四川大学“精品课程”

计算机科学与技术专业（本科）

《数据结构与算法分析》课程

考试说明与模拟试卷

第一部分　考试说明

数据结构与算法分析》是计算机科学与技术专业统设的一门重要的必修专业基础课，它主要研究数据的各种逻辑结构和在计算机中的存储结构，还研究对数据进行的插入、查找、删除、排序、遍历等基本运算或操作以及这些运算在各种存储结构上具体实现的算法。由于本课程的主教材采用 C++语言描述算法，期末卷面考试也采用 C++语言描述，因而要求在做平时作业和上机实验操作时用 C++开发工具（如：Visual C++或 C++ Builder 或 Borland C++）。

下面按照主教材中各章次序给出每章的具体复习要求，以便同学们更好地进行期末复习。

第一章　绪论

重点掌握的内容：

1. 数据结构的二元组表示，对应的图形表示，序偶和边之间的对应关系。
2. 集合结构、线性结构、树结构和图结构的特点。
3. 抽象数据类型的定义和表示方法。
4. **一维和二维数组中元素的按下标和按地址的访问方式以及相互转换，元素地址和数组地址的计算，元素占用存储空间大小和数组占用存储空间大小的计算。**
5. 普通函数重载和操作符函数重载的含义，定义格式和调用格式。
6. 函数定义中值参数和引用参数的说明格式及作用，函数被调用执行时对传送来的实际参数的影响。
7. **算法的时间复杂度和空间复杂度的概念，计算方法，数量级表示。**
8. **一个简单算法的最好、最差和平均这三种情况的时间复杂度的计算。**

对于本章的其余内容均作一般掌握。

第二章　线性表

重点掌握的内容：

1. **线性表的定义及判别**和抽象数据类型的描述，线性表中每一种操作的功能，对应的函数名、返回值类型和参数表中每个参数的作用。
2. 线性表的顺序存储结构的类型定义，即 List 类型的定义和每个域的定义及作用。
3. **线性表的每一种运算在顺序存储结构上实现的算法，及相应的时间复杂度**。
4. **链接存储的概念，线性表的单链接和双链接存储的结构，向单链表中一个结点之**

后插入新结点或从单链表中删除一个结点的后继结点的指针链接过程。

5. 单链表中结点的结构，每个域的定义及作用，即 LNode 类型的定义及结构。

6. **带表头附加结点的链表、循环链表、双向链表的结构特点。**

7. 线性表的每一种运算在单链表上实现的算法及相应的时间复杂度。

8. **在顺序存储或链接存储的线性表上实现指定功能的算法的分析和设计。**

9. Josephus 问题的求解过程。

10. 顺序表和线性链表的性能比较及各自使用背景。

对于本章的其余内容均作一般掌握。

第三章　数组和广义表

重点掌握的内容：

1. 多维数组的逻辑结构特征。

2. **多维数组的顺序存储结构及地址计算公式。**

3. 数组是一种随机存取结构的原因。

4. 特殊矩阵和稀疏矩阵的概念。

5. **特殊矩阵（包括对角矩阵）和压缩存储的下标变换方法及所需存储空间。**

6. **稀疏矩阵的定义和三元组线性表及三列二维数组表示。**

7. 稀疏矩阵的顺序存储、带行指针向量的链接存储，在每一种存储中非零元素结点的结构。

8. 稀疏矩阵的转置运算。

9. 广义表的定义和表示，广义表长度和深度的计算。

10. **广义表上的求表头、表尾运算。**

5. 广义表的链接存储结构中结点类型的定义，分别求广义表长度和深度的递归算法，它们对应的时间复杂度。

一般掌握的内容：

稀疏矩阵转置的算法描述。

对于本章的其余内容均作一般了解。

第四章　栈和队列

重点掌握的内容：

1. 栈的定义和抽象数据类型的描述，栈中每一种操作的功能，对应的函数名、返回值类型和参数表中每个参数的作用。

2. 栈的顺序存储结构的类型定义，即 Stack 类型的定义和每个域的定义及作用。

3. 栈的每一种运算在顺序存储结构上实现的算法，及相应的时间复杂度。

4. 栈的每一种运算在链接存储结构上实现的算法及相应的时间复杂度。

5. 算术表达式的中缀表示和后缀表示，以及相互转换的规则，后缀表达式求值的方法。

6. **给定 n 个栈元素，出栈可能或不可能的序列数。**

7. 队列的定义和抽象数据类型的描述，队列中每一种操作的功能，对应的函数名、返回值类型和参数表中每个参数的作用。

8. 队列的顺序存储结构的类型定义，即 Queue 类型的定义和每个域的定义及作用。

9. 队列的每一种运算在顺序存储结构上实现的算法及相应的时间复杂度。

10. 利用栈和队列解决简单问题的算法分析和设计。

11. 双端队的概念及可能出队序列。

12. **队和栈的应用背景，如 cpu 队、进程队、打印机队。**

13. 链队的各种存储表示。

一般掌握的内容：

1. 后缀表达式求值的算法，把中缀表达式转换为后缀表达式的算法。

2. 队列的链接存储结构，以及实现每一种队列运算的算法和相应的时间复杂度。

对于本章的其余内容均作一般了解。

第五章　字符串

重点掌握的内容：

1. 串的有关概念及基本运算。

2. 串与线性表的关系。

3. 串的各种存储结构。

4. **一个串中真子串和子串个数的确定。**

一般掌握的内容：

1. 串上各种运算的实现及其时间性能分析。

2. 使用 C++提供的操作函数构造与串相关的算法解决简单的应用问题。

第六章　树和二叉树

重点掌握的内容：

1. 树和二叉树的定义，对于一棵具体树和二叉树的二元组表示及广义表表示。

2. 树和二叉树的概念，如结点的度、树的度、树叶、分枝结点、树的层数、树的深度等。

3. 不同结点数的树和二叉树的形态。

4. 树和二叉树的性质，如已知树或二叉树的深度 h 可求出相应的最多结点数，已知结点数 n 可求出对应树或二叉树的最大和最小高度。

5. 二叉树中结点的编号规则和对应的顺序存储结构。

6. 二叉树的链接存储结构及存储结点的类型定义，即 BTreeNode 类型的定义和每个域的定义及作用。

7. 二叉树的先序、中序、后序、遍历的递归过程和递归算法，中序遍历的非递归算法，按层遍历的过程和算法，每种算法的时间复杂度。

8. 二叉树的先序、中序、后序遍历序列，唯一确定一棵二叉树的原则。

9. 算术表达式的二叉树表示及逆波兰表达式、中缀表示。

一般掌握的内容：

1. 普通树的链接存储结构，GTreeNode 类型的定义和每个域的定义及作用。

2. **普通树的先根、后根和按层遍历的过程**及算法。

3. 在链接存储的二叉树上实现指定功能的算法分析和设计。

对于本章的其余内容均作一般了解。

二叉树的应用

重点掌握的内容：

1. **二叉搜索树的定义和性质、建立。**

2. 二叉搜索树查找的递归算法和非递归算法，相应的时间复杂度，**查找一个元素的查找长度，即从树根结点到该结点的路径上的结点数。**

3. 二叉搜索树插入的递归算法和非递归算法，相应的时间复杂度，根据一组数据生成一棵二叉搜索树的过程。

4. **堆的定义和顺序存储结构，小根堆和大根堆的异同及堆的判别、建立堆的过程。**

5. 向堆中插入元素的过程、算法描述及时间复杂度。

6. 从堆中删除元素的过程、算法描述及时间复杂度。

7. **哈夫曼树的定义，树的带权路径长度的计算，根据若干个叶子结点的权构造哈夫曼树的过程。**

8. **顺序二叉树及二叉链表表示二叉树。**

9. **已知关键字序列{22,16,38,89,56，16，79}，试构造平衡二叉树。**

对本章的其余内容均作一般了解。

第七章　图

重点掌握的内容：

1. 图的顶点集和边集的表示。

2. 图的一些概念的含义，如顶点、边、度、完全图、子图、路径、路径长度、连通图、权、网等。

3. **图的邻接矩阵、邻接表、**邻接多重表和十字链表四种存储结构及相应的空间复杂度。

4. 存储图使用的 vexlist, adjmatrix, adjlist, edgenode, edgeset, edge 等类型的定义及用途。

5. 图的深度优先和广度优先搜索遍历的过程。

6. 对分别用邻接矩阵和用邻接表表示的图进行深度优先搜索遍历的过程、算法描述以及相应的时间复杂度。

7. 对分别用邻接矩阵和用邻接表表示的图进行广度优先搜索遍历的过程、算法描述以及相应的时间复杂度。

8. **图的生成树（若一个具有 n 个顶点，e 条边的无向图是一个森林（n>e），**

则该森林中必有多少棵树。）、深度优先生成树和广度优先生成树、生成树的权、最小生成树等的定义。

9. **根据普里姆算法求图的最小生成树的过程。**

10. 根据克鲁斯卡尔算法求图的最小生成树的过程。

11. 图的拓扑序列和拓扑排序的概念，求图的拓扑序列的方法，对用邻接表表示的图进行拓扑排序的过程。

12. **强连通图的最少边数。**

一般掌握的内容：

1. 根据普里姆算法求图的最小生成树的算法描述。
2. 根据克鲁斯卡尔算法求图的最小生成树的算法描述。
3. 对用邻接表表示的图进行拓扑排序的和算法描述。

对本章的其余内容均作一般了解。

第八章　查找

重点掌握的内容：

1. **在一维数组及单链表上进行顺序查找的过程、算法、成功及不成功的平均查找长度和时间复杂度。**

2. 在一维数组上进行二分查找的过程、递归和非递归算法、平均查找长度和时间复杂度，二分查找一个给定值元素的查找长度（即查找路径上的元素数），二分查找对应的判定树的性质。

3. 散列存储的概念，散列函数、散列表、冲突、同义词、装填因子等术语的含义。

4. **利用除留余数法建立散列函数求元素散列地址的方法。**

5. **利用开放定址法中的线性探查法处理冲突进行散列存储和查找的过程，利用链接法处理冲突进行散列存储和查找的过程。**

6. **根据除留余数法构造散列函数，采用线性探查法或链接法处理冲突，把一组数据散列存储到散列表中，计算出一个给定值元素的查找长度和查找所有元素的平均查找长度。**

7. B_树中每个结点的结构，树根结点或非树根结点中关键字的个数范围和子树的个数范围，B_的结构特性，从B_树上查找一个给定值元素的过程。

一般掌握的内容：

1. B_树查找算法。
2. 向B_树中插入元素的过程。

对本章的其余内容均作一般了解。

第九章　排序

重点掌握的内容：

1. 直接插入、直接选择和冒泡排序的方法，排序过程及时间复杂度。

2. **在堆排序中建立初始堆的过程和利用堆排序的过程，对一个分支结点进行筛运算**

的过程、算法及时间复杂度，整个堆排序的算法描述及时间复杂度。

3. 快速排序的方法，对一组数据的排序过程，对应的二叉搜索树，快速排序过程中划分的层数和递归排序区间的个数。

4. 递归排序的递归算法，它在平均情况下的时间和空间复杂度，在最坏情况下的时间和空间复杂度。

5. 二路归并排序的方法和对数据的排序过程，每趟排序前、后的有序表长度，二路归并排序的趟数、时间复杂度和空间复杂度。

6. **各种排序方法的不同数据序的比较、最好、最坏、平均情况。**

7. 哪些排序不受初始数据的影响。

一般掌握的内容：

1. 每一种排序方法的稳定性。
2. 直接插入排序和直接选择排序的算法。

一般了解的内容：

1. 二路归并排序过程中涉及的每个算法描述。
2. 冒泡排序算法。

第十章　文件

重点掌握的内容：

1. 文件的有关概念。
2. 文件的逻辑结构及其操作。
3. 索引文件的组织方式和特点。
4. 索引文件的的查询和更新操作的基本思想。
5. 两种最常用的索引顺序文件(ISAM 文件和 VSAM 文件) 的组织方式和特点。
6. 在 ISAM 文件和 VSAM 文件上查找和更新操作的基本思想。
7. 散列文件的组织方式和特点。
8. 散列文件的查询和更新操作的基本思想。
9. 多关键字文件和其它文件的差别。
10. 多重表文件和倒排文件组织方式和特点。
11. 多重表文件和倒排文件查询和更新操作的基本思想。

本章其它内容一般掌握

第二部分　模拟试卷

模拟试题（一）

一、单项选择题（每小题 2 分，共 20 分）

（1）以下数据结构中哪一个是线性结构？（　　）

A）有向图　　B）队列　　C）线索二叉树　　D）B 树

（2）在一个单链表 HL 中，若要在当前由指针 p 指向的结点后面插入一个由 q 指向的结点，则执行如下（　　）语句序列。

A）p=q; p->next=q;　　B）p->next=q; q->next=p;

C）p->next=q->next; p=q;　　D）q->next=p->next; p->next=q;

（3）（　　）不是队列的基本运算。

A）在队列第 i 个元素之后插入一个元素　B）从队头删除一个元素

C）判断一个队列是否为空　　D）读取队头元素的值

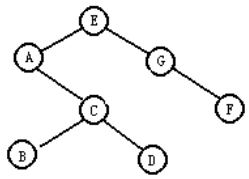
（4）字符 A、B、C 依次进入一个栈，按出栈的先后顺序组成不同的字符串，至多可以组成（　　）个不同的字符串。

A）14　　B）5　　C）6　　D）8

（5）由权值分别为 3,8,6,2 的叶子生成一棵哈夫曼树，它的带权路径长度为（　　）。

A）11　　B）35　　C）19　　D）53

以下 6-8 题基于下图：



（6）该二叉树结点的前序遍历的序列为（　　）。

A）E、G、F、A、C、D、B　　B）E、A、G、C、F、B、D

C）E、A、C、B、D、G、F　　D）E、G、A、C、D、F、B

（7）该二叉树结点的中序遍历的序列为（　　）。

A）A、B、C、D、E、G、F　　B）E、A、G、C、F、B、D

C）E、A、C、B、D、G、F　　D）B、D、C、A、F、G、E

（8）该二叉树的按层遍历的序列为（　　）。

A）E、G、F、A、C、D、B　　B）E、A、C、B、D、G、F

C）E、A、G、C、F、B、D　　D）E、G、A、C、D、F、B

（9）下面关于图的存储的叙述中正确的是（　　）。

A）用邻接表法存储图，占用的存储空间大小只与图中边数有关，而与顶点个数无关

B）用邻接表法存储图，占用的存储空间大小与图中边数和顶点个数都有关

C）用邻接矩阵法存储图，占用的存储空间大小与图中顶点个数和边数都有关

D）用邻接矩阵法存储图，占用的存储空间大小只与图中边数有关，而与顶点个数无关

（10）设有关键字序列('q', 'g', 'm', 'z', 'a', 'n', 'p', 'x', 'h')，下面哪一个序列是从上述序列出发建堆的结果?（　　）

A）'a', 'g', 'h', 'm', 'n', 'p', 'q', 'x', 'z　　B）'a', 'g', 'm', 'h', 'q', 'n', 'p', 'x', 'z'

C）'g', 'm', 'q', 'a', 'n', 'p', 'x', 'h', 'z'　　D）'h', 'g', 'm', 'p', 'a', 'n', 'q', 'x', 'z'

二、（每小题 4 分，共 8 分）

已知一个 6×5 稀疏矩阵如下所示，试写出它的三元组表表示。

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 \\ 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \end{bmatrix}$$

三、（本题 8 分）

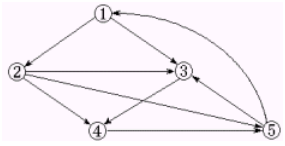
求网的最小生成树有哪些算法？它们的时间复杂度分别下多少，各适用何种情况？

四、（每小题 4 分，共 8 分）

对于如下图所示的有向图采用邻接表存储结构，并且每个顶点邻接表中的边结点都是按照终点序号从小到大的次序链接的，试写出：

（1）从顶点 v1 出发进行深度优先搜索所得到的顶点序列；

（2）从顶点 v2 出发进行广度优先搜索所得到的顶点序列。



五、（本题 8 分）

已知一个图的顶点集 V 和边集 E 分别为：

V={1,2,3,4,5,6,7};

E={<2,1>,<3,2>,<3,6>,<4,3>,<4,5>,<4,6>,<5,1>,<5,7>,<6,1>,<6,2>,<6,5>};

若采用邻接表存储结构，并且每个顶点邻接表中的边结点都是按照终点序号从小到大的次序链接的，试给出得到的拓扑排序的序列（入度为零的顶点可采用栈或队列来进行存储）。

六、（本题 8 分）

对于序列{8,18,6,16,29,28}，试写出堆顶元素最小的初始堆。

七、（本题 8 分）

一棵二叉树的先序、中序和后序序列分别如下，其中有一部分未显示出来。试求出空格处的内容。

先序序列：__B__F__ICEH__G

中序序列：D__KFIA__EJC__

后序序列：__K__FBHJ__G__A

八、（每小题 2 分，共 8 分）

设有序列：w={23,24,27,80,28}，试给出：

（1）二叉排序树；

（2）哈夫曼树；

（3）平衡二叉树；

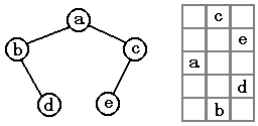
（4）对于增量 d=2 按降序执行一遍希尔排序的结果。

九、（本题 9 分）

有关键字序列{7,23,6,9,17,19,21,22,5}，Hash 函数为 H(key)=key % 5，采用链地址法处理冲突，试构造哈希表。

十、（本题 15 分）

假设二叉树中每个结点所含数据元素均为单字母，以二叉链表为存储结构，试编写算法按如下图所示的树状显示二叉树。



模拟试题（一）参考答案

一、单项选择题

（1）B　　（2）D　　（3）A　　（4）B　　（5）B

（6）C　　（7）A　　（8）C　　（9）B　　（10）B

二、（每小题 4 分，共 8 分）

三元组表表示如下：

$$\begin{bmatrix} 6 & 5 & 5 \\ 1 & 5 & 1 \\ 3 & 2 & -1 \\ 4 & 5 & -2 \\ 5 & 1 & 5 \\ 6 & 3 & 7 \end{bmatrix}$$

三、（本题 8 分）

求网的最小生成树可使用 Prim 算法，时间复杂度为 $O(n^2)$，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。

四、（每小题 4 分，共 8 分）

（1）DFS：v1 v2 v3 v4 v5

（2）BFS：v2 v3 v4 v5 v1

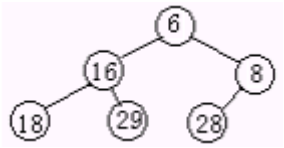
五、（本题 8 分）

用栈存储入度为零的顶点得到的拓扑排序为：　4　3　6　5　7　2　1

用队列存储入度为零的顶点得到的拓扑排序为：　4　3　6　2　5　1　7

六、（本题 8 分）

所构造的堆如下图所示：

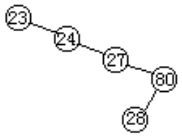


七、（本题 8 分）

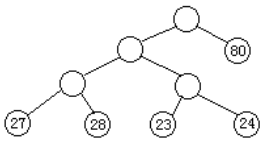
在先序序列空格中依次填 ADKJ，中序中依次填 BHG，后序中依次填 DIEC。

八、（每小题 2 分，共 8 分）

（1）二叉排序树如下图所示：



（2）哈夫曼树如下图所示：

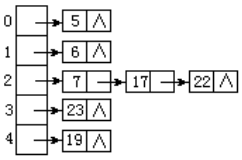


（3）平衡二叉树如下图所示：

（4）对于增量 d=2 按降序执行一遍希尔排序的结果：28,80,27,24,23

九、（本题 9 分）

哈希表如下图所示：



十、（本题 15 分）

从上图来看，二叉树的第一层显示在第一列，第二层显示在第二列，第三层显示在第三列；每行显示一个结点，从上至下是先显示右子树，再显示根，最后最左子树，也就是以先遍历右子树，最后遍历左子树的中序遍历次序显示各结点。

具体算法实现如下：

```
// 文件路径名:exam1\alg.h
ss ElemType>
void DisplayHelp(BinTreeNode<ElemType> *r, int level)
```

```
// 操作结果：按树状形式显示以 r 为根的二叉树，level 为层次数，可设根结点的层次数为 1
{
    if(r != NULL)
    {    // 空树不显式，只显式非空树
        DisplayHelp<ElemType>(r->rightChild, level + 1);    // 显示右子树
        cout << endl;                                       // 显示新行
        for(int i = 0; i < level - 1; i++)
            cout << " ";                                    // 确保在第 level 列显示结点
        cout << r->data;                                    // 显示结点
        DisplayHelp<ElemType>(r->leftChild, level + 1);     // 显示左子树
    }
}

template <class ElemType>
void Display(const BinaryTree<ElemType> &bt)
// 操作结果：树状形式显示二叉树
{
    DisplayHelp<ElemType>(bt.GetRoot(), 1);          // 树状显示以 bt.GetRoot()为根的二叉树
    cout << endl;                                    // 换行
}
```

模拟试题（二）

一、单项选择题（每小题 2 分，共 20 分）

（1）设 Huffman 树的叶子结点数为 m，则结点总数为（　　）。

A）2m　　B）2m-1

C）2m+1　　D）m+1

（2）若元素 a,b,c,d,e,f 依次入栈，允许入栈、出栈操作交替进行。但不允许连续三次进行出栈操作，则不可能得到的出栈序列是（　　）

A）dcebfa　　B）cbdaef　　C）dcebaf　　D）afedcb

（3）在一棵度为 4 的树 T 中，若有 20 个度为 4 的结点，10 个度为 3 的结点，1 个度为 2 的结点，10 个度为 1 的结点，则树 T 的叶节点个数是（　　）

A）41　　B）82　　C）113　　D）122

（4）设有一个二维数组 $A[m][n]$，假设 $A[0][0]$存放位置在 $600_{(10)}$，$A[3][3]$存放位置在 $678_{(10)}$，每个元素占一个空间，问 $A[2][3]_{(10)}$存放在什么位置？（脚注$_{(10)}$表示用 10 进制表示，m>3）（　　）。

A）658　　B）648　　C）633　　D）653

（5）下列关于二叉树遍历的叙述中，正确的是（　　）。

A）若一个叶子是某二叉树的中序遍历的最后一个结点，则它必是该二叉树的前序遍历最后一个结点

B）若一个结点是某二叉树的前序遍历最后一个结点，则它必是该二叉树的中序

遍历的最后一个结点

C）若一个结点是某二叉树的中序遍历的最后一个结点，则它必是该二叉树的前序最后一个结点

D）若一个树叶是某二叉树的前序最后一个结点，则它必是该二叉树的中序遍历最后一个结点

（6）k 层二叉树的结点总数最多为（　　）。

A）2^k-1　　B）2^{k+1}　　C）2K-1　　D）2^{k-1}

（7）对线性表进行二分法查找，其前提条件是（　　）。

A）线性表以链接方式存储，并且按关键字值排好序

B）线性表以顺序方式存储，并且按关键字值的检索频率排好序

C）线性表以顺序方式存储，并且按关键字值排好序

D）线性表以链接方式存储，并且按关键字值的检索频率排好序

（8）对 n 个记录进行堆排序，所需要的辅助存储空间为（　　）。

A）$O(1og_2n)$　　B）O(n)　　C）O(1)　　D）$O(n^2)$

（9）对于线性表（7，34，77，25，64，49，20，14）进行散列存储时，若选用 H（K）=K%7 作为散列函数，则散列函数值为为 0 的元素有（　　）个。

A）1　　B）2　　C）3　　D）4

（10）下列关于数据结构的叙述中，正确的是（　　）。

A）数组是不同类型值的集合

B）递归算法的程序结构比迭代算法的程序结构更为精炼

C）树是一种线性结构

D）用一维数组存储一棵完全二叉树是有效的存储方法

二、（本题 8 分）

有 5 个元素，其入栈次序为：A、B、C、D、E，在各种可能的出栈次序中，以元素 C 第一个出栈，D 第二个出栈的次序有哪几个？

三、（每小题 4 分，共 8 分）

已知一个无向图的顶点集为{a, b, c, d, e}，其邻接矩阵如下所示：

$$\begin{bmatrix} 01001 \\ 10010 \\ 00011 \\ 01101 \\ 10110 \end{bmatrix}$$

（1）画出该图的图形；

（2）根据邻接矩阵从顶点 a 出发进行深度优先遍历和广度优先遍历，写出相应的遍历序列。

四、（本题 8 分）

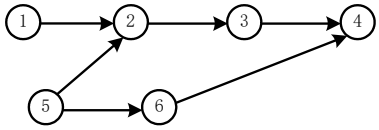
树有哪些遍历方法？它们分别对应于把树转变为二叉树的哪些遍历方法？

五、（本题 8 分）

将关键字序列（7、8、11、18、9、14, 30）散列存储到散列列表中，散列表的存储空间是一个下标从 0 开始的一个一维数组，散列函数为：H(key)=（key * 3）% m（m 为表长），处理冲突采用线性探测再散列法，要求装填（载）因子为 0.7，请画出所构造的散列表；计算查找成功的平均查找长度。

六、（本题 8 分）

试列出如下图中全部可能的拓扑排序序列。



七、（本题 8 分）

请说明对一棵二叉树进行按照先遍左子树，后右子树的方式进行前序、中序和后序遍历，其叶结点的相对次序是否会发生改变？为什么？

八、（本题 8 分）

设有一个输入数据的序列是 **{** 46, 25, 78, 62, 12, 80 **}**，试画出从空树起，逐个输入各个数据而生成的二叉排序树。

九、（本题 9 分）

试画出表达式(a+b/c)*(d-e*f)的二叉树表示，并写出此表达式的波兰式表示，中缀表示及逆波兰式表示。

十、（本题 15 分）

以二叉链表作存储结构，试编写计算二叉树中叶子结点数目的递归算法。

模拟试题（二）参考答案

一、单项选择题（每小题 2 分，共 20 分）

（1）B　（2）D　（3）B　（4）D　（5）A

（6）A　（7）C　（8）C　（9）D　（10）D

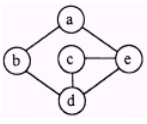
二、（本题 8 分）

按照栈的特点可知以元素 C 第一个出栈，D 第二个出栈的次序有 CDEBA 、CDBAE 和 CDBEA 3 种。

三、（每小题 4 分，共 8 分）

【解答】

（1）该图的图形如下图所示：



（2）深度优先遍历序列为：abdce；广度优先遍历序列为：abedc。

四、（本题 8 分）

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。

五、（本题 8 分）

由装载因子 0.7，由于有 7 个数据元素，可得 7/m=0.7，从而 m=10，所以，构造的散列表为：

下标	0	1	2	3	4	5	6	7	8	9
关键字	30	7	14	11	8	18	.	9	.	
比较次数	1	1	1	1	1	2		1		

H（7）=（7*3）%　10=1

H（8）=（8*3）% 10=4

H（11）=（11*3）% 10=3

H（18）=（18*3）% 10=4

H（9）=（9*3）% 10=7

H（14）=（14*3）% 10=2

H（30）=（30*3）% 10=2

（2）查找成功的 ASL=(1+1+1+1+1+2+1)/7=8/7

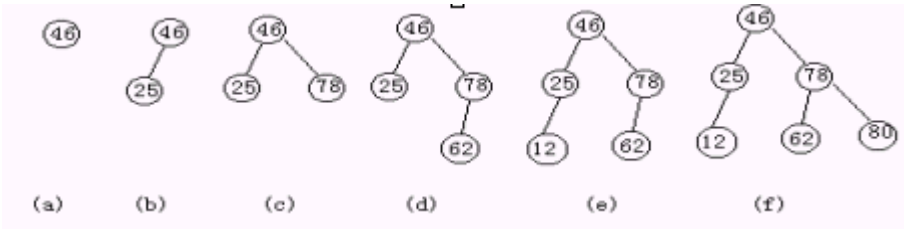
六、（本题 8 分）

全部可能的拓扑排序序列为：1523634、152634、156234、561234、516234、512634、512364

七、（本题 8 分）

二叉树任两个中叶结点必在某结点的左/右子树中，三种遍历方法对左右子树的遍历都是按左子树在前、右子树在后的顺序进行遍历的。所以在三种遍历序列中叶结点的相对次序是不会发生改变的。

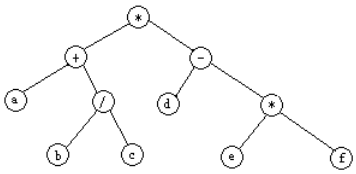
八、（本题 8 分）



九、（本题 9 分）

表达式的波兰式表示，中缀表示及逆波兰式表示分别是此表达式的二叉树表示的前序遍历、中序遍历及后序遍历序列。

二叉树表示如下图所示：



波兰式表示：*+a/bc-d*ef
中缀表示：a+b/c*d-e*f
逆波兰式表示：abc/+def*-*

十、（本题 15 分）

本题只要在遍历二叉树的过程序中对叶子结点进行记数即可。
具体算法实现如下：

```
// 文件路径名:exam2\alg.h
template <class ElemType>
long LeafCountHelp(BinTreeNode<ElemType> *r)
// 操作结果：按树状形式显示二叉树，level 为层次数，可设根结点的层次数为 1
{
    if (r == NULL)
    {   // 空二叉树
        return 0;                       // 空树返回 0
    }
    else if (r->leftChild == NULL && r->rightChild == NULL)
    {   // 只有一个结点的树
        return 1;                       // 只有一个结点的树返回 1
    }
    else
    {   // 其他情况，叶子结点数为左右子树的叶子结点数之和
        return LeafCountHelp(r->leftChild) + LeafCountHelp(r->rightChild);
    }
}

template <class ElemType>
long LeafCount(const BinaryTree<ElemType> &bt)
// 操作结果：计算二叉树中叶子结点数目
{
    return LeafCountHelp(bt.GetRoot());     // 调用辅助函数实现计算二叉树中叶子结点数目
}
```

模拟试题（三）

一、单项选择题（每小题 2 分，共 20 分）

（1）对一组数据（2，12，16，88，5，10）进行排序，若前三趟排序结果如下

第一趟：2，12，16，5，10，88

第二趟：2，12，5，10，16，88

第三趟：2，5，10，12，16，88

则采用的排序方法可能是（　　）。

A）起泡排序　　B）希尔排序　　C）归并排序　　D）基数排序

（2)在带有头结点的单链表HL中,要向表头插入一个由指针p指向的结点,则执行(　　)。

A）p->next=HL->next; HL->next=p　　B）p->next=HL; HL=p

C）p->next=HL; p=HL　　D）HL=p; p->next=HL

（3）对线性表，在下列哪种情况下应当采用链表表示？（　　）

A）经常需要随机地存取元素　　B）经常需要进行插入和删除操作

C）表中元素需要占据一片连续的存储空间D）表中元素的个数不变

（4）一个栈的输入序列为1 2 3，则下列序列中不可能是栈的输出序列的是(　　)。

A）2 3 1　　B）3 2 1　　C）3 1 2　　D）1 2 3

（5）每一趟都能选出一个元素放在其最终位置上，并且不稳定的排序算法是（　　）。

A）冒泡排序　　B）简单选择排序 C）希尔排序　　D）直接插入排序

（6）采用开放定址法处理散列表的冲突时，其平均查找长度（　　）。

A）低于链接法处理冲突　　B）高于链接法处理冲突

C）与链接法处理冲突相同　　D）高于二分查找

（7）若需要利用形参直接访问实参时，应将形参变量说明为（　　）参数。

A）值　　B）函数　　C）指针　　D）引用

（8）为解决计算机与打印机之间速度不匹配的问题，通常设置一个打印数据缓冲区，主机将要输出的数据依次写入该缓冲区，而打印机则依次从该缓冲区中取出数据。该缓冲区的逻辑结构应该是（　　）。

A）栈　　B）队列　　C）树　　D）图

在稀疏矩阵的带行指针向量的链接存储中，每个单链表中的结点都具有相同的（　　）。

A）行号　　B）列号　　C）元素值　　D）非零元素个数

（9）快速排序在最坏情况下的时间复杂度为（　　）。

A）$O(\log_2 n)$　　B）$O(n\log_2 n)$　　C）$O(n)$　　D）$O(n^2)$

（10）从二叉搜索树中查找一个元素时，其时间复杂度大致为（　　）。

A）$O(n)$　　B）$O(1)$　　C）$O(\log_2 n)$　　D）$O(n^2)$

二、（本题 8 分）

已知一个图的顶点集 V 和边集 E 分别为：

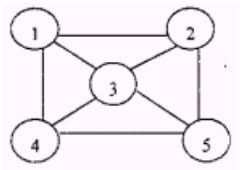
V={1,2,3,4,5,6,7};

E={(1,2)3,(1,3)5,(1,4)8,(2,5)10,(2,3)6,(3,4)15,(3,5)12,(3,6)9,(4,6)4,(4,7)20,(5,6)18,(6,7)25};

用克鲁斯卡尔算法得到最小生成树，试写出在最小生成树中依次得到的各条边。

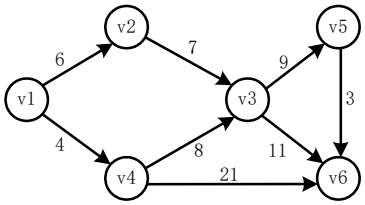
三、（本题 8 分）

请画出如下图所示的邻接矩阵和邻接表。



四、（每小题 4 分，共 8 分）

设有如下图所示的 AOE 网（其中 vi（i=l，2，…，6）表示事件，有向边上的权值表示活动的天数）。



（1）找出所有的关键路径。

（2）v3 事件的最早开始时间是多少。

五、（本题 8 分）

一棵非空的有向树中恰有一个顶点入度为 0，其他顶点入度为 1。但一个恰有一个顶点入度为 0、其他顶点入度为 1 的有向图却不一定是一棵有向树。请举例说明之。

六、（本题 8 分）

假设把 n 个元素的序列（$a_1,a_2,\ldots a_n$）满足条件 $a_k<\max\{a_t|1\leqslant t\leqslant k\}$的元素 a_k 称为“逆序元素”。若在一个无序序列中有一对元素 $a_i>a_j(i<j)$，试问，当 a_i 与 a_j 相互交换后，该序列中逆序元素的个数一定不会增加，这句话对不对？如果对，请说明为什么？如果不对，请举一例说明。

七、（本题 8 分）

带权图（权值非负，表示边连接的两顶点间的距离）的最短路径问题是找出从初始顶点到目标顶点之间的一条最短路径。假定从初始顶点到目标顶点之间存在路径，现有一种解决该问题的方法：

①设最短路径初始时仅包含初始顶点，令当前顶点 u 为初始顶点；

②选择离 u 最近且尚未在最短路径中的一个顶点 v，加入到最短路径中，修改当前顶点 u=v；

③重复步骤②，直到 u 是目标顶点时为止。

请问上述方法能否求得最短路径？若该方法可行，请证明之；否则，请举例说明。

八、（本题 8 分）

已知一组关键字为（19,14,23,1,68,20,84,27,55,11,10,79），哈希函数：H(key)=key MOD 13，哈希地址空间为 0~12，请构造用链地址法处理冲突的哈希表，并求平均查找长度。

九、（本题 9 分）

已知关键字序列{23，13，5，28，14，25}，试构造二叉排序树。

十、（本题 15 分）

编写一个算法求二叉树的深度。

模拟试题（三）参考答案

一、单项选择题（每小题 2 分，共 20 分）

（1）A　（2）A　（3）B　（4）C　（5）B

（6）B　（7）D　（8）B　（9）D　（10）C

二、（本题 8 分）

用克鲁斯卡尔算法得到的最小生成树为：

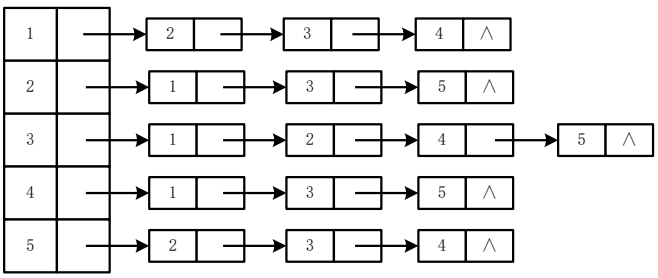
(1,2)3,　(4,6)4,　(1,3)5,　(1,4)8,　(2,5)10,　(4,7)20

三、（本题 8 分）

邻接矩阵：

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

邻接表如下图所示：



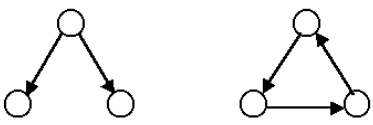
四、（每小题 4 分，共 8 分）

（1）找出所有的关键路径有：v1→v2→v3→v5→v6，以及 v1→v4→v6。

（2）v3 事件的最早开始时间是 13。

五、（本题 8 分）

如下图所示的有向图，只有一个顶点的入度为 0 外，其他每个顶点的入度都为 1，因为非连通，所以此图却不是有向树。

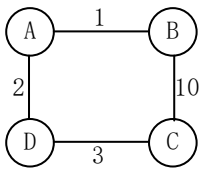


六、（本题 8 分）

不对，例如序列｛3、3、4、2、1｝的“逆序元素”个数是 2，2 和 1 是“逆序元素”；但是将第二个 3 和 2 交换后，成为｛3、2、4、3、1｝，此时“逆序元素”个数是 3，2、3 和 1 是“逆序元素”。然而交换后一定减少的是“逆序对”的个数，例如上例中｛3、3、4、2、1｝的逆序对的个数是 7，交换第二个 3 和 2 后，｛3、2、4、3、1｝的逆序对的个数是 6。

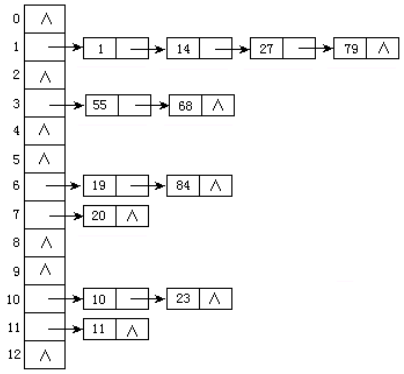
七、（每小题 4 分，共 8 分）

该方法求得的路径不一定是最短路径。例如，对于下图所示的带权图，如果按照题中的原则，从 A 到 C 的最短路径为 A→B→C，事实上其最短路径为 A→D→C。



八、（本题 8 分）

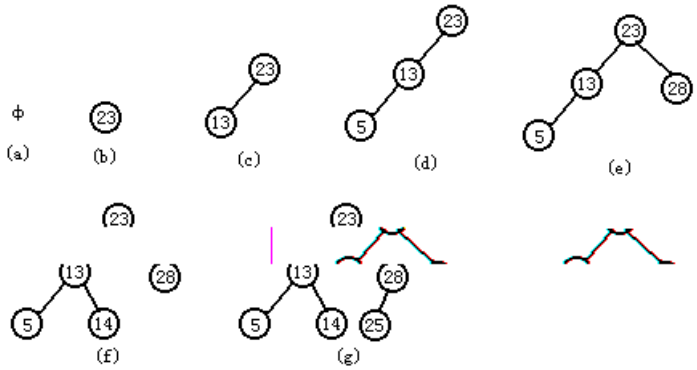
用链地址法处理冲突的哈希表如下图所示：



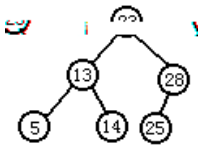
ASL=$\frac{1}{12}$ (1*6+2*4+3*1+4*1)=1.75

九、（本题 9 分）

构造二叉排序树的过程如下图所示。



构造的二叉排序树如下图所示：



十、（本题 15 分）

若二叉树为空，深度为 0；若二叉树不空，则二叉树的深度为左右子树深度的最大值加 1。本题最简单算法是递归算法。

具体算法实现如下：

```
// 文件路径名:exam3\alg.h
template <class ElemType>
int DepthHelp(BinTreeNode<ElemType> *r)
// 操作结果：求二叉树的深度
{
    if (r == NULL)
    {   // 空二叉树
        return 0;                                   // 空二叉树的深度为 0
    }
    else
    {   // 非空二叉树
        int lDepth = DepthHelp(r->leftChild);       // 左子树的深度
        int rDepth = DepthHelp(r->rightChild);      // 右子树的深度
        return ((lDepth > rDepth) ? lDepth : rDepth) + 1; // 返回左右子树的深度最大值加 1
    }
}

template <class ElemType>
int Depth(BinaryTree<ElemType> &bt)
// 操作结果：求二叉树的深度
{
    return DepthHelp(bt.GetRoot());                 // 调用辅助函数求二叉树的深度
}
```

模拟试题（四）

一、单项选择题（每小题 2 分，共 20 分）

（1）以下数据结构中哪一个是线性结构？（　　）

A）有向图　　B）栈　　C）二叉树　　D）B 树

（2）若某链表最常用的操作是在最后一个结点之后插入一个结点和删除最后一个结点，则采用（　　）存储方式最节省时间。

A）单链表　　B）双链表　C）带头结点的双循环链表　D）单循环链表

（3）（　　）不是队列的基本运算。

A）在队列第 i 个元素之后插入一个元素　B）从队头删除一个元素

C）判断一个队列是否为空　　D）读取队头元素的值

（4）字符 A、B、C、D 依次进入一个栈，按出栈的先后顺序组成不同的字符串，至多可以组成（　　）个不同的字符串？

A）15　　B）14　　C）16　　D）21

（5）由权值分别为 4,7,6,2 的叶子生成一棵哈夫曼树，它的带权路径长度为（　　）。

A）11　　B）37　　C）19　　D）53

以下 6-8 题基于下面的叙述：若某二叉树结点的中序遍历的序列为 A、B、C、D、E、F、G，后序遍历的序列为 B、D、C、A、F、G、E。

（6）则该二叉树结点的前序遍历的序列为（　　）。

A）E、G、F、A、C、D、B　　B）E、A、G、C、F、B、D

C）E、A、C、B、D、G、F　　D）E、G、A、C、D、F、B

（7）该二叉树有（　　）个叶子。

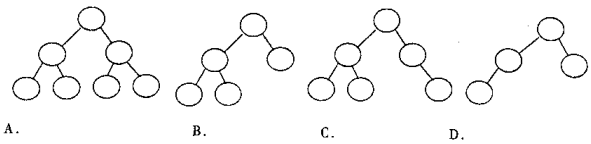
A）3　　B）2　　C）5　　D）4

（8）该二叉树的按层遍历的序列为（　　）。

A）E、G、F、A、C、D、B　　B）E、A、C、B、D、G、F

C）E、A、G、C、F、B、D　　D）E、G、A、C、D、F、B

（9）下面的二叉树中，（　　）不是完全二叉树。



（10）设有关键字序列('q', 'g', 'm', 'z', 'a')，（　　）序列是从上述序列出发建的小根堆的结果。

A）'a', 'g' , 'm', 'q', 'z'　　B）'a', 'g', 'm', 'z', 'q'

C）'g', 'm', 'q', 'a', 'z'　　D）'g', 'm', 'a', 'q', 'z'

二、（本题 8 分）

试述顺序查找法、折半查找法和分块查找法对被查找的表中元素的要求，对长度为 n 的查找表来说，三种查找法在查找成功时的查找长度各是多少？

三、（本题 8 分）

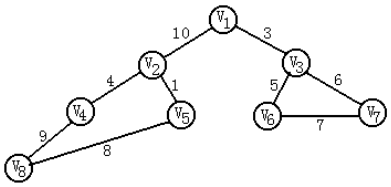
设有一个输入数据的序列是{ 46, 25, 78, 62, 12, 80 }，试画出从空树起，逐个输入各个数据而生成的二叉排序树。

四、（本题 8 分）

给定一个关键字序列｛24，19，32，43，38，6，13，22｝，请写出快速排序第一趟的结果；堆排序时所建的初始堆。

五、（本题 8 分）

设有带权无向网 Net 如下图所示。



试给出：

（1）Net 的邻接矩阵表示；

（2）从 V_1 开始的深度优先遍历；

（3）从 V_1 开始的广度优先遍历；

（4）从 V_1 开始执行的普里姆（Prim）算法过程中所选边的序列。

六、（本题 8 分）

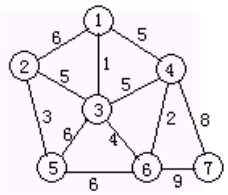
用一维数组存放一棵完全二叉树：ABCDEFGHIJKL。请写出后序遍历该二叉树的访问结点序列。

七、（本题 8 分）

已知哈希表地址空间为 0..8，哈希函数为 H(key)=key%7，采用线性探测再散列处理冲突，将数据序列{100,20,21,35,3,78,99,45}依次存入此哈希表中，列出插入时的比较次数，并求出在等概率下的平均查找长度。

八、（本题 8 分）

对于如下图所示的 G，用 Kruskal 算法构造最小生成树，要求图示出每一步的变化情况。



九、（本题 9 分）

已知一棵二叉树的先序序列与中序序列分别如下，试画出此二叉树。

先序序列：ABCDEFGHIJ

中序序列：CBEDAGHFJI

十、（本题 15 分）

试写一递归算法，从大到小输出二叉排序树中所有的关键字值小于 key 的元素值。

。

模拟试题（四）参考答案

一、单项选择题（每小题 2 分，共 20 分）

（1）B　　（2）C　　（3）A　　（4）B　　（5）B

（6）C　　（7）A　　（8）C　　（9）C　　（10）B

二、（本题 8 分）

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：

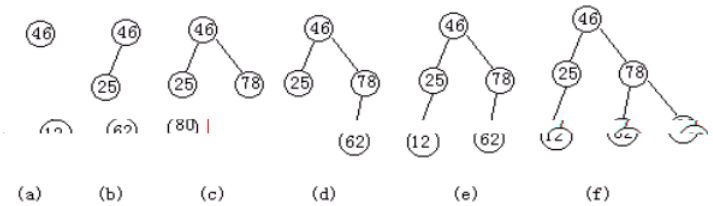
顺序查找法：查找成功的平均查找长度为$\frac{n+1}{2}$；

折半查找法：查找成功的平均查找长度为 $\log_2(n+1)+1$；

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$\log_2(\frac{n}{s}+1)+\frac{s}{2}$。

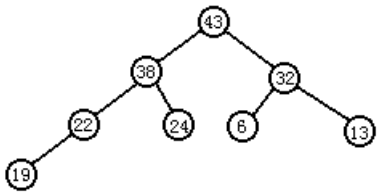
三、（本题 8 分）

如下图所示：



四、（本题 8 分）

快速排序的第一趟结果为｛22,19,13,6,24,38,43,12｝；堆排序时所建立的初始大顶堆如所图所示：



五、（本题 8 分）

（1）Net 的邻接矩阵表示：

$$\begin{bmatrix} \infty & 10 & 3 & \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & \infty & 4 & 1 & \infty & \infty & \infty \\ 3 & \infty & \infty & \infty & \infty & 5 & 6 & \infty \\ \infty & 4 & \infty & \infty & \infty & \infty & \infty & 9 \\ \infty & 1 & \infty & \infty & \infty & \infty & \infty & 8 \\ \infty & \infty & 5 & \infty & \infty & \infty & 7 & \infty \\ \infty & \infty & 6 & \infty & \infty & 7 & \infty & \infty \\ \infty & \infty & \infty & 9 & 8 & \infty & \infty & \infty \end{bmatrix}$$

（2）从 V_1 开始的深度优先遍历：V1　V2　V4　V8　V5　V3　V6　V7

（3）从 V_1 开始的广度优先遍历：V1　V2　V3　V4　V5　V6　V7　V8

（4）从 V_1 开始执行的普里姆（Prim）算法过程中所选边的序列：

(V1,V3)，(V3,V6)，(V3,V7)，(V1,V2)，(V2,V5)，(V2,V4)，(V5,V8)

六、（本题 8 分）

先画出该二叉树的树形结构。对其进行后序遍历得到后序序列为：HIDJKEBLFGCA。

七、（本题 8 分）

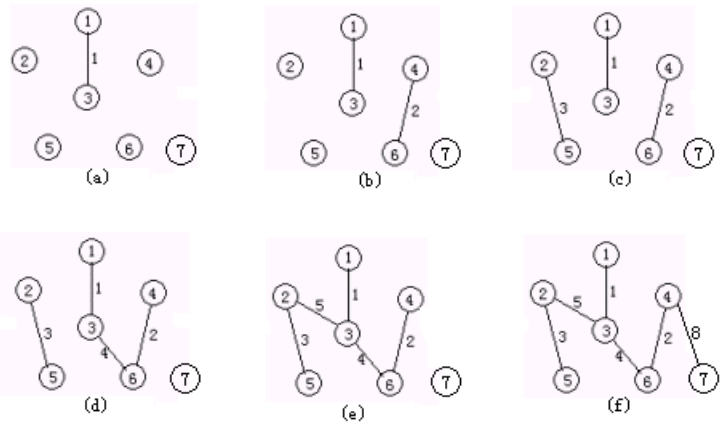
哈希表及查找各关键字要比较的次数如下图所示：

哈希地址	0	1	2	3	4	5	6	7	8
关键字	21	35	100	3	78	99	20	25	
比较次数	1	2	1	1	4	5	1	5	

$ASL=\frac{1}{8}(4\times1+1\times2+1\times4+2\times5)=2.5$

八、（本题 8 分）

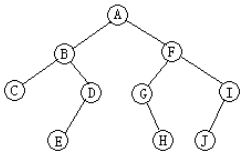
用 Kruskal 算法构造最小生成树的过程如下图所示：



九、（本题 9 分）

先由先序序列的第一个结点确定二叉树的根结点，再由根结点在中序序列中左侧部分为左子树结点，在右侧部分为右子树结点，再由先序序列的第一个结点确定根结点的左右

孩子结点，由类似的方法可确定其他结点，如下图所示。



十、（本题 15 分）

可按先遍历右子树，遍历根结点，再遍历左子树进行中序遍历，这样可实现由大到小遍历一棵二叉排序树。

具体算法实现如下：

```
// 文件路径名:exam4\alg.h
#include "binary_sort_tree.h"                          // 二叉排序树类

template <class ElemType, class KeyType>
void InOrderHelp(BinTreeNode<ElemType> *r, const KeyType &key)
// 操作结果: 从大到小输出以 r 为根的二叉排序树中所有的关键字值小于 key 的元素值
{
    if (r != NULL)
    {   // 非空二叉排序树
        InOrderHelp(r->rightChild, key);               // 遍历右子树
        if(r->data < key) cout << r->data << "  ";     // 输出根结点
        else return;
        InOrderHelp(r->leftChild, key);                // 遍历左子树
    }
}

template <class ElemType, class KeyType>
void InOrder(const BinarySortTree<ElemType, KeyType> &t, const KeyType &key)
// 操作结果: 从大到小输出二叉排序树中所有的关键字值不小于 key 的元素值
{
    InOrderHelp(t.GetRoot(), key);
        // 调用辅助函数实现从大到小输出二叉排序树中所有的关键字值不小于 key 的元素值
}
```

模拟试题（五）

一、单项选择题（每小题 2 分，共 20 分）

（1）队列的特点是（　　）。

A）先进后出　　B）先进先出

C）任意位置进出　　D）前面都不正确

（2）有 n 个记录的文件，如关键字位数为 d，基数为 r，则基数排序共要进行（　　）遍分配与收集。

A）n　　B）d　　C）r　　D）n - d

（3）在二叉树结点的先序序列、中序序列和后序序列中，所有叶子结点的先后顺序（　　）。

A）都不相同　　B）完全相同

C）先序和中序相同，而与后序不同　　D）中序和后序相同，而与先序不同

（4）限定在一端加入和删除元素的线性表称为（　　）。

A）双向链表　　B）单向链表　　C）栈　　D）队列

（5）若数据元素序列 11，12，13，7，8，9，23，4，5 是采用下列排序方法之一得到的第二趟排序后的结果，则该排序算法只能是（　　）。

A）起泡排序　　B）插入排序　　C）选择排序　　D）二路归并排序

（6）设森林 F 对应的二叉树为 B，它有 m 个结点，B 的根为 p，p 的右子树上的结点个数为 n，森林 F 中第一棵树的结点个数是（　　）。

A）m-n-1　　B）n+1　　C）m-n+1　　D）m-n

（7）对于具有 n 个顶点的强连有向图，其弧条数的最小值为（　　）。

A）n+1　　B）n　　C）n-1　　D）n-2

（8）下面关于广义表的叙述中，不正确的是（　　）。

A）广义表可以是一个多层次的结构　　B）广义表至少有一个元素

C）广义表可以被其他广义表所共享　　D）广义表可以是一个递归表

（9）设二叉树根结点的层次为 0，一棵深度（高度）为 k 的满二叉树和同样深度完全二叉树各有 f 个结点和 c 个结点，下列关系式不正确的是（　　）。

A）f>=c　　B）c>f　　C）$f=2^{k+1}-1$　　D）$c>2^{k}-1$

（10）设一棵二叉树中没有度为 1 的结点，已知叶子结点数为 n，此树的结点数为（　　）。

A）2n+2　　B）2n+1　　C）2n　　D）2n-1

二、（每小题 4 分，共 8 分）

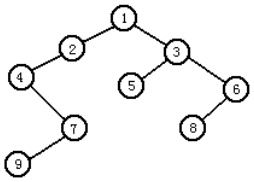
写出下列中缀表达式的后缀形式：

（1）3X/(Y-2)+1

（2）2+X*(Y+3)

三、（每小题 4 分，共 8 分）

试对如下图中的二叉树画出其：



（1）顺序存储表示；

（2）二叉链表存储表示的示意图。

四、（每小题 4 分，共 8 分）

判断以下序列是否是小根堆？如果不是，将它调整为小根堆。

（1）{ 12, 70, 33, 65, 24, 56, 48, 92, 86, 33 }

（2）{ 05, 23, 20, 28, 40, 38, 29, 61, 35, 76, 47, 100 }

五、（本题 8 分）

已知一个图的顶点集 V 和边集 E 分别为：

V={1,2,3,4,5,6,7};

E={(1,2)3,(1,3)5,(1,4)8,(2,5)10,(2,3)6,(3,4)15,(3,5)12,(3,6)9,(4,6)4,(4,7)20,(5,6)18,(6,7)25};

按照普里姆算法从顶点 1 出发得到最小生成树，试写出在最小生成树中依次得到的各条边。

六、（每小题 2 分，共 8 分）

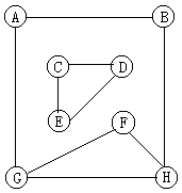
设有 12 个数据 25,40,33,47,12,66,72,87,94,22,5,58，它们存储在散列表中，利用线性探测再散列处理冲突，取散列函数为 H(key)=key % 13。

（1）顺次将各个数据散列到表中，并同时列出各元素的比较次数。

（2）计算查找成功的平均查找次数。

七、（第 1 小题 2 分，第 2、3 小题每小题 3 分，本题 8 分）

对于如下图所示的图 G，邻接点按从小到大的次序。



（1）图 G 有几个连通分量？

（2）按深度优先搜索所得的树是什么？

（3）按深度优先搜索所得的顶点序列是什么？

八、（本题 8 分）

已知一棵树边为：

{<I,M>,<I,N>,<E,I>,<B,E>,<B,D>,<C,B>,<G,L>,<G,K>,<A,G>,<A,F>,<A,H>,<C,A>}

试画出这棵树，并回答下列问题：

（1）哪个是根结点？

（2）哪些是叶子结点？

（3）树的深度是多少？

九、（本题 9 分）

给出一组关键字 T=(12,2,16,30,8,28,4,10,20,6,18)。写出用下列算法从小到大排序时第一趟结束时的序列。

（1）希尔排序（第一趟排序的增量为 5）

（2）快速排序（选第一个记录为枢轴）

十、（本题 15 分）

编写复制一棵二叉树的非递归算法。

模拟试题（五）参考答案

一、单项选择题（每小题 2 分，共 20 分）

（1）B　　（2）B　　（3）B　　（4）C　　（5）B

（6）D　　（7）B　　（8）B　　（9）B　　（10）D

二、（每小题 4 分，共 8 分）

（1）3　X　*Y　2　-/　1　+

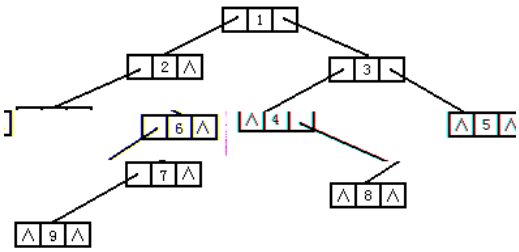
（2）2　X　Y　3　+　*　+

三、（每小题 4 分，共 8 分）

（1）二叉树的顺序存储表示如下所示：

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	1	2	3	4		5	6		7				8					9

（2）二叉树的二叉链表存储表示的示意图如下图所示：



四、（每小题 4 分，共 8 分）

（1）不是小根堆。调整为：{12,24,33,65,33,56,48,92,86,70}

（2）是小根堆。

五、（本题 8 分）

普里姆算法从顶点 1 出发得到最小生成树为：

(1,2)3,　(1,3)5,　(1,4)8,　(4,6)4,　(2,5)10,　(4,7)20

六、（每小题 2 分，共 8 分）

（1）取散列函数为 H(key)=key % 13。

（2）顺次将各个数据散列到表中，并同时列出各元素的比较次数如下表所示。

表10-1　各元素的比较次数

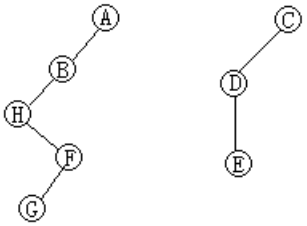
地址	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
关键字		40	66	94		5	58	33	47	72	87	22	25	12	
比较		1	2	1		1	1	1	1	3	2	3	1	2	

（4）计算查找成功的平均查找次数=（1×7+2×3+3×2）/12=19/12。

七、（第 1 小题 2 分，第 2、3 小题每小题 3 分，本题 8 分）

（1）图 G 有 2 个连通分量。

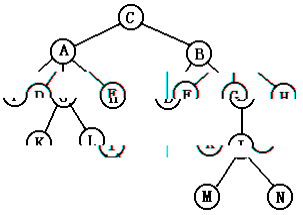
（2）按深度优先搜索所得的树如下图所示：



（3）按深度优先搜索所得的顶点序列：ABHFGCDE

八、（本题 8 分）

（1）树，如下图所示：



（2）C 是根结点。

（3）F，K，L，H，D，M，N 是叶子结点。

（3）深度是 5。

九、（本题 9 分）

（1）（12,2,10,20,6,18,4,16,30,8,28）

（2）（6,2,10,4,8,12,28,30,20,16,18）

十、（本题 15 分）

将算法实现函数声明为二叉树类的友元函数，可采用层次遍历的方式进行复制，将已复制的结点进入一个队列中即可。

具体算法实现如下：

```
// 文件路径名:exam5\alg.h
template <class ElemType>
void CopyBitree(BinaryTree<ElemType> *fromBtPtr, BinaryTree<ElemType> *&toBtPtr)
// 操作结果: 复制二叉树 fromBt 到 toBt 的非递归算法
{
    if (toBtPtr != NULL) delete toBtPtr;                          // 释放 toBtPtr
    if (fromBtPtr->Empty())
    {   // 空二叉树
        toBtPtr = NULL;                                           // 空二叉树
    }
    else
    {   // 非空二叉树
        LinkQueue<BinTreeNode<ElemType> *> fromQ, toQ;            // 队列
        BinTreeNode<ElemType> *fromPtr, *toPtr, *fromRoot, *toRoot;
        fromRoot = fromBtPtr->GetRoot();                          // 取出 fromBtPtr 的根
        toRoot = new BinTreeNode<ElemType>(fromRoot->data);       // 复制根结点
        fromQ.InQueue(fromRoot); toQ.InQueue(toRoot);             // 入队
        while (!fromQ.Empty())
        {   // fromQ 非空
            fromQ.OutQueue(fromPtr);                              // 出队
            toQ.OutQueue(toPtr);                                  // 出队
            if (fromPtr->leftChild != NULL)
            {   // 左子树非空
                toPtr->leftChild = new BinTreeNode<ElemType>(fromPtr->leftChild->data);
                    // 复制 fromPtr 左孩子
                fromQ.InQueue(fromPtr->leftChild); toQ.InQueue(toPtr->leftChild);      // 入队
            }
            if (fromPtr->rightChild != NULL)
            {   // 右子树非空
                toPtr->rightChild = new BinTreeNode<ElemType>(fromPtr->rightChild->data);
                    // 复制 fromPtr 左孩子
                fromQ.InQueue(fromPtr->rightChild); toQ.InQueue(toPtr->rightChild);   // 入队
            }
        }
        toBtPtr = new BinaryTree<ElemType>(toRoot);               // 生成 toBtPtr
    }
}
```

模拟试题（六）

一、单项选择题（每小题 2 分，共 20 分）

（1）设二叉树中有 n_2 个度为 2 的结点，n_1 个度为 1 的结点，n_0 个叶子结点，则此二

叉树采用二叉链表存储时空指针域个数为（　　）。

A）$n_0+n_1+n_2$　　B）$n_2+n_1+2n_0$　　C）$2n_2+n_1$　　D）$2n_0+n_1$

（2）若需在 O(nlogn)的时间内完成对数组的排序，且要求排序是稳定的，则可选择（　　）。

A）快速排序　　B）堆排序　　C）归并排序　　D）直接插入排序

（3）对于有 n 个顶点的有向图，由弗洛伊德（Floyd）算法求每一对顶之间的最短路径的时间复杂度是（　　）。

A）O(1)　　B）O(n)　　C）O(n)　　D）$O(n^3)$

（4）对 n 个元素的序列进行并归排序，所需要的辅助存储空间为（　　）。

A）O（1）　　B）O（$\log_2 n$）　　C）O（n）　　D）O（n^2）

（5）哈夫曼树中一定不存在（　　）。

A）度为 0 的结点　B）度为 1 的结点　　C）度为 2 的结点　D）带权的结点

（6）设 D={A,B,C,D}，R={<C,A>,<A,B>,<B,D>,<D,C>,<D,A>}，则数据结构(D,{R})是（　　）。

A）树　　B）图　　B）线性表　　D）前面都正确

（7）（　　）关键字序列不符合堆的定义。

A）'A'、'C'、'D'、'G'、'H'、'M'、'P'、'Q'、'R'、'X'

B）'A'、'C'、'M'、'D'、'H'、'P'、'X'、'G'、'Q'、'R'

C）'A'、'D'、'P'、'R'、'C'、'Q'、'X'、'M'、'H'、'G'

D）'A'、'D'、'C'、'M'、'P'、'G'、'H'、'X'、'R'、'Q'

（8）一组记录的排序码为(48,24,18,53,16,26,40)，采用冒泡排序法进行排序，则第一趟排序需要进行记录交换的次数是（　　）。

A）3　　B）4　　C）5　　D）6

（9）下面（　　）可以判断出一个有向图中是否有环（回路）？

A）求关键路径　　B）拓扑排序

C）求最短路径　　D）前面都不正确

（10）对线性表进行二分法查找，其前提条件是（　　）。

A）线性表以顺序方式存储，并且按关键字值排好序

B）线性表以顺序方式存储，并且按关键字值的检索频率排好序

C）线性表以链接方式存储，并且按关键字值排好序

D）线性表以链接方式存储，并且按关键字值的检索频率排好序

二、（本题 8 分）

在如下表所示的数组 A 中链接存储了一个线性表，表头指针存放在 A[0].next，试写出该线性表。

表10-2　线性表

A	0	1	2	3	4	5	6	7
data		60	50	78	90	34		40

next	4	0	5	2	7	1		3

三、（本题 8 分）

已知一棵二叉树的前序遍历的结果是 ABKCDFGHIJ，中序遍历的结果是 KBCDAFHIGJ，试画出这棵二叉树。

四、（本题 8 分）

已知一个图的顶点集 V 为： V={1,2,3,4,5,6,7}，弧如下表所示。

表10-3 图的弧集

起点	1	2	2	5	5	2	2	6	1	3
终点	6	4	5	4	7	6	7	7	7	5
权	1	1	2	2	2	3	3	4	5	7

试用克鲁斯卡尔算法依次求出该图的最小生成树中所得到的各条边及权值。

五、（本题 8 分）

向最小根堆中依次插入数据 4, 2, 5, 8, 3, 6, 10, 1 时，画出每插入一个数据后堆的变化。

六、（本题 8 分）

有二叉树中序序列为：ABCEFGHD；后序序列为：ABFHGEDC；请画出此二叉树。

七、（每小题 4 分，共 8 分）

对给定的有 7 个顶点的有向图的邻接矩阵如下：

（l）画出该有向图；

（2）若将图看成是 AOE-网，画出关键路径。

$$\begin{bmatrix} \infty & 2 & 5 & 2 & \infty & \infty & \infty \\ \infty & \infty & 2 & \infty & \infty & 8 & \infty \\ \infty & \infty & \infty & 1 & 3 & 5 & \infty \\ \infty & \infty & \infty & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 3 & 9 \\ \infty & \infty & \infty & \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

八、（本题 8 分）

给出一组关键字 29、18、25、47、58、12、51、10，分别写出按下列各种排序方法进行排序时的变化过程：

（1）归并排序，每归并一次书写一个次序。

（2）快速排序，每划分一次书写一个次序以及最后排好序后的序列。

（3）堆排序，先建成一个堆，然后每从堆顶取下一个元素后，将堆调整一次。

九、（本题 9 分）

试分别画出具有 3 个结点的树和具有 3 个结点的二叉树的所有不同形态。

十、（本题 15 分）

已知两个带头结点的单链表 A 和 B 分别表示两个集合，元素值递增有序，设计算法求出 A，B 的交集 C，并同样以递增的形式存储。

模拟试题（六）参考答案

一、单项选择题（每小题 2 分，共 20 分）

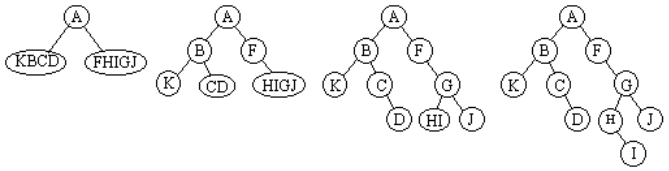
（1）D　　（2）C　　（3）D　　（4）C　　（5）B

（6）B　　（7）C　　（8）C　　（9）B　　（10）A

二、（本题 8 分）

线性表为：(90,40,78,50,34,60)

三、（本题 8 分）



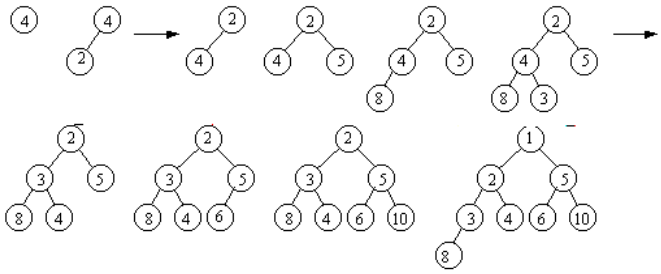
四、（本题 8 分）

用克鲁斯卡尔算法得到的最小生成树为：

(1,6)1,　(2,4)1,　(2,5)2,　(5,7)2,　(2,6)3,　(3,5)7

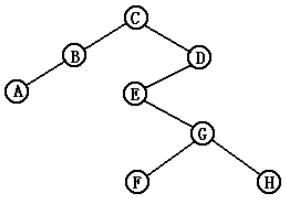
五、（本题 8 分）

如下图所示：



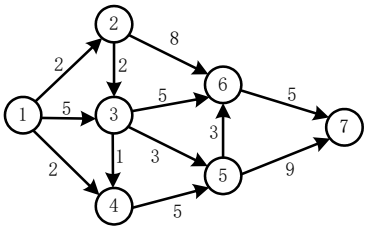
六、（本题 8 分）

根据后序序列知根结点为 C，所以左子树：中序序列为 AB，后序序列为 AB；右子树：中序序列为 EFGHD，后序序列为 FHGED，以此类推得该二叉树结构如下图所示。

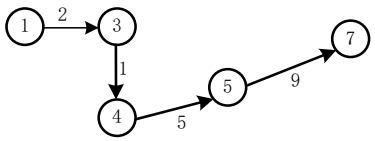


七、（每小题 4 分，共 8 分）

（1）由邻接矩阵所画的有向图如下图所示：



（2）关键路径如下图所示：



八、（本题 8 分）

变化过程如下：

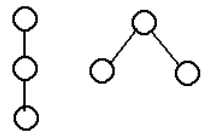
（1）归并排序：(l8,29)(25,47)(12,58)(l0,51)
(l8,25,29,47)(10,12,51,58)
(l0,12,18,25,29,47,51,58)

（2）快速排序：(10,18,25,12)29(58,51,47)
(10(18,25,12)29((47,51)58)
(10((12)18(25))29((47(51))58)
(10,12,18,25,29,47,51,58)

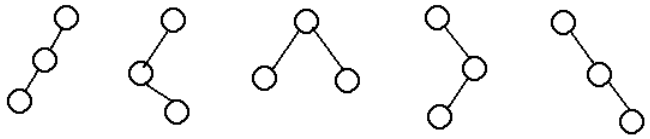
（3）堆排序：初始堆（大顶推）：(58,47,51,29,18,12,25,10)
第一次调整：(51,47,25,29,18,12,10)(58)
第二次调整：(47,29,25,10,18,12)(51,58)
第三次调整：(29,18,25,10,12)(47,51,58)
第四次调整：(25,18,12,10)(29,47,51,58)
第五次调整：(l8,10,12)(25,29,47,51,58)
第六次调整：(l2,10)(18,25,29,47,51,58)
第七次调整：(l0,12,18,25,29,47,51,58)

九、（本题 9 分）

具有 3 个结点的树的不同形态如下图所示。



具有 3 个结点的二叉树的不同形态如下图所示。



十、（本题 15 分）

解答：由于单链表 A 和 B 是递增有序的，可设置两个整型变量分别表示两个单链表的当前元素的位置，依次取出 A 与 B 的元素进行比较，当 A 的数据元素小于 B 的值时，将指向 A 的当前位置都后移；当 A 的数据元素大于 B 的值时，将指向 B 的当前位置都后移，否则复将 A（或 B）的当前元素制到 C 中，并同时将 A 与 B 的当前位置后移。具体算法如下：

用单链表 la 表示集合 la，用单链表 lb 表示集合 lb，用单链表 lc 表示集合 lc，具体算法如下：

```
// 文件路径名:exam6\alg.h
template <class ElemType>
void Interaction(const LinkList<ElemType> &la, const LinkList<ElemType> &lb,
     LinkList<ElemType> &lc)
// 初始条件: la 和 lb 中数据元素递增有序
// 操作结果: lc 返回 la 与 lb 表示的集合的交集，并使 lc 中数据元素仍递增有序
{
     ElemType aItem, bItem;                          // la 和 lb 中当前数据元素
     int aLength = la.Length(), bLength = lb.Length();   // la 和 lb 的长度
     int aPosition = 1, bPosition = 1;               // la 和 lb 的当前元素序号

     lc.Clear();                                     // 清空 lc
     while (aPosition <= aLength && bPosition <= bLength )
     {    // 取出 la 和 lb 中数据元素进行归并
          la.GetElem(aPosition, aItem);              // 取出 la 中数据元素
          lb.GetElem(bPosition, bItem);              // 取出 lb 中数据元素
          if (aItem < bItem)
          {    // aItem 插入到 lc
               aPosition++;                          // 指向 la 下一数据元素
          }
          else if (aItem > bItem)
          {    // lb 后移
               bPosition++;                          // 指向 lb 下一数据元素
          }
```

```
            else
            {     // aItem == bItem，la 和 lb 同时后移
                  lc.Insert(lc.Length() + 1, aItem);          // 插入 aItem 到 lc
                  aPosition++;                                // 指向 la 下一数据元素
                  bPosition++;                                // 指向 lb 下一数据元素
            }
      }
}
```

*模拟试题（七）

注：本套试题选作

一、单项选择题（每小题 2 分，共 20 分）

（1）若以 1234 作为双端队列的输入序列，则既不能由输入受限双端队列得到，也不能由输出受限双端队列得到的输出序列是（　　）。

A）1234　　B）4132　　C）4231　　D）4213

（2）将一个 A[1..100,1..100]的三对角矩阵，按行优先存入一维数组 B[298]中，A 中元素 $a_{66,65}$ 在 B 数组中的位置 k 为（　　）（假设 B[0]的位置是 1）。　A）198

B）195　　C）197　　D）198

（3）若度为 m 的哈夫曼树中，其叶结点个数为 n，则非叶结点的个数为（　　）。

A）n-1　　B）$\left\lfloor \frac{n}{m} \right\rfloor - 1$　　C）$\left\lceil \frac{n-1}{m-1} \right\rceil$　　D）$\left\lceil \frac{n}{m-1} \right\rceil - 1$

（4）若一个有向图具有拓扑排序序列，并且顶点按拓扑排序序列编号，那么它的邻接矩阵必定为（　　）。

A）对称矩阵　　B）稀疏矩阵　　C）三角矩阵　　D）一般矩阵

（5）设森林 F 对应的二叉树为有 m 个结点，此二叉树根的左子树的结点个数为 k，则另一棵子树的结点个数为（　　）。

A）m-k+1　　B）k+1　　C）m-k-1　　D）m-k

（6）假定有 K 个关键字互为同义词，若用线性探测法把这 K 个关键字存入散列表中，至少要进行（　　）次探测。

A）K-1 次　　B）K 次　　C）K＋l 次　　D）K(K+1)/2 次

（7）一棵深度为 k 的平衡二叉树，其每个非终端结点的平衡因子均为 0，则该树共有（　　）个结点。

A）$2^{k-1}-1$　　B）2^{k-1}　　C）$2^{k-1}+1$　　D）$2^{k}-1$

（8）如表 r 有 100000 个元素，前 99999 个元素递增有序，则采用（　　）方法比较次数较少。

A）直接插入排序　　B）快速排序　　C）归并排序　　D）选择排序

（9）如果只考虑有序树的情形，那么具有 7 个结点的不同形态的树共有（　　）棵。

A）132　　B）154　　C）429　　D）前面均不正确

（10）对 n(n>=2)个权值均不相同的字符构成哈夫曼树，关于该树的叙述中，错误的是（　　）。

A）该树一定是一棵完全二叉树

B）树中一定没有度为 1 的结点

C）树中两个权值最小的结点一定是兄弟结点

D）树中任一非叶结点的权值一定不小于下一任一结点的权值

二、（本题 8 分）

斐波那契数列 Fn 定义如下：

$$F_0=0，F_1=1，F_n=F_{n-1}+F_{n-2}$$

请就此斐波那契数列，回答下列问题：

（1）在递归计算 F_n 的时候，需要对较小的 F_{n-1}，F_{n-2}，…，F_1，F_0 精确计算多少次？

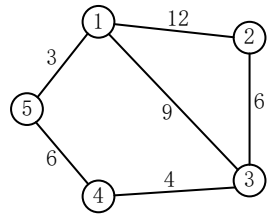
（2）若用有关大 O 表示法，试给出递归计算 Fn 时递归函数的时间复杂度是多少？

三、（本题 8 分）

证明：如果一棵二叉树的后序序列是 $u_1,u_2,\cdots,u_n$，中序序列是 $u_{p_1},u_{p_2},\cdots,u_{p_n}$，则由序列 1,2,…,n 可通过一个栈得到序列 $p_1,p_2,\cdots,p_n$。

四、（本题 8 分）

如下图所示为 5 个乡镇之间的交通图，乡镇之间道路的长度如图中边上所注。现在要在这 5 个乡镇中选择一个乡镇建立一个消防站，问这个消防站应建在哪个乡镇，才能使离消防站最远的乡镇到消防站的路程最短。试回答解决上述问题应采用什么算法，并写出应用该算法解答上述问题的每一步计算结果。



五、（本题 8 分）

证明一个深度为 n 的 AVL 树中的最少结点数为：$Nn=F_{n+2}-1$ $(n\geq 0)$

其中，Fi 为 Fibonacci 数列的第 i 项。

六、（本题 8 分）

简单回答有关 AVL 树的问题：（北方名校经典试题）

（1）在有 n 个结点的 AVL 树中，为结点增加一个存放结点高度的数据成员，那么每

一个结点需要增加多少个字位（bit）？

（2）若每一个结点中的高度计数器有 8bit，那么这样的 AVL 树可以有多少层？最少有多少个关键字？

七、（本题 8 分）

设有 12 个数据 {25，40，33，47，12，66，72，87，94，22，5，58}，它们存储在散列表中，利用线性探测再散列解决冲突，要求插入新数据的平均查找次数不超过 3 次。

（1）该散列表的大小 m 应设计多大？

（2）试为该散列表设计相应的散列函数。

（3）顺次将各个数据散列到表中。

（4）计算查找成功的平均查找次数。

八、（本题 8 分）

已知某电文中共出现了 10 种不同的字母，每个字母出现的频率分别为 A：8，B：5，C：3，D：2，E：7，F：23，G：9，H：11，I：2，J：35，现在对这段电文用三进制进行编码（即码字由 0，l，2 组成），问电文编码总长度至少有多少位？请画出相应的图。

九、（本题 9 分）

已知一棵度为 m 的树中有 N_1 个度为 1 的结点，N_2 个度为 2 的结点，…，N_m 个度为 m 的结点。试问该树中有多少个叶子结点？（北方名校经典试题）

十、（本题 15 分）

试用递归法编写输出从 n 个数中挑选 k 个进行排列所得序列的算法。

模拟试题（七）参考答案

一、单项选择题（每小题 2 分，共 20 分）

（1）参考答案：C）

（2）【分析】如下所示，三对角矩阵第 1 行和最后 1 行非零元素个数为 2 个，其余各行的非零元素个数是 3 个，所知 $a_{66,65}$ 前面共有 2+3*64＝194 个非零元素，$a_{66,65}$ 本身是第 195 个非零元。

$$A=\begin{bmatrix} a_{11} & a_{12} & & & & & & \\ a_{21} & a_{22} & a_{23} & & & & & \\ & a_{32} & a_{33} & a_{34} & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & a_{n-2,n-1} & a_{n-2,n-2} & a_{n-2,n-1} & & \\ & & & & a_{n-1,n} & a_{n-1,n-1} & a_{n-1,n} & \\ & & & & & a_{n,n-1} & a_{nn} \end{bmatrix}$$

参考答案：B）

（3）【分析】在哈夫曼树的非叶结点中最多只有 1 个结点的度不为 m，设非叶结点的个数为 k，则其中有 k-1 个结点的度为 m，设另 1 个结点的度为 u，则 2≤u≤m，设结点总数为 $n_{总}$，则有如下关系：

$n_{总}-1=m(k-1)+u$ ①

$n_{总}=k+n$ ②

将②代入①可得：k+n-1= m(k-1)+u，解得：$k=\dfrac{(n-1)+(m-u)}{m-1}$，由于 2≤u≤m，所以可得 0≤m-u＜m-1，所以可得：$\dfrac{n-1}{m-1}\leqslant k<\dfrac{n-1}{m-1}+1$，可知$k=\left\lceil\dfrac{n-1}{m-1}\right\rceil$。

参考答案：C）

（4）【分析】设顶点按拓扑排序序列为：$v_0,v_1,\ldots,v_{n-1}$，则对于邻接矩阵 A，只有当 i<j 时，才可能有弧< v_i,v_j>，也就是当 i>j 时，一定没有弧< v_i,v_j>，所以这时 A[i][j]=0，可知邻接矩阵为三角矩阵。

参考答案：C）

（5）【分析】设另一棵子树的结点个数为 n，所以有 m=n+k+1，可知 n= m-k-l。

参考答案：C）

（6）【分析】因为 K 个关键字互为同义词，只有在存入第一个关键字的情况下不发生冲突，所以至少需进行 1+2+…+K=K(K+1)/2 次探测。

参考答案：D）

（7）【分析】由于每个非终端结点的平衡因子均为 0，所以每个非终端结点必有左右两个孩子，且左子树的高度和右子树的高度相同，这样 AVL 树是满二叉树。高度为 k 的满二叉树的结点数为 2^k-l。

参考答案：D）

（8）【分析】本题中只有直接插入排序利用前面有序的子序列这个性质，如用直接插入排序对本题只需将最后一个元素插入到前面 99999 个元素的有序子序列中即可，显然比较次数较少。

参考答案：A）

（9）【分析】具有 n 个结点有不同形态的树的数目和具有 n-l 个结点互不相似的二叉树的数目相同（将树转化为二叉树时，根结点右子树为空，所以除根结点而外只有左子树，其不相似的二叉树的等价于不相似的左子树）。具有 n 个结点互不相似的二叉树的数目为 $\dfrac{1}{n+1}C_{2n}^{n}$，本题中应为$\dfrac{1}{6+1}C_{12}^{6}=132$。

参考答案：A）

（10）参考答案：A）

二、（本题 8 分）

【解答】

（1）设在计算 F_n 时，由 $F_{n-1}+F_{n-2}$ 可知 F_{n-1} 要精确计算 1 次；

由 $F_{n-1}=F_{n-2}+F_{n-3}$ 可知 $F_n=2F_{n-2}+F_{n-3}$，F_{n-2} 要精确计算 2 次；

由 $F_{n-2}=F_{n-3}+F_{n-4}$ 可知 $F_n=3F_{n-3}+2F_{n-4}$，F_{n-3} 要精确计算 3 次，$F_n=3F_{n-3}+2F_{n-4}$ 公式中 F_{n-3} 的系数为 F_{n-3} 要精确计算次数，而 F_{n-4} 的系数为 F_{n-2} 要精确计算次数，以此类推，设 F_{n-j} 的精确计算次为 a_j，则有：$F_n=a_j*F_{n-j}+a_{j-1}*F_{n-j-1}$。

由 $F_{n-j}=F_{n-j-1}+F_{n-j-2}$ 可知 $F_n=(a_j+a_{j-1})*F_{n-j-1}+a_j*F_{n-j-2}$ ，F_{n-j-1} 的精确计算次数为 a_{j+1}，所以有：

$$a_{j+1}=a_j+a_{j-1}$$

由于 F_{n-1} 要精确计算 a_1 为 1 次，即 $a_1=1$，即可知 F_{n-1}，F_{n-2}，…，F_1，F_0 的精确计算次为：1，2，3，5，……，$a_j=a_{j-1}+a_{j-2}$……

与斐波那契数列数列：

0，1，2，3，5，……，$F_n=F_{n-1}+F_{n-2}$……

比较可知 $a_j=F_{j+1}$。

（2）由于 F_n 的计算最终要转化为 F_0 与 F_1 之和，其加法的计算次数为 F_0 与 F_1 的精确计算次数之和再减 1 之差，由于 $F_0=F_{n-n}$ 与 $F_1=F_{n-(n-1)}$，所以计算 F_n 时，加法计算次数为：

$$a_n+a_{n-1}-1=F_{n+1}+F_n-1$$

由于 Fn= $\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n-\frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^n$，可知时间复杂度为 $O((\frac{1+\sqrt{5}}{2})^n)$。

三、（本题 8 分）

【解答】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。

四、（本题 8 分）

【解答】由弗洛伊德（Floyd）算法进行求解，具体步骤如下：

$$D^{(-1)}=\begin{bmatrix}0&12&9&\infty&3\\12&0&6&\infty&\infty\\9&6&0&4&\infty\\\infty&\infty&4&0&6\\3&\infty&\infty&6&0\end{bmatrix},\quad D^{(0)}=\begin{bmatrix}0&12&9&\infty&3\\12&0&6&\infty&15\\9&6&0&4&12\\\infty&\infty&4&0&6\\3&15&12&6&0\end{bmatrix};$$

$$D^{(1)}=\begin{bmatrix}0&12&9&\infty&3\\12&0&6&\infty&15\\9&6&0&4&12\\\infty&\infty&4&0&6\\3&15&12&6&0\end{bmatrix},\quad D^{(2)}=\begin{bmatrix}0&12&9&13&3\\12&0&6&10&15\\9&6&0&4&12\\13&10&4&0&6\\3&15&12&6&0\end{bmatrix};$$

$$D^{(3)}=\begin{bmatrix}0&12&9&13&3\\12&0&6&10&15\\9&6&0&4&12\\13&10&4&0&6\\3&15&10&6&0\end{bmatrix},\quad D^{(4)}=\begin{bmatrix}0&12&9&9&3\\12&0&6&10&15\\9&6&0&4&10\\9&10&4&0&6\\3&15&10&6&0\end{bmatrix}。$$

设乡镇 v_i 到其他各乡镇的最远距离为 max_disdance(v_i)，则有：max_disdance(v_1)=12，max_disdance(v_2)=15，max_disdance(v_3)=10，max_disdance(v4)=10，max_disdance(v_5)=15，所以可知消防站应建在 v_3 或 v_4 乡镇，才能使离消防站最远的乡镇到消防站的路程最短。

五、（本题 8 分）

【解答】对 n 用归纳法证明。当 n＝1 时，有 N1=F3-l=2-l=1 到。当 n=2 时，有 N2=F4-1=3-1=2。设 n<k 时也成立，即有 $N_n=F_{n+2}-1$ 成立。

当 n=k+1，对于一个 k+l 层深度的平衡二叉树而言，其左右子树都是平衡的。结点数为最少的极端情况，故左右子树中的结点数是不相等的，设其中一个是 k 层深度的二叉平衡树，另一个是 k-l 层深度的二叉平衡树。所以有：

$N_{k+1}=1+N_k+N_{k-1}==1+(F_{k+2}-1)+(F_{k+1}-1)= F_{k+2}+F_{k+1}-1= F_{k+3}-1$

当 n=k+1 时成立，由此可知深度为 n 都等式都成立。

六、（本题 8 分）

【解答】n 个结点的平衡二叉树的最大高度为 $h=\log_{\frac{1+\sqrt{5}}{2}}(\sqrt{5}(n+1))-2$，设表示高度需 xbit，则有关系式：$2^x \geqslant h > 2^{x-1}$，所以有：

$$x=\lceil \log_2 h \rceil=\left\lceil \log_2(\log_{\frac{1+\sqrt{5}}{2}}(\sqrt{5}(n+1))-2) \right\rceil$$

（2）设深度为 h 的平衡二叉树的最少关键字数为 n_h，则有公式：$n_h=\frac{(\frac{1+\sqrt{5}}{2})^{h+2}}{\sqrt{5}}-1$，本题中 8bit 的计数器共可以表示 2^8=256 层，即高度为 256，从而可知最少有 $n=\frac{(\frac{1+\sqrt{5}}{2})^{258}}{\sqrt{5}}-1$ 个关键字。

七、（本题 8 分）

【解答】

（1）线性探测再散列的哈希表查找成功的平均查找长度为：$S_{nl}\approx\frac{1}{2}(1+\frac{1}{1-\alpha})\leqslant 3$，解得 α ≤4 / 5，也就是 12/m≤4 / 5， 所以 m≥15，可取 m=15。

（2）散列函数可取为 H(key)=key % 13

（3）散列表如表 7-4 所示。

散列表

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	40	66	94		5	58	33	47	72	87	22	25	12	

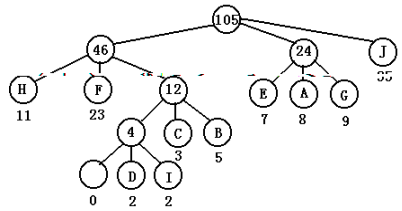
（4）12 个数据 {25，40，33，47，12，66，72，87，94，22，5，58} 的比较次数分别是：1，1，1，1，2，2，3，2，1，3，1，1。

可知查找成功的平均查找次数=(1+1+1+1+2+2+3+2+1+3+1+1)/12=1.25

八、（本题 8 分）

【解答】有 n 个叶子结点的带权路径长度最短的二叉树称哈夫曼树，同理，存在有 n 个叶子结点的带权路径长度最短的三叉、四叉、……、k 叉树，也称为哈夫曼树。如叶子结点数目不足以构成正则的 k 叉树（树中只有度为 k 或 0 的结点），即不满足 (n-l)MOD(k-l)=0，需要添加权为 0 的结点，添加权为 0 的结点的个数为：k-(n-1)MOD(k-1)-1。添加的位置应该是距离根结点的最远处。

所构造出的哈夫曼树如下图所示：



其中，每个字母的编码长度等于叶子结点的路径长度，电文的总长度为 $\sum_{i=1}^{n} w_i l_i$ =191。

注释：对于正则的 k 叉树，设叶结点数为 n，度为 k 的结点数为 n_k，结点总数为 m，则有 $m-1=kn_k$，$m=n_k+n$，两式相减可得 $n-1=(k-1)n_k$，即(n-l)MOD(k-l)=0；如 n 与 k 不满足此关系，n 加上 k-(n-1)MOD(k-1)-1 后，n'= n+(k-(n-1)MOD(k-1)-1)，这时：

(n'-l)MOD(k-l)=(n+(k-(n-1)MOD(k-1)-1)-l)MOD(k-l)

=((n-1)+(k-1)-(n-1)MOD(k-1))MOD(k-l)

=((n-1)-(n-1)MOD(k-1))MOD(k-l)

=0。

现有 12 个初始归并段，其记录数分别为{30,44,8,6,3,20,60,18,9,62,68,85}，采用 3-路平衡归并，画出最佳归并树。

九、（本题 9 分）

【解答】设该树中结点总数为 N，叶子结点个数为 N_0，则有：

$$N=\sum_{i=0}^{m} N_i \qquad (1)$$

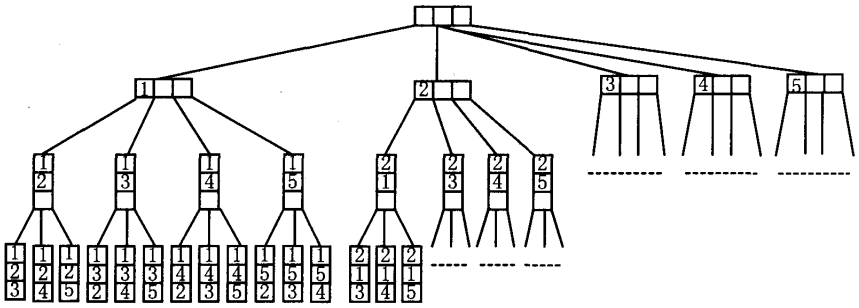
$$N-1=\sum_{i=1}^{m} iN_i \qquad (2)$$

由（2）-（1）再经过移项可得：$N_0=1+\sum_{i=1}^{m}(i-1)N_i$

十、（本题 15 分）

【解答】

对于排列的解空间可构造一个虚拟的解空间树，比如 n=5，k=3 时的解空间树如下图所示，可采用对此树进行先序遍历方式进行遍历，并用递归法进行递归输出从 n 个数中挑选 k 个进行排列所得序列。



具体算法实现如下：

```
// 文件路径名:exam7\alg.h
template <class ElemType>
void Arrage(ElemType a[],int k,int n, int outlen=0)
// 操作结果: 回溯法输出排列序列，a[0..k-1]为 k 个数的排列序列 outlen 为当前所求排列
//      序列的长度，其中 outlen=k 时的排列序列为所求；n 为 list 数组长度
{
    if (k < 0 || k >= n) return;                    // 此时无排列
    int i;                                          // 临时变量
    if (outlen == k + 1)
    {   // 得到一个排列
        for (i = 0; i < k; i++)
        {   // 输出一个排列
            cout << a[i];                           // 输出 a[i]
        }
        cout << "   ";                              // 用空格分隔不同排列
    }
    else
    {   // 对解空间进行前序遍历，a[outlen..n]有多个排列，递归的生成排列
        for (i = outlen; i < n; i++)
        {   // 处理 a[i]
            Swap(a[outlen+1], a[i]);        // 交换 a[outlen+1]与 a[i]
            Arrage(a, k, n, outlen + 1);    // 对序列长度 outlen+1 递归
            Swap(a[outlen+1], a[i]);        // 交换 a[outlen+1]与 a[i]
        }
    }
}
```

*模拟试题（八）

注：本套试题选作

一、单项选择题（每小题 2 分，共 20 分）

（1）一个 n*n 的带状矩阵 A=[a_{ij}]如下：

$$A=\begin{bmatrix} a_{11} & a_{12} & & & & & \\ a_{21} & a_{22} & a_{23} & & & & \\ & a_{32} & a_{33} & a_{34} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_{n-2,n-1} & a_{n-2,n-2} & a_{n-2,n-1} & \\ & & & & a_{n-1,n} & a_{n-1,n-1} & a_{n-1,n} \\ & & & & & a_{n,n-1} & a_{nn} \end{bmatrix}$$

将带状区域中的元素 a_{ij}（|i-j|≤1）按行序为主序存储在一维数组 B[3n-2]中，元素 a_{ij} 在 B 中的存储位置是（　　）。

A）i+2j-2　　B）2i+j-3　　C）3i-j　　D）i+j+1

（2）一 n×n 的三角矩阵 A=[a_{ij}]如下：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & \cdots & a_{2n} \\ & & \cdots & \\ & & & a_{nn} \end{bmatrix}$$

将三角矩阵中元素 a_{ij}(i≤j)按行序为主序的顺序存储在一维数组 B[n(n+1)/2]中，则 a_{ij} 在 B 中的位置是（　　）。

A）(i-1)(2n+i)/2+i-j　　B）(i-1)(2n-i+2)/2+j-i

C）(i-1)(2n-i)/2+j-i-1　　D）(i-1)(2n-i+2)/2+j-i-1

（3）设有一棵度为 3 的树，其叶结点数为 n_0，度为 1 的结点数为 n_1，度为 2 的结点数为 n_2，度为 3 的结点数为 n_3，则 n_0 与 n_1，n_2，n_3 满足关系（　　）。

A）$n_0=n_2+1$　　B）$n_0=n_1+2n_3+1$　　C）$n_0=n_2+n_3+1$　　D）$n_0=n_1+n_2+n_3$

（4）G 是一个非连通无向图，共有 28 条边，则该图至少有（　　）个顶点。

A）6　　B）7　　C）8　　D）9

（5）设图 G 用邻结表存储，则拓扑排序的时间复杂度为（　　）。

A）O(n)　　B）O(n+e)　　C）O(n^2)　　D）O(n×e)

（6）用 n 个键值构造一棵二叉排序树，最低高度为（　　）。

A）$\frac{n}{2}$　　B）$\sqrt{n}$　　C）$n\log_2 n$　　D）$\lfloor \log_2 n \rfloor+1$

（7）若需在 O(nlogn)的时间内完成对数组的排序，且要求排序是稳定的，则可选择的

排序方法是（　　）。

A）快速排序　　B）堆排序　　C）归并排序　　D）直接插入排序

（8）在文件“局部有序”或文件长度较小的情况下，最佳内部排序的方法是（　　）。（北方名校经典试题）

A）直接插入排序　B）起泡排序　　C）简单选择排序 D）基数排序

（9）一棵有 124 个叶结点的完全二叉树，最多有（　　）个结点。

A）247　　B）248　　C）249　　D）250

（10）一个序列中有 10000 个元素，若只想得到其中前 10 个最小元素，最好采用（　　）方法。

A）快速排序　　B）堆排序

C）插入排序　　D）二路归并排序

二、（本题 8 分）

要借助栈由输入序列是输入 1,2,3,…,n 得到的输出序列为 $p_1,p_2,p_3,\cdots,p_n$(此输出序列是输入序列经过栈操作后的某个排列)，则在输出序列中不可能出现当 i<j<k 时有 $p_j<p_k<p_i$ 的情况。

三、（本题 8 分）

已知某一完全 k 叉树只有度为 k 的结点及叶结点，设叶结点数为 n_0，试求它的树高 h。（南方名校经典试题）

四、（本题 8 分）

试讨论怎样在一棵中序线索二叉树上查找给定结点 x 在后序序列中的后继。

五、（本题 8 分）

具有 n 个关键字的 B 一树的查找的最大路径长度是多少?

六、（本题 8 分）

对长度为 12 的有序表(a_1，a_2，…，a_{12})（其中 $a_i<a_j$ 当 i<j 时）进行折半查找，在设定查找不成功时，关键字 $x<a_1$、$x>a_{12}$ 以及 $a_i<x<a_{i+1}$(i=1,2,…,11)等情况发生的概率相等，则查找不成功的平均查找长度是多少?

八、（本题 8 分）

（1）设 T 是具有 n 个内结点的判断二叉树，I 是它的内路径长度，E 是它的外路径长度，试利用归纳法证明 E＝I＋2n，n＞0。

（2）利用（1）的结果，试说明，成功查找的平均比较次数 s 与不成功查找的平均比较次数 u 之间的关系，可用公式 $s=(1+\frac{1}{n})u-1$，n＞0 表示。

提示：判断二叉树只有度为 0 或度为 2 的结点；判断二叉树成功查找的比较次数为内路径长度与内结点数之和，不成功查找的比较次数为外路径长度。

九、（本题 9 分）

一个深度为 h 的满 m 叉树有如下性质：第 h 层上的结点都是叶结点，其余各层上每个结点有 m 棵非空子树。问：

（1）第 k 层有多少个结点？（k≤h）

（2）整棵树有多少个结点？

（3）若按层次从上到下，每层从左到右的顺序从 1 开始对全部结点编号，编号为 i 的结点的双亲结点的编号是什么？编号为 i 的结点的第 j 个孩子结点（若存在）的编号是什么？

十、（本题 15 分）

设散列表的关键字取值范围为 0～m-1，n 为对散列表的最大插入次数，设计散列表，允许使用以 O(m+n)空间，要求查找、插入和删除算法的时间复杂度都是 O(1)。

模拟试题（八）参考答案

一、单项选择题（每小题 2 分，共 20 分）

（1）参考答案：B）

（2）【分析】存储位置=n+(n-1)+…+(n-i+2)+i-j=(i-1)(2n-i+2)/2+j-i。

参考答案：B）

（3）【分析】用 n 表示结点总数，则有：$n= n_0+n_1+n_2+n_3$；

由于除根结点而外，结点与分支一一对应，而分支数= $n_1+2n_2+3n_3$，即有：$n-1= n_1+2n_2+3n_3$。

由上面两式可得：$n_0=n_1+2n_3+1$

参考答案：B）

（4）【分析】本题中由于是非连图，至少有一个顶点与其他顶点不连，这个顶点是孤立点，其他顶点可组成一个连通图，由于 8 个顶点的完全图共有 28 条边，所以具体 28 个顶点的连通图的顶点个数至少为 8，这样非连通图至少有 9 个顶点。

参考答案：D）

（5）【分析】对于有 n 个顶点 e 条边的有向图，建立各顶点的入度时间复杂度为 O(e)，建立入度为零的栈的时间复杂度为 O(n)，在拓扑排序过程中，最多每个顶点进一次栈，入度减 1 的操作最多总共执行 e 次，可知总的时间复杂度为 O(n+e)

参考答案：B）

（6）【分析】当用 n 个键值构造一棵二叉排序树是一棵完全二叉树时，高度最低，此时高度为$\lfloor \log_2 n \rfloor+1$。

参考答案：D）

（7）【分析】快速排序和堆排序都是不稳定的，应排除；归并排序稳定，时间复杂度 O(nlogn)，满足条件；直接插入排序，时间复杂度为 $O(n^2)$，排除。

参考答案：C）

（8）【分析】对直接插入排序而言，算法时间复杂度为 $O(n^2)$，但若待排记录序列为“正序”时，其时间复杂度可提高至 O(n)。若待排记录序列按关键字“基本有序”，直接插入排序的效率就可大大提高，此外由于直接插入排序算法简单，在 n 值很小时效率也高。

参考答案：A）

（9）【分析】完全二叉树中度为 1 的结点最多只有 1 个，由二叉树的度和结点的关系：

$n=n_0+n_1+n_2$ (1)

$n=n_1+2n_2+1$ (2)

由 2(1)-(2)得，$n=2n_0+n_1-1=247+n_1\leqslant 248$，所以本题应选择 B）。

参考答案：A）

（10）参考答案：B）

二、（本题 8 分）

【解答】设 $p_j<p_k<p_i$ 成立，则表示在 p_i 出栈之前 p_j 和 p_k 都已入栈，并且还留在栈中，但要满足 $p_j<p_k$ 的出栈次序是不可能的，由于按照 i<j<k 的次序，当 p_j 和 p_k 同时在栈中时，必然 p_k 被 p_j 盖住，由栈的后进先出的性质，当 i<j<k 有 $p_k< p_j <p_i$，与假设矛盾。

三、（本题 8 分）

【解答】设度为 k 的结点数为 n_k，结点总数为 n，则有如下关系：

$$n= n_k+n_0 \qquad ①$$

又由于树中只有 n-1 条边，所以：

$$n-1=k\times n_k \qquad ②$$

由①与②可得：$n_k=(n_0-1)/(k-1)$，进而有 $n=\dfrac{k\times n_0-1}{k-1}$

对于 k 叉完全树有如下关系：$k^{h-1}-1<n\times(k-1)\leqslant k^h-1$

即有：$k^{h-1}\leqslant n\times(k-1)<k^h$，从而：$h-1\leqslant \log_k(n\times(k-1))<h$，进而：$h=\lfloor \log_k(n\times(k-1))\rfloor+1$

所以：$h=\lfloor \log_k(k\times n_0-1)\rfloor+1$

四、（本题 8 分）

【解答】由于后序遍历二叉树需要知道的关键是访问当前结点的双亲结点，需要由中序线索树才能得到当前结点的双亲，中序线索树有如下性质：

若 x 是 parent 的左孩子，则 parent 是 x 的最右子孙的右线索；

若 x 是 parent 的右孩子，则 parent 是 x 的最左子孙的左线索。

用以上性质能找到 x 的双亲结点 parent。

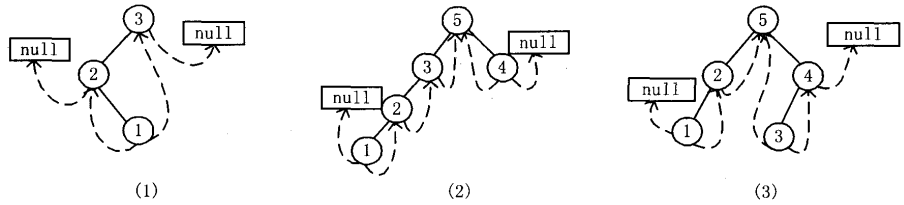
若 x 是 parent 的右孩子，则 parent 结点就是 x 的后序序列的后继结点；如下图（1）中结点①的后继是结点②。

若 x 是 parent 的左孩子，则：

如果 parent 的右指针域为线索的话，那么 parent 就是 x 的后序序列的后继结点，如下

图（2）中结点②的后继是结点③。

否则 parent 右子树中最左边第一个左右孩子均为线索的结点，就是 x 的后序序列的后继结点。如下图（3）中结点②的后继是结点③。



(1)　　(2)　　(3)

五、（本题 8 分）

【解答】树的查找路径是从根结点开始到所要查找的结点的路径，最大不会超过 B-树的深度。在 B 树中，除根结点外所有非终端结点都含有$\left\lceil \frac{m}{2} \right\rceil$棵子树，所以有 n 个关键字的 B-树的最大深度为根结点具有两棵子树，其余非终端点有$\left\lceil \frac{m}{2} \right\rceil$棵子树，设最大路径长度是 x，由于叶子结点表示查找不成功，叶子结点不含关键字，可知 B-树的深度为 x+1，

第 1 层共有 1 个结点，含 1 个关键字；

第 2 层共有 2 个结点，含 2($\left\lceil \frac{m}{2} \right\rceil$-1)个关键字；

第 3 层共有 2$\left\lceil \frac{m}{2} \right\rceil$个结点，含 2$\left\lceil \frac{m}{2} \right\rceil$($\left\lceil \frac{m}{2} \right\rceil$-1)个关键字；

……

第 x 层共有 2$\left\lceil \frac{m}{2} \right\rceil^{x-2}$个结点，含 2$\left\lceil \frac{m}{2} \right\rceil^{x-2}$($\left\lceil \frac{m}{2} \right\rceil$-1)个关键字；

故，共含有的关键个数为：

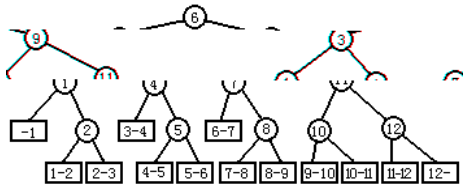
$1+2(\left\lceil \frac{m}{2} \right\rceil-1)+2\left\lceil \frac{m}{2} \right\rceil(\left\lceil \frac{m}{2} \right\rceil-1)+\cdots+2\left\lceil \frac{m}{2} \right\rceil^{x-2}(\left\lceil \frac{m}{2} \right\rceil-1)=2\left\lceil \frac{m}{2} \right\rceil^{x-1}-1$

由此可得：$n=2\left\lceil \frac{m}{2} \right\rceil^{x-1}-1$，解得：$x=1+\log_{\left\lceil \frac{m}{2} \right\rceil}(\frac{n+1}{2})$，这就是具有 n 个关键字的 B 一树的查找的最大路径长度。

六、（本题 8 分）

【解答】折半查找对应的判定树如下图所示。查找不成功的对应于外部结点。查找不

成功所走的路径是从根结点到每个外部结点（图中方块结点），和给定值进行比较的关键字个数等于该路径上内部结点个数。



在不成功情况下，一共比较的次数为 3*3+4*10=49 次。

平均查找长度为 49/13≈3.77

七、（本题 8 分）

【解答】对于 a 来说，在 S2 中总可找到离 a 最近的祖先结点 d，这时 a<d，这时如 d 为 b 的右孩子，则有 a 在 b 的右子树上，所以 a>b，也就是说 a<b 并不总是成立，同理 b<c 也并不总是成立。

八、（本题 8 分）

【解答】

（1）证明：n=1 时显然成立，设 n=k 时成立，即 $E_k=I_k+2k$，当 n=k+1 时，设所增加的结点的路径长度为 D_k，则有：

$I_{k+1}=I_k+D_k$

$E_{k+1}=E_k-D_k+2(D_k+1)=E_k+D_k+2=I_k+2k+D_k+2=I_{k+1}+2(k+1)=I_{k+1}+2n$

结论也成立。

（2）根据二叉树性质：度为 0 的结点数=度为 2 的结点数+1，所以外结点数=n+l。

s=查找成功总的比较次数/内结点数=(I+n)/n ①

u=查找失败总的比较次数/外结点数=E/(n+1)=(I+2n)/(n+1) ②

由①得：I=ns-n，由②得：I=(n+1)u-2n，所以可得：ns-n=(n+1)u-2n，进一步得：

$s=(1+\frac{1}{n})u-1$。

九、（本题 9 分）

【解答】

（1）设第 v 层有 u 个结点(m<h)，则由于第 v 层的每个结点有 m 个孩子，所以第 v+1 层的结点个数为 mu。

第 1 层有 1 个结点，第 2 层有 m 个结点，第 3 层有 m^2 结点，用数学归纳法易知 k 层共有 m^{k-1} 个结点。

（2）整棵树结点数=$1+m+m^2+\cdots+m^{h-1}=\frac{m^h-1}{m-1}$。

（3）设编号为 i 的结点双亲的结点编号为 x，将编号为 i 的结点视为完全二叉树的最后一个结点，因此，此完全 m 叉树中至少有 x-l 个度为 m 的结点，而 x 号结点的度 d（1≤d≤m），其余的结点均为叶子结点，而编号 i 就是此完全 m 叉树的总结局数，于是有：

(x-1)m+d+1=i

所以有 $x=\frac{i+m-d-1}{m}$，由于 1≤d≤m，所以有：

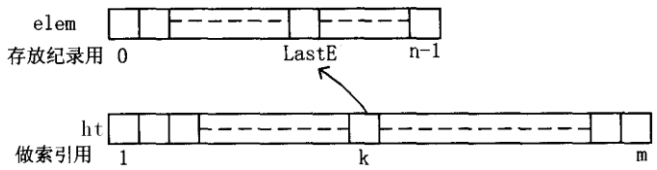
$$\frac{i+m-2}{m}-1<\frac{i+m}{m}-1\le x\le\frac{i+m-2}{m}$$

可知：$x=\left\lfloor\frac{i+m-2}{m}\right\rfloor$。

设编号为 i 的第 j 个子结点为完全 m 叉树的最后一个结点 n，此完全 m 叉树中有 i-1 个度为 m 的结点，一个度为 j 的结点，其他结点均为叶子结点，可得：n=(i-l)×m+j+1

十、（本题 15 分）

【解答】在理想情况下，可实现散列表查找、插入、删除操作的操作时间为 O(1)，对于本题，由于已经知道了关键字的取值范和散列表的最大插入次数，可作理想化处理哈希方法：将查找与数据在一起的哈希表一分为二：采用两个数组 ht 和 elem，其中 ht 是从 0 到 m-1 的整数数组（做索引用）；elem 数组的数据类型为哈希表中各元素的类型，其元素个数为 n（即最大的插入次数），两个数组都不需要初始化。使用计数器 lastE 表示上次所用到的 elem 中的位置，初值为-1。ht[k]中的值可能无效（用-1 表示）也可能是 elem 的索引（用来寻找关键字为 k 的元素），如下图所示：



具体算法实现当如下：

```
// 文件路径名:exam8\alg.h
// 理想散列表类
template <class ElemType, class KeyType>
class IdealHashTable
{
protected:
//  散列表的的数据成员:
    int *ht;                    // 用作 elem 的索引
    ElemType *elem;             // 在放插入的哈希表的元素
    int lastE;                  // 计数器，表示上次用到的 elem 位置
    int n;                      // 最大插入次数
    int m;                      // 关键字范围 0~m-1

public:
```

```
//   理想散列表方法声明及重载编译系统默认方法声明:
      IdealHashTable(int maxInsertCount, int size);              // 构造函数
      ~IdealHashTable();                                         // 析造函数
      void Traverse(void (*Visit)(const ElemType &)) const;  // 遍历散列表
       bool Search(const KeyType &key, ElemType &e) const;      // 查寻关键字为 key 的元素
       bool Insert(const ElemType &e);                          // 插入元素 e
       bool Delete(const KeyType &key, ElemType &e);// 删除关键字为 key 的元素，用 e 返回元素值
      IdealHashTable(const IdealHashTable<ElemType, KeyType> &copy);     // 复制构造函数
      IdealHashTable<ElemType, KeyType> &operator=
            (const IdealHashTable<ElemType, KeyType> &copy);              // 赋值语句重载
};

// 理想散列表类的实现部分
template <class ElemType, class KeyType>
IdealHashTable<ElemType, KeyType>::IdealHashTable(int maxInsertCount, int size)
// 操作结果: 构造最大插入次数为 maxInsertCount, 关键字范围为 0~size-1 的空理想散列表
{
      n = maxInsertCount;                    // 最大插入次数
      m = size;                              // 关键字范围为 0~size-1
      ht = new int[n];                       // 为 elem 索引表分配存储空间
      elem = new ElemType[m];                // 为元素分配存储空间
      lastE = -1;                            // 初始化 lastE
      for (int pos = 0; pos < n; pos++)
      {     // 初始化 ht
            ht[pos] = -1;
      }
}

template <class ElemType, class KeyType>
IdealHashTable<ElemType, KeyType>::~IdealHashTable()
// 操作结果: 销毁散列表
{
      delete []ht;                           // 释放 ht
      delete []elem;                         // 释放 elem
}

template <class ElemType, class KeyType>
void IdealHashTable<ElemType, KeyType>::Traverse(void (*Visit)(const ElemType &)) const
// 操作结果: 依次对散列表的每个元素调用函数(*visit)
{
      for (int pos = 0; pos < m; pos++)
      {     // 对散列表的每个元素调用函数(*visit)
            if (ht[pos] != -1) (*Visit)(elem[ht[pos]]);
      }
}
```

```
template <class ElemType, class KeyType>
bool IdealHashTable<ElemType, KeyType>::Search(const KeyType &key, ElemType &e) const
// 初始条件: 关键字 key 的范围为 0~m-1，ht[key]的范围为 0~lastE
// 操作结果: 查寻关键字为 key 的元素的值,如果查找成功,返回 true,并用 e 返回元素的值,
//      否则返回 false
{
    if (key < 0 || key >= m)
    {    // 范围错
        return false;              // 查找失败
    }
    else if (ht[key] < 0 || ht[key] > lastE || elem[ht[key]] != key)
    {    // 表中没有要查找的元素
        return false;              // 查找失败
    }
    else
    {    // 存在要查找的元素
        e = elem[ht[key]];         // 用 e 返回元素值
        return true;               //查找成功
    }
}

template <class ElemType, class KeyType>
bool IdealHashTable<ElemType, KeyType>::Insert(const ElemType &e)
// 操作结果: 将元素插入到 elem 数组 lastE 位置，同时修改索引值
{
    KeyType key = e;               // e 的关键字
    ElemType eTmp;                 // 临时变量
    if (key < 0 || key >= m)
    {    // 范围错
        return false;              // 插入失败
    }
    else if (Search(key, eTmp))
    {    // 查找成功
        return false;              // 插入失败
    }
    else if (lastE == n - 1)
    {    // 超过最大插入次数
        return false;              // 插入失败
    }
    else
    {
        elem[++lastE] = e;         // 修改计数器，将元素插入到表中
        ht[key] = lastE;           // 索引表 ht 中 ht[key]记录关键字为 key 的元素在 elem 中的位置
        return true;               // 插入成功
    }
}
```

```
template <class ElemType, class KeyType>
bool IdealHashTable<ElemType, KeyType>::Delete(const KeyType &key, ElemType &e)
// 操作结果: 删除关键字为 key 的数据元素,删除成功返回 true,并用 e 返回元素值，否则返回 false,
//      通过将 lastE 位置元素移到删除位置，lastE 指向倒数第二个元素，从逻辑上完成删除
{
    if (!Search(key, e))
    {   // 查找失败
        return false;                       // 删除失败
    }
    else
    {   // 查找成功
        e = elem[ht[key]];                  // 用 e 返回被删除元素值
        elem[ht[key]] = elem[lastE--]; // 将最后元素移到所删除处,将修改 LastE 指向新的末记录
        int k = elem[ht[key]];              // 原最后元素的关键字
        ht[k] = ht[key];                    // 修改索引值
        ht[key] = -1;                       // 修改 ht[key]为-1，表示关键字为 key 的元素已初删除
        return true;                        // 删除成功
    }
}

template <class ElemType, class KeyType>
IdealHashTable<ElemType, KeyType>::IdealHashTable(
    const IdealHashTable<ElemType, KeyType> &copy)
// 操作结果：由散列表 copy 构造新散列表--复制构造函数
{
    n = copy.n;                             // 最大插入次数
    m = copy.m;                             // 关键字范围为 0~m-1
    ht = new int[n];                        // 为 elem 索引表分配存储空间
    elem = new ElemType[m];                 // 为元素分配存储空间
    lastE = copy.lastE;                     // 计数器，表示上次用到的 elem 位置

    int pos;                                // 临时变量
    for (pos = 0; pos < m; pos++)
    {   // 复制元素索引
        ht[pos] = copy.ht[pos];             // 复制元素索引值
    }
    for (pos = 0; pos < m; pos++)
    {   // 依次复制数据元素
        elem[pos] = copy.elem[pos];         // 复制元素值
    }
}

template <class ElemType, class KeyType>
IdealHashTable<ElemType, KeyType> &IdealHashTable<ElemType, KeyType>::
operator=(const IdealHashTable<ElemType, KeyType> &copy)
// 操作结果：将散列表 copy 赋值给当前散列表--赋值语句重载
```

```
{
    if (&copy != this)
    {
        delete []ht;                        // 释放 ht
        delete []elem;                      // 释放 elem

        n = copy.n;                         // 最大插入次数
        m = copy.m;                         // 关键字范围为 0~m-1
        ht = new int[n];                    // 为 elem 索引表分配存储空间
        elem = new ElemType[m];             // 为元素分配存储空间
        lastE = copy.lastE;                 // 计数器，表示上次用到的 elem 位置

        int pos;                            // 临时变量
        for (pos = 0; pos < m; pos++)
        {   // 复制元素索引
            ht[pos] = copy.ht[pos];         // 复制元素索引值
        }
        for (pos = 0; pos < m; pos++)
        {   // 依次复制数据元素
            elem[pos] = copy.elem[pos];     // 复制元素值
        }
    }
    return *this;
}
```