7.4散列查找：

一、散列表

**散列表**（Hash Table），⼜称**哈希表**。是⼀种数据结构。

特点是：数据元素的**关键字**与其**存储地址**直接相关。

二、如何建⽴“关键字” 与“存储地址” 的联系?

通过“**散列函数**（**哈希**函数）”：**Addr=H(key)**

三、同义词和冲突

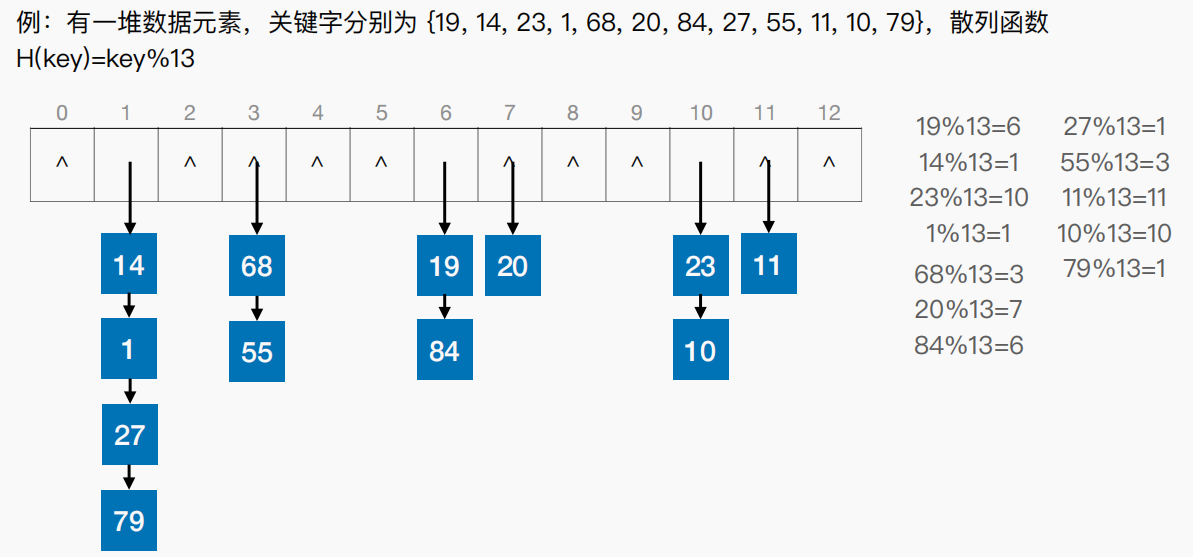
**同义词**：若不同的关键字通过散列函数映射到同⼀个值，则称它们为“同义词”

**冲突：**通过散列函数确定的位置已经存放了其他元素，则称这种情况为“冲突”

四、处理冲突的办法：

①拉链法：

⽤拉链法（⼜称链接法、链地址法）处理“冲突”：把**所有“同义词”存储在⼀个链表**中



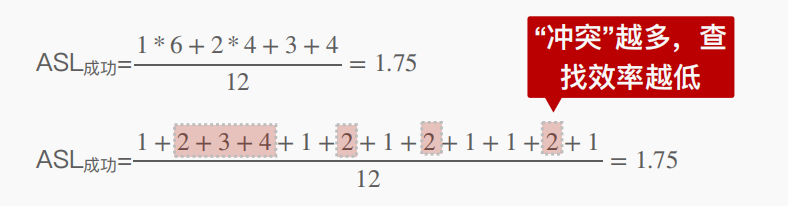
查找⻓度——在查找运算中，需要**对⽐关键字的次数**称为查找⻓度

查找⽬标： 27 27%13=1 27的查找⻓度=3

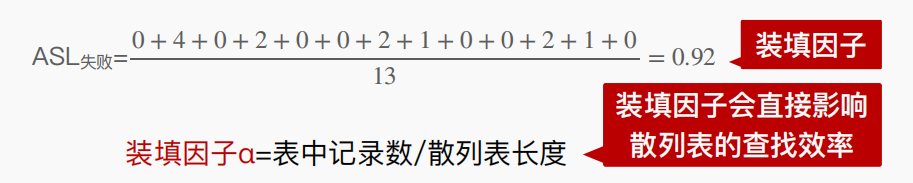
查找⽬标： 21 21%13=8 27的查找⻓度=**0**

查找⽬标： 66 66%13=1 27的查找⻓度=4

查找成功的平均查找长度：

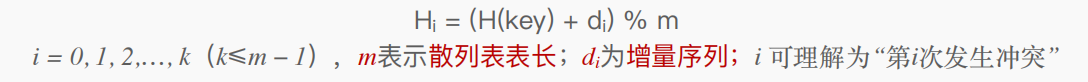


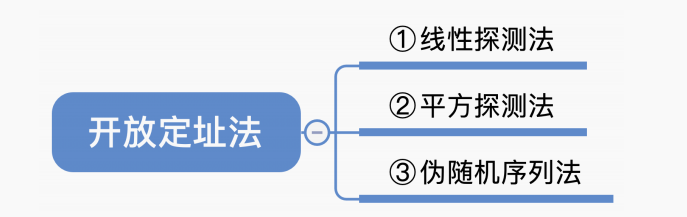
查找失败的平均查找长度：



②处理冲突的⽅法——**开放定址**法

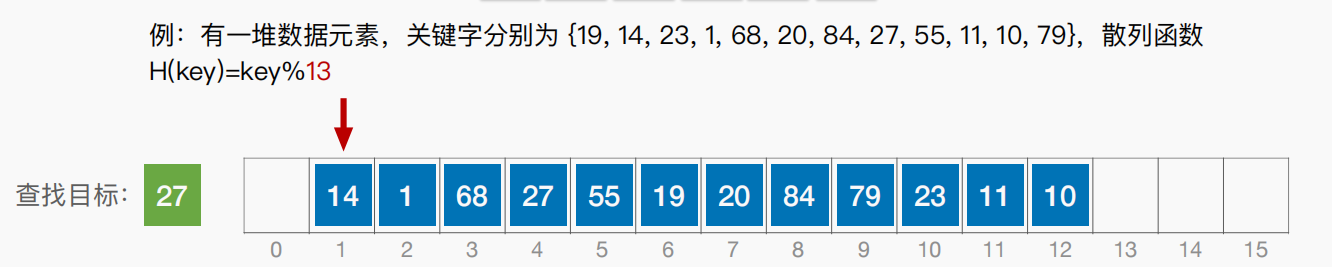
所谓开放定址法，是指可存放新表项的空闲地址既向它的同义词表项开放，⼜向它的⾮ 同义词表项开放。其数学递推公式为：

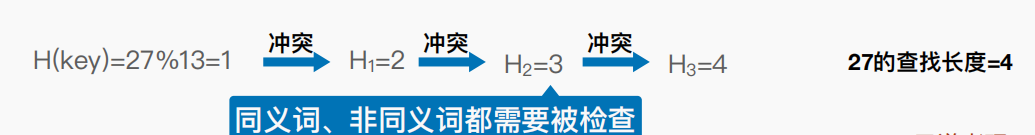


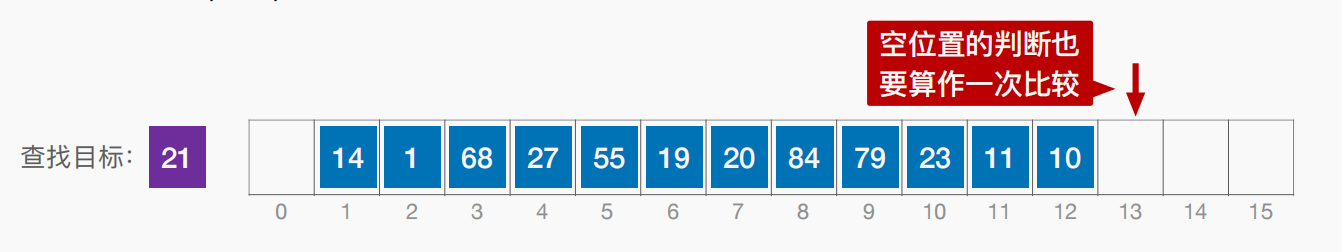


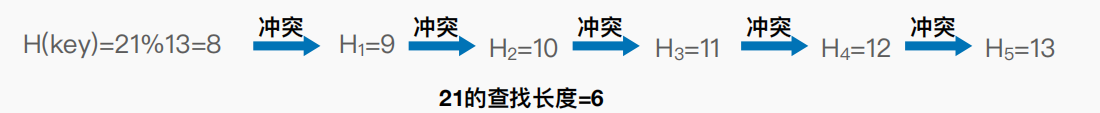
1、线性探测法—— di = 0, 1, 2, 3, …, m-1；即发⽣**冲突**时，每**次往后探测相邻的下⼀个单 元是否为空**

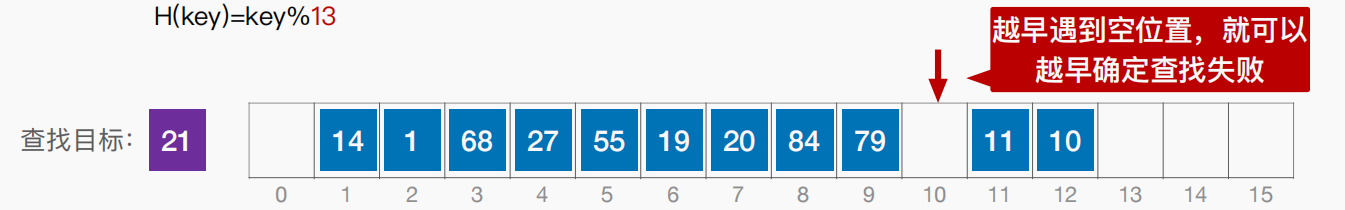
（1）查找操作









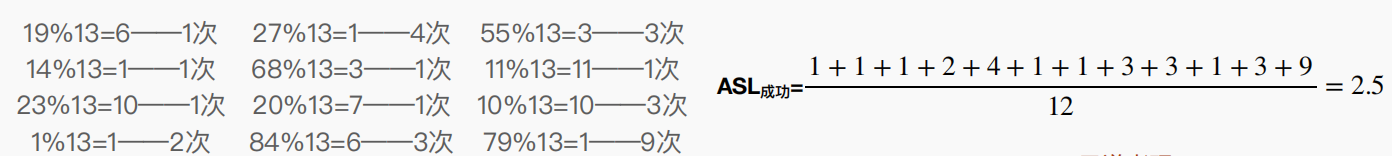


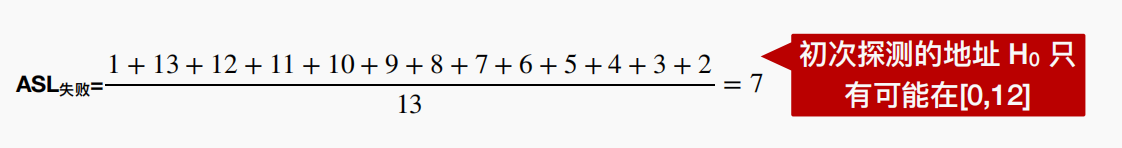


1. 删除操作

注意：采⽤“开放定址法”时，删除结点**不能简单地将被删结点的空间置为空**，否则**将截断在它之后填⼊散列表的同义词结点的查找路径**，可以做⼀个“**删除标记**”，进⾏**逻辑删除**。

1. 平均查询长度

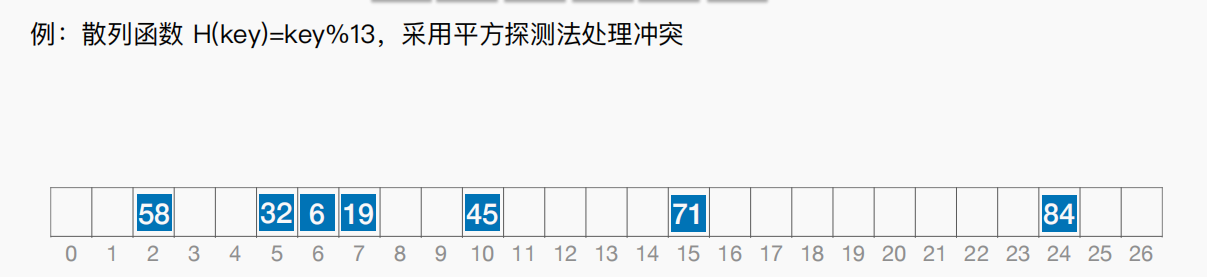




效率较低的原因：线性探测法很容易造成**同义词、⾮同义词的“聚集（堆积）”现象**，严重影响查找效率 (产⽣原因——冲突后再探测⼀定是放在某个连续的位置)

1. 平方探测法



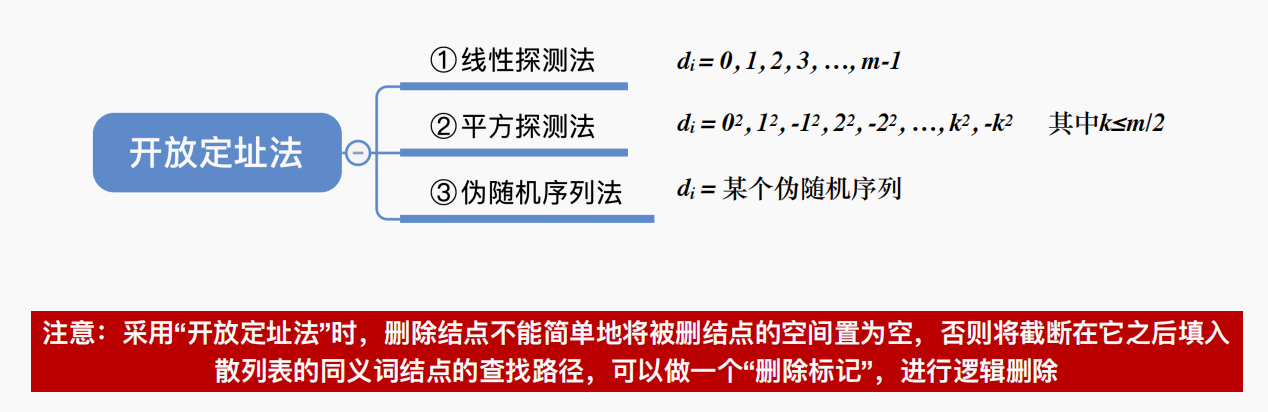


平⽅探测法：⽐起线性探测法更不易产⽣“聚集（堆积）”问题

⾮重点⼩坑：散列表⻓度m必须是⼀个可以表示成**4j + 3的素数**，才能探测到所有位置

1. 伪随机序列法





③处理冲突的⽅法——**再散列**法

再散列法（**再哈希法**）：除了原始的散列函数 H(key) 之外，多准备⼏个散列函数，当 散列函数冲突时，⽤下⼀个散列函数计算⼀个新地址，直到不冲突为⽌：

Hi = RHi(Key) i=1,2,3….,k

五、常⻅的散列函数设计⽬标——让**不同关键字**的**冲突尽可能地少**：

1.**除留余数**法 —— **H(key) = key % p**

（散列表表⻓为m，取⼀个**不⼤于m**但**最接近或等于m的质数p**）

2.**直接定址**法 —— **H(key) = key 或 H(key) = a\*key + b**

（这种⽅法计算最简单，且**不会产⽣冲突**。它适合**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）

3.**数字分析**法 —— 选取**数码分布较为均匀的若⼲位**作为散列地址

（设关键字是r进制数（如⼗进制数），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，此时可选取数码分布较为均匀的若⼲位作为散列地址。这种⽅法适合于已知的关键字集合，若更换了关键字，则需要重新构造新的散列函数。）

4.**平⽅取中**法——取**关键字的平⽅值的中间⼏位**作为散列地址

（具体取多少位要视实际情况⽽定。这种⽅法得到的**散列地址与关键字的每位都有关系**，因此使得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**。）

六、总结：

**散列查找**是典型的“**⽤空间换时间**”的算法，只要散列函数设计的合理，则散列表越

⻓，冲突的概率越低。

