

2020 年南昌航空大学硕士研究生入学考试试卷

考试科目：数据结构

(答案做在答题纸上，做在试卷上无效)

一、解答题 (60 分，每小题 10 分)

1、已知一通讯系统有 10 个符号:A、B、C、D、E、F、G、H，它们出现的频率依次为：7、19、2、6、32、3、21、10，试画出对应的哈夫曼树（任何层次要求左子树根结点的权小于右子树），并写出各符号的哈夫曼编码。

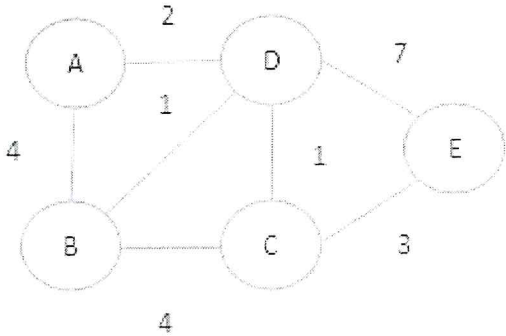
2、一棵二叉树的先序遍历序列为：A C G F K J B I，中序遍历序列为：F G C K J A I B

要求：(1) 画出该二叉树；
(2) 写出该二叉树的后序序列。

3. 下面是一个有向图的邻接矩阵，请画出对应的有向图，并且写出其一个拓扑排序序列。

	A	B	C	D	E	F
A	∞	5	∞	7	∞	∞
B	∞	∞	4	∞	∞	∞
C	∞	∞	∞	∞	∞	9
D	∞	∞	5	∞	∞	6
E	∞	∞	∞	5	∞	∞
F	∞	∞	∞	∞	∞	∞

4. 一个无向图如下图所示，用 Dijkstra 算法求从顶点 A 到其它各顶点的最短路径和距离，要求写出详细求解过程。



终点	从 V_i 到各终点的 D 值和最短路径的求解详细过程					
	i=1	i=2	i=3	i=4		
B						
C						
D						
E						
V_j						
S						

5. 设哈希表的地址范围为 0~12，哈希函数为： $H(K)=K \% 13$ ，K 为关键字，采用开放定址法中的线性探测再散列解决冲突，探测序列为 1、2、3.，依次输入 11 个关键字：12、28、17、19、36、69、2、10、76、59，构造出哈希表，试回答下列问题：

- (1) 画出哈希表示意图；
- (2) 求装填因子 α ；
- (3) 假定每个关键字的查找概率相等，求查找成功时的平均查找长度。

散列地址	0	1	2	3	4	5	6	7	8	9	10	11	12
关键字													
查找次数													

6. 给出一组关键字 T= (35、63、49、32、43、80、26、48、37、73)。

采用下列算法从小到大排序，写出第一趟结束时的序列：

- (1) 希尔排序（第一趟步长为 5）
- (2) 快速排序（选第一个记录为枢轴）
- (3) 堆排序（建好大堆的状态）

二、算法设计题（90 分，每小题 15 分，请用 C 语言或类 C 语言写出实现算法的函数）

1、设有两个按元素值非递增有序排列的链式线性表 LA 和 LB，请编写算法将 LA 表和 LB 表合并成一个按元素值非递减有序排列的链式线性表 LC，要求使用 LA 和 LB 的结点，不生成新结点。线性表的链式存储结构如下：

```
typedef int ElemType;
typedef struct LNode{
    ElemType data; // 存储空间基址
    struct LNode *next;
}LNode, *LinkList;
```

2、写出括号匹配算法。假设在一个算术表达式中，可以包含三种括号：“(”和“)”、“[”和“]”、“{”和“}”，并且这三种括号可以按任意的次序嵌套使用。

比如，... [... { ... } ... [...] ...] ... [...] ... (...) ...

现在需要设计一个算法，用来检验在输入的算术表达式中所使用括号的合法性。

3、写出字符串的一般模式匹配算法。字符串的存储结构如下：

```
typedef char SString[maxlen+1]; // 0 单元存放字符串长度。
```

4. 写出稀疏矩阵的快速转置算法。稀疏矩阵以三元组顺序表存储结构存储，具体存储结构定义如下：

```
#define maxsize 12500
typedef struct{
    int i,j;
    ElemType e;
```

```

}Triple;
typedef struct{
    Triple data[maxsize+1]; // data[0]未用
    int mu,nu,tu;
}TSMatrix;

```

5、编写一个算法，返回二叉树 T 中节点的值(data)大于 60 的节点个数（如果 T 为空，则返回 0）。二叉树的链式存储结构对应的节点类型定义如下：

```

typedef int elemType;
typedef struct node{
    elemType data;
    struct node *lchild, *rchild;
}BiNode,BiTree;

```

6、写出二叉树的层次遍历算法，二叉树的存储结构如上题，队列基本操作如下：

```

InitQueue (&Q); //初始化队列
EnQueue (&Q, e); //入队
DeQueue (&Q, &e); //出队
EmptyQueue (Q); //判断队列是否为空;

```