1. 设有5个数据do、for、if、repeat、while，它们排在一个有序表中，其查找概率分别是p1=0.2，p2=0.15，p3=0.1，p4=0.03，p5=0.01。而查找它们之间不存在数据的概率分别为q0=0.2，q1=0.15，q2=0.1，q3=0.03，q4=0.02，q5=0.01，该有序表如下：

（1）试画出对该有序表分别采用顺序查找和折半查找时的判定树。

（2）分别计算顺序查找的查找成功和不成功的平均查找长度。

（3）分别计算折半查找的查找成功和不成功的平均查找长度。

答：（1）顺序查找判定树如下：

图示, 工程绘图

描述已自动生成

（2）折半查找判定树如下：

图示

描述已自动生成

（2）

顺序查找成功概率：p1+2\*p2+3\*p3+4\*p4+5\*p5=0.97

顺序查找失败概率：q0+2\*q1+3\*q2+4\*q3+5\*q4+5\*q5=1.07

(3)

折半查找成功概率：p3+2\*p1+2\*p4+3\*p2+3\*p5=1.04

折半查找失败概率：2\*q0+3\*q1+3\*q2+2\*q3+3\*q4+3\*q5=1.04

3. 有以下查找算法：

（1）指出fun(a，n，k)算法的功能。

（2）当a[]={2，6，3，8，1，7，4，9}时，执行fun(a，n，1)后的返回结果是什么？一共进行了几次比较。

（3）当a[]={2，6，3，8，1，7，4，9}时，执行fun(a，n，5)后的返回结果是什么？一共进行了几次比较。

答：

1. fun函数的作用是查找数组a中存储元素值为k的元素，查找顺序为：先查找偶数序号后查找奇数序号，找到了返回其位置，未找到返回-1。
2. 返回4。一共进行了3次比较。
3. 返回-1。一共进行了8次比较。

6.由23、12、45 关键字构成的二叉排序树有多少棵，其中属于平衡二叉树的有多少

棵？

答：二叉排序树个数满足卡塔兰数，当n=3时，结果为5，其中属于平衡二叉树的只有一棵。

8. 将整数序列（4，5，7，2，1，3，6）中的元素依次插入到一棵空的平衡二叉树中，试构造相应的平衡二叉树，要求用图形给出构造过程。

文本, 信件

描述已自动生成

9. 已知一棵5阶B-树中有53个关键字，则树的最大高度是多少？

答：当每个结点的关键字个数都最少时，该B-树的高度最大。

根结点最少有1个关键字、2棵子树，第1层至少有1个结点。除根结点外每个结点至少有个关键字、3棵子树，则第2层至少有2个结点，共2×2=4个关键字。第3层至少有2×3个结点，共2×3×2=12个关键字。第4层至少有6×2个结点，共6×3×2=36个关键字。而1+4+12+36=53，加上外部结点层，该B-树中最大高度是5层。

10. 设有一组关键字（19，1，23，14，55，20，84，27，68，11，10，77），其哈希函数为h(key)=key % 13。采用开放地址法的线性探测法解决冲突，试在0～18的哈希表中对该关键字序列构造哈希表，并求在成功和不成功情况下的平均查找长度。

计算结果如下：

文本, 信件

描述已自动生成

构建的哈希表如下：

文本

中度可信度描述已自动生成

因为查找次数与探测次数相同，故查找成功的平均查找长度为(1+2+1+4+3+1+1+3+1+1+3+2)/12=1.92

如果查询的数据遇到了空的关键字，则说明其不存在，即查找失败，根据这个原则可以写出查找失败的哈希表：

图形用户界面

中度可信度描述已自动生成

则查找失败的平均查找长度为：(1+9+8+7+6+5+4+3+2+1+5+4+3)/13=58/13=4.46

11. 设计一个折半查找算法，求查找到关键字为k的记录所需关键字的比较次数。假设k与R[i].key的比较得到3种情况，即k==R[i].key，k<R[i].key或者k>R[i].key，计为1次比较（在教材中讨论关键字比较次数时都是这样假设的）。

int BinSearch1(RecType R[], int n, KeyType k)

{

int low = 0, high = n - 1, mid;

int cnum = 1; //成功查找需要1次比较

while (low <= high)

{

mid = (low + high) / 2;

if (R[mid].key == k)

return cnum;

else if (k < R[mid].key)

high = mid - 1;

else

low = mid + 1;

cnum++;//已经进行了一次比较

}

cnum--; //不成功查找比较次数需要减1

return cnum;

}

12. 设计一个算法，判断给定的二叉树是否是二叉排序树。假设二叉树中结点关键字均为正整数且均不相同。

因为二叉排序树的满足左孩子<当前结点大小<右孩子，如果使用中序遍历，则可以发现其输出顺序时从小到大，通过这个特点可以写出如下算法：

KeyType predt = -32768; //predt为全局变量,保存当前结点中序前驱的值,初值为-∞

bool JudgeBST(BSTNode\* bt)

{

bool b1, b2;

if (bt == NULL)

return true;

else

{

b1 = JudgeBST(bt->lchild); //判断左子树

if (b1 == false) //左子树不是BST，返回假

return false;

if (bt->key < predt) //当前结点违反BST性质，返回假

return false;

predt = bt->key;

b2 = JudgeBST(bt->rchild); //判断右子树

return b2;

}

}