

第3章 基本数据结构

数据结构是指相互之间存在一种或多种特定关系的数据元素集合,是带有结构的数据元素的集合,它指数据元素之间的相互关系,即数据的组织形式。根据数据元素间关系的不同特性,通常有下列4类基本的数据结构。

- 集合结构。结构中的数据元素间除了同属于一个集合的关系外,无任何其他关系。
- 线性结构。结构中的数据元素间存在着一对一的线性关系。
- 树状结构。结构中的数据元素间存在着一对多的层次关系。
- 图状结构或网状结构。结构中的数据元素间存在着多对多的任意关系。

数据结构的定义形式为:数据结构是一个二元组 $\text{Data_Structure}=(D,R)$,其中 D 是数据元素的有限集, R 是 D 上关系的有限集。

上述数据结构定义的是对操作对象的一种数学描述,结构中定义的“关系”描述的是数据元素之间的逻辑关系,因此称为数据的逻辑结构。存储结构(又称物理结构)是逻辑结构在计算机中的存储映像,是逻辑结构在计算机中的实现,它包括数据元素的表示和关系的表示。逻辑结构与存储结构的关系为:存储结构是逻辑关系的映像与元素本身映像。逻辑结构是抽象,存储结构是实现,两者综合起来建立了数据元素之间的结构关系。

数据元素之间的关系在计算机中有两种不同的表示方法:顺序存储结构和链式存储结构。

在过去的10年中,该部分考试的分值如表3-1所示。

表 3-1 2009—2018 年基本数据结构知识考试题目统计表

年份	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
分值	10.5	12	7.5	6	10.5	9.5	14	11	7.5	8

分值的历史趋势变化如图3-1所示。

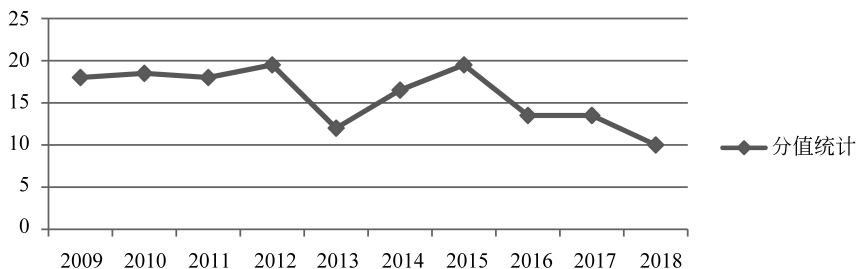


图 3-1 2009—2018 年数据结构知识分值统计分值历史趋势图

从该图可以看出,该部分的考试分值近年来略有下降,一般考查3~5道题,分值为7~8分。但该部分的内容相对比较简单,知识点变化不大,所以容易得分。

3.1 线 性 表

3.1.1 基本知识介绍

线性表(Linear List)是由 $n(n \geq 0)$ 个类型相同的数据元素 $a_1, a_2, \dots, a_n$ 组成的有限序列,记为 $(a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n)$ 。这里的数据元素 $a_i (1 \leq i \leq n)$ 只是一个抽象的符号,其具体含义在不同情况下也不同,它既可以是原子类型,也可以是结构类型,但同一线性表中的数据元素必须属于同一数据对象。此外,线性表中相邻数据元素之间存在着序偶关系,即对于非空的线性表,表中 a_{i-1} 领先于 a_i ,称 a_{i-1} 是 a_i 的直接前驱,而称 a_i 是 a_{i-1} 的直接后继。除了第一个元素 a_1 外,每个元素 a_i 有且仅有一个被称为其直接前驱的节点 a_{i-1} ,除了最后一个元素 a_n 外,每个元素 a_i 有且仅有一个被称为其直接后继的节点 a_{i+1} 。线性表中元素的个数 n 被定义为线性表的长度, $n=0$ 时称为空表。

线性表的特点可概括如下。

- 同一性: 线性表由同类数据元素组成,每一个 a_i 必须属于同一数据对象。
- 有穷性: 线性表由有限个数据元素组成,表长度就是表中数据元素的个数。
- 有序性: 线性表中相邻数据元素之间存在着序偶关系 $\langle a_i, a_{i+1} \rangle$ 。

由此可以看出,线性表是一种最简单的数据结构,因为数据元素之间是由“一前驱一后继”的直观有序的关系确定的;线性表也是一种最常见的数据结构,因为矩阵、数组、字符串、栈、队列等都符合线性条件。

线性表根据数据存储的不同分为顺序表和链表。

顺序表指用一组地址连续的存储单元依次存储线性表的数据元素,称为线性表的顺序存储结构或顺序映像(sequential mapping),它以“物理位置相邻”表示线性表中数据元素之间的逻辑关系,可随机存取表中任一元素。图 3-2 表示一个具体的顺序表的例子。

地址	1	2	3	4	5
数据	3	5	7	9	11

图 3-2 顺序表的例子

在使用链表结构表示数据元素 a_i 时,除了存储 a_i 本身的信息之外,还需要一个存储指示其后继元素的存储位置,这两个部分组成了 a_i 的存储映像,通常称为节点(node)。

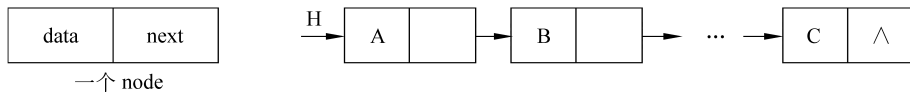


图 3-3 链表的例子

其中, data 域为数据域, 用来存储节点的值; next 域为指针域, 用来存储节点的直接后继的存储地址(位置)。n 个节点组成一个链表, 即线性表($a_1, a_2, \dots, a_n$)的链式存储结构, 其抽象表示如图 3-3 所示。

3.1.2 历年真题解析

题目 1 2016 年第 10 题(顺序表)

以下关于字符串的判定语句中正确的是()。

- A. 字符串是一种特殊的线性表
- B. 串的长度必须大于零
- C. 字符串不可以用数组表示
- D. 空格字符组成的串就是空串

解析: 字符串(String)是由数字、字母、下画线组成的一串字符。一般记为 $s = a_1, a_2, \dots, a_n (n \geq 0)$, 它是编程语言中表示文本的数据类型。在程序设计中, 字符串(string)为符号或数值的一个连续序列。

本题考查字符串的特点, 首先字符串是一种特殊的线性表, 串的长度可以为 0, 既可以用数组存储, 也可以用链表存储。空串是没有任何字符的串, 与空格字符组成的串有本质的区别。

参考答案: A

题目 2 2015 年第 13 题(链表)

链表不具备的特点是()。

- A. 可随机访问任何一个元素
- B. 插入、删除操作不需要移动元素
- C. 无须事先估计存储空间的大小
- D. 所需存储空间与存储元素个数成正比

解析: 链表中的元素在内存中不是顺序存储的, 而是通过存在元素中的指针联系到一起的。如果要访问链表中一个元素, 则需要从第一个元素开始, 一直找到需要的元素位置。但是增加和删除一个元素对于链表数据结构就非常简单了, 只要修改元素中的指针就可以了。如果应用需要经常插入和删除元素, 则利用链表的效率非常高。

本题中 A 是数组的特点, 不是链表的特点, 其他选项都是链表的特点。

参考答案: A

题目 3 2015 年第 14 题(链表)

线性表若采用链表存储结构, 则要求内存中可用存储单元地址()。

- A. 必须连续
- B. 部分地址必须连续
- C. 一定不连续
- D. 连续不连续均可

解析: 链表利用指针域确定下一个元素的位置, 所以存储单元地址连续不连续均可。

参考答案: D

题目 4 2014 年第 10 题(链表)

链表不具有的特点是()。

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

解析：参考题目1的解析。

参考答案：B

题目5 2013年第5题(哈希表)

将(2,6,10,17)分别存储到某个地址区间为0~10的哈希表中,如果哈希函数 $h(x) = (\quad)$,将不会产生冲突,其中 $a \bmod b$ 表示 a 除以 b 的余数。

- A. $x \bmod 11$
- B. $x^2 \bmod 11$
- C. $2x \bmod 11$
- D. $\lfloor \sqrt{2} \rfloor \bmod 11$, 其中 $\lfloor \sqrt{2} \rfloor$ 表示 $\sqrt{x}$ 向下取整

解析：哈希表是根据关键码值(Key value)而直接进行访问的,它通过把关键码值映射到表中的一个位置访问记录,以加快查找的速度,这个映射函数称为哈希函数。

本题中,利用哈希函数 $h(x)$ 取得哈希地址,各选项计算的答案分别如下。

- A. (2,6,10,6)有冲突,其中 $17 \bmod 11 = 6$
- B. (4,3,1,3)有冲突,其中 $17^2 \bmod 11 = 3$
- C. (4,1,9,1)有冲突,其中 $2 \times 17 \bmod 11 = 1$
- D. (1,2,3,4)无冲突,其中 $\lfloor \sqrt{17} \rfloor \bmod 11 = 4$

参考答案：D

题目6 2010年第16题(链表)

双向链表中有两个指针域llink和rlink,分别指向该节点的前驱及后继。设p指向链表中的一个节点,它的左右节点均非空。现要求删除节点p,则下面语句序列中错误的是()。

- A. $p \rightarrow rlink \rightarrow llink = p \rightarrow rlink$;
 $p \rightarrow llink \rightarrow rlink = p \rightarrow llink$; delete p;
- B. $p \rightarrow llink \rightarrow rlink = p \rightarrow rlink$;
 $p \rightarrow rlink \rightarrow llink = p \rightarrow llink$; delete p;
- C. $p \rightarrow rlink \rightarrow llink = p \rightarrow llink$;
 $p \rightarrow rlink \rightarrow llink \rightarrow rlink = p \rightarrow rlink$; delete p;
- D. $p \rightarrow llink \rightarrow rlink = p \rightarrow rlink$;
 $p \rightarrow llink \rightarrow rlink \rightarrow llink = p \rightarrow llink$; delete p;

解析：本题双向链表的示意图如图3-4所示。

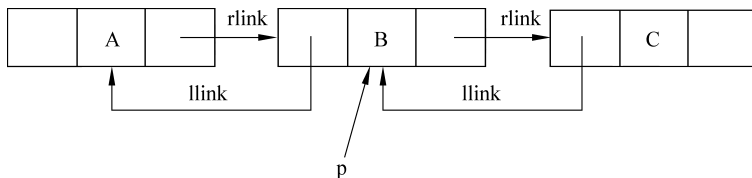


图3-4 双向链表示意图

根据双向链表示意图,将各选项模拟一遍可以发现选项A不能实现,其他选项都能

够实现。

参考答案: A

题目7 2009年第16题(顺序表)

有一个由4000个整数构成的顺序表,假设表中的元素已经按升序排列,若采用二分查找法定位一个元素,则最多需要()比较就能确定是否存在所要查找的元素。

- A. 11次 B. 12次 C. 13次 D. 14次

解析: 二分查找也称折半查找(Binary Search),每次查找都去除一半数据,是一种高效的查找方法,但该算法有两个先决条件: ①必须采用顺序存储结构; ②必须按关键字大小有序排列。

二分查找每次将表中间位置记录的关键字与查找关键字比较,如果两者相等,则查找成功;否则利用中间位置记录将表分成前、后两个子表,如果中间位置记录的关键字大于查找关键字,则进一步查找前一子表,否则进一步查找后一子表。重复以上过程,直到找到满足条件的记录使查找成功或子表不存在为止,此时查找不成功。

二分查找的最大比较次数是 $\log_2(n)$ 取整数,该题即 $\log_2(4000) = 12$ 。

参考答案: B

3.1.3 知识点巩固

从历年考点角度分析,本节的考点主要分为顺序表、链表、哈希表三个知识点。具体的出题数目如表3-2所示。

表3-2 历年知识点出现次数统计表

知识点	顺序表	链表	哈希表
出现个数	2	4	1
本节占比	28.57%	57.14%	14.29%

从历年知识点的出现次数可以看出,链表知识点所占分值最高,一方面是因为链表知识点有一定难度,所考知识点较多;另一方面是因为链表知识点在程序设计题目中出现的次数较少。其他两个知识点的考查比较随机。

根据以上知识点,本书提供几道习题供大家复习巩固。

1. 在一个长度为 n 的顺序表中删除第 i 个元素($1 \leq i \leq n$)时,需要向前移动()个元素。

- A. n B. $i-1$ C. $n-i$ D. $n-i+1$

2. 若某线性表最常用的操作是存取任一指定序号的元素和在最后进行插入与删除运算,则利用()存储方式最节省时间。

- A. 顺序表 B. 双链表
C. 带头节点的双循环链表 D. 单循环链表

3. 设线性表中一共有 $2n$ 个元素,则下列()选项的操作适合用链表存储。

- A. 删除所有值为 x 的元素

- B. 在最后一个元素的后面插入一个新元素
C. 顺序输出前 k 个元素
D. 交换第 i 个元素和第 $2n-i-1$ 个元素的值($i=0,1,\dots,n-1$)
4. 在双链表中向 p 所指的节点之前插入一个节点 q 的操作为()。
- A. $p \rightarrow \text{prior} = q; q \rightarrow \text{next} = p; p \rightarrow \text{prior} \rightarrow \text{next} = q; q \rightarrow \text{prior} = p \rightarrow \text{prior};$
B. $q \rightarrow \text{prior} = p \rightarrow \text{prior}; p \rightarrow \text{prior} \rightarrow \text{next} = q; q \rightarrow \text{next} = p; p \rightarrow \text{prior} = q \rightarrow \text{next};$
C. $q \rightarrow \text{next} = p; p \rightarrow \text{next} = q; q \rightarrow \text{prior} \rightarrow \text{next} = p; q \rightarrow \text{next} = p;$
D. $p \rightarrow \text{prior} \rightarrow \text{next} = q; q \rightarrow \text{next} = p; q \rightarrow \text{prior} = p \rightarrow \text{prior}; p \rightarrow \text{prior} = q;$
5. 以下数据结构中,()平均获取任意一个指定数据的速度最快。
- A. 二叉排序树 B. 队列 C. 栈 D. 哈希表
6. 对于线性表(7,34,55,25,64,46,19,10)进行散列存储时,使用 $H(K) = ()$ 作为散列函数最合适。
- A. $K \% 9$ B. $K \% 10$ C. $K \% 11$ D. $K \% 12$

3.2 栈和队列

3.2.1 基本知识介绍

栈(Stack)是限定在表的一端进行插入和删除运算的线性表,通常将插入、删除的一端称为栈顶(top),将另一端称为栈底(bottom)。不含元素的空表称为空栈。

假设栈 $S = (a_1, a_2, \dots, a_n)$,若栈中元素按 $a_1, a_2, \dots, a_n$ 的次序进栈,其中 a_1 为栈底元素, a_n 为栈顶元素,而退栈的次序却是 $a_n, a_{n-1}, \dots, a_1$ 。也就是说,栈的修改是按后进先出的原则进行的。因此,栈又称后进先出(Last In First Out)的线性表,简称 LIFO 表,如图 3-5 所示。栈在现实生活中也有很多例子,比如作业的批改和发放就是入栈与出栈的操作。

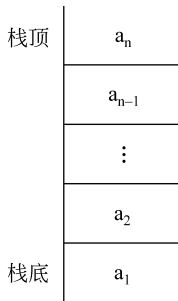


图 3-5 栈示意图

队列(Queue)也是一种受限的线性表,它只允许在表的一端进行元素的插入,而在另一端进行元素的删除。允许插入的一端称为队尾(rear),允许删除的一端称为队头(front)。

在队列中,通常把元素的插入称为入队,把元素的删除称为出队。队列概念与现实生活中的排队相似,新来的成员总是排在队尾,排在队列最前面的成员总是最先离开队列,即先进先出,因此又称队列为先进先出(First In First Out, FIFO)表。

假设队列 $q = (a_1, a_2, \dots, a_n)$,在空队列情况下,依次加入元素 $a_1, a_2, \dots, a_n$ 之后, a_1 就是队头元素, a_n 则是队尾元素。退出队列也是按此顺序进行的,也就是说,只有在 $a_1, a_2, \dots, a_{n-1}$ 都出队之后, a_n 才能出队。队列的示意图如图 3-6 所示。

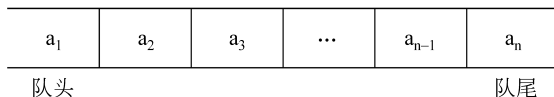
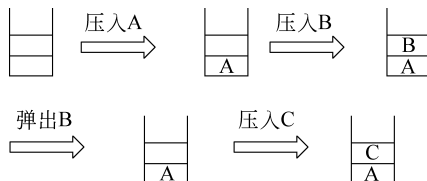


图 3-6 队列示意图

3.2.2 历年真题解析

题目 1 2018 年第 15 题(栈)

下图所使用的数据结构是()。



- A. 哈希表 B. 栈 C. 队列 D. 二叉树

解析: 该题比较简单, 根据压入和弹出数据的特点: 先进后出, 这是栈的典型特点, 所以可知该数据结构为栈。

参考答案: B

题目 2 2017 年第 12 题(栈)

表达式 $a * (b + c) * d$ 的后缀形式是()。

- A. $abcd * + *$ B. $abc + * d *$ C. $a * bc + * d$ D. $b + c * a * d$

解析: 四则运算表达式一共有前缀表达式、中缀表达式和后缀表达式三种形式, 用于表达式求值。

中缀表达式就是常见的运算表达式, 如 $(3 + 4) * 5 - 6$ 。本题中的表达式也属于中缀表达式。

前缀表达式又称波兰式, 前缀表达式的运算符位于操作数之前, 如: $- * + 3 4 5 6$ 。

后缀表达式又称逆波兰式, 与前缀表达式相似, 只是运算符位于操作数之后, 如: $3 4 + 5 * 6 -$ 。

中缀表达式对于计算机自动计算表达式的结果不太方便, 转换成前缀表达式和后缀表达式后非常方便计算, 所以表达式转换是常见操作。转换方法主要有两种: 一是手工转换, 二是计算机转换。

(1) 手工转换

以后缀表达式为例, 本题的转换步骤如下。

步骤 1: 按照运算符的优先级对所有运算单位加括号。

$$((a * (b + c)) * d)$$

步骤 2: 把运算符移动到对应的括号后面。

$$((a(b+c)+)*d)*$$

步骤 3: 把括号去掉, 即可得到后缀表达式。

$a\ b\ c + * \ d\ *$

前缀表达式的转换方法与后缀转换的方法类似,不同的是步骤 2 需要将运算符移动到对应括号的前面。

(2) 计算机转换

仍以后缀表达式为例,本题的转换步骤如下。

步骤 1: 初始化两个栈,运算符栈 s_1 和存储中间结果的栈 s_2 。

步骤 2: 从左至右扫描中缀表达式。

- ① 当遇到操作数时,将其压入 s_2 。
- ② 当遇到运算符时,比较其与 s_1 栈顶运算符的优先级。
 - 如果 s_1 为空或栈顶运算符为左括号,则直接将此运算符入栈。
 - 否则,若优先级比栈顶运算符的高,也将运算符压入 s_1 (注意转换为前缀表达式时是优先级较高或相同,而这里则不包括相同的情况)。
 - 否则,将 s_1 栈顶的运算符弹出并压入 s_2 ,再次与 s_1 中新的栈顶运算符相比较。
- ③ 当遇到括号时。
 - 如果是左括号,则直接压入 s_1 。
 - 如果是右括号,则依次弹出 s_1 栈顶的运算符并压入 s_2 ,直到遇到左括号为止,此时将这一对括号丢弃。

步骤 3: 重复步骤 2,直到表达式的最右端。

步骤 4: 将 s_1 中剩余的运算符依次弹出并压入 s_2 。

步骤 5: 依次弹出 s_2 中的元素并输出,结果的逆序即为中缀表达式对应的后缀表达式。

例如: $a * (b + c) * d$ 的具体转换过程如表 3-3 所示。

表 3-3 题目 1 后缀表达式示意表

扫描到的元素	s_2 (栈底->栈顶)	s_1 (栈底->栈顶)	说 明
a	a	空	数字,直接入栈
*	a	*	s_1 为空,运算符直接入栈
(	a	* (	左括号,直接入栈
b	a b	* (	数字
+	a b	* (+	s_1 栈顶为左括号,运算符直接入栈
c	a b c	* (+	数字
)	a b c +	*	右括号,弹出运算符直至遇到左括号,并舍弃一对括号
*	a b c + *	*	s_1 栈顶为 * 括号,将 s_1 栈顶运算符弹出并压入 s_2 ,再次比较,栈顶元素为空,压入 s_1
d	a b c + * d	*	数字
到达最右端	a b c + * d *	空	s_1 中剩余的运算符

前缀表达式的计算方法与后缀表达式类似,具体步骤如下。

步骤 1: 初始化运算符栈 s_1 和存储中间结果的栈 s_2 。

步骤 2: 从右至左扫描中缀表达式。

① 当遇到操作数时,将其压入 s_2 。

② 当遇到运算符时,比较其与 s_1 栈顶运算符的优先级。

- 如果 s_1 为空或栈顶运算符为右括号,则直接将此运算符入栈。
- 否则,若优先级比栈顶运算符的高或相等,也将运算符压入 s_1 。
- 否则,将 s_1 栈顶的运算符弹出并压入 s_2 ,再次转到步骤 4 与 s_1 中新的栈顶运算符相比较。

③ 当遇到括号时

- 如果是右括号,则直接压入 s_1 。
- 如果是左括号,则依次弹出 s_1 栈顶的运算符并压入 s_2 ,直到遇到右括号为止,此时将这一对括号丢弃。

步骤 3: 重复步骤 2,直到表达式的最左端。

步骤 4: 将 s_1 中剩余的运算符依次弹出并压入 s_2 。

步骤 5: 依次弹出 s_2 中的元素并输出,结果即为中缀表达式对应的前缀表达式。

例如: $a * (b + c) * d$ 的具体转换过程如表 3-4 所示。

表 3-4 题目 1 前缀表达式示意图

扫描到的元素	s_2 (栈底- $\rightarrow$ -栈顶)	s_1 (栈底- $\rightarrow$ -栈顶)	说 明
d	d	空	数字,直接入栈
*	d	*	s_1 为空,运算符直接入栈
)	d	*)	右括号直接入栈
c	d c	*)	数字直接入栈
+	d c	*) +	s_1 栈顶是右括号,直接入栈
b	d c b	*) +	数字直接入栈
(	d c b +	*	左括号,弹出运算符直至遇到右括号
*	d c b +	* *	与栈顶运算符优先级相等,压入 s_1
a	d c b + a	* *	优先级与-相同,入栈
到达最左端	d c b + a * *	空	s_1 剩余运算符

参考答案: B

题目 3 2017 年第 13 题(栈)

向一个栈顶指针为 hs 的链式栈中插入一个指针 s 指向的节点时,应执行()。

- A. $hs \rightarrow next = s$;
- B. $s \rightarrow next = hs; hs = s$;
- C. $s \rightarrow next = hs \rightarrow next; hs \rightarrow next = s$;

D. $s \rightarrow next = hs; hs = hs \rightarrow next;$

解析：向栈顶指针为 hs 的链式栈中插入指针 s 指向的节点的示意图如图 3-7 所示。

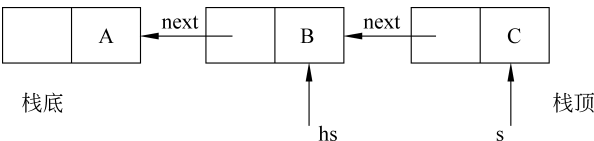


图 3-7 题 3 题解示意图

参考答案：B

题目 4 2017 年第 16 题(栈)

对于入栈顺序为 a, b, c, d, e, f, g 的序列, 下列()不可能是合法的出栈序列。

- A. a, b, c, d, e, f, g
- B. a, d, c, b, e, g, f
- C. a, d, b, c, g, f, e
- D. g, f, e, d, c, b, a

解析：栈的特点是先进后出, 根据栈的特点依次模拟各选项。

对于选项 A:

栈操作	a 入	a 出	b 入	b 出	c 入	c 出	d 入	d 出	e 入	e 出	f 入	f 出	g 入	g 出
栈数据	a	空	b	空	c	空	d	空	e	空	f	空	g	空

对于选项 B:

栈操作	a 入	a 出	b 入	c 入	d 入	d 出	c 出	b 出	e 入	e 出	f 入	g 入	g 出	f 出
栈数据	a	空	b	bc	bcd	bc	b	空	e	空	f	fg	f	空

对于选项 C:

栈操作	a 入	a 出	b 入	c 入	d 入	d 出	此时栈顶元素为 c, 必须 c 出栈后, b 才能出栈, 所以选项 C 无法实现 b 出栈。
栈数据	a	空	b	bc	bcd	bc	

对于选项 D:

栈操作	a 入	b 入	c 入	d 入	e 入	f 入	g 入	g 出	f 出	...	a 出
栈数据	a	ab	abc	abcd	abcde	abcdef	abcdefg	abcdef	abcde	...	空

参考答案：C

题目 5 2015 年第 15 题(栈)

如图 3-8 所示, 有一空栈 S , 对下列待进栈的数据元素序列 a, b, c, d, e, f 依次进行进栈、进栈、出栈、进栈、进栈、出栈的操作, 则此操作完成后, 栈 S 的栈顶元素为()。

- A. f
- B. c
- C. a
- D. b

解析：对数据进行模拟, 即可得出答案。

参考答案：B

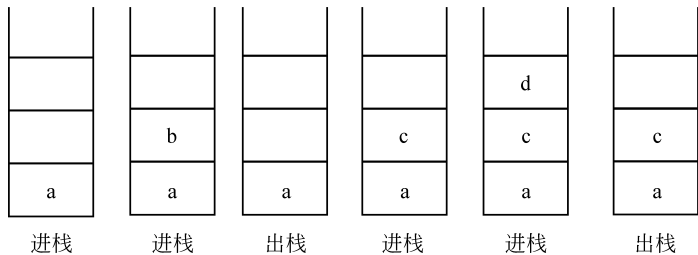
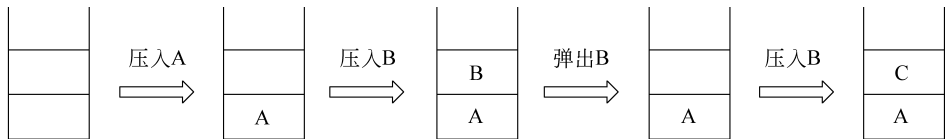


图 3-8 题目 5 模拟图

题目 6 2013 年第 7 题(栈)

下图所使用的数据结构是()。



- A. 哈希表 B. 栈 C. 队列 D. 二叉树

解析：根据数据结构先进后出的特点，可以确定该数据结构为栈。

参考答案：B

题目 7 2012 年第 2 题(队列)

()是一种先进先出的线性表。

- A. 栈 B. 队列
C. 哈希表(散列表) D. 二叉树

解析：先进先出是队列的典型特点。

参考答案：B

题目 8 2012 年第 12 题(栈)

如果一个栈初始时空，且当前栈中的元素从栈底到栈顶依次为 a,b,c(如图 3-9 所示)，另有元素 d 已经出栈，则可能的入栈顺序是()。

- A. a, d, c, b B. b, a, c, d
C. a, c, b, d D. d, a, b, c

解析：该题的解题思路与题目 3 和题目 4 相似，可以利用模拟的方法。

参考答案：D

题目 9 2011 年第 11 题(队列)

广度优先搜索时，需要用到的数据结构是()。

- A. 链表 B. 队列
C. 栈 D. 散列表

解析：广度优先搜索算法(又称宽度优先搜索，BFS)是很多重要的图的算法的原型。Dijkstra 单源最短路径算法和 Prim 最小生成树算法都采用了与广度优先搜索类似的思

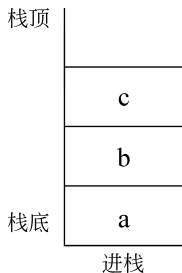


图 3-9 题目 8 示意图

想。从算法的观点,所有因为展开节点而得到的子节点都会被加入一个先进先出的队列中。

参考答案: B

题目 10 2010 年第 9 题(栈)

前缀表达式 $+3 * 2+5 12$ 的值是()。

A. 23

B. 25

C. 37

D. 65

解析: 前缀表达式的计算机求值过程如下。

从右至左扫描表达式,当遇到数字时,将数字压入堆栈,遇到运算符时,弹出栈顶的两个数,用运算符对它们做相应的计算(栈顶元素 op 次顶元素),并将结果入栈;重复上述过程直到表达式最左端,最后运算得出的值即为表达式的结果。

例如: $+3 * 2+5 12$ 。

步骤 1: 从右至左扫描,将 12、5 压入堆栈。

步骤 2: 遇到 + 运算符,弹出 5 和 12,计算出 $12+5$ 的值,得 17,再将 17 入栈。

步骤 3: 将 2 压入堆栈。

步骤 4: 遇到 * 运算符,弹出 2 和 17,计算出 17×2 的值,得 34,再将 34 入栈。

步骤 5: 将 3 入栈。

步骤 6: 最后是 + 运算符,弹出 3 和 34,计算出 $34+3$ 的值,即 37,由此得出最终结果。

参考答案: C

题目 11 2010 年第 15 题(栈)

元素 R1、R2、R3、R4、R5 入栈的顺序为 R1、R2、R3、R4、R5。如果第 1 个出栈的是 R3,那么第 5 个出栈的不可能是()。

A. R1

B. R2

C. R4

D. R5

解析: 根据已知条件可知,当 R3 出栈时,栈中从栈底到栈顶的元素为 R1、R2。第 5 个出栈,即最后一个出栈元素肯定不是 R2,因为根据栈的特点,必须 R2 出栈后 R1 才能出栈,所以 R2 不可能是最后一个出栈。

参考答案: B

题目 12 2009 年第 12 题(栈)

有 6 个元素 FEDCBA 从左至右依次顺序进栈,在进栈过程中会有元素被弹出栈。下列不可能是合法的出栈序列的是()。

A. EDCFAB

B. DECABF

C. CDFEBA

D. BCDAEF

解析: 该题方法与题目 3 方法一样,采用模拟法即可。

参考答案: C

题目 13 2009 年第 13 题(栈)

表达式 $a * (b+c) - d$ 的后缀表达式是()。

A. $abcd * + -$

B. $abc + * d -$

C. $abc * + d -$

D. $- + * abcd$

解析: 参考题目 1 解析。

参考答案: B

3.2.3 知识点巩固

从历年考点角度分析,本节的考点主要为栈和队列这两个知识点。具体的出题数目比较如表 3-5 所示。

表 3-5 历年知识点出现次数统计表

知识点	栈	队 列
出现个数	11	2
本节占比	84.62%	15.38%

从历年知识点的出现次数可以看出,栈知识点所占分值较高。栈的应用比较广泛,所以可考查的知识点也比较多。队列知识点相对较少。

根据以上知识点,本书提供几道习题供大家复习巩固。

- 栈和队列都是特殊的线性表,其共同点是()。(栈和队列)
 - 只允许在端点处插入和删除元素
 - 都是先进后出
 - 都是先进先出
 - 都可以用链表存储
- 栈的插入和删除操作在()进行。(栈)
 - 栈顶
 - 栈底
 - 任意位置
 - 指定位置
- 假如一个栈的压入序列为 123,则不可能是栈的输出序列的是()。(栈)
 - 2 3 1
 - 3 2 1
 - 3 1 2
 - 1 2 3
- 对图进行深度优先搜索时,需要用到的数据结构是()。(栈)
 - 链表
 - 队列
 - 栈
 - 散列表
- 链栈执行 pop 操作,并将出栈的元素存在 x 中应该执行()。(栈)
 - $x=top; top=top->next;$
 - $x=top->data;$
 - $top=top->next; x=top->data;$
 - $x=top->data; top=top->next;$
- 设栈 S 和队列 Q 的初始状态为空,元素 e1,e2,e3,e4,e5,e6 依次通过栈 S,一个元素出栈后立即进入队列 Q,若 6 个元素出栈的序列是 e2,e4,e3,e6,e5,e1,则栈 S 的容量最多应该是()。(栈)
 - 6
 - 4
 - 3
 - 2
- 若已知一个栈的入栈顺序是 1,2,3,4,其出栈序列为 P1,P2,P3,P4,则 P2,P4 不可能是()。(栈)
 - 2,4
 - 2,1
 - 4,3
 - 3,4
- 循环队列存储在数组 $A[0 \cdots n]$,则入队时的操作为()。(队列)
 - $rear=rear+1;$
 - $rear=(rear+1) \bmod (n-1);$
 - $rear=(rear+1) \bmod n;$
 - $rear=(rear+1) \bmod (n+1);$
- 若用数组 $A[0..5]$ 实现循环队列,且当前 rear 和 front 的值分别为 1 和 5,当从队列中删除一个元素并再加入两个元素后,rear 和 front 的值分别为()。(队列)
 - 3 和 4
 - 3 和 0
 - 5 和 0
 - 5 和 1

10. 前缀表达式 $* + 2\ 3\ 4$ 的计算结果是()。(栈)
 A. 24 B. 20 C. 18 D. 14
11. 算术表达式 $a + b * (c + d / e)$ 转为后缀表达式后为()。(栈)
 A. $ab + cde / *$ B. $abcde / + * +$ C. $abcde / * + +$ D. $abcde * / + +$

3.3 树

3.3.1 基本知识介绍

树是一类重要的非线性数据结构,树中节点之间具有明确的层次关系,并且节点之间有分支,它非常类似于真正的树。树状结构在客观世界中大量存在,如行政组织机构和人类社会的家谱等都可用树状结构形象地表示。

树是 $n(n \geq 0)$ 个节点的有限集 T 。 T 要么是空集(空树),要么是非空集。对于一棵非空树:有且仅有一个特定的称为根(Root)的节点;其余的节点可分为 $m(m > 0)$ 个互不相交的有限集 $T_1, T_2, \dots, T_m$,其中每个集合本身又是一棵树,并称为根的子树。例如,图 3-10(a)表示的是一个有 9 个节点的树,其中 A 是根节点,其余的节点分成 3 棵互不相交的子集: $T_1 = \{B, E, F, G\}$, $T_2 = \{C\}$, $T_3 = \{D, H, I\}$; T_1, T_2 和 T_3 都是根 A 的子树,且本身也是一棵子树。

一个节点拥有的子树数称为该节点的度(Degree)。一棵树中节点的最大度数称为该树的度。在图 3-10(b)所示的某大学的行政组织结构树中,文学院节点的度为 3,是最大的,应该作为树的度。

树中节点的最大层数称为树的深度(Depth)或高度。节点层数(Level)从根开始算起,根为第 1 层,其余节点的层次等于其双亲节点的层数加 1。

度数为零的节点称为叶子(Leaf)节点。度数不为零的节点称为非终端节点。

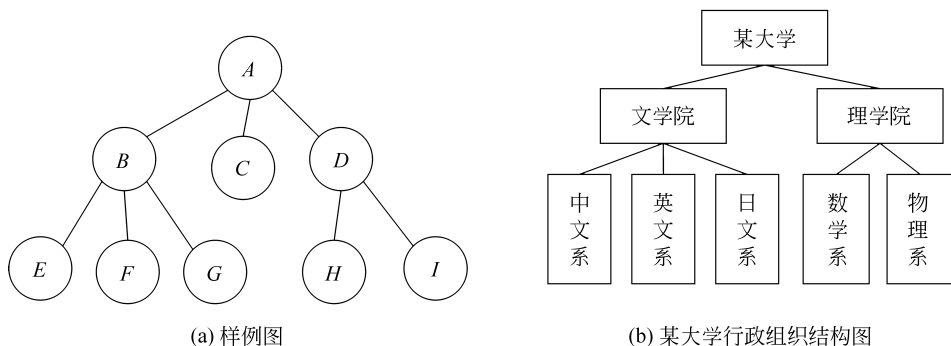


图 3-10 树的示意图

森林(Forest)是 $m(m \geq 0)$ 棵互不相交的树的集合。若将一棵树的根节点被删除,就得到该树的子树所构成的森林,将森林中所有树作为子树用一个根节点连起来,森林就算成了一棵树。

二叉树(Binary Tree)是一种特殊的树,其每个节点的度都不大于2,并且每个节点的孩子节点的次序不能任意颠倒。由此可知,一个二叉树中的每个节点只能含有0,1或2个孩子,而且每个孩子有左右之分。通常把位于左边的孩子称为左孩子,把位于右边的孩子称为右孩子。二叉树的基本形态有以下五种,如图3-11所示。

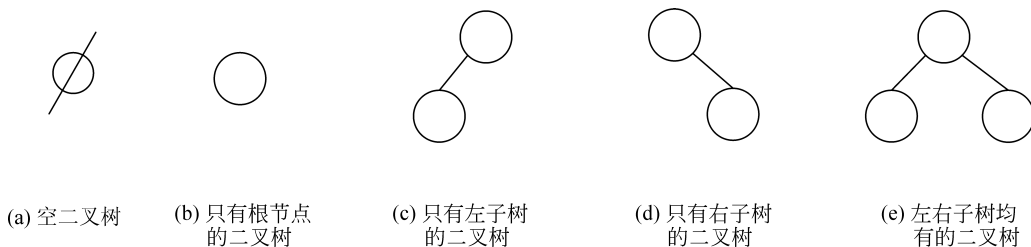


图 3-11 二叉树的基本形态

3.3.2 历年真题解析

题目 1 2018 年第 7 题(树)

根节点深度为0,一棵深度为 h 的满 K 叉树除最后一层无任何子节点外,每一层上所有节点都有 k 个子节点的树,则该树共有()个节点。

A. $(k^{h+1}-1)/(k-1)$

B. k^{h-1}

C. k^h

D. $(k^{h-1})/(k-1)$

解析:该题可以利用带入法,将 K 设为2,即二叉树,带入各个答案进行筛选。

也可以直接画出 K 叉树,如图3-12所示。

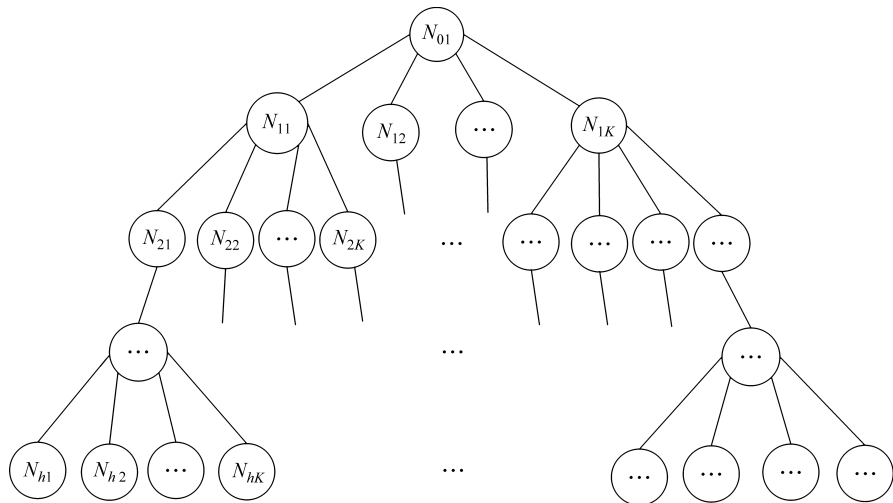


图 3-12 完全 K 叉树样例

从图中可以得出: K 叉树第0层的节点数为1,第1层的节点数为 k ,第2层的节点数为 k^2 ,第 h 层的节点数为 k^h 。

所以,最后总节点数目为 $1+k^1+k^2+k^3+\cdots+k^h=(k^{h+1}-1)/(k-1)$ 。

注:可以利用等比数列公式计算。

参考答案: A

题目2 2017年第10题(树)

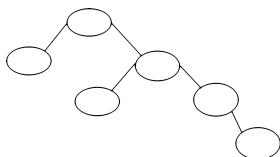
设 G 是有 n 个节点、 m 条边($n \leq m$)的连通图,必须删去 G 的()条边,才能使得 G 变成一棵树。

- A. $m-n+1$ B. $m-n$ C. $m+n+1$ D. $n-m+1$

解析:一个具有 n 个节点的树共有 $n-1$ 条边。所以 G 必须删去 $m-(n-1)=m-n+1$ 条边。

参考答案: A

题目3 2016年第11题(二叉树)



一棵二叉树如左图所示,若采用顺序存储结构,即用一维数组元素存储该二叉树中的节点(根节点的下标为1,若某节点的下标为 i ,则其左孩子位于下标 $2i$ 处、右孩子位于下标 $2i+1$ 处),则图中所有节点的最大下标为()。

- A. 6 B. 10 C. 12 D. 15

解析:该题目主要考查二叉树的顺序存储结构,二叉树的顺序存储方式的数组下标可以利用二进制的方式表示,如图3-13所示。

每个节点的下标均从根节点开始到所在节点的二进制序列,该树的最大下标为 $(1111)_2$,可以转换成十进制数15。

参考答案: D

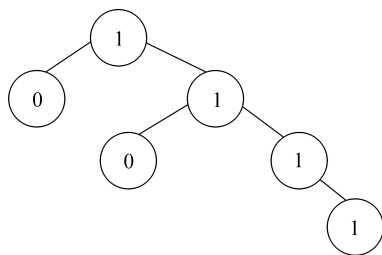


图3-13 题目3解析示意图

题目4 2016年第2题(二叉树)

约定二叉树的根节点高度为1。一棵节点数为2016的二叉树最少有_____个叶子节点;一棵节点数为2016的二叉树最小的高度值是_____。

解析:二叉树有一个性质,即叶子节点=度为2的节点数+1。

当二叉树叶子节点最少时,即度为2的节点数也最少,最少为0,即二叉树节点没有度为2的节点,所有非叶子节点都只有一个子树,此时会有1个叶子节点。

一棵二叉树的高度值要求最小,这棵二叉树肯定是完全二叉树。对于完全二叉树,深度为 h 的完全二叉树至少有 2^{h-1} 个节点,至多有 2^h-1 个节点。树高 h 为

$$h = \log_2(n+1), \quad n \text{ 为所有节点数}$$

参考答案: 1,12

题目5 2015年第16题(二叉树)

前序遍历序列与中序遍历序列相同的二叉树为()。

- A. 根节点无左子树的二叉树
B. 根节点无右子树的二叉树
C. 只有根节点的二叉树或非叶子节点只有左子树的二叉树

D. 只有根节点的二叉树或非叶子节点只有右子树的二叉树

解析: 二叉树的遍历根据遍历根节点、左子树和右子树的顺序的不同分为前序遍历、中序遍历和后序遍历三种, 这三种遍历方法是根据遍历根节点的先后次序区分的。先遍历根节点的称为前序遍历, 中间遍历根节点的称为中序遍历, 最后遍历根节点的称为后序遍历。左子树和右子树的遍历都是先遍历左子树, 后遍历右子树。

要实现二叉树的前序遍历序列与中序遍历序列相同, 即(根、左、右)=(左、根、右), 只有当左子树为空时, 该等式才成立。

参考答案: A

题目 6 2015 年第 17 题(二叉树)

如果根的高度为 1, 则具有 61 个节点的完全二叉树的高度为()。

A. 5 B. 6 C. 7 D. 8

解析: 参考题目 2 解析。

参考答案: B

题目 7 2015 年第 4 题(二叉树)

一棵节点数为 2015 的二叉树最多有_____个叶子节点。

解析: 根据题目 2 中二叉树的性质: 叶子节点=度为 2 的节点数+1。

叶子节点数最多, 即度为 2 的节点数最多, 由于:

二叉树总节点数=叶子节点 N_0 + 度为 1 的节点 N_1 + 度为 2 的节点 N_2 , 使 $N_1=0$, 叶子节点最多, 即 $2015=N_0+N_2=N_0+N_0-1=2\times N_0-1$ 。

参考答案: 1008

题目 8 2014 年第 16 题(二叉树)

一棵具有 5 层的满二叉树的节点数为()。

A. 31 B. 32 C. 33 D. 16

解析: 一棵二叉树, 如果每一个层的节点数都达到最大值, 则这个二叉树就是满二叉树。也就是说, 如果一个二叉树的层数为 K , 且节点总数是 2^k-1 , 则它就是满二叉树。

参考答案: A

题目 9 2013 年第 9 题(二叉树)

已知一棵二叉树有 10 个节点, 则其中至多有()个节点有 2 个子节点。

A. 4 B. 5 C. 6 D. 7

解析: 根据公式: $N=N_0+N_1+N_2$, 其中 N 为总节点数, N_i 为度为 i 的节点数。再根据 $N_0=N_2+1$, 可得出: $N=N_1+2N_2+1$ 。

由于 $N=10$, 是偶数, 所以 N_1 最小取为 1, 可得出 $N_2=4$ 。

参考答案: A

题目 10 2013 年第 11 题(二叉树)

二叉树的()第一个访问的节点是根节点。

A. 先序遍历 B. 中序遍历 C. 后序遍历 D. 以上都是

解析: 参考题目 5 解析。

参考答案: A

题目 11 2012 年第 6 题(二叉树)

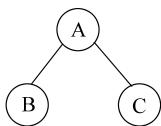
如果一棵二叉树的中序遍历是 BAC,那么它的先序遍历不可能是()。

- A. ABC B. CBA C. ACB D. BAC

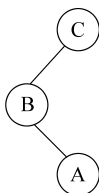
解析: 该题采用构造法,利用中序遍历和先序遍历可以把二叉树构造出来。根据中序遍历(左、根、右)和先序遍历(根、左、右)利用递归的方法可以构造出二叉树。

例如,选项 A 先利用先序遍历序列 ABC 可知 A 是根节点。再结合中序遍历序列 BAC 可知 B 为左子树,C 为右子树。

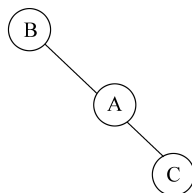
对于选项 B,先利用先序遍历序列 CBA 可知 C 是根节点。再结合中序遍历序列 BAC 可知 BA 为左子树,右子树为空。再利用先序遍历 CBA 可知左子树中 B 又为子树根节点,再利用中序遍历 BA 可知 A 为右子树。



(a) 选项 A 构造的二叉树



(b) 选项 B 构造的二叉树



(c) 选项 D 构造的二叉树

图 3-14 题目 11 解析示意图

选项 C 前后矛盾,无法构造出二叉树。

参考答案: C

题目 12 2011 年第 7 题(二叉树)

如果根节点的深度记为 1,则一棵恰有 2011 个叶节点的二叉树的深度最少是()。

- A. 10 B. 11 C. 12 D. 13

解析: 根据满二叉树中深度 h 和节点数的关系得:

$$h = \log_2(n+1) = \log_2 2012 = 11$$

参考答案: B

题目 13 2010 年第 5 题(二叉树)

如果树根算第 1 层,那么一棵 n 层的二叉树最多有()个节点。

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

解析: 参考题目 4 解析。

参考答案: A

题目 14 2010 年第 17 题(二叉树)

一棵二叉树的前序遍历序列是 ABCDEFG,后序遍历序列是 CBFEGDA,则根节点的左子树的节点个数可能是()。

- A. 2 B. 3 C. 4 D. 5

解析: 根据二叉树前序遍历序列(根、左、右)可知 A 是根节点。又由后序遍历序列(左、右、根)得知 D 必然是右子树的根节点。

再看前序遍历序列,D 前面的 ABC 中的 A 是根节点,剩下的 BC 节点必然是左子树。

参考答案: A

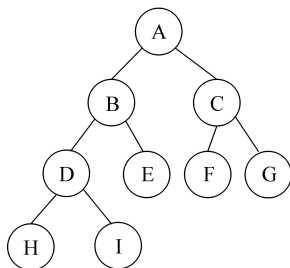
题目 15 2010 年第 19 题(二叉树)

完全二叉树的顺序存储方案是指将完全二叉树的节点从上至下、从左至右依次存放在一个顺序结构的数组中。假定根节点存放在数组的 1 号位置,则第 k 号节点的父节点如果存在,则应当存放在数组的()号位置。

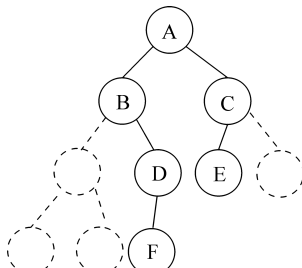
- A. $2k$ B. $2k+1$ C. $k/2$ 下取整 D. $(k+1)/2$ 下取整

解析: 完全二叉树的存储可以按照从上到下、从左到右的顺序依次存储在一维数组中。完全二叉树的顺序存储如图 3-15 所示。

如果按照从上到下、从左到右的顺序把非完全二叉树也同样的编号,将节点依次存放在一维数组中,为了能够正确反映二叉树中节点之间的逻辑关系,需要在一维数组中将二叉树中不存在的节点位置空出来。



(a) 完全二叉树样例



(b) 非完全二叉树样例

节点序号	1	2	3	4	5	6	7	8	9	10
数组	A	B	C	D	E	F	G	H	I	^

(c) 完全二叉树样例顺序存储

节点序号	1	2	3	4	5	6	7	8	9	10
数组	A	B	C	^	D	E	^	^	^	F

(d) 非完全二叉树样例顺序存储

图 3-15 题目 15 解析示意图

参考答案: C

题目 16 2009 年第 14 题(二叉树)

一个包含 n 个分支节点(非叶节点)的非空二叉树,它的叶节点数目最多为()。

- A. $2n+1$ B. $2n-1$ C. $n-1$ D. $n+1$

解析: 根据 $n = n_1 + n_2$, 这里 n_1 和 n_2 分别指度为 1 和 2 的节点。

又根据 $n_0 = n_2 + 1$, n_0 为度为 0 的叶子节点。带入上式可得

$$n = n_0 + n_1 - 1$$

叶节点数目最多,即 $n_1 = 0$,可得出 $n_0 = n + 1$ 。

参考答案: D

3.3.3 知识点巩固

从历年考点角度分析,本节的考点主要分为树和二叉树两个知识点。具体的出题数目比较如表 3-6 所示。

表 3-6 历年知识点出现次数统计表

知识点	树	二 叉 树
出现个数	2	14
本节占比	12.50%	87.50%

从历年知识点的出现次数可以看出,二叉树知识点所占分值极高,二叉树是树形结构的主要应用形式,由于计算简单,所以应用非常广泛。树的概念相对比较宽泛,因此相对较难考查。

根据以上知识点,本书提供几道习题供大家复习巩固。

1. 已知某二叉树深度为 4,则该二叉树最多节点和最少节点数分别为()个。(二叉树)

- A. 15,4 B. 15,6 C. 16,4 D. 16,6

2. 在具有 200 个节点的完全二叉树中,利用顺序存储,设根节点的编号为 1,则编号为 60 的节点其左孩子节点的编号为()。(二叉树)

- A. 61 B. 62 C. 120 D. 121

3. 一棵具有 124 个叶子节点的完全二叉树最多有()个节点。(二叉树)

- A. 247 B. 248 C. 249 D. 250

4. 已知一棵含 50 个节点的二叉树中只有一个叶子节点,则该树中度为 1 的节点的个数为()。(二叉树)

- A. 0 B. 1 C. 48 D. 49

5. 一棵完全二叉树有 64 个叶节点,则该树可能达到的最大深度为()。(二叉树)

- A. 8 B. 9 C. 10 D. 11

6. 一棵二叉树有 11 个叶节点,则该二叉树中度为 2 的节点个数是()。(二叉树)

- A. 10 B. 11 C. 12 D. 不确定的

7. 具有 $n(n>0)$ 个节点的完全二叉树的深度为()。(二叉树)

- A. $\log_2(n)$ B. $\log_2(n)$ C. $\log_2(n)+1$ D. $\log_2(n)+1$

注: $\lceil x \rceil$ 表示向上取整,即不小于 x 的最小整数; $\lfloor x \rfloor$ 表示向下取整,即不大于 x 的最大整数。

8. 设树 T 的度为 4,其中度为 1、2、3、4 的节点个数分别为 4、2、1、1,则 T 中的叶子数为()。(树)

- A. 5 B. 6 C. 7 D. 8

9. 将二叉树的概念推广到三叉树,一棵有 244 个节点的完全三叉树的高度为()。(树)

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