

7.1 查找的基本概念

基本概念

查找：在数据集中寻找满足某种条件的数据元素的过程称为查找

查找表（查找结构）：用于查找的数据集合称为查找表，它由同一类型的数据元素（或记录）组成，可以是一个数组或链表等数据类型

查找表操作

- 查询某个特定的数据元素是否在查找表中
- 检索满足条件的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
- 从查找表中删除某个数据元素

静态查找表 查找后不会对表进行任何修改

动态查找表 查找后对表进行修改

关键字：数据元素中唯一标识该元素的某个数据项的值，使用基于关键字的查找，查找结果应该是唯一的

在查找过程中，一次查找的长度是指需要比较的关键字次数，而平均查找长度则是所有查找过程中进行关键字的比较次数的平均值

平均查找长度

数学表达式 $ASL = \sum_{i=1}^n P_i C_i$

n 是查找表的长度

参数含义 P 是查找第 i 个数据元素的概率，一般认为每个数据元素的查找概率相等，即 $P=1/n$

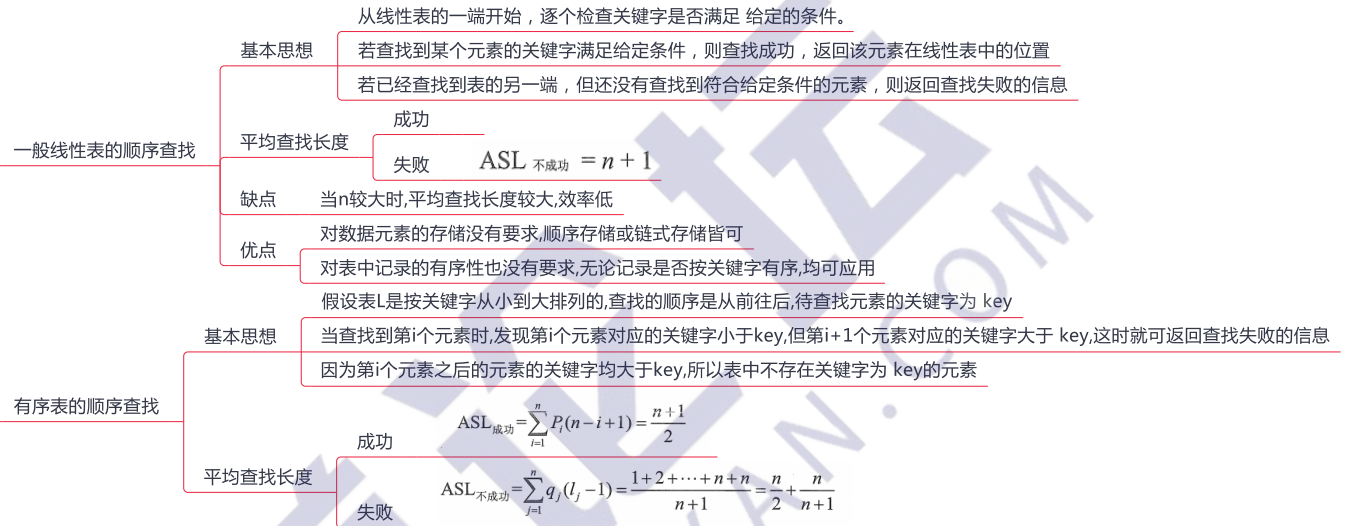
C_i 是找到第 i 个数据元素所需进行的比较次数

平均查找长度是衡量查找算法效率的最主要的指标

7.2 顺序查找和折半查找

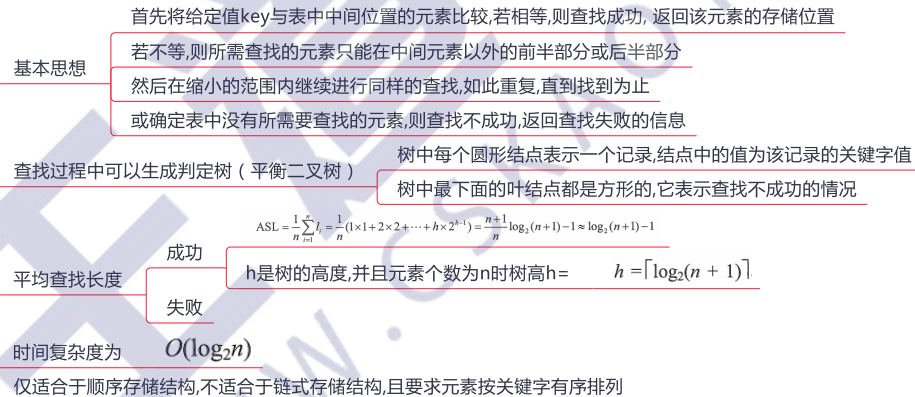
顺序查找

又称线性查找，主要用于在线性表中进行查找。顺序查找通常分为对一般的无序线性表的顺序查找和对按关键字有序的顺序表的顺序查找



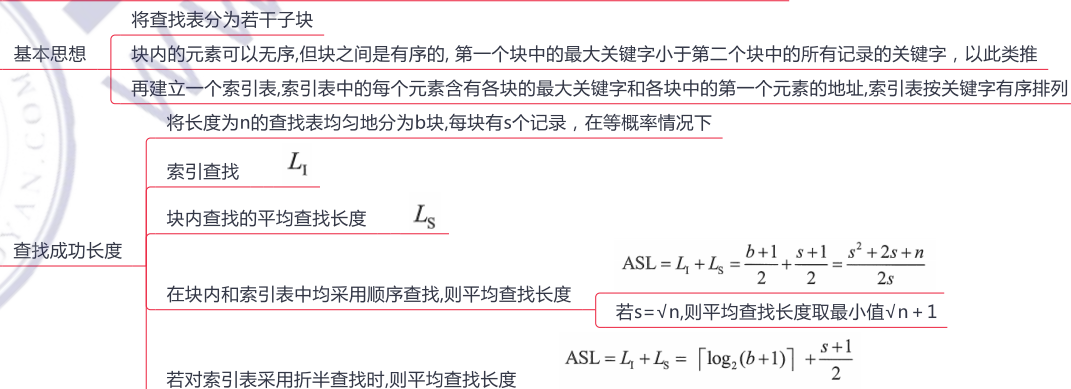
折半查找

折半查找又称二分查找，它仅适用于有序的顺序表



分块查找

又称索引顺序查找，它吸取了顺序查找和折半查找各自的优点，既有动态结构，又适于快速查找



注：仅供王道VIP学员使用 严禁外部传播！

7.3 B树和B+树

B树及其基本操作

概述：又称多路平衡查找树，B树中所有结点的孩子个数的最大值称为B树的阶，通常用m表示

树中每个结点至多有m棵子树，即至多含有 $m-1$ 个关键字

若根结点不是终端结点，则至少有两棵子树

除根结点外的所有非叶结点至少有 $\lceil m/2 \rceil$ 棵子树，即至少含有 $\lceil m/2 \rceil - 1$ 个关键字

所有的叶结点都出现在同一层次上，并且不带信息

B树是所有结点的平衡因子均等于0的多路平衡查找树

结点的孩子个数等于该结点中关键字个数加1

如果根结点没有关键字就没有子树，此时 B树为空

如果根结点有关键字，则其子树必然大于等于两棵，因为子树个数等于关键字个数加1

结点中关键字从左到右递增有序，关键字两侧均有指向子树的指针，左边指针所指子树的所有关键字均小于该关键字，右边指针所指子树的所有关键字均大于该关键字

对任意一棵包含n个关键字、高度为h、阶数为 m的B树

B树的高度（磁盘存取次数） $h \geq \log_m(n+1)$

让每个结点中的关键字个数达到最少，则容纳同样多关键字的B树的高度达到最大

在B树上查找到某个结点后，先在有序表中进行查找

若找到则查找成功，否则按照对应的指针信息到所指的子树中去查找

查找到叶结点时（对应指针为空指针），则说明树中没有对应的关键字，查找失败

定位：利用前述的B树查找算法，找出插入该关键字的最低层中的某个非叶结点（注意：插入位置一定是最低层中的某个非叶结点）

插入：在B树中，每个非失败结点的关键字个数都在区间 $[\lceil m/2 \rceil - 1, m - 1]$

插入后的结点关键字个数小于 m，可以直接插入，插入后检查被插入结点的内关键字的个数，当插入后的结点关键字个数大于m-1时，必须对结点进行分裂

直接删除关键字：若被删除关键字所在结点的关键字个数 $\geq \lceil m/2 \rceil$ ，表明删除该关键字后仍满足B树的定义，则直接删去该关键字

兄弟够借：若被删除关键字所在结点删除前的关键字个数 $= \lceil m/2 \rceil - 1$ 且与此结点相邻的右（或左）兄弟结点的关键字个数 $\geq \lceil m/2 \rceil$ 则需要调整该结点、右（或左）兄弟结点及其双亲结点（父子换位法），以达到新的平衡

兄弟不够借：若被删除关键字所在结点删除前的关键字个数 $= \lceil m/2 \rceil - 1$ 且此时与该结点相邻的左、右兄弟结点的关键字个数均 $= \lceil m/2 \rceil - 1$ ，则将关键字删除后与左（或右）兄弟结点及双亲结点中的关键字进行合并

B+树的基本概念

每个分支结点最多有m棵子树（孩子结点）

非叶根结点至少有两棵子树，其他每个分支结点至少有 $\lceil m/2 \rceil$ 棵子树

结点的子树个数与关键字个数相等。

所有叶结点包含全部关键字及指向相应记录的指针，叶结点中将关键字按大小顺序排列，并且相邻叶结点按大小顺序相互链接起来

所有分支结点（可视为索引的索引）中仅包含它的各个子结点（即下一级的索引块）中关键字的最大值及指向其子结点的指针

M阶的B+树与M阶的B树的主要差异

在B+树中，具有n个关键字的结点只含有n棵子树，即每个关键字对应一棵子树；而在B树中，具有n个关键字的结点含有 $n+1$ 棵子树

在B+树中，每个结点（非根内部结点）的关键字个数n的范围是： $\lceil m/2 \rceil \leq n \leq m$ （根结点： $1 \leq n \leq m$ ）；在B树中，每个结点（非根内部结点）的关键字个数n的范围是： $\lceil m/2 \rceil - 1 \leq n \leq m - 1$ （根结点： $1 \leq n \leq m - 1$ ）

在B+树中，叶结点包含信息，所有非叶结点仅起索引作用，非叶结点中的每个索引项只含有对应子树的最大关键字和指向该子树的指针，不含有该关键字对应记录的存储地址

在B+树中，叶结点包含了全部关键字，即在非叶结点中出现的关键字也会出现在叶结点中；而在B树中，叶结点包含的关键字和其他结点包含的关键字是不重复的

7.4散列表（上）

散列表的基本概念

散列函数:一个把查找表中的关键字映射成该关键字对应的地址的函数,记为 $\text{Hash}(\text{key}) = \text{Addr}$

冲突:散列函数可能会把两个或两个以上的不同关键字映射到同一地址

散列表:根据关键字而直接进行访问的数据结构

对散列表进行查找的时间复杂度为 $O(1)$

散列函数的构造方法

注意

散列函数的定义域必须包含全部需要存储的关键字,而值域的范围则依赖于散列表的大小或地址范围

散列函数计算出来的地址应该能等概率、均匀地分布在整个地址空间中,从而减少冲突的发生

散列函数应尽量简单,能够在较短的时间内计算出任一关键字对应的散列地址

直接取关键字的某个线性函数值为散列地址

直接定址法

散列函数为 $H(\text{key}) = \text{key}$ 或 $H(\text{key}) = a * \text{key} + b$ a 和 b 是常数

特点 计算最简单,且不会产生冲突

适合关键字的分布基本连续的情况

若关键字分布不连续,空位较多,则会造成存储空间的浪费

除留余数法

假定散列表表长为 m ,取一个不大于 m 但最接近或等于 m 的质数 p ,利用以下公式把关键字转换成散列地址

散列函数为 $H(\text{key}) = \text{key} \% p$

特点 最简单、最常用的方法

关键是选好 p ,使得每个关键字通过该函数转换后等概率地映射到散列空间上的任一地址,从而尽可能减少冲突的可能性

数字分析法

设关键字是 r 进制数,而 r 个数码在各位上出现的频率不一定相同,可能在某些位上分布均匀一些,每种数码出现的机会均等

而在某些位上分布不均匀,只有某几种数码经常出现,此时应选取数码分布较为均匀的若干位作为散列地址

特点 适合于已知关键字集合,若更换了关键字

则需要重新构造新的散列函数

平方取中法

取关键字的平方值的中间几位作为散列地址

特点 散列地址分布比较均匀

适用于关键字的每位取值都不够均匀或均小于散列地址所需的位数

7.4散列表（下）

处理冲突的方法

开放定址法

数学递推公式

$$H_i = (H(\text{key}) + d_i) \% m$$

$H(\text{key})$ 为散列函数

m 表示散列表表长; d_i 为增量序列

$$d_i = 0, 1, 2, \dots, m-1$$

线性探测法

当出现冲突时, 就会顺序的向下一个单元探测, 直到单元没有发生冲突

优点: 简单 容易实现

缺点: 会出现聚集现象, 降低查找效率

平方探测法

$$d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \dots, k^2, -k^2$$

优点: 可以避免出现"堆积"问题,

缺点: 不能探测到散列表上的所有单元,但至少能探测到一半单元

增量的取值方法

再散列法

$$d_i = \text{Hash}_2(\text{key})$$

当通过第一个散列函数 $H(\text{key})$ 得到的地址发生冲突时,则利用第二个散列函数 $\text{Hash}_2(\text{key})$ 计算该关键字的地址增量

$$\text{计算公式公式 } H_i = (H(\text{key}) + i \times \text{Hash}_2(\text{key})) \% m$$

伪随机序列法

$d_i =$ 伪随机数序列时,称为伪随机序列法

拉链法

把所有的同义词存储在一个线性链表中,这个线性链表由其散列地址唯一标识

所有同义词就像被拉链串在一起一样

散列查找及性能分析

实现过程

①检测查找表中地址为Addr的位置上是否有记录,若无记录, 返回查找失败; 若有记录, 比较它与key的值, 若相等, 则返回查找成功标志, 否则执行步骤②

②用给定的处理冲突方法计算"下一个散列地址",并把Addr置为此地址,转入步骤①

特点

平均查找长度作为衡量散列表的查找效率的度量

散列表的查找效率取决于三个因素:散列函数、处理冲突的方法和装填因子

$$\alpha = \frac{\text{表中记录数 } n}{\text{散列表长度 } m}$$

装填因子

装填因子越大越容易发生冲突