

本节内容

# 散列查找

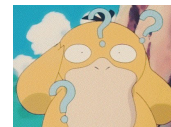
王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

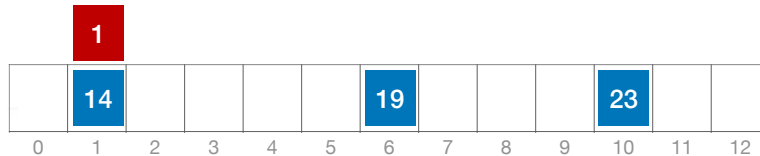
## 前情回顾

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）”： $\text{Addr} = H(\text{key})$



例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



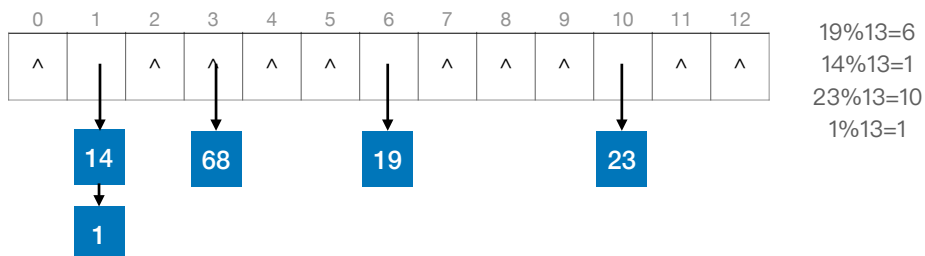
$19 \% 13 = 6$   
 $14 \% 13 = 1$   
 $23 \% 13 = 10$   
 $1 \% 13 = 1$

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**”  
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**”

王道考研/CSKAOYAN.COM

## 处理冲突的方法——拉链法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



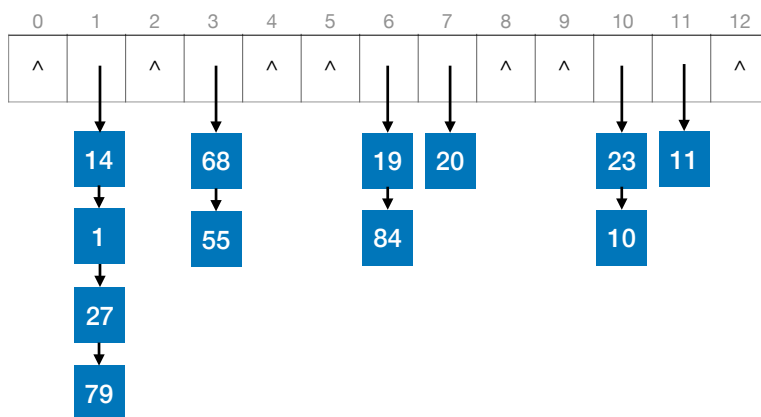
用**拉链法**（又称链接法、链地址法）处理“冲突”：把所有“同义词”存储在一个链表中

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 处理冲突的方法——拉链法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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

王道考研/CSKAOYAN.COM

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

	14					19				23					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

开放定址法

① 线性探测法

② 平方探测法

③ 伪随机序列法

王道考研/CSKAOYAN.COM

公众号： 考研发条 一手课程！

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

	1														
	14					19				23					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$$H(\text{key}) = 1 \% 13 = 1$$

$$H_0 = (1 + d_0) \% 16 = 1$$

冲突

$$H_1 = (1 + d_1) \% 16 = 2$$

发生第1次冲突后重新计算得到的哈希地址

王道考研/CSKAOYAN.COM

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

	14	1				19				23					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$$H(\text{key}) = 1 \% 13 = 1$$

$$H_0 = (1 + d_0) \% 16 = 1$$

冲突

$$H_1 = (1 + d_1) \% 16 = 2$$

发生第1次冲突后重新计算得到的哈希地址

王道考研/CSKAOYAN.COM

公众号： 考研发条

一手课程！

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

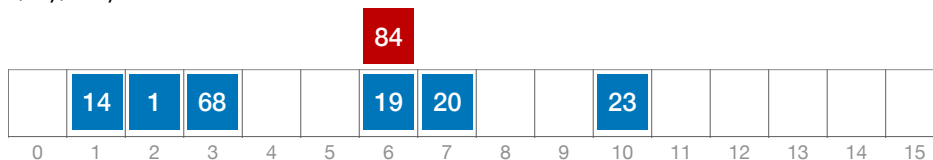
$$H(\text{key}) = 68 \% 13 = 3$$

$$20 \% 13 = 7$$

王道考研/CSKAOYAN.COM

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

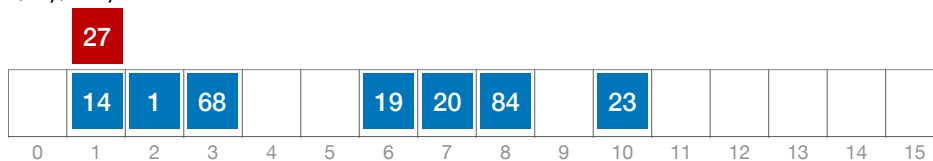
$$H(\text{key}) = 84 \% 13 = 6 \quad H_0 = (6 + 0) \% 16 = 6 \xrightarrow{\text{冲突}} H_1 = (6 + 1) \% 16 = 7 \xrightarrow{\text{冲突}} H_2 = 8$$

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$$H(\text{key}) = 27 \% 13 = 1 \xrightarrow{\text{冲突}} H_1 = 2 \xrightarrow{\text{冲突}} H_2 = 3 \xrightarrow{\text{冲突}} H_3 = 4$$

王道考研/CSKAOYAN.COM

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 55 \% 13 = 3$     **冲突** →  $H_1 = 4$     **冲突** →  $H_2 = 5$

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 11 \% 13 = 11$

王道考研/CSKAOYAN.COM

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 10 \% 13 = 10$  **冲突**  $\rightarrow H_1 = 11$  **冲突**  $\rightarrow H_2 = 12$

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 79 \% 13 = 1$  **冲突**  $\rightarrow H_1 = 2$  **冲突**  $\rightarrow H_2 = 3$  ... **冲突**  $\rightarrow H_8 = 9$

王道考研/CSKAOYAN.COM

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

												25			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空



$$H(25) = 25 \% 13 = 12$$

$$H_1 = (H(\text{key}) + 1) \% 16 = 13$$

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 处理冲突的方法——开放定址法

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

	14	1	68	27	55	19	20	84	79	23	11	10	25		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空



$$H(25) = 25 \% 13 = 12$$

$$H_1 = (H(\text{key}) + 1) \% 16 = 13$$

哈希函数值域[0,12]

冲突处理函数值域[0,15]

王道考研/CSKAOYAN.COM



## 查找操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：

27

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

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$$H(\text{key}) = 27 \% 13 = 1$$

冲突

$$H_1 = 2$$

冲突

$$H_2 = 3$$

冲突

$$H_3 = 4$$

27的查找长度=4

同义词、非同义词都需要被检查



白眼

王道考研/CSKAOYAN.COM

公众号： 考研发条

一手课程！

## 查找操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：

11

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

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$$H(\text{key}) = 11 \% 13 = 11$$

11的查找长度=1

王道考研/CSKAOYAN.COM

## 查找操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：21

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

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 21 \% 13 = 8$     冲突 →  $H_1 = 9$     冲突 →  $H_2 = 10$     冲突 →  $H_3 = 11$     冲突 →  $H_4 = 12$     冲突 →  $H_5 = 13$   
21的查找长度=6

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 查找操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：21

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

空位置的判断也要算作一次比较

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 21 \% 13 = 8$     冲突 →  $H_1 = 9$     冲突 →  $H_2 = 10$     冲突 →  $H_3 = 11$     冲突 →  $H_4 = 12$     冲突 →  $H_5 = 13$   
21的查找长度=6

王道考研/CSKAOYAN.COM

## 查找操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：21

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

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 21 \% 13 = 8$  **冲突**  $\rightarrow H_1 = 9$  **冲突**  $\rightarrow H_2 = 10$

王道考研/CSKAOYAN.COM

公众号： 考研发条

一手课程！

## 查找操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：21

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

越早遇到空位置，就可以  
越早确定查找失败

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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

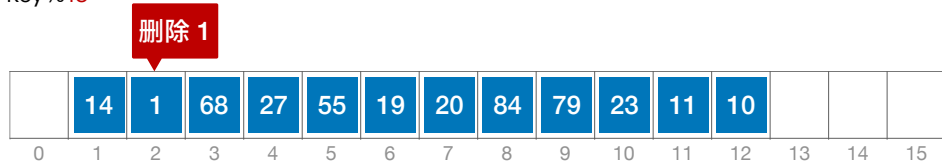
①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 21 \% 13 = 8$  **冲突**  $\rightarrow H_1 = 9$  **冲突**  $\rightarrow H_2 = 10$  **21的查找长度=3**

王道考研/CSKAOYAN.COM

## 删除操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

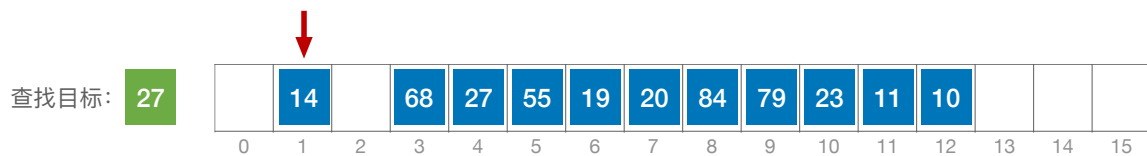
①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 删除操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 27 \% 13 = 1$       **冲突** →  $H_1 = 2$

王道考研/CSKAOYAN.COM

## 删除操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m - 1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

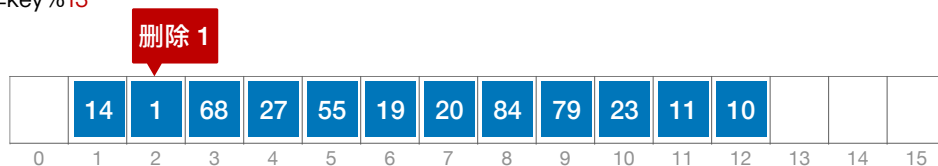
$H(\text{key}) = 27 \% 13 = 1$  冲突  $\rightarrow H_1 = 2$

王道考研/CSKAOYAN.COM

公众号： 考研发条 一手课程！

## 删除操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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：

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

$i = 0, 1, 2, \dots, k \ (k \leq m - 1)$ ， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m - 1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

**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

王道考研/CSKAOYAN.COM

## 删除操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：



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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 27 \% 13 = 1$  **冲突**  $\rightarrow H_1 = 2$  **冲突**  $\rightarrow H_2 = 3$  **冲突**  $\rightarrow H_3 = 4$  **27的查找长度=4**

王道考研/CSKAOYAN.COM

公众号： 考研发条

一手课程！

## 删除操作

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

查找目标：



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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示**散列表表长**； $d_i$ 为**增量序列**； $i$ 可理解为“第*i*次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

$H(\text{key}) = 79 \% 13 = 1$  **冲突**  $\rightarrow H_1 = 2$  **冲突**  $\rightarrow H_2 = 3$  ... **冲突**  $\rightarrow H_8 = 9$



王道考研/CSKAOYAN.COM

## 查找效率分析 (ASL)

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

19%13=6——1次    27%13=1——4次    55%13=3——3次  
14%13=1——1次    68%13=3——1次    11%13=11——1次  
23%13=10——1次    20%13=7——1次    10%13=10——3次  
1%13=1——2次    84%13=6——3次    79%13=1——9次

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

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 查找效率分析 (ASL)

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

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

初次探测的地址  $H_0$  只  
有可能在  $[0, 12]$

王道考研/CSKAOYAN.COM

## 查找效率分析 (ASL)

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

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

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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

①**线性探测法**—— $d_i = 0, 1, 2, 3, \dots, m-1$ ；即发生冲突时，每次往后探测相邻的下一个单元是否为空

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

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 处理冲突的方法——开放定址法

例：散列函数  $H(\text{key}) = \text{key} \% 13$ ，采用平方探测法处理冲突

						84	71																						
						58	45																						
						32	19																						
						6																							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26			

**开放定址法**： $H_i = (H(\text{key}) + d_i) \% m$

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示散列表表长； $d_i$ 为增量序列；

②**平方探测法**。当 $d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \dots, k^2, -k^2$ 时，称为平方探测法，又称**二次探测法**其中 $k \leq m/2$

$$d_0 = 0$$

$$d_1 = 1$$

$$d_2 = -1$$

$$d_3 = 4$$

$$d_4 = -4$$

$$d_5 = 9$$

$$d_6 = -9$$

....

王道考研/CSKAOYAN.COM



## 处理冲突的方法——开放定址法

例：散列函数  $H(\text{key}) = \text{key} \% 13$ ，采用平方探测法处理冲突

		58			32	6	19			45				71										84		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

开放定址法：  $H_i = (H(\text{key}) + d_i) \% m$

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示散列表表长； $d_i$ 为增量序列；

②平方探测法。当 $d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \dots, k^2, -k^2$ 时，称为平方探测法，又称二次探测法其中 $k \leq m/2$

$$d_0 = 0$$

$$d_1 = 1$$

$$d_2 = -1$$

$$d_3 = 4$$

$$d_4 = -4$$

$$d_5 = 9$$

$$d_6 = -9$$

....

注意负数的模运算， $(-3) \% 27 = 24$ ，而不是3

《数论》中模运算的规则： $a \text{ MOD } m == (a + km) \text{ MOD } m$ ，其中， $k$ 为任意整数

王道考研/CSKAOYAN.COM

公众号： 考研发条 一手课程！

## 处理冲突的方法——开放定址法

例：散列函数  $H(\text{key}) = \text{key} \% 13$ ，采用平方探测法处理冲突

查找目标：71

		58			32	6	19			45				71										84		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

开放定址法：  $H_i = (H(\text{key}) + d_i) \% m$

$i = 0, 1, 2, \dots, k$  ( $k \leq m - 1$ )， $m$ 表示散列表表长； $d_i$ 为增量序列；

②平方探测法。当 $d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \dots, k^2, -k^2$ 时，称为平方探测法，又称二次探测法其中 $k \leq m/2$

$$d_0 = 0$$

$$d_1 = 1$$

$$d_2 = -1$$

$$d_3 = 4$$

$$d_4 = -4$$

$$d_5 = 9$$

$$d_6 = -9$$

....

平方探测法：比起线性探测法更不易产生“聚集（堆积）”问题

王道考研/CSKAOYAN.COM

## 处理冲突的方法——开放定址法

表长  $m=7$

若当前  $H(\text{key})=5$

0	1	2	3	4	5	6

经过7次探测可以探测到所有位置

$d_0 = 0 \rightarrow H_0 = 5$   
 $d_1 = 1 \rightarrow H_1 = 6$   
 $d_2 = -1 \rightarrow H_2 = 4$   
 $d_3 = 4 \rightarrow H_3 = 2$   
 $d_4 = -4 \rightarrow H_4 = 1$   
 $d_5 = 9 \rightarrow H_5 = 0$   
 $d_6 = -9 \rightarrow H_6 = 3$

表长  $m=8$

0	1	2	3	4	5	6	7

经过8次探测不能探测到所有位置

$d_0 = 0 \rightarrow H_0 = 5$   
 $d_1 = 1 \rightarrow H_1 = 6$   
 $d_2 = -1 \rightarrow H_2 = 4$   
 $d_3 = 4 \rightarrow H_3 = 1$   
 $d_4 = -4 \rightarrow H_4 = 1$   
 $d_5 = 9 \rightarrow H_5 = 6$   
 $d_6 = -9 \rightarrow H_6 = 4$   
 $d_7 = 16 \rightarrow H_7 = 5$

开放定址法:  $H_i = (H(\text{key}) + d_i) \% m$

$i = 0, 1, 2, \dots, k$  ( $k \leq m-1$ ),  $m$ 表示散列表表长;  $d_i$ 为增量序列;

②平方探测法。当 $d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \dots, k^2, -k^2$ 时, 称为平方探测法, 又称二次探测法其中 $k \leq m/2$

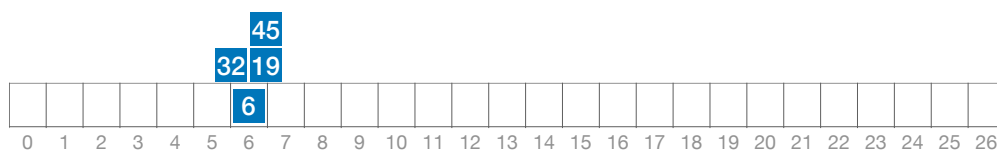
非重点小坑: 散列表长度 $m$ 必须是一个可以表示成 $4j+3$ 的素数, 才能探测到所有位置

王道考研/CSKAOYAN.COM

公众号： 考研发条 一手课程！

## 处理冲突的方法——开放定址法

例: 散列函数  $H(\text{key}) = \text{key} \% 13$ , 采用平方探测法处理冲突



开放定址法:  $H_i = (H(\text{key}) + d_i) \% m$

$i = 0, 1, 2, \dots, k$  ( $k \leq m-1$ ),  $m$ 表示散列表表长;  $d_i$ 为增量序列;

③伪随机序列法。 $d_i$ 是一个伪随机序列, 如  $d_i = 0, 5, 24, 11, \dots$

王道考研/CSKAOYAN.COM

## 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：

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

$i = 0, 1, 2, \dots, k$  ( $k \leq m-1$ )， $m$ 表示散列表表长； $d_i$ 为增量序列； $i$ 可理解为“第 $i$ 次发生冲突”

### 开放定址法

① 线性探测法

$$d_i = 0, 1, 2, 3, \dots, m-1$$

② 平方探测法

$$d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \dots, k^2, -k^2 \quad \text{其中 } k \leq m/2$$

③ 伪随机序列法

$d_i$  = 某个伪随机序列

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

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！

## 处理冲突的方法——再散列法

严蔚敏《数据结构》

**再散列法**（再哈希法）：除了原始的散列函数  $H(\text{key})$  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：

$$H_i = RH_i(\text{Key}) \quad i=1,2,3,\dots,k$$

王道《数据结构》

3) 再散列法。当  $d_i = \text{Hash}_2(\text{key})$  时，称为再散列法，又称双散列法。需要使用两个散列函数，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。它的具体散列函数形式如下：↵

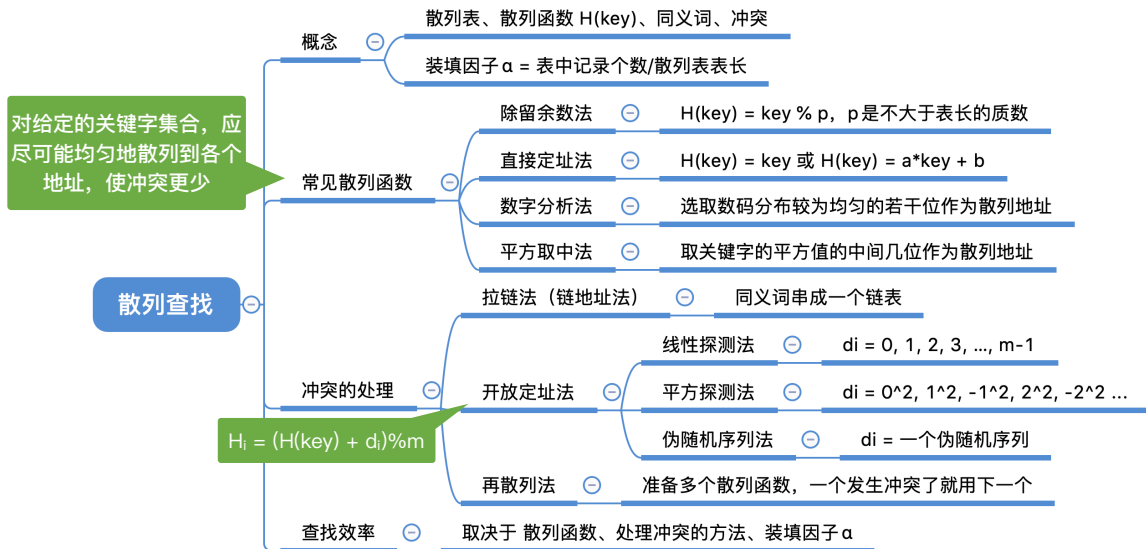
$$H_i = (H(\text{key}) + i \times \text{Hash}_2(\text{key})) \% m$$

初始探测位置  $H_0 = H(\text{key}) \% m$ 。  $i$  是冲突的次数，初始为 0。在散列法中，最多经过  $m-1$  次探测就会遍历表中所有位置，回到  $H_0$  位置。↵



王道考研/CSKAOYAN.COM

## 知识回顾与重要考点



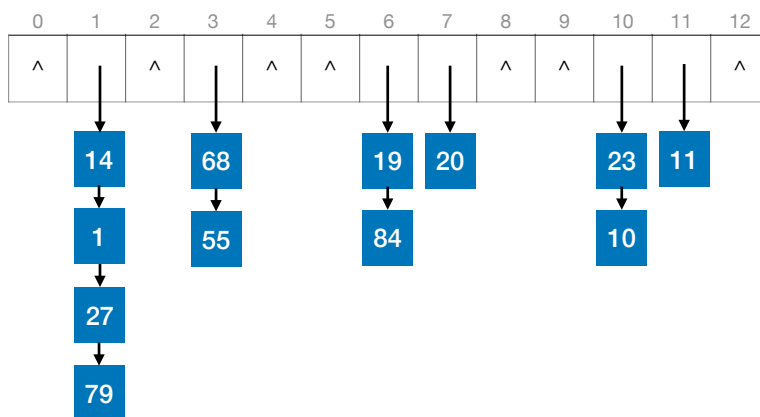
王道考研/CSKAOYAN.COM

公众号： 考研发条

一手课程！

## 拉链法的小优化

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$

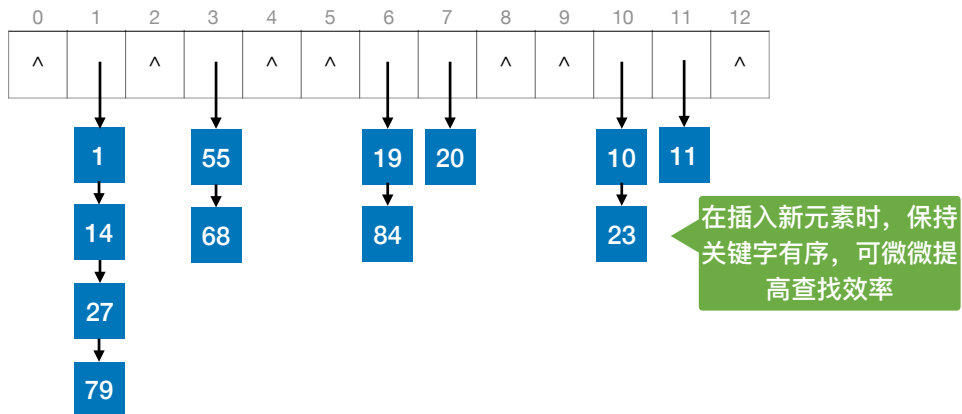


用**拉链法**（又称链接法、链地址法）处理“冲突”：把所有“同义词”存储在一个链表中

王道考研/CSKAOYAN.COM

## 拉链法的小优化

例：有一堆数据元素，关键字分别为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}，散列函数  $H(\text{key}) = \text{key} \% 13$



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

王道考研/CSKAOYAN.COM

公众号： 考研发条      一手课程！



@王道论坛



@王道计算机考研备考



@王道咸鱼老师-计算机考研

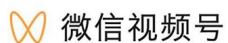
@王道楼楼老师-计算机考研



@王道计算机考研



@王道计算机考研



@王道计算机考研



@王道在线