

# 算法与数据结构体系课程

liuyubobobo

# 数据结构基础

liuyubobobo

# 数据结构基础

数据结构研究的是数据如何在计算机中进行组织和存储，使得我们可以高效的获取数据或者修改数据。

# 数据结构基础

## 线性结构

数组； 栈；  
队列； 链表；  
哈希表 ....

## 树结构

二叉树； 二分搜索树；  
AVL； 红黑树； Treap； Splay；  
堆； Trie； 线段树； K-D树；  
并查集； 哈夫曼树； ...

## 图结构

邻接矩阵；  
邻接表；

我们需要根据应用的不同， 灵活选择最合适的数据结构

# 数据结构无处不在

数据库



```
SELECT * FROM 慕课网  
WHERE title = “数据结构”
```

树结构：

AVL；红黑树；B 类树；

哈希表

# 数据结构无处不在

操作系统



优先队列

内存管理：内存堆栈

文件管理

快速在多任务间切换

# 数据结构无处不在

文件压缩



压缩算法

哈夫曼树



# 数据结构无处不在

大量的算法，以数据结构为基石



图论算法；

DFS：使用栈

BFS：使用队列

寻路算法



# 数据结构基础

数据结构研究的是数据如何在计算机中进行组织和存储，使得我们可以高效的获取数据或者修改数据。

在内存世界的增删改查

不要小瞧数组

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

# 数组基础

- 把数据码成一排进行存放

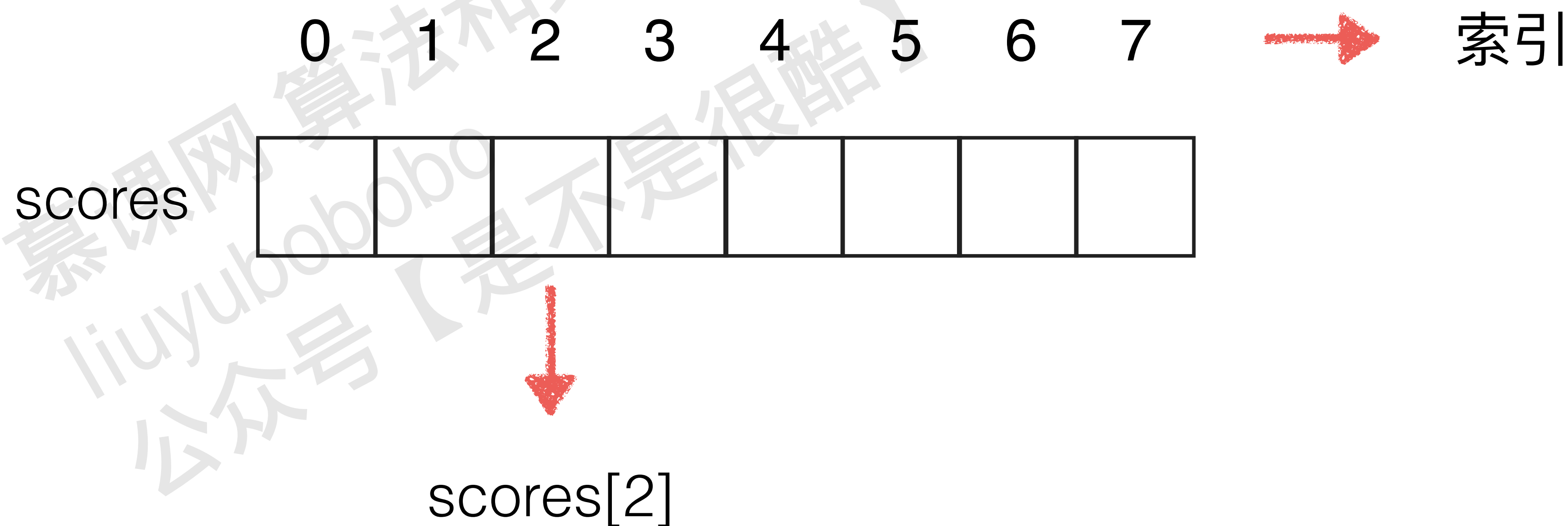
arr





# 数组基础

- 把数据码成一排进行存放

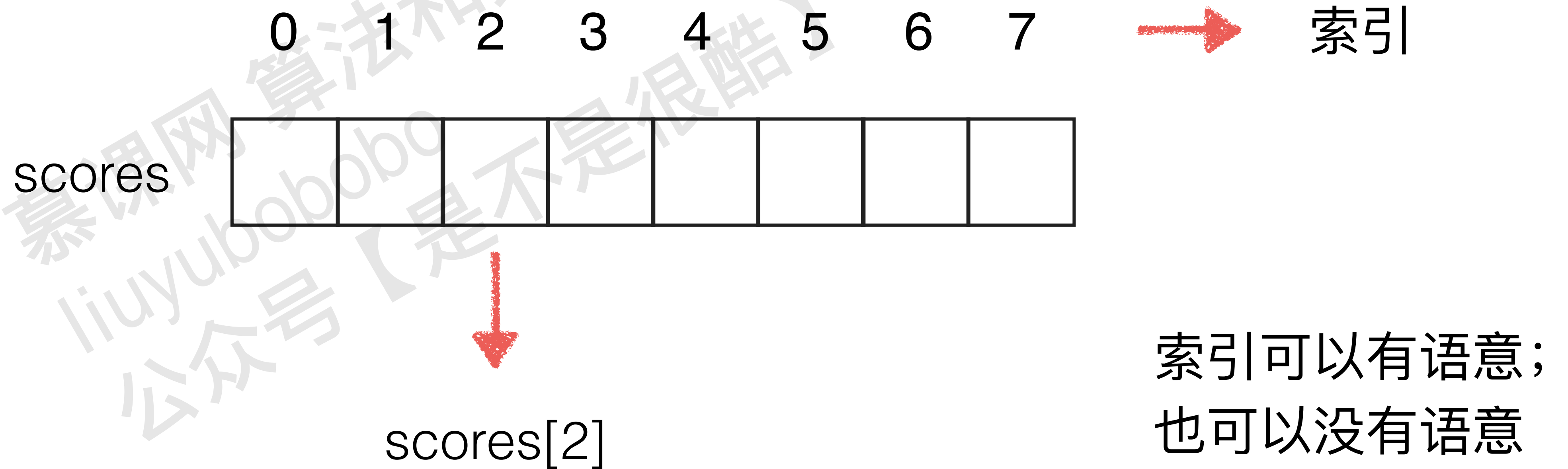


二次封装属于我们自己的数组

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

# 数组基础

- 把数据码成一排进行存放





# 数组基础

- 数组最大的优点：快速查询。scores[2]
- 数组最好应用于“索引有语意”的情况。
- 但并非所有有语意的索引都适用于数组

身份证号：110103198512166666

# 数组基础

- 但并非所有有语意的索引都适用于数组

身份证号：110103198512166666

- 数组也可以处理“索引没有语意”的情况。
- 我们在这一章，主要处理“索引没有语意”的情况数组的使用。

# 数组基础

- 我们在这一章，主要处理“索引没有语意”的情况数组的使用。

	0	1	2	3	4	5	6	7
scores	100	99	66					

- 索引没有语意，如何表示没有元素？
- 如何添加元素？ 如何删除元素？
- .....

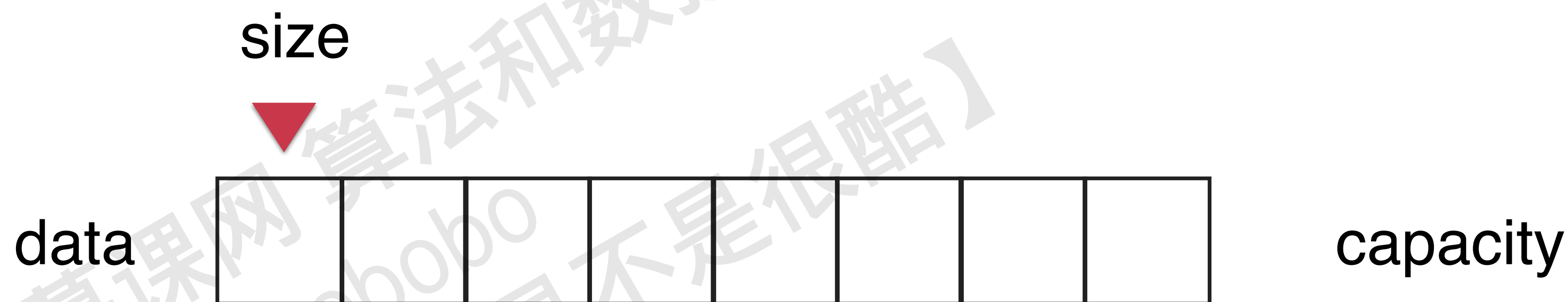


# 制作属于我们自己的数组类

- 索引没有语意，如何表示没有元素？
- 如何添加元素？如何删除元素？
- .....
- 基于java的数组，二次封装属于我们自己的数组类

# 制作属于我们自己的数组类

class Array



增

删

改

查

实践：二测封装属于我们自己的数组

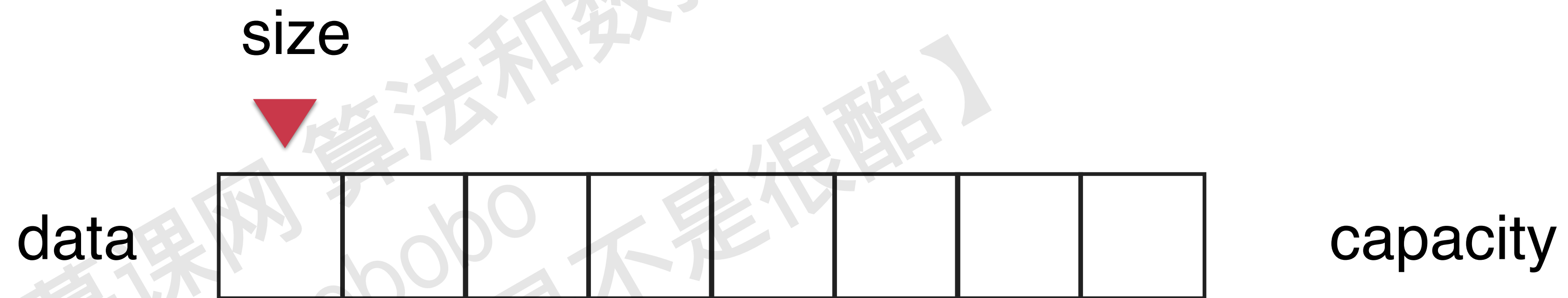


# 向数组添加元素

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

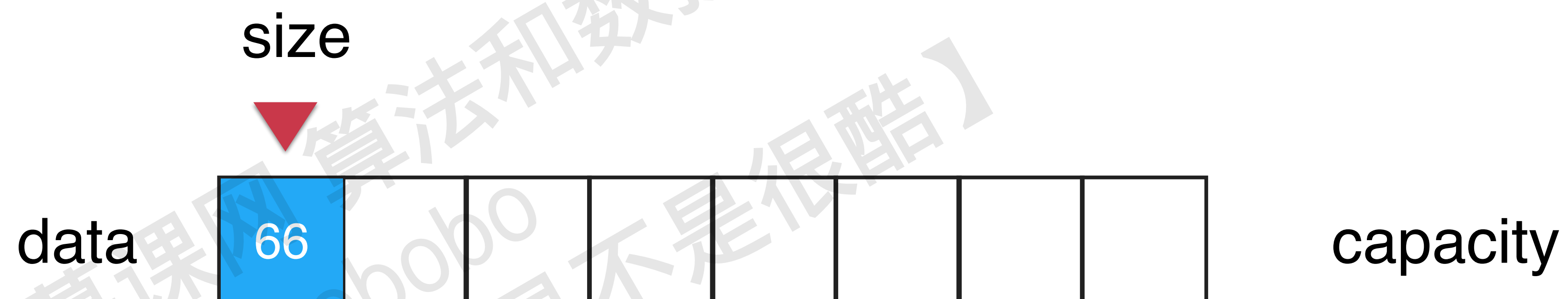
# 向数组中添加元素

向数组末添加元素



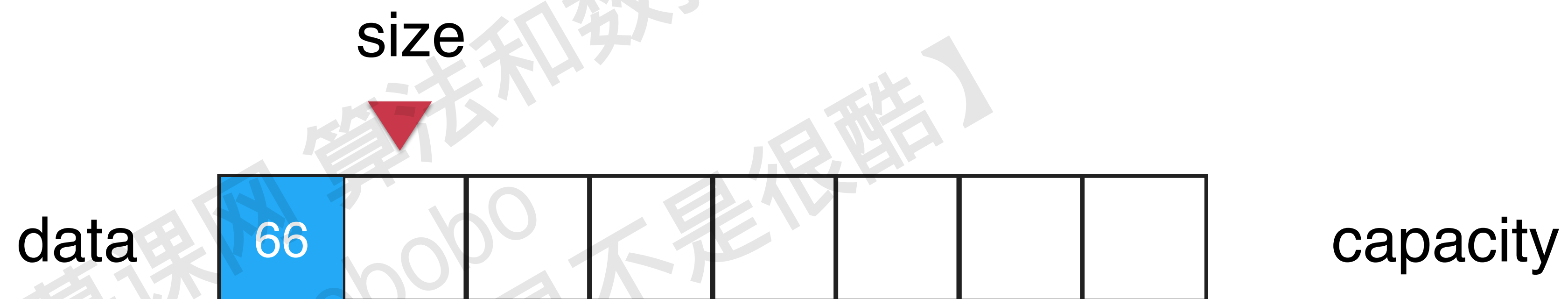
# 向数组中添加元素

向数组末添加元素



# 向数组中添加元素

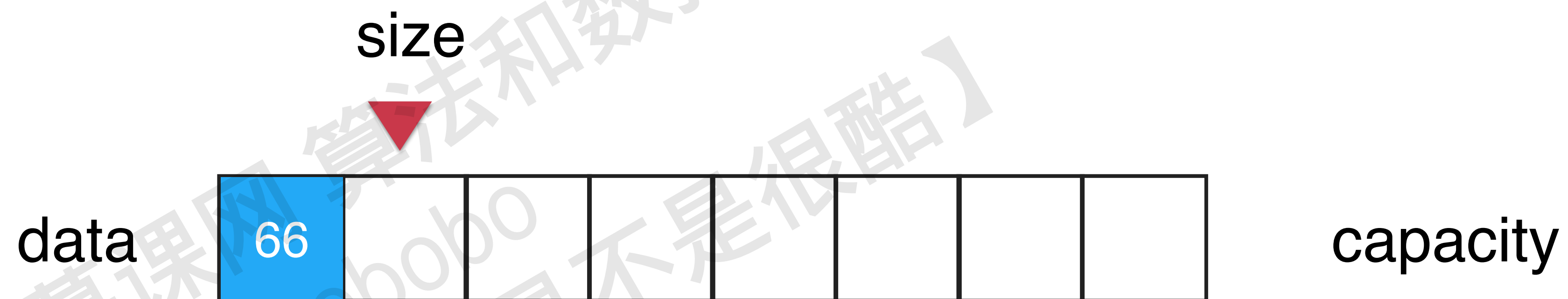
向数组末添加元素





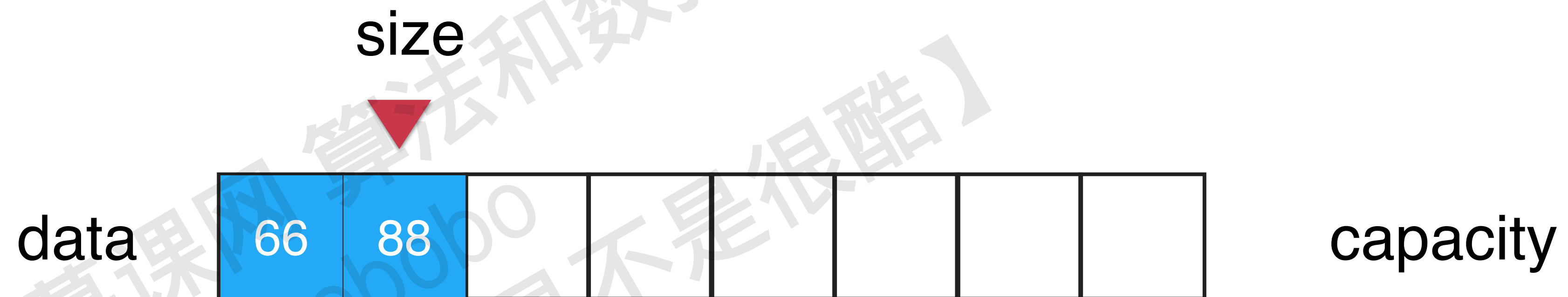
# 向数组中添加元素

向数组末添加元素



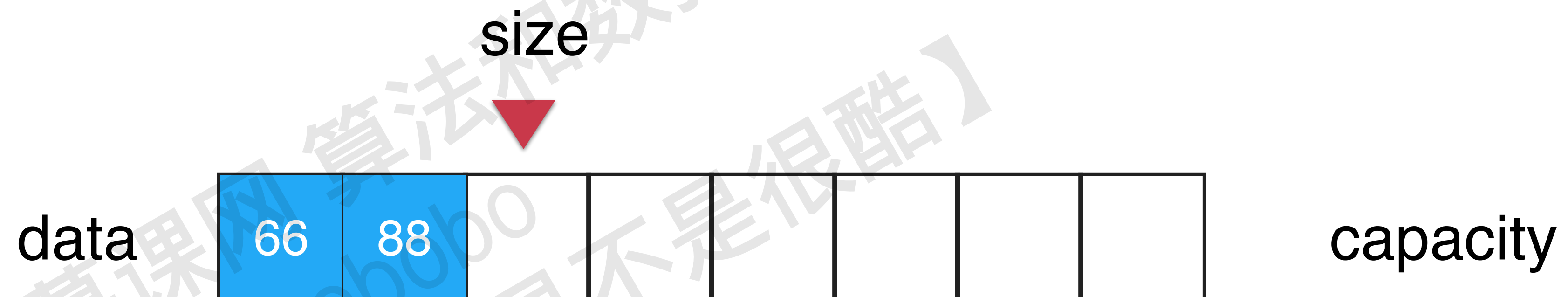
# 向数组中添加元素

向数组末添加元素



# 向数组中添加元素

向数组末添加元素

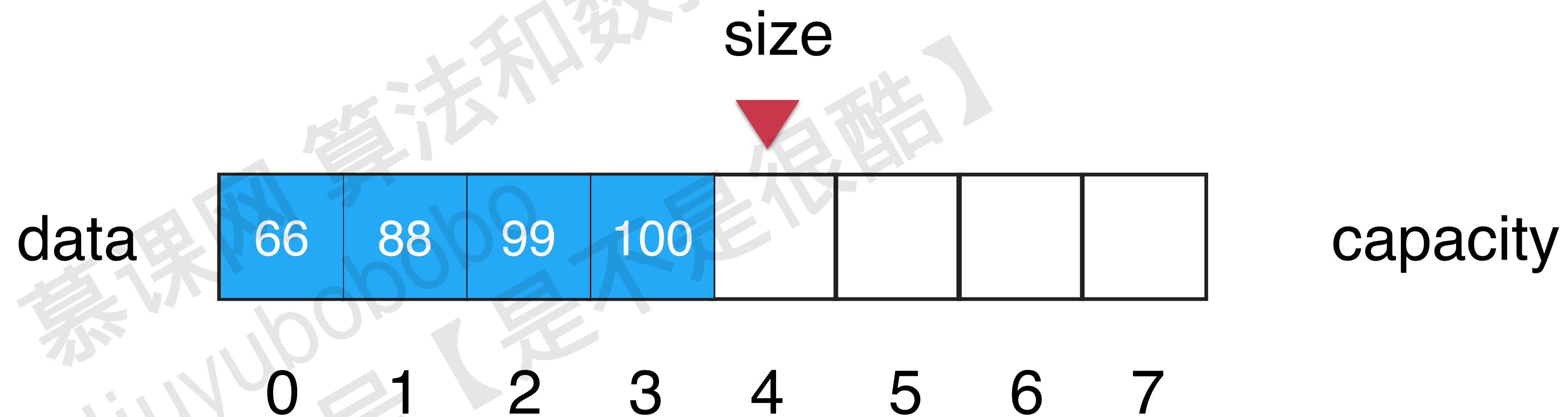


# 实践：向数组末尾添加元素

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

# 向数组中添加元素

向指定位置添加元素



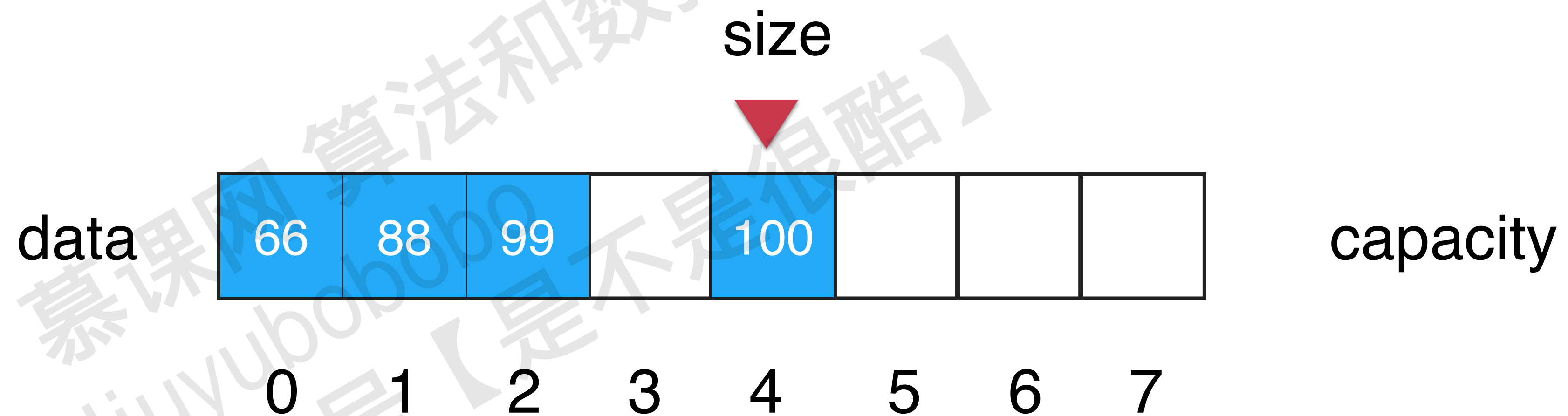
把77插入到索引为1的位置

77



# 向数组中添加元素

向指定位置添加元素

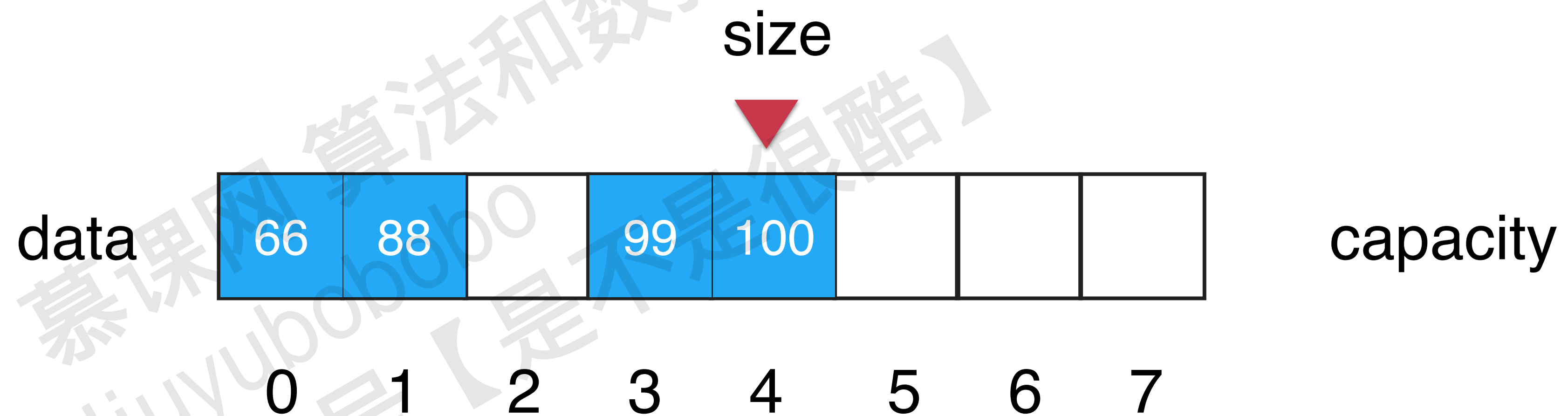


把77插入到索引为1的位置

77

# 向数组中添加元素

向指定位置添加元素

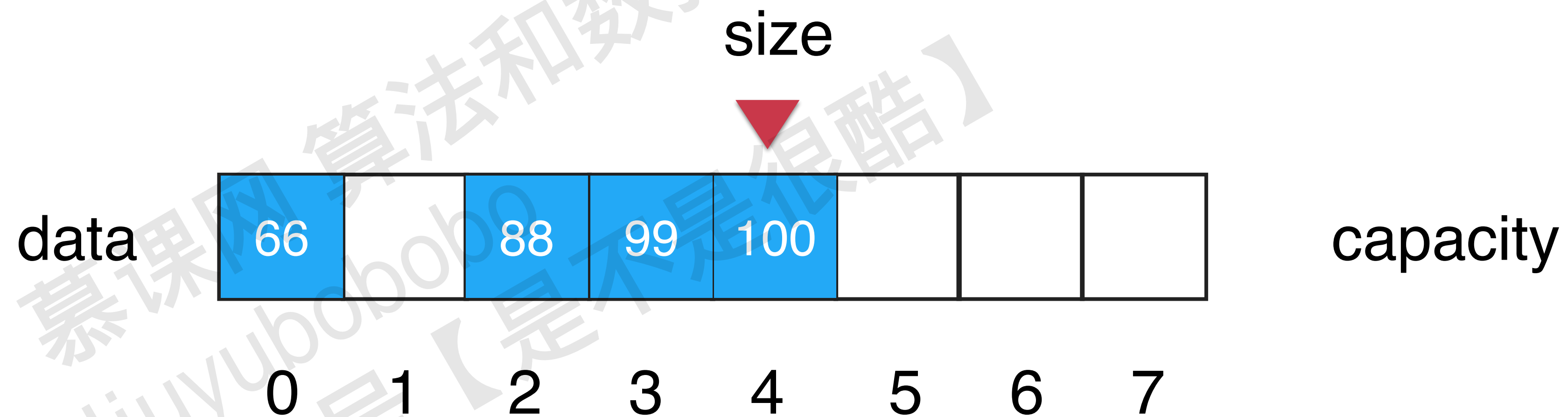


把77插入到索引为1的位置

77

# 向数组中添加元素

向指定位置添加元素

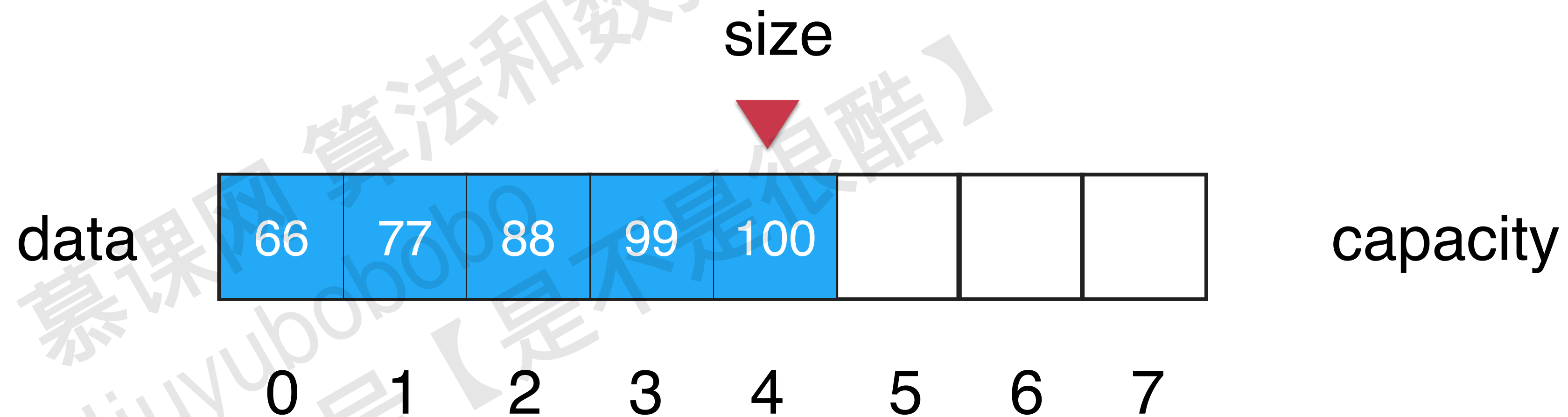


把77插入到索引为1的位置

77

# 向数组中添加元素

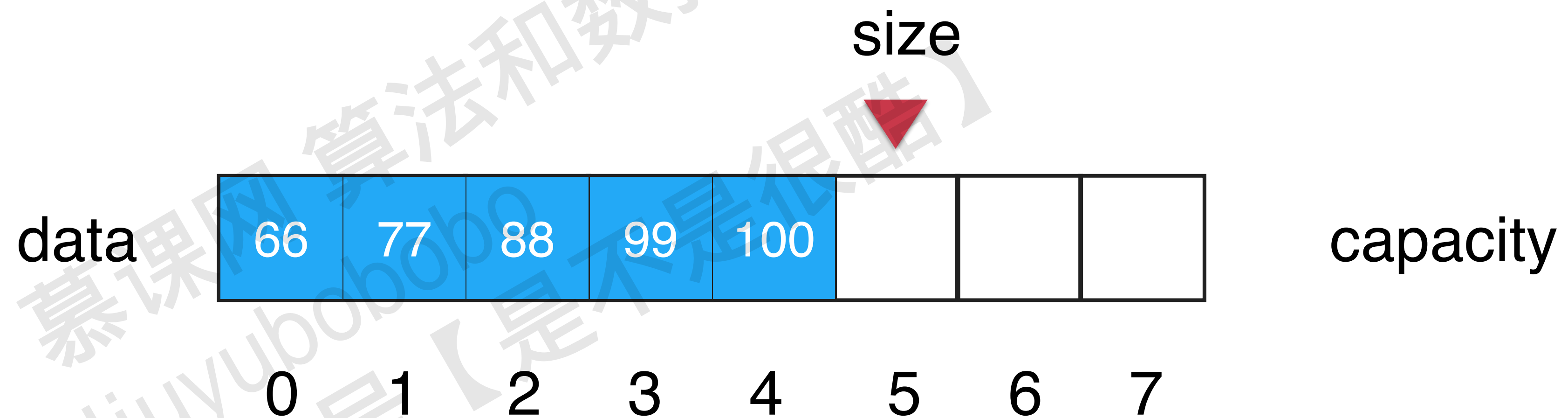
向指定位置添加元素



把77插入到索引为1的位置

# 向数组中添加元素

向指定位置添加元素



把77插入到索引为1的位置



实践：向数组任意位置添加元素

# 在数组中查询元素和修改元素

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：在数组中查询元素和修改元素

# 数组中的包含，搜索和删除元素

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

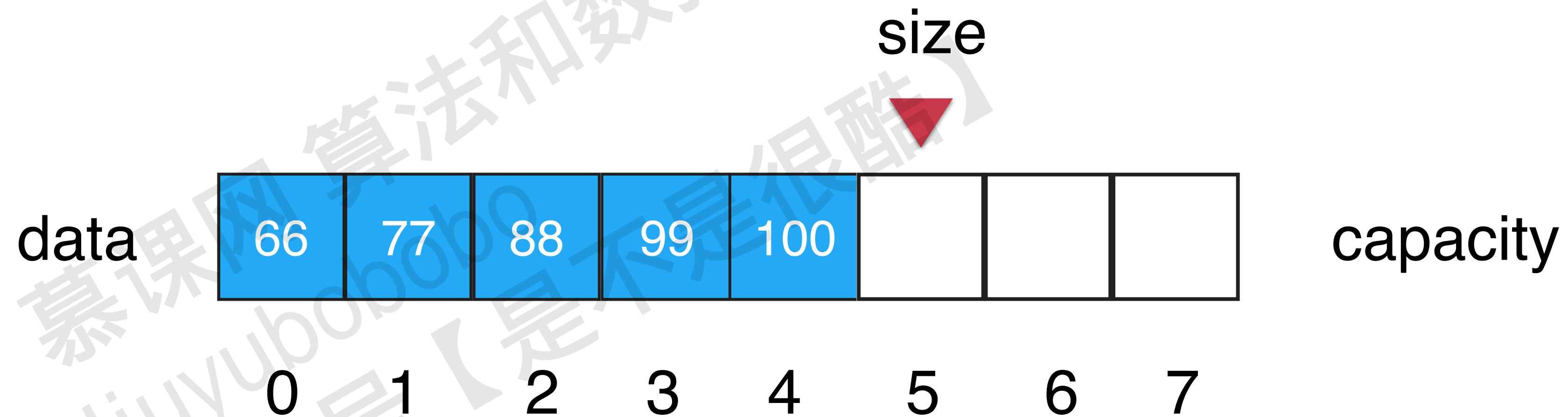
# 实践：数组中的包含和搜索

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



# 从数组中删除元素

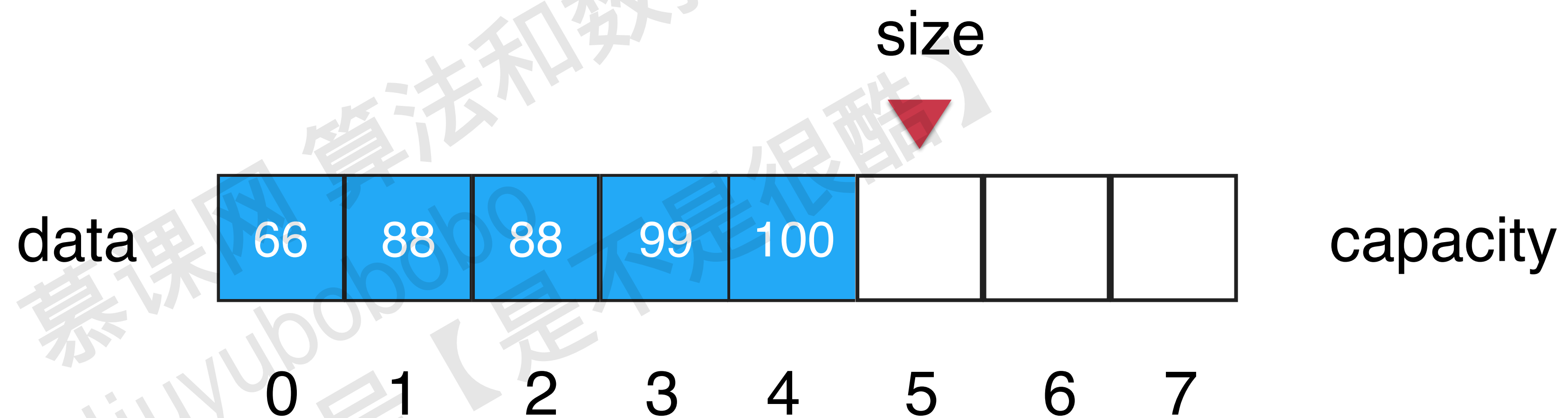
删除指定位置元素



删除索引为1的元素

# 从数组中删除元素

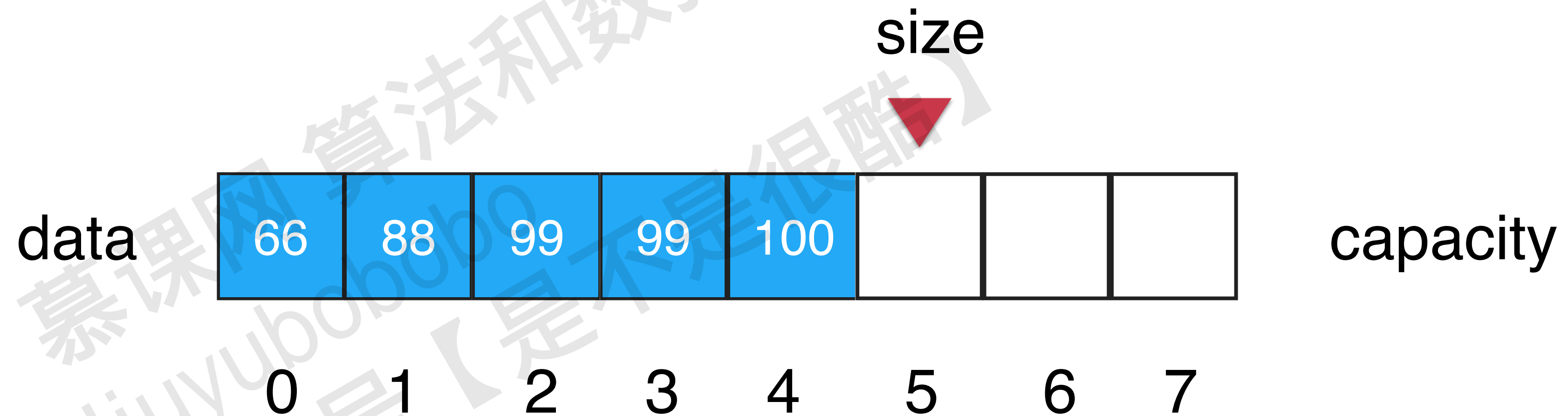
删除指定位置元素



删除索引为1的元素

# 从数组中删除元素

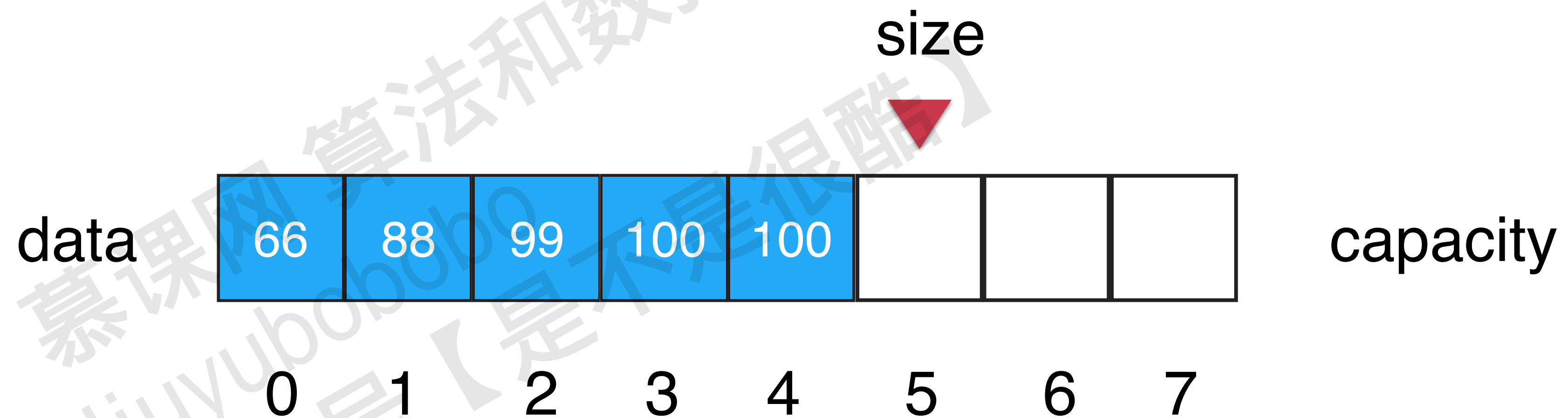
删除指定位置元素



删除索引为1的元素

# 从数组中删除元素

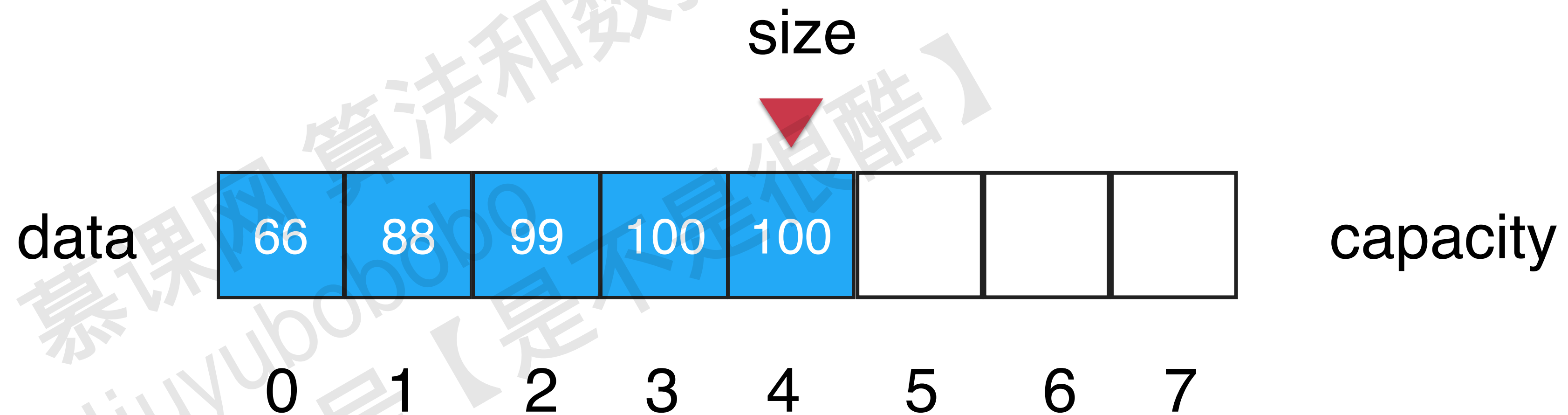
删除指定位置元素



删除索引为1的元素

# 从数组中删除元素

删除指定位置元素



删除索引为1的元素

# 实践：从数组中删除元素

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



# 使用泛型

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

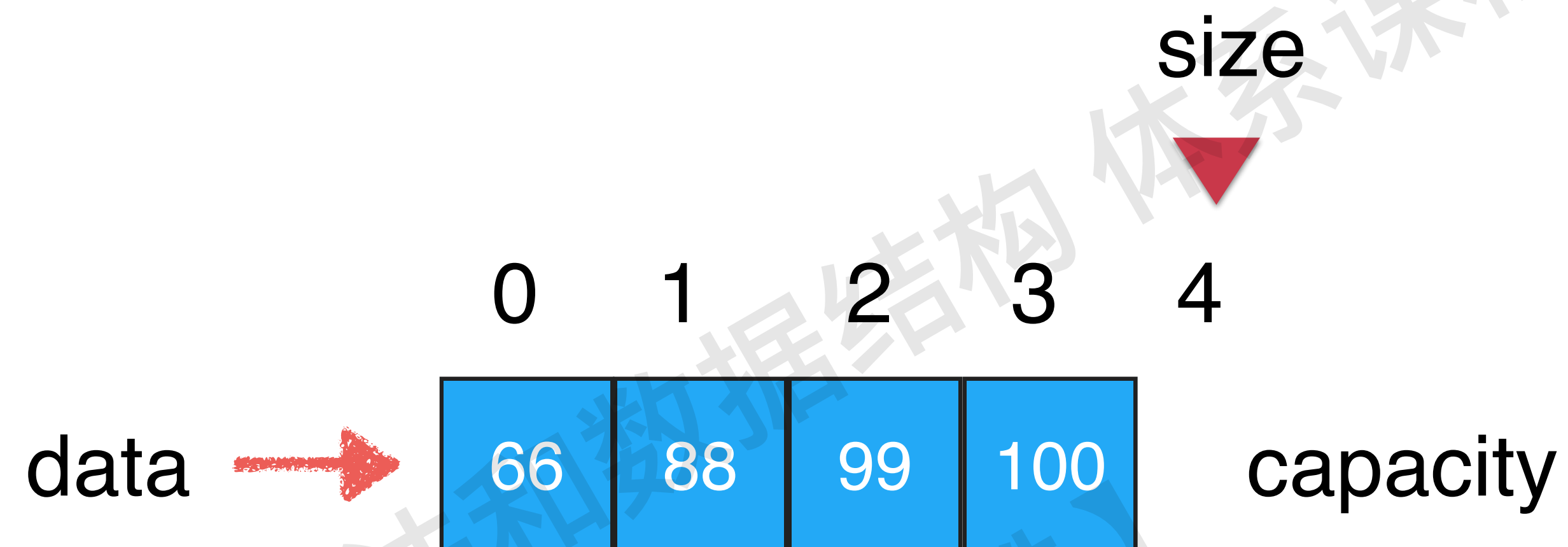
# 实践：使用泛型

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

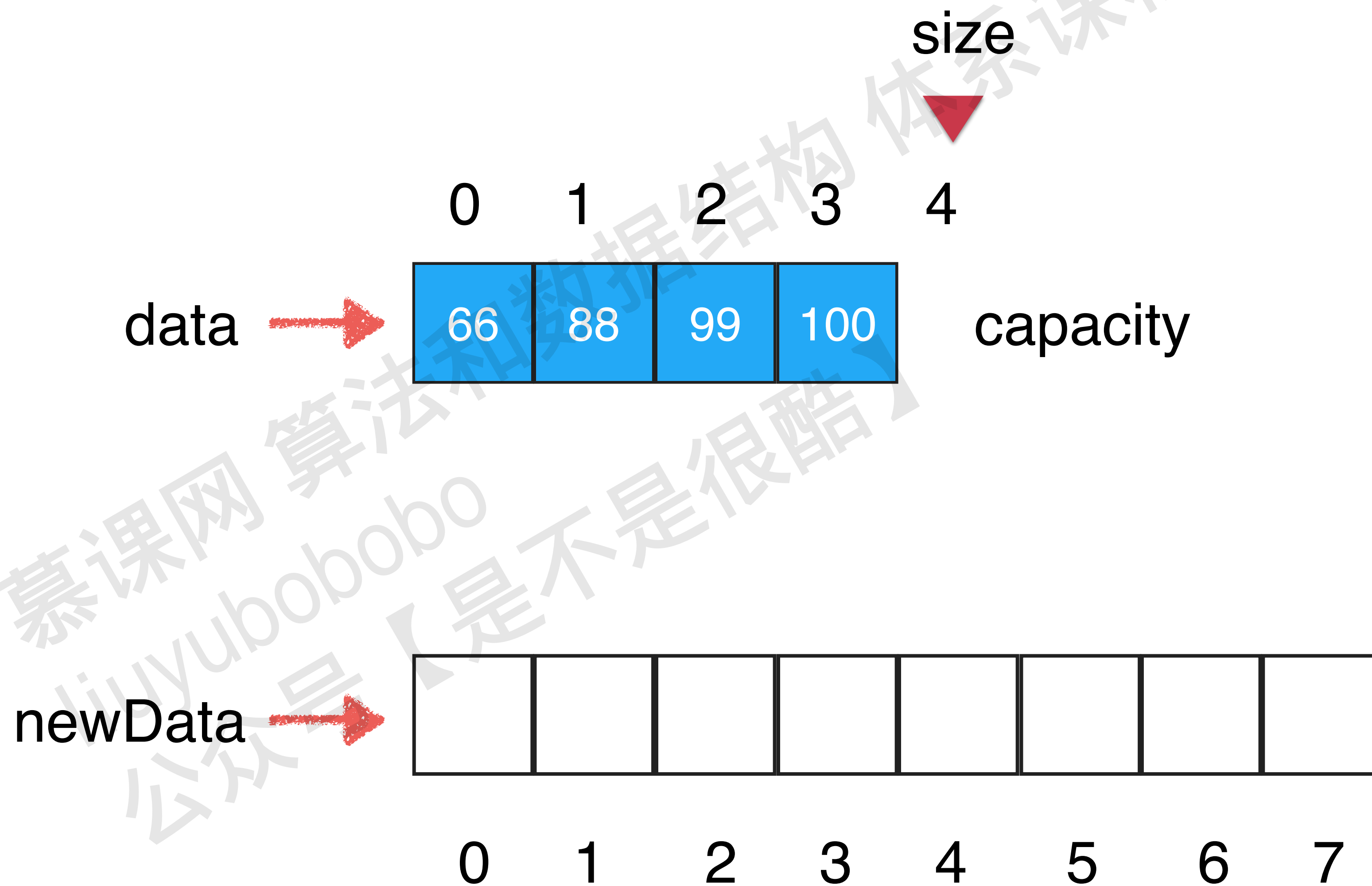
# 动态数组

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

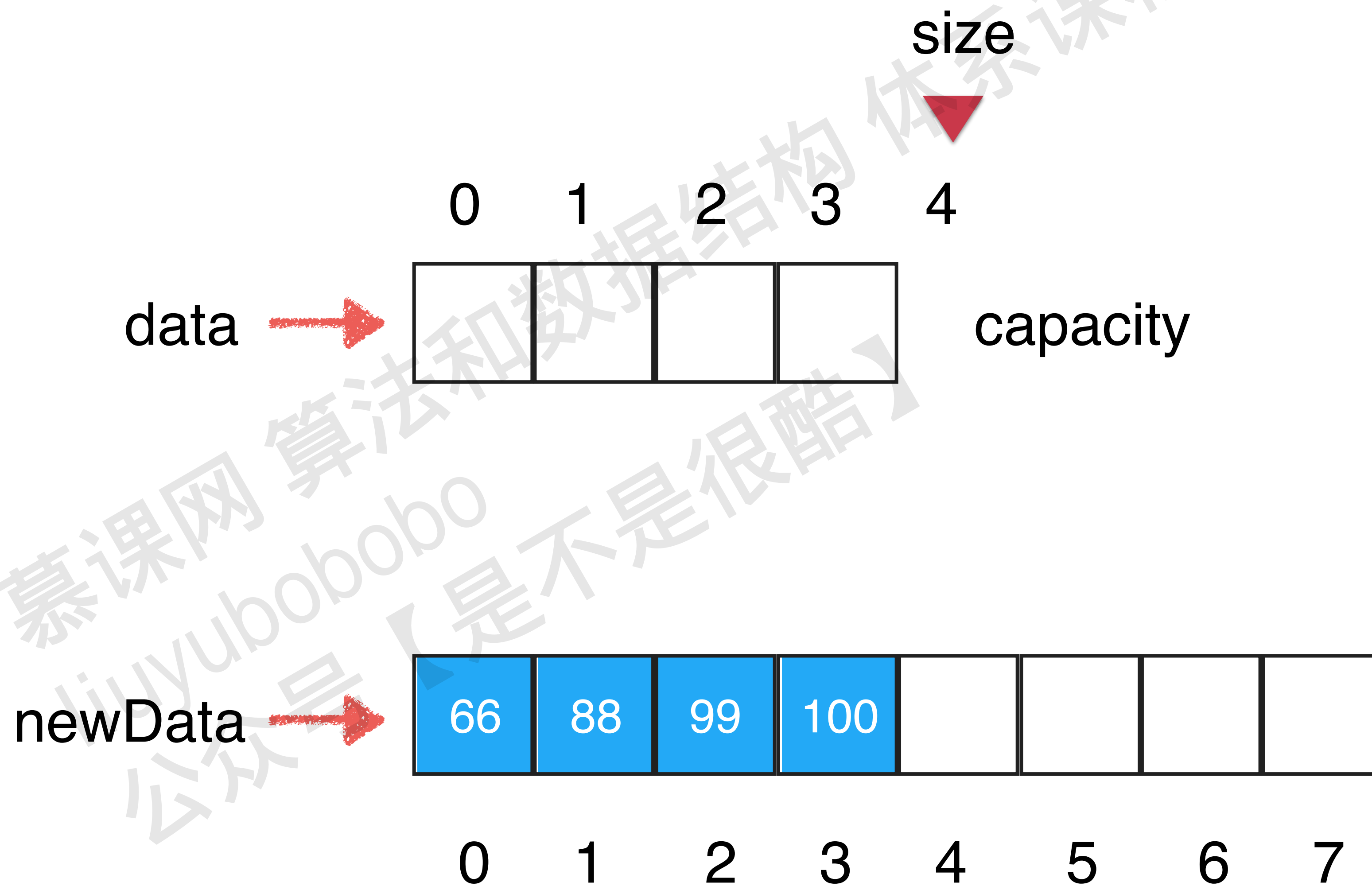
# 动态数组



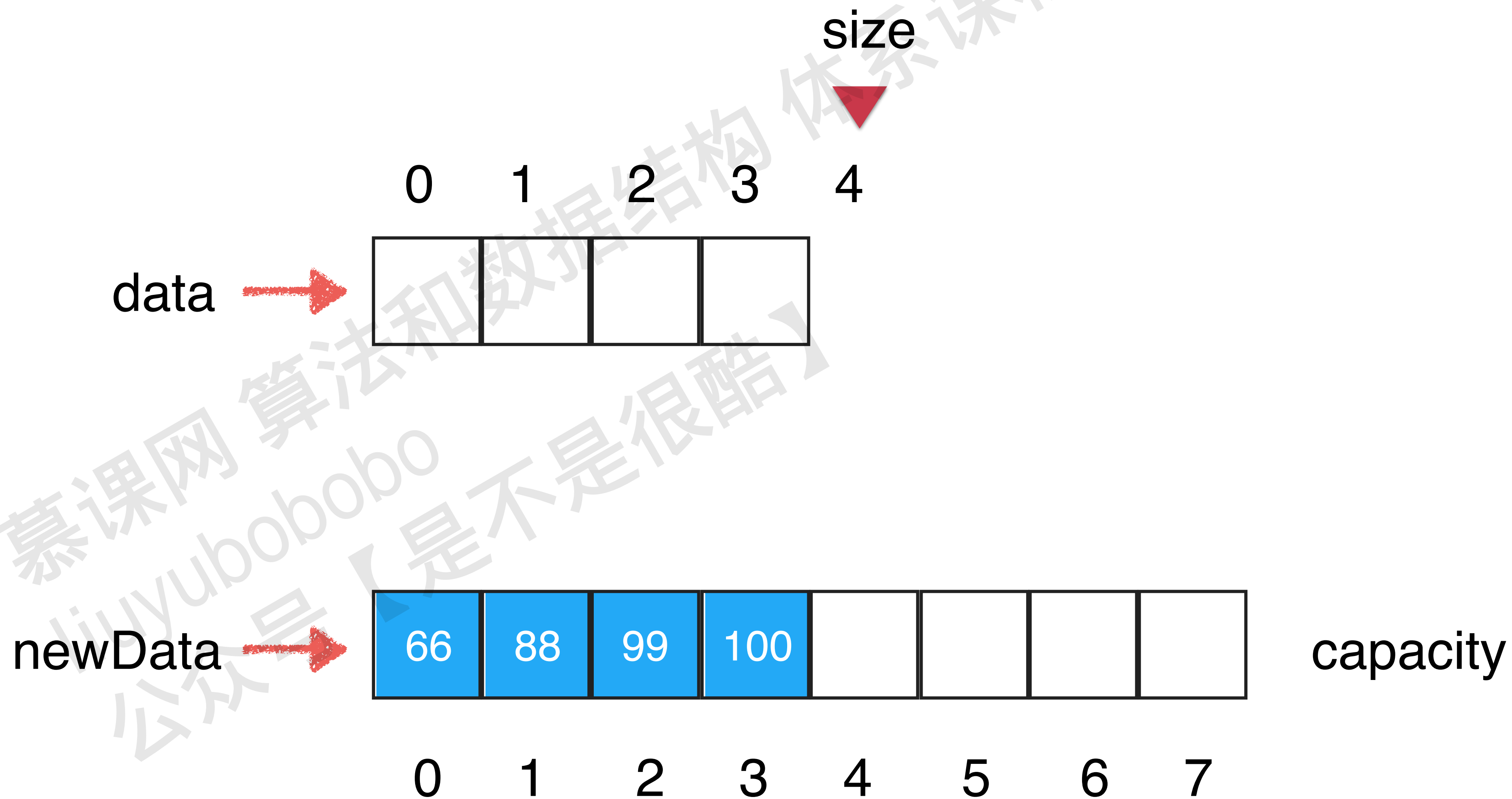
# 动态数组



# 动态数组

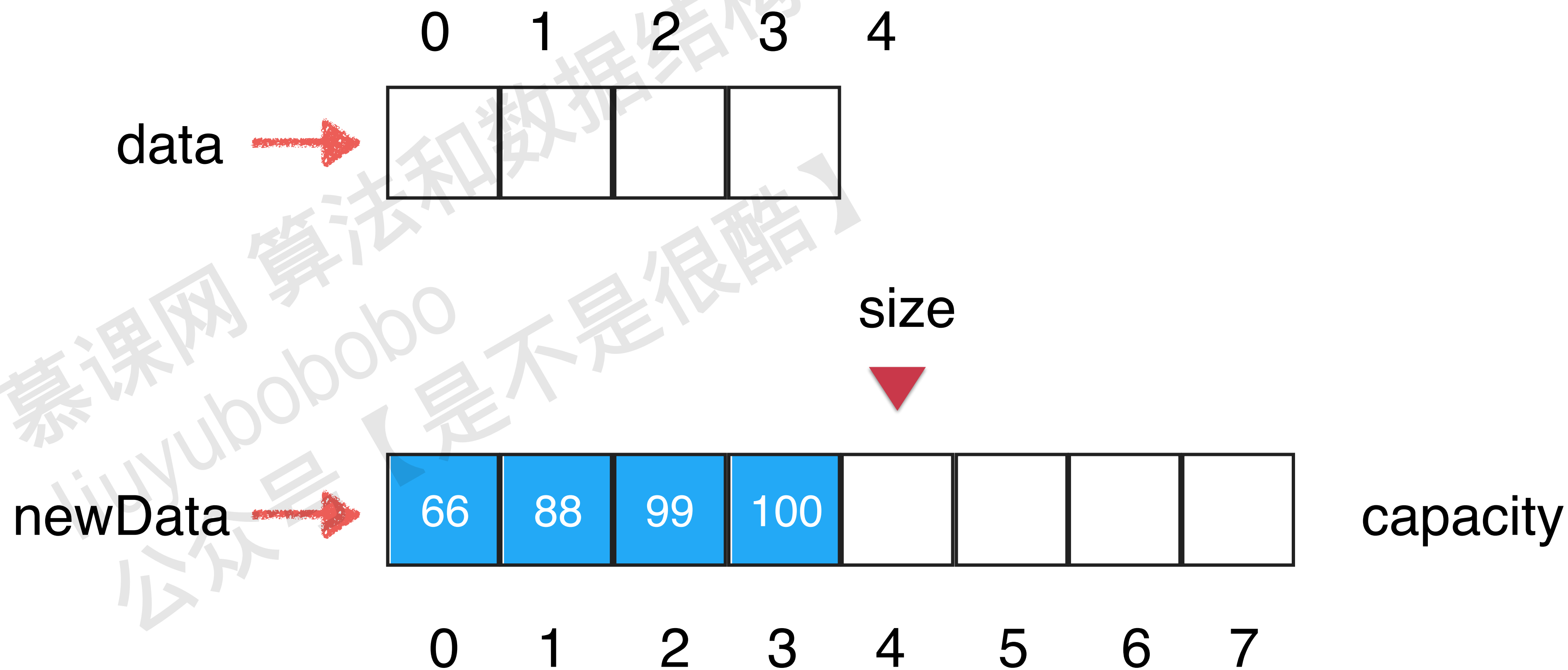


# 动态数组

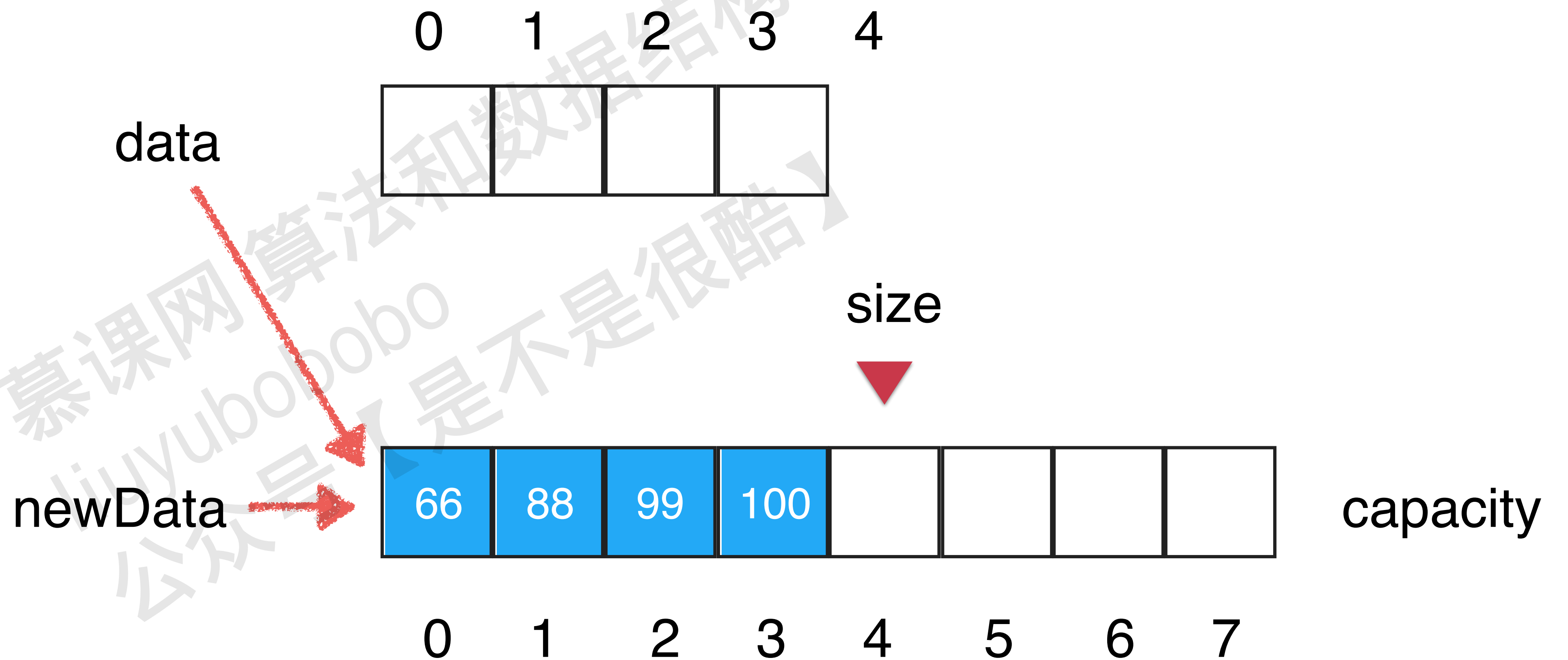




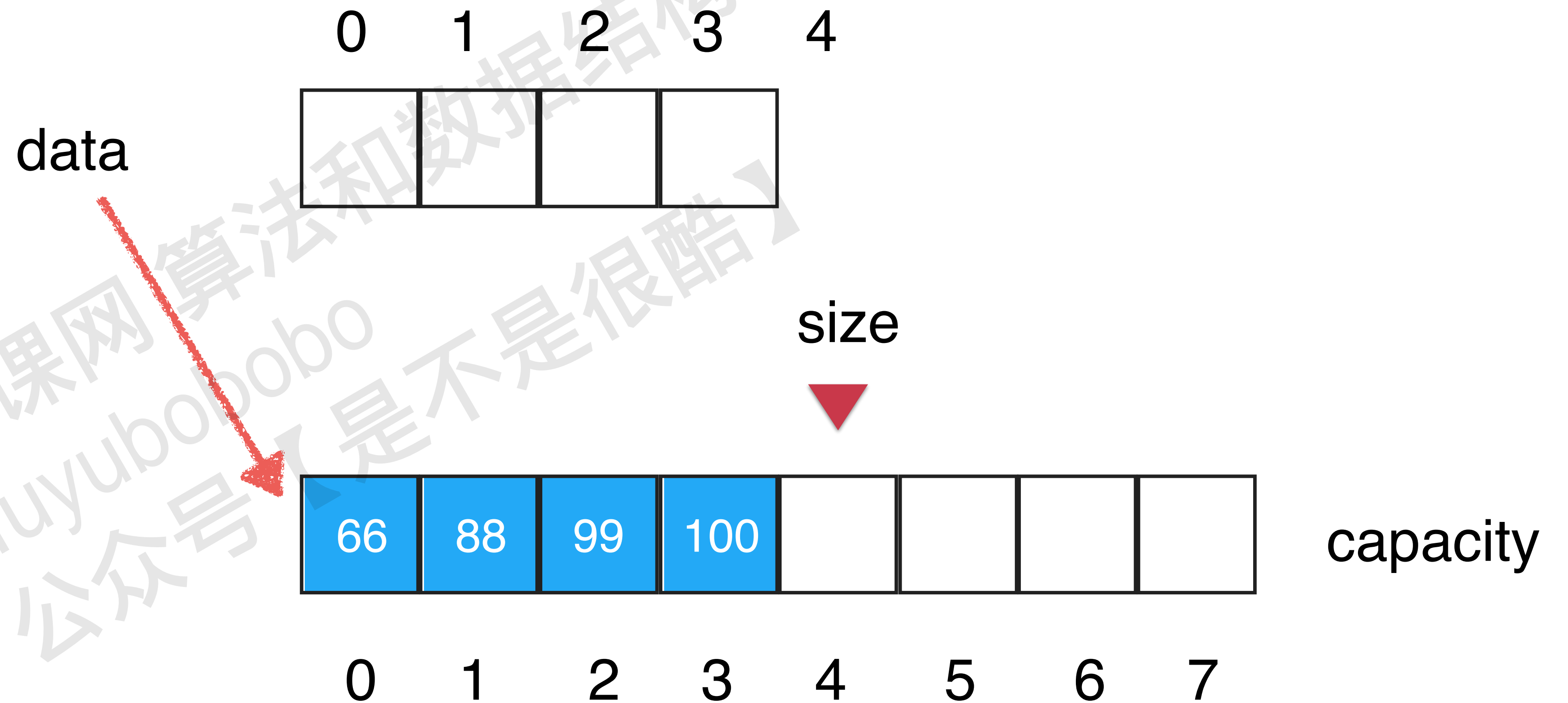
# 动态数组



# 动态数组



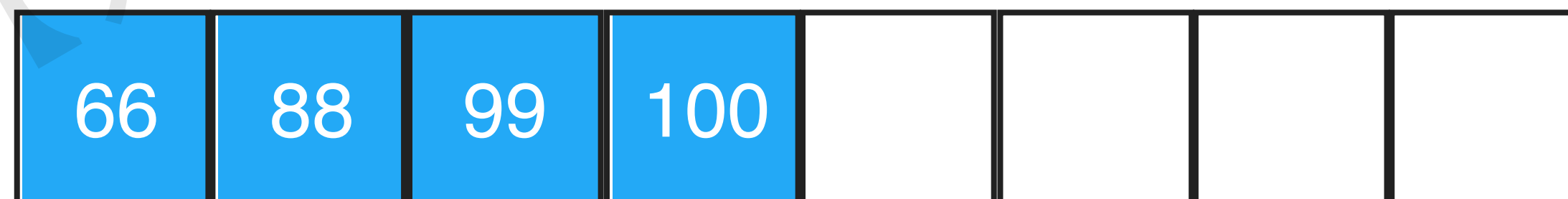
# 动态数组



# 动态数组

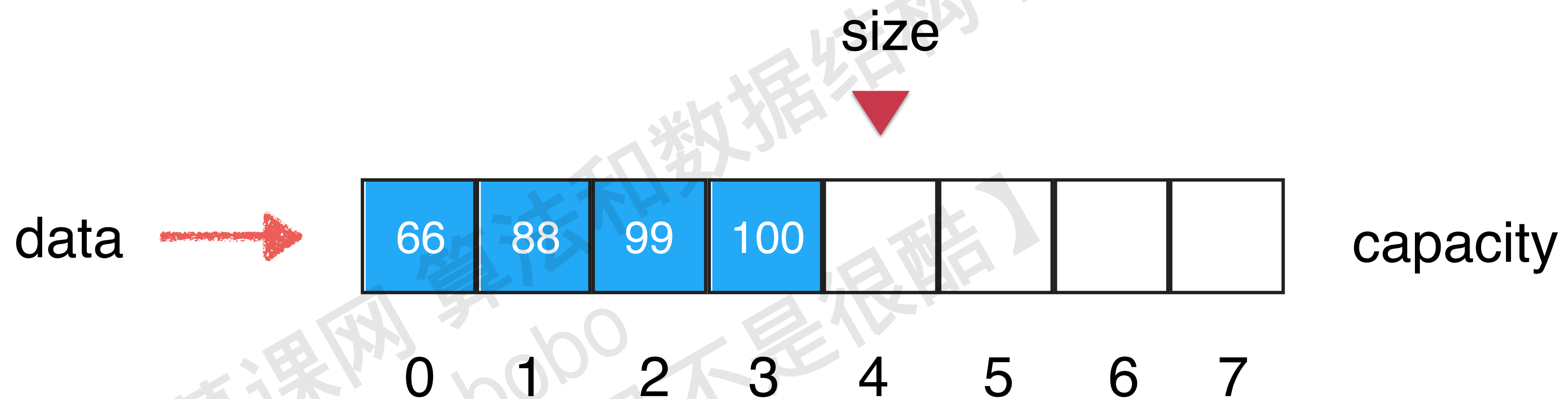
data

size



capacity

# 动态数组



# 实践：动态数组

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

# 简单的时间复杂度分析

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



# 分析动态数组的时间复杂度

- 添加操作  $O(n)$

addLast(e)  $O(1)$

addFirst(e)  $O(n)$

add(index, e)  $O(n/2) = O(n)$

$O(n)$

resize  $O(n)$

最坏情况

严格计算需要一些概率论知识

# 分析动态数组的时间复杂度

- 删除操作  $O(n)$

removeLast(e)  $O(1)$

removeFirst(e)  $O(n)$

remove(index, e)  $O(n/2) = O(n)$

$O(n)$

resize  $O(n)$

# 分析动态数组的时间复杂度

- 修改操作  $O(1)$

set(index, e)  $O(1)$

# 分析动态数组的时间复杂度

- 查找操作

get(index)       $O(1)$

contains(e)       $O(n)$

find(e)       $O(n)$

# 分析动态数组的时间复杂度

- 增： $O(n)$

- 删： $O(n)$

- 改：已知索引  $O(1)$ ；未知索引  $O(n)$

- 查：已知索引  $O(1)$ ；未知索引  $O(n)$

如果只对最后一个元素操作  
依然是 $O(n)$ ？因为resize？

均摊复杂度分析和防止复杂度震荡

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

# resize的复杂度分析

- 添加操作  $O(n)$

addLast(e)

$O(1)$

addFirst(e)

$O(n)$

add(index, e)

$O(n/2) = O(n)$

$O(n)$

最坏情况

resize  $O(n)$



# resize的复杂度分析

- 添加操作  $O(n)$

**addLast(e)**

**$O(1)$**

addFirst(e)

$O(n)$

add(index, e)

$O(n/2) = O(n)$

$O(n)$

最坏情况

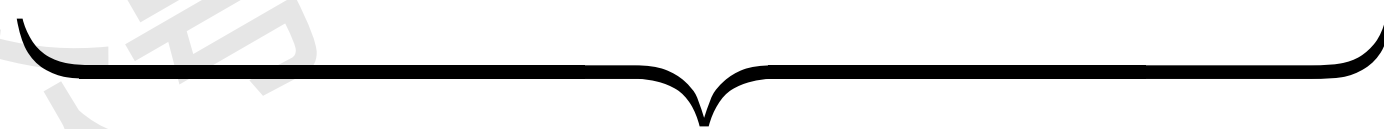
**resize  $O(n)$**

# resize的复杂度分析

resize  $O(n)$

假设当前capacity = 8, 并且每一次添加操作都使用addLast

1 1 1 1 1 1 1 1 8 + 1



9次addLast操作, 触发resize, 总共进行了17次基本操作

# resize的复杂度分析

**resize**  $O(n)$

9次addLast操作，触发resize，总共进行了17次基本操作

平均，每次addLast操作，进行2次基本操作

假设capacity =  $n$ ， $n+1$ 次addLast，触发resize，总共进行 $2n+1$ 次基本操作

平均，每次addLast操作，进行2次基本操作

# resize的复杂度分析

**resize**  $O(n)$

平均，每次addLast操作，进行2次基本操作

这样均摊计算，时间复杂度是 $O(1)$ 的！

在这个例子里，这样均摊计算，比计算最坏情况有意义。

# 均摊复杂度 amortized time complexity

**resize**  $O(n)$

addLast 的均摊复杂度为 $O(1)$

同理，我们看removeLast操作，均摊复杂度也为 $O(1)$

# 复杂度震荡

但是，当我们同时看addLast和removeLast操作：



capacity = n

addLast       $O(n)$

removeLast    $O(n)$

# 复杂度震荡

但是，当我们同时看addLast和removeLast操作：



capacity = n

addLast  $O(n)$

removeLast  $O(n)$

addLast  $O(n)$

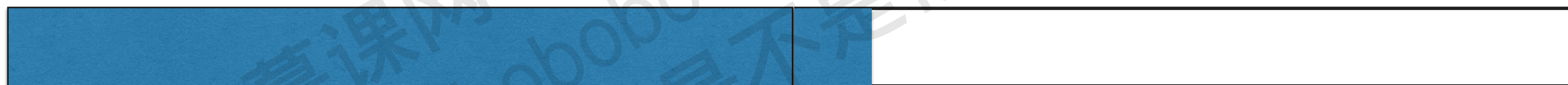
removeLast  $O(n)$



# 复杂度震荡

出现问题的原因：removeLast 时 resize 过于着急（Eager）

解决方案：Lazy



# 复杂度震荡

出现问题的原因：removeLast 时 resize 过于着急（Eager）

解决方案：Lazy



当  $\text{size} == \text{capacity} / 4$  时，才将capacity减半

# 实践：防止复杂度震荡

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

不要小瞧数组

慕课网 算法与数据结构 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