电子科技大学

数据结构与算法

习
题
册

姓名：_____

学号：_____

专业：□ 计算机大类

□ 网络空间安全

□ 物联网+实验班

□ 英才实验班

仅限 2020 年秋季使用
P0800335.06、P2700530.01、P0800335.01、H0816130.01

周益民 博士 教授
2020 年 9 月 7 日

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。

(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？

【分析】线性表$(0,1,2,\cdots,n-1)$的就地逆置即将对称元素交换。设线性表的长度为n，将线性表中第i个元素与第$n-i-1$个元素交换。

【解答】

```
void Reverse (SeqList L)
```

(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？

【分析】首先顺序查找元素x在表L中的位置i，然后将位置i以后的元素向后移动一个位置，最后将x插入。

【解答】

```
void Insert (SeqList L, int x)
```

(3) 设计算法实现将单链表就地逆置？

【分析】扫描过程中要设置两个指针 p 和 pre，另外还需要一个临时指针 r。其中 pre 始终指向原表中 p 的前驱，r 暂存节点 p 的后继节点的地址。

【解答】

```
void Reverse (Node *first)
```

(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。

【解答】

```
Node * Union (Node * la, Node * lb)
```

(5) 用顺序表表示集合，设计算法实现集合的求交集运算。

【分析】本题的要求即$C = A \bigcap B$，C中元素即是A和B中的公共元素。扫描A，A中的元素A.data[i]若与表B中某个元素相同，则表示是交集中的元素，将其放入到C中。

【解答】

```
void Interest(SeqList A, SeqList B, SeqList C)
```

(6) 用顺序表表示集合，设计算法实现集合的求并集运算。

【分析】本题的要求即$C = A \bigcup B$，C中元素即是A和B中非重复出现的所有元素。先将表A复制到C中，然后扫描B。对B中的元素B.data[i]若与表A中所有元素皆不相同，则表示是并集中的元素，将其放入到C中。

【解答】

```
void Union(SeqList A, SeqList B, SeqList C)
```

(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。

【解答】

```
void Store(Node * head, float n, int m) // head为单链表的头指针
```

(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
```

【解答】

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;
}
```

【解答】

```
void PushBothStack(BothStack S, int i, int intnumber);
int PopBothStack(BothStack S, int i);
```

(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。

【解答】

```
void Enqueue (Node *rear, ElemType x)  void Dequeue (Node * rear)
```

(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。

【解答】

```
void Decimaltor (int num, int r)
```

(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。

【解答】

```
void Andian (int A[][], int M, int N)
```