姓名 学号 专业班级 学院、系 密 封 线

齐鲁工业大学 2022/2023 学年第 1 学期《数据结构》

参考答案

（ B 卷）

一、简答题（每题 5 分，共 20 分）

1、求网的最小生成树可使用 Prim 算法，时间复杂度为 O(n2)，此算法适用于边较多的稠密图，也可使用 Kruskal 算法，时间复杂度为 O(eloge)，此算法适用于边较少的稀疏图。

2、树的遍历方法有先根序遍历和后根序遍历，它们分别对应于把树转变为二叉树后的先序遍历与中序遍历方法。

3. 三种方法对查找的要求分别如下：

- 顺序查找法：表中元素可以任意存放；
- 折半查找法：表中元素必须以关键字的大小递增或递减的次序存放：
- 分块查找法：表中元素每块内的元素可任意存放，但块与块之间必须以关键字的大小递增（或递减）存放，即前一块内所有元素的关键字都不能大于（或小）后一块内任何元素的关键字。

三种方法的平均查找长度分别如下：

- 顺序查找法：查找成功的平均查找长度为 $\frac{n+1}{2}$；
- 折半查找法：查找成功的平均查找长度为 log2(n+1)+1；
- 分块查找法：若用顺序查找确定所在的块，平均查找长度为 $\frac{1}{2}(\frac{n}{s}+s)+1$；若用折半确定所在块，平均查找长度为 $\log_2(\frac{n}{s}+1)+\frac{s}{2}$。

4. 当 n=1 时，结论显然成立。

设 n<=k 时结论成立，当 n=k+1 时，设一棵二叉树的后序序列是 $u_1,u_2,\cdots,u_n$，中序序列是 $u_{p_1},u_{p_2},\cdots,u_{p_n}$，可知 u_n 是二叉树的根结点，设 $p_j=n$，可知 $\{u_{p_1},u_{p_2},\cdots,u_{p_{j-1}}\}$ 是左子树的结点集合，$\{u_{p_{j+1}},u_{p_{j+2}},\cdots,u_{p_n}\}$ 是右子树的结点集合，进一步可知：

（1）左子树的后序序列是 $u_1,u_2,\cdots,u_{j-1}$，中序序列是 $u_{p_1},u_{p_2},\cdots,u_{p_{j-1}}$，由归纳假设知序列 1,2,⋯,j-1 可以通过一个栈得序列 $p_1,p_2,\cdots,p_{j-1}$。

（2）右子树的后序序列是 $u_j,u_{j+1},\cdots,u_{n-1}$，中序序列是 $u_{p_{j+1}},u_{p_{j+2}},\cdots,u_{p_n}$，设 $u'_1=u_j$，$u'_2=u_{j+1}$，⋯，$u'_{n-j}=u_{n-1}$；$u'_{p'_1}=u_{p_{j+1}}$，$u'_{p'_2}=u_{p_{j+2}}$，⋯，$u'_{p'_{n-j}}=u_{p_n}$，则 $p'_1=p_{j+1}-j+1$，$p'_2=p_{j+2}-j+1$，⋯，$p'_{n-j}=p_n-j+1$，由归纳假设知序列 1,2,⋯,n-j 可以通过一个栈得序列 p'_1，p'_2，⋯，p'_{n-j}，显然按同样的方式，j,j+1,⋯,n-1 可以通过一个栈得序列 $j-1+p'_1,j-1+p'_2,\cdots,j-1+p'_{n-j}$，也就是 p_{j+1}，p_{j+2}，⋯，p_n。

由（1）（2）及 $p_j = n$ 可知由 1,2,…,n 可通过一个栈得到序列 $p_1, p_2, \cdots, p_n$。由数学归纳法可知本题结论成立。

二、分析题（每题 10 分，共 20 分）

1. 答：DEBCA

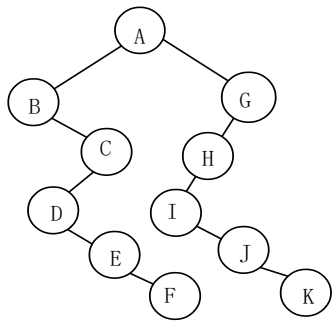
2. 答：E={(1,5),(5,2),(5,3),(3,4)},W=10

三、应用题（每题 10 分，共 30 分）

1. 答：(8,9,4,3,6,1),10,(12,18,18)

2. 答：ASL=7/6

3. (1) ABCDEF;　BDEFCA；(2) ABCDEFGHIJK;　BDEFCAIJKHG 林转换为相应的二叉树；



四、算法和计算题（每小题 10 分，共 20 分）

1. 设计一个在链式存储结构上统计二叉树中结点个数的算法。

```
void countnode(bitree *bt,int &count)
{
    if(bt!=0)
{count++; countnode(bt->lchild,count); countnode(bt->rchild,count);}
}
```

2. 设计一个算法将无向图的邻接矩阵转为对应邻接表的算法。

```
typedef struct {int vertex[m]; int edge[m][m];}gadjmatrix;
typedef struct node1{int info;int adjvertex; struct node1 *nextarc;}glinklistnode;
typedef struct node2{int vertexinfo;glinklistnode *firstarc;}glinkheadnode;
void adjmatrixtoadjlist(gadjmatrix g1[ ],glinkheadnode g2[ ])
{
int i,j; glinklistnode *p;
for(i=0;i<=n-1;i++) g2[i].firstarc=0;
for(i=0;i<=n-1;i++) for(j=0;j<=n-1;j++)
if (g1.edge[i][j]==1)
{
p=(glinklistnode *)malloc(sizeof(glinklistnode));p->adjvertex=j;
p->nextarc=g[i].firstarc; g[i].firstarc=p;
p=(glinklistnode *)malloc(sizeof(glinklistnode));p->adjvertex=i;
p->nextarc=g[j].firstarc; g[j].firstarc=p;
}
}
```

3.

（1）查询链表的尾结点

（2）将第一个结点链接到链表的尾部，作为新的尾结点

（3）返回的线性表为（$a_2,a_3,\cdots,a_n,a_1$）