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

学号	220110515	姓名	金正达	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：

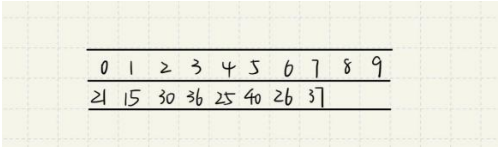
(1)设计哈希函数；

(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。

答：（1）hash(key) = key % 7

（2）



（3）ASLu =(1 + 1 + 1 + 3 + 1 + 1 + 2 + 6) / 8 = 2；

ASLs = (9 + 8 + 7 + 6 + 5 + 4 + 3 + 2) / 7 = 6.28.

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。

答：（1）第一次合并 A 和 B，得到 AB；

第二次合并 AB 和 C，得到 ABC；

第三次合并 D 和 E，得到 DE；

第四次合并 ABC 和 DE，得到 ABCDE；

第五次合并 ABCDE 和 G，得到 ABCDEFG。

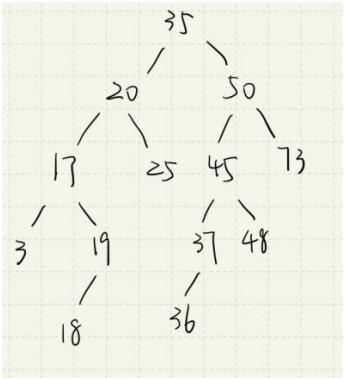
最坏情况需要 825 次。

（2）两两合并，每次均合并含有数据最少的两个有序表，即构造赫夫曼树，赫夫曼树为最优二叉树，合并的比较次数取决于表的大小。

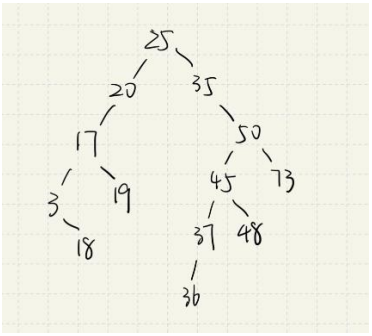
1-3 给出关键字序列（35 20 25 50 17 73 45 19 37 3 18 48 36），分别创建二

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。

答：BST：



AVL：



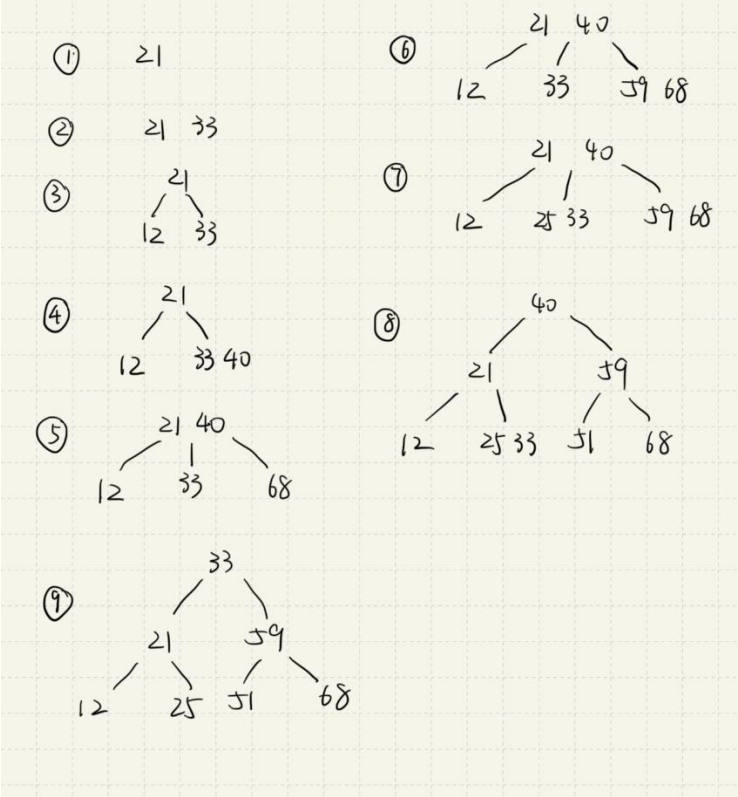
1-4 由 17 个元素构成升序序列，计算在该序列上进行二分查找的查找成功与查找失败的平均查找长度。

答：ASLu = (5 * 2 + 4 * 8 + 3 * 4 + 2 * 2 + 1) / 17 = 3.47;

ASLs = (14 * 4 + 4 * 5) / 18 = 4.22

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。

答：



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。

答：在二叉排序树中查找，若节点键值大于等于 x，则打印该节点和其右子树键值并在左子树递归调用该函数；若节点键值小于 x，则在其右子树递归调用该函数。

时间复杂度为 O(N)，N 为二叉树节点数量。

```
typedef struct BST
{
    int data;
    struct BST* lchild;
    struct BST* rchild;
}BSTNode;

void inOrderTravel(BSTNode* root) {
    if (root == NULL) {
        return;
    }
    printf(root->data);
    inOrderTravel(root->lchild);
    inOrderTravel(root->rchild);
}

void findNoLessThanX(BSTNode* root, int x) {
    if (root == NULL) {
        return;
    }
    if (root->data >= x) {
        printf("%d", root->data);
        inOrderTravel(root->rchild);
        findNoLessThanX(root->lchild, x);
    } else if (root->data < x) {
        findNoLessThanX(root->rchild, 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,统计表中有多少个记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给出适用于计数排序的数据表定义。

②编写实现计数排序的算法。

③对于有 n 个记录的表,关键码比较次数是多少?

④与简单选择排序相比较,这种方法是否更好?为什么?

答 ：（1）排序的键值应为有固定范围的整数数据集合。

（2）

```
int* CountingSort(int* keys, int n) {
    int min = keys[0];
    int max = keys[0];
    //寻找最大和最小值，确定键值范围
    for (int i = 0; i < n; i++) {
        if (keys[i] > max) {
            max = keys[i];
        }
        if (keys[i] < min) {
            min = keys[i];
        }
    }
    int length = max - min;
    //创建计数数组，初始化，计算各键值出现次数
    int count[length];
    for (int i = 0; i < length; i++) {
        count[i] = 0;
    }
    for (int i = 0; i < n; i++) {
        int index = keys[i] - min;
        count[index] += 1;
    }

    //计算各个键值之前的键值的出现次数
    for (int i = 1; i < length; i++) {
        count[i] += count[i - 1];
    }


    int sort[n];
    for (int i = 0; i < n; i++) {
        int index = keys[i] - min;
        sort[count[index]] = keys[i];
        count[index] += 1;
    }
    return sort;
}
```

(3)由题意，针对表中的每个记录,扫描待排序的表一趟,统计表中有多少个记录的关键码比该记录的关键码小，故共比较 n^2 次。

(4)并非更好，简单选择排序一共比较(n*(n - 1)) / 2 次，不需要额外的空间，而计数排序比较 n^2 次，并需要额外的数组空间。