哈尔滨工业大学(深圳)2023 年春《数据结构》
第四次作业 查找与排序 参考答案

学号		姓名		成绩	

1、简答题

1-1 对下面的关键字集{30，15，21，40，25，26，36，37}若查找表的装填因子为 0.8，采用线性探测再散列方法解决冲突，完成下列内容：

(1)设计哈希函数；

(2)画出哈希表；

(3)计算查找成功和查找失败的平均查找长度。

【参考答案】

由于装填因子为 0.8，关键字有 8 个，所以表长为 8/0.8=10。

(1)用除留余数法，哈希函数为 H(*key*) = *key* % 7

(2)哈希表如下：

散列地址	0	1	2	3	4	5	6	7	8	9
关键字	21	15	30	36	25	40	26	37		
成功比较次数	1	1	1	3	1	1	2	6		
失败比较次数	9	8	7	6	5	4	3			

(3)计算查找失败时的平均查找长度，必须计算不在表中的关键字，当其哈希地址为 *i* (*0≤i≤m-l*)时的查找次数。本例中 *m=100* 故查找失败时平均查找长度为：

ASL_{unsucc} = (9+8+7+6+5+4+3) /7 = 6 ；

ASLsucc = 16/8 = 2 。

1-2 设有 6 个有序表 A、B、C、D、E、F，分别含有 10、35、40、50、60 和 200 个数据元素，各表中元素按升序排列。要求通过 5 次两两合并，将 6 个表最终合并成 1 个升序表，并在最坏情况下比较的总次数达到最小。请回答下列问题。

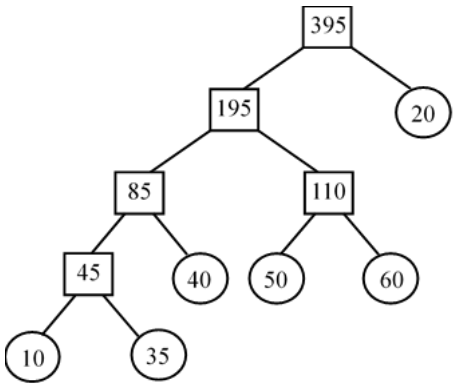
（1）给出完整的合并过程，并求出最坏情况下比较的总次数。

（2）根据你的合并过程，描述 n（n≥2）个不等长升序表的合并策略，并说明理由。

【参考答案】

（1）6 个表的合并顺序如下图所示。

根据上图中的哈夫曼树，6 个序列的合并过程为：



对应于合并过程的哈夫曼树

第 1 次合并：表 A 与表 B 合并，生成含 45 个元素的表 AB；

第 2 次合并：表 AB 与表 C 合并，生成含 85 个元素的表 ABC；

第 3 次合并：表 D 与表 E 合并，生成含 110 个元素的表 DE；

第 4 次合并：表 ABC 与表 DE 合并，生成含 195 个元素的表 ABCDE；

第 5 次合并：表 ABCDE 与表 F 合并，生成含 395 个元素的最终表；

由于合并两个长度分别为 m 和 n 的有序表，最坏情况下需要比较 m+n-1 次，故最坏情况下比较的总次数计算如下：

第 1 次合并：最多比较次数= 10+35-1 = 44

第 2 次合并：最多比较次数 = 45+40-1 = 84

第 3 次合并：最多比较次数 = 50+60-1 = 109

第 4 次合并：最多比较次数 = 85+110-1 = 194

第 5 次合并：最多比较次数 = 195+200-1 = 394

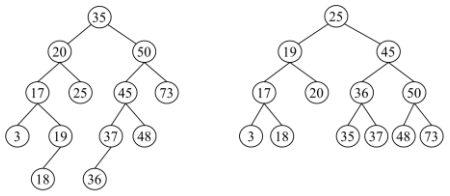
比较的总次数最多为：44+84+109+194+394 = 825

（2）各表的合并策略

在对多个有序表进行两两合并时，若表长不同，则最坏情况下总的比较次数依赖于表的合并次序。可以借用哈夫曼树的构造思想，依次选择最短的两个表进行合并，可以获得最坏情况下最佳的合并效率。

1-3 给出关键字序列（35 20 25 50 17 73 45 19 37 3 18 48 36），分别创建二初始为空的二叉排序树（BST）和平衡二叉树（AVL），只要求给出最后的结果。

【参考答案】

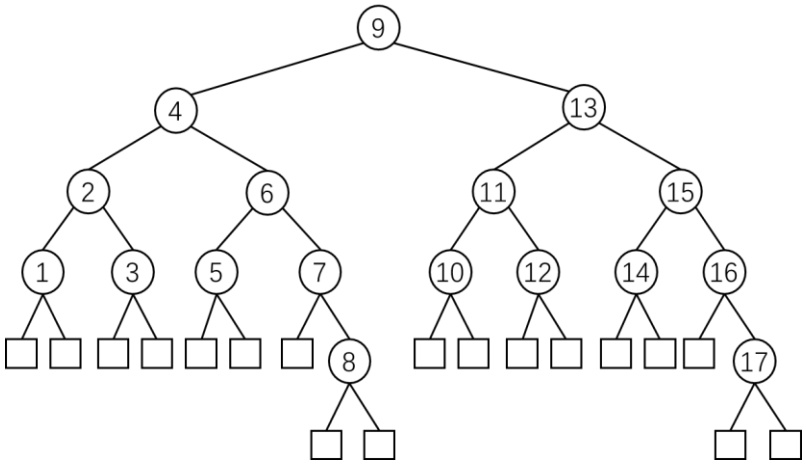


二叉排序树BST　　平衡二叉树AVL

1-4 由 17 个元素构成升序序列，计算在该序列上进行二分查找的查找成功与查找失败的平均查找长度。

【参考答案】

查找树如下图所示。

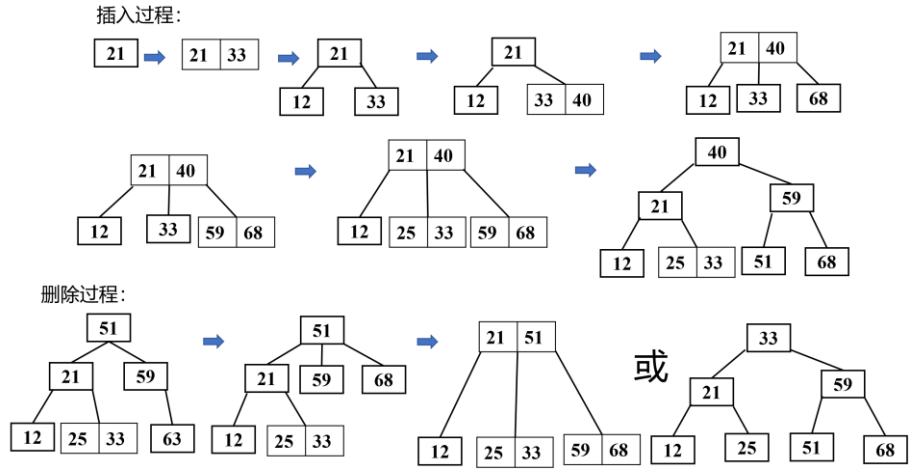


$ASL_{成功}$ =（1×1+2×2+4×3+8×4+2×5）/17 = 59/17

$ASL_{失败}$ =（14×4+4×5）/18 = 76/18

1-5 已知一组关键字为{21，33，12，40，68，59，25，51}，依次插入关键字，生成一棵 3 阶 B-树；如果此后删除 40，请画出每次插入与删除执行后 B-树的状态。

【参考答案】



2、算法设计

2-1 已知二叉排序树采用二叉链表存储结构（lchild,data,rchild），根结点的指针为 T，且有 int data。现已知 int x,请设计算法，从大到小输出二叉排序树中所有值不小于 x 的结点的 data。

【参考答案】

（1） 设计思想

已知在二叉排序树中，按照 LDR 的顺序遍历可以得到一个递增的序列，因此按照 RDL 的顺序遍便可得到逆序序列。

（2）算法描述

```
void RDLPrint(BTREE T, int    x)
{
    if (T) {
            RDLPrint(T->rchild , x);
            if(T->data>=x)    printf("%d,",T->data) ;
            RDLPrint(T->lchild, x);
            }
 }
```

2-2 有一种简单的排序算法,叫做计数排序(Count sorting)。这种排序算法对一个待排序的表用数组表示进行排序,并将排序结果存放到另一个新的表中。必须注意的是,表中所有待排序的关建码互不相同。计数排序算法针对表中的每个记录,扫描待排序的表一趟,统计表中有多少个录的关键码比该记录的关键码小。假设针对某一个记录,统计出的计数值为 c,那么,这个记录在新的有序表中的合适的存放位置即为 c。

①给出适用于计数排非序的教据表定义。

②编写实现计数排序的算法。

③对于有 n 个记录的表,关键码比较欠数是多少?

④与简单选择排序相比较,这种方法是否更好?为什么?

【参考答案】

（1） 给出适用于计数排非序的教据表定义

```
Typedef struct
{    int key;       //基于比较关键字
     Datatype info;   //数据其他属性
} RecordType;
```

（2） 算法

```
void CountSort(RecordType a[], RecordType b[], int n)
{
    int i,j,cnt;
    for(i=0;i<n;i++)
    {    for(j=0,cnt=0;j<n;j++)
         if(a[j].key<a[i].key) cnt++;   //统计关键字比 a[i]小的元素个数
         b[cnt]=a[i];
    }
```

}

（3） 比较次数

n 个元素的计数排序比较次数为 n^2；

（4） 简单选择排序比较次数为 n(n+1)/2，尽管其时间复杂度都是 $O(n^2)$,但很显然简单选择排序的比较次数少于计数排序；

从空间复杂度来看，简单选择排序的空间复杂度为 O(1)，而计数排序的空间复杂度为 $O(n^2)$.