桂 林 电 子 科 技 大 学 试 卷

2020-2021 学年第 1 学期　　课号 2010 423-403-411-439-586-416-407

课程名称:数据结构与算法 (A 卷，闭卷) 适用年级+专业 2019 级计科、软工、智能、物联网

考试时间 120 分钟　学号＿＿＿＿＿＿＿＿＿＿　姓名＿＿＿＿＿＿＿＿

题　号	一					二				成绩
	1	2	3	4	5	6	7	8	9	
满　分	10	10	10	10	10	10	12	12	16	**100**
得　分										
评卷人										

一、应用题（共 50 分）

1. （本题 10 分）一组记录的关键码为 {46, 79, 56, 38, 40, 84，12}，则

(1) 写出一趟希尔排序后的结果；（3 分）

(2) 利用快速排序的方法，以第一个记录为基准，写出一趟排序后的结果；（3 分）

(3) 若使用堆排序，且排序后的关键码要求是递增序，画出初始堆；（4 分）

2.（本题 10 分）给定一组字符以及它们出现的权值（括号内的值为各字母出现的权值），
D(7),E(31),I(19)，L(23),A(11),S(5),V(4)

(1) 画出对应的哈夫曼树；（在构造哈夫曼树时要求每棵子树的左孩子的权值不大于右孩子的权值）(4 分)

(2) 给出每个字符的哈夫曼编码；（4 分）

(3) 若有编码序列 “00011001011011101010111”，请依据上面的 huffman 树写出译码结果。 (2 分)

3.（本题 10 分）将序列{4, 5, 8, 2, 1, 3, 6}中的整数依次插入一棵空的平衡二叉树中，要求依次画出插入各整数后得到的平衡二叉树。

4.将关键字序列{7，8，30，11，18，9，14}散列存储到散列表中。散列函数为：H（key）=（key*3）%7,处理冲突采用线性探查法，要求负载因子为 0.7 。

(1) 请画出所构造的散列表；（7 分）

(2) 分别计算在等概率情况下，查找成功和不成功时的平均查找长度（3 分）。

5. （本题 10 分）已知图 G 的邻接矩阵如下所示，其顶点 V={ v1, v2, v3, v4, v5, v6, v7, v8}。

	v1	v2	v3	v4	v5	v6	v7	v8
	0	6	∞	3	∞	50	∞	∞
	6	0	43	11	6	∞	∞	∞
	∞	43	0	∞	∞	∞	∞	8
	3	11	∞	0	12	∞	∞	∞
	∞	6	∞	12	0	1	24	∞
	50	∞	∞	∞	1	0	12	∞
	∞	∞	∞	∞	24	12	0	12
	∞	∞	8	∞	∞	∞	12	0

(1) 画出图 G（3 分）

(2) 若图中 8 个顶点代表城市，边的权值代表城市之间修建公路花费的代价，请选择能沟通每个城市，且总造价最小的 7 条公路，写出选择过程（7 分）

二、算法分析及设计题（共 50 分）

6 .（本题 10 分）试分别写出以下 4 个算法片段的时间复杂度，用大 O 阶表示。

（1）
```
int num1, num2 ;
for( int i =0;i<n;i++)
{
   num1 = num1 + 1;
   for(int j = 1 ;j <=n ;j = j*2)
     num2 = num2 + num1 ;
      }
```

(2)
```
int i = 1 ,s = 0;
while( i < n)
{
  s = s + i;
  i = i+1;
  }
```

(3)
```
int fun(int *a ,int n)
{
  int min =a[0];
  for(int i =1; i<n ;i = i+2)
    if(min > a[i] ) min = a[i];
 return min;
}
int  test   ( int a[][12] , int n)
{
   for(int i = 0 ;i< n;i++)
      printf("%d  ", fun(a[i], 12);
 }
```

（4）
```
void  func(int  n)
{ int  i=0,s=0;
  while  (s<=n)
    { i++;
      s=s+i;
    }
}
```

7.（本题 12 分）若有程序如下：

```
#include <stdio.h>
void qSort(int a[],int n)//对数组 a 中 n 个元素进行排序
{ int flag = 0 ,tmp;
  for(int i = 0;i < n-1 ;i++)
  {
   flag = 0 ;
    for(int j = 0;j < n-i-1;j++)
   if(a[j] > a[j+1])
    {
      tmp = a[j]; a[j] = a[j+1] ; a[j+1] = tmp;
        flag = 1 ;
        }
      if(flag==0) break;
      for(int i=0;i<n;i++)
        printf("%d ",a[i]);
      printf("\n");
}
}
int main(void)
 {
   int b[]={23,14,5,15,67,8,98};
   qSort(b+1,5);

}
```

写出程序运行后的输出结果（9 分），并说明 flag 在函数中的作用（3 分）。

8.（本题 12 分）给定顺序线性表定义如下：

```
typedef  int  DataType;
struct seqList
{
  int MaxNum; //用于记录顺序线性表中能存放的最大元素个数
  DataType *Elem;// 用于存放顺序线性表数据元素的连续空间的起始地址
  int curNum;  //线性表中现有元素个数
};
```

假设一组未排序的整数放在顺序线性表 L 中，请编写算法找出其中没有出现的最小的正整数。要求：算法时间复杂度不超过 O(n)；例如，{-5，3，2，3}，未出现的最小正整数是 1；{1，2，3}，未出现的最小正整数是 4。

```
int findMin(struct seqList*L)
{//完成算法，找出 L 中没有出现的最小的正整数


}
```

9.（本题 16 分）

以下给出了栈和二叉树的抽象数据类型，以及二叉树定义和相关操作的部分实现，采用左-右指针表示法存储二叉树：

typedef DataType struct BinTreeNode;

//栈抽象数据类型

ADT Stack is

Operations

Stack createEmptyStack (void);

int isEmptyStack (Stack st);

void push (Stack st, DataType x);

void pop (Stack st);

DataType top (Stack st);

End ADT Stack

//二叉树抽象数据类型

ADT BinTree is

Operations

BinTree createEmptyBinTree (void); //创建空二叉树

BinTreeNode root (BinTree t); // 求二叉树 t 的根

BinTree leftChild (BinTree t); //求二叉树 t 的左孩子

BinTree rightChild (BinTree t); //求二叉树 t 的右孩子

End ADT BinTree

```
struct BinTreeNode{
    int value;
    ___（1）_____;        //请完善二叉树结构定义（3 分）
    };

typedef struct BinTreeNode *BinTree;
```

(2)函数 Traverse 的周游策略是先序遍历、中序遍历还是后序遍历？___（2）_____（3 分）

```
void Traverse(BinTree root)
{
    Stack s=createEmptyStack();
    BinTree t=root;
    if(t==NULL)    return;
    do{
        while (t!=NULL) { push(s,t);    t=leftChild(t); }
        t=top(s);    pop(s);    visit(root(t));    t=rightChild(t);
    } while ( t!=NULL || !isEmptyStack(s) );
}
```

（3）函数 Swap( )是在二叉树周游的过程中，通过交换每一非叶子结点的左右子树，使得左子树根结点总是大于右子树根结点（若只有右子树或左子树，均将其置为左子树），请填空。（10 分，每空 2.5 分）

```
void Swap(BinTree root){
   BinTreeNode p=root, q;
   if (p==NULL) return;
   else
      if ( p->lchild==NULL)    { p->lchild=p->rchild; p->rchild=NULL; }
      else if ( (p->rchild!=NULL) && ( _____（3）_____ ) )
      { q=p->lchild;    _____（4）____;     ____（5）_____; }

   Swap( leftChild(root) );
   _____（6）_____;
}
```