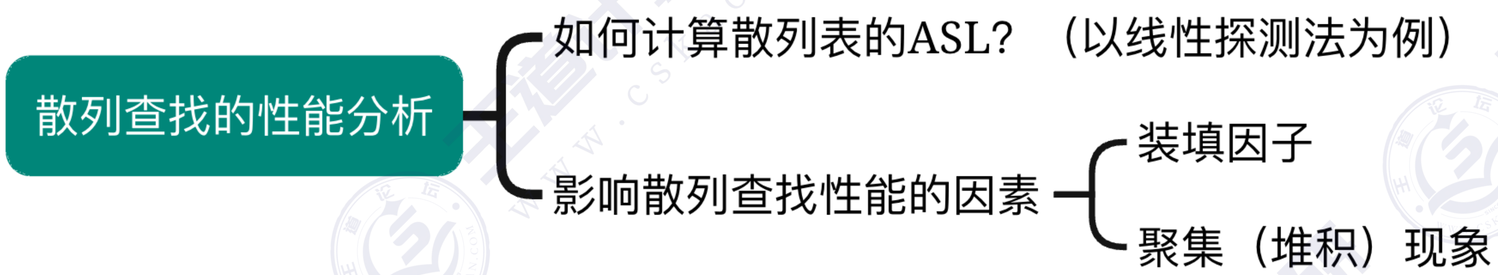
本节内容

散列查找的性能分析

以线性探测法为例

知识总览

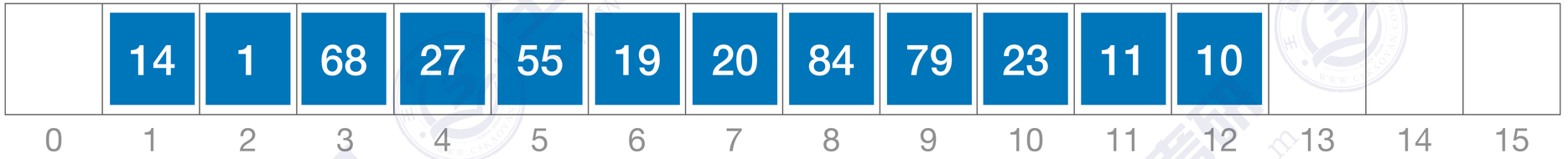


散列查找的性能分析

例：设散列表HT长度为16，初始为空。散列函数 H(key)=key%13，处理冲突采用线性探测法。依次插入关键字序列 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}

（1）请画出所构造的散列表

（2）请计算在等概率情况下，查找成功、查找失败的平均查找长度ASL。



计算每个关键字的初始散列地址，若发生冲突，采用线性探测法解决冲突：

H(19) = 19%13 = 6
H(14) = 14%13 = 1
H(23) = 23%13 = 10
H(1) = 1%13 = 1 →2
H(68) = 68%13 = 3
H(20) = 20%13 = 7

H(84) = 84%13 = 6 →7 →8
H(27) = 27%13 = 1 →2 →3 →4
H(55) = 55%13 = 3 →4 →5
H(11) = 11%13 = 11
H(10) = 10%13 = 10 →11 →12
H(79) = 79%13 = 1 →2 →3 →4 →5 →6 →7 →8 →9

$$ASL_{成功}=\frac{1+1+1+2+1+1+3+4+3+1+3+9}{12}=2.5$$

计算“查找成功”的ASL：假设查找每一个已存在关键字的概率相同

散列查找的性能分析

例：设散列表HT长度为16，初始为空。散列函数 H(key)=key%13，处理冲突采用线性探测法。依次插入关键字序列 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}

（1）请画出所构造的散列表

（2）请计算在等概率情况下，查找成功、查找失败的平均查找长度ASL。

Tips：任何一个关键字的初始散列地址都不可能落在13~15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	14	1	68	27	55	19	20	84	79	23	11	10			

啊～对对对
太✅辣

$$ASL_{失败}=\frac{1+13+12+11+10+9+8+7+6+5+4+3+2}{13}=7$$

计算“查找失败”的ASL：假设出现每一个初始散列地址的概率相同

错错错，是我的错

$$ASL_{失败}=\frac{1+13+12+11+10+9+8+7+6+5+4+3+2+1+1+1}{16}$$

经典错误！注意：虽然散列表长度为16，但散列函数的取值范围仅有 13 种可能性

散列查找的性能分析

例：设散列表HT长度为16，初始为空。散列函数 H(key)=key%13，处理冲突采用线性探测法。依次插入关键字序列 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}

（3）若删除元素20，请再计算在等概率情况下，查找成功、查找失败的平均查找长度ASL。

H(19) = 19%13 = 6
H(14) = 14%13 = 1
H(23) = 23%13 = 10
H(1) = 1%13 = 1 →2
H(68) = 68%13 = 3
H(84) = 84%13 = 6 →7 →8
H(27) = 27%13 = 1 →2 →3 →4
H(55) = 55%13 = 3 →4 →5
H(11) = 11%13 = 11
H(10) = 10%13 = 10 →11 →12
H(79) = 79%13 = 1 →2 →...→9

$$ASL_{成功}=\frac{1+1+1+2+1+3+4+3+1+3+9}{11}=2.63$$

$$ASL_{失败}=\frac{1+13+12+11+10+9+8+7+6+5+4+3+2}{13}=7$$

注意！注意！

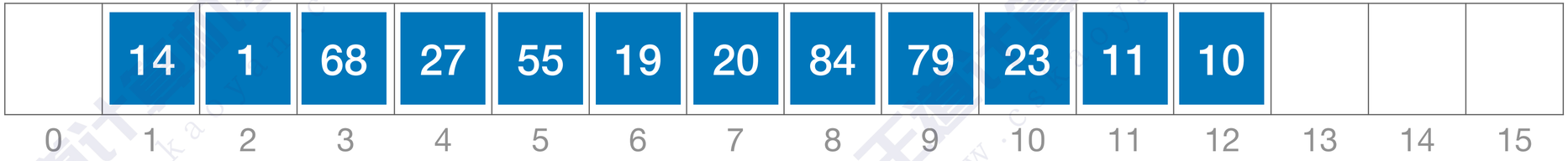
注意易错点：采用开放定址法时，删除一个元素是逻辑删除，而非物理删除！这一点将影响查找长度的计算

装填因子

散列表的装填因子 $\alpha = \dfrac{\text{表中记录数}n}{\text{散列表长度}m}$

反映了一个散列表“满”的程度

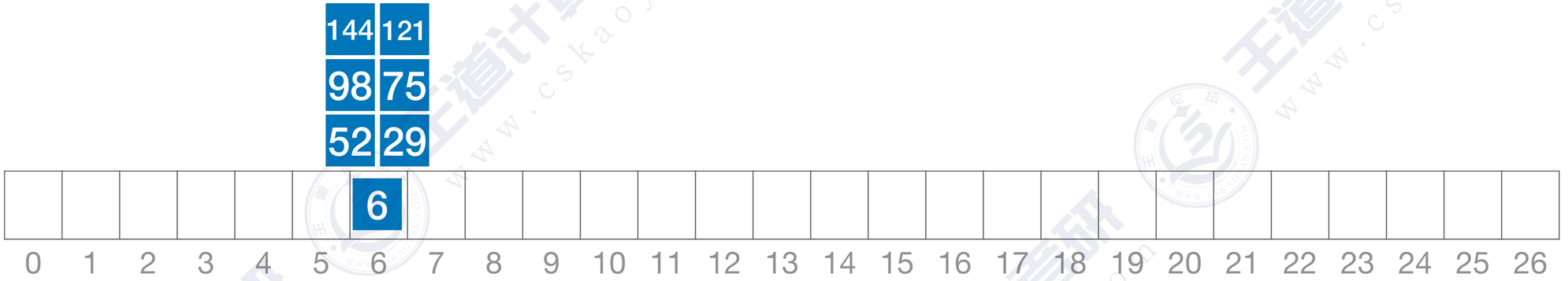
显然，**装填因子越大，越容易发生冲突。从而导致插入、查找操作效率降低，ASL增大**



例：如上所示，散列表HT长度为16，散列函数 H(key)=key%13，其中已插入12个元素，则装填因子 α = 12/16 = 0.75

聚集（堆积）现象

例：散列表长度为27，散列函数 H(key)=key%23，采用线性探测法处理冲突



聚集（堆积）现象

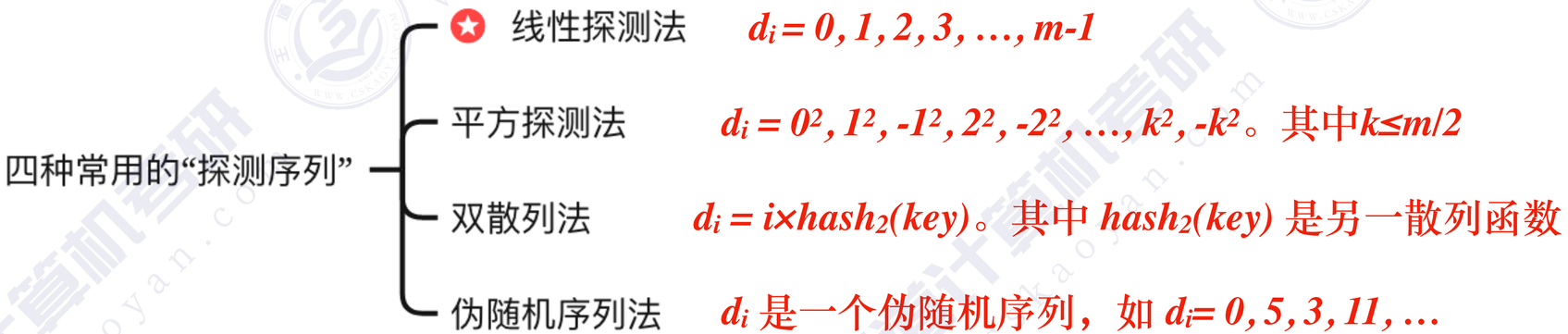
例：散列表长度为27，散列函数 H(key)=key%23，采用线性探测法处理冲突

聚集（堆积）现象：在处理冲突的过程中，几个初始散列地址不同的元素争夺同一个后继散列地址的现象称作“聚集”（或称作“堆积”）

线性探测法在发生冲突时，总是往后探测相邻的后一个单元，很容易造成同义词、非同义词的“聚集（堆积）现象”，从而影响查找效率，导致ASL提升

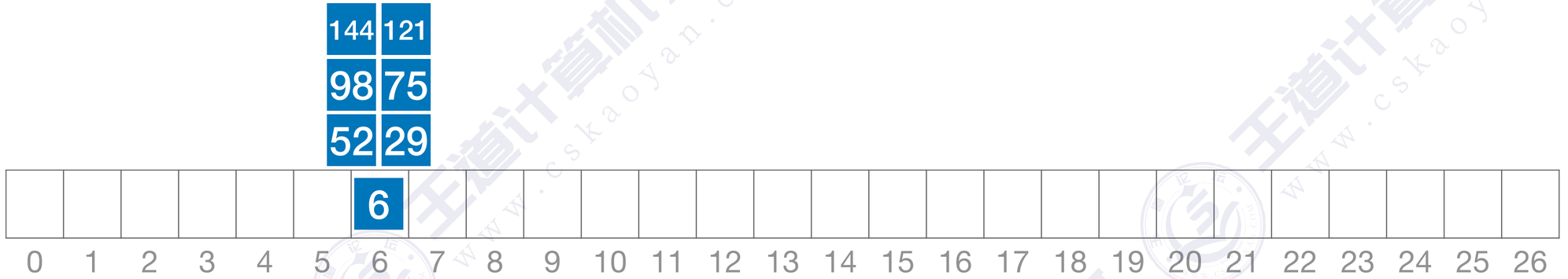
回顾开放定址法的几种常用“探测序列”

开放定址法：$H_i = (H(key) + d_i) \% m$，其中，$H_i$ 表示第 i 次发生冲突时的散列地址；$i = 0, 1, 2, \ldots, k$（$k \leqslant m-1$），m表示散列表表长；d_i为增量序列；



使用“平方探测法”减少聚集现象

例：散列表长度为27，散列函数 H(key)=key%23，采用平方探测法（二次探测法）处理冲突



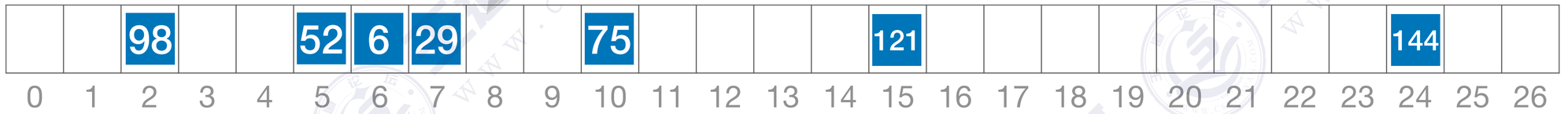
平方探测法（又称二次探测法）： $d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \ldots, k^2, -k^2$，其中$k \le m/2$

$d_0 = 0$，$d_1 = 1$，$d_2 = -1$，$d_3 = 4$，$d_4 = -4$，$d_5 = 9$，$d_6 = -9$….

使用“平方探测法”减少聚集现象

例：散列表长度为27，散列函数 H(key)=key%23，采用平方探测法（二次探测法）处理冲突

散列地址 #8，将被初始散列地址为7、8 的元素争夺。显然，“聚集”现象没有线性探测法那么严重。



平方探测法（又称二次探测法）：$d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \ldots, k^2, -k^2$，其中$k \le m/2$

$d_0 = 0$，$d_1 = 1$，$d_2 = -1$，$d_3 = 4$，$d_4 = -4$，$d_5 = 9$，$d_6 = -9$….

结论：采用线性探测法处理冲突更容易导致“聚集（堆积）现象”，用其他方法处理冲突可以将元素“打散”，从而减少“聚集（堆积）现象”

知识回顾与重要考点

散列查找的性能分析

- **如何计算散列表的ASL**
  - $\mathrm{ASL}_{成功}=\frac{1}{n}\sum_{i=1}^{n}C_i$ —— 其中，n为散列表中已存在的元素个数，C_i为成功查找第 i 个元素所需的比较次数
  - $\mathrm{ASL}_{失败}=\frac{1}{r}\sum_{i=1}^{r}C_i$ —— 其中，r 为散列函数取值的个数，C_i为散列函数取值为 i 时查找失败的比较次数
    - 特别注意：散列函数取值的个数、散列表长度可能不相同
- **装填因子**
  - $\alpha=\frac{表中记录数n}{散列表长度m}$ —— 反映了一个散列表“满”的程度
  - 装填因子越大，越容易发生冲突。从而导致插入、查找效率降低，ASL增大
- 聚集（堆积）现象
  - 在处理冲突的过程中，几个初始散列地址不同的元素争夺同一个后继散列地址的现象称作“聚集”（或称作“堆积”）
  - 聚集（堆积）现象会影响查找效率，导致ASL提升
  - 采用线性探测法处理冲突更容易导致“聚集（堆积）现象”，用其他方法处理冲突可以将元素“打散”，从而减少“聚集（堆积）现象”