# 查找表--21计科李明

## 相关概念/逻辑结构

- 逻辑结构
  - 由同一类型的数据元素构成的集合
- 关键字
  - 关键字是数据元素中某个数据项的值，用它可以标识一个数据元素。若此关键字可以唯一地标识一个数据元素，则称此关键字为主关键字。反之，称可以标识若干数据元素的关键字为次关键字。当数据元素只有一个数据项时，其关键字即为该元素的值。
- 查找
  - 根据给定的某个值，在查找表中确定一个其关键字等于给定值的数据元素。若查找中存在这样的数据元素，则称查找成功，否则称查找不成功。（此处只考虑相等关系）
- 平均查找长度
  - 在查找表中找到数据元素的位置，需和给定值进行比较的关键字个数的期望值称为查找算法在查找成功时的平均查找长度。在查找表中找不到数据元素的位置，需和给定值进行比较的关键字个数的期望值称为查找算法在查找不成功时的平均查找长度。

## 线性表的查找

### 顺序查找/线性表

- 基本思想
  - 从线性表的一端向另一端逐个将关键码与给定值进行比较，
    若相等，则查找成功，给出该记录在表中的位置；
    若整个表检测完仍未找到与给定值相等的关键码，则查找失败，给出失败信息。
- 哨兵的概念
  - 哨兵就是待查值，将哨兵放在查找方向的尽头处，免去了在查找过程中每一次比较后都要判断查找位置是否越界，从而提高查找速度。
- 优点/应用特点
  - 算法简单而且使用面广。
  - 对表中记录的存储结构没有任何要求，顺序存储和链接存储均可；
  - 对表中记录的有序性也没有要求，无论记录是否按关键码有序均可。
- 缺点
  - 平均查找长度较大，特别是当待查找集合中元素较多时，查找效率较低
- 适用条件
  - 线性表中的记录必须按关键码有序；必须采用顺序存储。

### 折半查找/线性表

- 基本思想/过程
  - 在有序表中（low, high,low<=high），取中间记录作为比较对象
  - 若给定值与中间记录的关键码相等，则查找成功；
  - 若给定值小于中间记录的关键码，则在中间记录的左半区继续查找；
  - 若给定值大于中间记录的关键码，则在中间记录的右半区继续查找。
  - 不断重复上述过程，直到查找成功，或所查找的区域无记录，查找失败。
- 折半查找判定树
  - 概念：
    - 折半查找的过程可以用二叉树来描述，树中的每个结点对应有序表中的一个记录，结点的值为该记录在表中的位置。
      通常称这个描述折半查找过程的二叉树为折半查找判定树，简称判定树。
      - （1）当n=0时，折半查找判定树为空；
        （2）当n＞0时，折半查找判定树的根结点为mid=(n+1)/2，根结点的左子树是与有序表r[1] ~ r[mid-1]相对应的折半查找判定树，根结点的右子树是与r[mid+1] ~ r[n]相对应的折半查找判定树
  - 特点
    - 任意两棵折半查找判定树，若它们的结点个数相同，则它们的结构完全相同。
      具有n个结点的折半查找树的高度为：$(\log_2 n) + 1$
  - 性质
    - 任意结点的左右子树中结点个数最多相差1
    - 任意结点的左右子树的高度最多相差1
    - 任意两个叶子所处的层次最多相差1

### 分块查找/索引顺序查找

- 分块查找表存储结构
  - 查找表由"分块有序"的线性表和索引表组成。
- 基本思想
  - （1）首先查找索引表
    索引表是有序表，可采用二分查找或顺序查找，以确定待查的结点在哪一块。
    （2）然后在已确定的块中进行顺序查找
    由于块内无序，只能用顺序查找。
- 应用特点/优缺点
  - 优点：插入删除比较容易，无需进行大量移动
  - 缺点：要增加一个索引表的存储空间并对初始索引表进行排序运算

## 树表的查找

### 二叉排序树/二叉查找树

- 定义
  - 二叉排序树或者是一棵空树，或者是一棵具有下列特性的非空二叉树：
    1） 若左子树非空，则左子树上所有结点关键字均小于根结点的关键字值；
    2） 若右子树非空，则右子树上所有结点关键字均大于根结点的关键字值；
    3） 左、右子树本身也分别是一棵二叉排序树。
- 查找
  - 若二叉排序树非空，则将给定值与根结点的关键字比较，若相等，则查找成功；
    若不等，则当根结点的关键字值大于给定关键字值时，在根结点的左子树中查找；否则在根结点的右子树中查找。
- 操作
  - 插入
    - 若原二叉排序树为空，则直接插入结点；否则，若关键字＜根结点关键字，则插入左子树，若关键字＞根结点关键字，则插入右子树。
  - 删除
    - 若被删除结点z是叶结点，则直接删除，不会破坏二叉排序树的性质。
    - 若结点z只有一棵左子树或右子树，则让z的子树成为z父结点的子树，替代z的位置。
    - 若结点z有左、右两棵子树，则令z的直接后继（或直接前驱）替代z，然后从二叉排序树中删除这个直接后继（或直接前驱），这样就转换成了第一或第二种情况。
  - 构造
    - 每读入一个元素，就建立一个新结点，若二叉排序树为空，则将新结点作为二叉排序树的根结点；若二叉排序树非空，则将新结点的值与根结点的值比较，若小于根结点的值，则插入左子树，否则插入右子树。
- 效率
  - 平均查找长度
    - Σ(本层高度*本层结点个数) / 结点总数
  - 查找不成功的平均查找长度
    - Σ(本层高度*本层补上的叶子结点个数) / 补上的叶子总数

### 平衡二叉树

- 定义
  - 任意节点的子树的高度差都小于等于 1
  - 每个结点附加一个数字，给出该结点左子树与右子树的高度差．这个数字称为结点的平衡因子（B F），节点因子只能是-1 1 0
- 查找效率
  - O(logn)

### B-树/平衡查找树

- 特点
  - B树可以定义一个m值作为预定范围，即m路(阶)B树。
    每个节点最多有m个孩子。
    每个节点至少有ceil(m/2)个孩子，除了根节点和叶子节点外。
    对于根节点，子树个数范围为[2,m]，节点内值的个数范围为[1,m-1]。
    对于非根节点，节点内的值个数范围为[ceil(m/2)-1,m-1]。
    根节点(非叶子节点)至少有两个孩子。
    一个有k个孩子的非叶子节点包含k-1个值。
    所有叶子节点在同一层。
    节点内的值按照从小到大排列。
    父节点的若干值作为分离值分成多个子树，左子树小于对应分离值，对应分离值小于右子树
- 操作
  - 添加，删除，查找
- 应用特点/优点
  - B树的每一个结点都包含key(索引值) 和 value(对应数据)，因此方位离根结点近的元素会更快速。（相对于B+树）
- 缺点
  - 不利于范围查找(区间查找)，如果要找 0~100的索引值，那么B树需要多次从根结点开始逐个查找。
  - 而B+树由于叶子结点都有链表，且链表是以从小到大的顺序排好序的，因此可以直接通过遍历链表实现范围查找。

### B+树

- 与B树的区别
  - B+树内部有两种结点，一种是索引结点，一种是叶子结点。
    B+树的索引结点并不会保存记录，只用于索引，所有的数据都保存在B+树的叶子结点中。而B树则是所有结点都会保存数据。
    B+树的叶子结点都会被连成一条链表。叶子本身按索引值的大小从小到大进行排序。即这条链表是 从小到大的。多了条链表方便范围查找数据。
    B树的所有索引值是不会重复的，而B+树 非叶子结点的索引值 最终一定会全部出现在 叶子结点中
- 优点
  - 可以直接通过遍历链表实现范围查找

## 散列表的查找

- 基本概念
  - 表中的数据元素是分散排列的
- 构造方法
  - 直接寻址法
    - 取keyword或keyword的某个线性函数值为散列地址。即H(key)=key或H(key) = a•key + b，当中a和b为常数（这样的散列函数叫做自身函数）
  - 数字分析法
    - 找出数字的规律，尽可能利用这些数据来构造冲突几率较低的散列地址
  - 平方取中法
    - 取keyword平方后的中间几位作为散列地址
  - 折叠法
    - 将keyword切割成位数同样的几部分，最后一部分位数能够不同，然后取这几部分的叠加和（去除进位）作为散列地址
  - 伪随机数法
    - 取keyword被某个不大于散列表表长m的数p除后所得的余数为散列地址
  - 除留余数法
    - 选择一随机函数，取keyword的随机值作为散列地址，通经常使用于keyword长度不同的场合
- 处理散列冲突
  - 开放地址法
  - 再散列函数法
  - 链地址法
  - 公共溢出区法

## 操作

- 查询元素是否在表中
- 检索某个特定的数据元素的各种属性
- 在查找表中插入一个数据元素
- 从查找表中删去某个数据元素

动态查找表的范畴

## 分类/应用

- 静态查找表
  - 不会对查找表中的元素进行增删操作
    - 静态查找适用于：查找集合一经生成，便只对其进行查找，而不进行插入和删除操作; 或经过一段时间的查找之后，集中地进行插入和删除等修改操作；
- 动态查找表
  - 需要对查找表中的元素进行增删操作
    - 动态查找适用于：查找与插入和删除操作在同一个阶段进行，例如当查找成功时，要删除查找到的记录，当查找不成功时，要插入被查找的记录。