数据结构导论

线性表

顺序表的结构定义

```
const int Maxsize = 100;    //预先定义一个足够大的常数
typedef struct
{
    DataType data[Maxsize];     //存放数据的数组
    int length;                 //顺序表的实际长度
} SeqList;      //顺序表类型名为SeqList
SeqList L;  //定义L为一个顺序表
```

单链表的类型定义

```
typedef struct node  //struct node 表示链表的结点，结点包含两个域：数据域（data）和指针域（next）
{
    DataType data;  //数据域
    struct node * next; //指针域
} Node, *LinkList;
```

循环链表 ：在单链表中，如果让最后一个结点的指针域指向第一个结点可以构成循环链表。在循环链表中，从任一结点出发能够扫描整个链表。

双向循环链表

```
struct dbnode
{
    DataType data;
    struct dbnode *prior, *next;
}
typedef struct dbnode *dbpointer;
typedef struct DLinkList;
```

第 3 章 栈、队列和数组

1. 栈

   顺序栈

   ```
   const int maxsize = 6;  //顺序栈的容量
   typedef struct seqstack
   {
       DataType data[maxsize];    //存储栈中数据元素的数组
       int top;                   //标志栈顶位置的变量
   } SeqStk;
   ```

顺序栈进栈 赋值语句：`top++`，`SeqStk[stk->top] = x`；

顺序栈出栈 赋值语句：`top--`；

链栈

```
typedef struct node
{
    DataType data;
    struct node *next;
} LkStk;
```

链栈进栈

```
void Push (LkStk *LS, DataType x)
{
    LkStk *temp;
    temp = (LkStk *)malloc(sizeof(LkStk));
    temp->data = x;
    temp->next = LS->next;  //将新结点插入到链表中
    LS->next = temp;        //修改栈顶指针指向新结点
}
```

链栈出栈 Pop(S)：删除栈顶元素

```
int Pop(LkStk *LS)
{
    LkStk *temp;
    if (!EmptyStack(LS))
    {
        temp = LS->next;
        LS->next = temp->next;
        free(temp); //释放原栈顶结点空间
        return 1;
    }
    else return 0;
}
```

2. 队列

顺序队列

```
const int maszize = 20;
typedef struct seqqueue
{
    DataType data[maxsize];
    int front, rear;
}SeqQue;
SeqQue SQ;
```

顺序队列入队 可用两条赋值语句：`SQ.rear = SQ.rear + 1`；`SQ.data[SQ.rear] = x`；

顺序队列出队列 赋值语句：`SQ.front = SQ.front + 1`；

循环队列的入队列 赋值语句：`SQ.rear = (SQ.rear + 1) % maxsize`；`SQ.data[SQ.rear] = x`；

循环队列出队列 赋值语句：`SQ.front = (SQ.front + 1) % maxsize`；

循环队列满 条件为：`(CQ.rear + 1) % maxsize == CQ.front`；

循环队列空 条件为：`CQ.rear == CQ.front`；

链接队列：队列的链接实现实际上是使用一个 带有头结点的单链 表来表示队列，称为 链队列 。头指针指向链表的头结点，单链表的头结点的 next 域指向队列首结点，尾指针指向队列尾结点，即单链表的最后一个结点。

```
typedef struct LinkQueueNode
{
    DataType data;
    struct LinkQueueNode *next;
} LkQueNode;

typedef struct LKQueue
{
    LKQueNode *front, *rear;
} LKQue;
LkQue LQ;
```

入队列

```
void EnQueue(LKQue *LQ; DataType x)
{
 LKQueNode *temp;
 temp = (LKQueNode *)malloc(sizeof(LKQueNode));
 temp->data = x;
 temp->next = NULL;
 (LQ->next)->next = temp;  //新结点入队
 LQ->rear = temp;  //修改队尾指针
}
```

出队列

```
OutQueue(LKQue *LQ)
{
    LKQueNode *temp;
     if (EmptyQueue(CQ))
         {error("队空"); return 0;}
     else {
         temp = (LQ->front)->next;    //指向队列首结点
         (LQ->front)->next = temp->next;     //修改头结点指针域指向新的首结点
         if (temp->next == NULL)
         LQ->rear = LQ->front;   //无首结点时，front和rear都指向头结点
         free(temp);
         return 1;
     }
}
```

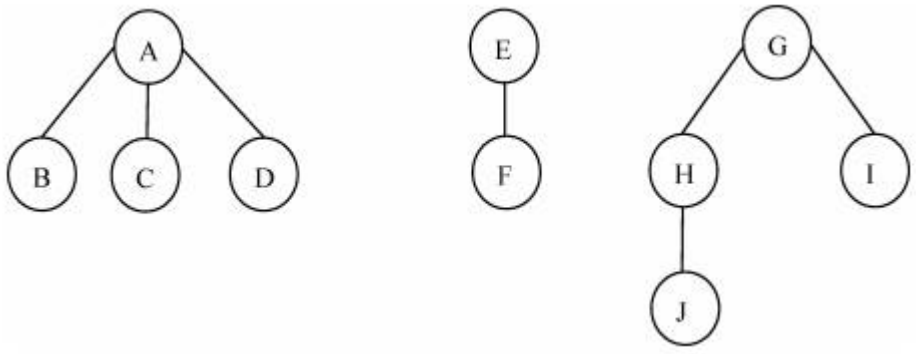
1）设指针变量 front 表示链队列的队头指针，指针变量 rear 表示链队列的队尾指针，指针变量 s 指向将要入队列的结点 X，则入队列的操作序列为（ `s->next=NULL; rear->next=s; rear=s;` ）
2）循环队列结构类型中含有三个域：data、front 和 rear，循环队列 SQ 为空的条件是 `SQ.rear==SQ.front`
3）链队列中，单链表的 头结点 的 next 域指向队列 首 结点
4）设以数组 Q[m]存放循环队列的元素，变量 rear 和 queuelen 分别表示循环队列中队尾元素的下标位置和元素的个数。则计算该队列中队头元素下标位置的公式是 `front = (rear − queuelen + 1 + m) % m`

3. 数组

1）对于 下三角矩阵中 的元素 a[i][j],(i>=j)， 在一维数组中的索引 为： `index = i(i+1)/2 + j` ；
2）假设一个 10X10 的 上三角矩阵 A 按照 列优先顺序压缩 存储在一维数组 B 中，则 B 数组的大小应为 `n(n+1)/2 = 10(11)/2 = 55` ；
3）对称矩阵有近一半元素可以通过其对称元素获得，因此可将含有 n^2 个元素的 对称矩阵压缩 存储到含有 `n(n+1)/2` 个元素的一维数组中；

第 4 章 树和二叉树

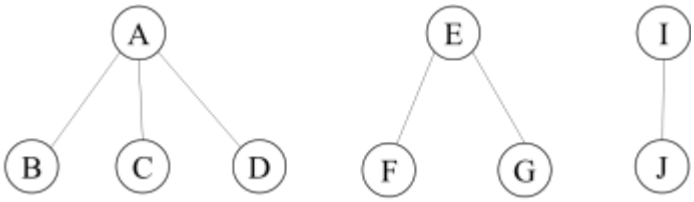
1. 设有一森林 F 如图所示，请分别写出先序遍历和中序遍历的序列。（6 分）

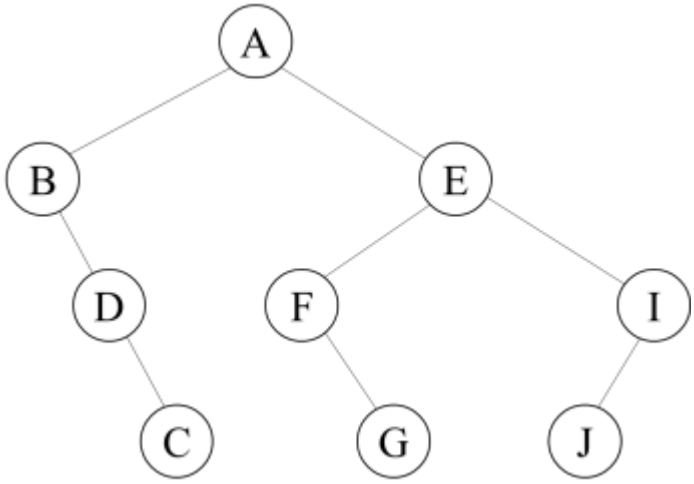


   先序序列为 ABCDEFGHJI；

   中序序列为 `BCDA` FE `JHIG`。 过程：① 中序遍历森林中第一棵树的根结点的子树森林；② 访问第一棵树的根结点；③ 中序遍历除去第一棵树之后剩余的树构成的森林。

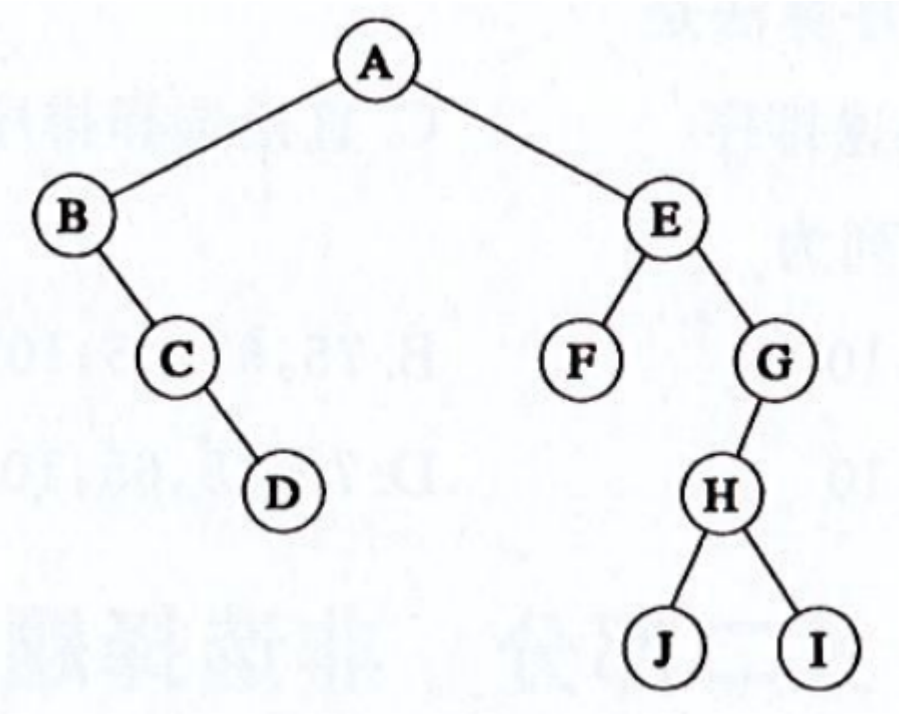
2. 将下图所示的 `森林转换成二叉树`。

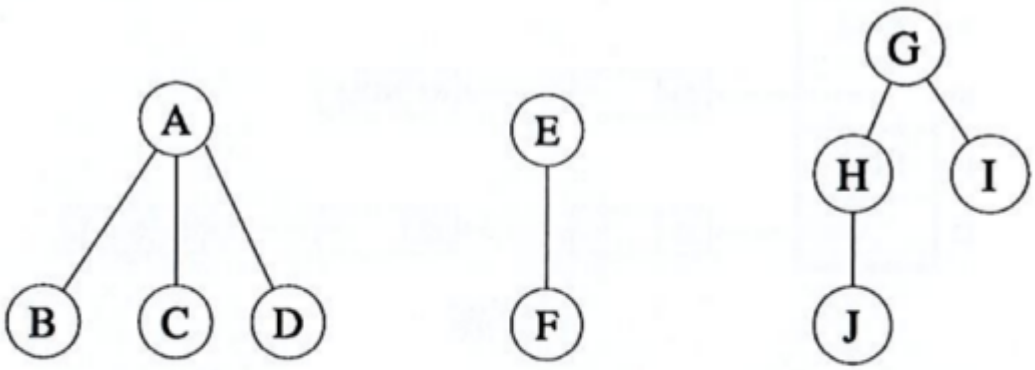




   1）第一颗子树根节点为二叉树根结点；2）将所有树转换为二叉树（兄弟之间加线，除了第一个孩子其他孩子的线去掉）；3）将后一棵树的根结点作为前一棵树的右孩子；

3. 已知二叉树如下图所示,请将该 `二叉树转换为对应的森林`。





树T1　　树T2　　树T3

1图　　2图　　3图

1）给右孩子去线（只给右孩子的右孩子去线）；2）将二叉树转化为树（将右孩子与根连线）；

4. 设某通信系统中一个待传输的文本有 6 个不同字符，它们的出现频率分别是 0.5,0.8,1.4,2.2,2.3,2.8，试设计哈夫曼编码。（6 分）

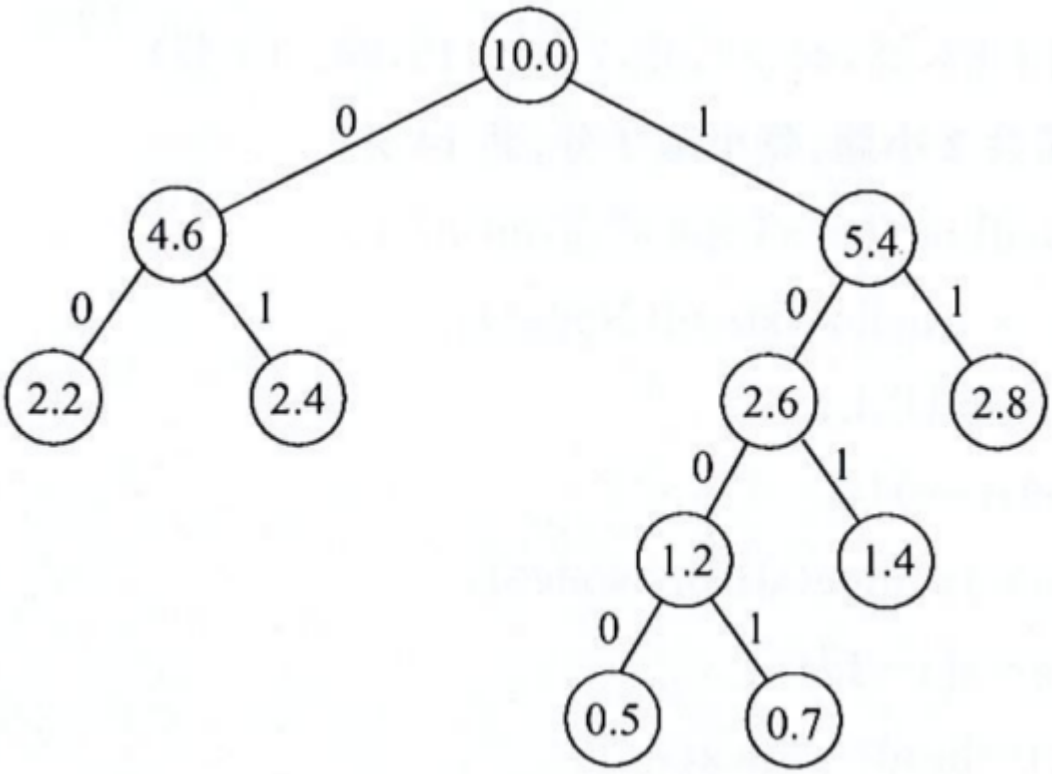
```
假设字符分别为 A,B,C,D,E,F 对应频率：0.5, 0.8, 1.4, 2.2, 2.3, 2.8。
(0.5,0.8,1.4,2.2,2.3,2.8) -> (0.5, 0.8) -> (1.3,1.4,2.2,2.3,2.8) 新节点 AB(1.3)
(1.3,1.4,2.2,2.3,2.8) -> (1.3, 1.4) -> (2.7,2.2,2.3,2.8) 新节点 ABC(2.7)
(2.7,2.2,2.3,2.8)->(2.2, 2.3)-> (2.7,4.5,2.8) 新节点 ABDE(2.7)
(2.7,4.5,2.8)->(2.7, 2.8)-> (4.5, 5.5) 新节点 ABDEF(5.5)
(4.5, 5.5) -> 10 根节点 ROOT(10.0)
```

```
          ROOT(10.0)
        /           \
DE(4.5)           ABCF(5.5)
/     \           /       \
```

```
        D(2.2)  E(2.3)   ABC(2.7)   F(2.8)
                           /    \
                        AB(1.3)   C(1.4)
                        /    \
                    A(0.5)  B(0.8)
```

生成编码：从根节点出发，为左分支分配 “0”，为右分支分配 “1”，得到每个字符的哈夫曼编码。

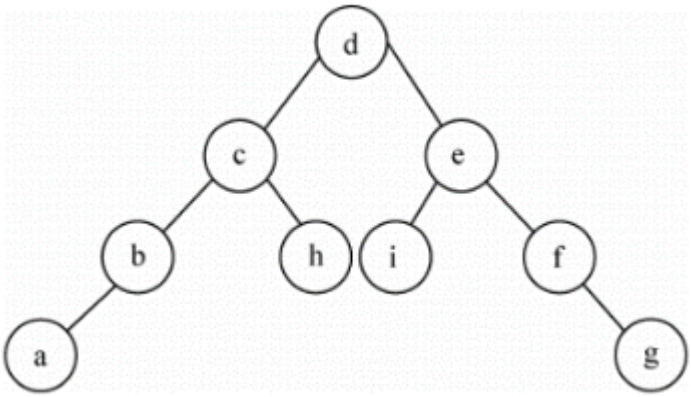
5. 设某通信系统中一个待传输的文本有 6 个不同字符，它们的出现频率分别是 0.5，0.7，1.4，2.2，2.4，2.8，试画出哈夫曼树，并给出每个字符的哈夫曼编码。（要求任一结点的左孩子权值小于右孩子）



哈夫曼树

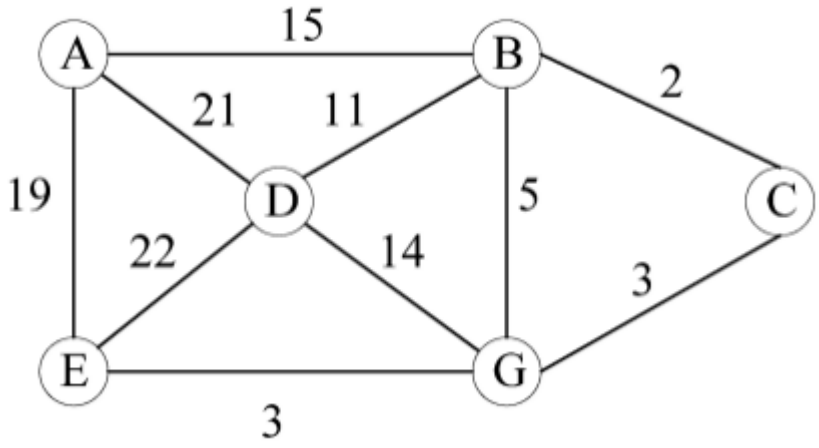
出现频率为 0.5 的字符编码为 1000
出现频率为 0.7 的字符编码为 1001
出现频率为 1.4 的字符编码为 101
出现频率为 2.2 的字符编码为 00
出现频率为 2.4 的字符编码为 01
出现频率为 2.8 的字符编码为 11

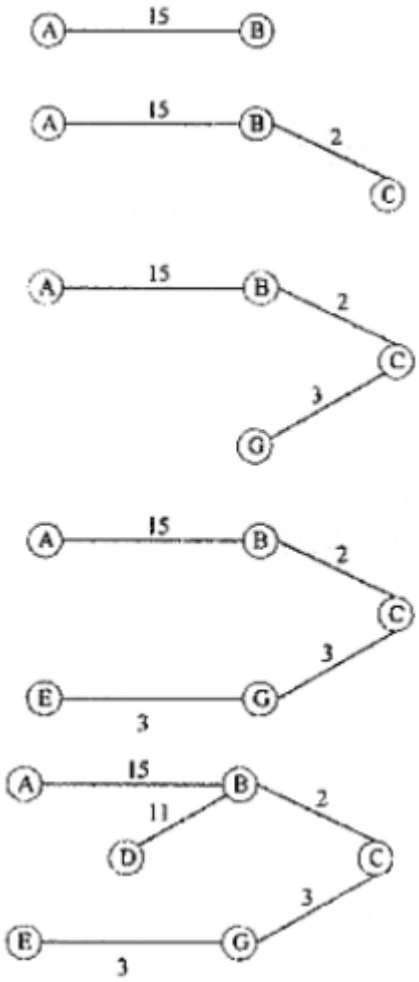
6. 设序列{d c b a h e i f g}和{a b c h d i e f g}分别是一棵二叉树的先序序列和中序序列，请画出该二叉树。



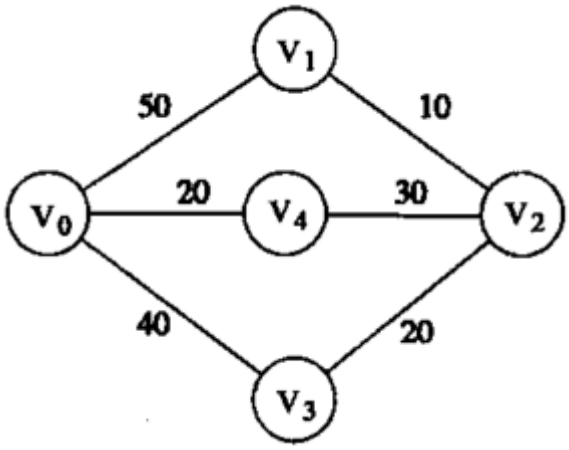
第 5 章 图

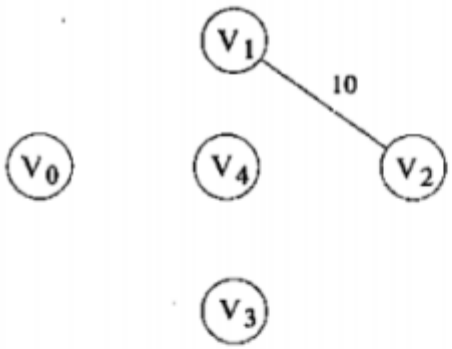
1. 已知如下图所示的无向带权图，请从结点 A 出发，用 `普里姆（Prim）算法求其最小生成树` ，并画出过程示意图。



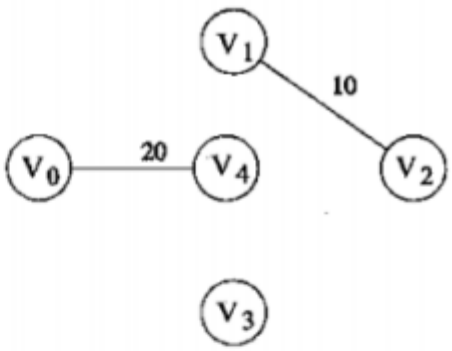


2. 用 克鲁斯卡尔（Kruskal） 方法 求下图所示的图的最小生成树。

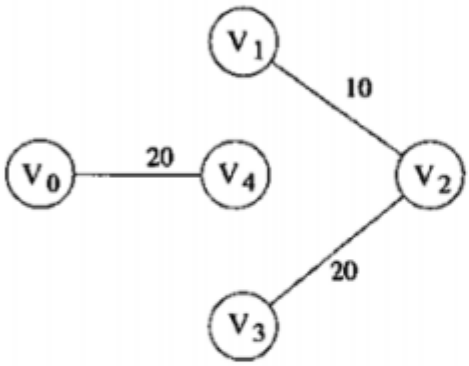




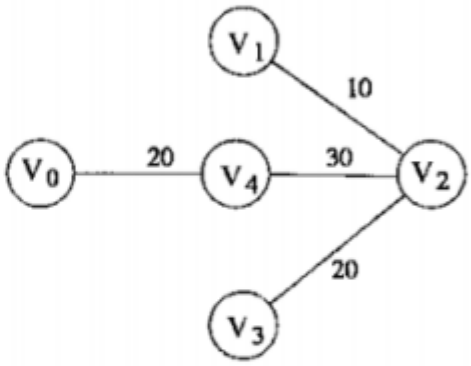
步骤 1



步骤 2

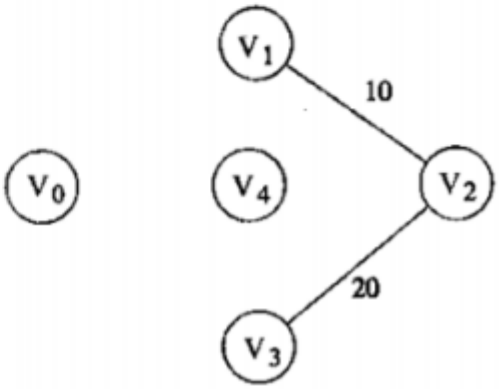


步骤 3

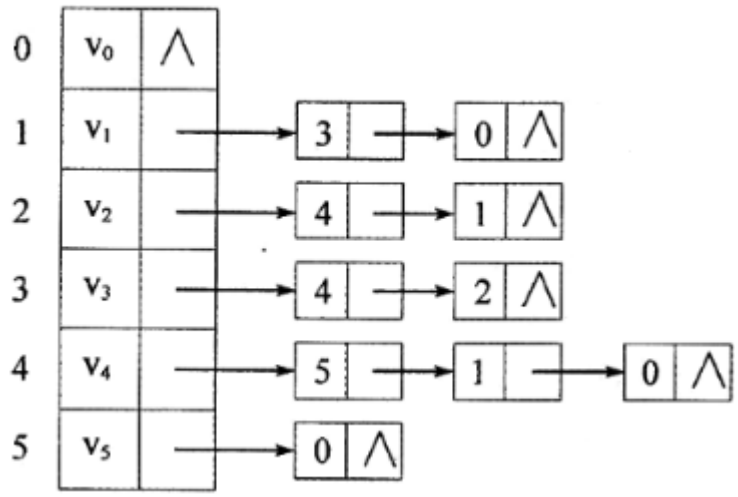


步骤 4

其中步骤 2 也可以是：

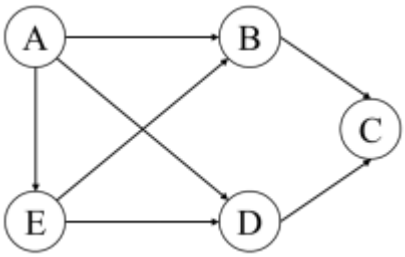


3. 设 有向图的邻接表 表示如下图所示，请给出每个顶点的入度和出度。

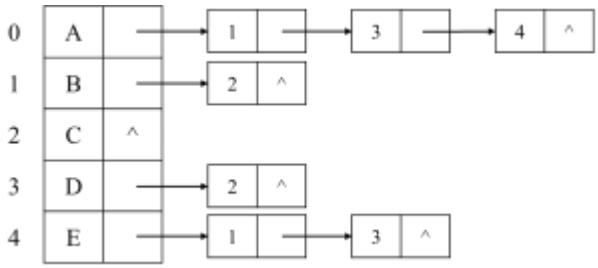


顶点	v0	v1	v2	v3	v4	v5
入度	3	2	1	1	2	1
出度	0	2	2	2	3	1

4. 下图所示为一有向图，试给出该 有向图的邻接表表示 及对该图进行拓扑排序的各种可能的拓扑序列。

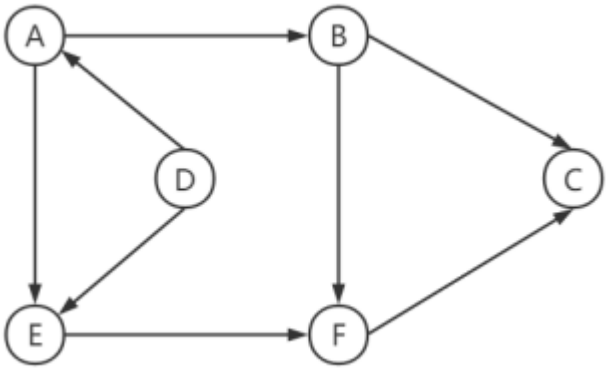


（1）该有向图的邻接表为：



（2）各种可能的拓扑排序序列为：AEDBC；AEBDC。

5. 写出如下图所示的 有向图邻接矩阵表示 和所有拓扑排序序列。

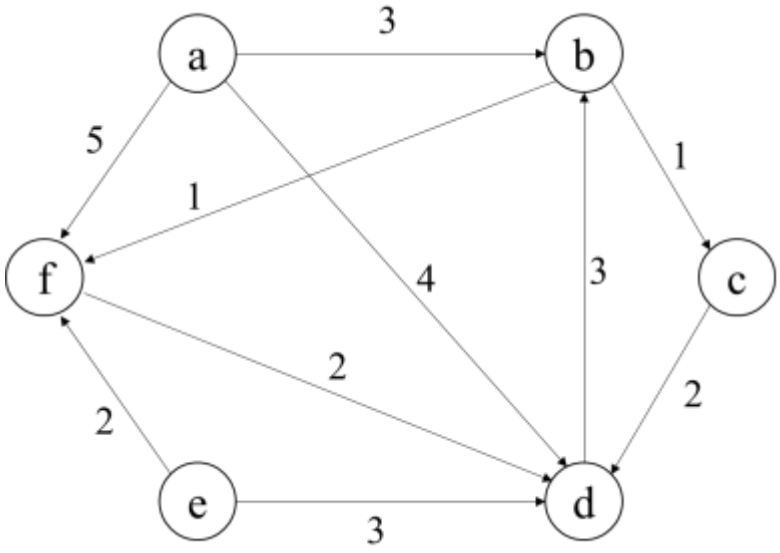


（1）邻接矩阵表示

	A	B	C	D	E	F
A	0	1	0	0	1	0
B	0	0	1	0	0	1
C	0	0	0	0	0	0
D	1	0	0	0	1	0
E	0	0	0	0	0	1
F	0	0	1	0	0	0

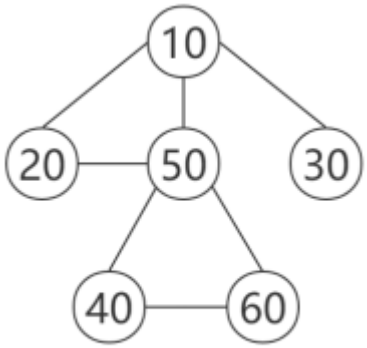
（2）拓扑排序序列：DAEBFC；DABEFC。

6. 写出下图所示的 有向带权图的邻接矩阵 。

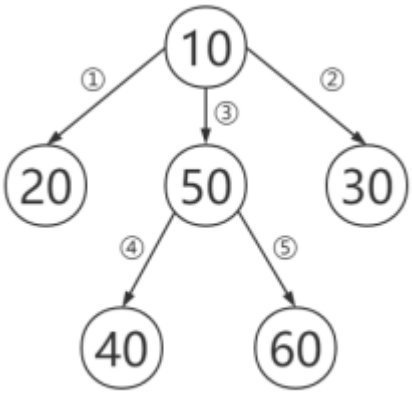


	a	b	c	d	e	f
a	∞	3	∞	4	∞	5
b	∞	∞	1	∞	∞	1
c	∞	∞	∞	2	∞	∞
d	∞	3	∞	∞	∞	∞
e	∞	∞	∞	3	∞	2
f	∞	∞	∞	2	∞	∞

7. 如题 30 图所示的图结构，请写出以 10 为源点的 `广度优先搜索` 得到的顶点访问序列，并画出搜索过程图。（同等情况下，值小的结点优先访问）



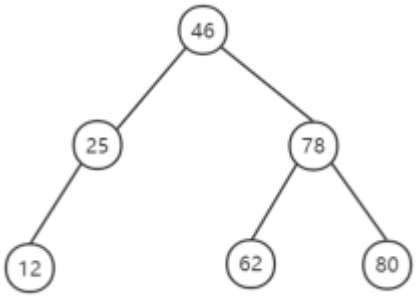
题30图



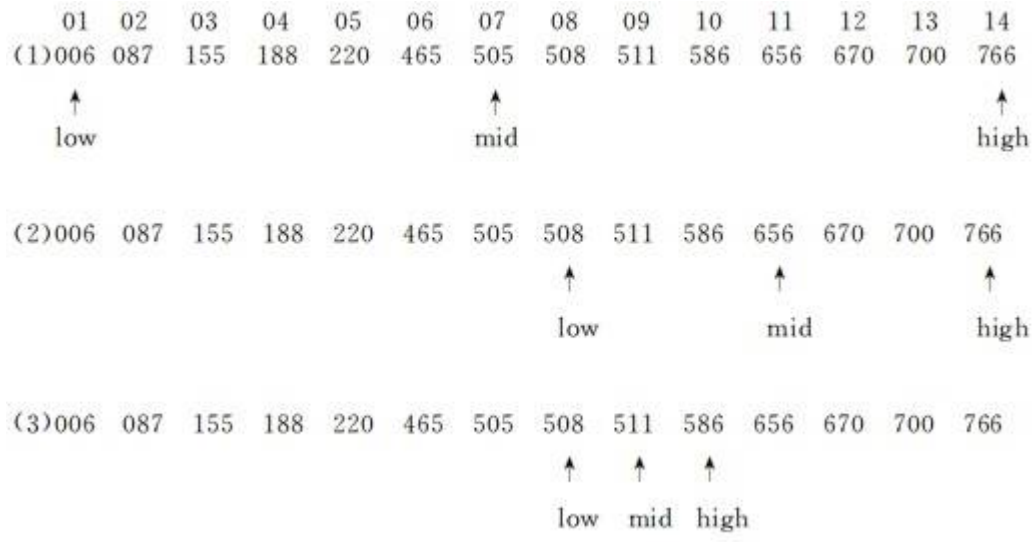
答30图

第 6 章 查找

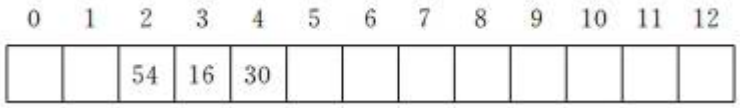
1. 给定数据序列{46,25,78,62,12,80}，试按元素在序列中的次序将它们依次插入一棵初始为空的二叉排序树，`画出插入完成后的二叉排序树`。



2. （6 分）给定有序表 D={006，087，155，188，220，465，505，508，511，586，656，670，700，766}，用 `二分查找法` 在 D 中查找 511，试给出查找过程。



3. 如图所示长度为 13 的散列表，其 散列函数 为 H（key）=key mod 13，在表中已填入键值分别为 16，30，54 的元素。



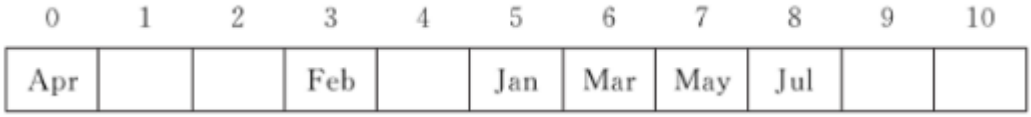
1）（3 分）现要插入键值为 29 的元素，应用线性探测法， 计算填入散列表中单元的序号 。（要求给出求解过程）

散列函数求出其散列地址为 3，在地址 3 上面已有元素 16，发生冲突。应用线性探测法，得到下一个地址为 d+1=4，仍冲突，则再求下一个地址 d+2=5，这个位置上没有元素，将元素填入散列表中序号为 5 的单元。

2）（3 分）线性探测法中，如何减少堆积的机会？

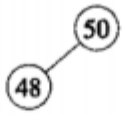
应设法使后继散列地址尽量均匀地分散在整个散列表中。

4. 给定表(Jan, Feb, Mar, Apr, May, Jul)。散列表的地址空间为 0~10，设散列函数 H(x)=⌊i/2⌋，其中 i 为键值中第一个字母在英语字母表中的序号，要求 画出以线性探测法解决冲突的散列表 。

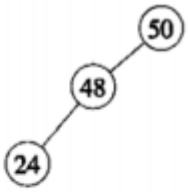


5. 根据二叉排序树的插入算法，从空树开始建立键值序列{50，48，24，55，53，90}的 二叉排序树，要求给出建立过程 。

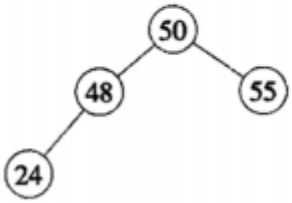
1图



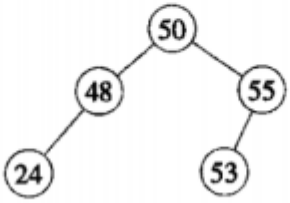
2图



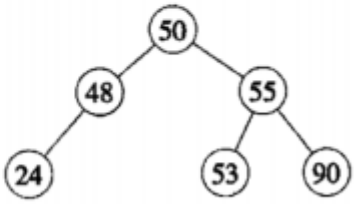
3图



4图



5图



6图