**数据结构部分课后习题答案**

# 第一章

1.1

数据的逻辑结构是从具体问题中抽象出来的数学模型，体现了事物的组成和事物之间的逻辑关系。

数据的存储结构主要用来解决各种逻辑结构在计算机中物理存储表示的问题。

1.2

事前分析和事后统计

事前分析：

优点，程序不必运行，所得结果只依赖于算法本身

缺点，不够精确

事后统计：

优点，精确

缺点，必须运行程序，所得结果依赖于硬件、环境等因素

1.3

|  |
| --- |
| void func(int n)  {  int i = 1, k = 100;  while(i < n)  {  k++; i+=2;  }  } |

考虑赋值、运算操作执行的次数

第3行赋值2次

第6行赋值执行n次，加法执行n次

所以，总共2n+2次操作，算法复杂度为O(n)

1.4

y= y + i \* j 执行次数：

1.5

# 第二章

2.9

内存中一片连续空间（不妨假设地址从1到m）提供给两个栈S1和S2使用，怎样分配这部分存储空间，使得对任一个栈，仅当这部分空间全满时才发生上溢。

　答：S1和S2共享内存中一片连续空间（地址1到m），可以将S1和S2的栈底设在两端，两栈顶向共享空间的中心延伸，仅当两栈顶指针相邻（两栈顶指针值之差的绝对值等于1）时，判断为栈满，当一个栈顶指针为0，另一个栈顶指针m+1时为两栈均空。

2.10

线性表是数据项组成的一种有限且有序的序列，各元素之间呈线性关系。从逻辑结构来说，栈和队列与线性表相同，都是典型的线性结构。与线性表不同的是，栈和队列的操作特殊，受到一定的限制，仅允许在线性表的一端或两端进行。栈是限定仅在一端进行插入删除的线性表，无论插入、删除还是读取都在一端进行，按后进先出的原则。队列的元素只能从一端插入，从另一端删除，按先进先出的原则进行数据的存取。

2.11

共有132种合法序列。

235641序列可以。

154623序列不可以。

对于每一个数来说，必须进栈一次、出栈一次。我们把进栈设为状态‘1’，出栈设为状态‘0’。n个数的所有状态对应n个1和n个0组成的2n位二进制数。由于等待入栈的操作数按照1‥n的顺序排列、入栈的操作数b大于等于出栈的操作数a(a≤b)，因此输出序列的总数目=由左而右扫描由n个1和n个0组成的2n位二进制数，1的累计数不小于0的累计数的方案种数。

在2n位二进制数中填入n个1的方案数为c(2n,n),不填1的其余n位自动填0。从中减去不符合要求（由左而右扫描，0的累计数大于1的累计数）的方案数即为所求。

不符合要求的数的特征是由左而右扫描时，必然在某一奇数位2m+1位上首先出现m+1个0的累计数和m个1的累计数，此后的2(n-m)-1位上有n-m个 1和n-m-1个0。如若把后面这2(n-m)-1位上的0和1互换，使之成为n-m个0和n-m-1个1，结果得1个由n+1个0和n-1个1组成的2n位数，即一个不合要求的数对应于一个由n+1个0和n-1个1组成的排列。

反过来，任何一个由n+1个0和n-1个1组成的2n位二进制数，由于0的个数多2个，2n为偶数，故必在某一个奇数位上出现0的累计数超过1的累计数。同样在后面部分0和1互换，使之成为由n个0和n个1组成的2n位数，即n+1个0和n-1个1组成的2n位数必对应一个不符合要求的数。

因而不合要求的2n位数与n＋1个0，n－1个1组成的排列一一对应。

显然，不符合要求的方案数为c(2n,n+1)。由此得出 输出序列的总数目=c(2n,n)-c(2n,n+1)=1/(n+1)\*c(2n,n)

2.16

next数组值：

0,0,0,1,1,2,0,0,1,2,3,4,5,6,0,1,2

# 第三章

3.1

（1）n个结点可构造出多少种不同形态的二叉树？

解：

当n=1时，只有1个根节点，则只能组成1种形态的二叉树，令n个节点可组成的二叉树数量表示为f(n)，则f(1)=1;

当n=2时，1个根节点固定，还有n-1个节点，可以作为左子树，也可以作为右子树，即：f(2)=f(0)\*f(1)+f(1)\*f(0)=2，则能组成2种形态的二叉树。这里f(0)表示空，所以只能算一种形态，即f(0)=1;

当n=3时，1个根节点固定，还有n-1=2个节点，可以在左子树或右子树，即：f(3)=f(0)\*f(2)+f(1)\*f(1)+f(2)\*f(0)=5，则能组成5种形态的二叉树。

以此类推，当n>=2时，可组成的二叉树数量为f(n)=f(0)\*f(n-1)+f(1)\*f(n-2)+...+f(n-1)\*f(0)种。

即符合Catalan数的定义，可直接利用通项公式得出结果。

递归式：

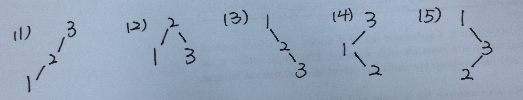
　　h(n)=h(n-1)\*(4\*n-2)/(n+1);

该递推关系的解为：

h(n)=C(2n,n)/(n+1) (n=1,2,3,...)

（2）若有3个数据1,2,3，输入它们构造出来的中序遍历结果都为1,2,3的不同二叉树有哪些？

解：有五种，如下：



3.2 树深度为6，17个叶子结点，度为1的节点为0

3.3 某二叉树有20个叶结点，有30个结点仅有一个孩子，求该二叉树的总结点数是多少？

解：设二叉树中度为0、1、2的结点数分别为n0、n1 、n2。

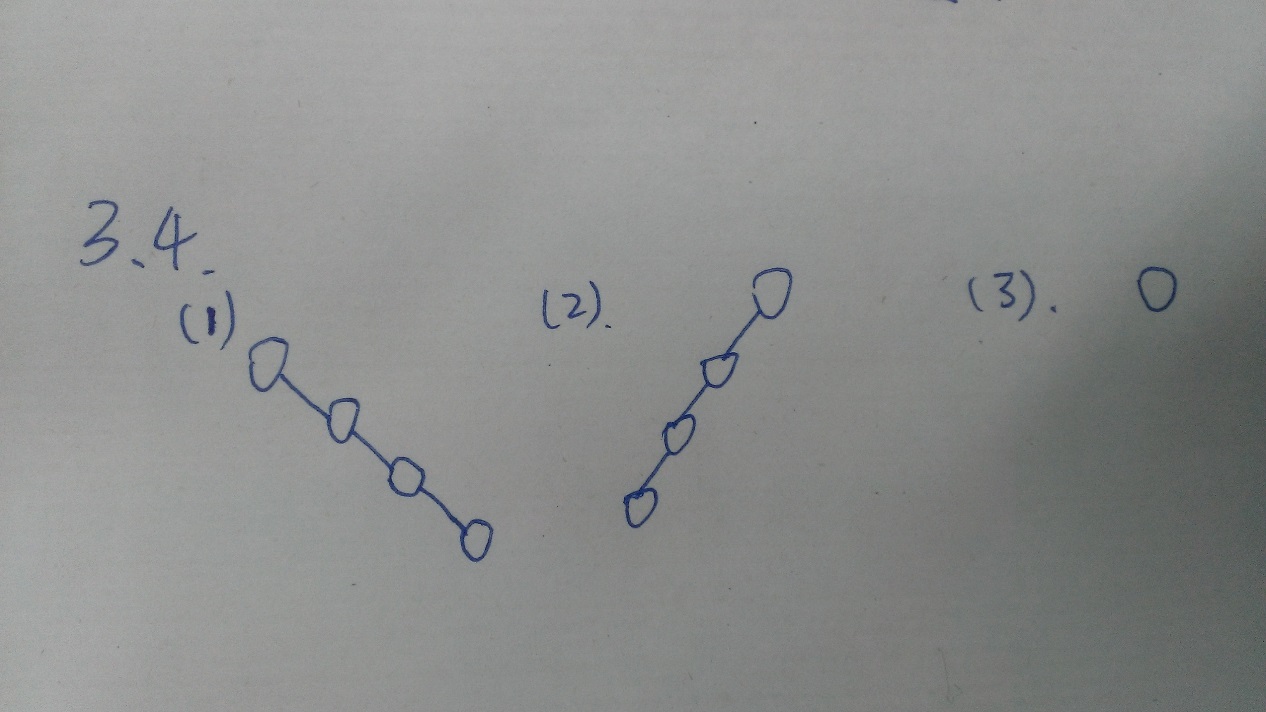
由题可知：n0=20, n1 =30。

由性质：任何一棵二叉树，度为0的结点比度为2的结点多一个，可知

n2= n0-1=20-1=19，即度为2 的结点个数为19个。

因此总结点数n= n0+n1 +n2=20+30+19=69个。

3.4

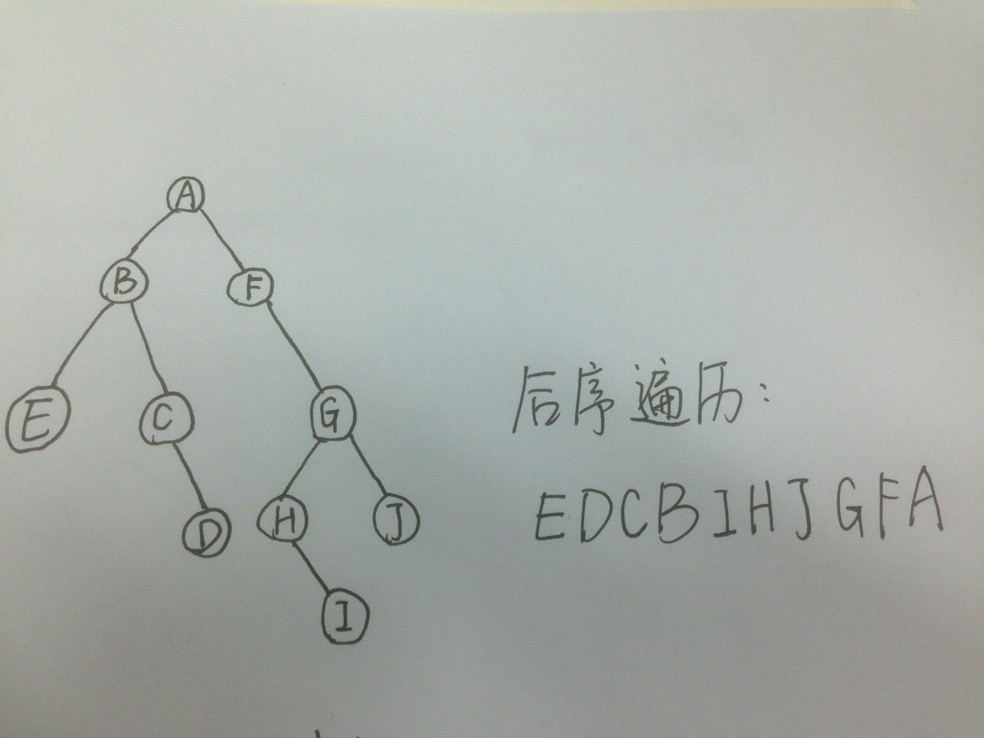


3.7 在中序线索二叉树中如何查找给定结点的前序后继,后序后继

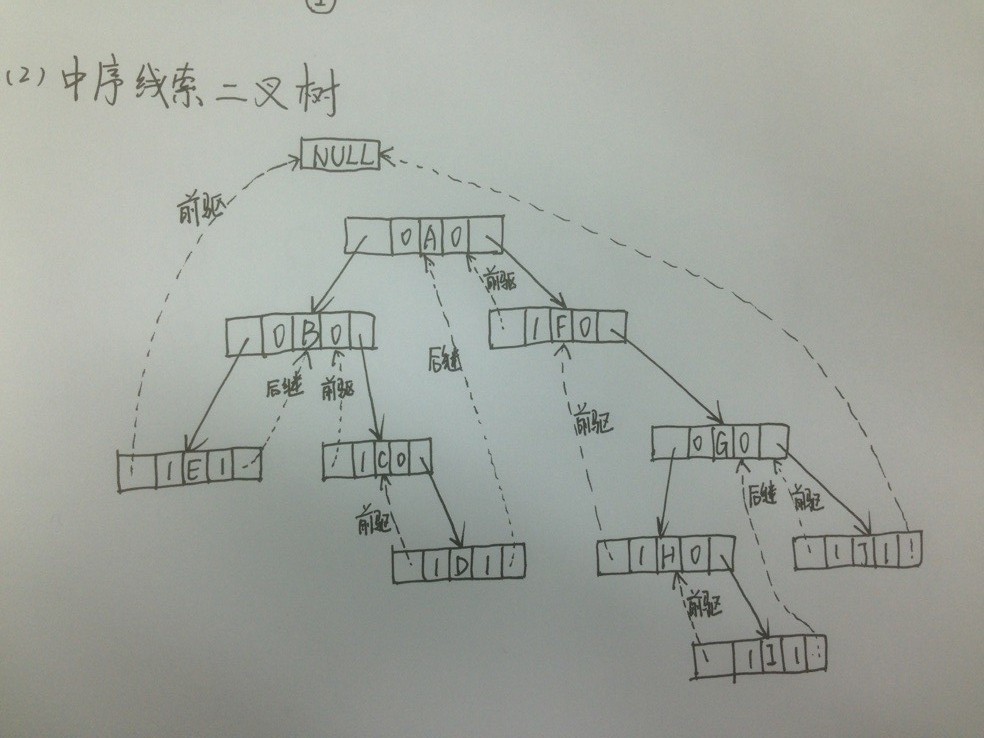
前序后继：如果节点的ltag==0，那么后继是节点的左孩子，否则，如果ltag==1&&rtag==0，后继是右孩子，如果ltag==1&&rtag==1，那么找到节点的父节点p，如果该节点是p的左孩子，且p->rtag==0，那么p的右孩子是节点的后继，如果p->rtag==1，那么q=p->rchild，q是节点p在中序时的后继，如果q->rtag==0，q的右孩子是后继，否则q=q->rchild,直到找到q->rtag==0的节点或者q==null为止，q==null说明所求节点没有后继。

后序后继：先根据中序全线索二叉树的性质找出p的父节点r：  
1）如果r->RightChild ！= p则对r的右子树进行后序遍历后访问的第一个节点就是p在后序序列中的后继；如果没有右子树，p在后序遍历中的后继就是r  
2）如果r->RightChild == p 则r就是p在后序序列中的后继。

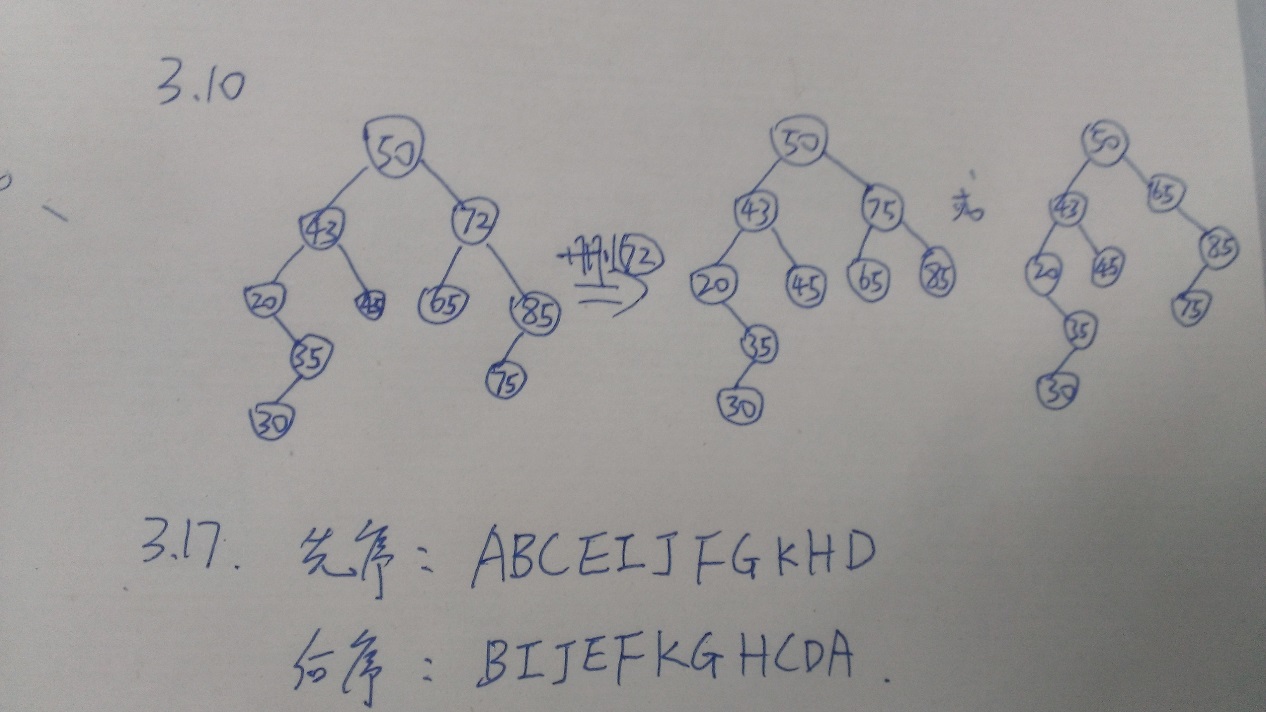
3.9

(1)

(2)



3.10



3.11

解答：高度为h的AVL树，最少节点数为：

当节点数为n时，根据上式可求得，数的最大高度为:

其中

最小高度为:

注：以上最少节点数可以利用归纳法可以得到，如下规律：

当h=1，N(1)=1

当h=2，N(2)=2

当h=3，N(3)=4

当h=4，N(4)=7

当h=5，N(5)=12

……

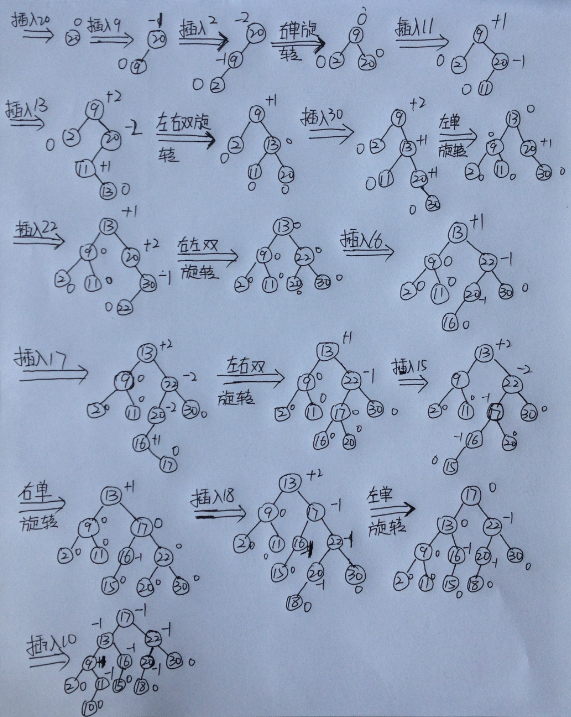
归纳可以发现类似于斐波那契数列的规律：

其中h>2。利用特征根求数列的方法，可以求得结果。

3.12 若关键字的输入序列为20,9,2,11,13,30,22,16,17,15,18,10。

（1）试从空树开始顺序输入各关键字建立平衡二叉树。画出每次插入时二叉树的形态，若需要平衡化旋转则做旋转并注明旋转类型；

解：插入过程及旋转如图所示：



（2）计算该平衡二叉搜索树在等概率下的查找成功的平均查找长度；

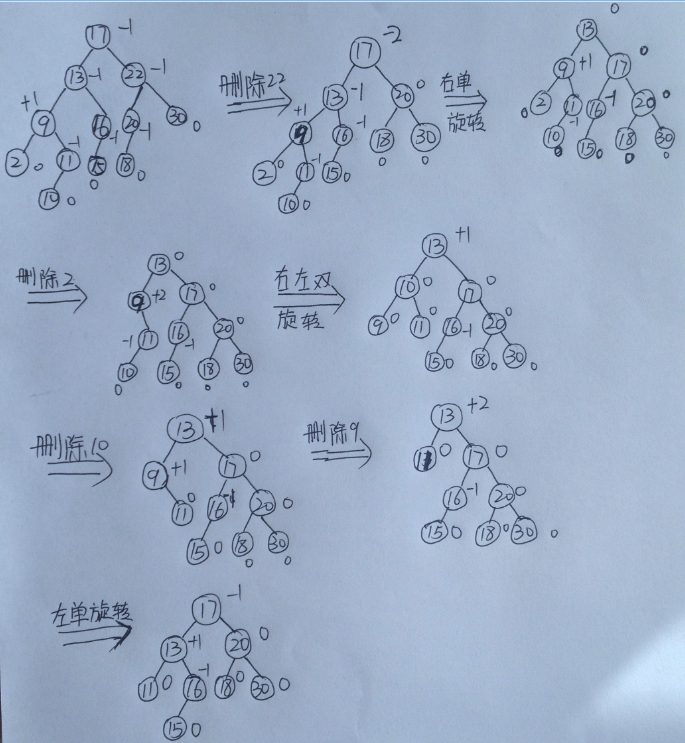
解：12个结点在等概率查找的情况下，每个结点被查找的概率为。

由上题所得最后的结果可知：查找长度为0的结点个数为1，查找长度为1的结点个数为2，查找长度为2的结点个数为4，查找长度为3的结点个数为4，查找长度为4的结点个数为1。因此：

查找成功的平均查找长度=

（3）基于上面的建树的结果，画出从树中删除22，删除2，删除10与9后树的形态和旋转类型。

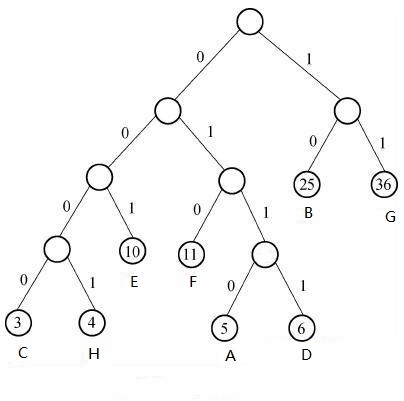
解：删除后的结果和旋转如下：



3.13



3.15、假定用于通信的电文仅由8个字母A，B，C，D，E，F，G，H组成,各字母在电文中出现的频率分别为5,25,3,6,10,11,36,4。试为这8个字母设计不等长Huffman编码,并给出该电文的总码数。



A:0110 B:10 C:0000 D:0111 E:001 F:010 G:11 H:0001

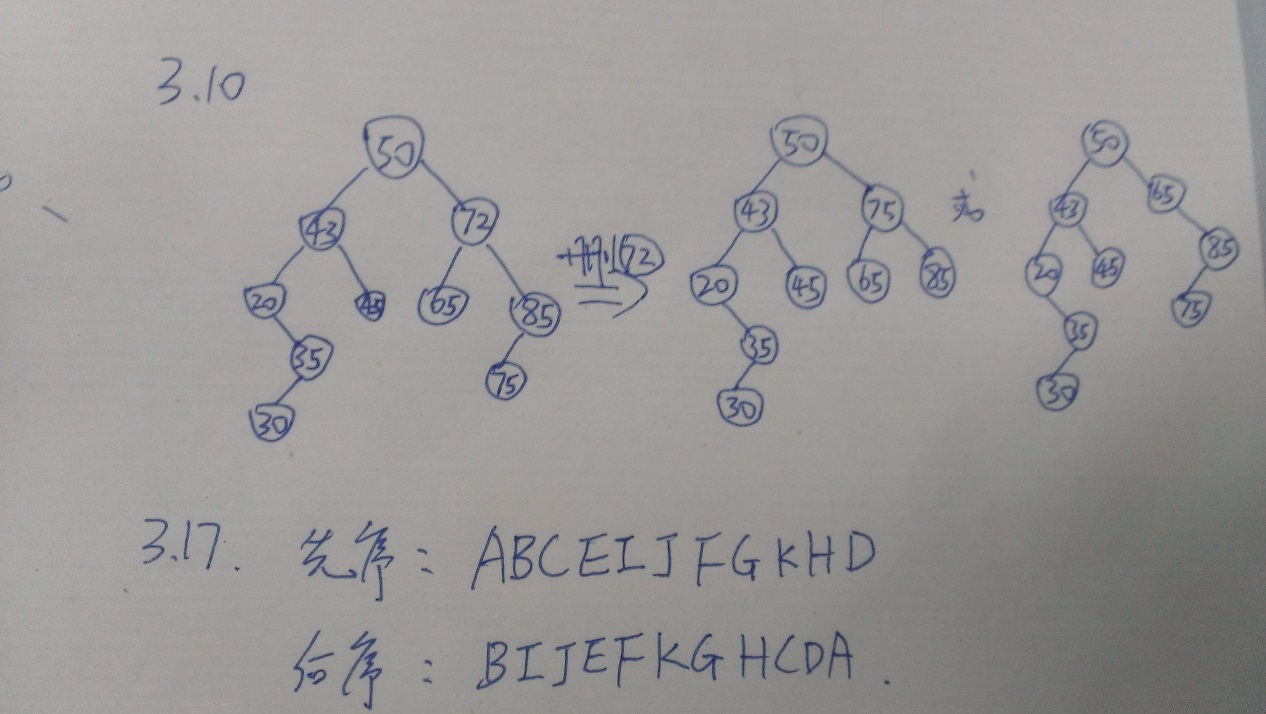
对应字母的码数加和为4+2+4+4+3+3+2+4=26。

3.16

解答：高度最小为2，n-1个叶结点，1个分支结点。

高度最大为n，1个叶结点，n-1个分支结点

3.17



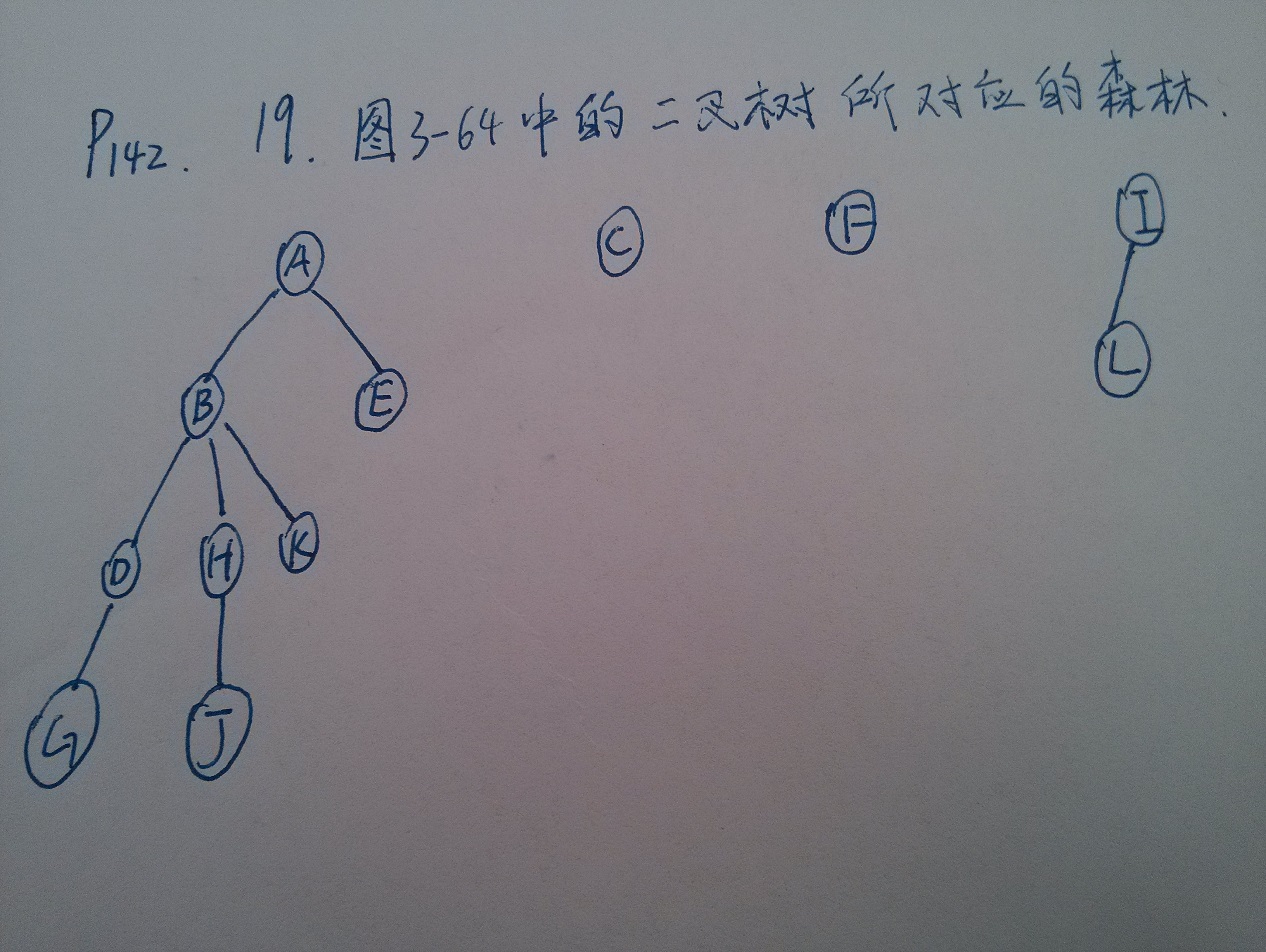
3.18

先根序列：ABCDEF GHIJK LMNPQRO

后根序列：BDEFCA IJKHG MPRQNOL

层次序列：ABCDRF GHIJK LMNOPQR

3.19

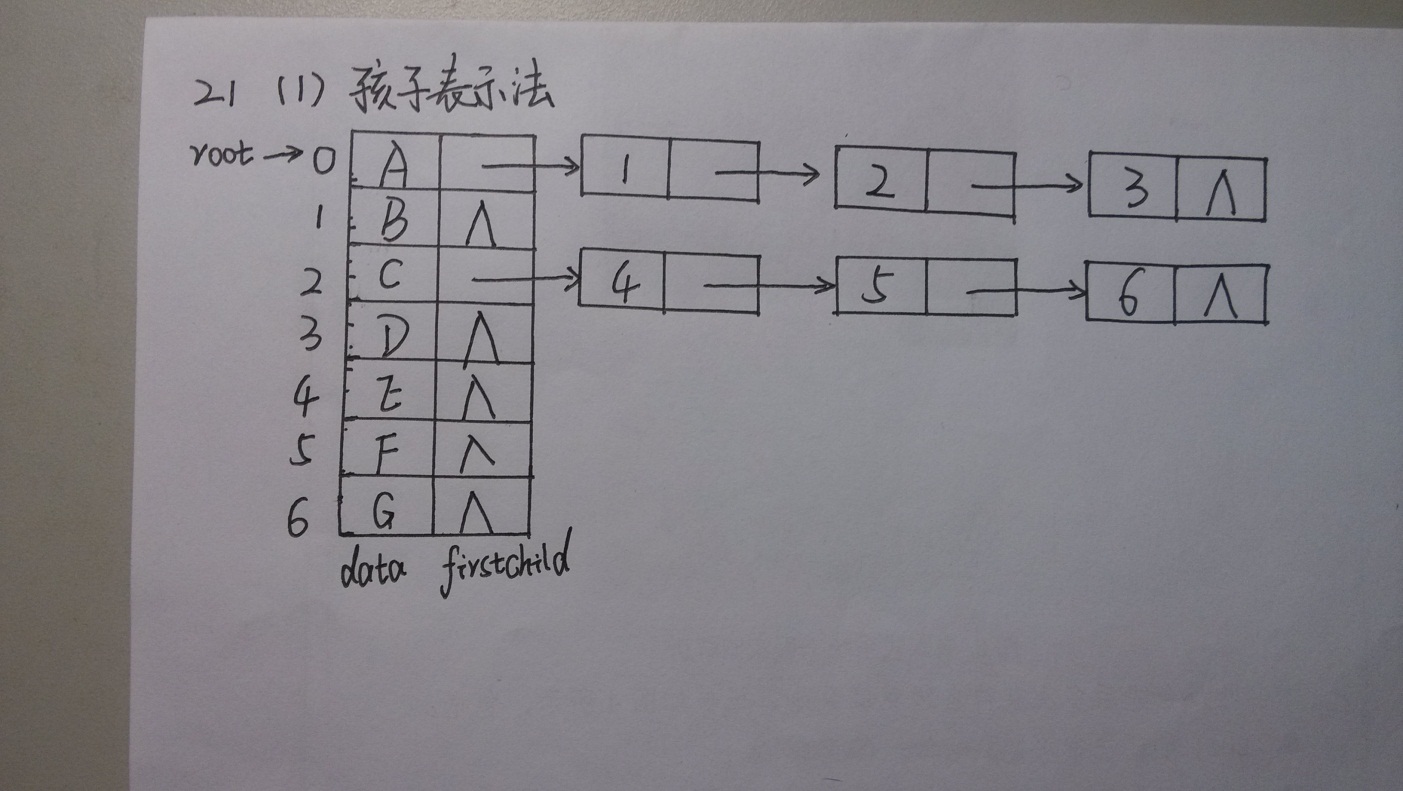


3.20

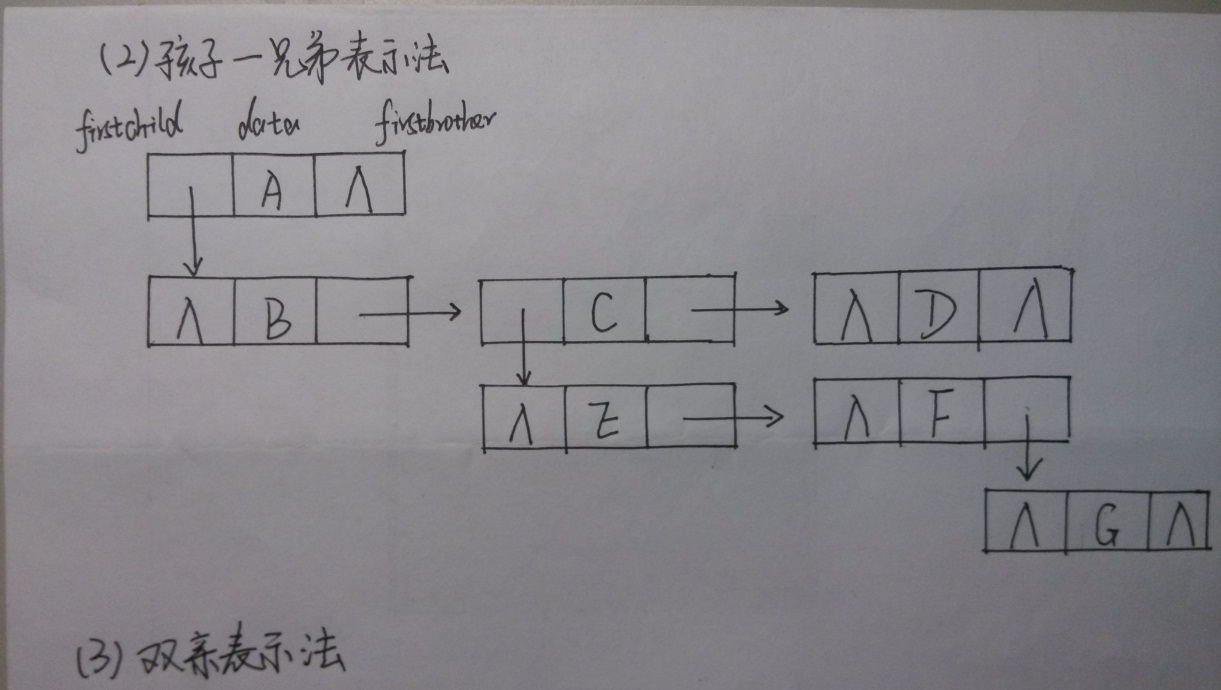


3.21

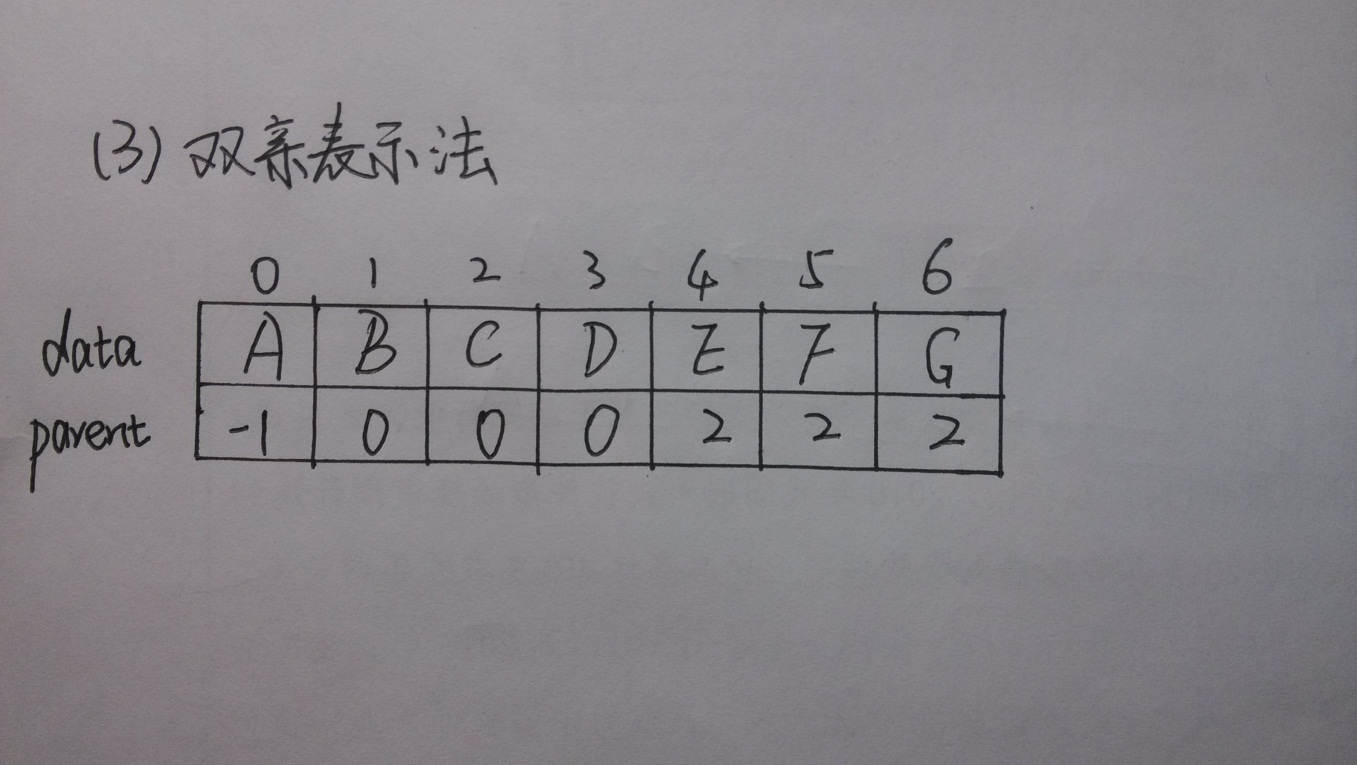
（1）孩子表示法：



（2）孩子—兄弟表示法：



（3）双亲表示法：



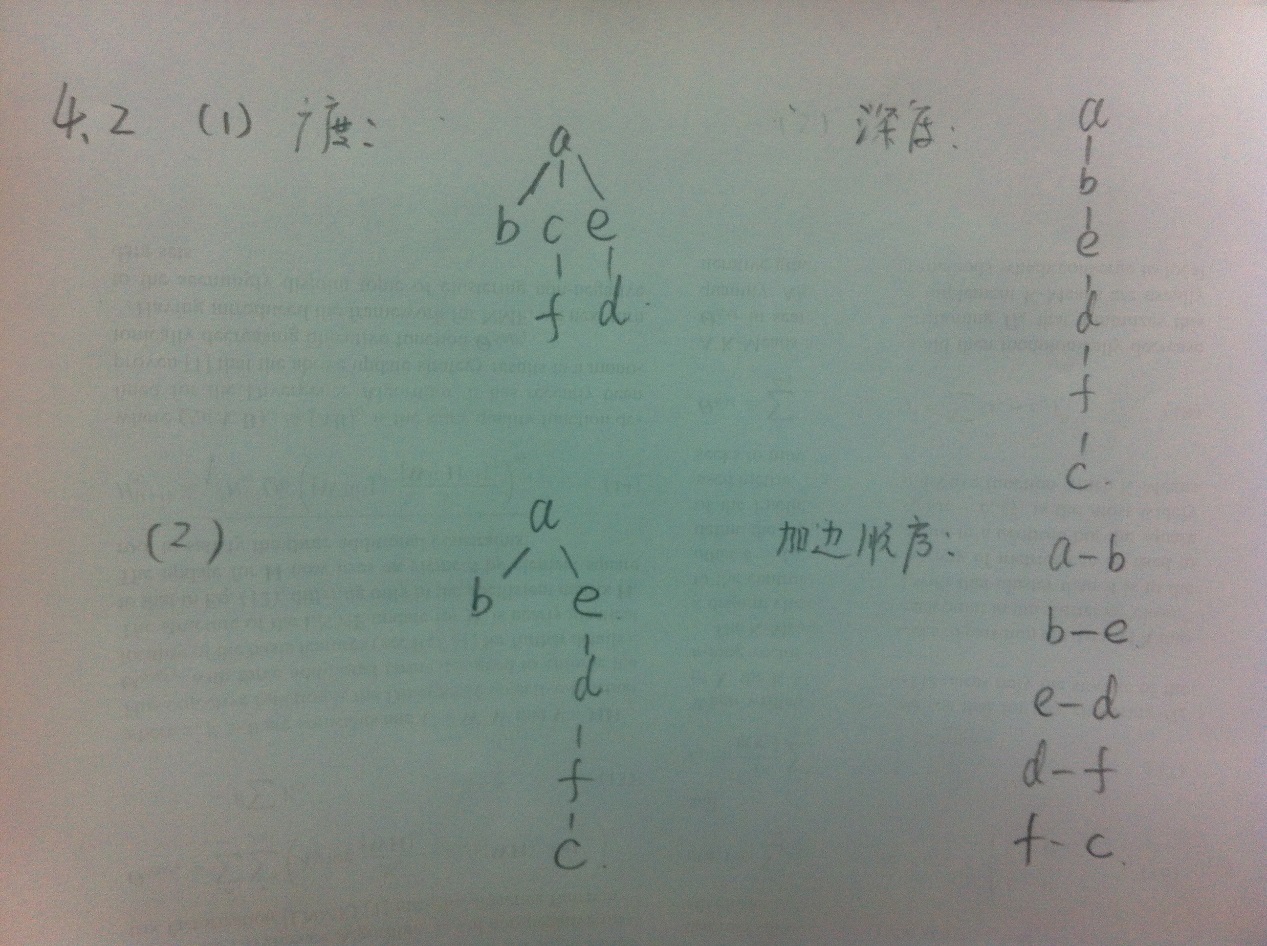
# 第四章

4.1

广度优先生成树(黑体加粗边)：

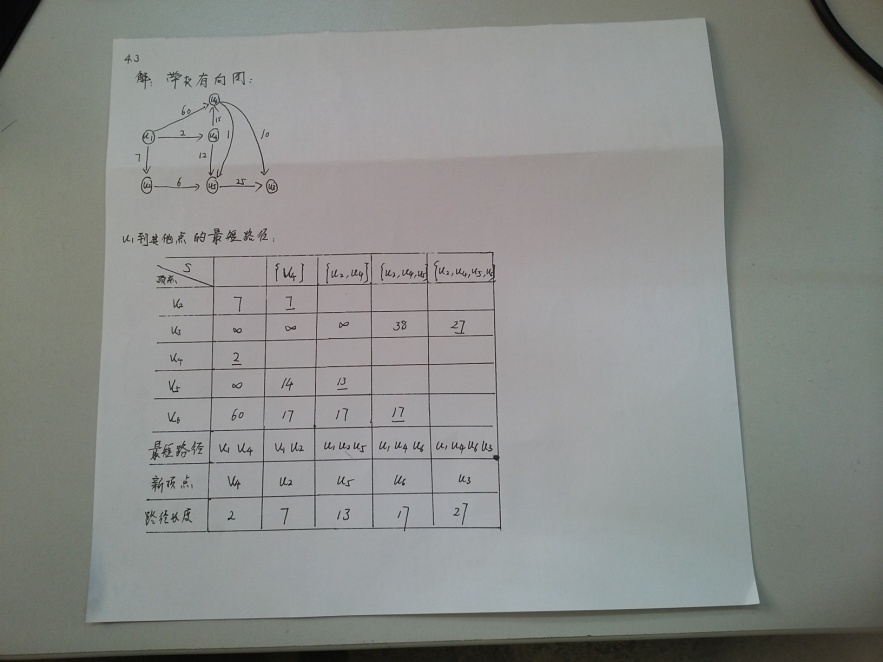
深度拓扑排序序列：v0-v2-v3-v1-v4

4.2



4.3、 如图所示为一个有6个顶点{u1,u2,u3,u4,u5,u6}的带权有向图的邻接矩阵。

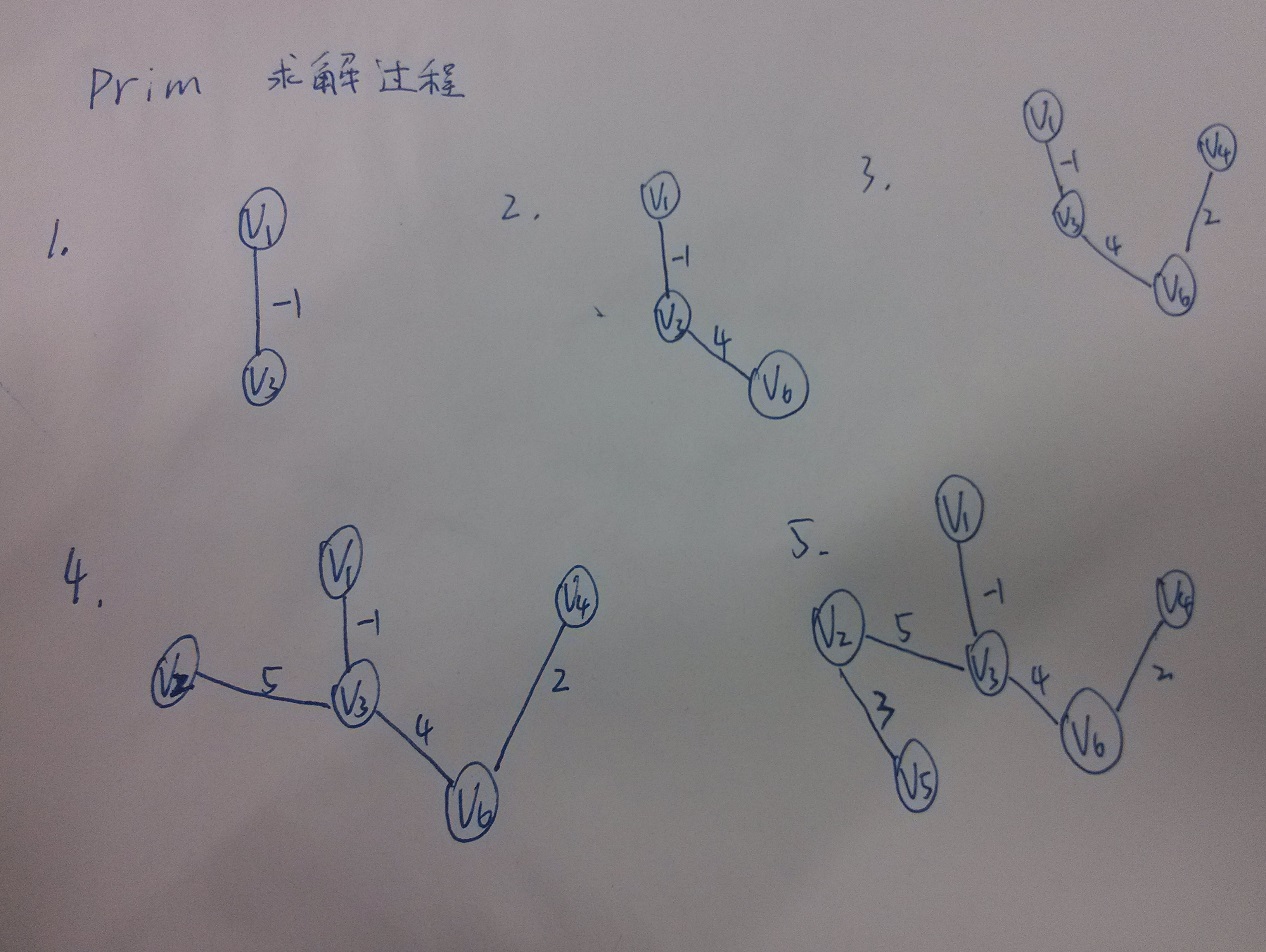
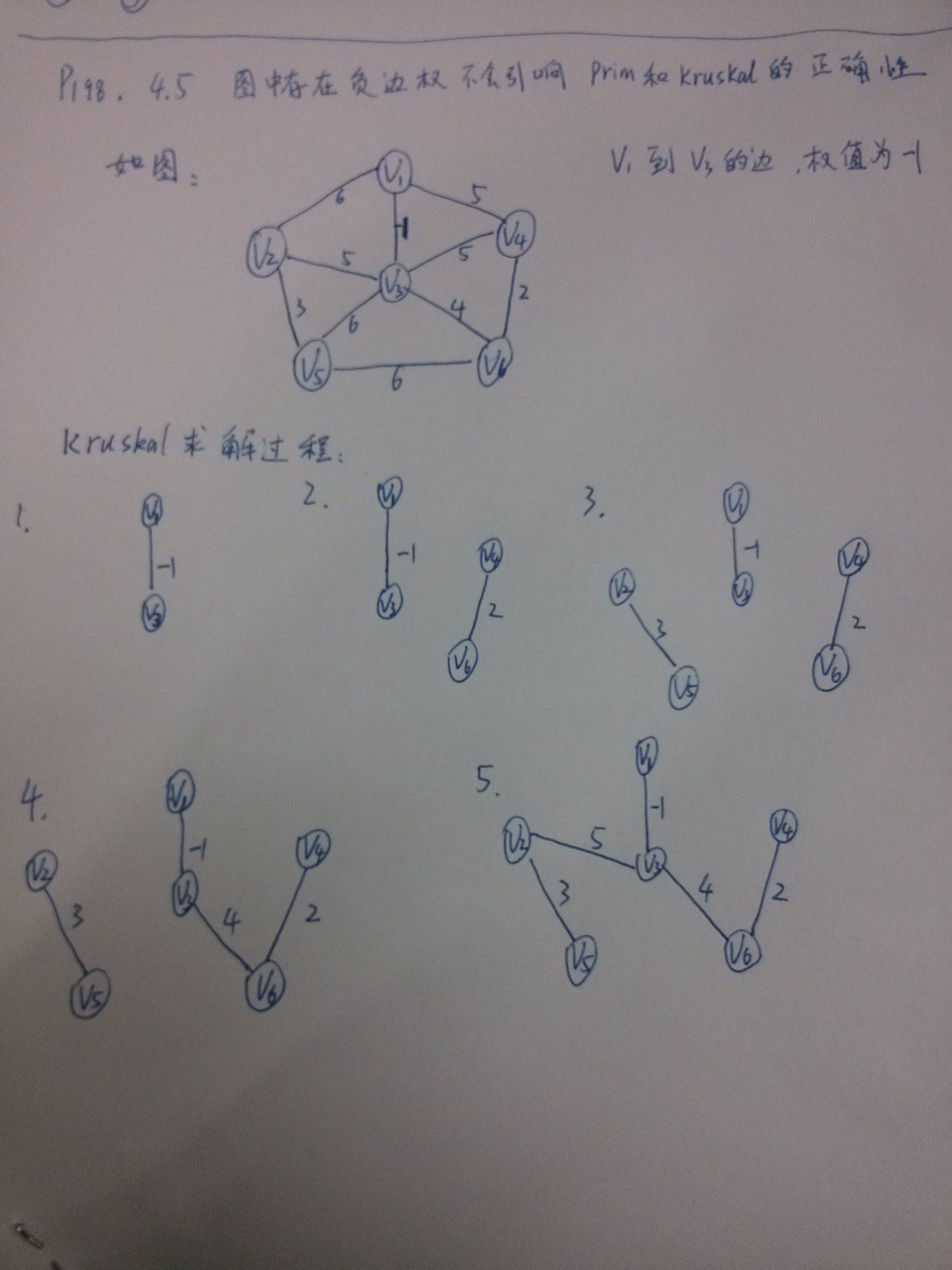
根据此邻接矩阵画出相应的带权有向图，利用dijkstra算法求第一个顶点u1到其余各顶点的最短路径，并给出计算过程。



4.4证明在图中边权为负时Dijkstra算法不能正确运行

若允许边上带有负权值，有可能出现当与S（已求得最短路径的顶点集，归入S内的结点的最短路径不再变更）内某点（记为a）以负边相连的点（记为b）确定其最短路径时，它的最短路径长度加上这条负边的权值结果小于a原先确定的最短路径长度，而此时a在Dijkstra算法下是无法更新的。

4.5

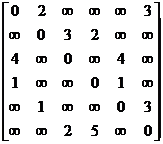
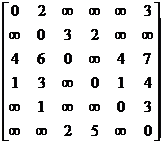
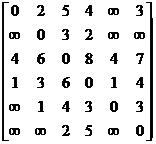
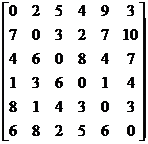
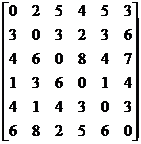
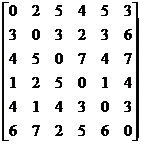
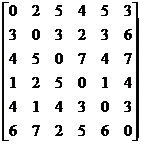


4.6 Dijkstra算法如何应用到无向图？

答：Dijkstra算法通常是运用在带非负权值的有向图中，但是无向图其实就是两点之间两条有向边权值相同的特殊的有向图，这样就能将Dijkstra算法运用到无向图中。

4.7

用FLOYD算法求出任意两顶点的最短路径（如图A（6）所示）。

A(0)= A(1)= A(2)= A(3)= A(4)= A(5)= A(6)=

V1到V2、V3、V4、V5、V6往返路径长度分别为5，9，5，9，9，最长为9，总的往返路程为37

同理V2到V1、V3、V4、V5、V6分别为5，8，4，4，13，最长为13，总和34

V3对应分别为9，8，12，8，9，最长为12，总和为46

V4对应分别为5，4，12，4，9，最长为12，总和为34

V5对应分别为9，4，8，4，9，最长为9，总和为34

V6对应分别为9，13，9，9，9，最长为13，总和为49

题目要求娱乐中心“距其它各结点的最长往返路程最短”，结点V1， V5最长往返路径最短都是9。按“相同条件下总的往返路径越短越好”，选顶点V5，总的往返路径是34。

4.8



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **i** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **w** |
| **最早** | 0 | 1 | 6 | 3 | 4 | 24 | 13 | 39 | 22 | 52 |
| **最晚** | 0 | 29 | 24 | 3 | 7 | 31 | 13 | 39 | 22 | 52 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **i** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **w** |
| **i** | - | a1 | a2 | a3 | a4 |  |  |  |  |  |
| **a** |  |  |  |  |  | a5 |  |  |  |  |
| **b** |  |  |  |  |  | a6 |  |  |  | a7 |
| **c** |  |  |  |  |  |  | a8 | a9 |  |  |
| **d** |  |  |  |  |  |  | a10 |  |  |  |
| **e** |  |  |  |  |  |  |  | a11 |  |  |
| **f** |  |  |  |  |  | a12 |  |  | a13 | a14 |
| **g** |  |  |  |  |  |  |  |  |  | a15 |
| **h** |  |  |  |  |  |  |  | a16 |  | a17 |
| **w** |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10 | a11 | a12 | a13 | a14 | a15 | a16 | a17 |
| 最早 | 0 | 0 | 0 | 0 | 1 | 6 | 6 | 3 | 3 | 4 | 24 | 13 | 13 | 13 | 39 | 22 | 22 |
| 最晚 | 28 | 18 | 0 | 3 | 29 | 24 | 31 | 3 | 34 | 7 | 31 | 20 | 13 | 36 | 39 | 22 | 40 |

关键路径为（i,c,f,h,g,w）