

考点脉络



数据结构是指数据对象及其相互关系和构造方法。在软件设计过程中，不同的数据结构的选用，对系统最终效果的影响极大。所以该知识点是软件设计师核心考点，无论是上午综合知识部分，还是下午的软件设计部分，考查分值都很高。

根据考试大纲，本章要求考生掌握数组、链表、队列和栈、树、图、杂凑相关知识，从历年的考试情况来看，本章主要考查常见数据结构的逻辑结构特性及存储的相关内容。

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

数组与线性表

按数据的逻辑结构来划分，常见的数据结构包括：数组（静态数组、动态数组）、线性表（顺序表、链表、队列、栈）、树（二叉树、查找树、平衡树、线索树、堆）、图。本节将介绍数组与线性表的相关内容。

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

考点精讲

1. 数组

数组是一种常见的数据结构，根据数组下标的个数，可以把数组分为一维、二维、...、多维数组，如表4-1所示。维度是指下标的个数。一维数组只有一个下标；二维数组则有两个下标，第一个称为行下标，第二个称为列下标。根据数组的定义，计算存储地址是一个经常考查的知识点。

表4-1数组类型

数组类型	存储地址计算
一维数组 a[n]	a[i]的存储地址为：a+i*len
二维数组 a[m][n]	a[i][j]的存储地址（按行存储）为：a+(i*n+j)*len a[i][j]的存储地址（按列存储）为：a+(j*m+i)*len
三维数组 a[m][n][o]	a[i][j][k]的存储地址为：a+(i*n*o+j*o+k)*len

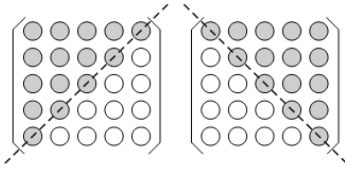
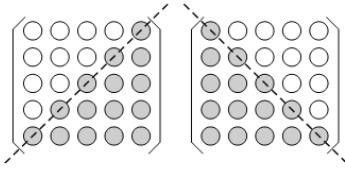
注：表4-1计算公式中的a为数组首地址，len为每个数据对象的长度，i与j的下标默认从0开始。

2. 稀疏矩阵

在计算机中存储一个矩阵时，可使用二维数组。例如， $M \times N$ 阶矩阵可用一个数组 $a[M][N]$ 来存储（可按照行优先或列优先的顺序）。如果一个矩阵的元素绝大部分为零，则称为稀疏矩阵。若直接用一个二维数组表示稀疏矩阵，则会因存储太多的零元素而浪费大量的内存空间。

在稀疏矩阵中，有一种情况非常常见，即稀疏矩阵内部存在对称性。这样，我们可以采用一维数组来表示它们，这也常称为压缩存储，如表4-2所示。

表4-2常见的特殊稀疏矩阵

稀疏矩阵	示意图	要点
上三角矩阵	 <p>图 4-1 上三角矩阵示意图</p>	<p>在矩阵第 i 行第 j 列的元素，对应的一维数组的下标计算公式为：$(2n-i-1) \times i/2+j$</p>
下三角矩阵	 <p>图 4-2 下三角矩阵示意图</p>	<p>在矩阵第 i 行第 j 列的元素，对应的一维数组的下标计算公式为：$(i+1) \times i/2+j$</p>

注：图4-1到图4-2中，着色的点表示为非零值，虚线是对角线。

3. 线性表

线性表是用来表示数据对象之间的线性结构，通俗地说，线性结构就是指所有结点是按“一个接着一个排列”的方式相互关联而组成一个整体。

线性结构是 n 个结点的有穷序列。通常表示为 (a_1, a_2, \dots, a_n) ， a_1 称为起始结点， a_n 称为结束结点， i 称为 a_i 在线性表中的序号或位置，线性表所含结点的个数称为线性表的长度，长度为0的线性表称为空表。

线性表主要的存储结构有两种：顺序存储结构和链式存储结构。采用顺序存储结构，就称为顺序表（常用数组实现）；采用链式存储结构则称为线性链表（即链表）。

（1）顺序表

顺序存储是最简单的存储方式，通常用一个数组，从数组的第一个元素开始，将线性表的结点依次存储在数组中，即线性表的第 i 个结点存储在数组的第 i （ $0 \leq i \leq n-1$ ）个元素中，用数组元素的顺序存储来体现线性表中结点的先后次序关系。

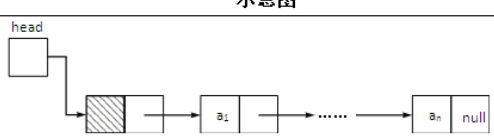
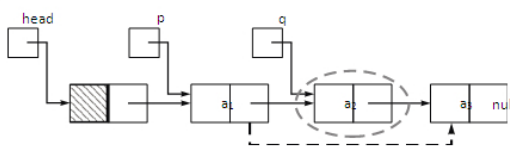
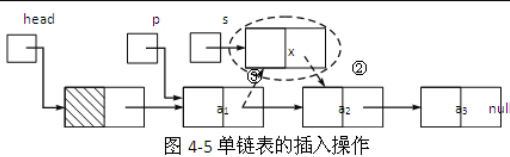
顺序存储线性表的最大优点就是能随机存取线性表中的任何一个结点，缺点主要有两个，一是数组的大小通常是固定的，不利于任意增加或减少线性表的结点个数；二是插入和删除线性表的结点时，要移动数组中的其他元素，操作复杂。

（2）链表

链表就是采用链式存储实现的线性表。它是动态分配链表结点，通过链接指针，将各个节点按逻辑顺序连接起来。根据其存储结构的不同，可以分为单链表、循环链表和双链表三种，软设目前主要考查前两种。

单链表（如表4-3所示）

表4-3单链表

项目	示意图	说明
定义	 图 4-3 单链表的定义	每个结点包括两个域：数据、指针（指向下一个结点），最后一个结点的指针为 null，head 表示头指针
删除	 图 4-4 单链表的删除操作	1. 找到要删除的结点 q 和该结点的前趋 p 2. 将 p 的指针指向 q 所指向的后继结点 $p \rightarrow next = q \rightarrow next$; 该操作的算法平均时间复杂度为 $O(n)$
插入	 图 4-5 单链表的插入操作 要注意，步骤 2 和 3 不能互换，那样将产生信息的丢失	1. 找到插入点的前趋 p 2. 将要插入的结点 x 的指针指向 p 的后继结点： $s \rightarrow next = p \rightarrow next$ 3. 将 p 的指针指向 x： $p \rightarrow next = s$ 该操作的算法平均时间复杂度为 $O(n)$

循环链表

循环链表与单链表的区别仅仅在于其尾结点的指针域值不是null，而是指向头结点的指针。这样做的好处是，从表中的任一结点出发都能够通过后移操作扫描整个循环链表。

（3）顺序表与链表的比较

在实际应用中，应该如何在顺序实现和链式实现中进行选择呢？通常是从时间和空间性能的角度来进行判断，如表4-4所示。

表4-4链表的空间性能和时间性能

性能类别	具体项目	顺序存储	链式存储
空间性能	存储密度	=1，更优	<1
	容量分配	事先确定	动态改变，更优
时间性能	定位运算	$O(n)$	$O(n)$
	读运算	$O(1)$ ，更优	$O([n+1]/2)$ ，最好情况为 1，最坏情况为 n
	插入运算	$O(n/2)$ ，最好情况为 0，最坏情况为 n	$O(1)$ ，更优
	删除运算	$O([n-1]/2)$	$O(1)$ ，更优

（4）队列

队列也是一种特殊的线性表，只允许在一端进行插入，另一端进行删除运算。允许删除运算的那一端称为队首，允许插入运算的一端称为队尾。称队列的结点插入为进队，结点删除为出队。因最先进入队列的结点将最先出队，所以队列具有先进先出的特征。

实现队列，可以使用顺序存储（如：数组方式）也可以用链表。

顺序存储

用顺序存储线性表来表示队列，为了指明当前执行出队运算的队首位置，需要一个指针变量 head（称为头指针），为了指明当前执行进队运算的队尾位置，也需要一个指针变量 tail（称为尾指针）。

若用有N个元素的数组表示队列，随着一系列进队和出队运算，队列的结点移向存放队列的数组的尾端，会出现数组的前端空着，而队列空间已用完的情况。一种可行的解决办法是当发生这样的情况时，把队列中的结点移到数组的前端，修改头指针和尾指针。另一种更好的解决办法是采用循环队列。

循环队列就是将实现队列的数组a[N]的第一个元素a[0]与最后一个元素a[N-1]连接起来。队空的初态为 head=tail=0。在循环队列中，当tail 赶上head时，队列满。反之，当head赶上tail时，队列变为空。这样队空和队满的条件都同为head=tail，这会给程序判别队空或队满带来不便。因此，可采用当队列只剩下一个空闲结点的空间时，就认为队列已满的简单办法，以区别队空和队满。即队空的判别条件是head=tail，队满的判别条件是head=tail+1。

链式存储

队列也可以用链接存储线性表实现，用链表实现的队列称为链接队列。链表的第一个结点是队列首结点，链表的末尾结点是队列的队尾结点，队尾结点的链接指针值为NULL。队列的头指针head 指向链表的首结点，队列的尾指针tail指向链表的尾结点。当队列的头指针head值为NULL时，队列为空。

(5) 栈

栈是另一种特殊的线性表，栈只允许在同一端进行插入和删除运算。允许插入和删除的一端称为栈顶，另一端为栈底。称栈的结点插入为进栈，结点删除为出栈。因为最后进栈的结点必定最先出栈，所以栈具有后进先出的特征。

顺序存储

采用顺序实现的栈中，初始化运算负责将栈顶变量top初始化为“0”，使栈为空；在进栈操作时，需要判断栈是否满（top=N说明栈满），如果未满，则将新元素插入栈，并将top的值加1；在出栈操作时，需要判断栈是否空（top=0说明栈空），如果非空，则取出栈顶元素，将top的值减1。

顺序栈的缺点在于为了避免栈满时发生溢出，需预先为栈设立足够大的空间，但太大会造成空间浪费，太小又容易引发溢出。

链式存储

栈也可以用链表实现，用链表实现的栈称为链接栈。链表的第一个结点为顶结点，链表的首结点就是栈顶指针top，top为NULL的链表是空栈。

(6) 字符串

字符串是由某字符集上的字符所组成的任何有限字符序列。当一个字符串不包含任何字符时，称它为空字符串。一个字符串所包含的有效字符个数称为这个字符串的长度。一个字符串中任一连续的子序列称为该字符串的子串。

字符串通常存于足够大的字符数组中，每个字符串的最后一个有效字符之后有一个字符串结束标志，记为“\0”。通常由系统提供的库函数形成的字符串的末尾会自动添加“\0”，但当由用户的应用程序来形成字符串时，必须由程序自行负责在最后一个有效字符之后添加“\0”，以形成字符串。

对字符串的操作通常有：

- 统计字符串中有效字符的个数；
- 把一个字符串的内容复制到另一个字符串中；
- 把一个字符串的内容连接到另一个足够大的字符串的末尾；
- 在一个字符串中查找另一个字符串或字符；
- 按字典顺序比较两个字符串的大小。

一点一练

试题1

对于一个长度大于1且不存在重复元素的序列，令其所有元素依次通过一个初始为空的队列后，再通过一个初始为空的栈。设队列和栈的容量都足够大，一个序列通过队列（栈）的含义是序列的每个元素都入队列（栈）且出队列（栈）一次且仅一次。对于该序列在上述队列和栈上的操作，正确的叙述是__ (1) __。

- (1) A. 出队序列和出栈序列一定相同
 B. 出队序列和出栈序列一定互为逆序
 C. 入队序列与出队序列一定相同，入栈序列与出栈序列不一定相同
 D. 入栈序列与出栈序列一定互为逆序，入队序列与出队序列不一定互为逆序

试题2

若二维数组arr[1..M, 1..N]的首地址为base，数组元素按列存储且每个元素占用K个存储单元，则元素arr[i, j]在该数组空间的地址为__ (2) __。

- (2) A. $\text{base} + ((i-1) * M + j - 1) * K$ B. $\text{base} + ((i-1) * N + j - 1) * K$
 C. $\text{base} + ((j-1) * M + i - 1) * K$ D. $\text{base} + ((j-1) * N + i - 1) * K$

试题3

对于线性表（由n个同类元素构成的线性序列），采用单向循环链表存储的特点之一是__ (3) __。

- (3) A. 从表中任意结点出发都能遍历整个链表
 B. 对表中的任意结点可以进行随机访问
 C. 对于表中的任意一个结点，访问其直接前驱和直接后继结点所用时间相同
 D. 第一个结点必须是头结点

试题4

设下三角矩阵（上三角部分的元素值都为0） $A[0..n, 0..n]$ 如图4-6所示，将该三角矩阵的所有非零元素（即行下标不小于列下标的元素）按行优先压缩存储在容量足够大的数组M[]中（下标从1开始），则元素 $A[i,j]$ ($0 \leq i \leq n, j \leq i$) 存储在数组M的__ (4) __中。

$$\begin{bmatrix} A_{0,0} & & & & & & & & \\ A_{1,0} & A_{1,1} & & & & & & & \\ \cdot & & \cdot & & & & & & \\ \cdot & & & \cdot & & & & & \\ \cdot & & & & & & & & \\ A_{7,0} & A_{7,1} & A_{7,2} & \cdots & & & A_{7,7} & & \\ A_{8,0} & A_{8,1} & A_{8,2} & A_{8,3} & \cdots & & & & A_{8,8} \end{bmatrix}$$

图4-6下三角矩阵

- (4) A. $M[\frac{i(i+1)}{2} + j + 1]$ B. $M[\frac{i(i+1)}{2} + j]$
 C. $M[\frac{i(i-1)}{2} + j]$ D. $M[\frac{i(i-1)}{2} + j + 1]$

试题5

设循环队列Q的定义中有rear和len两个域变量，其中rear表示队尾元素的指针，len表示队列的长度，如图4-7所示（队列长度为3，队头元素为e）。设队列的存储空间容量为M，则队头元素的指针为__ (5) __。

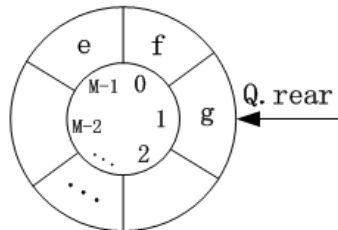


图4-7循环队列

- (5) A. $(Q.rear + Q.len - 1)$ B. $(Q.rear + Q.len - 1 + M) \% M$
 C. $(Q.rear - Q.len + 1)$ D. $(Q.rear - Q.len + 1 + M) \% M$

试题6

对于二维数组a[1..N,1..N]中的一个元素a[i,j] ($1 \leq i, j \leq N$)，存储在a[i,j]之前的元素个数__ (6) __。

- (6) A. 与按行存储或按列存储方式无关
 B. 在*i=j*时与按行存储或按列存储方式无关
 C. 在按行存储方式下比按列存储方式下要多
 D. 在按行存储方式下比按列存储方式下要少

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

解析与答案

试题1分析

本题主要考查队列和栈的特性。队列具有先进先出的特点，而栈具有后进先出的特点。因此我们可以知道入队序列与出队序列一定相同，但入栈序列与出栈序列不一定相同。比如a, b, c这样一个序列，那么按照a, b, c的顺序入队列，那么其出队列的次序一定是a, b, c。而按照a, b, c的顺序入栈，那么可能是a入栈后就出栈，然后b入栈又出栈，然后C入栈出栈。也可能是等a, b, c都入栈后再出栈，那么出栈序列就是c, b, a。

试题1答案

- (1) C

试题2分析

题目告诉我们是按列存储，那么在存储元素arr[i, j]以前，应该存放了j-1列，而每一列中有M个

元素（即数组的行数），那么应该有 $(j-1)*M$ 个元素，而在第 j 列中，存放元素 $arr[i, j]$ 以前，应该有 $i-1$ 个元素被存放，因此，在存放元素 $arr[i, j]$ 以前总共有 $(j-1)*M+i-1$ 个元素被存放，而每个元素占用 K 个存储单元，因此本题答案选 C。

试题2答案

(2) C

试题3分析

采用单向循环链表存储的特点之一是从表中任意结点出发都能遍历整个链表，另外便于元素的元素节点的删除与插入。如需要对表中的任意节点进行随机访问需采用顺序存储结构。

试题3答案

(3) A

试题4分析

本题考查数据结构基础知识。

如图4-6所示，按行方式压缩存储时， $A[i, j]$ 之前的元素数目为 $(1+2+\dots+i+j)$ 个，因为第 i 行前面的每行的元素个数分别为 $1, 2, 3, \dots, i$ 。数组 M 的下标从 1 开始，因此 $A[i, j]$ 的值存储在

$$M\left[\frac{i(i+1)}{2} + j + 1\right] \text{ 中。}$$

试题4答案

(4) A

试题5分析

对于循环队列，求队头元素的指针的计算公式为： $(rear-len+1+M)\%M$ 。

求队列中元素个数公式为： $(rear-fear+M)\%M$ 。其中 $fear$ 表示队列的对头指针。

试题5答案

(5) D

试题6分析

按行存储即先存储完第一行元素，再开始存储第二行元素，依次类推。按列存储即先存储完第一列元素，再开始存储第二列元素，依次类推。

因为按行存储和按列存储是两种不同的存储方式，因此在本题中，存储在 $a[i, j]$ 之前的元素个数与存储方式的选择有关。另外，由于二维数组 a 的行和列都是 n ，是相等的，那么在 $i=j$ （即行等于列）的情况下，存储元素的个数也是与存储方式无关的。

试题6答案

(6) B

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

树是一种典型的非线性数据结构，它能够很好地应用于描述分支和层次特性的数据集合。树是由一个或多个结点组成的有限集合 T ，它满足以下两个条件：

- (1) 有一个特定的结点，称为根结点；
- (2) 其余的结点分成 m ($m \geq 0$) 个互不相交的有限集合。其中每个集合又都是一棵树，称 T_1, T_2, \dots, T_{m-1} 为根结点的子树。

显然，以上定义是递归的，即一棵树由子树构成，子树又由更小的子树构成。由条件(1)可知，一棵树至少有一个结点(根结点)。一个结点的子树数目称为该结点的度(次数)，树中各结点的度的最大值称为树的度(树的次数)。度为0的结点称为叶子结点(树叶)，除叶子结点外的所有结点称为分支结点，根以外的分支结点称为内部结点。例如，在图4-8所示的树中，根结点的度数为3，结点2的度数为4，结点4的度数为1，结点9的度数为2，其他结点的度数为0，该树的度数4。

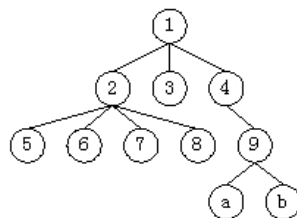


图4-8树

在用图形表示的树中，对两个用线段连接的相关联的结点而言，称位于上端的结点是位于下端的结点的父结点或双亲结点，称位于下端的结点是位于上端的结点的(孩)子结点，称同一父结点的多个子结点为兄弟结点，称处于同一层次上、不同父结点的子结点为堂兄弟结点。例如在图4-8中，结点1是结点2,3,4的父结点。反之，结点2,3,4都是结点1的子结点。结点2,3,4是兄弟结点，而结点5,6,7,8,9是堂兄弟结点。

定义一棵树的根结点所在的层次为1，其他结点所在的层次等于它的父结点所在的层次加1。树中各结点的层次的最大值称为树的层次。

2. 树的遍历

另外一个重点则是树的遍历问题，也就是根据某种顺序逐个获得树中全部结点的信息，常见的遍历方法有三种：

前序遍历：“根左右”，即先访问根结点，然后再从左到右按前序遍历各棵子树。以图4-8为例，前序遍历结果为：1, 2, 5, 6, 7, 8, 3, 4, 9, a, b。

后序遍历：“左右根”，即从左到右遍历根结点的各棵子树，最后访问根结点。以图4-8为例，后序遍历结果为：5, 6, 7, 8, 2, 3, a, b, 9, 4, 1。

层次遍历：首先访问处于0层的根结点，然后从左到右访问1层上的结点，以此类推，层层向下访问。以图4-8为例，层次遍历结果为：1, 2, 3, 4, 5, 6, 7, 8, 9, a, b

3. 二叉树的概念

二叉树是一个有限的结点集合，该集合或者为空，或者由一个根结点及其两棵互不相交的左、右二叉子树所组成。

二叉树的结点中有两棵子二叉树，分别称为左子树和右子树。因为二叉树可以为空，所以二叉树中的结点可能没有子结点，也可能只有一个左子结点(右子结点)，也可能同时有左右两个子结点。如图4-9所示是二叉树的4种可能形态(如果把空树计算在内，则共有5种形态)。

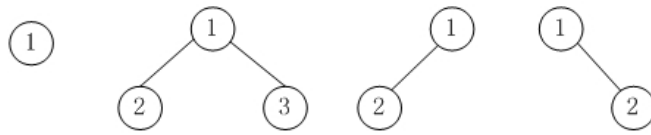


图4-9二叉树的4种不同形态

在二叉树中，有两种表现极为特殊，即满二叉树和完全二叉树，如图4-10所示。

满二叉树：一棵深度为 k 且有 2^k-1 ($k \geq 1$) 个结点的二叉树就称为满二叉树。

完全二叉树：如果深度为 k 、有 n 个结点的二叉树中各结点能够与深度 k 的顺序编号的满二叉树从1到 n 标号的结点相对应即为完全二叉树。

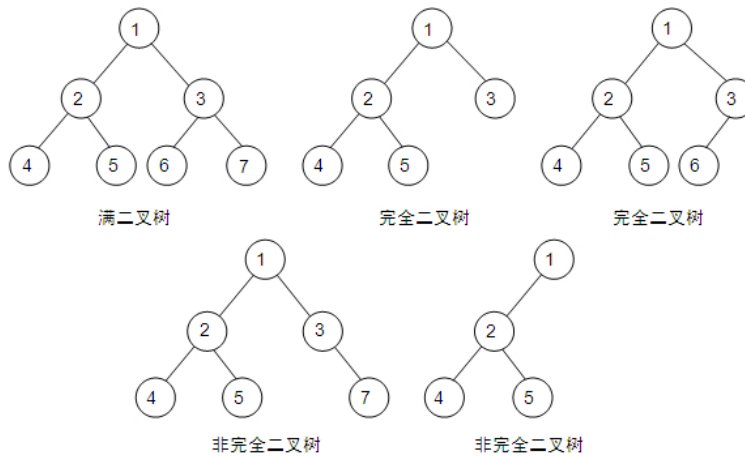


图4-10满二叉树和完全二叉树

二叉树具有以下几个重要性质，经常成为出题的依据。

在二叉树的第 i 层上最多有 2^{i-1} 个结点 ($i \geq 1$) ；

深度为 k 的二叉树最多有 2^k-1 个结点 ($k \geq 1$) ；

对任何一棵二叉树，如果其叶子结点数为 n_0 ，度为2的结点数为 n_2 ，则 $n_0 = n_2 + 1$ ；

具有 n 个结点的完全二叉树的深度为 $\lfloor \log_2 n \rfloor$ ， ($\lfloor m \rfloor$ 运算是表示大于等于 m 的整数) ；

如果对一棵有 n 个结点的完全二叉树的结点按层序编号（从第1层到 $\lfloor \log_2 n \rfloor + 1$ 层，每层从左到右），则对任一结点 i ($1 \leq i \leq n$) ，有：

如果 $i=1$ ，则结点 i 无父结点，是二叉树的根；如果 $i > 1$ ，则父结点是 $\lfloor i/2 \rfloor$ ；

如果 $2i > n$ ，则结点 i 为叶子结点，无左子结点；否则，其左子结点是结点 $2i$ ；

如果 $2i+1 > n$ ，则结点 i 无右子结点，否则，其右子结点是结点 $2i+1$ 。

在考试时，对这几个特性的灵活应用是十分关键的，因此应熟练掌握它们，其实这些性质都十分明显，只需绘制出二叉树，结合着体会将能更有效地记忆。

4. 二叉树的遍历

树的遍历方法也同样适用于二叉树（如右图4-11所示），不过由于二叉树的自身特点，还有一种中序遍历法。

前序遍历（根左右，先访问根结点，然后分别用前序分别遍历左、右子树，也称为前根遍历）。图4-11的前序遍历结果是：1, 2, 4, 5, 7, 8, 3, 6。

中序遍历（左根右，先按中序遍历左子树，再访问根结点，然后再按中序遍历右子树，也称为中根遍历）。图4-11的中序遍历的结果是：4, 2, 7, 8, 5, 1, 3, 6。

后序遍历（左右根，分别按后序遍历左、右子树，然后再访问根结点，也称为后根遍历）。

图4-11的后序遍历的结果是：4, 8, 7, 5, 2, 6, 3, 1。

层次遍历（首先访问处于0层的根结点，然后从左到右访问1层上的结点，以此类推，层层向下访问）。图4-11的层次遍历的结果是：1, 2, 3, 4, 5, 6, 7, 8。

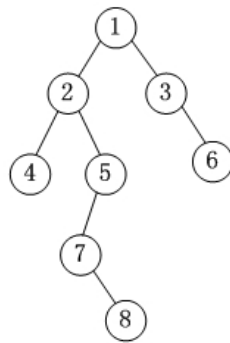


图4-11二叉树遍历

根据上面的定义和描述，我们可以发现遍历是递归定义的，最适合使用递归函数来实现。在学习遍历时，最重要的是结合其概念来灵活应用。

5. 二叉查找树

二叉查找树，它或者是一棵空树；或者是具有下列性质的二叉树：

- （1）若左子树不空，则左子树上所有结点的值均小于它的根结点的值；
- （2）若右子树不空，则右子树上所有结点的值均大于它的根结点的值；
- （3）左、右子树也分别为二叉查找树；

当对这样的二叉树进行中序遍历，就可以得到一个排好序的结点序列，因此，二叉查找树也称为二叉排序树。二叉查找树主要考查其遍历。

6. 平衡二叉树

平衡二叉树又被称为AVL树，它具有以下性质：它的左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树。满二叉树就是一种平衡二叉树。

7. 线索二叉树

二叉树在通常情况下是无法直接找到某结点在某种遍历序列中的前驱和后继结点的。而线索二叉树则通过利用二叉树上空的指针域来存放这些“线索”信息，通常其采用以下做法：

若前驱结点不为空，而且其右指针域为空，则将根结点的地址赋给前驱结点的右指针域，并将前驱结点的右线索标志置1；

若根结点的左指针域为空，则把前驱结点的地址赋给根结点的左指针域，同时将根结点的左线索标志置1；

将根结点地址赋给保存前驱结点指针的变量，以便当访问下一个结点时，此根结点成为前驱结点。

8. 哈夫曼树

在理解哈夫曼树之前，必须了解一些最基本的概念。

树的路径长度：是从树根到树中每一结点的路径长度之和，在结点数目相同的二叉树中，完全二叉树的路径长度最短；

权：在一些应用中会赋予树中结点一个有意义的实数，这个数字称为权；

带权路径长度：结点到树根之间的路径长度与该结点上权的乘积，称为结点的带权路径长度；

树的带树路径长度（树的代价）：所有叶结点的带树路径长度之和。

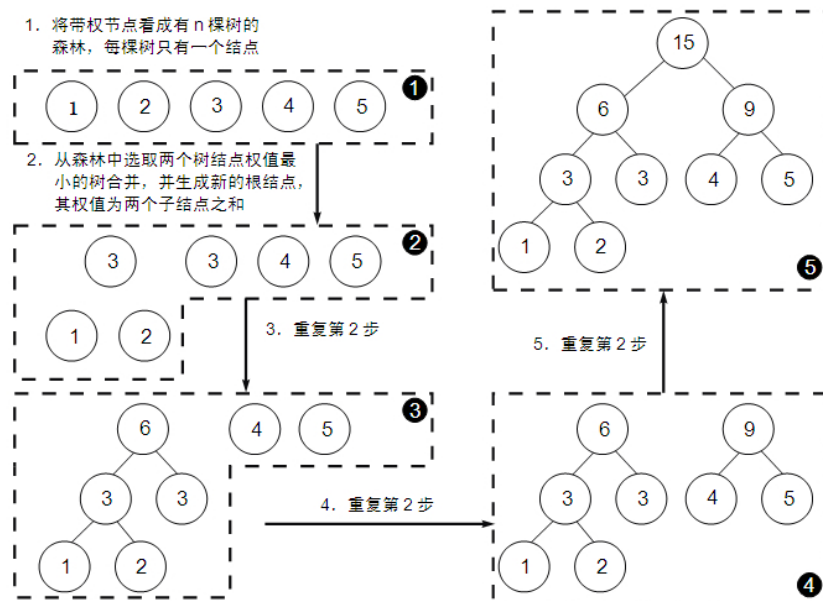


图4-12哈夫曼树构造过程示意图

而在权值相同的 n 个叶子结点构成的所有二叉树中，带权路径长度最小的二叉树称为最优二叉树，也称为哈夫曼树。构造哈夫曼树的过程如图4-12所示。

从图4-12中，我们可以发现每次都是选取最小权值的二叉树进行合并，因此它使用的是贪算法。而且，我们还可以发现，使用这种构造过程，哈夫曼树是不可能存在度为1的分支结点，而最初的 n 个节点将经过 $n-1$ 次合并，生成 $n-1$ 个新结点，因此哈夫曼树的总结点数是 $2n-1$ 的结点，而叶子结点数正是 n 。

版权方授权希赛网发布，侵权必究

上一节 本书简介 下一节

第 4 章：数据结构

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

一点一练

试题1

若 n_2 、 n_1 、 n_0 分别表示一个二叉树中度为2、度为1和叶子结点的数目（结点的度定义为结点的子树数目），则对于任何一个非空的二叉树，__ (1) __。

(1) A. n_2 一定大于 n_1 B. n_1 一定大于 n_0

C. n_2 一定大于 n_0 D. n_0 一定大于 n_2

试题2

一棵满二叉树，其每一层结点个数都达到最大值，对其中的结点从1开始顺序编号，即根结点编号为1，其左、右孩子结点编号分别为2和3，再下一层从左到右的编号为4、5、6、7，依此类推，每一层都从左到右依次编号，直到最后的叶子结点层为止，则用__ (2) __可判定编号为 m 和 n 的两个结点是否在同一层。

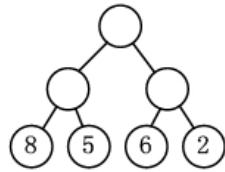
(2) A. $\log_2 m = \log_2 n$ B. $\lfloor \log_2 m \rfloor = \lfloor \log_2 n \rfloor$

C. $\lfloor \log_2 m \rfloor + 1 = \lfloor \log_2 n \rfloor$ D. $\lfloor \log_2 m \rfloor = \lfloor \log_2 n \rfloor + 1$

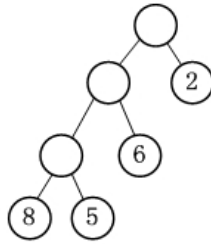
试题3

__ (3) __ 一 是由权值集合 {8, 5, 6, 2} 构造的哈夫曼树 (最优二叉树) 。

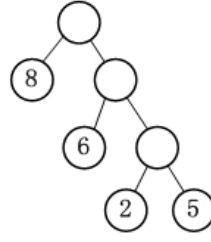
(3) A .



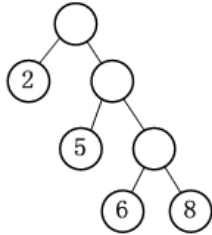
B .



C .



D .



试题4

在 __ (4) __ 中 , 任意一个结点的左、右子树的高度之差的绝对值不超过 1 。

(4) A . 完全二叉树

B . 二叉排序树

C . 线索二叉树

D . 最优二叉树

试题5

下面关于哈夫曼树的叙述中 , 正确的是 __ (5) __ 。

(5) A . 哈夫曼树一定是完全二叉树

B . 哈夫曼树一定是平衡二叉树

C . 哈夫曼树中权值最小的两个结点互为兄弟结点

D . 哈夫曼树中左孩子结点小于父结点、右孩子结点大于父结点

试题6

已知一棵度为 3 的树 (一个结点的度是指其子树的数目 , 树的度是指该树中所有结点的度的最大值) 中有 5 个度为 1 的结点 , 4 个度为 2 的结点 , 2 个度为 3 的结点 , 那么 , 该树中的叶子结点数目为 __ (6) __ 。

(6) A . 10

B . 9

C . 8

D . 7

版权方授权希赛网发布，侵权必究

[上一节](#)

[本书简介](#)

[下一节](#)

解析与答案

试题1分析

根据二叉树的性质 , 我们知道 $n_0 = n_2 + 1$, 因此在一棵二叉树中 , 叶子结点的数目一定是大于度为 2 的结点的个数。

试题1答案

(1) D

试题2分析

如果是满二叉树，那么其第n层的结点数应该是第n-1层结点的两倍，从根（第一层）开始，各层的结点数应分别是 2^{n-1} 个，其中n为当前的层次，因此一颗m层的满二叉树，其总的结点数位 2^m-1 个。而如果知道结点编号x，我们可以用 \log_2^{x+1} 来求取该结点属于那一层。

试题2答案

(2) B

试题3分析

构造哈夫曼树的过程是首先从给出的权值集合中找出最小的两个权值，即2和5，用它们作为子结点构建一个父结点，其权值为7，然后将7放入权值集合中并将2和5去掉，再在集合中找出两个最小权值，即6和7，而7已经在我们构造的树种，然后用6和7作为子结点构建一个父结点，其权值为 $6+7=13$ ，然后同样将13放入权值集合中并将6和7去掉，最好集合中只有8和13，将它们作为子结点构建一个父结点，就得到了这棵哈夫曼树。

试题3答案

(3) C

试题4分析

本题主要考查一些特殊二叉树的性质。

若二叉树中最多只有最下面两层的结点度数可以小于2，并且最下面一层的叶子结点都依次排列在该层最左边的位置上，则这样的二叉树称为完全二叉树，因此在完全二叉树中，任意一个结点的左、右子树的高度之差的绝对值不超过1。

二叉排序树的递归定义如下：二叉排序树或者是一棵空树；或者是具有下列性质的二叉树：

- (1) 若左子树不空，则左子树上所有结点的值均小于根结点的值；
- (2) 若右子树不空，则右子树上所有结点的值均大于根结点的值；
- (3) 左右子树也都是二叉排序树。

在n个结点的二叉树链式存储中存在n+1个空指针，造成了巨大的空间浪费，为了充分利用存储资源，可以将这些空链域存放指向结点在遍历过程中的直接前驱或直接后继的指针，这种空链域就称为线索，含有线索的二叉树就是线索二叉树。

最优二叉树即哈夫曼树。

试题4答案

(4) A

试题5分析

哈夫曼树是一种特殊的二叉树，但它不是完全二叉树，也不是平衡二叉树，给出n个权值

$\{w_1, w_2, \dots, w_n\}$ 构造一棵具有n个叶子结点的哈夫曼树的方法如下：

第一步，构造n个只有根结点的二叉树集合 $F=\{T_1, T_2, \dots, T_n\}$ ，其中每棵二叉树 T_i 的根结点带权为 w_i ($1 \leq i \leq n$)；

第二步，在集合F中选取两棵根结点的权值最小的二叉树作为左右子树，构造一棵新的二叉树，令新二叉树根结点的权值为其左、右子树上根结点的权值之和；

第三步，在F中删除这两棵二叉树，同时将新得到的二叉树加入到F中；

第四步，重复第二步和第三步，直到F只含有一棵二叉树为止，这棵二叉树便是哈夫曼树。

综上所述，我们可以知道哈夫曼树中权值最小的两个结点互为兄弟结点。

试题5答案

(5) C

试题6分析

由于叶子节点没有子树，因此它的度为0。而除根节点外，其它的节点都应该可以做为子节点，即可以用于计算度。

在本题中告诉我5个度为1的结点，4个度为2的结点，2个度为3的结点，那么树中总的度数为 $5+8+6=19$ ，因此树中除根节点外，就应该有19个节点，所以树中总的节点数应该为20，那么叶子节点数 $=20-5-4-2=9$ 。

试题6答案

(6) B

[版权方授权希赛网发布，侵权必究](#)

[上一节](#) [本书简介](#) [下一节](#)

第 4 章：数据结构

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

图

图是一种比树更复杂的非线性结构，它是由顶点集合V和边集合E组成的，通常记做 $G=(V, E)$ 。它可以用来模拟现实世界中许多图状结构的事物与问题。

[版权方授权希赛网发布，侵权必究](#)

[上一节](#) [本书简介](#) [下一节](#)

第 4 章：数据结构

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

考点精讲

1. 图的相关概念

有向图：若一个图中的每条边都是有方向的，则称为有向图。在有向图中， $\langle V_i, V_j \rangle$ 表示一条有向边， V_i 是始点（起点）， V_j 是终点。 $\langle V_i, V_j \rangle$ 和 $\langle V_j, V_i \rangle$ 表示的是两条不同的边。有向边也称为弧，边的始点称为弧头，终点称为弧尾。

无向图：若一个图中的每条边都是无方向的，则称为无向图。无向图的边是顶点的无序对，通常使用 (V_i, V_j) 来表示一条边，无向图的边没有起点和终点， (V_i, V_j) 和 (V_j, V_i) 表示的是同一条边。

无向完全图：如果限定任何一条边的两个顶点都不相同，则有n个顶点的无向图至多有 $n(n-1)/2$ 条边，这样的无向图称为无向完全图。

有向完全图：恰好有 $n(n-1)$ 条边的有向图称为有向完全图。

连通图：如果图中两个顶点间存在路径，则称它们是连通的；而如果图中任意两个顶点间都是连通的，则称该图为连通图。

2. 图的存储结构

图有两种主要的存储结构，它们是邻接矩阵表示法和邻接表表示法，如表4-5所示。

表4-5图的存储结构

	有向图	无向图
例子图		
邻接矩阵	有向图邻接矩阵 $\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$	无向图邻接矩阵 $\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$
邻接矩阵	特点： 1. 表示 n 个顶点的图，需要使用 $n \times n$ 矩阵表示 2. 用 $A[i,j]$ 的值表示从结点 i 到结点 j 是否存在边，存在置 1，不存在置 0 3. 利用邻接矩阵很容易求得各个顶点的度：无向图中顶点 v_i 的度是第 i 行（列）的元素之和；有向图中顶点 v_i 的出度是第 i 行元素之和，入度是第 i 列之和 4. 无向图对应的邻接矩阵是对称的，因此可以采用压缩存储法	
邻接表	有向图邻接表 	无向图邻接表
邻接表	特点： 1. 为每个顶点建立一个单链表， n 个顶点就要创建 n 个链表；除了头结点外，每个节点存放的都是其邻接的结点号 2. n 个顶点 e 条边的无向图，需要 n 个头结点和 $2e$ 个表结点；而 n 个顶点 e 条边的有向图，则需要 n 个头结点和 e 个表结点 3. 无向图中顶点 v_i 的度恰好是第 i 个单链表中的结点数；而有向图中，第 i 个单链表的结点数只是顶点 v_i 的出度，要求入度必须遍历所有链表	

3. 图的遍历

图的遍历也是从某个顶点出发，沿着某条搜索路径对图中每个顶点各做一次且仅做一次访问，常用的遍历算法包括以下深度优先和广度优先两种，如表4-6所示。

表4-6图的深度优先和广度优先

遍历方法	说明	示例	例子图
深度优先	1. 首先访问出发顶点 v 2. 依次从 v 出发搜索 v 的每个邻接点 w 3. 若 w 未访问过，则从该点出发继续深度优先遍历 它类似于树的前序遍历	$V_1, V_2, V_4, V_8, V_5, V_3, V_6, V_7$	
广度优先	1. 首先访问出发顶点 v 2. 然后访问与顶点 v 邻接的全部未访问顶点 w, x, y, \dots 3. 然后再依次访问 w, x, y, \dots 邻接的未访问的顶点	$V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8$	

4. 最小生成树

如果连通图 G 的一个子图是一棵包含 G 所有顶点的树，则该子图称为 G 的生成树。生成树是含有

该连通图全部顶点的一个极小连通子图，它并不是唯一的，从不同的顶点出发可以得到不同的子树。含有 n 个顶点的连通图的生成树有 n 个顶点和 $n-1$ 条边。

要求一个连通图的生成树很简单，只需从任何一个顶点出发，作一次深度优先或广度优先的搜索，将所经过的 n 个顶点和 $n-1$ 条边连接起来，就形成了极小连通子图，也就是一棵生成树。

对一个带权的图，在一棵生成树中，各条边的权值之和为这棵生成树的代价，其中代价最小的生成树称为最小生成树。普里姆算法（Prim算法）和克鲁斯卡尔算法（Kruskal算法）是求连通的带权无向图的最小代价树的常用算法。

注：带权的图是指每条边上带权值的图，常用于表示通路的代价。

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第 4 章：数据结构

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

一点一练

试题1

从存储空间的利用率角度来看，以下关于数据结构中图的存储的叙述，正确的是__(1)___。

- (1) A . 有向图适合采用邻接矩阵存储，无向图适合采用邻接表存储
B . 无向图适合采用邻接矩阵存储，有向图适合采用邻接表存储
C . 完全图适合采用邻接矩阵存储
D . 完全图适合采用邻接表存储

试题2

无向图中一个顶点的度是指图中与该顶点相邻接的顶点数。若无向图 G 中的顶点数为 n ，边数为 e ，则所有顶点的度数之和为__(2)___。

- (2) A . $n \cdot e$ B . $n + e$ C . $2n$ D . $2e$

试题3

设一个包含 N 个顶点、 E 条边的简单无向图采用邻接矩阵存储结构（矩阵元素 $A[i][j]$ 等于1/0分别表示顶点 i 与顶点 j 之间有/无边），则该矩阵中的非零元素数目为__(3)___。

- (3) A . N B . E C . $2E$ D . $N + E$

试题4

无向图中一个顶点的度是指图中__(4)___。

- (4) A . 通过该顶点的简单路径数 B . 通过该顶点的回路数
C . 与该顶点相邻的顶点数 D . 与该顶点连通的顶点数

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第 4 章：数据结构

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

解析与答案

试题1分析

本题主要考查图的存储结构，常见的图的存储结构有邻接矩阵存储和邻接表存储，其中在邻接矩阵存储方式中，矩阵中每个元素的值都表示两个点之间的边的信息，如果每两个点之间都有变的信息，那么矩阵中的所有元素都是有效元素，那么从存储空间的利用率角度来看，其利用率是100%，而采用邻接表存储其存储空间利用率肯定低于100%，因为采用邻接表存储，不仅要存储边的信息，还要存储节点信息，指针信息等。

这种情况下，这个图很显然是一个完全图，因此从存储空间的利用率角度来看，完全图适合采用邻接矩阵存储。

试题1答案

(1) C

试题2分析

在无向图中，一条边连接两个顶点，即如果存在一条边，那么与这条边相关的两个顶点的度都为加1，那么总的度就应该加2，因此，如果图中有n条边，那么所有顶点的度数之和就应该为2e。

试题2答案

(2) D

试题3分析

本题主要考查图的邻接矩阵存储结构。

设 $G=(V, E)$ 是具有n个顶点的图，其中V是顶点的集合，E是边的集合，那么邻接矩阵中的每个元素的定义如下：

$$A[i, j] = \begin{cases} 1 & (v_i, v_j) \in E \text{ 或 } \langle v_i, v_j \rangle \in E \\ 0 & (v_i, v_j) \notin E \text{ 或 } \langle v_i, v_j \rangle \notin E \end{cases}$$

从这个定义我们可以知道，一条边在矩阵中有两个1表示，比如顶点1和顶点2之间有一条边，那么矩阵元素A[1,2]和A[2,1]的值都是1。

在本题中，题目告诉我们有E条边，那么其邻接矩阵中的非零元素数目应该为2E。

试题3答案

(3) C

试题4分析

此题是纯概念题。

(1) 无向图中顶点V的度 (Degree)。

无向图中顶点V的度是关联于该顶点的边的数目，也可以说是直接与该顶点相邻的顶点个数，记为D(V)。

例如，在图4-13中， V_1 的度为1， V_2 的度为2， V_3 的度为3， V_4 的度为2。

(2) 有向图顶点V的入度 (InDegree)。

有向图中，以顶点V为终点的边的数目称为V的入度，记为ID(V)。

例如，在图4-14中， V_1 的入度为1， V_2 的入度为2， V_3 的入度为1， V_4 的入度为0。

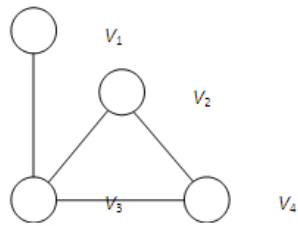


图4-13无向图示例

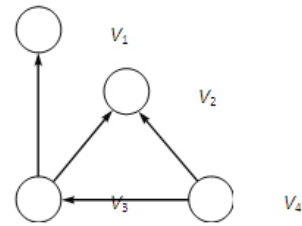


图4-14有向图示例1

(3) 有向图顶点V的出度 (OutDegree)

有向图中，以顶点V为始点的边的数目，称为V的出度，记为 $OD(V)$ 。

例如，在图4-15中， V_1 的出度为0， V_2 的出度为0， V_3 的出度为2， V_4 的出度也为2。

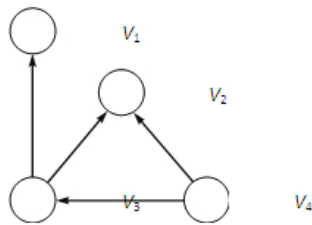


图4-15有向图示例2

试题4答案

(4) C

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

考前冲刺

试题1

单向链表中往往含有一个头结点，该结点不存储数据元素，一般令链表的头指针指向该结点，而该结点指针域的值则为第一个元素结点的指针。以下关于单链表头结点的叙述中，错误的是__ (1) __。

- (1) A. 若在头结点中存入链表长度值，则求链表长度运算的时间复杂度为 $O(1)$
- B. 在链表的任何一个元素前后进行插入和删除操作可用一致的方式进行处理
- C. 加入头结点后，代表链表的头指针不会因为链表为空而改变
- D. 加入头结点后，在链表中进行查找运算的时间复杂度为 $O(1)$

试题2

循环链表的主要优点是__(2)__。

- (2) A . 不再需要头指针了
- B . 已知某个节点的位置后, 能很容易找到它的直接前驱节点
- C . 在进行删除操作后, 能保证链表不断开
- D . 从表中任一节点出发都能遍历整个链表

试题3

设栈S和队列Q的初始状态为空, 元素按照a、b、c、d、e的次序进入栈S, 当一个元素从栈中出来后立即进入队列Q。若队列的输出元素序列是c、d、b、a、e, 则元素的出栈顺序是__(3)__, 栈S的容量至少为__(4)__。

- (3) A . a、b、c、d、e B . e、d、c、b、a
- C . c、d、b、a、e D . e、a、b、d、c
- (4) A . 2 B . 3 C . 4 D . 5

试题4

输入受限的双端队列是指元素只能从队列的一端输入, 但可以从队列的两端输出, 如图4-16所示。若有 8、1、4、2 依次进入输入受限的双端队列, 则得不到输出序列__(5)__。



图4-16输入受限的双端队列

- (5) A . 2、8、1、4 B . 1、4、8、2 C . 4、2、1、8 D . 2、1、4、8

试题5

若二叉树的先序遍历序列为ABDECF, 中序遍历序列为DBEAF C, 则其后序遍历序列为__(6)__。

- (6) A . DEBAFC B . DEFBCA C . DEBCFA D . DEBFCA

试题6

下面关于二叉排序树的叙述, 错误的是__(7)__。

- (7) A . 对二叉排序树进行中序遍历, 必定得到结点关键字的有序序列
- B . 依据关键字无序的序列建立二叉排序树, 也可能构造出单支树
- C . 若构造二叉排序树时进行平衡化处理, 则根结点的左子树结点数与右子树结点数的差值一定不超过1
- D . 若构造二叉排序树时进行平衡化处理, 则根结点的左子树高度与右子树高度的差值一定不超过1

试题7

下面关于二叉树的叙述, 正确的是__(8)__。

- (8) A . 完全二叉树的高度h与其结点数n之间存在确定的关系
- B . 在二叉树的顺序存储和链式存储结构中, 完全二叉树更适合采用链式存储结构
- C . 完全二叉树中一定不存在度为1的结点
- D . 完全二叉树中必定有偶数个叶子结点

试题8

若将某有序树T转换为二叉树T1, 则T中结点的后(根)序序列就是T1中结点的__(9)__遍历序列。例如, 图4-17(a)所示的有序树转化为二叉树后如图4-17(b)所示。

(9) A. 先序 B. 中序 C. 后序 D. 层序

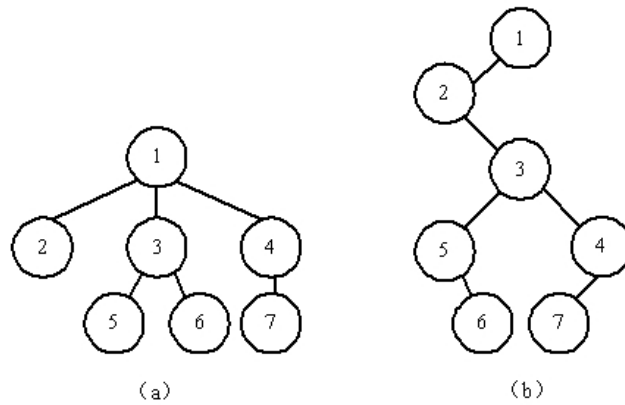


图4-17树转换二叉树

试题9

在平衡二叉树中，__(10)__。

- (10) A. 任意节点的左、右子树节点数目相同
- B. 任意节点的左、右子树高度相同
- C. 任意节点的左、右子树高度之差的绝对值不大于1
- D. 不存在度为1的节点

试题10

在如图4-18所示的平衡二叉树（树中任一结点的左右子树高度之差不超过1）中，结点A的右子树AR高度为h，结点B的左子树BL高度为h，结点C的左子树CL、右子树CR高度都为h-1。若在CR中插入一个结点并使得CR的高度增加1，则该二叉树__(11)__。

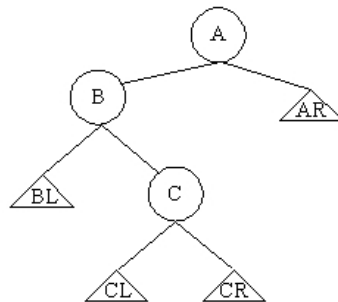


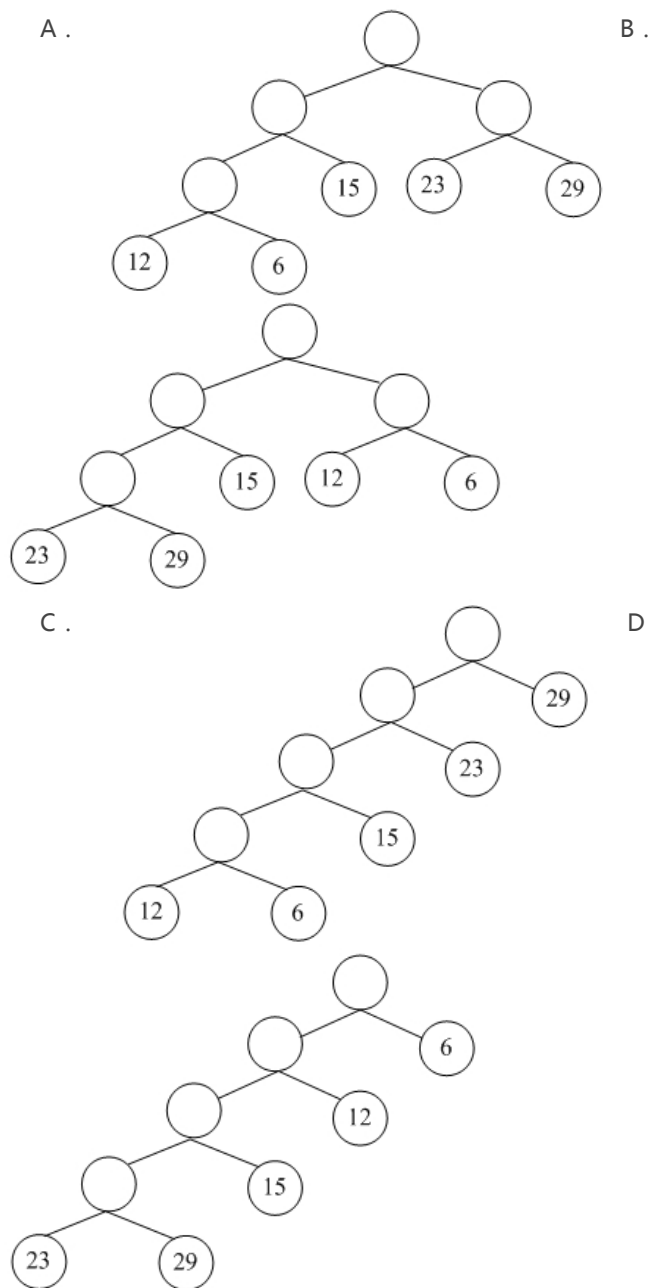
图4-18平衡二叉树

- (11) A. 以B为根的子二叉树变为不平衡
- B. 以C为根的子二叉树变为不平衡
- C. 以A为根的子二叉树变为不平衡
- D. 仍然是平衡二叉树

试题11

由权值为29、12、15、6、23的5个叶子结点构造的哈夫曼树为__(12)__，其带权路径长度为__(13)__。

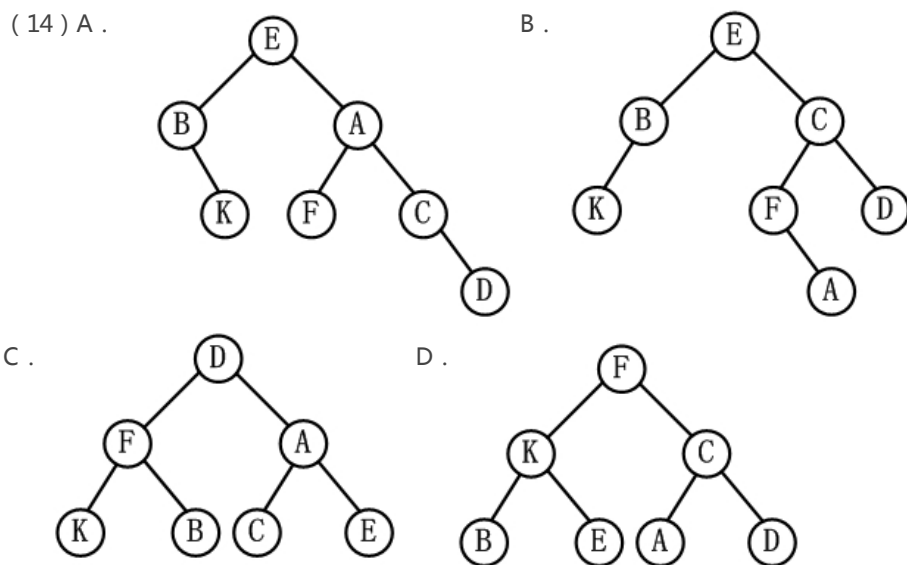
(12)



(13) A . 85 B . 188 C . 192 D . 222

试题12

若某二叉树的后序遍历序列为KBFDCAE，中序遍历序列为BKEFACD，则该二叉树为_(14)_____。



试题13

拓扑排序是将有向图中所有顶点排成一个线性序列的过程，并且该序列满足：若在AOV网中从顶点 V_i 到 V_j 有一条路径，则顶点 V_i 必然在顶点 V_j 之前。对于如图4-19所示的有向图，__(15)__是其拓扑序列。

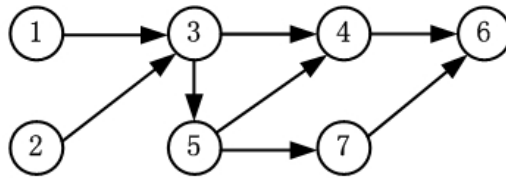


图4-19有向图

(15) A . 1234576 B . 1235467 C . 2135476 D . 2134567

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第4章：数据结构

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

习题解析

试题1分析

本题考查数据结构基础知识。

含有头结点的单链表如图4-20所示。

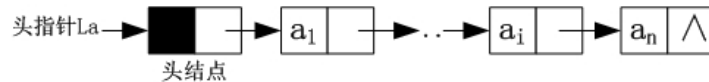


图4-20单链表

在链表中加入头结点后，查找表中某一元素仍然要从头指针出发，顺序找到目标元素或失败时找到表尾为止，时间复杂度与表长成正比。

试题1答案

(1) D

试题2分析

此题考查循环链表的基础知识，所以我们来了解一下什么是循环链表，一个带头节点的线性链表如图4-21所示。

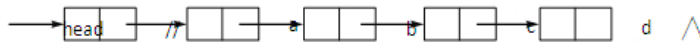


图4-21带头节点的线性链表

若将此链表的最后一个节点d的next域指向头，则产生了循环链表，如图4-22所示。

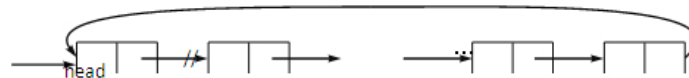


图4-22循环链表

实现循环链表的类型定义与线性链表完全相同，它的所有操作也都与线性链表类似。只是判断链表结束的条件有所不同。对照图4-22，我们现在来分析题目的备选答案。选项A “不再需要头指针

了”，言下之意就是线性链表一定需要头指针，但实际上不管是线性链表还是循环链表，头指针都是可要可不要的，所以选项A错误。再来看B选项，“已知某个节点的位置后，能很容易找到它的直接前驱节点”，题目中只说是循环链表，没有说是双向的循环链表。在单向循环链表中，已知某个节点的位置很难得到它的直接前驱节点，所以不对。接着看C选项，“在进行删除操作后，能保证链表不断开”，在线性链表中也能满足这个要求，所以不能算作循环链表的主要优点，也不正确。其实到这里我们已经可以知道答案为D了，但我们还是看看D到底对不对。D选项是这样的：“从表中任一节点出发都能遍历整个链表”。我们首先看看在线性链表中，是否能满足这个要求。以线性链表中c为例，c只能往右走到d，然后d的next域为空，无路可走，所以线性链表无法满足这个要求。再看循环链表，无论从哪一点出发，都可以到达任一节点，因为所有的节点围成了一个圈。所以此题的正确答案为D。

试题2答案

(2) D

试题3分析

栈 (stack) 在计算机科学中是限定仅在表尾进行插入或删除操作的线形表，它按照先进后出的原则存储数据，先进入的数据被压入栈底，最后的数据在栈顶，需要读数据时从栈顶开始弹出数据（最后一个数据被第一个读出来）。

队列是一种特殊的线性表，它只允许在表的前端 (front) 进行删除操作，而在表的后端 (rear) 进行插入操作。进行插入操作的端称为队尾，进行删除操作的端称为队头。队列中没有元素时，称为空队列。

第 (3) 空，根据队列的定义可知，它的输出元素序列是c、d、b、a、e，即出栈序列也是这个。

第 (4) 空，要求栈的大小，就是要看栈底元素到输出时栈中元素最多时的大小。由于入栈的顺序是a、b、c、d、e，而出栈的顺序是c、d、b、a、e，所以在栈中元素停留最多时为：a、b、c，则栈S的容量至少为3个。

试题3答案

(3) C (4) B

试题4分析

本题考查队列基本操作。可通过对备选答案逐个进行入队、出队操作验证其正确性。

对于输出序列2、8、1、4，其运算过程为：元素8、1、4、2依次进入队列，此时，元素2先出队列，元素8、1、4再依次出队，可得到输出序列2、8、1、4，但是在元素4和8出队列之前，元素1不能出队，所以得不到输出序列2、1、4、8。

对于输出序列1、4、8、2，其运算过程为：元素8、1先进入队列，然后元素1出队，元素4入队并出队，元素2入队并出队，最后元素8出队，得到输出序列1、4、8、2。

对于输出序列4、2、1、8，其运算过程为：元素8、1、4依次进入队列，然后元素4出队，元素2入队并出队，最后元素1和8依次出队，得到输出序列4、2、1、8。

试题4答案

(5) D

试题5分析

此题要求根据二叉树的先序遍历和中序遍历求后序遍历。我们可以根据这棵二叉树的先序和中

序遍历画出这棵二叉树。

习题解析



先看先序，先序遍历中第一个访问的节点是

根、左、右）。然后看中序，中序中A前面有节点D，D在E前面，后面有节点F，C。这说明D，B，E是A的左子树，F，C是A的右子树。我们再回到先序遍历中看D，B，E的排列顺序（此时可以不看其他节点），我们发现在先序中B排在最前，所以B是A左子树的根节点。接下来又回到了中序，中序中D在B前面，E在B后面，所以D是B的左子树，E是B的右子树。依此规则可构造二叉树，如图4-23所示。然后对这棵二叉树进行后序遍历得到DEBFCA。

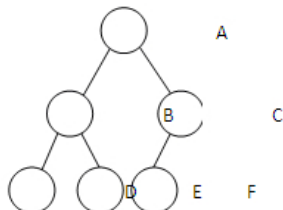


图4-23 二叉树

试题5答案

(6) D

试题6分析

本题考查数据结构方面的基础知识。

二叉排序树或者是一颗空树，或者是具有如下性质的二叉树：

- ①若它的左子树非空，则其左子树上所有节点的关键字均小于根节点的关键字；
- ②若它的右子树非空，则其右子树上所有节点的关键字均大于根节点的关键字；
- ③左、右子树本身就是两颗二叉排序树。

由上述定义可知，二叉排序树是一个有序表，对二叉排序树进行中序遍历，可得到一个关键字递增排序的序列。

对于给定的关键字序列，可从空树开始，逐个将关键字插入树中来构造一颗二叉排序树。其过程是：每读入一个关键字值，就建立一个新节点。若二叉排序树非空，则将新节点的关键字与根节点的关键字相比较，如果小于根节点的值，则插入到左子树中，否则插入到右子树中；若二叉排序树为空树，则新节点作为二叉排序树的根节点。

显然，若关键字初始序列已经有序，则构造出的二叉排序树一定是单枝树（每个节点只有一个孩子）。

为了使在二叉排序树上进行的查找操作性能最优，构造二叉排序树时需要进行平衡化处理，使每个节点左、右子树的高度差的绝对值不超过1。

试题6答案

(7) C

试题7分析

本题考查数据结构方面的基础知识。

根据其定义，一棵完全二叉树除了最后一层外，其余层的节点数都是满的，最后一层的节点也必须自左至右排列，如图4-24，(a)是高度为3的满二叉树，图(b)是完全二叉树，图(c)不是完全二叉树。

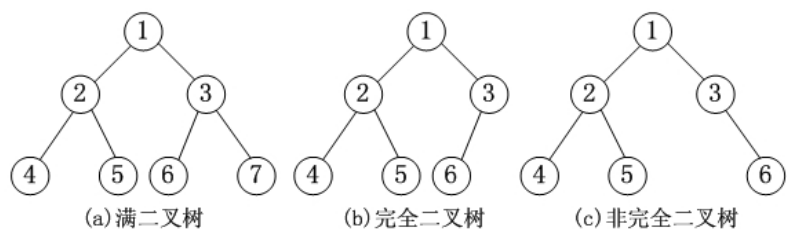


图4-24二叉树

二叉树采用顺序存储结构时，对于编号为 i 的节点，则有：

若 $i = 1$ 时，该节点为根节点，无双亲；

若 $i > 1$ 时，该节点的双亲节点为 $\lfloor i/2 \rfloor$ ；

若 $2i \leq n$ ，则该节点的左孩子编号为 $2i$ ，否则无左孩子；

若 $2i+1 \leq n$ ，则该节点的右孩子编号为 $2i+1$ ，否则无右孩子。

图（d）为具有10个节点的完全二叉树及其顺序存储结构，图（e）为某非完全二叉树的顺序存储结构，从中可以看出，完全二叉树适合采用顺序存储结构。

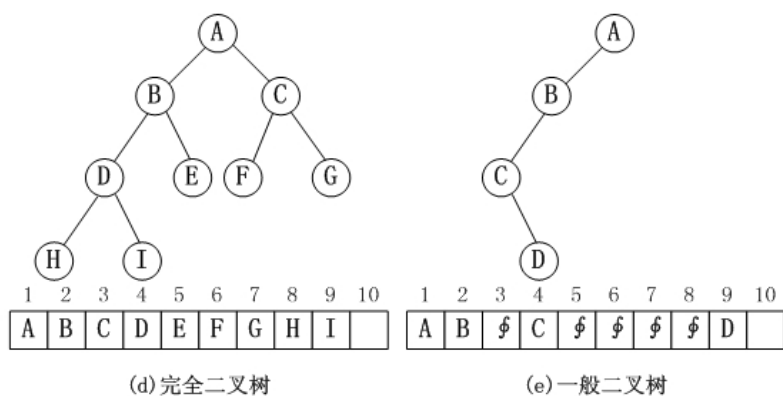


图4-25二叉树存储

可以推导出具有 n 个节点的完全二叉树的深度为 $\lfloor \log_2 n \rfloor + 1$ 。

试题7答案

(8) A

试题8分析

本题考查数据结构的树转换成二叉树，是常考的知识点。

树转换成二叉树的规则是：树中某结点 M 的孩子结点，在生成二叉树后放在 M 结点的左孩子位置；而 M 的兄弟结点，在生成二叉树后放在 M 结点的右孩子位置。所以如图4-17（a）所示的有序树的后根序列为：2-5-6-3-7-4-1；如图4-17（b）所示的二叉树的中序遍历为：2-5-6-3-7-4-1。

所以第（9）空选择B答案。

试题8答案

(9) B

试题9分析

平衡二叉树（Balanced Binary Tree）又称AVL树，它或者为空树，或者左、右子树均为平衡二叉树，且左、右子树的深度之差的绝对值不超过1。所以此题应选C。

试题9答案

(10) C

试题10分析

依据平衡二叉树的定义，树中任一结点的左右子树高度不超过1。

在没加结点前，是一棵平衡二叉树，结点C的CL和CR左右子树的高度为 $h-1$ ，即B结点的右子树BR（C结点为顶点）的高度为 h 。还有A结点的右子树AR的高度为 h ，AL（B结点为顶点）子树的高度为 $h+1$ ，满足定义。

当CR加入一个结点并使得CR的高度增加1以后：

（1）以C为顶点的子树仍为一棵平衡二叉树，因为CL为 $h-1$ ，而CR为 h ，相差1，满足定义。

（2）以B为顶点的子树仍为一棵平衡二叉树，因为BL为 h ，而BR（以C为顶点）子树的高度为 $h+1$ ，相差1，满足定义。

（3）以A为顶点的二叉树就不平衡了，因为AR的高度为 h ，但AL（以B为顶点）子树的高度为 $h+2$ 了，相差2，不满足定义。

所以本题的正确答案是C答案。

试题10答案

（11）C

试题11分析

最优树，又称哈夫曼树（Huffman Tree），是一类带权路径长度最短的树。

哈夫曼算法：

（1）根据给定的 n 个权值 $\{W_1, W_2, \dots, W_n\}$ ，构成 n 棵二叉树的集合 $F=\{T_1, T_2, \dots, T_n\}$ ，其中每棵二叉树 T_i 中只有一个带权为 W_i 的根结点，其左右子树均空。

（2）在 F 中选取两棵根结点的权值最小的树作为左、右子树，构造一棵新的二叉树，且置新的二叉树的根结点的权值为其左、右子树上根结点的权值之和。

（3）在 F 中删除这两棵树，同时将新得到的二叉树加入 F 中。

重复（2）和（3），直到 F 只含一棵树为止。这棵树便是所求的哈夫曼树。

满足题目要求的是A答案： $(12+6)*3+(15+23+29)*2 = 188$ ，带权路径长度为188。

试题11答案

（12）A （13）B

试题12分析

本题考查二叉树的遍历。

二叉树的主要遍历方式有：前序遍历、中序遍历、后序遍历、层次遍历。如果已知中序遍历，并知道前序遍历与后序遍历中的任意一个，便可得到一棵唯一的二叉树。

具体是怎么做的呢？

利用的是遍历的特点。中序遍历的顺序是：左、根、右。而后序遍历的顺序是：左、右、根。

回到题目里面来，从“后序遍历序列为KBFDCAE”，可以得知，二叉树的根结点为：E（此时已经可以排除选项C与选项D了）。继续分析，由“中序遍历序列为BKEFACD”，可以得知，二叉树的左子树包括结点：BK。右子树包括结点：FACD。

重复上面的步骤，对左子树与右子树看成独立的两棵树进行分析。在后序遍历中，左子树的结点BK的顺序为“KB”，所以B是根结点；右子树的结点FACD的顺序为“FDCA”，所以右子树的根结点为A。当分析到这一步时，已经可以得到本题答案为A。

试题12答案

（14）A

试题13分析

本题考查数据结构中的拓扑排序。

拓扑排序通俗一点来讲，其实就是依次遍历没有前驱结点的结点。而某一时刻没有前驱结点的结点有可能存在多个，所以一个图的拓扑排序可能有多。

以本题为例，1号结点与2号结点都没有前驱结点，所以拓扑排序的第一个元素可以是1，也可以是2。当1与2都访问完了，便可访问3号结点，3号结点访问完了，便可访问5号结点，访问完5号结点，可访问4号，或是7号结点。

所以拓扑排序结果为：（ 1 2 ） 3 5 （ 4 7 ） 6。括号中有多个数字，则代表在这多个数字的顺序可以变化。

这样，具体的拓扑排序结果为：1235476、1235746、2135476、2135746。

试题13答案

（ 1 5 ） C

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第 5 章：数据库系统基础知识

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

考点突破

根据考试大纲，本章要求考生掌握以下几个方面的知识。

- （ 1 ）数据库模型（概念模式、外模式、内模式）
- （ 2 ）数据模型，ER图，规范化
- （ 3 ）数据操作
- （ 4 ）数据库语言
- （ 5 ）数据库管理系统的功能和特征
- （ 6 ）数据库的控制功能
- （ 7 ）数据仓库和分布式数据库基础知识

从历年的考试情况来看，本章的考点主要集中在：E-R模型、关系代数、元组演算、规范化理论（键、范式、模式分解）、SQL语言等。

版权方授权希赛网发布，侵权必究

[上一节](#) [本书简介](#) [下一节](#)

第 5 章：数据库系统基础知识

作者：希赛教育软考学院 来源：希赛网 2014年05月04日

数据库模式及ER模型

数据库是长期存储在计算机内的、有组织的、可共享的数据集合，数据库系统是指在计算机信息系统中引入数据库后的系统，一般由数据库、数据库管理系统（ DataBase Management System，DBMS）、应用系统、数据库管理员（ DataBase Administrator，DBA）和用户构成。数