长度都缩短时，整个图的关键路径才能有效缩短，但也不能任意缩短，一旦缩短到一定程度，该关键活动就可能会变成非关键活动。

16．A

设在图中有顶点 v_i，它有后继顶点 v_j，即存在边<v_i, v_j>。根据 DFS 的规则，v_i 入栈后，必先遍历完其后继顶点后 v_i 才会出栈，也就是说 v_i 会在 v_j 之后出栈，在如题所指的过程中，v_i 在 v_j 后打印。由于 v_i 和 v_j 具有任意性，可以从上面的规律看出，输出顶点的序列是逆拓扑有序序列。

17．B

可以得到 3 种不同的拓扑序列，即 *abced*, *abecd* 和 *aebcd*，故选 B。

18．A

最小生成树的树形可能不唯一（因为可能存在权值相同的边），但代价一定是唯一的，选项 I 正确。若权值最小的边有多条并且构成环状，则总有权值最小的边将不出现在某棵最小生成树中，选项 II 错误。设 N 个结点构成环，$N-1$ 条边权值相等，另一条边权值较小，则从不同的顶点开始 Prim 算法会得到 $N-1$ 种不同的最小生成树，选项 III 错误。当最小生成树唯一时（各边的权值不同），Prim 算法和 Kruskal 算法得到的最小生成树相同，选项 IV 错误。

19．C

从 a 到各顶点的最短路径的求解过程下：

顶点	第 1 轮	第 2 轮	第 3 轮	第 4 轮	第 5 轮
b	(a, b) 2				
c	(a, c) 5	(a, b, c) 3			
d	∞	(a, b, d) 5	(a, b, d) 5	(a, b, d) 5	
e	∞	∞	(a, b, c, e) 7	(a, b, c, e) 7	(a, b, d, e) 6
f	∞	∞	(a, b, c, f) 4		
集合 S	$\{a, b\}$	$\{a, b, c\}$	$\{a, b, c, f\}$	$\{a, b, c, f, d\}$	$\{a, b, c, f, d, e\}$

后续目标顶点依次为 f, d, e。

本题也可用排除法：对于 A，若下一个顶点为 d，路径 a, b, d 的长度为 5，而 a, b, c, f 的长度

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

仅为 4，显然错误。同理可排除选项 B。将 f 加入集合 S 后，采用上述方法也可排除选项 D。

20. C

找出 AOE 网的全部关键路径为 *bdcg*、*bdeh* 和 *bfh*。根据定义，只有关键路径上的活动时间同时减少时，才能缩短工期。选项 A、B 和 D 并不包含在所有的关键路径中，只有 C 包含，因此只有加快 f 和 d 的进度才能缩短工期（建议读者在图中检验）。

21. C

对角线以下的元素均为零，表明只有从顶点 i 到顶点 j（$i<j$）可能有边，而从顶点 j 到顶点 i 一定无边，即有向图是一个无环图，因此一定存在拓扑序列。对于拓扑序列是否唯一，试举一例：设有向图的邻接矩阵为 $\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$，则存在两个拓扑序列，因此该图存在可能不唯一的拓扑序列。若题目中说对角线以上的元素均为 1，以下的元素均为 0，则拓扑序列唯一。

22. D

按照拓扑排序的算法，每次都选择入度为 0 的结点从图中删除，此图中一开始只有结点 3 的入度为 0；删除结点 3 后，只有结点 1 的入度为 0；删除结点 1 后，只有结点 4 的入度为 0；删除结点 4 后，结点 2 和结点 6 的入度都为 0，此时选择删除不同的结点，会得出不同的拓扑序列，分别处理完毕后可知可能的拓扑序列为 3, 1, 4, 2, 6, 5 和 3, 1, 4, 6, 2, 5，选 D。

23. C

从 V_4 开始，Kruskal 算法选中的第一条边一定是权值最小的(V_1, V_4)，选项 B 错误。由于 V_1 和 V_4 已经可达，含有 V_1 和 V_4 的权值为 8 的第二条边一定符合 Prim 算法，排除 A、D。

24. B

根据 Dijkstra 算法，从顶点 1 到其余各顶点的最短路径如下表所示。

顶　点	第 1 轮	第 2 轮	第 3 轮	第 4 轮	第 5 轮
2	5 $v_1 \to v_2$	5 $v_1 \to v_2$			
3	∞	∞	7 $v_1 \to v_2 \to v_3$		
4	∞	11 $v_1 \to v_5 \to v_4$	11 $v_1 \to v_5 \to v_4$	11 $v_1 \to v_5 \to v_4$	11 $v_1 \to v_5 \to v_4$
5	4 $v_1 \to v_5$				
6	∞	9 $v_1 \to v_5 \to v_6$	9 $v_1 \to v_5 \to v_6$	9 $v_1 \to v_5 \to v_6$	
集合 S	{1, 5}	{1, 5, 2}	{1, 5, 2, 3}	{1, 5, 2, 3, 6}	{1, 5, 2, 3, 6, 4}

关注公众号【乘龙考研】
一手更新 稳定有保障

25. B

采用邻接表作为 AOV 网的存储结构进行拓扑排序，需要对 n 个顶点做进栈、出栈、输出各一次，当处理 e 条边时，需要检测这 n 个顶点的边链表的 e 个边结点，总共需要的时间代价为 $O(n+e)$。若采用邻接矩阵作为 AOV 网的存储结构进行拓扑排序，在处理 e 条边时需对每个顶点检测相应矩阵中的某一行，寻找与它相关联的边，以便对这些边的入度减 1，需要的时间代价为 $O(n^2)$。

26. D

拓扑排序每次选取入度为 0 的结点输出，经观察不难发现拓扑序列前两位一定是 1, 5 或 5, 1，（因为只有 1 和 5 的入度均为 0，且其他结点都不满足仅有 1 或仅有 5 作为前驱）。选项 D 错误。

27. C

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

活动 d 的最早开始时间等于该活动弧的起点所表示的事件的最早发生时间，活动 d 的最早开始时间等于事件 2 的最早发生时间 $\max\{a,b+c\}=\max\{3,12\}=12$。活动 d 的最迟开始时间等于该活动弧的终点所表示的事件的最迟发生时间与该活动所需时间之差，先算出图中关键路径长度为 27（对于不复杂的选择题，找出所有路径计算长度），那么事件 4 的最迟发生时间为 $\min\{27-g\}=\min\{27-6\}=21$，活动 d 的最迟开始时间为 $21-d=21-7=14$。

常规方法：按照关键路径算法算得到下表。

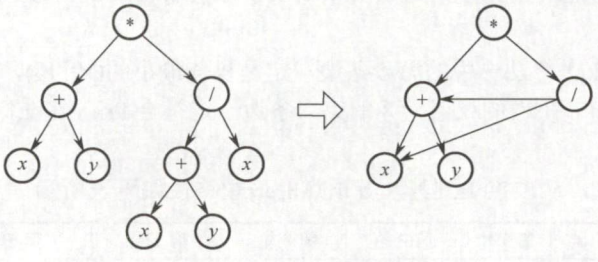
	v_1	v_2	v_3	v_4	v_5	v_6
$ve(i)$	0	12	8	19	18	27
$vl(i)$	0	12	8	21	18	27

	a	b	c	d	e	f	g	h
$e(i)$	0	8	0	12	12	8	19	18
$l(i)$	9	8	0	14	12	8	21	18
$l(i)-e(i)$	9	0	0	2	0	0	2	0

从表中可知，活动 d 的最早开始时间和最迟开始时间分别为 12 和 14，故选 C。

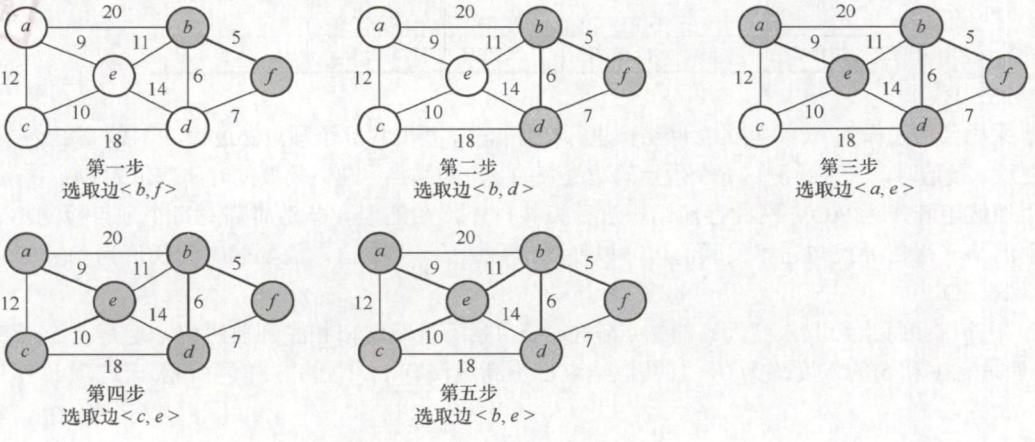
28．A

先将该表达式转换成有向二叉树，注意到该二叉树中有些顶点是重复的，为了节省存储空间，可以去除重复的顶点（使顶点个数达到最少），将有向二叉树去重转换成有向无环图，如下图所示。



29．A

Kruskal 算法：按权值递增顺序依次选取 $n-1$ 条边，并保证这 $n-1$ 条边不构成回路。初始构造一个仅含 n 个顶点的森林；第一步，选取权值最小的边(b,f)加入最小生成树；第二步，剩余边中权值最小的边为(b,d)，加入最小生成树，第二步操作后权值最小的边(d,f)不能选，因为会与之前已选取的边形成回路；接下来依次选取权值 9、10、11 对应的边加入最小生成树，此时 6 个顶点形成了一棵树，最小生成树构造完成。按照上述过程，加到最小生成树的边依次为(b,f), (b,d), (a,e), (c,e), (b,e)。其生成过程如下所示。



有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

30．B

DFS 是一个递归算法，在遍历过程中，先访问的顶点被压入栈底。设在图中有顶点 v_i，它有后继顶点 v_j，即存在边<v_i, v_j>。根据 DFS 的规则，v_i 入栈后，必先遍历完其后继顶点后 v_i 才会出栈，也就是说 v_i 会在 v_j 之后出栈，在如题所指的过程中，v_i 在 v_j 后打印。由于 v_i 和 v_j 具有任意性，从上面的规律可以看出，输出顶点的序列是逆拓扑有序序列。

31．B

关键路径是指权值之和最大而非边数最多的路径，故选项 A 错误。选项 B 正确，是关键路径的概念。无论是存在一条还是存在多条关键路径，增加任意一个关键活动的时间都会延长工程的工期，因为关键路径始终是权值之和最大的那条路径，选项 C 错误。仅有一条关键路径时，减少关键活动的时间会缩短工程的工期；存在多条关键路径时，缩短一条关键活动的时间不一定会缩短工程的工期，缩短了路径长度的那条关键路径不一定还是关键路径，选项 D 错误。

32．A

求拓扑序列的过程：从图中选择无入边的结点，输出该结点并删除该结点的所有出边，重复上述过程，直至全部结点都已输出，这样求得的拓扑序列为 ABCDEF。每次输出一个结点并删除该结点的所有出边后，都发现有且仅有一个结点无入边，因此该拓扑序列唯一。

33．C

在执行 Dijkstra 算法时，首先初始化 dist[]，若顶点 1 到顶点 i（i = 2, 3, 4, 5）有边，就初始化为边的权值；若无边，就初始化为∞；初始化顶点集 S 只含顶点 1。Dijkstra 算法每次选择一个到顶点 1 距离最近的顶点 j 加入顶点集 S，并判断由顶点 1 绕行顶点 j 后到任意一个顶点 k 是否距离更短，若距离更短（即 dist[j] + arcs[j][k] < dist[k]），则将 dist[x]更新为 dist[j] + arcs[j][k]；重复该过程，直至所有顶点都加入顶点集 S。数组 dist 的变化过程如下图所示，可知将第二个顶点 5 加入顶点集 S 后，数组 dist 更新为 21, 3, 14, 6，故选 C。

$$\text{dist}\{26, 3, \infty, 6\} \xrightarrow{\text{顶点3入S}} \{25, 3, \infty, 6\} \xrightarrow{\text{顶点5入S}} \{21, 3, 14, 6\}$$

34．B

在 AOE 网中，活动的时间余量＝结束顶点的最迟开始时间－开始顶点的最早开始时间－该活动的持续时间。根据关键路径算法得到下表：

结点编号	1	2	3	4	5	6
最早开始时间 $ve(i)$	0	2	5	8	9	12
最迟开始时间 $vl(i)$	0	4	5	8	11	12

c 的时间余量 $= vl(3) - ve(2) - 1 = 5 - 2 - 1 = 2$，g 的时间余量 $= vl(6) - ve(3) - 1 = 12 - 5 - 1 = 6$，h 的时间余量 $= vl(5) - ve(4) - 1 = 11 - 8 - 1 = 2$，j 的时间余量 $= vl(6) - ve(5) - 1 = 12 - 9 - 1 = 2$，时间余量最大的活动是 g。

二、综合应用题

关注公众号【乘龙考研】
一手更新 稳定有保障

01．【解答】

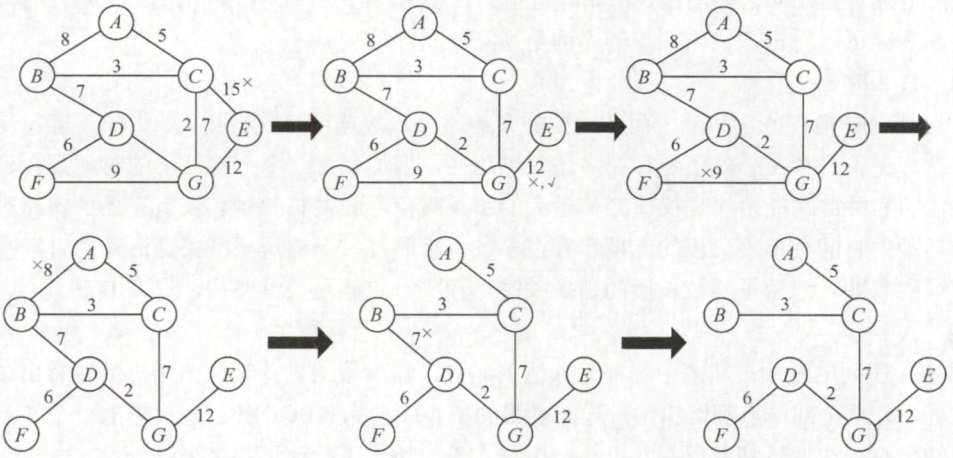
这种方法是正确的。

由于经过“破圈法”之后，最终没有回路，故一定可以构造出一棵生成树。下面证明这棵生成树是最小生成树。记“破圈法”生成的树为 T，假设 T 不是最小生成树，则必然存在最小生成

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

树 T_0，使得它与 T 的公共边尽可能地多，则将 T_0 与 T 取并集，得到一个图，此图中必然存在回路，由于“破圈法”的定义就是从回路中去除权最大的边，此时生成的 T 的权必然是最小的，这与原假设 T 不是最小生成树矛盾，从而 T 是最小生成树。下图说明了“破圈法”的过程：



02.【解答】

1）该图的邻接矩阵为

$$A=\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\end{array}\begin{array}{c}\begin{array}{ccccccc}1&2&3&4&5&6&7\end{array}\\ \begin{bmatrix}0&3&3&6&\infty&\infty&\infty\\ \infty&0&4&\infty&5&\infty&\infty\\ \infty&\infty&0&\infty&4&\infty&\infty\\ \infty&\infty&\infty&0&\infty&5&\infty\\ \infty&\infty&\infty&\infty&0&\infty&3\\ \infty&\infty&3&\infty&\infty&0&7\\ \infty&\infty&\infty&\infty&\infty&\infty&0\end{bmatrix}\end{array}$$

得到的深度优先遍历序列为 1, 2, 3, 5, 7, 4, 6。

2）解题思路：当某个顶点只有出弧而没有入弧时，其他顶点无法到达这个顶点，不可能与其他顶点和边构成强连通分量（这个单独的顶点构成一个强连通分量）。

① 顶点 1 无入弧构成第一个强连通分量。删除顶点 1 及所有以之为尾的弧。

② 顶点 2 无入弧构成一个强连通分量。删除顶点 2 及所有以之为尾的弧。

③ ……

以此类推，最后得到每个顶点都是一个强连通分量，故强连通分量数目为 7。

3）该图的两个拓扑序列如下：

① 1, 2, 4, 6, 3, 5, 7

② 1, 4, 2, 6, 3, 5, 7

4）若视该图为无向图：

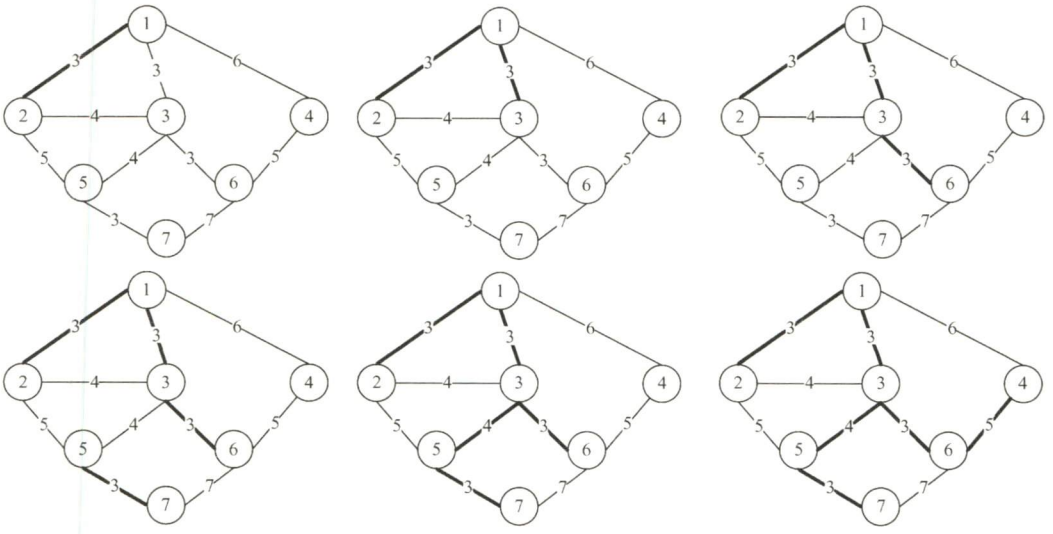
用 Prim 算法生成最小生成树的过程如下：

1—2，1—3，3—6，3—5，5—7，6—4（图略）。

用 Kruskal 算法生成最小生成树的过程如下图所示。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】



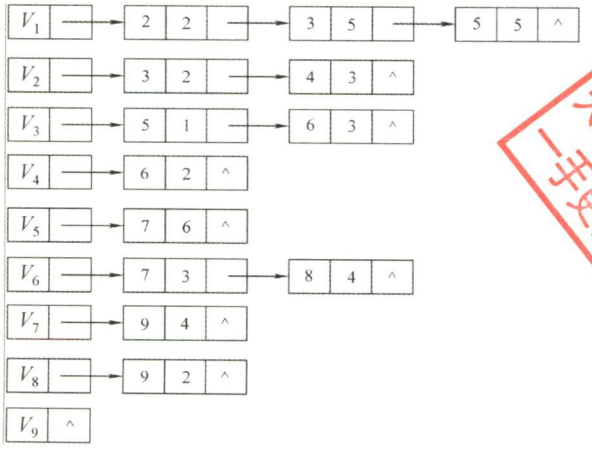
03.【解答】

根据 Dijkstra 算法，求从顶点 1 到其余各顶点的最短路径如下表所示。

顶点	从顶点 1 到各终点的 dist 值				
	第 1 轮	第 2 轮	第 3 轮	第 4 轮	
2	**7** **v_1, v_2**				
3	11 v_1, v_3	**11** **v_1, v_3**			
4	∞	16 v_1, v_2, v_4	**16** **v_1, v_2, v_4**		
5	∞	∞	18 v_1, v_3, v_5	**18** **v_1, v_3, v_5**	
6	∞	∞	19 v_1, v_3, v_6	19 v_1, v_3, v_6	**19** **v_1, v_3, v_6**
集合 S	**{1, 2}**	**{1, 2, 3}**	**{1, 2, 3, 4}**	**{1, 2, 3, 4, 5}**	**{1, 2, 3, 4, 5, 6}**

04.【解答】

1）该图的邻接表表示如下图所示。



关注公众号【乘龙考研】一手更新 稳定有保障

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

求关键路径的算法如下：

① 输入 e 条弧$<j, k>$，建立 AOE 网的存储结构。

② 从源点 v_1 出发，令 $v_e(1)=0$，求 $v_e(j)$，$2\leqslant j\leqslant n$。

③ 从汇点 v_n 出发，令 $v_l(n)=v_e(n)$，求 $v_l(i)$，$1\leqslant i\leqslant n-1$。

④ 根据各顶点的 v_e 和 v_l 值，求每条弧 s（活动）的最早开始时间 $e(s)$和最晚开始时间 $l(s)$，其中 $e(s)=l(s)$为关键活动。

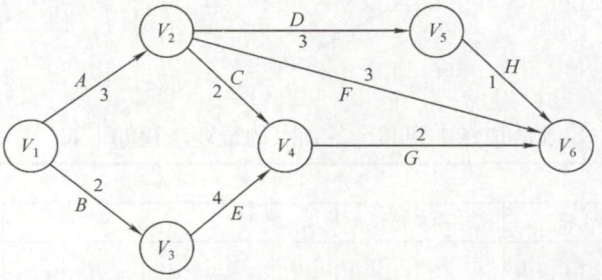
2）根据以上算法可以得到至少需要时间 16。

3）关键路径为$(V_1, V_3, V_5, V_7, V_9)$。

4）活动 a_2, a_6, a_9, a_{12} 加速，可以缩短工程所需的时间。

05.【解答】

1）根据题表可以画出 AOE 网如下图所示。



求解各事件和活动的最早发生时间与最迟发生时间公式分别如下：

① v_e(源点) = 0，$v_e(k)=\mathrm{Max}\{v_e(j)+\mathrm{Weight}(v_j, v_k)\}$，$\mathrm{Weight}(v_j, v_k)$表示从 v_j 指向 v_k 的弧的权值。

② v_l(汇点) = v_e(汇点)，$v_l(j)=\mathrm{Min}\{v_l(k)-\mathrm{Weight}(v_j, v_k)\}$，$\mathrm{Weight}(v_j, v_k)$表示从 v_j 指向 v_k 的弧的权值。

③ 若边$<v_k, v_j>$表示活动 a_i，则有 $e(i)=v_e(k)$。

④ 若边$<v_k, v_j>$表示活动 a_i，则有 $l(i)=v_l(j)-\mathrm{Weight}(v_k, v_j)$。

⑤ $d(i)=l(i)-e(i)$。

关键路径即由 $d(i)=0$ 的 i 构成。

2）根据上述公式，各事件的最早发生时间 v_e 和最迟发生时间 v_l 如下表所示。

	v_1	v_2	v_3	v_4	v_5	v_6
$v_e(i)$	0	3	2	6	6	8
$v_l(i)$	0	4	2	6	7	8

3）根据上述公式，各活动最早发生时间 e、最迟发生时间 l 和时间余量 $d(i)=l(i)-e(i)$如下表所示。

	A	B	C	D	E	F	G	H
$e(i)$	0	0	3	3	2	3	6	6
$l(i)$	1	0	4	4	2	5	6	7
$l(i)-e(i)$	1	0	1	1	0	2	0	1

所以关键路径为 B、E、G，完成该工程最少需要 8（单位依题意而定）。

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

06.【解答】

对于有向无环图 G 中的任意结点 u, v，它们之间的关系必然是下列三种之一：

1）假设结点 u 是结点 v 的祖先，则在调用 DFS 访问 u 的过程中，必然会在这个过程结束之前递归地对 v 调用 DFS 访问，即 v 的 DFS 函数结束时间先于 u 的 DFS 结束时间。从而可以考虑在 DFS 调用过程中设定一个时间标记，在 DFS 调用结束时，对各结点计时。因此，祖先的结束时间必然大于子孙的结束时间。

2）若 u 是结点 v 的子孙，则 v 为 u 的祖先，按上述思路，v 的结束时间大于 u 的结束时间。

3）若 u 和 v 没有关系，则 u 和 v 在拓扑序列的关系任意。

从而按结束时间从大到小，可以得到一个拓扑序列。

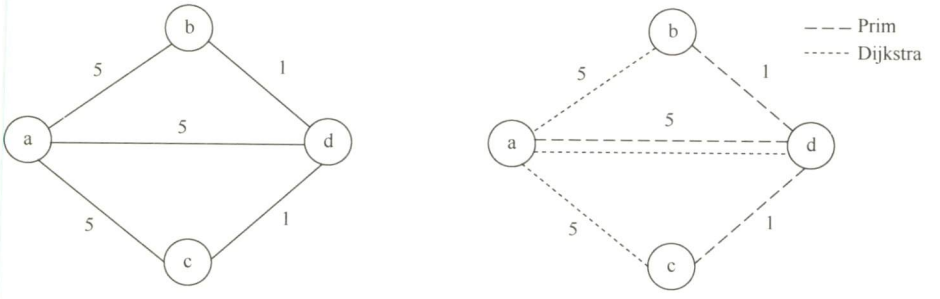
下面给出利用 DFS 求各结点结束时间的代码。至于拓扑序列，将结束时间从大到小排序即可得到（实际上和深度优先遍历算法完全相同，只不过加入了 time 变量）。

```
bool visited[MAX_VERTEX_NUM];        //访问标记数组
void DFSTraverse(Graph G){
//对图 G 进行遍历，访问函数为 visit()
    for(v=0;v<G.vexnum;++v)
        visited[v]=FALSE;            //初始化访问标记数组
    time=0;
    for(v=0;v<G.vexnum;++v)          //本代码中是从 v=0 开始遍历
        if(!visited[v]) DFS(G,v);
}
void DFS(Graph G,int v){
    visited[v]=TRUE;
    visit(v);
    for(w=FirstNeighbor(G,v);w>=0;w=NextNeighbor(G,v,w))
        if(!visited[w]){             //w 为 v 的尚未访问的邻接顶点
            DFS(G,w);
        }//if
    time=time+1;finishTime[v]=time;
}
```

关注公众号【乘龙考研】
一手更新 稳定有保障

07.【解答】

Dijkstra 算法每一步都会贪婪地选择与源点 v_0 最近的下一条边，直到 v_0 连接到图中所有顶点。Prim 算法（已知是最小生成树算法）与 Dijkstra 算法高度相似，但是在每个阶段，它贪婪地选择与该阶段已加入 MST 中任意一个顶点最近的下一条边。显然，Dijkstra 算法可以产生一棵生成树，但该树不一定是最小生成树，只需举出一个反例即可，以下图 G 为例（将 a 作为源点）。



(a) 图 G　　(b) 两种算法产生的生成树

Dijkstra 算法得到的路径集合为{(a, b), (a, c), (a, d)}，该生成树的总权值为 5 + 5 + 5 = 15。

Prim 算法得到的边集合为{(a, d), (b, d), (c, d)}，该最小生成树的总权值为 5 +1 + 1 = 7。

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

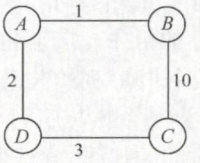
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

显然，Dijkstra 算法得到的生成树不一定是最小生成树。

08.【解答】

该方法不一定能（或不能）求得最短路径。

例如，对于下图所示的带权图，若按照题中的原则，从 A 到 C 的最短路径是 $A \to B \to C$，事实上其最短路径是 $A \to D \to C$。

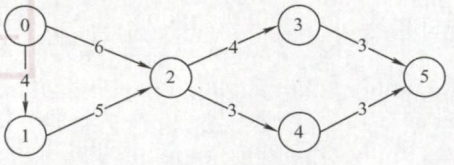


09.【解答】

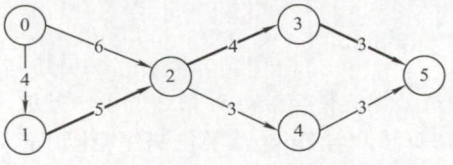
1）图 G 的邻接矩阵 $\boldsymbol{A}$ 为

$$\boldsymbol{A}=\begin{bmatrix} 0 & 4 & 6 & \infty & \infty & \infty \\ \infty & 0 & 5 & \infty & \infty & \infty \\ \infty & \infty & 0 & 4 & 3 & \infty \\ \infty & \infty & \infty & 0 & \infty & 3 \\ \infty & \infty & \infty & \infty & 0 & 3 \\ \infty & \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

2）有向带权图 G 如下图所示。



3）关键路径为 $0 \to 1 \to 2 \to 3 \to 5$（如下图中的粗线表示），长度为 $4+5+4+3=16$。



注意：读者务必熟练掌握根据邻接矩阵（邻接表）画出有向图（或无向图）的方法，及计算其关键路径、最短路径及长度等的方法。

10.【解答】

考查具体模型中数据结构的应用。很多考生乍看之下，以为这是关于网络的题目，其实该题本身并未涉及太多的网络知识点，只是应用了网络的模型，实际考查的还是数据结构。

1）图题中给出的是一个简单的网络拓扑图，可以抽象为无向图。

2）链式存储结构的如下图所示。

弧结点的两种基本形态

Flag=1	Next
ID	
IP	
Metric	

Flag=2	Next
Prefix	
Mask	
Metric	

表头结点结构示意

RouterID
LN_link
Next

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

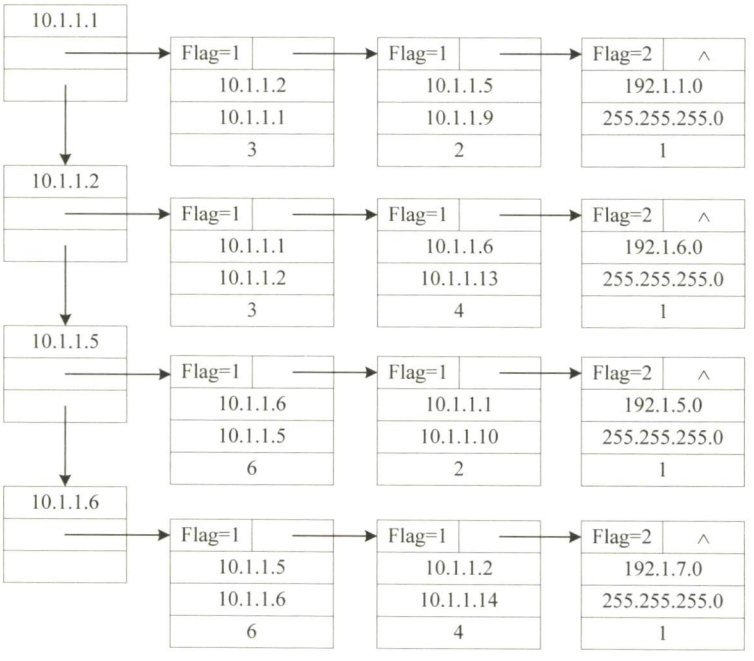
更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

其数据类型定义如下：

```
typedef struct{
unsigned int ID, IP;
}LinkNode;                //Link 的结构
typedef struct{
unsigned int Prefix, Mask;
}NetNode;                 //Net 的结构
typedef struct Node{
int Flag;                 //Flag=1 为 Link;Flag=2 为 Net
union{
    LinkNode Lnode;
    NetNode Nnode
}LinkORNet;
Unsigned int Metric;
struct Node *next;
}ArcNode;                 //弧结点
typedef struct hNode{
unsigned int RouterID;
ArcNode *LN_link;
Struct hNode *next;
}HNODE;                   //表头结点
```

关注公众号【乘龙考研】
一手更新 稳定有保障

对应表的链式存储结构示意图如下所示。



3）计算结果如下表所示。

	目的网络	路径	代价（费用）
步骤 1	192.1.1.0/24	直接到达	1
步骤 2	192.1.5.0/24	R1→R3→192.1.5.0/24	3
步骤 3	192.1.6.0/24	R1→R2→192.1.6.0/24	4
步骤 4	192.1.7.0/24	R1→R2→R4→192.1.7.0/24	8

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】

11.【解答】

1）Prim 算法属于贪心策略。算法从一个任意的顶点开始，一直长大到覆盖图中的所有顶点为止。算法的每一步在连接树集合 S 的顶点和其他顶点的边中，选择一条使得树的总权重增加最小的边加入集合 S。当算法终止时，S 就是最小生成树。

① S 中顶点为 A，候选边为(A, D), (A, B), (A, E)，选择(A, D)加入 S。

② S 中顶点为 A, D，候选边为(A, B), (A, E), (D, E), (C, D)，选择(D, E)，加入 S。

③ S 中顶点为 A, D, E，候选边为(A, B), (C, D), (C, E)，选择(C, E)加入 S。

④ S 中顶点为 A, D, E, C，候选边为(A, B), (B, C)，选择(B, C)加入 S。

⑤ S 就是最小生成树。

依次选出的边为

$$(A, D), (D, E), (C, E), (B, C)$$

2）图 G 的 MST 是唯一的。第一小题的最小生成树包括了图中权值最小的 4 条边，其他边除(A, E)外都比这 4 条边大，但若用(A, E)替换同权值的(C, E)，A, D, E 三个顶点构成了回路，因此不能替换，所以此图的 MST 唯一。

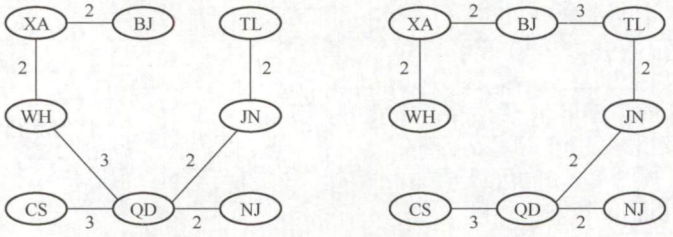
3）当带权连通图的任意一个环中所包含的边的权值均不相同时，其 MST 是唯一的。此题不要求回答充分必要条件，所以回答一个限制边权值的充分条件即可。

12.【解答】

1）为了求解最经济的方案，可把问题抽象为求无向带权图的最小生成树。可以采用手动 Prim 算法或 Kruskal 算法作图。注意本题的最小生成树有两种构造，如下图所示。

方案的总费用为 16。

2）存储题中的图可采用邻接矩阵（或邻接表）。构造最小生成树采用 Prim 算法（或 Kruskal 算法）。



方案1　　方案2

3）TTL = 5，即 IP 分组的生存时间（最大传递距离）为 5，方案 1 中 TL 和 BJ 的距离过远，TTL = 5 不足以让 IP 分组从 H1 传送到 H2，因此 H2 不能收到 IP 分组。而方案 2 中 TL 和 BJ 邻近，H2 可以收到 IP 分组。

归纳总结

1. 关于图的基本操作

本章中的很多程序对采用邻接表或邻接矩阵的存储结构都适用，主要原因是在图的基本操作函数中保持了相同的参数和返回值，而封闭了内部实现细节。

例如，取 x 邻接顶点 y 的下一个邻接顶点的函数 `NextNeighbor(G,x,y)`。

1）用邻接矩阵作为存储结构

```
int NextNeighbor(MGraph& G,int x,int y){
    if(x!=-1 && y!=-1){
```

有任何网盘群类资料需求都可添加学长微信 Learn52013　祝您考研上岸

更多考研考证类日更网盘群&笔记类资料 请关注微信公众号【乘龙考研】


```
        for(int col=y+1;col<G.vexnum;col++)
            if(G.Edge[x][col]>0 && G.Edge[x][col]<maxWeight)
return col;                //maxWeight 代表∞
    }
    return -1;
}
```

2）用邻接表作为存储结构

```
int NextNeighbor(ALGraph& G,int x,int y){
    if(x!=-1){                                  //顶点 x 存在
        ArcNode *p=G.vertices[x].first;         //对应边链表第一个边结点
        while(p!=NULL && p->data!=y)            //寻找邻接顶点 y
            p=p->next;
        if(p!=NULL && p->next!=NULL)
            return p->next->data;               //返回下一个邻接顶点
    }
    return -1;
}
```

2. 关于图的遍历、连通性、生成树、关键路径的几个要点

1）在执行图的遍历时，由于图中可能存在回路，且图的任意一个顶点都可能与其他顶点相连，所以在访问完某个顶点后可能会沿某些边又回到了曾经访问过的顶点。因此，需要设置一个辅助数组 visited[]标记顶点是否已被访问过，避免重复访问。

2）深度优先搜索时利用回溯法对图遍历，一般利用递归方法实现，每当向前递归查找某一邻接结点之前，必须判断该结点是否访问过。另外，递归算法均可借助栈来实现非递归算法，深度优先搜索也不例外，具体程序见 6.3.4 节的综合应用题 3。

3）广度优先搜索是一种分层的遍历过程，每向前走一步可能访问一批顶点，不像深度优先搜索那样有回退的情况。因此，它不是一个递归的过程。

4）一个给定的图的邻接矩阵表示是唯一的，但对于邻接表来说，若边的输入先后次序不同，则生成的邻接表表示也不同。

5）图的最小生成树首先必须是带权连通图，其次要在 n 个顶点的图中选择 $n-1$ 条边将其连通，使得其权值总和达到最小，且不出现回路。

6）加速某一关键活动不一定能缩短整个工程的工期，因为 AOE 网中可能存在多条关键路径。可能存在称为“桥”的一种特殊关键活动，它位于所有的关键路径上，只有它加速才会缩短整个工期。

关注公众号【乘龙考研】
一手更新 稳定有保障

思维拓展

【网易有道笔试题】求一个无向连通图的割点。割点的定义是，若除去此结点和与其相关的边，无向图不再连通，描述算法。

（**提示**：要判断一个点是否为割点，最简单且最直接的方法是，先把这个点和所有与它相关的边从图中去掉，再用深搜或广搜来判断剩下的图的连通性，这种方法适合判断给定结点是否为割点；还有一种比较复杂的方法可以快速找出所有割点，有兴趣的读者可自行上网查询相关资料。）

有任何网盘群类资料需求都可添加学长微信 Learn52013 祝您考研上岸