


本节内容

散列表的 基本概念

知识总览



散列表的基本概念

散列表、散列函数

冲突、同义词

关于散列表，有待解决的问题？

散列表、散列函数

散列表（哈希表，Hash Table）：是一种数据结构。特点是：可以根据数据元素的关键字计算出它在散列表中的存储地址

散列函数（哈希函数）： $\text{Addr}=\text{H}(\text{key})$ 建立了“关键字”→“存储地址”的映射关系。

例：某散列表的长度为13，散列函数 $\text{H}(\text{key})=\text{key}\%13$ 。依次将数据元素 19、14、23 插入散列表：

$$19\%13=6$$

$$14\%13=1$$

$$23\%13=10$$

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

理想情况下，在散列表中查找一个元素的时间复杂度为 $O(1)$ 。比如：

查找元素 19：根据散列函数计算出元素的存储地址= $19\%13=6$ ，检查位置#6。查找成功

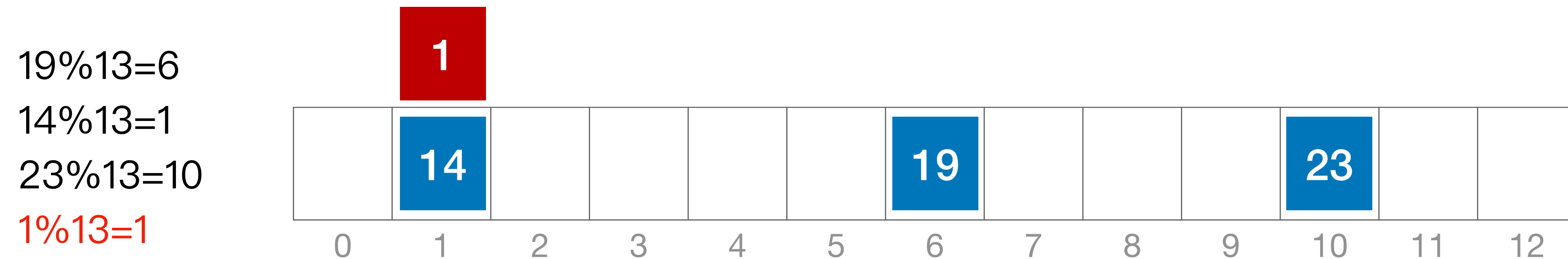
查找元素 16：根据散列函数计算出元素的存储地址= $16\%13=3$ ，检查位置#3。查找失败

冲突、同义词

冲突（碰撞）：在散列表中插入一个数据元素时，需要根据关键字的值确定其存储地址，若该地址已经存储了其他元素，则称这种情况为“冲突（碰撞）”

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

例：某散列表的长度为13，散列函数 $H(\text{key}) = \text{key} \% 13$ 。依次将数据元素 19、14、23 插入散列表：

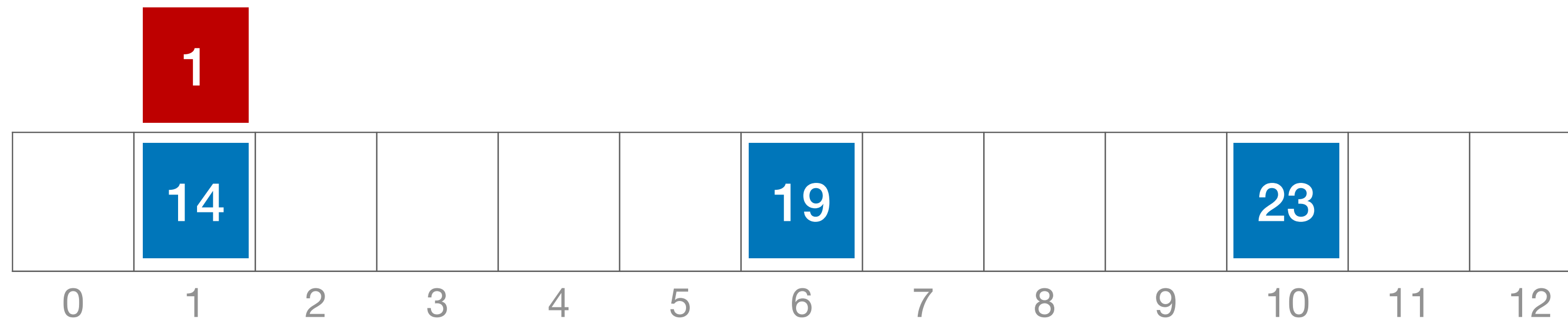


继续插入元素 1：存储地址=1%13=1，该地址已经存储了其他元素——发生“冲突”

对于散列函数 $H(\text{key}) = \text{key} \% 13$ 来说，1 和 14 是“同义词”



如何减少“冲突”？



对于散列函数 $H(\text{key}) = \text{key} \% 13$ 来说，1 和 14 是“同义词”

如何减少“冲突”？

构造更适合的散列函数，让各个关键字尽可能地映射到不同的存储位置，从而减少“冲突”



Eg: 把散列函数改为 $H(\text{key}) = \text{key} \% 12$ ，则不发生冲突

待解决的问题：如何构造散列函数？

$19 \% 12 = 7$
 $14 \% 12 = 2$
 $23 \% 12 = 11$
 $1 \% 12 = 1$

若“冲突”无可避免，如何处理冲突？



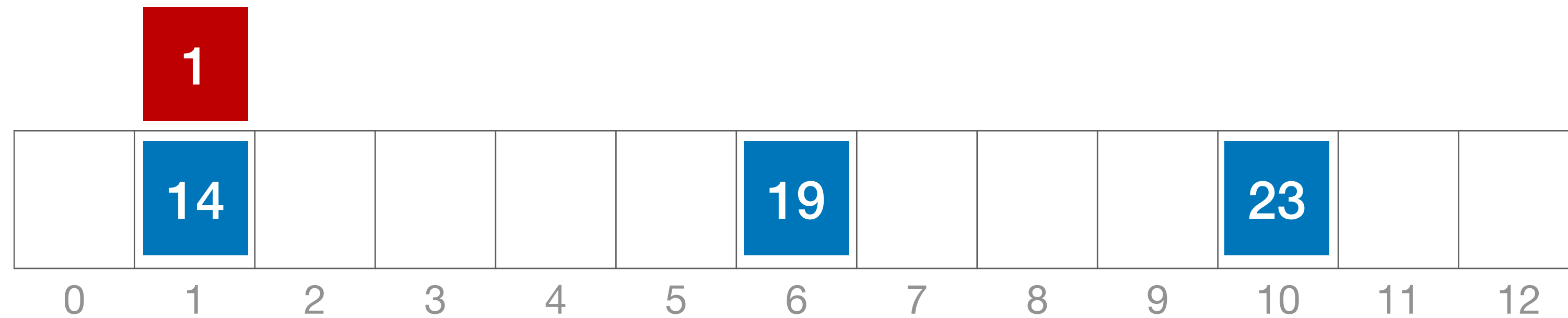
例：某散列表的长度为13，散列函数 $H(\text{key}) = \text{key} \% 13$ 。依次将数据元素 19、14、23、1 插入散列表：

$$19 \% 13 = 6$$

$$14 \% 13 = 1$$

$$23 \% 13 = 10$$

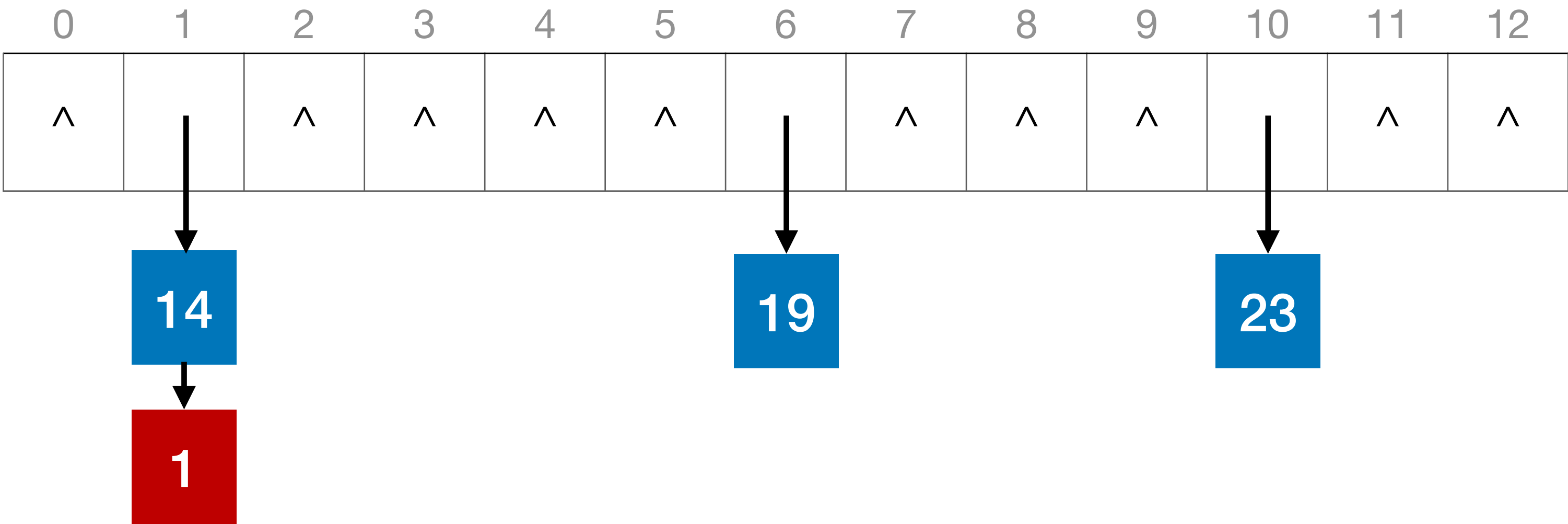
$$1 \% 13 = 1$$



如何处理冲突？——拉链法



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

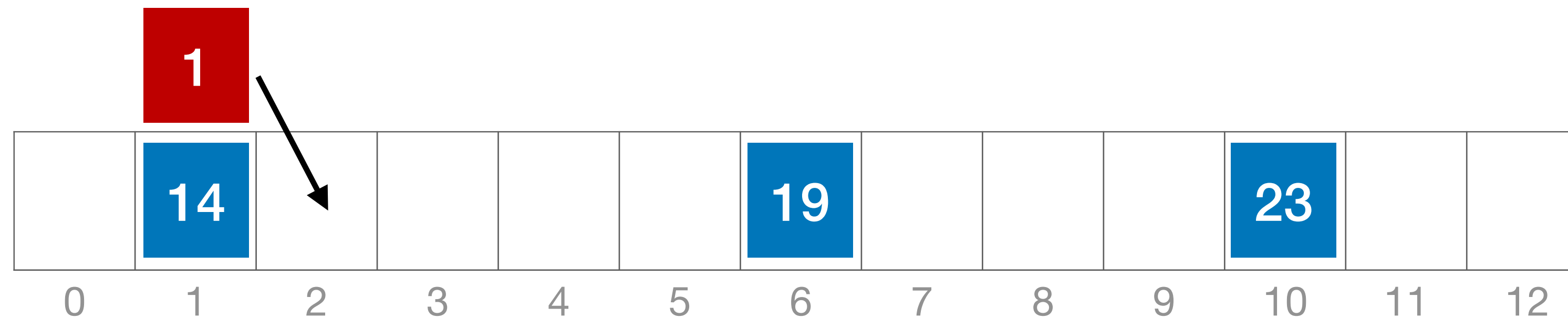


拉链法（又称链接法、链地址法）：把所有“同义词”存储在一个链表中

如何处理冲突？——开放定址法



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



开放定址法：如果发生“冲突”，就给新元素找另一个空闲位置。



待解决的问题：用什么规则确定“另一个空闲位置”？

知识回顾与重要考点

散列表的基本概念

