

算法与数据结构体系课程

liuyubobobo

哈希表

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

从一个简单的问题开始

慕课网 算法与数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

实践：leetcode 387

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

什么是哈希表

int[26] freq 就是一个哈希表！

每一个字符都和一个索引相对应

a → 0

b → 1

c → 2

...

...

z → 25



$\text{index} = \text{ch} - \text{'a'}$

哈希函数

$f(\text{ch}) = \text{ch} - \text{'a'}$

O(1)的查找操作！

在哈希表上操作

哈希表

哈希函数 “键”转换为“索引”

$$f(ch) = ch - 'a'$$

一个班的学生学号：1-30

身份证号 110108198512166666

字符串

浮点数

日期

很难保证每一个“键”

通过哈希函数的转换

对应不同的“索引”

在哈希表上操作



解决哈希冲突



→ 哈希冲突

哈希表

哈希表充分体现了算法设计领域的经典思想：空间换时间

身份证号 110108198512166666

如果我们有 99999999999999999999 的空间，我们可以用 $O(1)$ 时间完成各项操作

如果我们有 1 的空间，我们只能用 $O(n)$ 时间完成各项操作（线性表）

哈希表是时间和空间之间的平衡

哈希表

哈希表充分体现了算法设计领域的经典思想：空间换时间

哈希表是时间和空间之间的平衡

哈希函数的设计是很重要的

“键”通过哈希函数得到的“索引”分布越均匀越好

哈希函数的设计

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

哈希函数的设计

“键”通过哈希函数得到的“索引”分布越均匀越好

对于一些特殊领域，有特殊领域的哈希函数设计方式
甚至有专门的论文

这个课程主要关注一般的哈希函数的设计原则

哈希函数的设计

整型

小范围正整数直接使用

小范围负整数进行偏移 $-100 \sim 100 \rightarrow 0 \sim 200$

大整数

身份证号 110108198512166666 \rightarrow 11010819851216**6666**

通常做法：取模

比如，取后四位。等同于 $\text{mod } 10000$

哈希函数的设计

大整数

身份证号 110108198512166666 \rightarrow 11010819851216**6666**

通常做法：取模 比如，取后四位。等同于 $\text{mod } 10000$

取后六位？ 等同于 $\text{mod } 1000000$ \rightarrow 110108198512**166666**

分布不均匀

具体问题具体分析

哈希函数的设计

大整数

取后六位？ 等同于 $\text{mod } 1000000$  110108198512**166666**

没有利用所有信息

一个简单的解决办法：模一个素数

哈希函数的设计

大整数

一个简单的解决办法：模一个素数 背后的数学理论超出课程范畴

$$10 \% 4 \longrightarrow 2$$

$$20 \% 4 \longrightarrow 0$$

$$30 \% 4 \longrightarrow 2$$

$$40 \% 4 \longrightarrow 0$$

$$50 \% 4 \longrightarrow 2$$

$$10 \% 7 \longrightarrow 3$$

$$20 \% 7 \longrightarrow 6$$

$$30 \% 7 \longrightarrow 2$$

$$40 \% 7 \longrightarrow 4$$

$$50 \% 7 \longrightarrow 1$$

哈希函数的设计

大整数

一个简单的解决办法：模一个素数

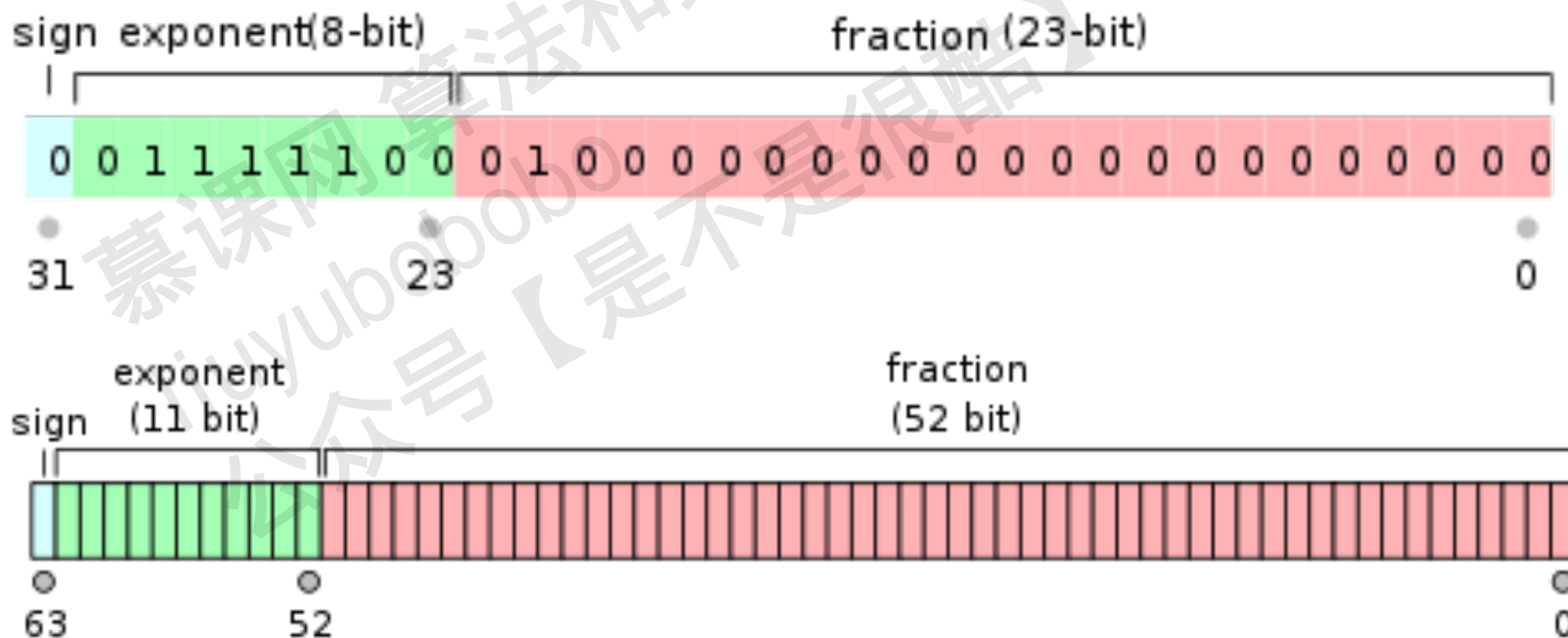
<http://planetmath.org/goodhashtableprimes>

lwr	upr	% err	prime
2^5	2^6	10.416667	53
2^6	2^7	1.041667	97
2^7	2^8	0.520833	193
2^8	2^9	1.302083	389
2^9	2^{10}	0.130208	769
2^{10}	2^{11}	0.455729	1543
2^{11}	2^{12}	0.227865	3079
2^{12}	2^{13}	0.113932	6151
2^{13}	2^{14}	0.008138	12289
2^{14}	2^{15}	0.069173	24593
2^{15}	2^{16}	0.010173	49157
2^{16}	2^{17}	0.013224	98317
2^{17}	2^{18}	0.002543	196613
2^{18}	2^{19}	0.006358	393241
2^{19}	2^{20}	0.000127	786433
2^{20}	2^{21}	0.000318	1572869
2^{21}	2^{22}	0.000350	3145739
2^{22}	2^{23}	0.000207	6291469
2^{23}	2^{24}	0.000040	12582917
2^{24}	2^{25}	0.000075	25165843
2^{25}	2^{26}	0.000010	50331653
2^{26}	2^{27}	0.000023	100663319
2^{27}	2^{28}	0.000009	201326611
2^{28}	2^{29}	0.000001	402653189
2^{29}	2^{30}	0.000011	805306457
2^{30}	2^{31}	0.000000	1610612741

哈希函数的设计

浮点型

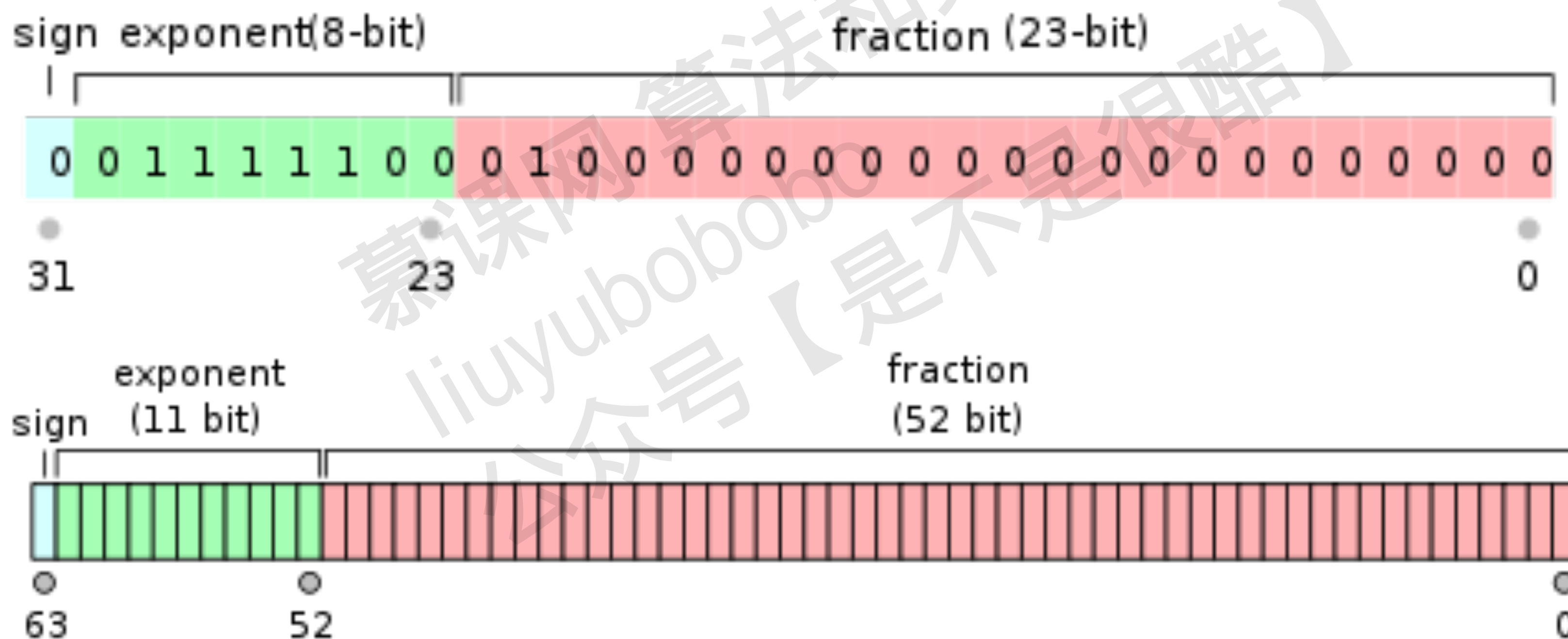
在计算机中都是32位或者64位的二进制表示，只不过计算机解析成了浮点数



哈希函数的设计

浮点型

在计算机中都是32位或者64位的二进制表示，只不过计算机解析成了浮点数



转成整型处理

哈希函数的设计

字符串 转成整型处理

$$166 = 1 * 10^2 + 6 * 10^1 + 6 * 10^0$$

$$\text{code} = c * 26^3 + o * 26^2 + d * 26^1 + e * 26^0$$

$$\text{code} = c * B^3 + o * B^2 + d * B^1 + e * B^0$$

$$\text{hash}(\text{code}) = (c * B^3 + o * B^2 + d * B^1 + e * B^0) \% M$$

哈希函数的设计

字符串

转成整型处理

$$\text{hash}(\text{code}) = (c * B^3 + o * B^2 + d * B^1 + e * B^0) \% M$$

$$\text{hash}(\text{code}) = (((c * B) + o) * B + d) * B + e \% M$$

$$\text{hash}(\text{code}) = (((((c \% M) * B + o) \% M * B + d) \% M * B + e) \% M$$

哈希函数的设计

字符串 转成整型处理

$$\text{hash}(\text{code}) = (((((c \% M) * B + o) \% M * B + d) \% M * B + e) \% M$$

```
int hash = 0
```

```
for(int i = 0 ; i < s.length() ; i ++)
```

```
    hash = (hash * B + s.charAt(i)) % M
```

哈希函数的设计

复合类型 转成整型处理

$$\text{hash}(\text{code}) = (((((c \% M) * B + o) \% M * B + d) \% M * B + e) \% M$$

Date: year, month, day

$$\text{hash}(\text{date}) = (((\text{date.year} \% M) * B + \text{date.month}) \% M * B + \text{date.day}) \% M$$

哈希函数的设计

转成整型处理 并不是唯一的方法!

原则

1. 一致性: 如果 $a == b$, 则 $\text{hash}(a) == \text{hash}(b)$
2. 高效性: 计算高效简便
3. 均匀性: 哈希值均匀分布

Java 中的 hashCode

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

实践：hashCode

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

实践：自定义类中的hashCode

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

实践：HashMap, HashSet 中使用自定义类

实践：没有hashcode和定义hashcode;
HashMap, HashSet 中使用自定义类区别

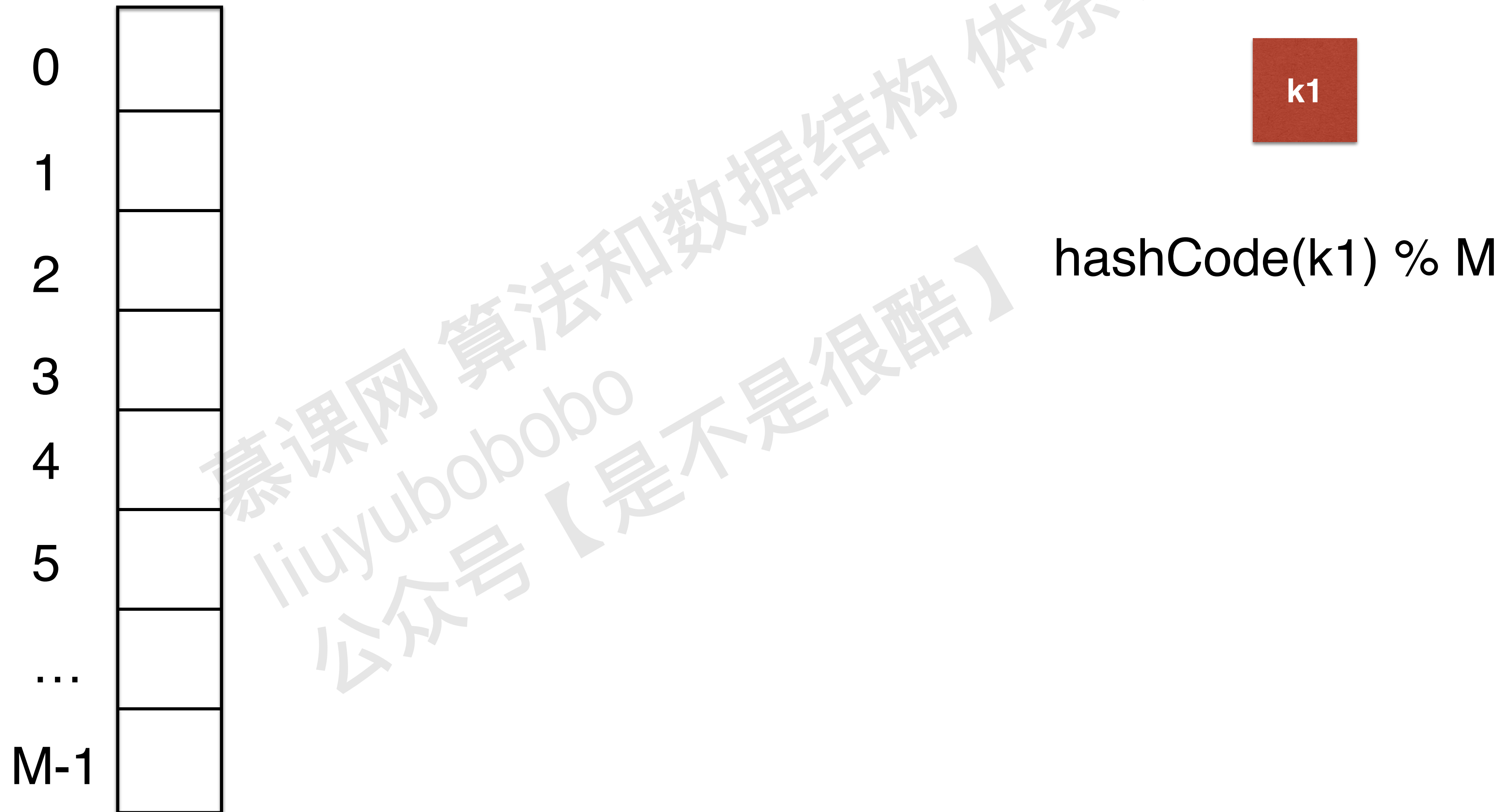
实践：实现equals

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

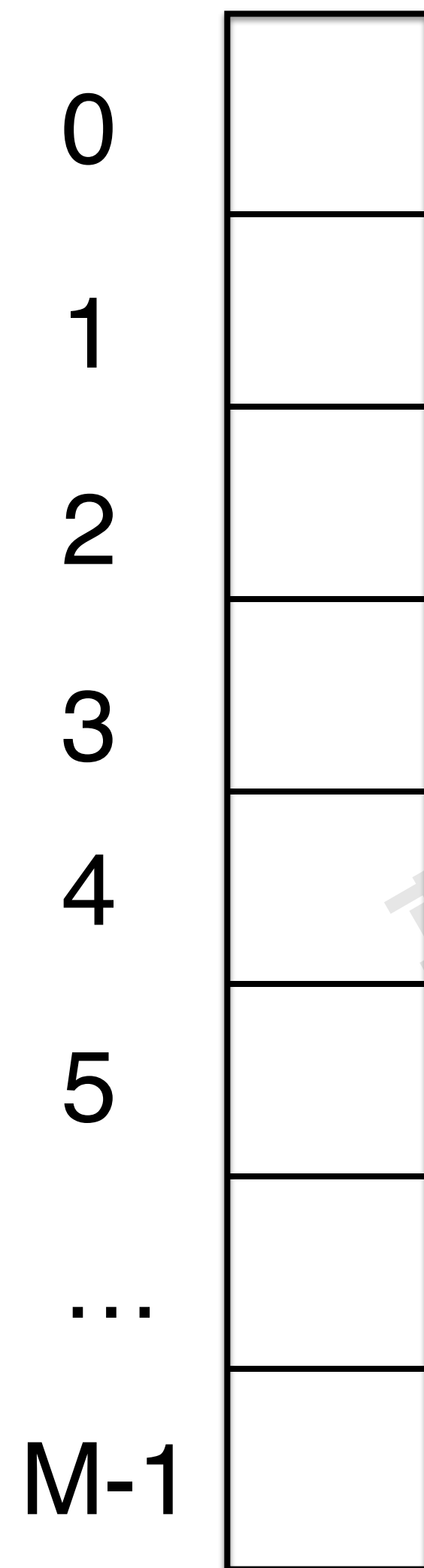
哈希冲突的处理 链地址法

Seperate Chaining

哈希冲突的处理 链地址法



哈希冲突的处理 链地址法



$$(\text{hashCode}(k1) \& 0x7fffffff) \% M$$

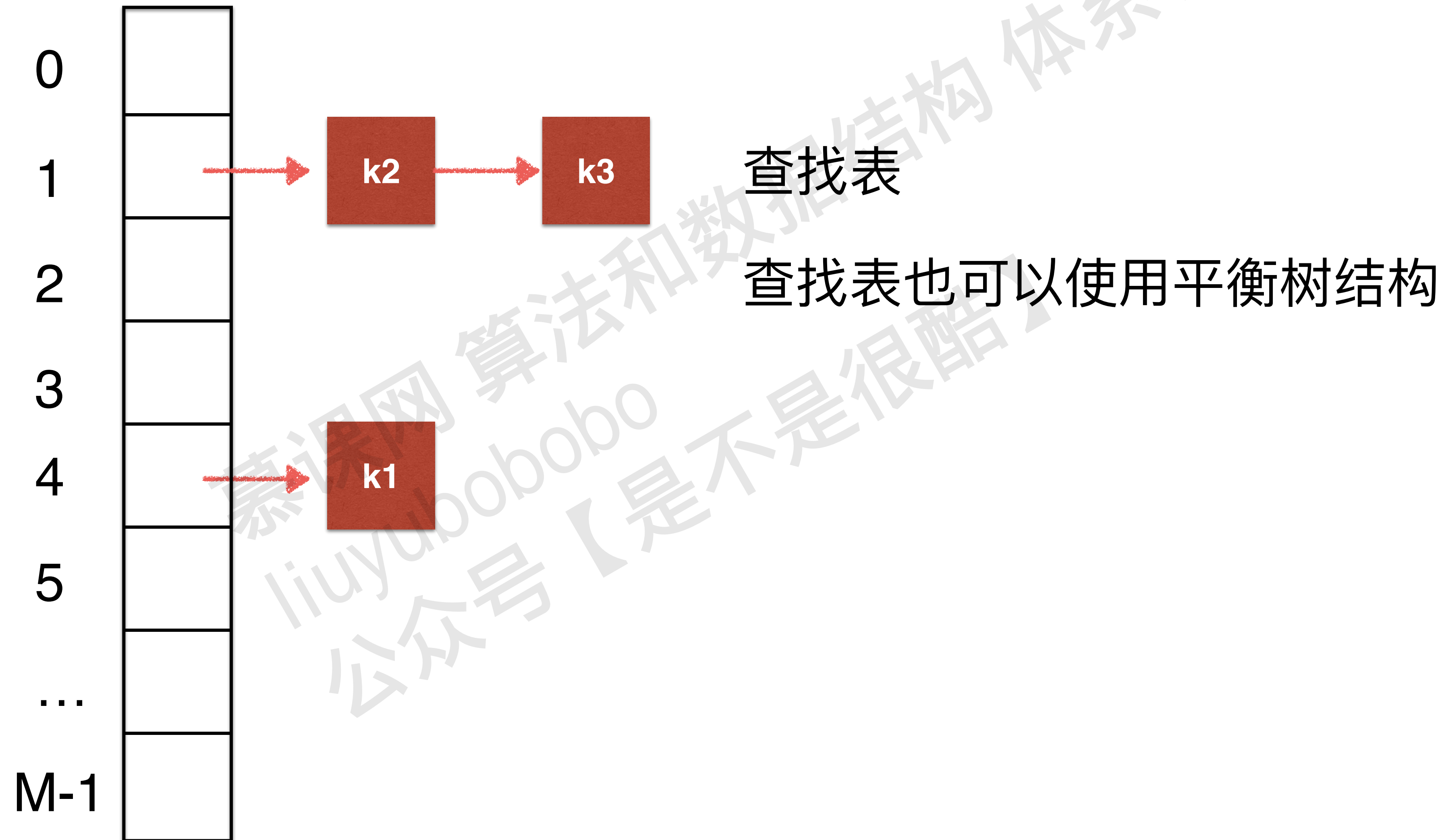
哈希冲突的处理 链地址法



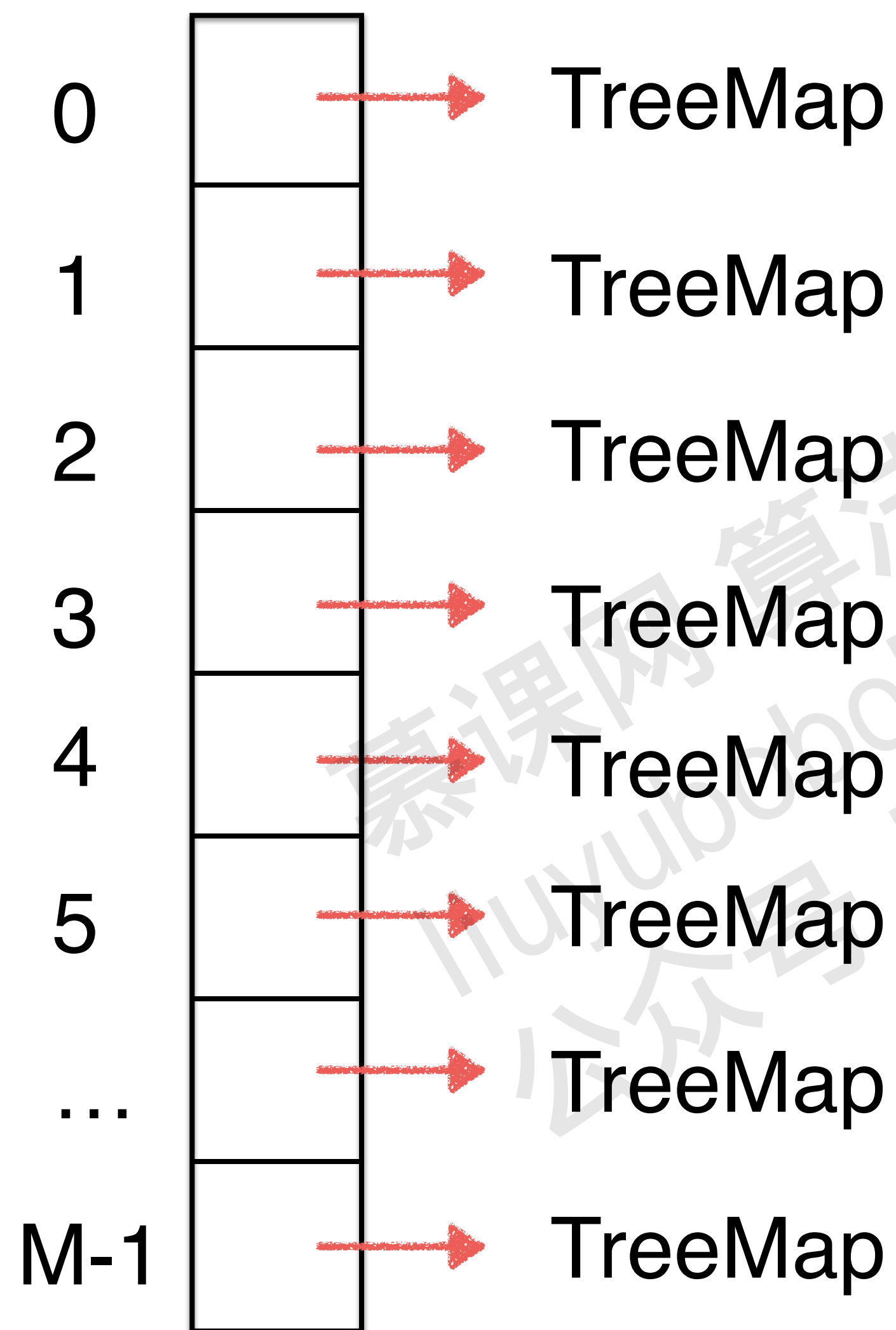
哈希冲突的处理 链地址法



哈希冲突的处理 链地址法



哈希冲突的处理 链地址法



HashMap 就是一个 TreeMap 数组

HashSet 就是一个 TreeSet 数组

Java8之前， 每一个位置对应一个链表

Java8开始， 当哈希冲突达到一定程度

每一个位置从链表转成红黑树

实现属于我们自己的哈希表

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

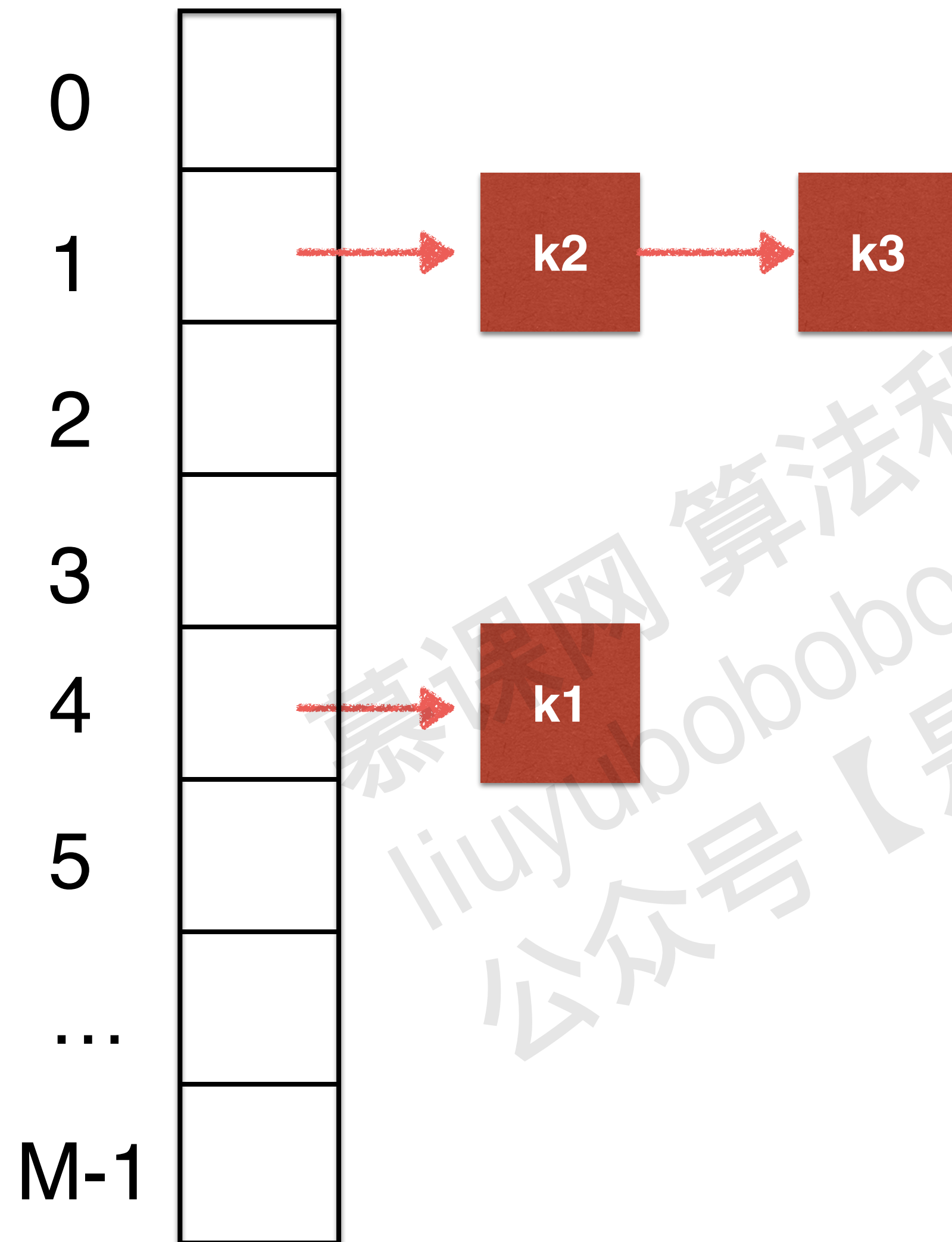
实践：实现基于链地址法的哈希表

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

哈希表的动态空间处理

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

哈希表 链地址法



总共有 M 个地址

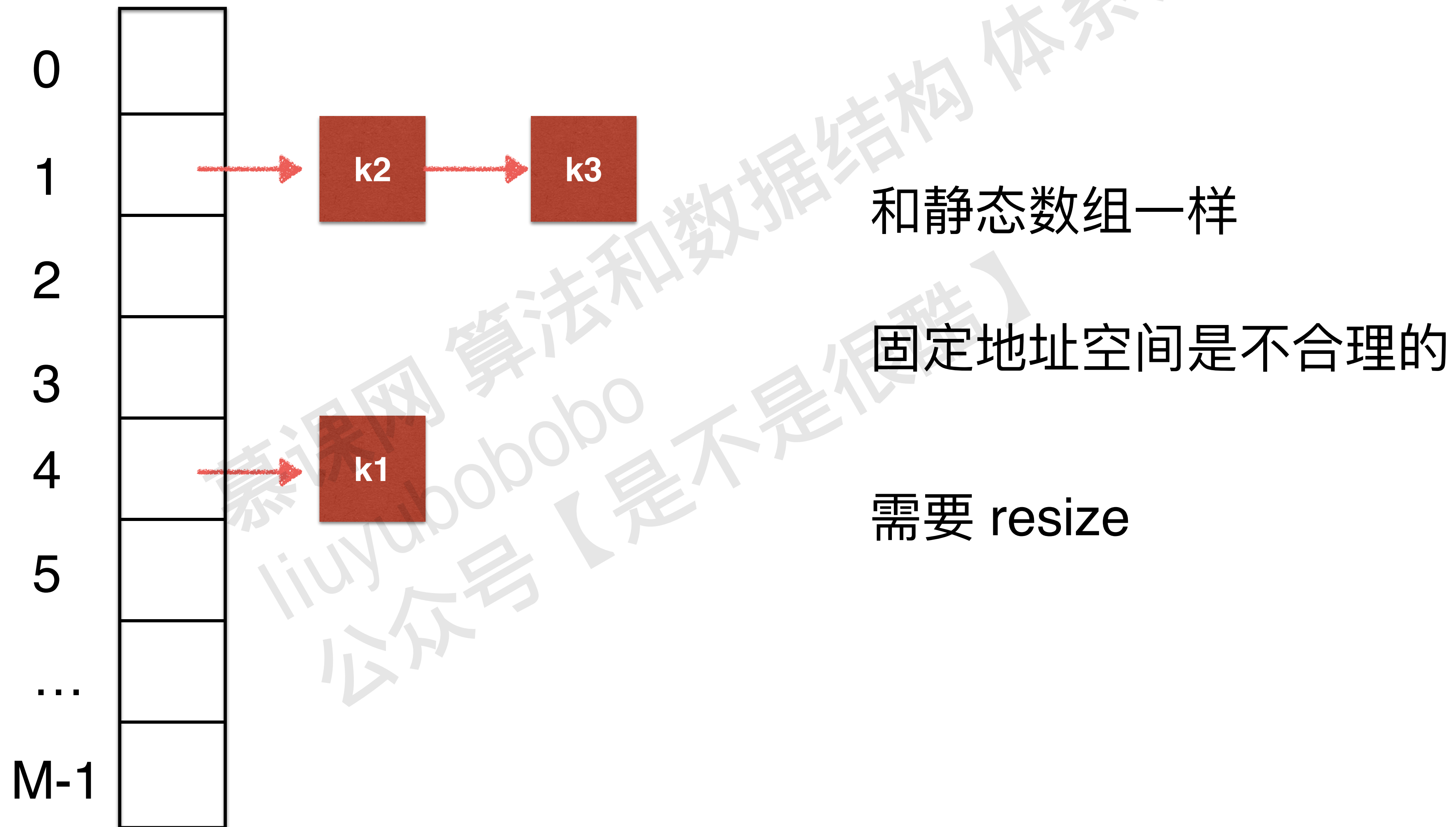
如果放入哈希表的元素为 N

如果每个地址是链表: $O(N/M)$

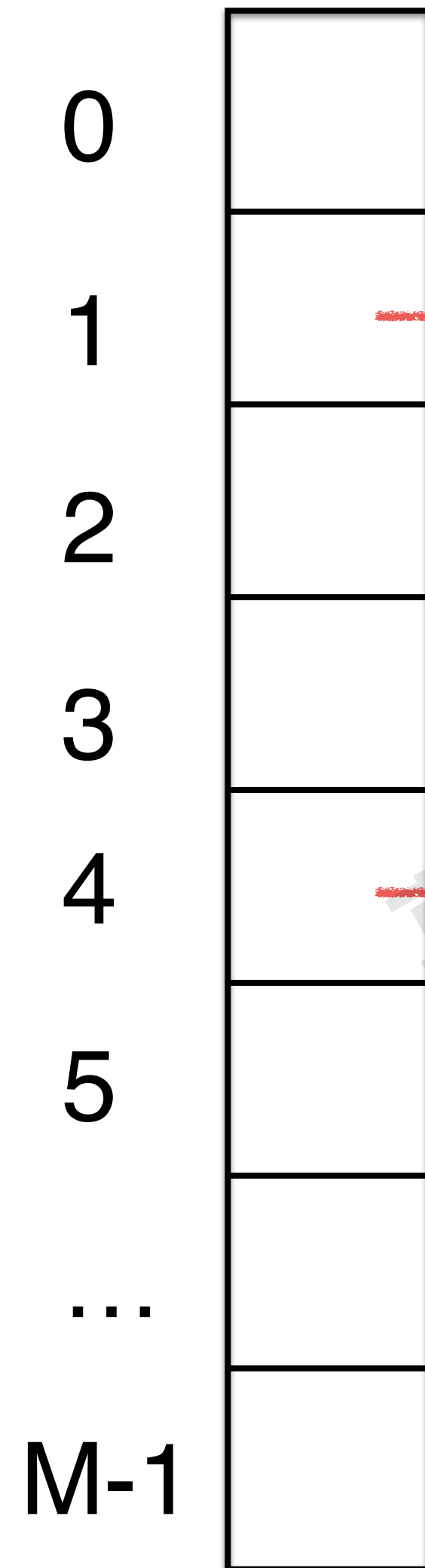
如果每个地址是平衡树: $O(\log(N/M))$

说好的 $O(1)$ 呢?

哈希表 链地址法



哈希表的动态空间处理



平均每个地址承载的元素多过一定程度，即扩容

$$N / M \geq \text{upperTol}$$

平均每个地址承载的元素少过一定程度，即缩容

$$N / M < \text{lowerTol}$$

实践：哈希表的动态空间处理

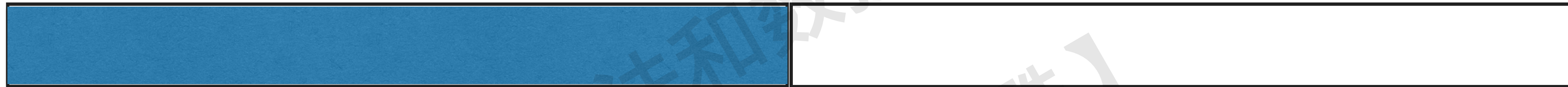
慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

哈希表的复杂度分析

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

哈希表的复杂度分析

回忆动态数组的均摊复杂度分析 平均复杂度 $O(1)$




对于哈希表来说，元素数从 N 增加到 $\text{upperTol} * N$ ；地址空间增倍

平均复杂度 $O(1)$

哈希表的复杂度分析

对于哈希表来说，元素数从 N 增加到 $\text{upperTol} * N$ ；地址空间增倍

平均复杂度 $O(1)$

每个操作在 $O(\text{lowerTol}) \sim O(\text{upperTol})$  $O(1)$

缩容同理

更复杂的动态空间处理方法

扩容 $M \rightarrow 2 \cdot M$

扩容 $2 \cdot M$ 不是素数

解决方案

lwr	upr	% err	prime
2^5	2^6	10.416667	53
2^6	2^7	1.041667	97
2^7	2^8	0.520833	193
2^8	2^9	1.302083	389
2^9	2^{10}	0.130208	769
2^{10}	2^{11}	0.455729	1543
2^{11}	2^{12}	0.227865	3079
2^{12}	2^{13}	0.113932	6151
2^{13}	2^{14}	0.008138	12289
2^{14}	2^{15}	0.069173	24593
2^{15}	2^{16}	0.010173	49157
2^{16}	2^{17}	0.013224	98317
2^{17}	2^{18}	0.002543	196613
2^{18}	2^{19}	0.006358	393241
2^{19}	2^{20}	0.000127	786433
2^{20}	2^{21}	0.000318	1572869
2^{21}	2^{22}	0.000350	3145739
2^{22}	2^{23}	0.000207	6291469
2^{23}	2^{24}	0.000040	12582917
2^{24}	2^{25}	0.000075	25165843
2^{25}	2^{26}	0.000010	50331653
2^{26}	2^{27}	0.000023	100663319
2^{27}	2^{28}	0.000009	201326611
2^{28}	2^{29}	0.000001	402653189
2^{29}	2^{30}	0.000011	805306457
2^{30}	2^{31}	0.000000	1610612741

实践：哈希表更复杂的空间处理方法

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

哈希表

哈希表：均摊复杂度为 $O(1)$

牺牲了什么？

顺序性

集合，映射

有序集合，有序映射

无序集合，无序映射

平衡树

哈希表

我们的哈希表的bug :-)

```
public class HashTable<K, V> {
```

```
    private TreeMap<K, V>[] hashtable;
```

不要要求Comparable

要求Comparable

矛盾!

Java8之前，每一个位置对应一个链表

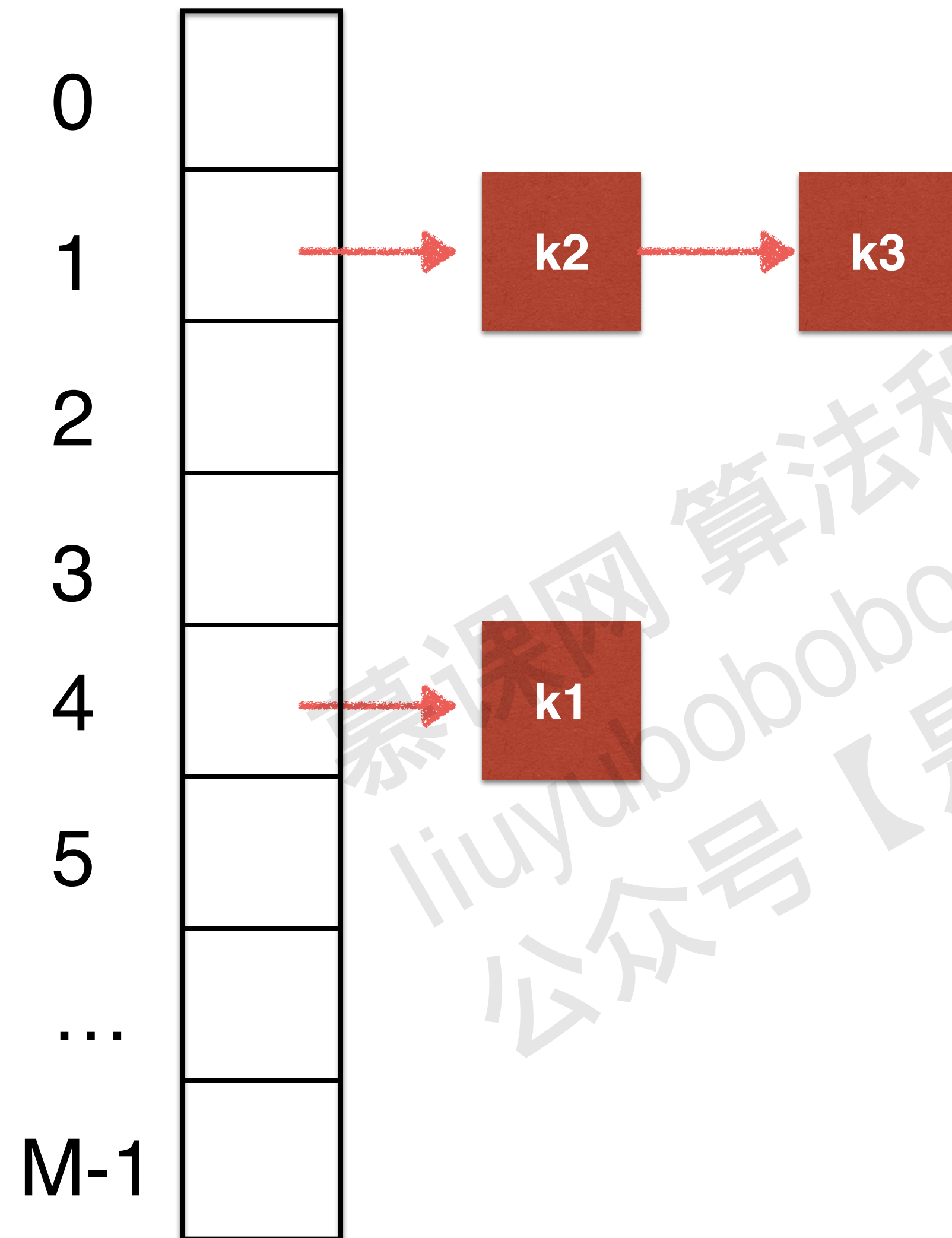
Java8开始，当哈希冲突达到一定程度

每一个位置从链表转成红黑树

更多哈希冲突的处理方法

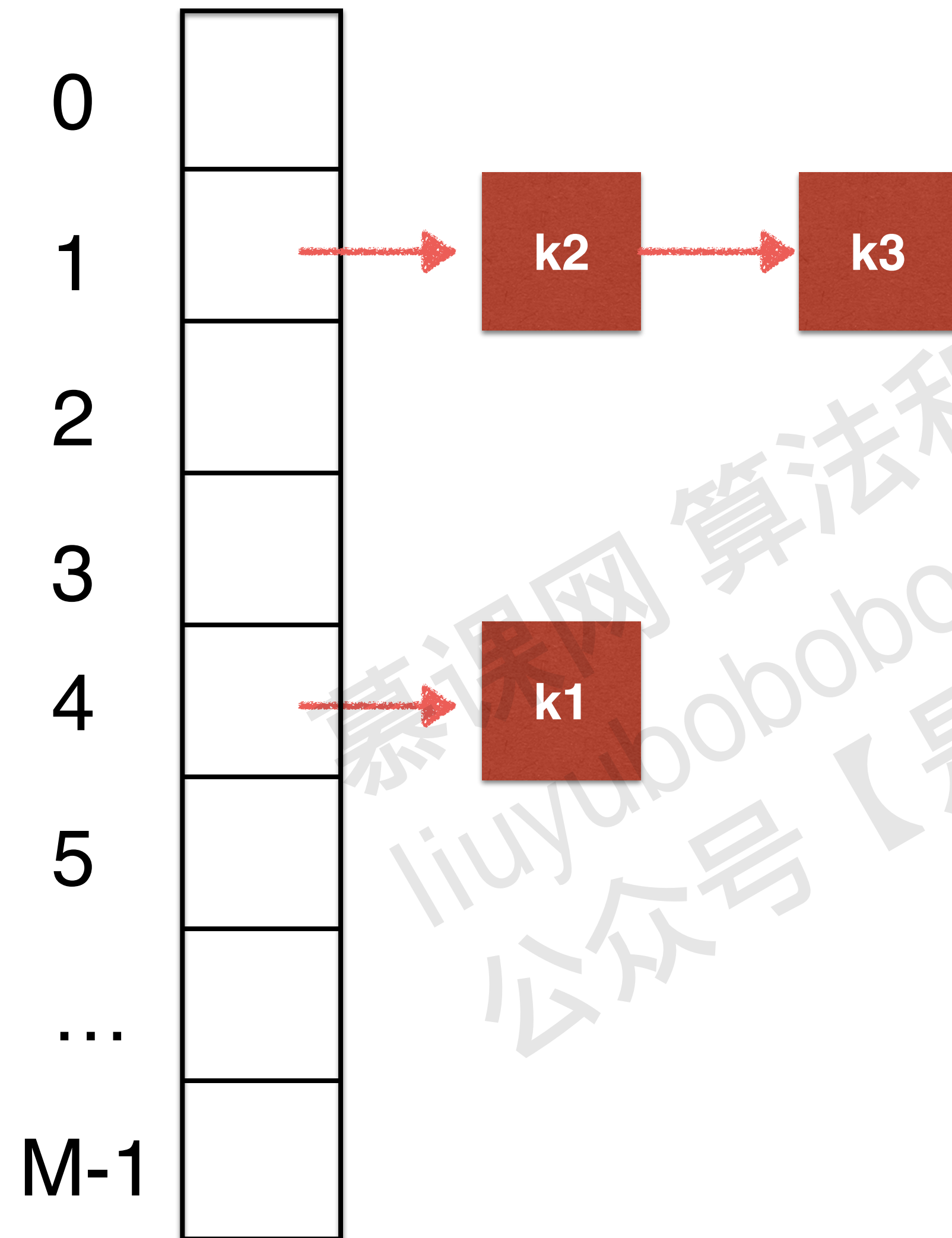
慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

开放地址法



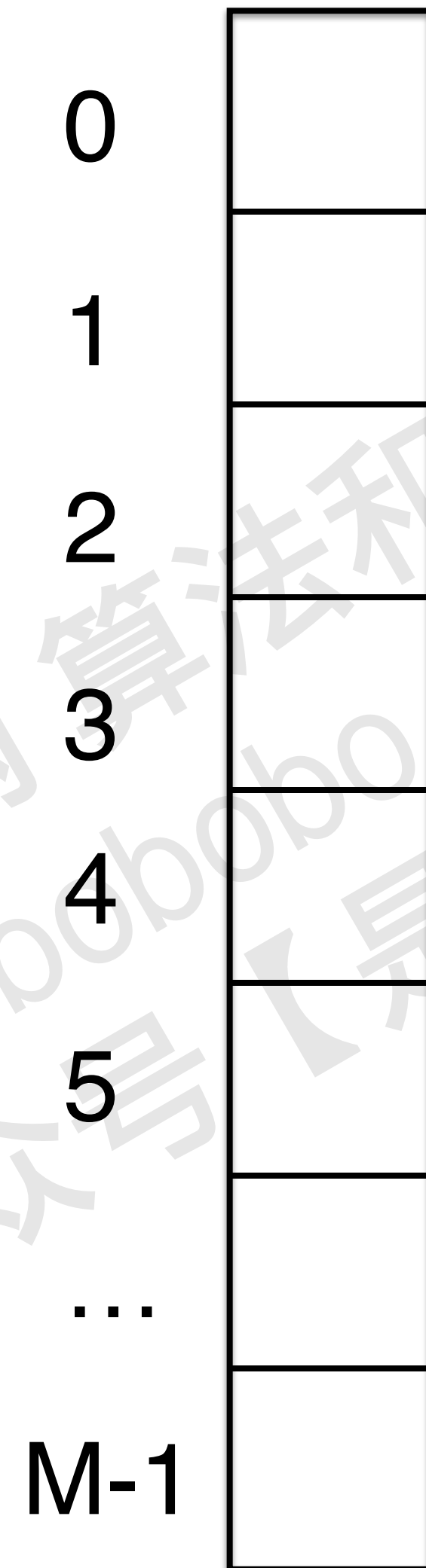
封闭地址!

开放地址法



封闭地址!

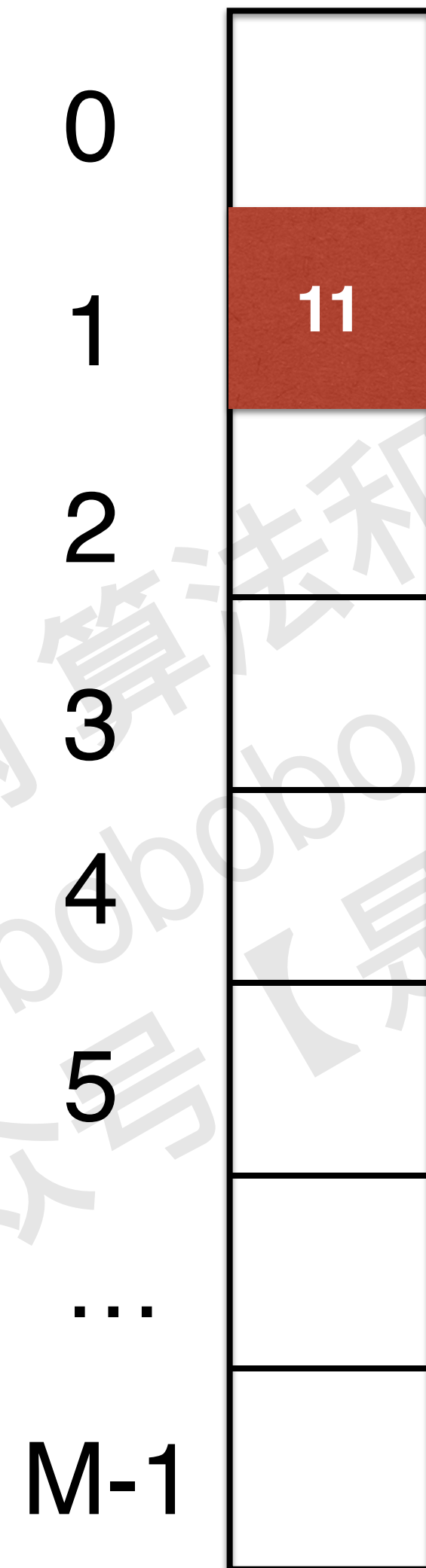
开放地址法



11

$$\text{hash}(x) = x \% 10$$

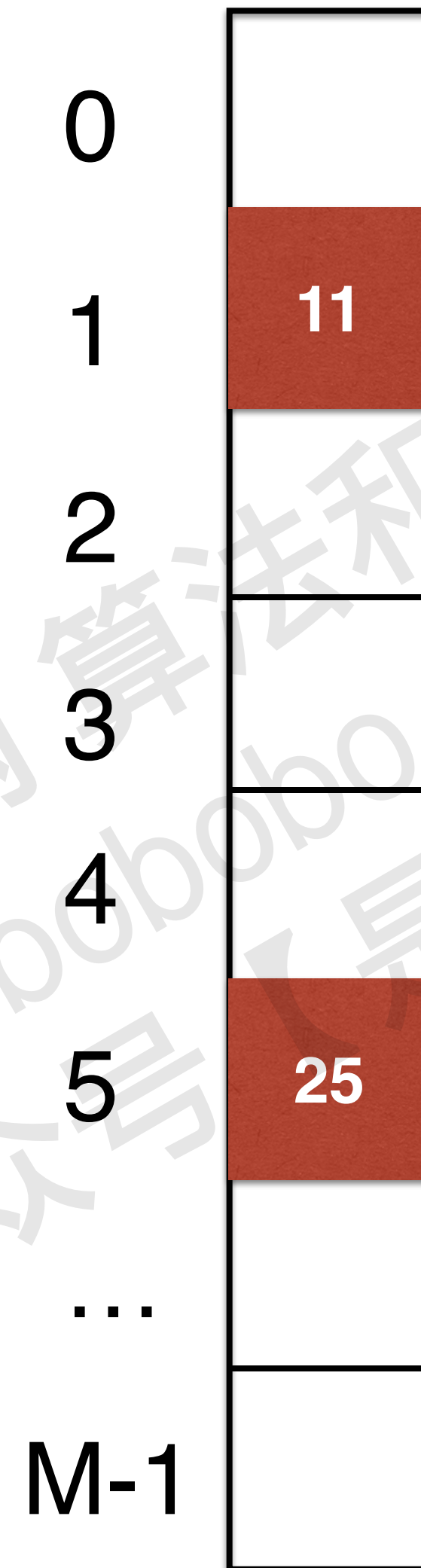
开放地址法



25

$$\text{hash}(x) = x \% 10$$

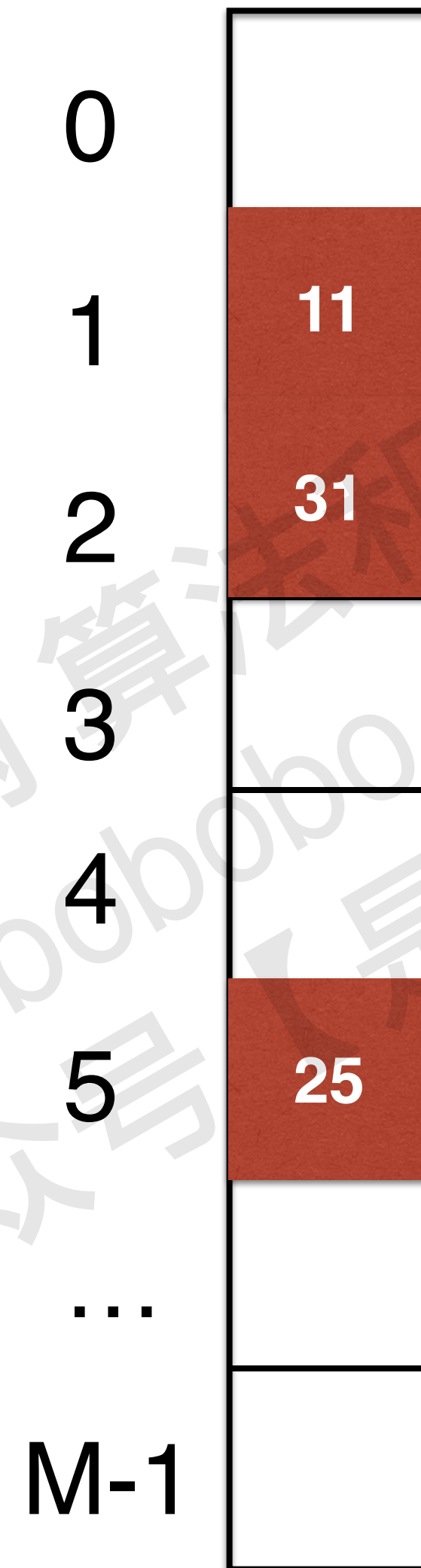
开放地址法



31

$$\text{hash}(x) = x \% 10$$

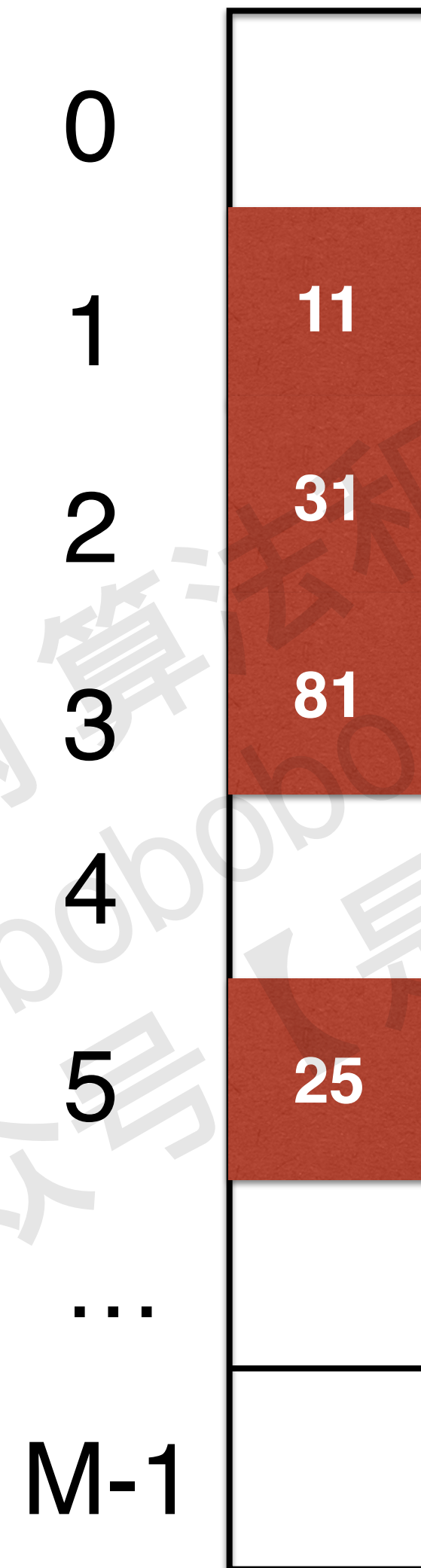
开放地址法



81

$$\text{hash}(x) = x \% 10$$

开放地址法

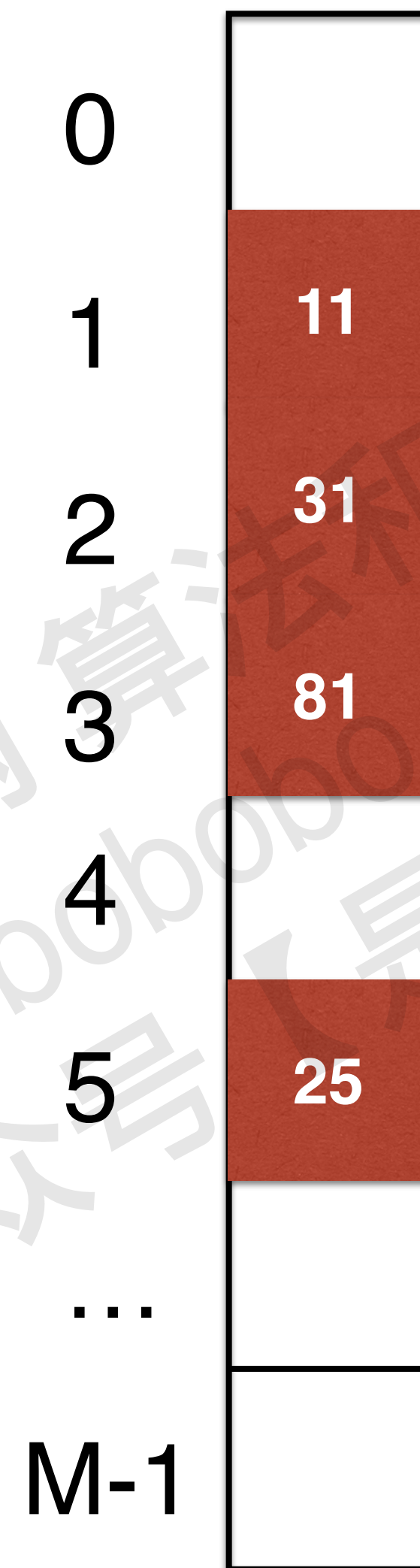


线性探测

遇到哈希冲突 + 1

$$\text{hash}(x) = x \% 10$$

开放地址法



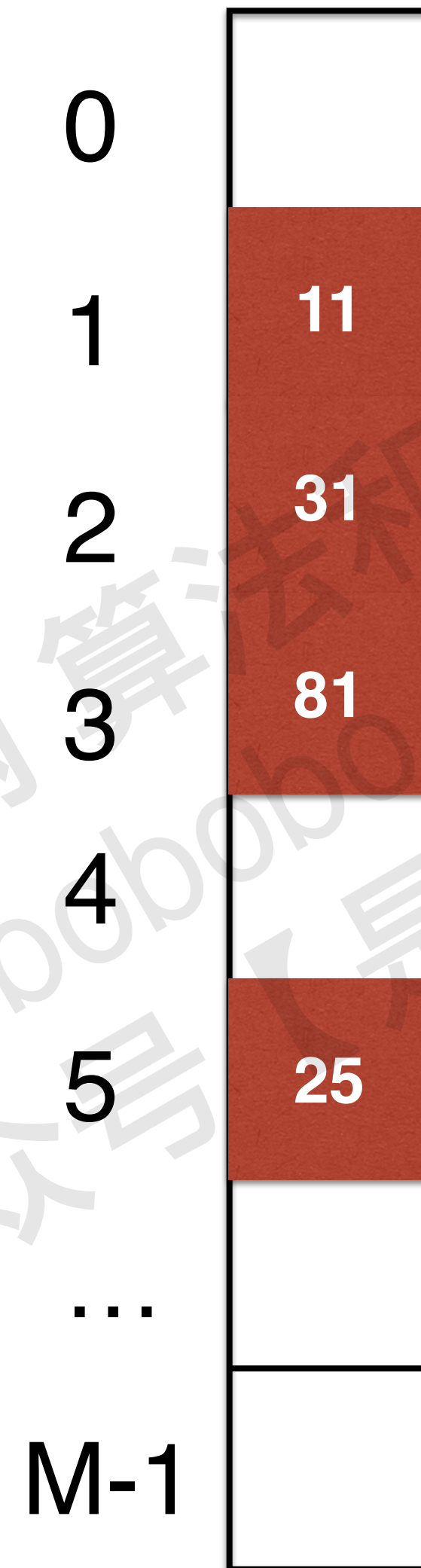
平方探测

遇到哈希冲突

+1 +4 +9 +16

$$\text{hash}(x) = x \% 10$$

开放地址法



二次哈希

遇到哈希冲突

+ hash2(key)

$$\text{hash}(x) = x \% 10$$

更多哈希冲突的处理方法

再哈希法 Rehashing

Coalesced Hashing

综合了 Seperate Chaining 和 Open Addressing

哈希表

慕课网 算法和数据结构 体系课程
liuyubobobo
公众号【是不是很有趣】

其他

欢迎大家关注我的个人公众号：是不是很酷



算法与数据结构体系课程

liuyubobobo