中 国 科 学 技 术 大 学

2022--2023学年第一学期《数据结构》练习题

得分: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 专业:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

姓名:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 学号:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**一、单项选择题（2分×10=20分）**

1. 下列程序段的时间复杂度为\_\_\_**A**\_\_\_。

i=0，s=0； while (s<n) {s=s+i；i++；}

A. O(n1/2) B. O(log2n) C. O(n) D. O(n2)

1. 设指针q指向单链表中结点A，指针p指向单链表中结点A的后继结点B，指针s指向被插入的结点X，则在结点A和结点B插入结点X的操作序列为\_\_\_**B**\_\_\_。

A. s->next=p->next；p->next=s； B. q->next=s；s->next=p；

C. p->next=s->next；s->next=p； D. p->next=s；s->next=q；

1. 若元素a、b、c、d、e、f依次进栈，允许进栈、退栈操作交替进行，但不允许连续三次进行退栈操作，则不可能得到的出栈序列是\_\_\_**D**\_\_\_。

A．d c e b f a B．c b d a e f C．b c a e f d D．a f e d c b

1. 设二维数组A[1..m,1..n]按行存储在一维数组B[1..mn]中，则二维数组元素A[i,j]在一维数组B中的下标为\_\_\_**A**\_\_\_。

A．n\*(i-1)+j B． n\*(i-1)+j-1 C．i\*(j-1) D．j\*m+i-1

1. 如果某二叉树中序序列的最后一个结点与后序序列的最后一个结点是同一个结点，则这棵二叉树\_\_\_\_**B**\_\_\_\_。

A. 左子树为空 B. 右子树为空 C. 左、右子树都不空 D. 只有一个叶子

1. 5个字符有如下4种编码方案，不是前缀编码的是\_\_\_**D**\_\_\_。

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

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

1. 中缀表达式(a-b)\*(c+d)的后缀表达式是\_\_\_\_\_**B**\_\_\_\_\_。  
   A. abcd+\*- B. ab-cd+\* C. ab-\*cd+ D. a-bcd+\*

8. 设F是一个森林，B是由F转换得到的二叉树，F中有N个非终端结点，则B中右指针域为空的结点有\_\_\_**C**\_\_个。

A N-1 B N C N+1 D N+2

9. 用单循环链表表示的队列,若仅设一个指针,但要求出队和入队操作方便,应当选用\_\_\_**B**\_\_\_。

A 头指针 B 尾指针

10. 若仅已知某结点A的地址，则\_\_**A**\_\_结构不能删除A结点。

A 单链表 B 双向链表 C 循环链表

**二、判断题。（正确的在括号里打√，错误的打×，共13分）**

1. 【×】线性表的长度是线性表所占用的存储空间的大小。
2. 【×】线性表的顺序存储结构优于链式存储结构。
3. 【×】顺序存储只能用于存储线性结构。
4. 【×】在对链队列做出队操作时，不会改变front指针的值。
5. 【×】算法的时间复杂度仅取决于问题的规模。
6. 【×】哈夫曼树中可能存在度为1的结点。
7. 【×】完全二叉树中，若一个结点没有右孩子，则必是树叶。
8. 【√】完全二叉树的某结点若无左孩子，则它必是叶结点。
9. 【×】若A和B都是一棵二叉树的叶子结点，则存在这样的二叉树，其前序遍历序列为…A…B…,而中序遍历序列为…B…A…。
10. 【√】存在这样的二叉树，对它采用任何次序的遍历，其遍历产生的结点序列相同。
11. 【×】由一棵二叉树的前序序列和后序序列可以唯一确定它。
12. 【×】二叉树第i层上有2i-1个结点，深度为h的二叉树上有2h-1个结点。
13. 【√】将一棵树转换为二叉树后，根结点没有右子树。

**三、填空题（1分×13=13分）**

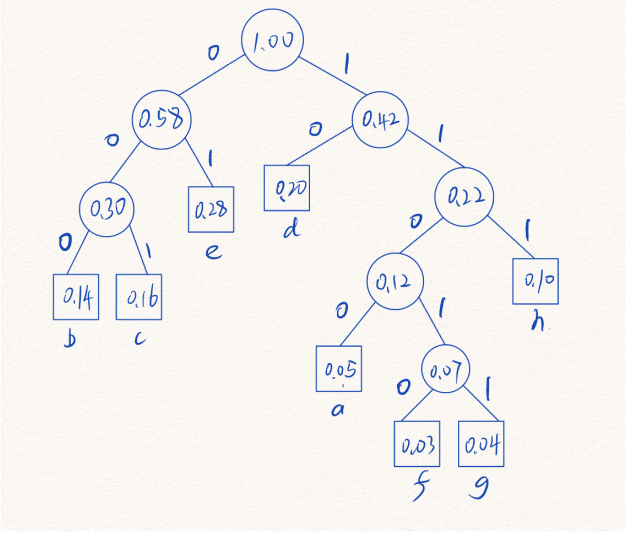
1. 二叉树第i(根为的第一层)层上的结点数目最多为\_\_\_**2i-1**\_\_\_；深度为k的二叉树至多有\_\_**2k-1**\_\_\_个结点。
2. 设高度为h的二叉树上只有度为0和度为2的结点，则此类二叉树中所含的结点数至少为\_\_\_\_\_\_\_\_2h-1\_\_\_\_\_\_\_，至多为\_\_\_\_\_\_\_\_\_2h-1\_\_\_\_\_\_\_\_\_\_\_。
3. 用一维数组Q[0..19]存放一个循环队列，队头指示器front指向队头元素的位置，队尾指示器rear指向队尾元素的下一个位置，队列中元素个数最多不超过19个。当front＝18、rear＝3时，该队列中有 **5** 个元素。若此后又进行了4次出队操作和1次入队操作，则front和rear的值分别为 **2** 和 **4** 。
4. 已知一棵二叉树的前序和中序序列分别为：ABCDEFG和CBDEAFG,它的后序序列是\_\_ CEDBGFA \_\_\_\_\_。
5. 数据元素之间的关系在计算机中有两种不同的表示方法：顺序映像和非顺序映像，并由此得到两种不同的存储结构：**顺序存储**和**链式存储**。
6. 具有*n*个节点的完全二叉树的深度为⌈log2 (n+1)⌉或 ⌊log2n ⌋+1。
7. 若森林F 有15 条边、25 个结点，则F 包含树的个数是\_\_\_\_**10**\_\_\_\_\_\_\_。
8. 在长度为n 的有序单链表中插入一个新结点，并仍然保持有序的时间复杂度是\_\_\_\_\_\_\_**O(n)**\_\_\_\_\_\_\_\_\_。
9. 包含999个结点的完全二叉树中的叶子结点数为\_\_\_500\_\_\_\_。

**四、解答题（共54分）**

1. 假设用于通信的电文仅由‘a’~‘h’的8个字母组成，字母在电文中出现的频率分别为0.05,0.14,0.16,0.20,0.28,0.03,0.04,0.10。试给出这8个字母的哈夫曼编码方案；若电文由100个字母组成，请计算电文长度约有多少个二进制位。（15分）

解：

1. 哈夫曼树：



1. 哈夫曼编码：

|  |  |
| --- | --- |
| 字符 | 编码 |
| a | 1100 |
| b | 000 |
| c | 001 |
| d | 10 |
| e | 01 |
| f | 11010 |
| g | 11011 |
| h | 111 |

1. 电文长度：

记pi ,di为第i个字母的频率和编码长度，则电文长度为

n = = 271

1. 假设一字符串用单循环链表表示，链表中每一结点存储一个字符，链表中最后一个结点存放的是字符串中的最后一个字符。试编写一算法判断该字符串是否有中心对称关系，例如，xyzzyx、xyzyx都算是中心对称的字符串。（15分）

代码：

int Judge(List\*l){

    Stack S;

    int i;

    List\*p = l;

    char ch;

    for(i = 0; i < l->length; i++){

        Push(S, p->data);

        p = p->next;

    }//P中的元素入栈

    for(i = 0; i < l->length; i++, l=l->next){

        if(Pop(S,ch) && ch != l->data)

            break;

    }

    if(i == l->length)

        return True;

    else

        return False;

}

1. 假设二叉树中结点的存储结构定义如下：

typedef struct BTNode{

ElemType data;  
struct BTNode \*lchild, \*rchild, \*next;  
}BTNode, \*BiTree;

其中lchild和rchild用来保存结点的左、右孩子结点的指针，next保存结点在后序遍历中的直接后继结点的指针。初始时，二叉树中各结点的next均为空。

各题代码如下：

1. 试编写一算法，为二叉树中各结点的next填上合适的指针值，即其后序遍历的直接后继结点的指针。（8分）

BTNode Pre;//全局变量记录上一个访问的结点

void FindNext(BTNode\*T)

{

    if(T)

    {

        FindNext(T->lchild);

        FindNext(T->rchild);

        if(Pre)

        {

            Pre->next = T;

        }

        Pre = T;

    }

}

1. 试编写一算法，给定两个指针ptrx和ptry指向树上节点，判断这两个节点是否是祖先和子孙关系。（8分）

bool judge(BTNode \* ptrx, BTNode \*ptry){    // x是y祖先 或 y是x祖先，均返回true

    if(!ptrx || !ptry) return false;

    bool func(BTNode \* ptrx, BTNode \*ptry);

    // ptrx == ptry 不算祖先/子孙关系

    return func(ptrx->lchild,ptry) || func(ptrx->rchild,ptry) || \

           func(ptry->lchild,ptrx) || func(ptry->rchild,ptrx);

}

bool func(BTNode \* ptrx, BTNode \*ptry){     // x是y的祖先 或 x=y 时返回true

    if(!ptrx) return false;

    if(ptrx == ptry) return true;

    return func(ptrx->lchild,ptry) || func(ptrx->rchild,ptry);

}

1. 求出指定结点ptr在给定二叉树root中的层次。（假设根在第1层）（8分）

int compute\_deep(BTNode \* root, BTNode\* ptr){

    if(!root) return 0;

    if(ptr == root) return 1;

    int left = compute\_deep(root->lchild, ptr), right = compute\_deep(root->rchild, ptr);

    if(left == right && left == 0) return 0;    // ptr不在树root中

    return left > right ? left + 1 : right + 1; // left与right有1个是0，有一个不是0 //（ptr不在左子树就是右子树）

}