EDU
CSDN学院 IT实战派

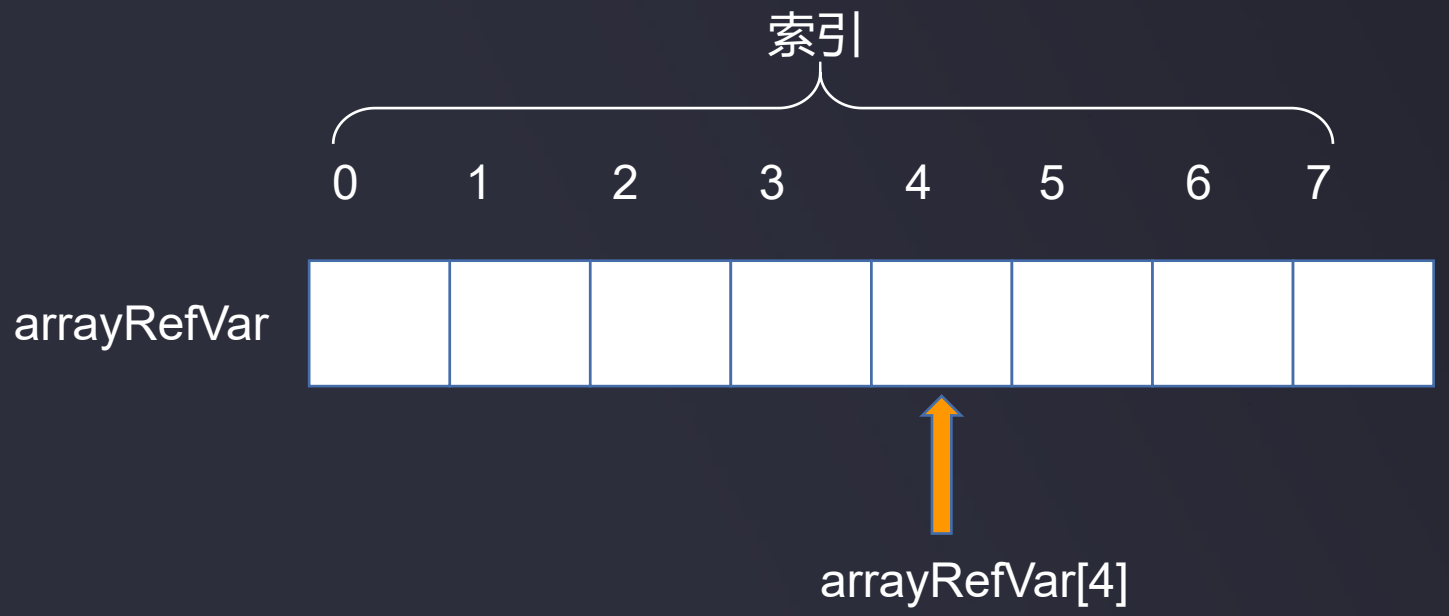
图解数据结构和算法

数组

讲师：Samuel

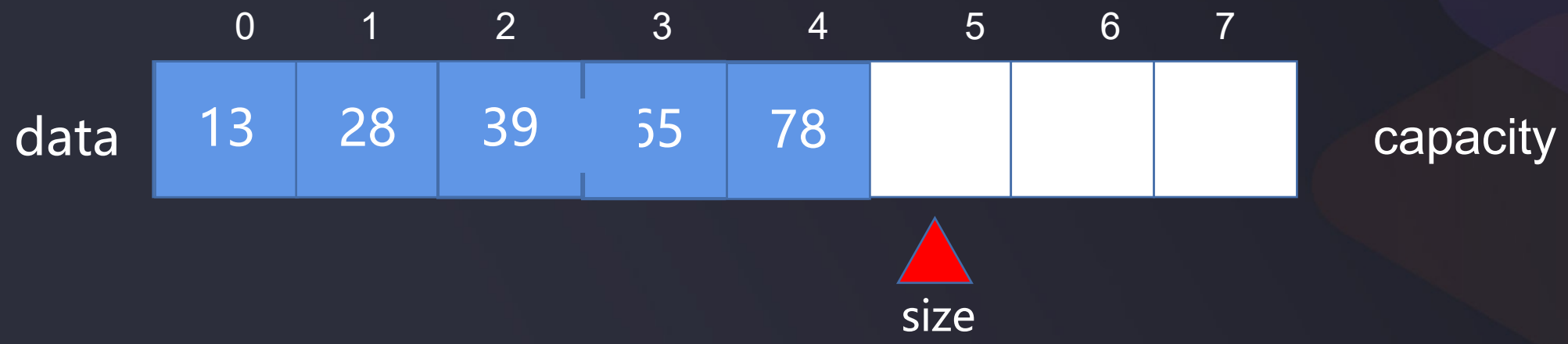
本章概述

- 数组是内存中一片连续的内存空间
- 可以根据索引下标非常快地定位到某一个元素
- 但是新增、删除元素的时候需要移动元素位置以保证其连续，效率会变低



本章概述

本章概述



addLast

removeLast

addLast

removeLast

Content

数组的特征

1 数组的声明和创建

- Java 语言中提供的数组是用来存储固定大小的同类型元素
- Java中可以使用两种方式来声明数组
  - dataType[] arrayRefVar
  - dataType arrayRefVar[]
- Java中数组的创建方式同样有两种
  - arrayRefVar = new dataType[arraySize]
  - dataType[] arrayRefVar = {value0, value1, ..., valuek}

自定义数组

1 数组索引的语意

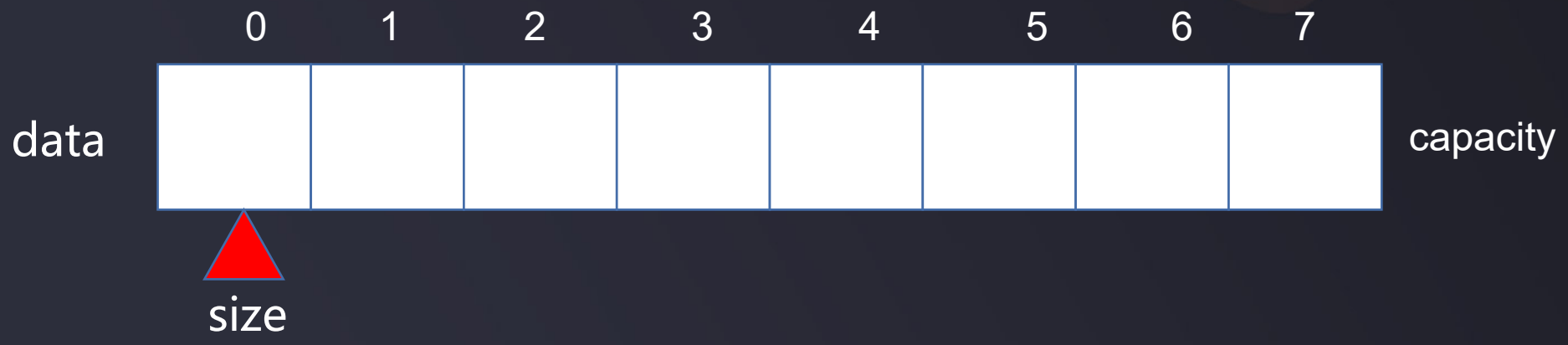
- 索引可以有语意，也可以无语意
- 有语意的情况下数组最大的优点就是快速查询，scores[2]
- 没有语意的下数组的如何判断没有元素？添加元素？删除呢？

自定义数组

2 自定义数组

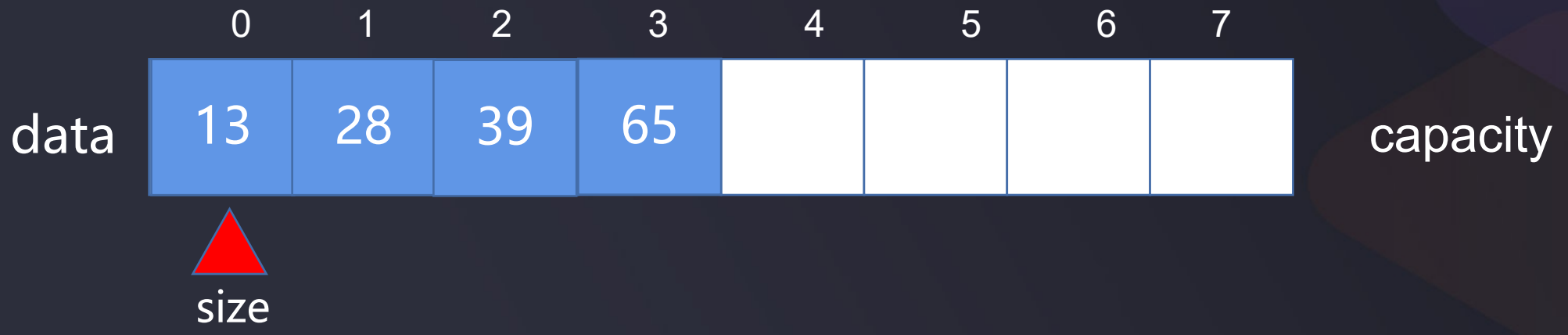
```
public class void Array() {
    private int[] data;
    private int size;
//    private int capacity;
}
```

增 删 改 查



数组基本操作

1 数组添加元素

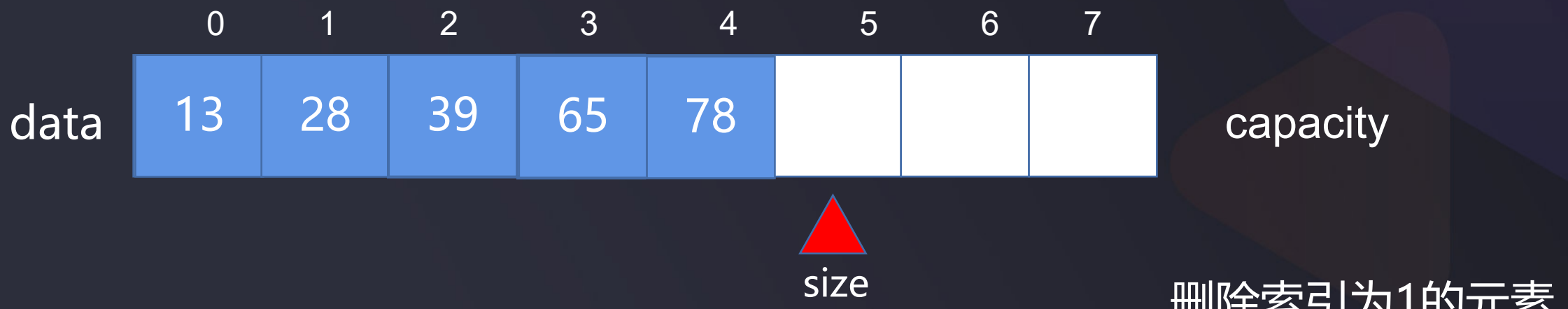


2 数组查询和修改元素

- 给定索引获得到元素
- 设置某个索引位置上的元素为e
- contains和find

数组基本操作

3 数组删除元素



删除索引为1的元素

数组基本操作

4 泛型数组

- 改造自定义数组使其可以存放任何类型的元素
- 可以使用泛型来约束数组中的元素，需注意只能存放类对象，不能使用基本类型
- 基本类型都有对应的包装类，java内部可以自动拆装箱完成基本类型和包装类的相互转换

数组基本操作

5 动态数组

时间复杂度分析

1 时间复杂度

原则：忽略常数项、忽略低次项、忽略系数

时间复杂度由小到大依次为：$O(1) < O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < O(n^3) < O(n^k) < O(2^n)$

2 添加元素时间复杂度 O(n)

addLast(e)	O(1)	最坏情况 O(n)	resize()	O(n)
addFrist(e)	O(n)			
add(index,e)	O(n/2)=O(n)			

时间复杂度分析

3 删除元素时间复杂度 O(n)

removeLast(e) O(1)

removeFrist(e) O(n)

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

最坏情况 O(n)

resize() O(n)

4 修改元素时间复杂度 O(1)

set(index,e) O(1)

时间复杂度分析

5 查找元素时间复杂度 O(n)

get(index)	O(1)
contains(e)	O(n)
find(e)	O(n)

已知索引O(1)，未知索引O(n)

6 时间复杂度分析

增	O(n)
删	O(n)

如果只对最后一个元素操作，因为有resize操作，所以依然是O(n)

改和查 已知索引O(1)、未知索引O(n)

时间复杂度分析

7 均摊时间复杂度

- 由于存在resize操作，所以即使向最后一个位置添加元素，那么添加的最坏的时间复杂度是O(n)
- 不可能每次向最后一个位置添加元素都触发resize，那么使用最坏时间复杂度分析添加算法不合理
  - 假设当前的capacity=8，并且每一次添加都是使用的addLast
  - 当第9次添加的时候触发了resize,那么需要将原来的8个元素copy并添加第9个新元素，总共执行了17次
  - 对于添加9个元素来说每次addLast平均上大概进行了17/9=2次的操作
  - 结论：capacity=n,n+1次的addLast操作一共执行了2n+1次操作，平均每次addLast进行2次操作
  - 这样均摊计算，时间复杂度就是O(1)级别的，对于addLast操作均摊复杂度比最坏复杂度更有参考意义
- 同理，removeLast的均摊时间复杂度依然是O(1)的

时间复杂度分析

8 复杂度的震荡

- 同时进行addLast和removeLast的时候回发现每次都需要进行resize操作，那么就会出现复杂度的震荡

addLast

removeLast

- 出现这个问题的原因就是删除的时候过于eager了
- 解决方案是可以适当lazy处理

addLast

removeLast

addLast

removeLast

removeLast

二维数组

1 声明和初始化

- 二维数组其实就是盛放数组的数组，换言之二维数组就是一个元素为一维数组的数组
- 数据类型[][] 变量名=new 数据类型[m][n]
  - m表示这个二维数组有多少个数组
  - n表示每一个一维数组的元素个数
- 数据类型[][] 变量名=new 数据类型[m][];
  - m表示这个二维数组有多少个数组
  - 这一次没有直接给出一维数组的元素个数，可以动态的给出
- 数据类型[][] 变量名={{元素...},{元素...},{元素...}}

二维数组

2 二维数组的使用

- 某公司按照季度和月份统计的数据如下

  第一季度：22，66，44

  第二季度：77，33，88

  第三季度：11，66，99

  第四季度：25，45，65

- 求公司的年销售额

二维数组

3 稀疏数组sparsearray

- 当数组中大部分元素为0，或者为同一个值的数组时，可以使用稀疏数组来保存该数组
  - 记录数组一共有几行几列，有多少个不同的值
  - 把具有不同值的元素的行列及值记录在一个小规模的数组中，从而缩小程序的规模

$$\begin{pmatrix} 0 & 0 & 0 & 22 & 0 & 0 & 15 \\ 0 & 11 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 39 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 \end{pmatrix}$$

	row	col	val
[0]	6	7	8
[1]	0	3	22
[2]	0	6	15
[3]	1	1	11
[4]	1	5	17
[5]	2	3	16
[6]	3	5	39
[7]	4	0	91

二维数组

4 稀疏数组的应用

- 某些平面的游戏，会用到很多的稀疏数组来存储eg:棋盘、地图等等
- 编写的五子棋程序中，有存盘退出和续上盘的功能

EDU

CSDN学院 IT实战派

下节课再见，记得关注公众号