電子科技大學

數據結構與算法

習
題
冊

僅限 2016 年秋季立人樓 B202 選課 110 人使用

周益民

2016 年 9 月 1 日

<table>
<tr><td>课程名称</td><td>数据结构与算法</td><td rowspan="2">授课专业</td><td rowspan="2">计算机科学与技术
信息安全</td><td>班级</td><td>2015 级
本科</td></tr>
<tr><td>课程编号</td><td>E0600645.23</td><td>修课人数</td><td>110</td></tr>
<tr><td rowspan="3">课程类型</td><td>必修</td><td colspan="4">公共基础课(　　)；学科基础课(☑)；专业核心课(　　)</td></tr>
<tr><td>选修</td><td colspan="4">专业选修(　　)；任选课（　　)；公选课(　　)；</td></tr>
<tr><td></td><td colspan="4">理论课(　　)；实践课(　　)</td></tr>
<tr><td>授课方式</td><td colspan="3">课堂讲授为主（ ☑)；实验为主（　　)；
自学为主（　　)；专题讨论为主（　　)；</td><td>是否采用
多媒体授
课</td><td>是</td></tr>
<tr><td>考核方式及成
绩构成</td><td colspan="3">考　试(☑)　考　查(　　)
成绩构成及比例：
实验 20%　半期 0%　平时 10%　期末 70%</td><td>是否采用
双语教学</td><td>否</td></tr>
<tr><td>学时分配</td><td colspan="5">共 72 学时：理论 64 学时，实验 8 学时；另有实践 8 学时；</td></tr>
<tr><td>教材</td><td>名称</td><td colspan="2">作者</td><td colspan="2">出版社及出版时间</td></tr>
<tr><td></td><td>数据结构与算法</td><td colspan="2">吴跃，李树全，
尚明生，陈端兵</td><td colspan="2">机械工业出版社，2010.2</td></tr>
<tr><td>参考书目</td><td>算法设计与分析
（第二版）

数据结构(C 语言版)

大话数据结构</td><td colspan="2">王晓东

严蔚敏，吴伟民

程杰</td><td colspan="2">清华大学出版社

清华大学出版社，2007

清华大学出版社, 2011</td></tr>
<tr><td>授课时间</td><td colspan="5">2016-2017 学年　第 1 学期
第 1 周 - 第 14 周</td></tr>
</table>

线性表

1. 选择题

(1) 线性表是(　　)。
A. 一个有限序列，可以为空　　B. 一个有限序列，不能为空
C. 一个无限序列，可以为空　　D. 一个无限序列，不能为空

(2) 关于线性表，下列说法中正确的是(　　)。
A. 线性表中的每个元素都有一个直接前驱和一个直接后继。
B. 线性表中的数据元素可以具有不同的数据类型。
C. 线性表中的数据元素类型是确定的。
D. 线性表中任意一对相邻的数据元素之间存在序偶关系。

(3) (　　)是一个线性表
A. 由n个实数组成的集合　　B. 由所有实数组成的集合
C. 由所有整数组成的集合　　D. 由n个字符组成的序列

(4) 线性表的顺序存储是一种(　　)结构。
A. 随机存取　　B. 顺序存取　　C. 索引存储　　D. 散列存储

(5) n个节点的线性表采用数组实现，算法的时间复杂度为$O(1)$的操作是(　　)。
A. 访问第i个节点$(1 \le i \le n)$和求第i个节点的直接前驱$(2 \le i \le n)$
B. 在第i个节点之后插入一个新节点$(1 \le i \le n)$
C. 删除第i个节点$(1 \le i \le n)$
D. 以上都不对

(6) 对于顺序存储的线性表，访问某个元素和增加一个元素的时间复杂度(　　)。
A. $O(n), O(n)$　　B. $O(n), O(1)$　　C. $O(1), O(n)$　　D. $O(1), O(1)$

(7) 顺序表的插入算法中，当n个空间已满时，可以再申请增加分配m个空间，若申请失败，则说明系统没有(　　)可分配的存储空间。
A. m个　　B. m个连续的　　C. $n+m$个　　D. $n+m$个连续的

(8) 将长度为n的单链表接在长度为m的单链表之后，其时间复杂度为(　　)。
A. $O(1)$　　B. $O(n)$　　C. $O(m)$　　D. $O(n+m)$

(9) 在单链表中附加头节点的目的是(　　)。
A. 保证单链表中至少有一个节点　　B. 标识单链表中首节点的位置
C. 方便运算实现　　D. 说明单链表是线性表的链式存储

(10) 线性表的链接存储结构是一种(　　)的存储结构。
A．随机存取　　B．顺序存取　　C．索引存取　　D．散列存取

(11) 链表不具有的特点是(　　)。
A．可以随机访问任一元素　　B．插入、删除操作不需要移动元素
C．不必事先估计存储空间　　D．所需空间与线性表长度成正比

(12) 循环链表的主要优点是(　　)。
A．不再需要头指针了
B．从表中任意节点出发都能扫描到整个链表
C．已知某个节点位置后，能够很容易找到它的直接前驱
D．在进行插入、删除操作时能够更好地保证链表不断开

(13) 将线性表$(a_1,a_1,\cdots,a_n)$组织成为一个带头节点的循环链表，设H为链表的头指针，则链表中最后一个节点的指针域中存放的是(　　)。
A．变量H的地址　　B．变量H的值
C．元素a_1的地址　　D．元素a_1的值

(14) 若要在$O(1)$的时间复杂度内实现两个循环单链表的首尾相接，则两个循环单链表应该各设置一个指针，分别指向(　　)。
A．各自的头节点　　B．各自的尾节点
C．各自的第一个元素节点　　D．一个表的头节点，一个表的尾节点

(15) 设有两个长度为n的单链表，以h为头指针的链表是非循环的，以r为尾指针的链表是循环的，则(　　)。
A．在两个链表上删除第一个节点的操作，其时间复杂度均为$O(1)$
B．在两个链表的表尾插入一个节点的操作，其时间复杂度均为$O(n)$
C．循环链表要比非循环链表占用更多的存储空间
D．循环链表要比非循环链表占用更少的存储空间

(16) 若某线性表中最常用的操作是取第i个元素和找第i个元素的前驱，则采用(　　)存储方法最节省时间。
A．顺序表　　B．单链表　　C．双链表　　D．循环单链表

(17) 若链表中最常用的操作是在最后一个节点之后插入一个节点和删除第一个节点，则采用(　　)存储方法最节省时间。
A．单链表　　B．带头指针的循环单链表
C．双链表　　D．带尾指针的循环单链表

(18) 若链表中最常用的操作是在最后一个节点之后插入一个节点和删除最后一个节点，则采用(　　)存储方法最节省运算时间。
A．单链表　B．循环双链表　　C．循环单链表　　D．带尾指针的循环单链表

(19) 如果对具有n，$(n>1)$，个元素的线性表的基本操作只有四种：删除第一个元素，删除最后一个元素，在第一个元素前插入新元素，在最后一个元素的后面插入新元素。那么最好使用(　　)。

A．只设尾指针的循环链表　　B．只设尾指针的非循环双链表

C．只设头指针的循环双链表　　D．同时设置头指针和尾指针的循环单链表

(20) 若计算斐波拉契数F_n的运算时间为T_n，且有$T_n=T_{n-1}+T_{n-2}$。那么计算F_n的时间复杂度为(　　)。

A．$O(n)$　　B．$O(n\log n)$　　C．$O(n^2)$　　D．$O(2^n)$

2. 简答题

(1) 在什么情况下线性表使用顺序存储比较好？

(2) 单链表设置头节点的作用是什么？

(3) 若频繁地对一个线性表进行插入和删除操作，选用什么存储结构比较好？为什么？

(4) 如果某线性表中数据元素的类型不一致，但希望能够根据下标随机存取每个元素，请为这个线性表设计一个合适的存储结构。

i

i

(5) 请比较线性表的两种基本存储结构：顺序表和单链表。

(6) 举例说明对于相同的逻辑结构，同一种运算在不同的存储方式下实现，算法的效率不同。

3. 算法设计题

(1) 试以顺序表作存储结构，实现线性表就地逆置？

(2) 已知顺序表 L 中的元素为递增有序排列，设计一个算法实现将元素 x 插入到表 L 中，并保持 L 仍然递增有序？

(3) 设计算法实现将单链表就地逆置？

(4) 有两个递增有序的单链表 la 和 lb，设计算法将这两个链表合并成一个有序链表。

(5) 用顺序表表示集合，设计算法实现集合的求交集运算。

(6) 用顺序表表示集合，设计算法实现集合的求并集运算。

(7) 在某商店的仓库中，对电视机按照其价格从低到高建立一个单链表，链表的每个节点指出同样价格的电视机的台数。现有m台价格为n元的电视机入库。请编写算法完成仓库的进货管理。

【分析】根据题意，定义存储结构如下

```
struct Node
{
   float price;
   int   num;
   struct Node *next;
}
```

算法首先查找链表中是否具有相同价格的电视机(注意利用单链表的有序性)，然后根据查找结果进行相应的处理，具体需要处理以下两种情况：

① 有同样价格的电视机，则直接将台数加到相应的节点 num 域中。

② 没有价格相同的电视机，则申请新节点 s，将节点 s 插入到相应的位置。

(8) 约瑟夫环问题。

【其一】据说著名犹太历史学家 Josephus 有过以下的故事：在罗马人占领乔塔帕特后，39 个犹太人与 Josephus 及他的朋友躲到一个洞中，39 个犹太人决定宁愿死也不要被敌人抓到，于是决定了一个自杀方式，41 个人排成一个圆圈，由第 1 个人开始报数，每报数到第 3 人该人就必须自杀，然后再由下一个重新报数，直到所有人都自杀身亡为止。然而 Josephus 和他的朋友并不想遵从，Josephus 要他的朋友先假装遵从，他将朋友与自己安排在第 16 个与第 31 个位置，于是逃过了这场死亡游戏。

【其二】17 世纪的法国数学家加斯帕在《数目的游戏问题》中讲了这样一个故事：15 个教徒和 15 个非教徒在深海上遇险，必须将一半的人投入海中，其余的人才能幸免于难，于是想了一个办法：30 个人围成一圆圈，从第一个人开始依次报数，每数到第九个人就将他扔入大海，如此循环进行直到仅余 15 个人为止。问怎样排法，才能使每次投入大海的都是非教徒。

【其三】N 个人，编号$0..N-1$，从 0 开始报数，报到$M-1$的退出，通常取$M<N$。剩下的人继续从 0 开始报数，报到$M-1$的退出，如此往复。求最终胜利者的编号。

【分析】由于约瑟夫环问题本身具有循环性质，考虑采用循环链表。由于游戏需要沿指

针域一圈圈周游链表，为了统一对节点的操作，选用不带头节点的循环链表为好。存储结构定义为

```
typedef struct Jonse
{
    int code;
    struct Jonse *next;
} Jonse;
Jonse* Create(int); // 创建约瑟夫环
void   ShowList(Jonse *); // 将约瑟夫环上所有节点值打印出来
Jonse* JonseOut(Jonse *, int, int); // 执行约瑟夫生死游戏
```

栈、队列、数组

1. 选择题

(1) 设有一个空栈，栈顶指针为 1000H(十六进制)，每个元素需要 2 个单位的存储空间，则执行 PUSH，PUSH，POP，PUSH，POP，PUSH，POP，PUSH 操作后，栈顶指针值为(　　)。
A．1002H　　B．1003H　　C．1004H　　D．1005H

(2) 一个栈的入栈序列是{1,2,3,4,5}，则栈不可能的输出序列是(　　)。
A．{5,4,3,2,1}　　B．{4,5,3,2,1}　　C．{4,3,5,1,2}　　D．{1,2,3,4,5}

(3) 向一个栈顶指针为 h 的带头结点链栈中插入指针 s 所指向节点时，应该执行(　　)。
A．h->next = s;　　B．s->next = h;
C．s->next = h; h->next = s;　　D．s->next = h->next; h->next = s;

(4) 从栈顶指针为 top 的链栈中删除一个节点，用 x 保存被删除节点的值，则应该执行(　　)。
A．x = top;　top = top->next;　　B．x = top->data;
C．top = top->next; x = top->data;　　D．x = top->data; top = top->next;

(5) 队列的先进先出特性是指(　　)。
A．最后插入队列中的元素总是最先被删除
B．当同时进行插入删除操作时，总是插入操作优先
C．每当有删除操作时，总要先做一次插入操作
D．每次从队列中删除的总是最早插入的元素

(6) 循环队列存储在数组 A[M]中，则入队时的操作为(　　)。
A．rear = rear + 1;　　B．rear = (rear + 1) % (M + 1)
C．rear = (rear + 1) % M;　　D．rear = (rear + 1) % (M - 1)

(7) 若用一个长度为 6 的数组来实现循环队列，且当前的 rear 和 front 值分别为 0 和 3。则从该队列中删除一个元素，再增加两个元素后，rear 和 front 值分别为(　　)。
A．1 和 5　　B．2 和 4　　C．4 和 2　　D．5 和 1

(8) 用不带头节点的单链表存储队列时，其队头指针指向队头节点，队尾指针指向队尾节点，则执行删除操作时，(　　)。
A．仅修改队头指针　　B．仅修改队尾指针
C．队头和队尾指针都需要修改　　D．队头指针和队尾指针可能都需要修改

(9) 在链队列中，设指针 f 和 r 分别指向队头和队尾，则插入 s 所指向的节点，操作应该是(　　)。
A．f->next = s; f = s;　B．r->next = s; r = s;　C．s->next = r; r = s;　D．s->next = f; f = s;

(10) 最不适合用作链队列的链表是(　　)。
A．只带队首指针的非循环双链表
B．只带队首指针的循环双链表
C．只带队尾指针的循环双链表
D．只带队尾指针的循环单链表
(11) 数组通常具有的两种基本操作是(　　)。
A．查找和修改　　B．查找和索引　　C．索引和修改　　D．建立和删除

(12) 将数组称为随机存取结构是因为(　　)。
A．数组元素是随机的　　B．对数组的任意一个元素的存取时间是相等的
C．随时可以访问数组元素　　D．数组的存储结构是不定的

(13) C 语言中定义的一维数组 a[50]和二维数组 b[10][5]具有相同的首元素地址，即&(a[0]) == &(b[0][0])为真。在以列为主序时，a[18]的地址和(　　)地址相同。
A．b[1][7]　　B．b[1][8]　　C．b[8][1]　　D．b[7][1]

(14) 对特殊矩阵采用压缩存储的目的主要是为了(　　)。
A．表达变得简单　　B．对矩阵元素的存取变得简单
C．去掉矩阵中多余的元素　　D．减少不必要的存储空间

(15) 下面哪种矩阵不属于特殊矩阵(　　)。
A．对角矩阵　　B．三角矩阵　　C．稀疏矩阵　　D．对称矩阵

(16) 一个$n\times n$的对称矩阵，如果以行或列为主序放入内存，则需要的存储单元个数为(　　)。
A．$n(n+1)/2$　　B．$n(n-1)/2$　　C．n^2　　D．$n^2/2$

(17) 若将n阶上三角矩阵 A 按列优先顺序压缩存放在一维数组 B[1..$n(n+1)/2$]中，则存放 B[k]中的非零元素$a_{i,j}$，$1\le i,j\le n$，的下标i,j与k的关系是(　　)。
A．$i(i+1)/2+j$　B．$i(i-1)/2+j-1$　C．$j(j+1)/2+i$　D．$j(j-1)/2+i-1$

(18) 若将n阶下三角矩阵 A 按列优先顺序压缩存放在一维数组 B[1..$n(n+1)/2$]中，则存放 B[k]中的非零元素$a_{i,j}$，$1\le i,j\le n$，的下标i,j与k的关系是(　　)。
A．$(j-1)(2n-j+1)/2+i-j$　　B．$(j-1)(2n-j+2)/2+i-j+1$
C．$(j-1)(2n-j+2)/2+i-j$　　D．$(j-1)(2n-j+2)/2+i-j-1$

2. 简答题

(1) 简述顺序队列假溢出的避免方法及队列满和空的判定条件。

(2) 简述①什么是递归程序？②递归程序的优、缺点是什么？③递归程序在执行时，应借助于什么来完成？④递归程序的入口语句、出口语句一般用什么语句实现？

(3) 讨论栈和队列的异同。

3. 算法设计题

(1) 两栈共享空间的存储结构。

如果同时使用具有相同数据类型的两个栈时，一种可取的方法是充分利用顺序栈单向延伸的特性，使用一个数组来存储两个栈，让一个栈的栈底为该数组的始端，另一个栈的栈底为该数组的末端，两个栈相向延伸。规定 top1 和 top2 分别为栈 1 和栈 2 的栈顶指针，StackSize 为整个数组空间的大小。请写出两栈共享空间的存储结构上的入栈算法和出栈算法。

【分析】数据结构定义参考为：

```
#define STACKSIZE 100
typedef struct BothStack
{
    int data[STACKSIZE ];
    int top1;
    int top2;
} BothStack;
```

```
void InitBothStack(BothStack S)
{
    S.top1 = -1;
    S.top2 = STACKSIZE;
}
```

(2) 不带头节点的循环链表表示队列，并且只设置一个指针指向队尾，不设头指针。设计相应的入队和出队算法。

【分析】出队操作是在循环链表的头部进行，相当于删除开始节点，入队操作是在循环链表的尾部进行，相当于在终端节点后插入一个节点。由于循环链表不带头节点，需要处理空表的特殊情况。

(3) 设计算法把一个十进制数转换为二至九进制之间的任意进制数输出。

【分析】算法的基本原理：$N=(N \operatorname{div} d)\times d+N \bmod d$，div是整除运算，mod为取余运算，先得到的余数为低位后输出，后得到的余数为高位先输出。因此将求得的余数放入栈中，再将栈元素依次输出即可得到转换结果。

(4) 马鞍点算法。若在矩阵 A 中存在这样一个元素 $a_{i,j}$，$0 \le i \le N-1, 0 \le j \le M-1$。该元素是第 i 行元素中的最小值，且又是第 j 列元素中的最大值，则称此元素为该矩阵的一个马鞍点。假设用一个二维数组来存储矩阵，即 `int A[N][M]`。请设计一个求该矩阵所有马鞍点的算法，将马鞍点数值输出打印，并分析算法的时间复杂度。

树和二叉树

1. 选择题

(1) 如果节点 A 有三个兄弟，B 是 A 的双亲节点，则节点 B 的度为(　　)。
A．1　　B．2　　C．3　　D．4

(2) 对一棵具有 n 个节点的树，树中所有度数之和为(　　)。
A．$n-1$　　B．n　　C．$n+1$　　D．不确定

(3) 下列说法中，正确的是(　　)。
A．二叉树就是度为 2 的树　　B．二叉树中不存在度大于 2 的节点
C．二叉树是有序树　　D．二叉树中每个节点的度均为 2

(4) 一个具有 1025 个节点的二叉树高度为(　　)。
A．11　　B．12　　C．11 - 1025　　D．12 - 1025

(5) 某完全二叉树的节点个数为 100，则第 60 个节点的度为(　　)。
A．0　　B．1　　C．2　　D．不确定

(6) 一个高度为 h 的满二叉树共有 n 个节点，其中有 m 个叶子节点，则(　　)。
A．$n=h+m$　　B．$2n=h+m$　　C．$m=h-1$　　D．$n=2^h-1$

(7) 设高度为 h 的二叉树只有度为 0 和 2 的节点，则此类二叉树中包含的节点数至少为(　　)。
A．$2h$　　B．$2h-1$　　C．$2h+1$　　D．$h-1$

(8) 每个节点的度或者为 0 或者为 2 的二叉树称为正则二叉树。n 个节点的正则二叉树中有多少片叶子？(　　)
A．$\lceil \log_2 n \rceil$　　B．$(n-1)/2$　　C．$\lceil \log_2 (n+1) \rceil$　　D．$(n+1)/2$

(9) 深度为 h 的满 m 叉树的第 k，$1 \le k \le h$，层有多少个节点？(　　)。
A．m^{k-1}　　B．m^k-1　　C．m^{h-1}　　D．m^h-1

(10) 一棵二叉树的先序(前序)遍历序列为 ABCDEFG，那它的中序遍历可能是(　　)。
A．CABDEFC　　B．BCDAEFG　　C．DACEFBG　　D．ADBCFEG

(11) 二叉树的前序序列和后序序列刚好相反，则该二叉树一定是(　　)。
A．空或只有根节点　　B．高度等于节点数
C．任一节点无左孩子　　D．任一节点无右孩子

(12) 任何一棵二叉树的叶子节点在先序、中序、后序遍历中相对次序(　　)。
A．肯定不发生改变　B．肯定发生改变　C．不能确定　D．有时发生变化

(13) 前序遍历和中序遍历结果相同的二叉树是(　　)。
A．根无左孩子　B．根无右孩子　C．所有节点只有左子树　D．所有节点只有右子树

(14) 设n和m是一棵二叉树上的两个节点，在中序遍历时n在m之前的条件是(　　)。
A．n在m的右方　B．n在m的左方　C．n是m的祖先　D．n是m的子孙

(15) 已知某完全二叉树采用顺序存储，节点数据信息存放顺序是ABCDEFGH，则该完全二叉树的后序遍历为(　　)。
A．HDEBFGCA　B．HEDBFGCA　C．HDEBAFGC　D．HDEFGBCA

(16) 下列的说法中，正确的是(　　)。
A．深度为k的二叉树中最多有2^k-1个节点，最少有k个节点
B．二叉树中一定存在度为2的节点
C．对二叉树的遍历是指先序、中序、后序遍历中的一种
D．构造线索二叉树的目的是为了能方便地找到每个节点的双亲

(17) 线索二叉树中的线索指的是(　　)。
A．左孩子　B．遍历　C．指针　D．标志

(18) 建立线索二叉树的目的是(　　)。
A．方便查找某节点的前驱和后继　B．方便二叉树的删除和插入操作
C．方便查找某节点的双亲　D．使二叉树的遍历结果唯一

(19) 具有n个节点的线索二叉树共有多少个线索(　　)。
A．$2n$　B．n　C．$n-1$　D．$n+1$

(20) 在线索二叉树中，一个节点是叶子节点的充分必要条件是(　　)。
A．左线索标志为0且右线索标志为1　B．左线索标志为1且右线索标志为0
C．左右线索标志皆为0　D．左右线索标志皆为0

(21) 关于线索二叉树，下列说法不正确的是(　　)。
A．中序线索二叉树中，若某节点有右孩子，则其后继是它的右子树的左分支末端节点
B．线索二叉树中利用二叉树的$n+1$个空指针域来存放其前驱和后继信息
C．在线索二叉树中，每个节点通过线索都可以直接找到它的前驱和后继
D．中序线索二叉树中，若某节点有左孩子，则其前驱是它的左子树的右分支末端节点

(22) 讨论树、森林和二叉树的关系，目的是为了(　　)。
A．借助二叉树上的运算方法去实现对树的一些运算
B．将树、森林按二叉树的存储方式进行存储并利用二叉树的算法来解决有关问题
C．将树、森林转换成二叉树
D．体现一种技巧，没有什么实际意义

(23) 由权值为{3,8,6,2,5}的叶子节点生成一棵哈夫曼树，其带权路径长度为(　　)。
A．24　　B．48　　C．53　　D．72

(24) 下述编码中哪一个不是前缀编码？(　　)
A．(00,01,10,11)　　B．(0,1,00,11)　　C．(0,10,110,111)　　D．(1,01,000,001)

(25) 用n个键值构造一棵二叉排序树，其最低高度为(　　)。
A．$n/2$　　B．n　　C．$\lfloor \log_2 n \rfloor$　　D．$\lfloor \log_2 n+1 \rfloor$

(26) 在含有n个节点的排序二叉树查找一个关键码，最多进行的比较次数为(　　)。
A．$n/2$　　B．n　　C．$\log_2 n$　　D．$\log_2 n+1$

(27) 二叉排序树中最小值的节点(　　)。
A．左指针一定为空　　B．右指针一定为空
C．左右指针一定为空　　D．左指针均不为空

(28) 已知 10 个元素{54,28,16,73,62,95,60,26,43}，按照依次插入的方法生成一棵二叉排序树，查找值为 62 的节点所需要的比较次数为(　　)
A．2　　B．3　　C．4　　D．5

(29) 按照某种遍历方式访问二叉排序树得到的序列是一个有序序列，该遍历为(　　)。
A．前序　　B．中序　　C．后序　　D．层序

(30) 在下列的二叉排序树中查找效率最高的是(　　)。
A．平衡二叉树　　B．二叉查找树　　C．没有左子树　　D．没有右子树

(31) 在二叉排序树上查找关键码为 28 的节点(假设存在)则依次比较的关键码可能是(　　)。
A．30,36,28　　B．38,48,28　　C．48,18,38,28　　D．60,30,50,40,38,36

(32) 关于二叉排序树，下面的说法正确的是(　　)。
A．二叉排序树是动态的树，在插入节点时会引起树的重新分裂或组合
B．对二叉排序树进行层序遍历可以得到有序序列
C．在构造二叉排序树时，若插入的关键码有序，则二叉排序树的深度最大
D．在二叉排序树中进行查找，关键码地比较次数不超过节点数目的一半

(33) 具有 5 层的平衡二叉树至少有多少个节点？(　　)
A．10　　B．12　　C．15　　D．17

(34) 在平衡二叉树中插入一个节点造成了不平衡，设最低的不平衡节点为 A，并已经知道 A 的左孩子的平衡因子为 0，右孩子的平衡因子为 1，则应该做哪种调整？(　　)
A．LL　　B．LR　　C．RL　　D．RR

(35) 一棵完全二叉树一定是一棵(　　)。
A．平衡二叉树　　B．哈夫曼树　　C．二叉排序树　　D．不确定

(36) 一棵深度为 k 的平衡二叉树，其每个非叶子节点的平衡因子均为 0，则该平衡二叉树的节点数目为(　　)。
A．$2^{k-1}-1$　　B．2^{k-1}　　C．$2^{k-1}+1$　　D．$2^{k}-1$

(37) 按照{12,24,36,90,52,30}的顺序构造成的平衡二叉树，其根节点是(　　)。
A．24　　B．36　　C．52　　D．30

2. 简答题

(1) 在顺序存储的二叉树中，编号为 i 和 j 的两个节点处在同一层的条件是什么？

(2) 设度为 m 的树采用多重链表存储。每个节点有 $m+1$ 个域，其中有 1 个数据域，m 个指向孩子的指针。则空指针的数目是多少？说明这种存储方式的利弊。

(3) 已知一棵二叉树的前序遍历序列为 ABCDEFGH，中序遍历序列分别和 CDBAFEHG。试构造该二叉树。
(4) 已知某二叉树的每个节点，要么其左右子树皆为空，要么其左右子树皆不为空。又知道该二叉树的前序序列为 JFDBACEHXIK，后序序列为 ACBEDXIHFKJ。试构造该二叉树。
(5) 已知一棵二叉树的层序遍历序列为 ABCDEFGHIJ，中序遍历序列为 DBGEHACIJF。试构造该二叉树。

(6) 为图 2-1 中所示二叉树建立中序线索二叉树。
(7) 将图 2-2 中所示的树转换为二叉树。

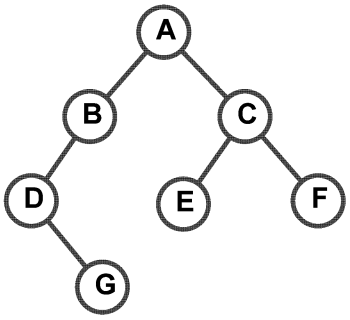


图 2-1

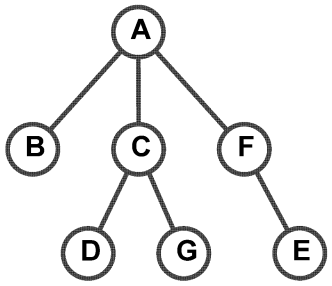


图 2-2

(8) 已知某字符串 S 中共有 8 种字符，各种字符出现的次数分别为 2 次、1 次、4 次、5 次、7 次、3 次、4 次和 9 次。对该字符串进行 0/1 进行前缀编码，问该字符串的编码至少有多少位？

3. 算法设计题

(1) 请写出二叉链表存储二叉树的先序、中序和后续遍历算法。

(2) 请写出二叉链表存储二叉树的按层序遍历算法。

(3) 二叉链表存储二叉树，要求不使用全局变量，请写出
①求叶子总数递归算法
②求节点总数递归算法
③求整个二叉树深度递归算法。

(4) 二叉排序树通常采用二叉链表存储，请写出交换左右子树的算法。

(5) 二叉排序树通常采用二叉链表存储，请写出二叉排序树的插入算法。设集合记录存放在数组 r[n]中，编写算法构造二叉排序树。

图

1. 选择题

(1) 在一个无向图中，所有顶点的度数之和等于所有边数的(　　)倍。
A．1/2　　B．1　　C．2　　D．4

(2) n 个顶点的强连通图至少有(　　)条边，其形状是(　　)。
A．n　　B．$n+1$　　C．$n-1$　　D．$n\times(n-1)$
E．无回路　　F．有回路　　G．环状　　H．树状

(3) 具有 10 个顶点的无向连通图最少有(　　)条边，最多有(　　)条边。
A．0　　B．9　　C．45　　D．90

(4) 含n个顶点的连通图中的任意一条简单路径，其长度不可能超过(　　)。
A．1　　B．$n/2$　　C．$n-1$　　D．n

(5) 在无向图中定义顶点v_i与顶点v_j之间的路径为从v_i到达v_j的一个(　　)。
A．顶点序列　　B．边序列　　C．权值之和　　D．边的条数

(6) 图的生成树(　　)，n 个顶点的生成树有(F)条边。
A．唯一　　B．不唯一　　C．唯一性不能确定
D．n　　E．$n+1$　　F．$n-1$

(7) 设无向图$G=(V,E)$和$G'=(V',E')$，如果G'是G的生成树，则下面的说法中错误的是(　　)。
A．G'是G的子图　　B．G'是G的连通分量
C．G'是G的极小连通子图且$V=V'$　　D．G'是G的一个无环子图

(8) G是一个非连通的无向图，共有 28 条边，则该图至少有(　　)个顶点。
A．6　　B．7　　C．8　　D．9

(9) 在一个具有 n 个顶点的无向图中包含有(　　)条边。
A．$n(n-1)/2$　　B．$n(n-1)$　　C．$n(n+1)/2$　　D．n^2

(10) 图中有关路径的定义是(　　)。
A．顶点序列中相邻顶点的序偶构成边　　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

(11) 一个具有n个顶点的无向图，最少有(　　)个连通分量，最多有(　　)个连通分量。
A．0　　B．1　　C．$n-1$　　D．n

(12) 用有向无环图描述表达式$(A+B)\times((A+B)/A)$，至少需要(　　)个顶点。
A．5　　B．6　　C．8　　D．9

(13) 以下关于有向图的说法中，正确的是(　　)。
A．强连通图中任何顶点到其他所有顶点都有弧
B．有向完全图一定是强连通图
C．有向图中某顶点的入度等于出度
D．有向图边集的子集可构成原有向图的子图

(14) 在如图 3-1 所示的有向图中，强连通分量的个数为(　　)。
A．1　　B．2　　C．3　　D．4

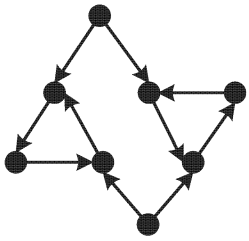


图 3-1　有向图的强连通分量

(15) 无向图 G 有 16 条边，度为 4 的顶点有 3 个，度为 3 的顶点有 4 个，其余顶点的度均小于 3，则图至少有(　　)个顶点。
A．10　　B．11　　C．12　　D．13

(16) 对于一个具有 n 个顶点的无向图，若采用邻接矩阵存储，则该矩阵的大小是(　　)。
A．n　　B．$(n-1)^2$　　C．$n-1$　　D．n^2

(17) 有一个图的邻接矩阵$\begin{bmatrix}0&2&3\\0&0&1\\7&5&0\end{bmatrix}$，可以看出，该图共有(　　)个顶点。

A．3　　B．6　　C．9　　D．以上答案均不正确

(18) 无向图的邻接矩阵是一个(　　)，有向图的邻接矩阵是一个(　　)。
A．上三角矩阵　　B．下三角矩阵　　C．对称矩阵　　D．无规律

(19) 具有n个顶点e条边的无向图采用邻接矩阵存储，则零元素的个数为(　　)。
A．e　　B．$2e$　　C．n^2-e　　D．n^2-2e

(20) 下列命题正确的是(　　)。
A．一个图的邻接矩阵表示是唯一的，邻接表表示也唯一
B．一个图的邻接矩阵表示是唯一的，邻接表表示不唯一
C．一个图的邻接矩阵表示不唯一，邻接表表示唯一
D．一个图的邻接矩阵表示不唯一，邻接表表示也不唯一

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

(22) 下列(　　)的邻接矩阵式对称矩阵。
A．有向图　　B．无向图　　C．AOV 网　　D．AOE 网

(23) 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

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

(25) 带权有向图 G 采用邻接矩阵存储，则顶点i的入度等于邻接矩阵中(　　)。
A．第i行非∞的元素之和　　B．第i列非∞且非 0 的元素个数
C．第i列非 0 的元素个数　　D．第i行非∞且非 0 的元素个数

(26) 下列说法中正确的是(　　)。
A．如果有向图的邻接矩阵是对称矩阵，则该有向图一定是有向完全图
B．如果某个图的邻接矩阵不是对称矩阵，则该图一定是有向图
C．如果某个图的邻接矩阵式对称矩阵，则该图一定是无向图
D．邻接矩阵表示法制存储了边的信息，没有存储顶点的信息

(27) 假设一个有向图具有 n 个顶点 e 条边，该有向图采用逆邻接矩阵存储，则删除与顶点i相关联的所有边的时间复杂度是(　　)。
A．$O(n)$　　B．$O(e)$　　C．$O(n+e)$　　D．$O(n*e)$

(28) 假设一个有向图具有n个顶点e条边，该有向图采用邻接表存储，则删除与顶点i相关联的所有边的时间复杂度是(　　)。
A．$O(n)$　　B．$O(e)$　　C．$O(n+e)$　　D．$O(n*e)$

(29) 用深度优先遍历方法便利一个有向无环图，并在深度优先遍历算法中按退栈次序打印出相应的顶点，则输出的顶点序列是(　　)。
A．逆拓扑有序　　B．拓扑有序　　C．无序　　D．顶点编号次序

(30) 图的 BFS 生成树的树高比 DFS 生成树的树高(　　)。
A．小或相等　　B．小　　C．大或相等　　D．大

(31) 图的广度优先遍历算法用到辅助队列，每个顶点最多进队(　　)次。
A．1　　B．2　　C．3　　D．不确定

(34) 下列说法不正确的是(　　)。
A．图的遍历是从给定的源点出发每一个顶点仅被访问一次

B．遍历的基本算法有两种：深度遍历和广度遍历
C．图的深度遍历不适用于有向图
D．图的深度遍历是一个递归过程

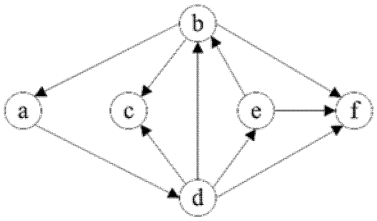


图 3-2 一个有向图

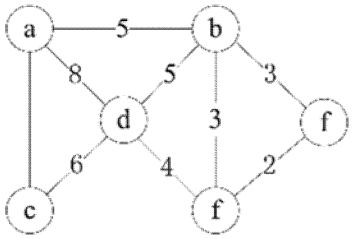


图 3-3 一个无相连通图

(35) 若一个无向图含有 n 个顶点 e 条边，该无向图采用邻接表存储，则深度优先遍历的时间复杂度为(　　)。
A．$O(n^2)$　　B．$O(e^2)$　　C.$O(n+e)$　　D．$O(e)$

(36) 对如图 3-2 所示的有向图从顶点 a 出发进行深度优先遍历，不可能得到的遍历序列是(　　)。
A．adbefc　　B．adcefb　　C．adcbfe　　D．adefbc

(37) 若要求连通图的生成树的高度最小，则采用(　　)方法。
A．深度优先遍历　　B．广度优先遍历　　C．Prim 算法　　D．Kruskal 算法

(38) 最小生成树指的是(　　)。
A．由连通网所得到的边数最少的生成树
B．由连通网所得到的顶点数相对较少的生成树
C．连通网中所有生成树中权值之和最小的生成树
D．连通网的极小联通子图

(39) 下列说法中，正确的是(　　)。
A．只要无相连通图中没有权值相同的边，其最小生成树就是唯一的
B．只要无相连通图中没有权值相同的边，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 条边的子图一定是图 G 的生成树

(40) 在图采用邻接表存储时，求最小生成树的 Prim 算法的时间复杂度(　　)。
A．$O(n)$　　B．$O(n+e)$　　C．$O(n^2)$　　D．$O(n^3)$

(41) Kruskal 算法的时间复杂度为(　　)，适合于求(E)的最小生成树。
A．$O(n\log_2 n)$　　B．$O(n+e)$　　C．$O(n^2)$　　D．$O(e\log_2 e)$
E．稀疏图　　F．稠密图　　G．连通图　　H．有向图

(42) 对如图 3-3 所示的无相连通网图从顶点 d 开始用 Prim 算法构造最小生成树，在构造过程中加入最小生成树的前 4 条边依次是(　　)。
A．$(d,f)4,(f,e)2,(f,b)3,(b,a)5$　　B．$(f,e)2,(f,b)3,(a,c)3,(f,d)4$
C．$(d,f)4,(f,e)2,(a,c)3,(b,a)5$　　D．$(d,f)4,(d,b)5,(f,e)2,(b,a)5$

(43) 求最短路径的 Dijkstra 算法的时间复杂度为(　　)。
A．$O(n)$　　B．$O(n+e)$　　C．$O(n^2)$　　D．$O(n*e)$

(44) 求最短路径的 Floyd 算法的时间复杂度(　　)。
A．$O(n^3)$　　B．$O(n+e)$　　C．$O(n^2)$　　D．$O(n*e)$

(45) 以下说法正确的是(　　)。
A．最短路径一定是简单路径
B．Dijkstra 算法不适用于有回路的网图
C．Dijkstra 算法不适用于求任意两顶点的最短路径
D．用 Floyd 算法求任意两顶点间的最短路径时，$Path_{k-1}[i][j]$一定是的子集

(46) 判定一个有向图是否存在回路除了可以利用拓扑排序方法外，还可以用(　　)。
A．求关键路径的方法　B．求最短路径的方法　C．广度优先遍历　D．深度优先遍历

(47) 已知有向图 $G=(V,E)$，其中 $V=\{V_1,V_2,V_3,V_4,V_5,V_6,V_7\}$，$E=\{<V_1,V_2>,<V_1,V_3>,<V_1,V_4>,<V_2,V_5>,<V_3,V_5>,<V_3,V_6>,<V_4,V_6>,<V_5,V_7>,<V_6,V_7>\}$，则 G 的拓扑序列是(　　)。
A．$V_1,V_3,V_4,V_6,V_2,V_5,V_7$　　B．$V_1,V_3,V_2,V_6,V_4,V_5,V_7$
C．$V_1,V_3,V_4,V_5,V_2,V_6,V_7$　　D．$V_1,V_2,V_5,V_3,V_4,V_6,V_7$

(48) 在有向图 G 的拓扑序列中，若顶点 V_i 出现在顶点 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

(50) 设某有向图含有 n 个顶点 e 条边，则该有向图采用邻接表表示，则拓扑排序算法时间复杂度为(　　)。
A．$O(n)$　　B．$O(n+e)$　　C．$O(n^2)$　　D．$O(n^3)$

(51) 若一个有向图中的全部顶点不能形成一个拓扑序列，则可断定该有向图(　　)。
A．是个根有向图　　B．是个强连通图
C．含有多人入度为 0 的顶点　　D．含有顶点数大于 1 的强连通图

(51) 若一个有向图中的全部顶点不能形成一个拓扑序列，则可断定该有向图(　　)。
A．是个根有向图　　B．是个强连通图
C．含有多人入度为 0 的顶点　　D．含有顶点数大于 1 的强连通图

(52) 下列说法中，正确的是(　　)
A．在拓扑排序算法中，暂存入度为 0 的顶点可以用栈，也可以用队列
B．AOV 网的拓扑序列是唯一的
C．若有向图的邻接矩阵中对角线以下的元素均为 0，则一定存在唯一的拓扑序列
D．若一个有向图存在拓扑序列，则该图一定是强连通图

(53) 设一项工程包含 7 个子工程，这些子工程之间有如下的优先关系：{1>2}、{2>3,2>4}、{3<5,7}、{4>5,6}、{7>6}，则这项工程有(　　)种可行的执行方案。
A．4　　B．5　　C．6　　D．7

(54) 下面关于工程计划的 AOE 网的叙述中，不正确的是(　　)。
A．关键活动不按期完成就会影响整个工程的完成时间
B．任何一个关键活动提前完成，那么整个工程将会提前完成
C．所有的关键活动都提前完成，那么整个工程将会提前完成
D．某些关键活动若提前完成，那么整个工程将会提前完成

(55) 关键路径是 AOE 网中(　　)。
A．从源点到终点的最长路径　　B．从源点到终点的最短路径
C．从源点到终点的边数最多的路径　C．从源点到终点的边数最少的路径

(56) 下面关于求关键路径的说法不正确的是(　　)。
A．求关键路径是以拓扑排序为基础的
B．一个事件的最早开始时间同以该事件为尾的弧的活动最早开始时间相同
C．一个事件的最迟开始时间为以该事件为尾的弧的活动最迟开始时间与该活动的持续事件的差
D．关键活动一定位于关键路径上

2. 简答题

(1) 图 3-4 所示是一个无向带权图，请分别按 Prim 算法和 Kruskal 算法给出最小生成树的求解过程。

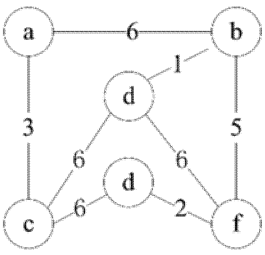
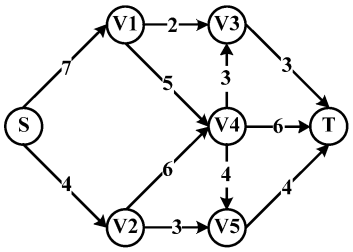


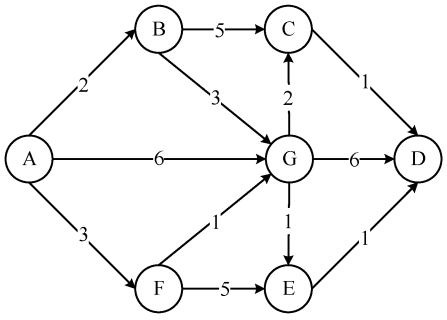
图 3-4 无向带权图

(2) 给定下图 AOV 网，求它的拓扑排序序列，并用标号法计算其关键路径。



(3) 给定有向图为 G=(V,E)，请分别画出相应的邻接矩阵和邻接表。其中，V={a,b,c,d,e,f}，E={<a,e>,<b,a>,<b,d>,<c,b>,<c,f>,<d,c>,<d,e>,<d,f>,<e,a>,<f,a>,<f,b>}。

(4) 根据给定的下图用 Dijkestra 算法计算从节点 A 开始的单源最短路径。



3. 算法设计

(1) 设计非递归算法完成图的深度优先遍历(C 语言描述)。

(2) 设计算法求无向图的深度优先生成树(C 语言描述)。

(3) 请给出采用邻接矩阵存储的 Floyd 算法(C 语言描述)。

查找

1. 选择题

(1) 静态查找和动态查找的根本区别在于(　　)。
A．它们的逻辑结构不一样　　B．施加在其上的操作不同
C．所包含的数据元素类型不一样　　D．存储实现不一样

(2) 平均查找长度与查找集合中记录个数 n 无关的查找方法是(　C)　。
A．折半查找　　B．平衡二叉树的查找　　C．散列查找　　D．不存在

(3) 在以下数据结构中，(　　)的查找效率最低。
A．有序顺序表　　B．二叉排序树　　C．堆　　D．散列表

(4) 对线性表进行顺序查找，要求线性表的存储结构为(　　)。
A．散列存储　　B．顺序存储　　C．链接存储　　D．顺序存储或链接存储

(5) 用顺序查找方法在长度为 n 的线性表中进行查找，在等概率情况下，查找成功的平均查找长度为(　　)。
A．n　　B．$n/2$　　C．$(n-1)/2$　　D．$(n+1)/2$

(6) 折半查找判定树不属于(　　)。
A．平衡二叉树　　B．二叉排序树　　C．完全二叉树　　D．二叉树

(7) 当 n 足够大时，在有序顺序表中进行折半查找，假设顺序表中每个元素的查找概率相同，则查找成功的平均查找长度为(　　)。
A．$(n+1)/2$　　B．$n/2$　　C．$\log_2(n+1)-1$　　D．$\log_2(n+1)$

(8) 对具有 14 个元素的有序表 $R[14]$ 进行折半查找，查找 $R[3]$ 时比较需要比较(　　)。
A．$R[0]R[1]R[2]R[3]$　　B．$R[6]R[2]R[4]R[3]$
C．$R[0]R[13]R[2]R[3]$　　D．$R[6]R[4]R[2]R[3]$

(9) 对有序表 $A[1..17]$ 进行折半查找，则查找长度为 5 的元素下表依次是(　　)。
A．8，17　　B．5，10，12　　C．9，16　　D．9,17

(10) 用 n 个键值构造一棵二叉排序树，其最低高度为(　　)。
A．$n/2$　　B．n　　C．$\lfloor \log_2 n \rfloor$　　D．$\lfloor \log_2 n+1 \rfloor$

(11) 在含有 n 个结点的二叉排序树中查找一个关键码，最多进行(　　)次比较。
A．$n/2$　　B．$\log_2 n$　　C．$\log_2 n+1$　　D．n

(12) 二叉排序树中，最小值结点的(　　)。
A．左指针一定为空　　B．右指针一定为空

C．左、右指针均为空　D．左、右指针均为空

(13) 已知 10 个元素(54,28,16,73,62,95,60,26,43)，按照依次插入的方法生成一棵二叉排序树，查找值为 62 的结点所需比较次数为(　　)。
A．2　　B．3　　C．4　　D．5

(14) 已知数据序列为(34,76,45,18,26,54,92,65)，按照依次插入结点的方法生成一棵二叉排序树，则该树的深度为(　　)。
A．4　　B．5　　C．6　　D．7

(15) 按(　　)遍历二叉排序树得到的序列式一个有序序列。
A．前序　　B．中序　　C．后序　　D．层次

(16) 下列二叉排序树中查找效率最高的是(　　)。
A．平衡二叉树　　B．二叉查找树
C．没有左子树的二叉排序树　　D．没有右子树的二叉排序树

(17) 在二叉排序树上查找关键码为 28 的结点(假设存在)，则依次比较的关键码有可能是(　　)。
A．30，36,28　　B．38，48，28
C．48，18，38，28　　D．60，30，50，40，38，36

(18) 不可能生成如图 4-1 所示二叉排序树的关键码序列是(　　)。
A．{4,2,1,3,5}　　B．{4,2,5,3,1}
C．{4,5,2,1,3}　　D．{4,5,1,2,3}

(19) 一棵二叉排序树如图 4-2 所示，该二叉树是(　　)。
A．平衡二叉树　　B．二叉排序树
C．堆　　D．以上都不是

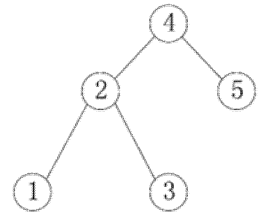


图 4-1　一棵排序二叉树

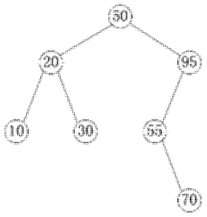


图 4-2　一棵二叉树

(20) 关于二叉排序树，下面说法中正确的是(　　)。
A．二叉排序树是动态树表，在插入新节点时会引起树的重新分裂或组合
B．对二叉排序树进行层次遍历可得到有序序列
C．在构造二叉排序树时，若插入的关键码有序，则二叉排序树的深度最大
D．在二叉排序树中进行查找，关键码的比较次数不超过节点数的一半

(21) 具有 5 层的平衡二叉树至少有(　　)个结点。
A．10　　B．12　　C．15　　D．17

(22) 在平衡二叉树中插入一个结点后造成了不平衡，设最低的不平衡结点为 A，并已知 A 的左孩子的平衡因子为 0 右孩子的平衡因子为 1，应做(　　)型调整以使其平衡。
A．LL　　B．LR　　C．RL　　D．RR

(23) 一棵完全二叉树一定是一棵(　　)。
A．平衡二叉树　　B．二叉排序树　　C．堆　　D．哈夫曼树

(24) 散列技术中的冲突指的是(　　)。
A．两个元素具有相同的序号　　B．两个元素的键值不同，而其他属性相同
C．数据元素过多　　D．不同键值的元素对应于相同的存储地址

(25) 设散列表表长 $m=14$，散列表 $H(k)=k \bmod 11$。表中已有 15、38、61、84 四个元素，如果用线性探测法处理冲突，则元素 49 的存储地址是(　　)。
A．8　　B．3　　C．5　　D．9

(26) 为一组关键码 {87,73,25,55,90,28,31,17,101,22,3,62} 构造散列表，设散列函数为 $H(key)=key \bmod 11$，在用链地址法处理后位于同一链表中的是(　　)。
A．81,90　　B．31,101　　C．3,78　　D．62,73

(27) 在采用线性探测法处理冲突所构成的散列表上进行查找，可能要探测多个位置，在查找成功的情况下，所探测的这些位置的键值(　　)。
A．一定都是同义词　　B．一定都不是同义词
C．不一定都是同义词　　D．$O(n^3)$

(28) 下面关于散列查找的说法正确的是(　　)。
A．散列函数越复杂越好，因为这样随机性好，冲突小
B．除留取余法是所有散列函数中最好的
C．不存在特别好与怀的散列函数，要视情况而定
D．若在散列表中删去一个元素，只要简单地将该元素删去即可

(29) 关于散列查找说法不正确的有几个(　　)。
□ 采用链地址法解决冲突，查找一个元素的时间是相同的
□ 采用链地址法解决冲突，若插入总是在链首，则插入任一个元素的时间是相同的
□ 用链地址法解决冲突易引起聚集现象
□ 再散列法不易产生聚集
A．1　　B．2　　C．3　　D．4

(30) 关于散列查找，下面说法正确的是(　　)。
A．再散列法处理冲突不会产生聚集
B．散列表的装填因子越大说明空间利用率越高，因此应使装填因子尽可能大
C．散列函数选择得好可以减少冲突现象
D．对任何关键码结合都无法找不到不产生冲突的散列函数

(31) 散列查找方法一般适用于(　　)情况下的查找。
A．查找表为链表　　B．查找表为有序表

C．关键码集合比地址集合大得多　　D．关键码集合与地址集合存在对应关系

(32) 设哈希表长 $m=15$，哈希函数 $H(key)=key \bmod 13$，关键码集合为{53, 17, 12, 61, 70, 87, 25, 64, 46}，采用二次探测再散列方法处理冲突，则查找成功的平均比较长度为(　　)。

A．1.4　　B．1.6　　C．1.8　　D．2.0

(33) 假设有 10 个关键码，他们具有相同的散列函数值，用线性探测法把这 10 个关键码存入散列表中至少需要做(　　)次探测。

A．110　　B．100　　C．55　　D．45

(34) 采用开放定址法解决冲突的散列查找中，发生聚集的原因主要是(　　)。

A．数据元素过多　　B．装填因子过大

C．散列函数选择不当　　D．解决冲突的算法不好

(35) 对具有 n 个关键码的散列表进行查找，平均查找长度是(　　)。

A．$O(\log_2 n)$　　B．$O(n)$　　C．$O(n\log_2 n)$　　D．与 n 无关

(36) 设某有向图含有 n 个顶点 e 条边，则该有向图采用邻接表表示，则拓扑排序算法时间复杂度为(　　)。

A．$O(n)$　　B．$O(n+e)$　　C．$O(n^2)$　　D．$O(n^3)$

2. 简答题

(1) 将关键字序列(7、8、1、2、20、28)散列存储到散列列表中，散列表的存储空间是一个下标从 0 开始到 6 的一个一维数组。散列函数为：$H(key)=key\%7$，处理冲突采用线性探测再散列法。(a) 请画出所构造的散列表；(b) 分别计算等概率情况下，查找成功的平均查找长度。

(2) 根据关键字序列(1、3、4、5、7、8、9、11、12)绘制出二分查找树。分别计算给出查找成功和查找失败的平均查找长度。

3. 算法设计

(1) 顺序查找算法。

(2) 折半查找非递归算法 BinSearch1。

(3) 折半查找递归算法 BinSearch2。

排序

1. 选择题

(1) 排序算法的稳定性是指(　　)。
A. 经过排序之后，能使值相同的数据保持原排序中的相对位置不变
B. 经过排序之后，能使值相同的数据保持原排序中的绝对位置不变
C. 排序算法的性能与被排序元素的数量关系不大
D. 排序算法的性能与被排序元素的数量关系密切

(2) 一个待排序的 n 个记录可分为 n/k 组，每组包含 k 个记录，且任一组内的各记录分别大于前一组内的所有记录切小于后一组内的所有记录，若采用基于比较的排序方法，其时间下限为(　　)。
A. $O(k\log_2 k)$　　B. $O(k\log_2 n)$　　C. $O(n\log_2 k)$　　D. $O(n\log_2 n)$

(3) 关于排序，下列说法正确的是(　　)。
A. 稳定的排序方法优于不稳定的排序方法，因为稳定的排序方法效率较高
B. 对同一个线性表使用不同的排序方法进行排序，得到的排序结果可能不同
C. 排序方法都是在排序表上实现的，在链表上无法实现排序方法
D. 在排序表上实现的排序方法在链表上也可以实现

(4) 下列不属于内部排序方法的是(　　)。
A. 归并排序　　B. 拓扑排序　　C. 堆排序　　D. 插入排序

(5) 用直接插入排序对下面的四个序列进行由小到大排序，元素比较次数最少的是(　　)。
A. 94,32,40,90,80,46,21,69　　B. 21,32,46,40,80,69,90,94
C. 32,40,21,46,69,94,90,80　　D. 90,69,80,46,21,32,94,40

(6) 当待排序序列基本有序或个数较小的情况下，最佳的内部排序方法是(　　)。
A. 直接插入排序　　B. 起泡排序　　C. 简单选择排序　　D. 快速排序

(7) 数据序列$\{8,9,10,4,5,6,20,1,2\}$只能是(　　)的两趟排序的结果。
A. 选择排序　　B. 冒泡排序　　C. 插入排序　　D. 堆排序

(8) 对数据序列$\{15,9,7,8,20,-1,4\}$进行排序，一趟后数据序列变为$\{9,15,7,8,20,-1,4\}$，则采用的是(　　)。
A. 选择排序　　B. 冒泡排序　　C. 插入排序　　D. 堆排序

(9) 下列排序算法中，(　　)可能会出现下列情况:在最后一趟开始之前，所有元素都不在最终位置上。
A. 起泡排序　　B. 插入排序　　C. 快速排序　　D. 堆排序

(10) 对有 n 个记录的线性表进行直接插入排序，在最后一趟的最坏情况下需比较(　　)次。

A．$n-1$　　B．$n+1$　　C．$n/2$　　D．$n(n-1)/2$

(11) 以下排序算法中，(　　)不能保证每趟至少能将一个元素放到其最终位置上。

A．快速排序　　B．希尔排序　　C．起泡排序　　D．堆排序

(12) 对数据序列$\{98,36,-9,0,47,23,1,8,10,7\}$采用希尔排序，下列(　　)是增量为 4 的排序结果。

A．$\{10,7,-9,0,47,23,1,8,98,36\}$

B．$\{-9,0,36,98,1,8,23,47,7,10\}$

C．$\{36,98,-9,0,23,47,1,8,7,10\}$

D．以上都不对

(13) 以下(　　)在数据序列基本有序时效率最好。

A．快速排序　　B．起泡排序　　C．堆排序　　D．希尔排序

(14) 下列序列中，(　　)是执行第一趟快速排序的结果。

A．$[da,ax,eb,de,bb]\ ff[ha,gc]$　　B．$[cd,eb,ax,da]\ ff[ha,gc,bb]$

C．$[gc,ax,eb,cd,bb]\ ff[da,ha]$　　D．$[ax,bb,cd,da]\ ff[eb,gc,ha]$

(15) 快速排序在(　　)情况下最不利于其长处。

A．待排序的数据量太大　　B．待排序的数据中含有多个相同值

C．待排序的数据已基本有序　　D．待排序的数据数量为奇数

(16) 对数列$\{25,84,21,47,15,27,68,35,20\}$进行排序，元素序列的变化情况如下:

□ 25,84,21,47,15,27,68,35,20　　□ 20,15,2125,47,27,68,35,84

□ 15,20,21,25,35,27,47,68,84　　□ 15,20,21,25,27,35,47,68,84

则采用的排序方法是(　　)。

A．希尔排序　　B．简单选择排序　　C．快速排序　　D．归并排序

(17) 一组记录的关键码为$\{46,79,56,38,40,84\}$，则利用快速排序的方法，以第一个记录为准得到的一次划分结果为(　　)。

A．$\{40,38,46,56,79,84\}$　　B．$\{40,38,46,79,56,84\}$

C．$\{40,38,46,84,56,79\}$　　D．$\{84,79,56,46,40,38\}$

(18) 就平均时间而言，(　　)最佳。

A．直接插入排序　　B．起泡排序　　C．简单选择排序　　D．快速排序

(19) 快速排序的最大递归深度是(　　)，最小递归深度是(　　)。

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

(20) (　　)方法是从未排序序列中挑选元素，并将其放入已排序序列的一端。

A．归并排序　　B．插入排序　　C．快速排序　　D．选择排序

(21) 对数据序列$\{84,47,25,15,21\}$进行排序，数据序列的变化为:$\{84,47,25,15,21\}$ $\rightarrow \{15,47,25,84,21\} \rightarrow \{15,21,25,84,47\} \rightarrow \{15,21,25,47,84\}$，这采用的排序方法是(　　)。

A．简单选择排序　　B．快速排序

C．起泡排序　　D．直接插入排序

(22) 简单选择排序的比较次数和移动次数分别是(　　)。

A．$O(n),O(\log_2 n)$　　B．$O(\log_2 n),O(n^2)$

C．$O(n^2),O(n)$　　D．$O(n\log_2 n),O(n)$

(23) 堆的形状是一棵(　　)。

A．二叉排序树　　B．满二叉树　　C．完全二叉树　　D．判定树

(24) 对数据序列$\{15,9,7,8,20,-1,7,4\}$，用堆排序的筛选法建立的初始堆为(　　)。

A．$\{-1,7,8,9,20,15,7,8\}$　　B．$\{-1,7,15,7,4,8,20,9\}$

C．$\{-1,4,7,8,20,15,7,9\}$　　D．以上都不对

(25) 下列说法中正确的是(　　)。

A．在链表上可以实现简单选择排序，但是难以实现堆排序

B．序列$\{30,40,50,15,25,35,38,10\}$是堆

C．在最坏情况下快速排序的时间性能也好于堆排序的时间性能

D．直接选择排序是一种稳定的排序方法

(26) 下面的序列中，(　　)是堆。

A．$\{1,2,8,4,3,9,10,5\}$　　B．$\{1,5,10,6,7,8,9,2\}$

C．$\{9,8,7,6,4,8,2,1\}$　　D．$\{9,8,7,6,5,4,3,7\}$

(27) 对关键码序列$\{23,17,72,60,25,8,68,71,52\}$进行堆排序，输出两个最小关键码后的剩余堆是(　　)。

A．$\{23,72,60,25,68,71,52\}$　　B．$\{23,25,52,60,71,72,68\}$

B．$\{71,25,23,52,60,72,68\}$　　D．$\{23,25,68,52,60,72,71\}$

(28) 在堆中删除一个关键码的时间复杂度为(　　)。

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

(29) 将两个各有 n 个元素的有序表归并成一个有序表，最少的比较次数是(　　)。

A．n　　B．$2n-1$　　C．$2n$　　D．$n-1$

(30) 若需在$O(n\log_2 n)$时间内完成对数组的排序，且要求排序是稳定的，这可选择的排序方法是(　　)。

A．快速排序　　B．堆排序　　C．归并排序　　D．希尔排序

(31) 二路归并的趟数是(　　)。

A．n　　B．$\log_2 n$　　C．$n\log_2 n$　　D．n^2

(32) (　　)在某趟排序结束后不一定能选出一个元素放到其最终位置上。
A．选择排序　　B．起泡排序　　C．归并排序　　D．堆排序

(33) 下列排序算法中，(　　)在最坏情况下的时间复杂度不高于$O(n\log_2 n)$。
A．起泡排序　　B．归并排序　　C．希尔排序　　D．快速排序

(34) 以下排序方法中，(　　)不需要进行关键码的比较。
A．快速排序　　B．归并排序　　C．基数排序　　D．堆排序

(35) 以下排序方法中，稳定的排序方法是(　　)。
A．快速排序　　B．希尔排序 C．基数排序　　D．堆排序

(36) 已知关键码序列$\{78,19,63,30,89,84,55,69,28,83\}$采用基数排序，第一趟排序后的关键码为(　　)。
A．$\{19,28,30,55,63,69,78,83,84,89\}$　　B．$\{28,78,19,69,89,63,83,30,84,55\}$
C．$\{30,63,83,84,55,78,28,19,89,69\}$　　D．$\{30,63,83,84,55,28,78,19,69,89\}$

(37) 将 1000 个英文单词进行排序，采用(　　)排序方法时间性能最好。
A．插入排序　　B．快速排序　　C．堆排序　　D．基数排序

(38) 假设线性表中每个记录有两个数据项$K1$和$K2$，现对线性按如下规则进行排序:先按数据项$K1$由小到大排序，在$K1$值相同的情况下，$K2$值小的在前面，大的在后面则应该采用的排序方法是(　　)。
A．先按$K1$值进行直接插入排序，再按$K2$的值进行简单选择排序
B．先按$K2$值进行直接插入排序，再按$K1$的值进行简单选择排序
C．先按$K1$值进行简单选择排序，再按$K2$的值进行直接插入排序
D．先按$K2$值进行简单选择排序，再按$K1$的值进行直接插入排序

(40) 下列排序方法中，时间性能与待排序记录的初始状态无关的是(　　)。
A．插入排序和快速排序　　B．归并排序和快速排序
C．选择排序和归并排序　　D．插入排序和归并排序

(41) 设要将序列$\{Q,H,C,Y,P,A,M,S,R,D,F,X\}$中的关键码按升序排列，则(　　)是起泡排序一趟扫描的结果，(　　)是增量为 4 的希尔排序扫描一趟的结果，(　　)是二路归并排序一趟的结果，(　　)是以第一个元素为轴值的快速排序一趟扫描的结果，(　　)是堆排序初始建堆的结果。
A．$\{F,H,C,D,P,A,M,Q,R,S,Y,X\}$　　B．$\{P,A,C,S,Q,D,F,X,R,H,M,Y\}$
C．$\{A,D,C,R,F,Q,M,S,Y,P,H,X\}$　　D．$\{H,C,Q,P,A,M,S,R,D,F,X,Y\}$
E．$\{H,Q,C,Y,A,P,M,S,D,R,F,X\}$

(42) 排序趟数与数据的原始状态有关的排序方法是(　　)。
A．直接插入排序　　B．简单选择排序　　C．快速排序　　D．归并排序

2. 算法设计

(1) 直接插入排序算法 InsertSort。

(2) 折半插入排序算法 BinSort。

(3) 希尔排序算法 ShellSort。

(4) 冒泡排序算法 BubbleSort。

(5) 快速排序一次划分算法 Partition。

(6) 快速排序算法 QuickSort。

(7) 快速排序非递归算法 Quicksort。

(8) 简单选择排序算法 SelectSort。

(9) 筛选法调整堆算法 Sift。

(10) 堆排序算法 HeapSort。

(11) 一次归并算法 Merge。

(12) 归并排序的递归算法 MergeSort。

算法分析

1. 简答题

(1) 描述递归方法及其特点

(2) 描述分治方法及其特点

(3) 描述动态规划法及其特点

2. 递归方程证明

(1) 用数学归纳法证明递归关系式$T(n)=2T(n/2)+1$的渐近解为$T(n)=O(n)$。

(2) 用数学归纳法证明递归式$T\left(n\right)=9T\left(n/3\right)+n^2$的渐近界为$T\left(n\right)=O\left(n^2\log n\right)$。

(3) 直接展开法求解递归式$T\left(n\right)=4T\left(n/2\right)+n^2$的渐近界。

3. 主方法求解递归方程

主方法是一类求解递归方程的快速便捷方法。它适用于求下面类型的递归形式：

$$T(n)=a\cdot T(\frac{n}{b})+f(n) \tag{1}$$

其中 $a\geq 0, b\geq 0$， $f(n)$ 为渐进函数。
求解方法是比较渐进函数 $f(n)$ 与 a,b 的指数形式 $n^{\log_b a}$。分三种情况进行讨论。
第一种情况，如果 $f(n)$ 可以被写作

$$f(n)=n^{\log_b a-\varepsilon}=n^{\log_b a}\cdot\frac{1}{n^{\varepsilon}} \tag{2}$$

其中 $\varepsilon\geq 0$。这说明 $f(n)$ 的增长速度比 $n^{\log_b a}$ 要慢，且慢 n^{ε} 倍。则可以直接解得 $T(n)$ 的时间复杂度为

$$T(n)=\Theta(n^{\log_b a}) \tag{3}$$

第二种情况，如果 $f(n)$ 可以被写作

$$f(n)=n^{\log_b a+\varepsilon}=n^{\log_b a}\cdot n^{\varepsilon} \tag{4}$$

其中 $\varepsilon\geq 0$。这说明 $f(n)$ 的增长速度比 $n^{\log_b a}$ 要快，且快 n^{ε} 倍。则可以直接解得 $T(n)$ 的时间复杂度为

$$T(n)=\Theta(f(n)) \tag{5}$$

第三种情况，如果 $f(n)$ 可以被写作

$$f(n)=C\cdot n^{\log_b a} \tag{6}$$

其中 $C=(\log n)^k, k\geq 0$。这说明 $f(n)$ 的增长速度与 $n^{\log_b a}$ 相同。则可以直接解得 $T(n)$ 的时间复杂度为

$$T(n)=\Theta((\log n)^{k+1}\cdot f(n)) \tag{7}$$

(1) 主方法求解下列递归关系式。

$$g(n) = 4 \cdot g(n/3) + 2 \cdot n$$

(2) 主方法求解下列递归关系式。

$$f(n) = 3 \cdot f(n/2) + n^2$$

(3) 主方法求解下列递归关系式。

$$T(n) = 2 \cdot T(n/2) + n$$

4. 动态规划求解

动态规划所处理的问题是一个多阶段决策问题。动态规划求解过程就可以用一个最优决策表来描述，最优决策表是一个二维表，其中行表示决策的阶段，列表示问题状态，表格需要填写的数据一般对应此问题的在某个阶段某个状态下的最优值，填表的过程就是根据递推关系，从 1 行 1 列开始，以行或者列优先的顺序，依次填写表格，最后根据整个表格的数据通过简单的取舍或者运算求得问题的最优解。

(1) 计算矩阵连乘积。给定 $n=4$ 个矩阵 $\{A,B,C,D\}$。要求以最小的计算量完成这 n 个矩阵的连乘积 $S=A_{35,40}\times B_{40,20}\times C_{20,10}\times D_{10,15}$。

(2) 0/1 背包问题。给定 N 中物品和一个背包。物品 i 的重量是 w_i,其价值 v_i，背包的容量为 C。问应该如何选择装入背包的物品，使得转入背包的物品的总价值为最大？

(3) 有编号分别为 a，b，c，d，e 的五件物品；它们的重量分别是 2，2，6，5，4；它们的价值分别是 6，3，5，4，6。现在给你个承重为 10 的背包，如何让背包里装入的物品具有最大的价值总和。

(4) 旅行计划规划。某同学想利用悠长寒假中的连续 14 天安排一系列激动人心旅行计划。他拟出的旅行活动目的地有五个$\{a,b,c,d,e\}$，以 14 天为单位，五项活动的启动日期分别是$S=\{6,1,7,6,11\}$，结束日期分别是$T=\{8,5,9,13,14\}$，将获得经验值$V=\{2,3,5,11,7\}$。请安排其旅行计划，以使得获得经验值最多。要求写出动态规划的递归关系式，并求解其最优解。

5. 分治策略设计

(1) 给定数组 `int array[n]`，设计一个算法，在最坏情况下用$n+\log n-2$次比较找出 `array` 中元素的最大数和次大数。

(2) 给定数组 int array[n]，n 为偶数。设计一个算法，在最坏情况下用 $3n/2-2$ 次比较找出 array 中元素的最大值和最小值。

編寫者的話：此數據結構習題册共計 50 頁系周益民選編，前後增删已歷五年，旨在普適性地選擇最典型的題型例子給予本科二年級學生用作日常復習和練習。周益民教學班所屬學員當使用此習題册無虞。懇請選課同學，若發現有字詞、語句、公式、數字、參考答案等任何疏漏，及時反饋給周益民以作修訂爲要，yiminzhou@uestc.edu.cn 。非常期待您的寶貴意見!