2019 年华南师范大学招收硕士研究生入学考试试题参考答案（仅供参考，如有疑问，欢迎一起讨论）

一、选择题（很基础）

1. 逻辑结构是（A）间的逻辑

A.数据元素

B.数据项

C.存储结构

D.物理结构

2. 循环队列（C）

A．不会产生上溢出

B．不会产生下溢出

C．不会产生假溢出

D．以上都不对

3. 数组 A[0..5][0..6]是按列优先存储的数组，则元素 A[4][6]之前有（D）个元素。

A.27

B.28

C.29

D.30

解析：数组 A[0..5][0..6]按列优先存储。简单的说就是以下的存储方式：

A[0,0]->A[1,0]->A[2,0]->A[3,0]->A[4,0]->A[5,0]

A[0,1]->A[1,1]->A[2,1]->A[3,1]->A[4,1]->A[5,1]

… … … …

A[0,6]->A[1,6]->A[2,6]->A[3,6]->A[4,6]->A[5,6]

因为 A[4,6]的列标为 4,那么它前面有 0~3 共有 4 列，每一列 6 个元素也就有 4*6=24 个元素。再加上列表为 4 的列中,含有 0~6(行标是 6) 共有 7 个元素 ，但是排除 A[4,6]本身，故就只有 6 个元素。最后 A[4,6]之前共有 30 个元素

4. 有 100 个元素，用折半查找成功次数最多的是（D）

A．20

B．50

C．40

D．7

解析：对折半查找，首先将待查记录所在范围缩小一半，然后逐步缩小，对 100 个元素的顺序表，第一次比较范围缩小到 50，第二次缩小到 25，第三次缩小到 13，第四次缩小到 7，第五次缩小到 4，第六次缩小到 2，第七次就可以找到查找的元素。

二、填空题

1. 评价哈希表的标准是__哈希表内数值分布是否均匀_____。

2. 单链表设置头结点的作用是 为了保证处理第一个节点和后面的节点的时候设计的算法相同,实现程序的高效性 。

3. N 阶下三角矩阵存储元素的个数是 n(n+1)/2 。

4. 树的后序遍历相当于二叉树的 中序 遍历。

5. 二叉树第 n 层的结点个数最多是 2^(n-1) 。

三、判断题（很基础）

1.最短路径一定是简单路径吗？对

解析：假设最短路径有环，则有以下三种情况：

1.正环，然后去掉正环会使最短路径更短，不符合

2.零环，零环对最短路径没有影响，可以去掉

3.负环，如果存在负环，则不会有最短路径，因为我循环一圈都可以是 u 到 v 的路径更短.

四、代码填空题

1. 一个带有头结点的单链表，删除 min 与 max 之间的结点。

```
//删除在 min 和 max 之间的结点
void RangeDelete(LinkList &L,int mins,int maxs)
{
    NODE *pre = L,*p = L->next;
    while(p!=NULL)
    {
        if(p->data>mins&&p->data<maxs)//找到被删除的结点
        {
            pre->next = p->next;
            free(p);
            p=pre->next;
        }
        else
        {
            //不是的话继续寻找，两者都往前移
            pre = p;
            p = p->next;
```

```
            }
        }//end while
}
```

2. 统计二叉树中小于 x 的数的结点个数。（参考树的各种遍历）

3. 给定一个数组（链表），将所有非零元素移到最后面。

解析：

```
public void moveZeroes(int[] nums) {
            int size = nums.length;
            int startIndex = 0;
           // 0 元素开始的位置
            int endIndex = 0;
           // 0 元素结束的位置
            int currentNum;
            int i= 0;
            //  第一步：找到第一个 0 元素开始的位置
            //  并将第一个 0 元素的游标赋值给 startIndex&endIndex
            while(i < size){
                currentNum = nums[i];
                if (currentNum == 0) {
                    startIndex = i;
                    endIndex = i;
```

```
            break;
        }
        ++i;
    }
    // 如果当前数组中没有找到 0 元素，则推出
    if (nums[endIndex] != 0)
        return;
    // 将当前 i 的值加 1；直接从刚才 0 元素位置的后一位置开始循环
    ++i;
    while (i < size) {
        currentNum = nums[i];
        if (currentNum == 0){
                //如果当前元素等于 0，则将 i 值赋值给 endIndex
                endIndex = i;
        } else {
            // 如果不为 0
            //则将当前元素赋值给 nums[startIndex]
            // 并将当前位置的元素赋值为 0
            // startIndex 和 endIndex 都加 1；
            nums[startIndex] = currentNum;
            nums[i] = 0;
            ++startIndex;
```

```
                ++endIndex;
            }
            ++i;
        }
    }
```

五、简答题

1. 给一颗树

问题：

1) 写出树的顺序存储结构。

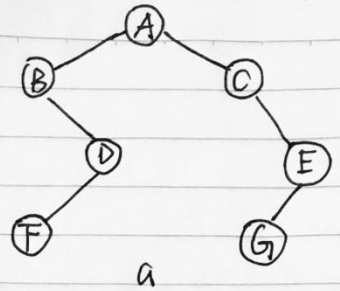
2) 写出树的后序遍历序列。

3) 把二叉树转化成森林。

解析：

No.

例：有一棵二叉树如图a所示

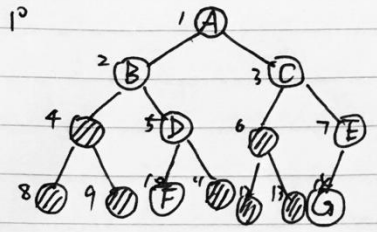


a

问：1）画出树的顺序存储结构：

/* 一棵二叉树只有先转换成完全二叉树后才能用 */

/* 顺序存储结构进行存储. */



图b 完全二叉树

A	B	C		D		E			F				G
1	2	3	4	5	6	7	8	9	10	11	12	13	14

表1 为树的顺序存储结构

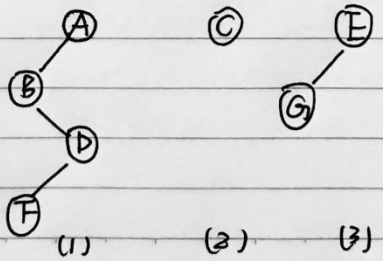
2）画出树的后序遍历序列

FDBGECA

3）把二叉树转化为森林.

1° 把右孩子去线

2° 形成为一棵树



2. 问题：

1）有 n 个顶点的有向强连通分量最多有多少个边？最少有多少个边？

2）表示一个有 1000 个顶点和 1000 条边的有向图的邻接矩阵有多少个矩阵元素。

3）写出两个判断有无环的方法。

解析：

1）最多有 n（n-1）条边，最少有 n 条边

2）1000^2

3）深度优先遍历、拓扑排序

3. 问题：

1）给出一组数，构造哈希函数，要求使用线性探测再散列方法。

2）写出哈希表。

3）写出查询成功查找次数和失败查找次数。

解析：

例：将关键字 (7, 8, 30, 11, 18, 9, 14) 散列存储到散列表中.
散列表的存储空间是一个下标为从0开始的一维数组.
要求处理冲突采用线性探测再散列法. 装填因子为0.7.

答：

/* 装填因子 = 关键字总数 / 哈希表的长度 */

1° 由装填因子0.7得，哈希表的长度 $L = 7/0.7 = 10$.
所以，需要建立下标为 0~9 的一维数组。

2° 确定哈希函数，用除留余数法，取关键字被某个不大于哈希表表长 L 的数 p 除后所得余数为哈希地址（p为素数）
故此题中 p 选为 7
哈希函数：$H(key) = key \ MOD \ p$，$p=7$.

3° 根据哈希函数处理关键字如下表

key	7	8	30	11	18	9	14
H(key)	0	3	6	5	5	6	0

4° 采用线性探测再散列法处理冲突，所构造的散列表为：

地址：	0	1	2	3	4	5	6	7	8	9
关键字：	7	14		8		11	30	18	9	

详细步骤：1' key=7，地址=0，因为散列表下标为0的位置为空，不冲突，插入。

2' key=8，地址=3，因为散列表下标为3的位置为空，不冲突，插入。

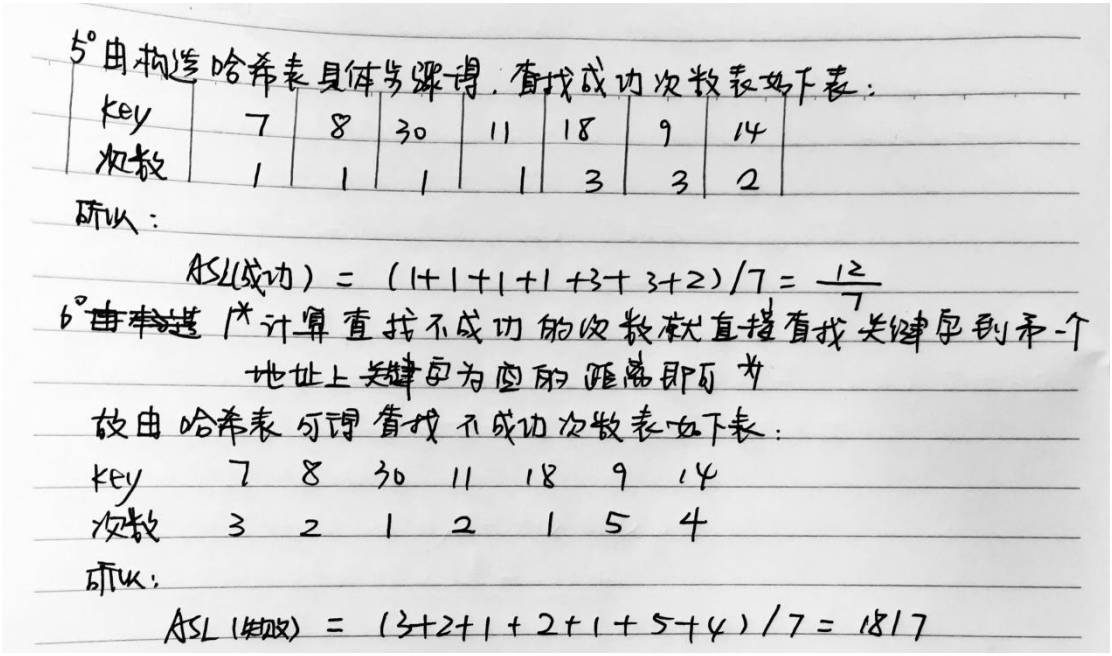
3' key=30，地址=6，因为散列表下标为6的位置为空，不冲突，插入。

4' key=11，地址=5，因为散列表下标为5的位置为空，不冲突，插入。

5' key=18，地址=5，因为散列表下标为5的位置已有关键字11，冲突，此时采用线性探测再散列法，探测下一位置为6，但散列表下标为6的位置已有关键字30，则继续探测下一位置为7，不冲突，插入。

6' key=9，地址=6，因为散列表下标为6的位置已有关键字30，则探测下一位置为7，但已有关键字18，探测下一位置为8，不冲突，插入。

7' key=14，地址为0，因为0地址已有关键字7，则探测下一位置为1，不冲突



4. 给出 6 个字母 abcdef 及其相应的权值（哈弗曼树）。

问题：

1）给 6 个字母编码。

2）求 wpl。

3）假设一个字节用 8 个二进制单位组成，需要多少个字节。

解析：

No.

Date.

例：设字符 abcdef 的权值依次为 2、4、7、11、8、9，构造哈夫曼树。

答：

1°把 a、b、c、d、e、f 看成树，每一棵树有一个结点。

2°在 a、b、c、d、e、f 中选出两个根结点的权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点的值为两个左、右子树的值之和。

3°在 a、b、c、d、e、f 中删掉已生成新树的两棵树，再重复 2°，最后构成一棵哈夫曼树。

1' ② ④ ⑦ ⑪ ⑧ ⑨

2' [6] ⑦ ⑪ ⑧ ⑨

3' [13] ⑪ ⑧ ⑨

4' [13] ⑪ [17]

5' [24] [17]

6' [41]

4° /*根据左0右1的编码或偏码*/

5° 偏码后的哈夫曼树：

Date.

6° 最后各字符偏码如下：

a: 1100　　b: 1101　　c: 111

d: 10　　e: 00　　f: 01

7° WPL = (8+9+11)×2 + 7×3 + (2+4)×4 = 101

8°：共需字节数：101/8 以13个字节

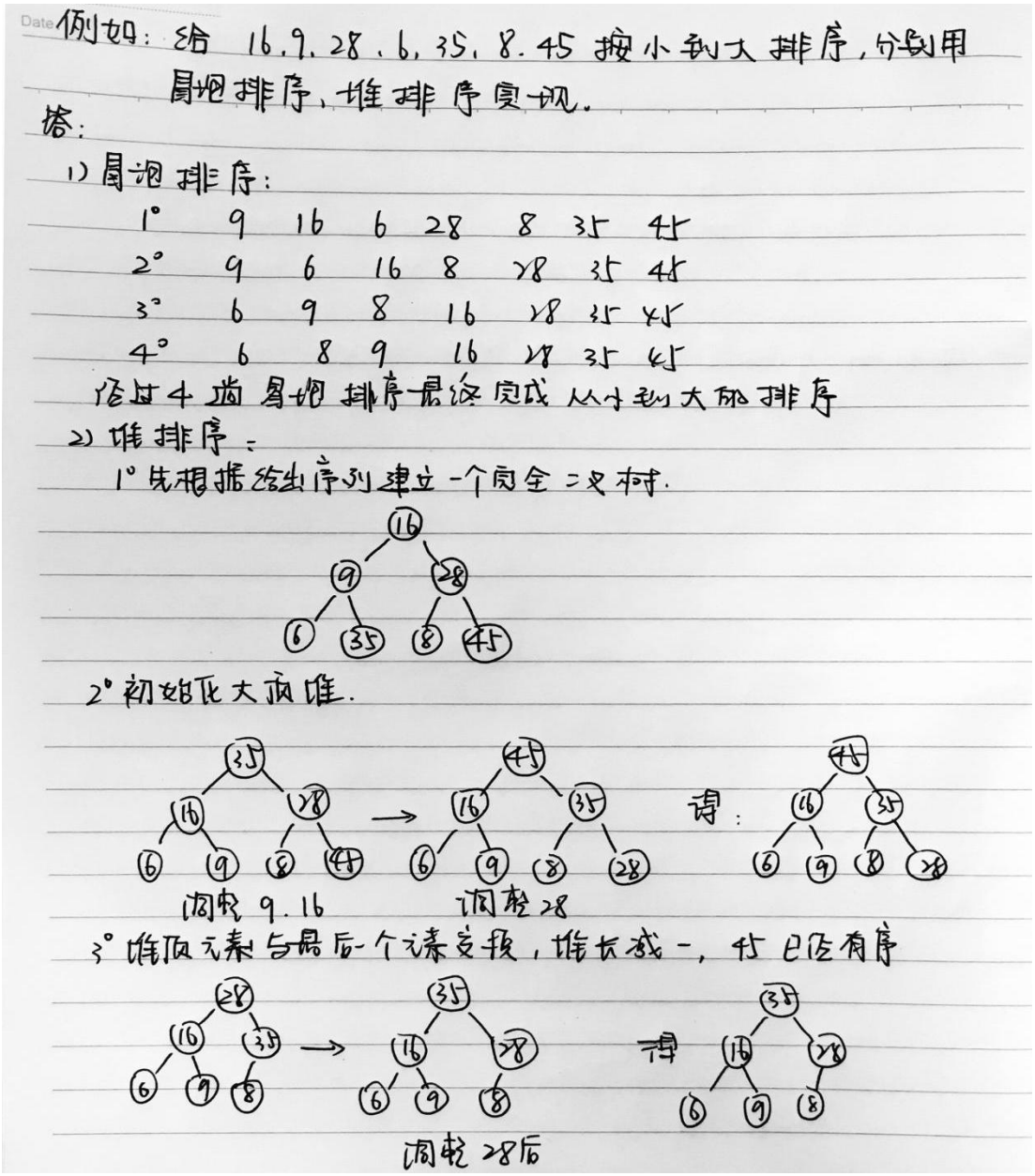
5. 给出一组数，按要求完成排序，并写出每一趟排序过程。

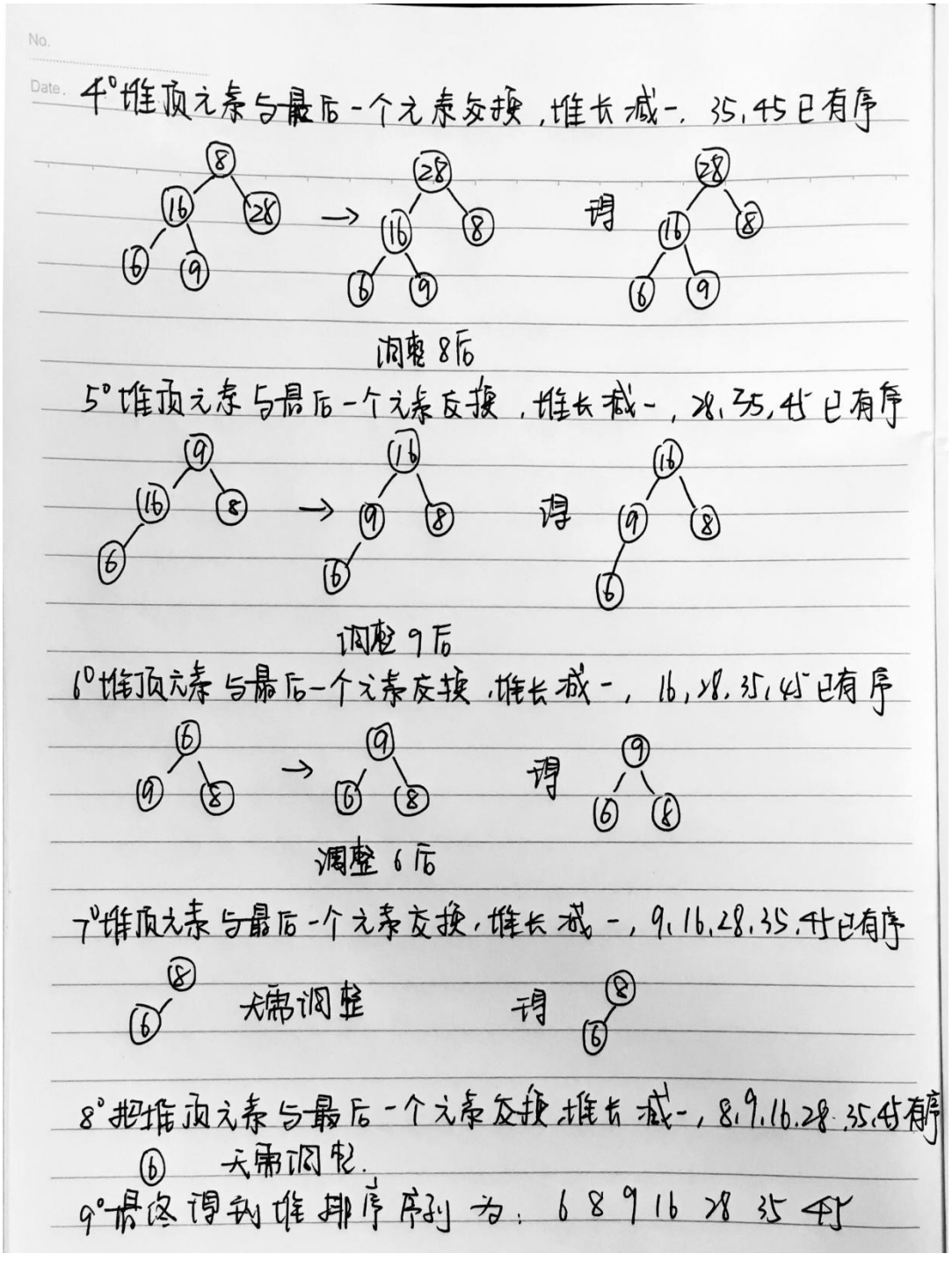
问题：

1）使用冒泡排序。

2）使用堆排序。

解析;



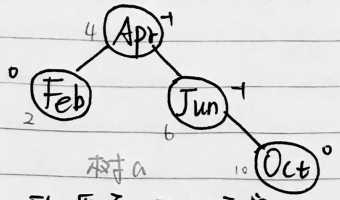


6. 给出 12 个月份的英文单词缩写，完成平衡二叉树的构建，并指出调整类型。

解析：

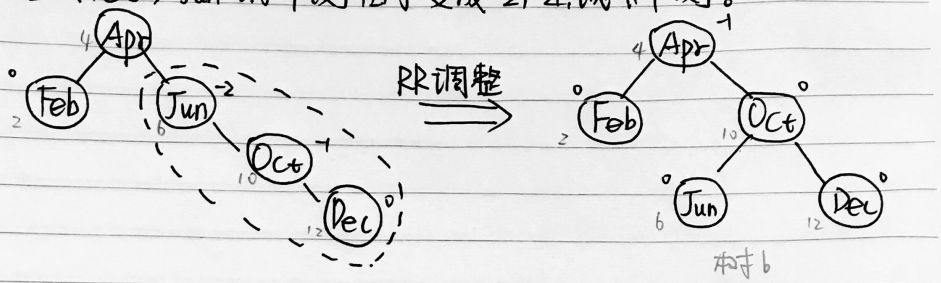
例：从 Apr, Jun, Feb, Oct, Dec, July, Jan, Mar, Aug, Nov, May 这组序列构造一个平衡二叉树（按时间序），并给出调整类型。

答：1° 以Apr为根结点，插入 Jun, Feb, Oct 得树 a.



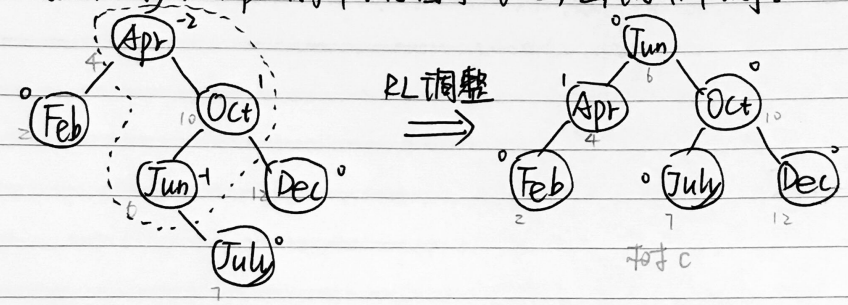
树a

2° 插入 Dec，Jun 的平衡因子变成 -2，出现不平衡。



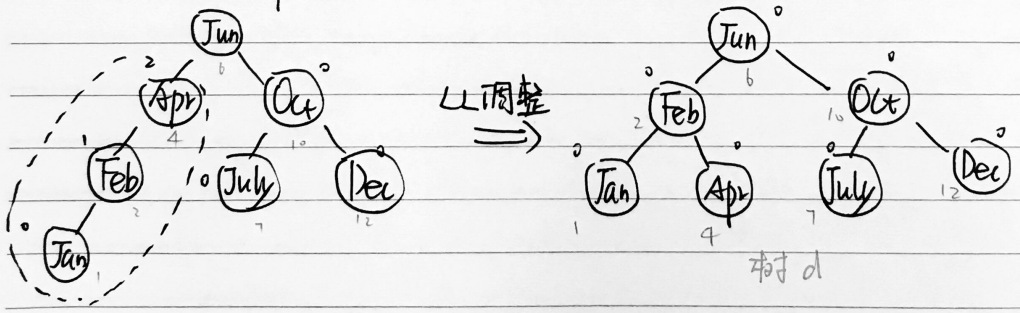
树b

3° 插入 July，Apr 的平衡因子为 -2，出现不平衡。

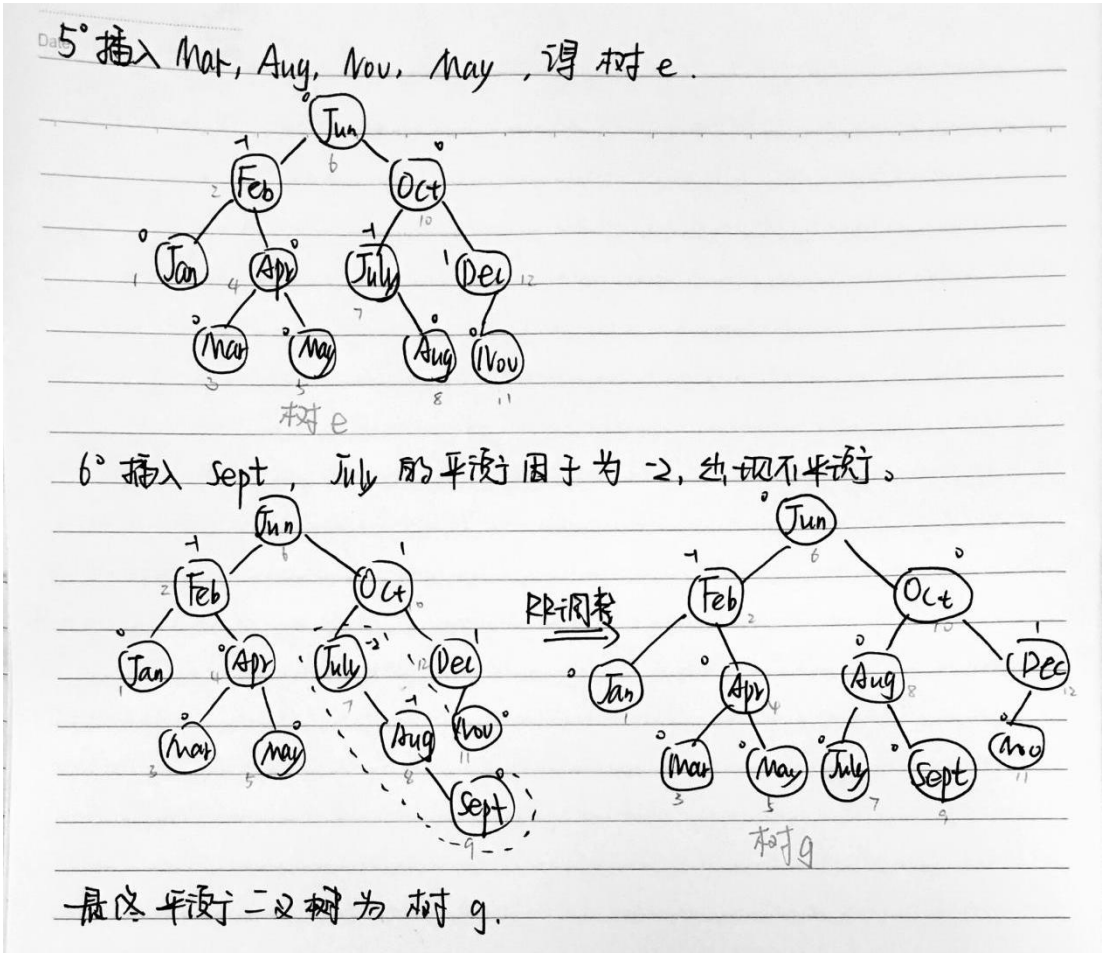


树c

4° 插入 Jan，Apr 的平衡因子为 2，出现不平衡。



树d



六、算法题

1.设计算法求链表 A-B，然后把 A，B 多余的结点释放掉。

```
#include <stdio.h>
#include <stdlib.h>
typedef struct list{
  int elem;//存数链表节点的元素值
  struct list *next;//指向下一节点
}LIST;
LIST *createlist(int n)
```

```
{
  LIST *head,*p,*p1;
  int i;
  head=(LIST *)malloc(sizeof(LIST));
  p=head;
  scanf("%d",&(p->elem));
  p->next=NULL;
  for(i=1;i<n;i++)
  {
    p1=(LIST *)malloc(sizeof(LIST));
    scanf("%d",&(p1->elem));
    p1->next=NULL;
    p->next=p1;
    p=p1;
  }
  return head;
}
LIST *A_B(LIST *A,LIST *B)//计算 A-B，计算结果保存在 A 中
{
  LIST *pa,*pa_prev,*pb;
  int e,flag;
  pa=A;
```

```
pa_prev=A;
while(pa!=NULL)//遍历链表 A 中的每一个元素
{
 flag=0;
 e=pa->elem;
 pb=B;//对 A 中的每一个元素 e，都从 LB 的表头开始查找
 while(pb!=NULL)
 {
  if (pb->elem==e) //若 A 中的元素 B 也有，则在 A 中删除该元素
  {
   if (pa==A) //pa 是表头
   {
      A=pa->next;
      pa_prev=pa->next;
      pa=pa->next;
      flag=1;
   }
   else if(pa->next==NULL)//pa 是表尾
   {
     pa=NULL;
     pa_prev->next=pa;
     flag=1;
```

```
        }
        else//非表头元素和表尾元素
        {
          pa_prev->next=pa->next;
          pa=pa->next;
          flag=1;
        }
        break;
      }
      else pb=pb->next;
    }
    if (flag==0) {pa_prev=pa;pa=pa->next;}
  }
  return A;
}
int main()
{
  LIST *LA,*LB;
  LIST *p;
  int num;
  printf("请输入链表 A 的长度：");
  scanf("%d",&num);
```

```
 printf("请输入链表 A 的元素，中间用空格隔开：");
 LA=createlist(num);
 printf("\n 请输入链表 B 的长度：");
 scanf("%d",&num);
 printf("请输入链表 B 的元素，中间用空格隔开：");
 LB=createlist(num);
 LA=A_B(LA,LB);//计算集合差
 p=LA;//输出 A-B 的结果
 printf("\nLA-LB=");
 while(p!=NULL)
 {
   printf("%4d",p->elem);
   p=p->next;
 }
 return 0;
 printf("\n");
}
```

2. 求图中一个原点到其他各个定点的最短路径的算法（迪杰斯特拉算法书上有，默出来即可）