1 检索:找到 关键码值=给定值的记录或记录集, 效率非常重要
2 平均检索长度: ASL = sum{pi * ci}
3 预排序->建立索引->散列技术
4 顺序检索:逐个进行比较,可以顺序存储或者链式存储, 不需要排序, 可以设立监视哨, 检索成功 ASL = (n + 1) / 2, 失败 n + 1
5 插入元素直接放在表尾,检索时间太长
6 二分检索:待检索序列有序, 最长检索长度[log(n + 1)](向上取整), 失败的检索长度[log(n + 1)](向上/下取整), 平均检索长度 log(n + 1) - 1
7 平均检索长度与最大检索长度相近,但需要排序,顺序存储,不易更新
8 分块检索:按块有序, 快内无序,每一块不一定满
9 索引表:各块的最大关键码,起始位置,有效元素个数, 索引表是一个递增有序表,分块有序
10 分块检索 = 索引表检索(可以二分检索,也可以顺序检索,因为不大) + 块内检索
11 如果都用顺序存储,块大小等于 sqrt(n)时,ASL 取最小值
12 插入删除容易,没有大量记录移动,但需要增加辅助的索引表,初始的线性表分块排序,元素大量增删,分布不均匀时性能下降
13 散列检索:hash, 建立起关键码与存储地址之间的直接映射关系
14 装载因子:已有节点数 / 散列表空间大小, 冲突->不同的关键码(同义词)映射到相同的散列地址
15 散列函数的构造:结点均匀分布,减少冲突的概率,运算简单,函数值在散列表范围内
16 常用散列函数选取方法:除余法(通常取质数), 乘余取整(乘一个小数,取小数部分,乘上散列表长度),平方取中(平方,取中间几位,或中间几位的组合, 统计:最接近于随机化), 数字分析(选取符号均匀分布的若干位,person 卡方检验,完全依赖于关键码集合), 基数转换(看成另一个进制上的数,再转换回来,取若干位为散列地址,一般区更大的互素的数作为转换的基数), 折叠(关键码分成位数相同的几部分,取几个部分的叠加和,不进位, 移位叠加<-实质上是平移/分界叠加<-真的去折叠), elfhash
17 冲突的解决办法:开散列办法:拉链法,所有同义词连接在一个链表上,这时装载因子可以大于 1, 闭散列方法:把发生冲突的关键码放在另一个空地址内
18 开散列方法:拉链方法(内存),把散列的每一个槽看成一个链表的表头,同义词表的组织(输入顺序/访问频率的顺序/值的顺序<-适用于检索不成功的情况), 适用于表长不确定的情况,删除结点易于实现,但是在磁盘中不易实现
19 桶式散列:散列文件分成若干个桶,桶内各个页块用指针连接,每一个页块包含若干记录
20 如果桶数很小,桶目录表可以存放在内存,也可以放到外存
21 计算 h(i)得到桶号,掉第一个页块,顺序查找
22 闭散列方法:基地址,探查序列,探查函数, 将第一个空闲位置作为存储位置
23 假设至少有一个存储位置是空的
24 线性探查法(找后一个位置, 所有的位置都可以作为候选的位置, 产生聚集问题,导致很长的探查序列 ==> 改进的线性探查, 每一次跳过常数个槽, 但还是会纠缠在一起), 二次探查法(探查增量序列为 1, -1, 4, -4, 9, -9..., 两个探查序列很快就会分开), 伪随机数序列探查法(探查函数为 perm[i - 1], 从一个随机序列中取值), 双散列探查法
25 基本聚集:基地址不同的关键码,探查序列重叠在一起, 二次聚集:两个关键码散列到同一个基地址 ==> 双散列探查法(探查函数 p(k, i) = i * h2(k), h2 必须和 M(散列表大小)互素, 不易产生聚集, 计算量增大)
26 字典, 键值对二元组
27 散列表的插入:基地址没有占用,插入记录,否则向后探查,直到 有空位置/关键码相等

28 散列表的检索:如果基地址没有占用,检索失败,否则向后探查,直到找到/空位置,失败
29 散列表的删除: 删除记录不能影响后续检索,释放存储位置为将来所用, 只有开散列表才会真正删除,闭散列方法只能做标记
30 被删除标记值成为墓碑 -> 插入时遇到墓碑,不可以放进去:可能出现相同的关键码, 如果确定之后没有重复的元素,可以插入到第一个墓碑处
31 效率分析:插入与删除基于检索实现,效率与负载因子有关
32 经验:散列方法效率只依赖于负载因子,临界值是 0.5, 超过一半,性能急剧下降
33 插入删除操作频繁降低检索效率:负载因子增加/墓碑数量增加
34 插入删除频繁的情况可以定期重新散列