数据结构课程考试

“数据结构算法应用”与“编程”

两种题型的考试样题

数据结构算法应用题

（1）根据栈的入栈顺序判断合法的出栈顺序并能给出解释

设A,B,C,D,E五个元素依次进栈（允许元素进栈后立即出栈）。问能否得到下列元素出栈序列。若能得到，则给出相应的push和pop操作序列；

若不能，则说明理由。

(1)E,D,C,B,A

(2)C,A,B,D,E

(3)A,C,E,B,D

(4)A,B,C,D,E

（2）串的KMP匹配算法的算法过程，能写出next数组。

目标串：aabaabaaab

模式串：abaabc

写出模式串的next数组

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 模式串t | a | b | a | a | b | c | a | c |
| next[j] | 0 | 1 | 1 | 2 | 2 | 3 | 1 | 2 |

对于模式串t，求解next数组的基本步骤为:

1. next[1]=0, next[2]=1。
2. 以后求解每一位的next[j]值时，根据j的前一位进行比较， 令k=next[j-1]。
3. 将t[j-1]与t[k] 进行比较，如果相等，则next[j]=k+1；如果不等，令k= next[k]，若k不等于0，重复步骤(3)，直到k等于0, next[j]=1。

根据以上步骤，对于所给的模式串“abaabc”，next[1]=0, next[2]=1，其余next[j]值求解过程如下：

1. j=3时，k=next[j-1]= next[2]=1，观察t[2]与t[k]是否相等，t[2]=b，t[1]=a，不等，此时k’= next[k]= next[1]=0，所以next[j] =1。
2. j=4时，k=next[j-1]= next[3]=1，观察t[3]与t[k]是否相等，t[3]=a，t[1]=a，相等，所以next[j]= k+1= 2。
3. j=5时，k=next[j-1]= next[4]=2，观察t[4]与t[k]是否相等，t[4]=a，t[2]=b，不等，此时k’= next[k]= next[2]=1，观察t[4]与t[1]是否相等，t[4]=a，t[1]=a，相等，所以next[j]= k’+1=1+1=2。

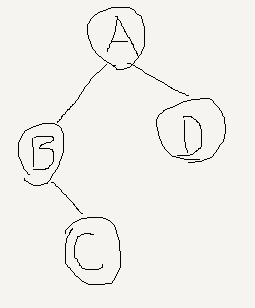
后面依次类推。

（3）根据二叉树的先序和中序（或者中序和后序）遍历结果给出对应的二叉树

P147 应用题（2）中的第一个问题

（要求画出对应的二叉树，并写出剩下的一个遍历结果）

（4）线索二叉树的创建（理解线索二叉树的结构体类型定义，并画出线索二叉树的线索）



typedef struct TBTNode

{

ElemType data;

sturct TBTNode \*lchild, \*rchild;

int ltag, rtag;

}TBTNode, \*TBTree;

用以上的数据类型定义，建立中序遍历线索二叉树，写出每个结点ltag与rtag的值，左右孩子指向的结点。

例：

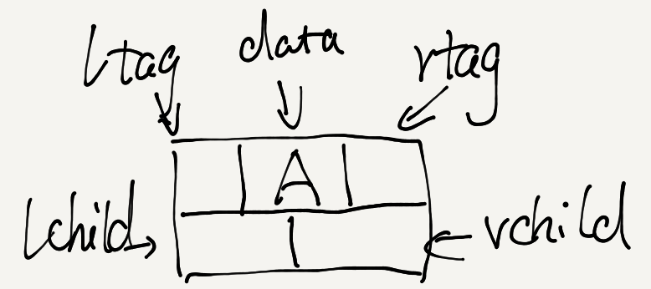
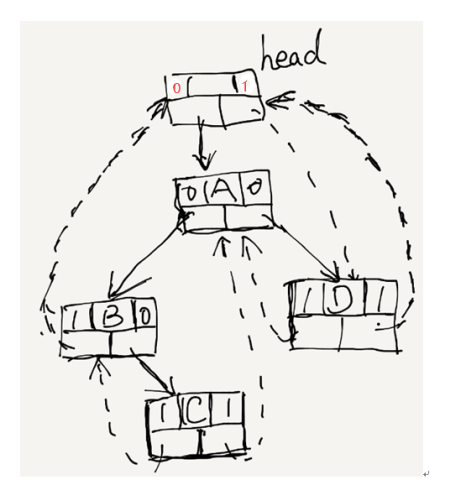
(以下面的图做为参考，考试时不需要在试卷上画出来（可以画在草稿纸上），只需要把下面各个结点中成员的值写出来即可)

结点A：ltag=0, rtag=0,lchild指向结点B,rchild指向D。

结点B：xxxxxxxxx, lchild指向head结点，xxxxxx。

结点C：……

……

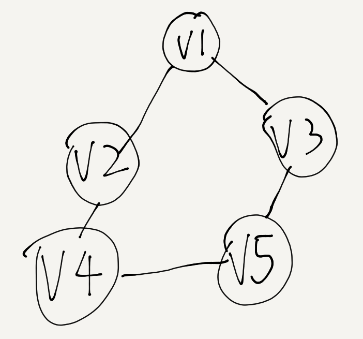
 

（5）哈夫曼树的创建与哈夫曼编码

参考P147 应用题（3）中的第一个问题和P142例5.3

（要求：左分支标记0，右分支标记1，先画出哈夫曼树，再给出每个字母对应的哈夫曼编码，看清创建哈夫曼树创建时的要求（例如，要求创建时左孩子结点的值小等））

（6）图的邻接矩阵，根据邻接矩阵写出DFS和BFS遍历结果



画出图的邻接矩阵：

以V1顶点为起点的DFS遍历结果：

以V1顶点为起点的BFS遍历结果：

（7）折半查找的查找过程

参考P194例7.1

有以下的有序线性表，在这个中使用折半查找算法查找21

11 12 13 21 23 25 30 35 36 37

写出查找过程中的比较次数，每次比较时low, higt和mid的值

例：第1次比较 low = 1,higt=10,mid=5

第2次比较：xxxxxx

……

共比较XX次查找成功

（8）快速排序的排序过程

参考P244的快速排序算法排序过程

编程题

（1）数据结构的基本算法（单链表的插入和删除，循环队列的出队入队，二叉树的遍历）

有以下的单链表的定义，请写出插入数据元素的函数

typedef struct LNode{

ElemType elem;

struct LNode \*next;

}LNode, \*LinkList;

// 参数含义：L-单键表头结点指针，i-插入位置下标，e-插入数据元素的值

int ListInsert(LinkList &L, int i, ELemType e)

{

}

P34-35 算法2.9，算法2.10

#define ERROR 0

#define OK 1

typedef struct LNode{

ElemType elem;

struct LNode \*next;

}LNode, \*LinkList;

//单链表的插入

int ListInsert(LinkList &L, int i, ELemType e)

{

LinkList p=L,s; //等价于 LNode \*p=L,\*s;

int j=0;

while(p &&(j<i-1))

{

p=p->next;

j++;

}

if(!p||(j>i-1)) return ERROR;

s=new LNode; //等价于s=(LNode \*)malloc(sizeof(LNode));

s->elem=e;

s->next=p->next;

p->next=s;

return OK;

}

//单链表的删除

int ListDelete(LinkList &L, int i)

{

LinkList p=L,q; //等价于 LNode \*p=L,\*q;

int j=0;

while((p->next) &&(j<i-1))

{

p=p->next;

j++;

}

if(!(p->next)||(j>i-1)) return ERROR; // 若p->next为空，则第i个结点不存在

q=p->next; //q指向第i个结点

p->next=q->next;

free(q);

return OK;

}

P72 算法3.13，算法3.14

#define ERROR 0

#define OK 1

#define MAXQSIZE 100 //最大长度

Typedef struct {

QElemType \*base; //初始化的动态分配存储空间

int front; //队头指针

int rear; //队尾指针

}SqQueue;

//循环队列的入队

int EnQueue(SqQueue &Q,QElemType e){

if((Q.rear+1)%MAXQSIZE==Q.front) return ERROR; //队列满

Q.base[Q.rear]=e;

Q.rear=(Q.rear+1)%MAXQSIZE;

return OK;

}

//循环队列的出队

int DeQueue (SqQueue &Q,QElemType &e)

{

if(Q.front==Q.rear) return ERROR; //队列空

e=Q.base[Q.front];

Q.front=(Q.front+1)%MAXQSIZE;

return OK;

}

P123 算法5.1

typedef int TElemType;

typedef struct BiNode{

TElemType data;

struct BiNode \*lchild,\*rchild; //左右孩子指针

}BiNode,\*BiTree;

//二叉树的中序遍历

void InOrderTraverse(BiTree T)

{

InOrderTraverse(T->lchild);

printf("%d ",T->data);

InOrderTraverse(T->rchild);

}

//二叉树的先序遍历

void PreOrderTraverse(BiTree T)

{

printf("%d ",T->data);

PreOrderTraverse(T->lchild);

PreOrderTraverse(T->rchild);

}

//二叉树的后序遍历

void PostOrderTraverse(BiTree T)

{

PostOrderTraverse(T->lchild);

PostOrderTraverse(T->rchild);

printf("%d ",T->data);

}

（2）递归

P127-128 算法5.5，算法5.6

//求二叉树的深度

int Depth(BiTree T)

{

if(T==NULL) return 0;

int m=Depth(T->lchild);

int n=Depth(T->rchild);

if(m>n) return m+1;

else return n+1;

}

//求二叉树的结点的个数

int NodeCount(BiTree T)

{

if(T==NULL) return 0;

return NodeCount(T->lchild)+NodeCount(T->rchild)+1;

}

（3）排序

基于下面的顺序表定义，请写出该顺序表（直接插入/冒泡/简单选择）排序的函数。（是基于关键字的排序，按照题目中定义的变量名写程序）

#define MAXSIZE 100

typedef int KeyType;

typedef struct{

KeyType key;

InfoType info;

}ElemType;

typedef struct{

//从下标为1的数组元素开始保存数据元素

ElemType elem[MAXSIZE+1];

int length;

}SqList;

void sort(SqList &list){

}

P237 算法8.1

//直接插入排序

void InsertSort(SqList ＆L)

{

int i,j;

for(i=2;i<L.length;++i)

if( L.elem[i].key<L.elem[i-1].key)//当elem[i]的关键字小于elem[i-1]时，需将L.elem[i]插入有序子表

{

L.elem[0]=L.elem[i]; // 复制为哨兵

L.elem[i]=L.elem[i-1];

for(j=i-2; L.elem[0].key<L.elem[j].key;--j) //从后向前寻找插入位置

L.elem[j+1]=L.elem[j]; // 记录逐个后移，直到找到插入位置

L.elem[j+1]=L.elem[0]; //将elem[0]即原elem[i]，插入到正确位置

}

}

P242 算法8.4

//冒泡排序

void bubble\_sort(SqList &L)

{

int m=L.length-1,i,j,flag=1; //flag用来标记某一趟排序是否发生交换

ElemType x;

while((m>0)&&(flag==1))

{

flag=0; //flag置为0，若本趟排序没有发生交换，则无需执行下一趟排序

for(j=1;j<=m;j++)

if(L.elem[j].key>L.elem[j+1].key)

{

flag=1; //flag置为1，表示本趟排序发生了交换

x=L.elem[j];L.elem[j]=L.elem[j+1];L.elem[j+1]=x; //交换

}

m--;

}

}

P246 算法8.6

//简单选择排序

void SelectSort(SqList &K)

{

int i,j,k;

ElemType t;

for (i=1; i<L.length; ++i)

{ //在L.r[i..L.length] 中选择key最小的记录

k=i; //k用于记录此趟排序中关键字最小的下标位置

for( j=i+1;j<=L.length ; j++)

if ( L.elem[j].key <L.elem[k].key) k=j;

if(k!=i) t=L.elem[i],L.elem[i]=L.elem[k],L.elem[k]=t;

}

}