算法与数据结构体系课程

liuyubobobo

MSD 基数排序和桶排序

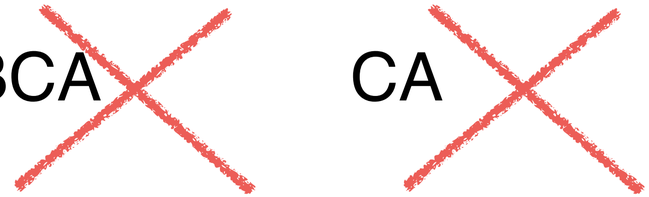
MSD 基数排序

MSD 基数排序

LSD　　Least Significant Digit

从最后向前计数排序　　只适用于等长字符串

CA	CA	BAC	AC	BAC
BCA	BCA	AC	CA	BCA
BAC	BAC	CA	BAC	AC
AC	AC	BCA	BCA	CA



MSD 基数排序

LSD　　Least Significant Digit

从最后向前计数排序　　只适用于等长字符串

CA
BCA
BAC
AC

CA
BCA
BAC
AC

BAC
AC
CA
BCA

AC
CA
BAC
BCA

BAC
BCA
AC
CA

AC
BAC
BCA
CA

MSD 基数排序

LSD Least Significant Digit

从最后向前计数排序

只适用于等长字符串

CA
BCA
BAC
AC

CA
BCA
BAC
AC

BAC
AC
CA
BCA

AC
CA
BAC
BCA

BAC
BCA
AC
CA

AC
BAC
BCA
CA

MSD 基数排序

LSD　　Least Significant Digit

从最后向前计数排序

只适用于等长字符串

MSD　　Most Significant Digit

从前向后计数排序

可用于不等长字符串

AC

BAC

BCA

CA

MSD 基数排序

BCA

CBAA

AC

BADFE

ABC

CBA

MSD 基数排序

BCA

CBAA

AC

BADFE

ABC

CBA

MSD 基数排序

BCA

CBAA

AC

BADFE

ABC

CBA

AC

ABC

BCA

BADFE

CBAA

CBA

MSD 基数排序

BCA

CBAA

AC

BADFE

ABC

CBA

AC

ABC

BCA

BADFE

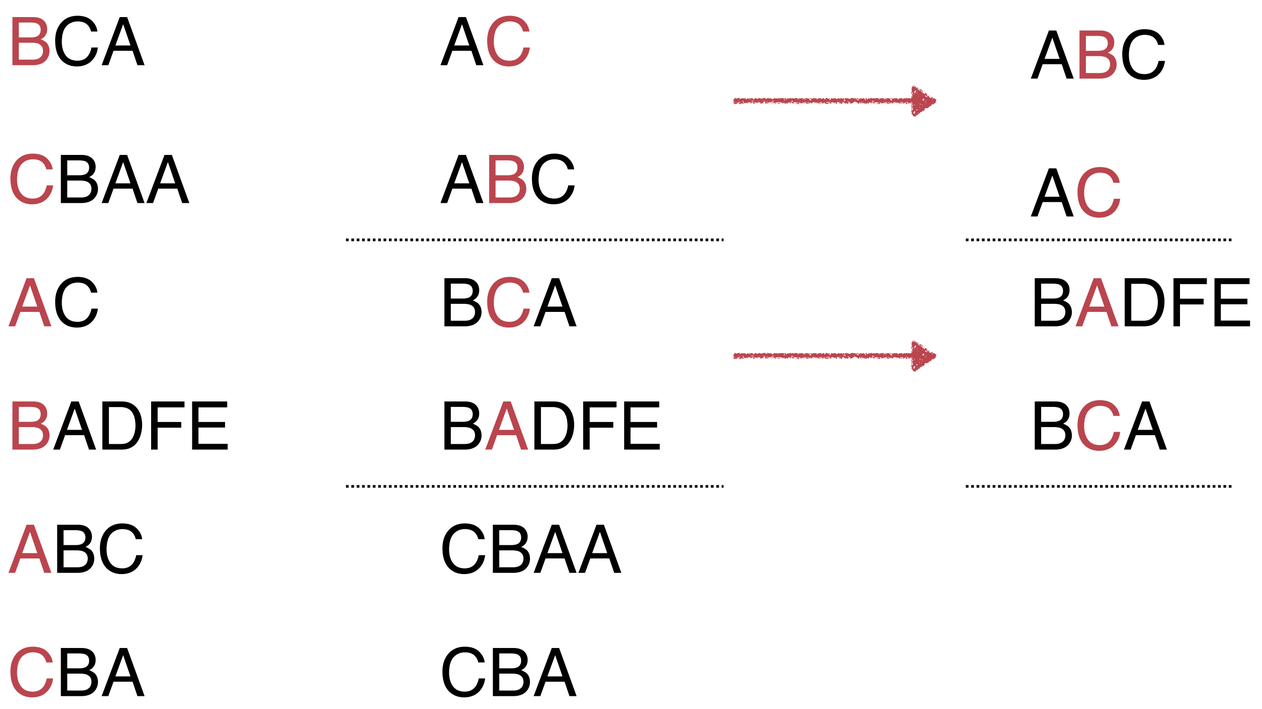
CBAA

CBA

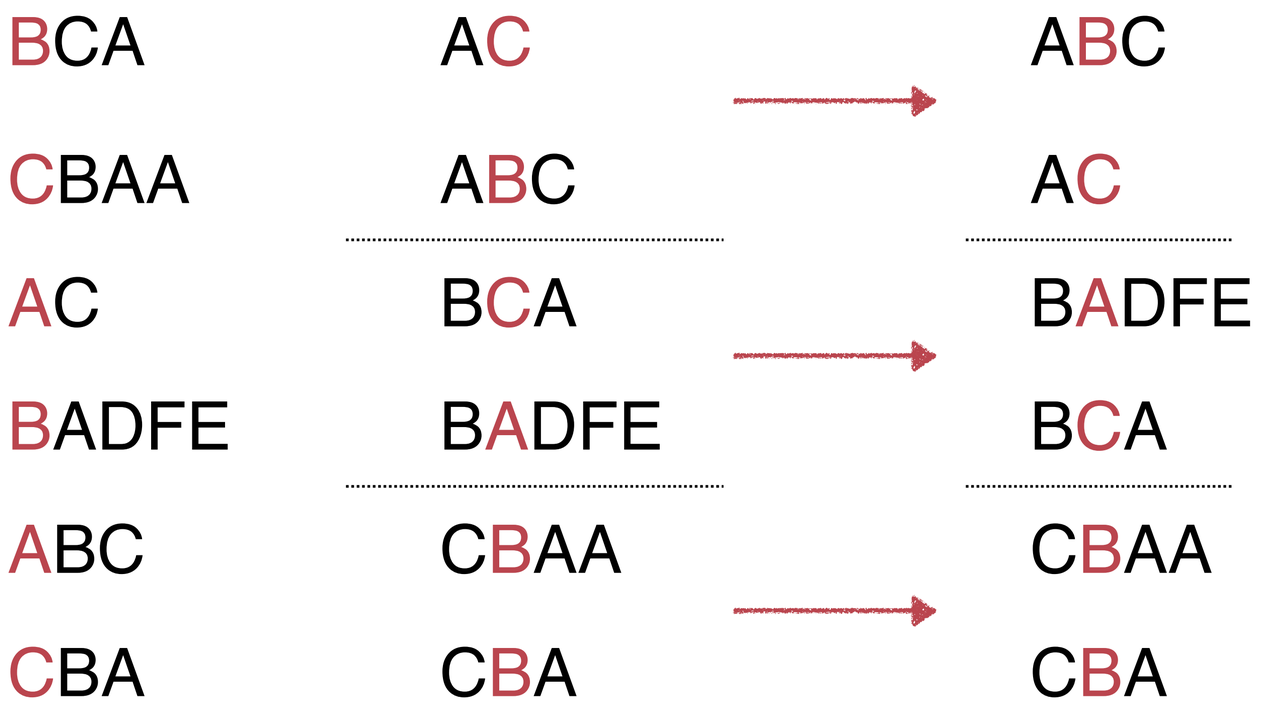
ABC

AC

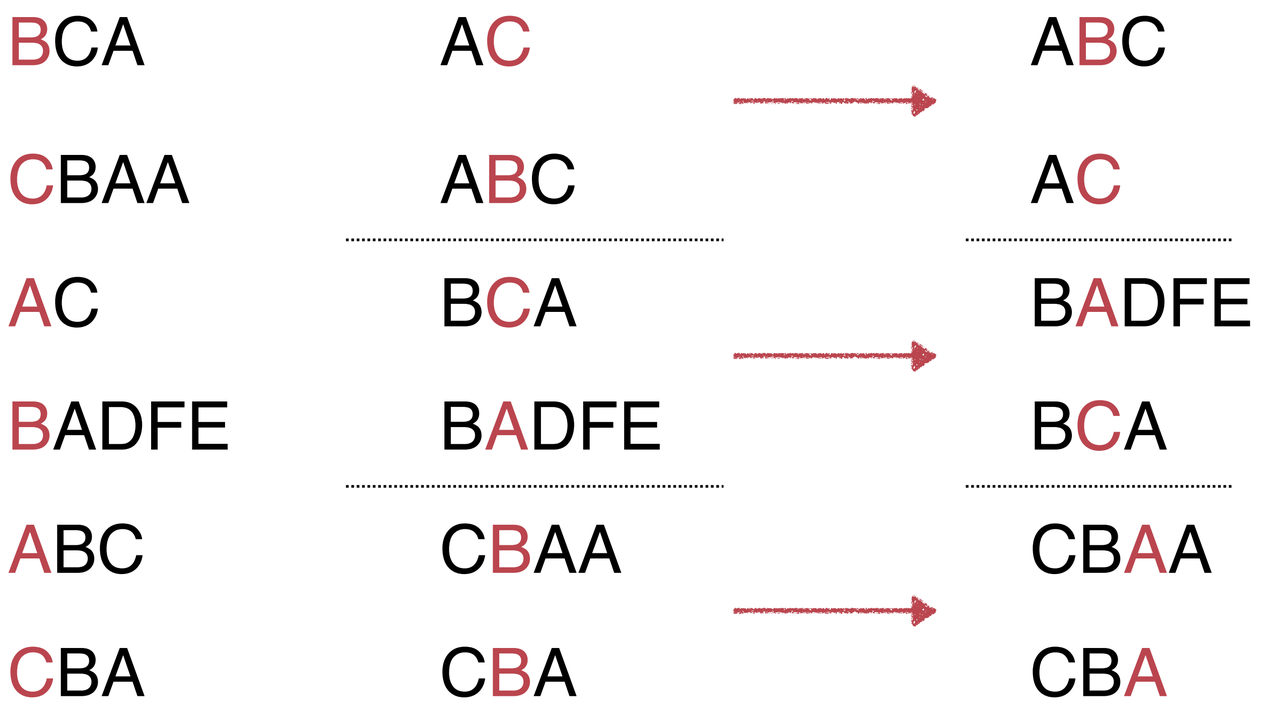
MSD 基数排序


BCA
CBAA
AC
BADFE
ABC
CBA
AC
ABC
BCA
BADFE
CBAA
CBA
ABC
AC
BADFE
BCA

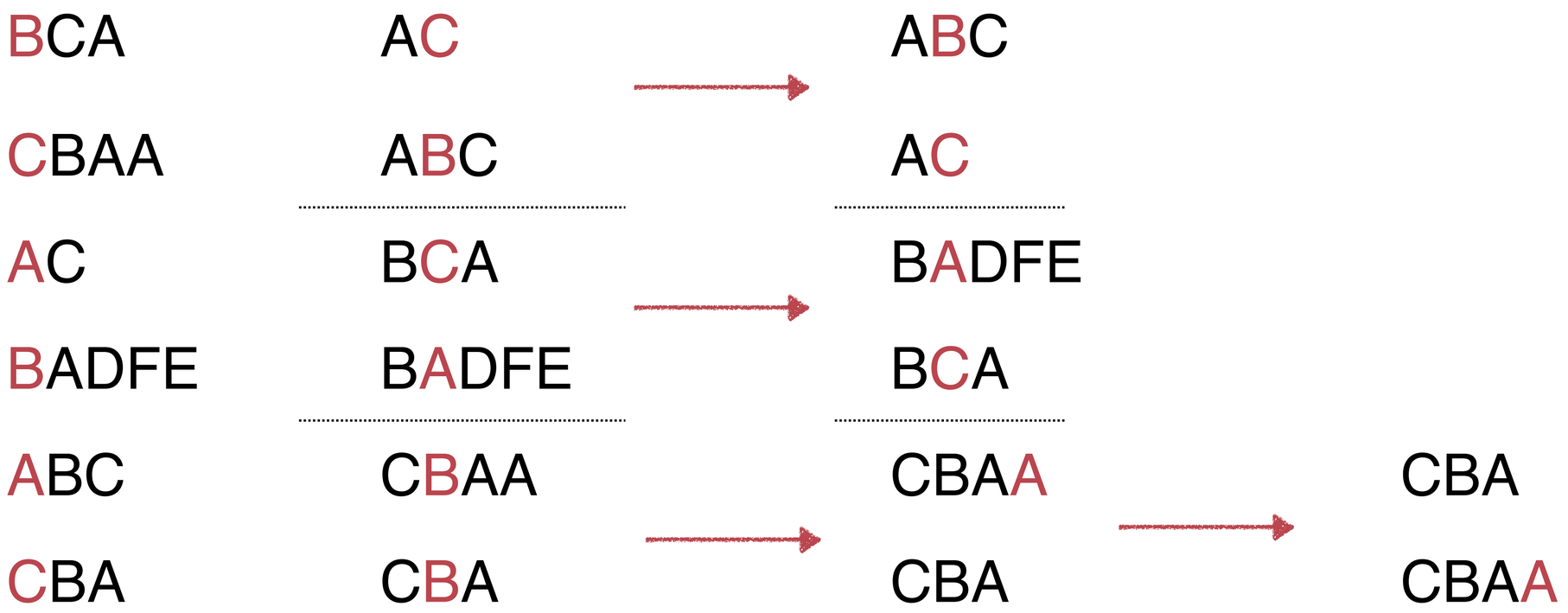
MSD 基数排序


BCA
CBAA
AC
BADFE
ABC
CBA
AC
ABC
BCA
BADFE
CBAA
CBA
ABC
AC
BADFE
BCA
CBAA
CBA

MSD 基数排序


BCA
CBAA
AC
BADFE
ABC
CBA
AC
ABC
BCA
BADFE
CBAA
CBA
ABC
AC
BADFE
BCA
CBAA
CBA

MSD 基数排序



边界处理：空（长度不够）对应的“字符的值”最小

MSD 基数排序

CBAA　　CBA

CBA　　CBAA

边界处理：空（长度不够）对应的“字符的值”最小

每轮计数排序　　字符取值范围：[0...R - 1] R 种可能

添加一种可能：空（长度不够）

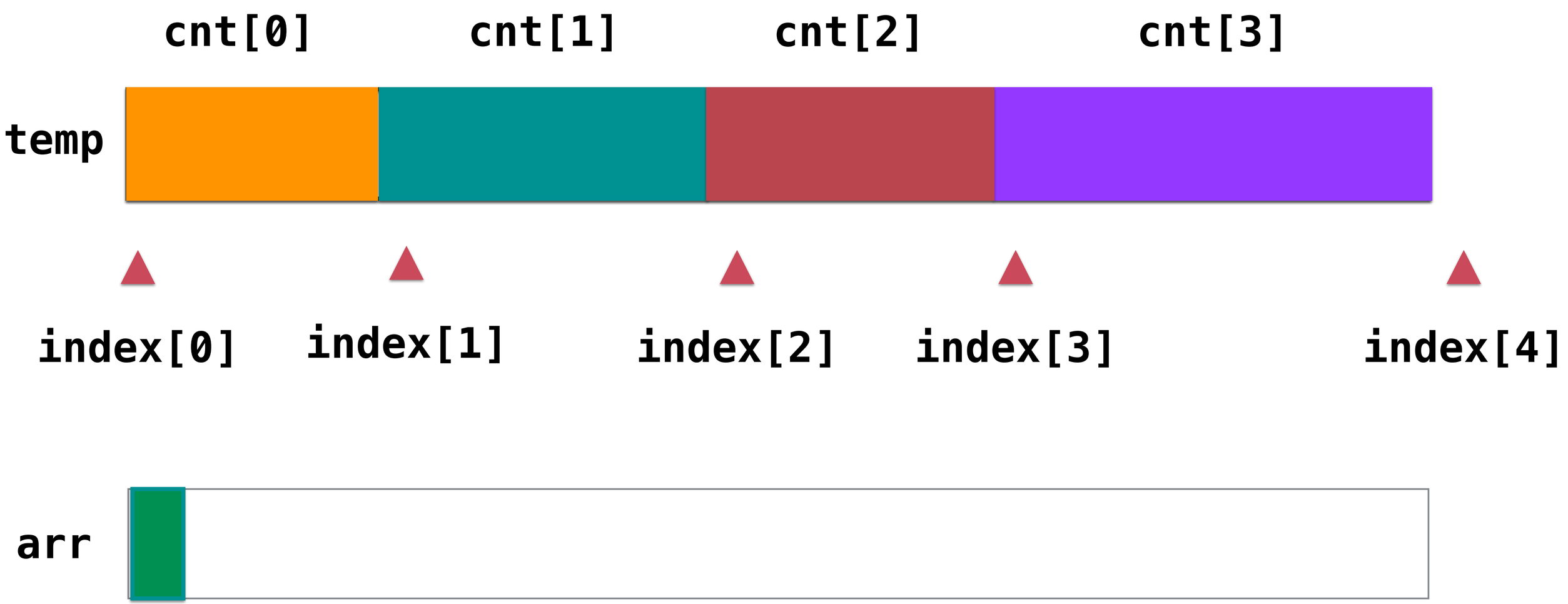
字符取值范围：[1...R]；0 表示空；R + 1 种可能

递归进行计数排序

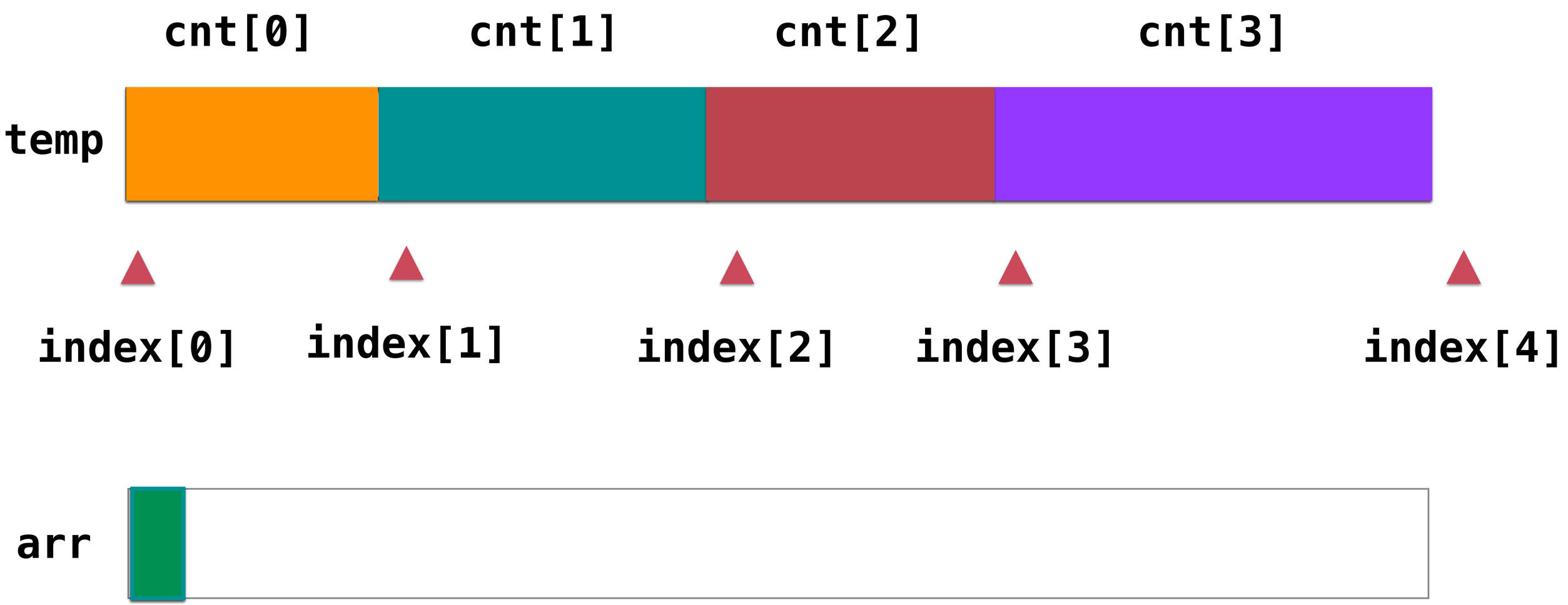
实现 MSD 基数排序

实践：实现 MSD 基数排序

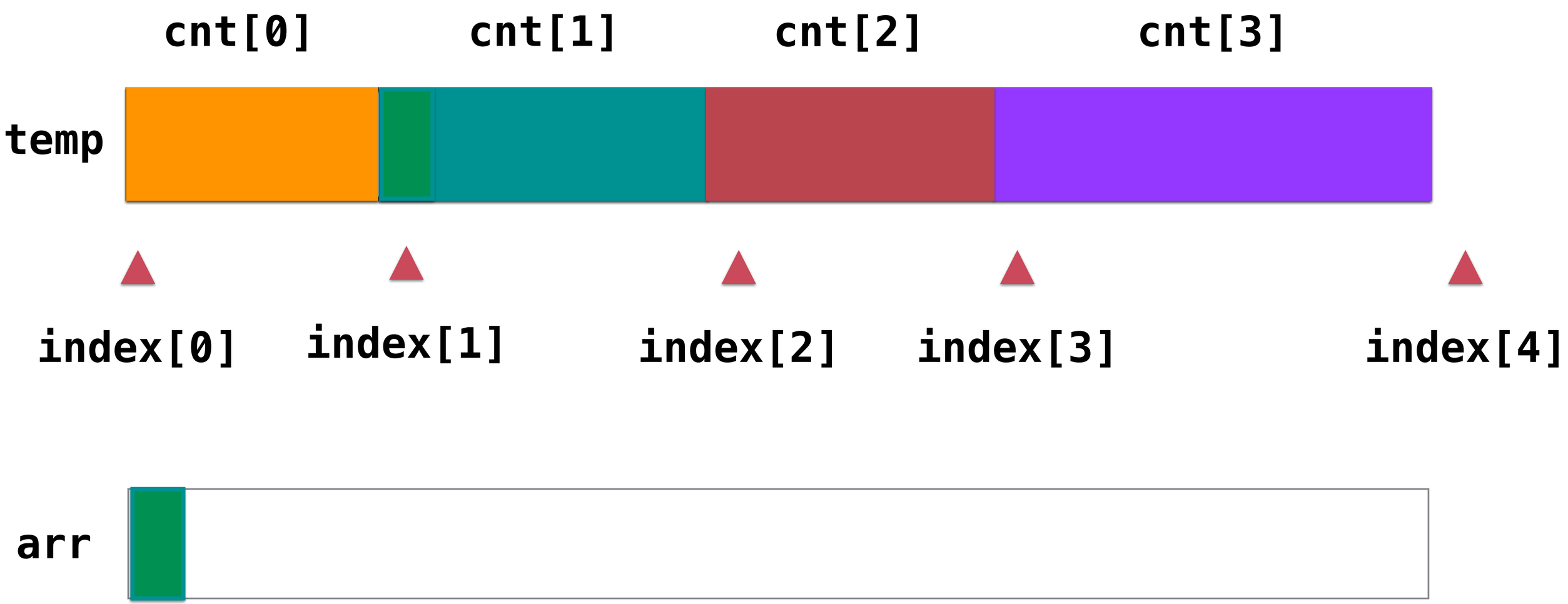
实现 MSD 基数排序


cnt[0]
cnt[1]
cnt[2]
cnt[3]
temp
index[0]
index[1]
index[2]
index[3]
index[4]
arr

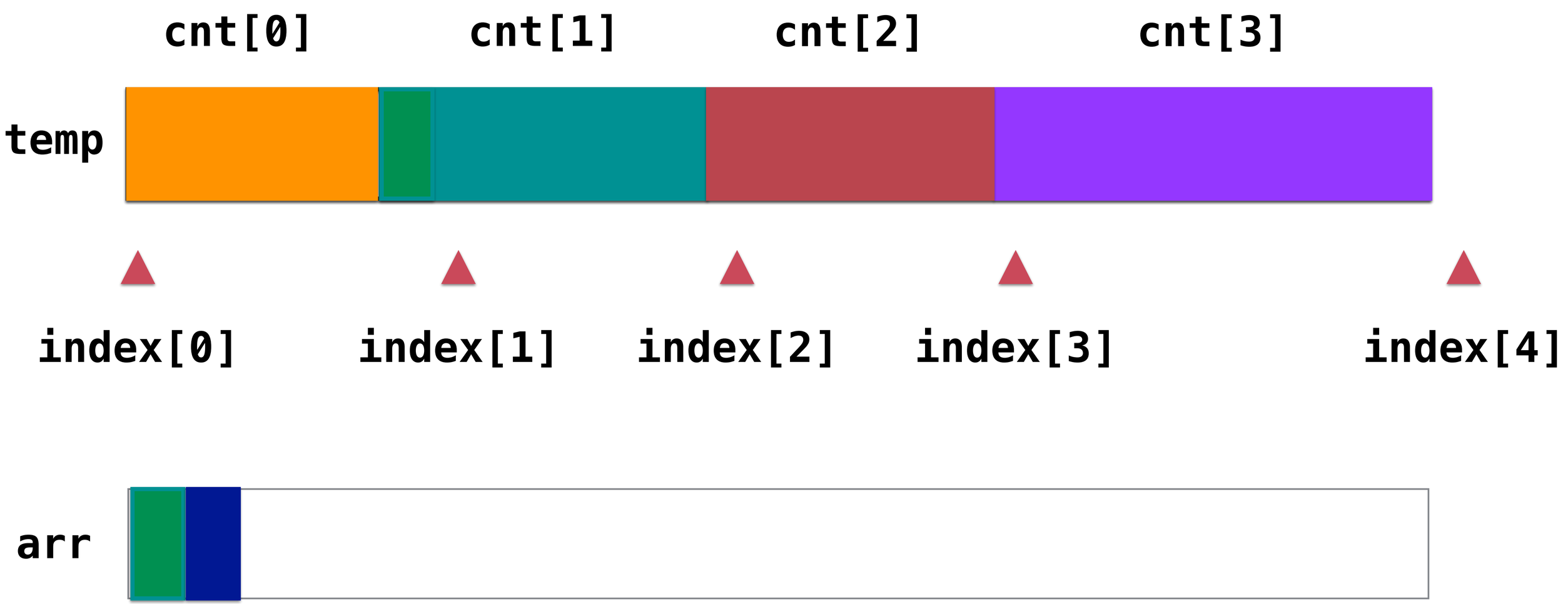
实现 MSD 基数排序


cnt[0]
cnt[1]
cnt[2]
cnt[3]
temp
index[0]
index[1]
index[2]
index[3]
index[4]
arr

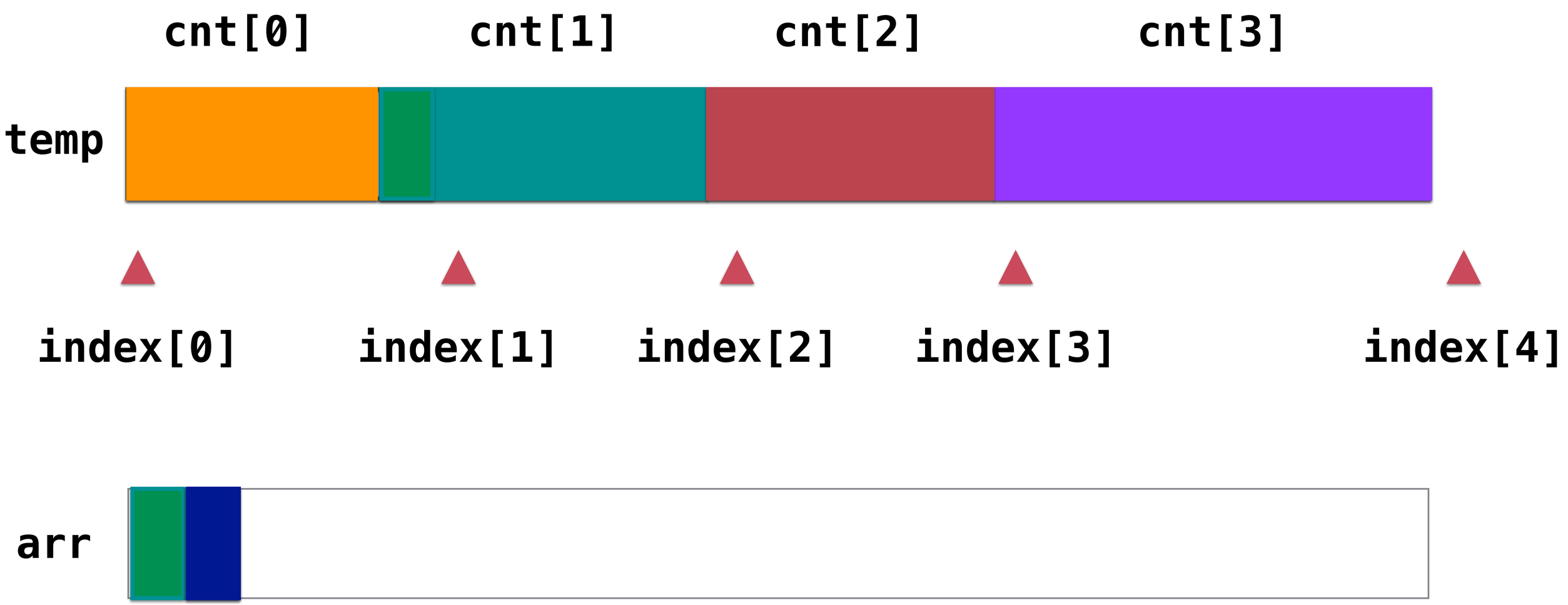
实现 MSD 基数排序


cnt[0]
cnt[1]
cnt[2]
cnt[3]
temp
index[0]
index[1]
index[2]
index[3]
index[4]
arr

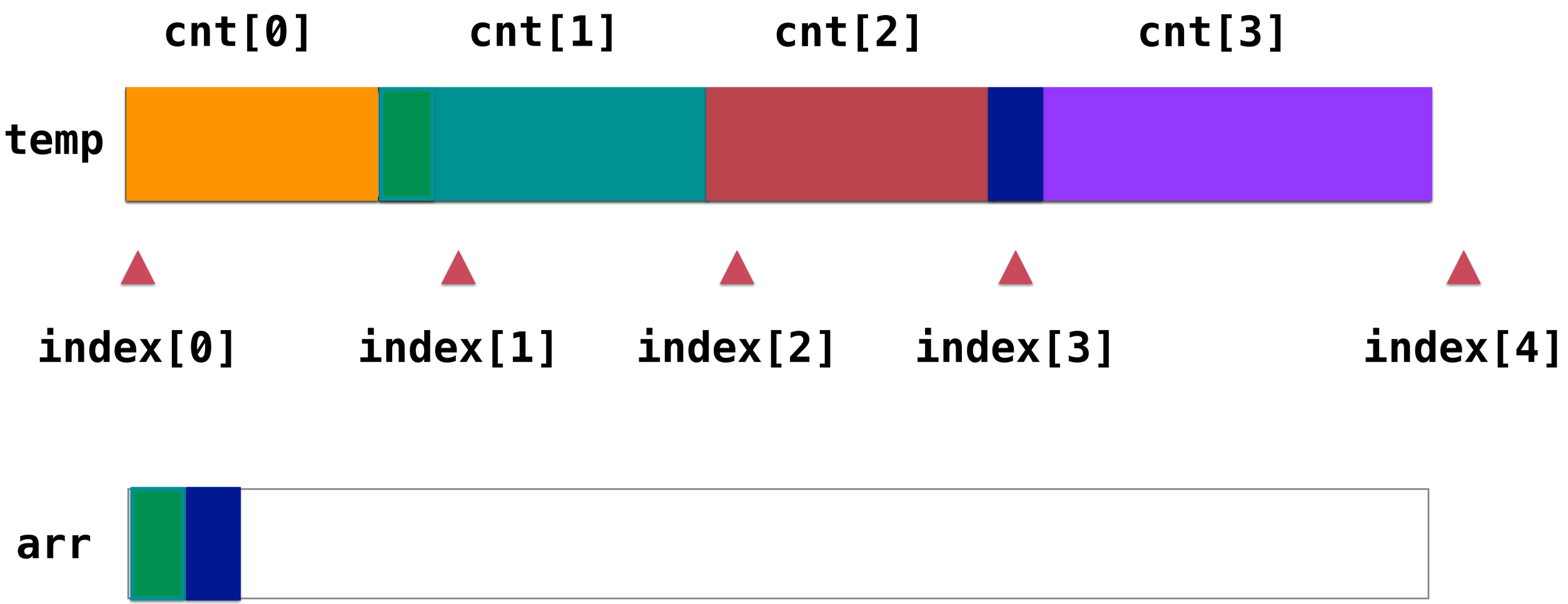
实现 MSD 基数排序


cnt[0]
cnt[1]
cnt[2]
cnt[3]
temp
index[0]
index[1]
index[2]
index[3]
index[4]
arr

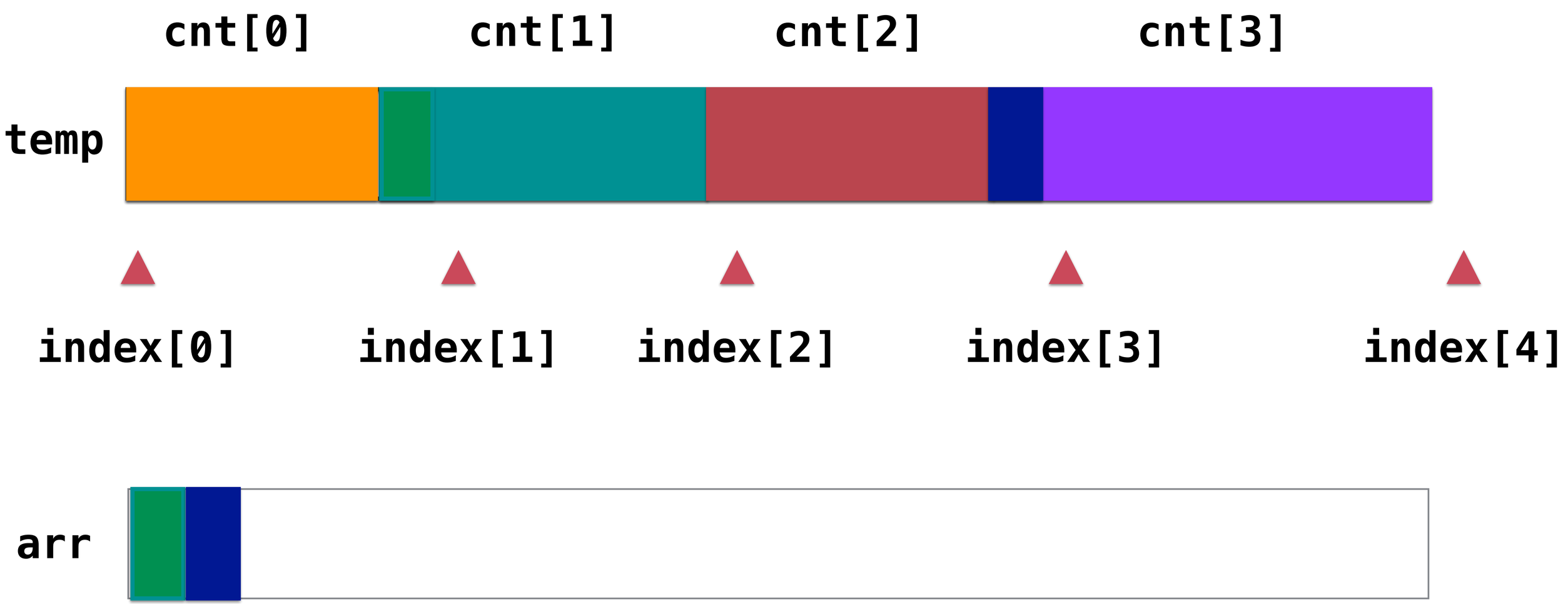
实现 MSD 基数排序


cnt[0]
cnt[1]
cnt[2]
cnt[3]
temp
index[0]
index[1]
index[2]
index[3]
index[4]
arr

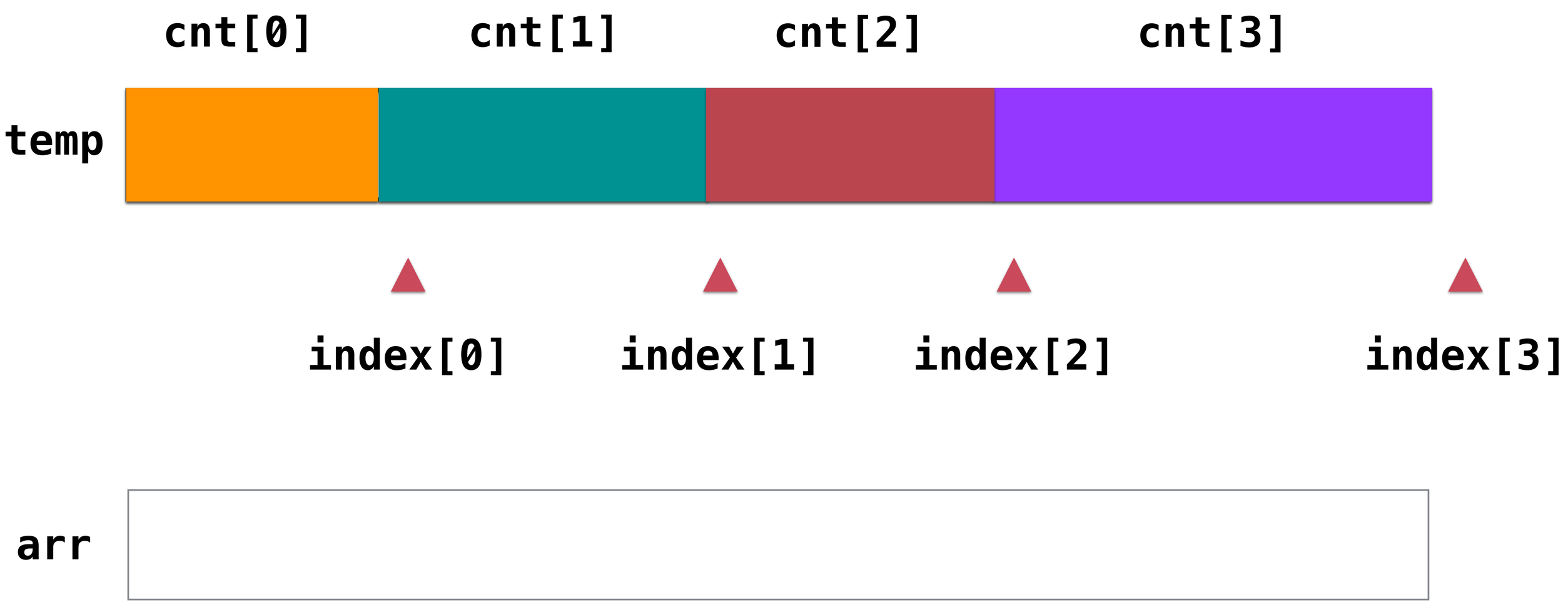
实现 MSD 基数排序


cnt[0]
cnt[1]
cnt[2]
cnt[3]
temp
index[0]
index[1]
index[2]
index[3]
index[4]
arr

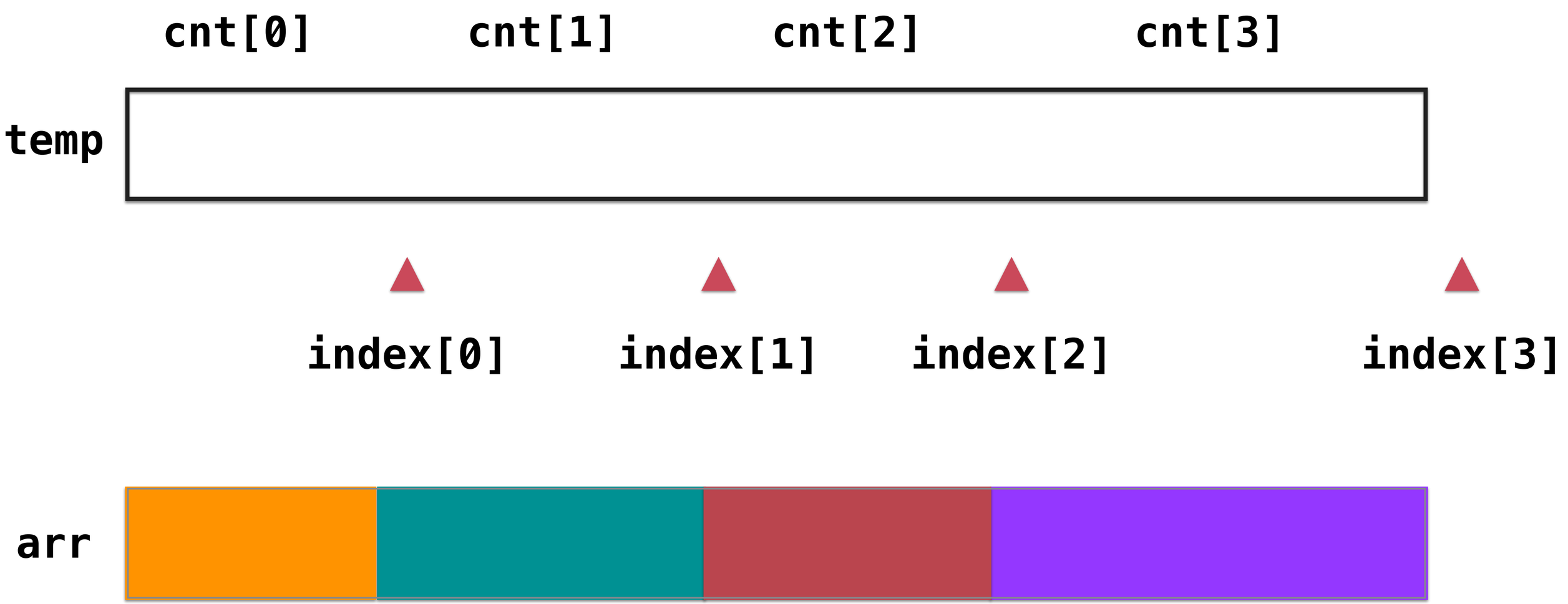
实现 MSD 基数排序


cnt[0]
cnt[1]
cnt[2]
cnt[3]
temp
index[0]
index[1]
index[2]
index[3]
index[4]
arr

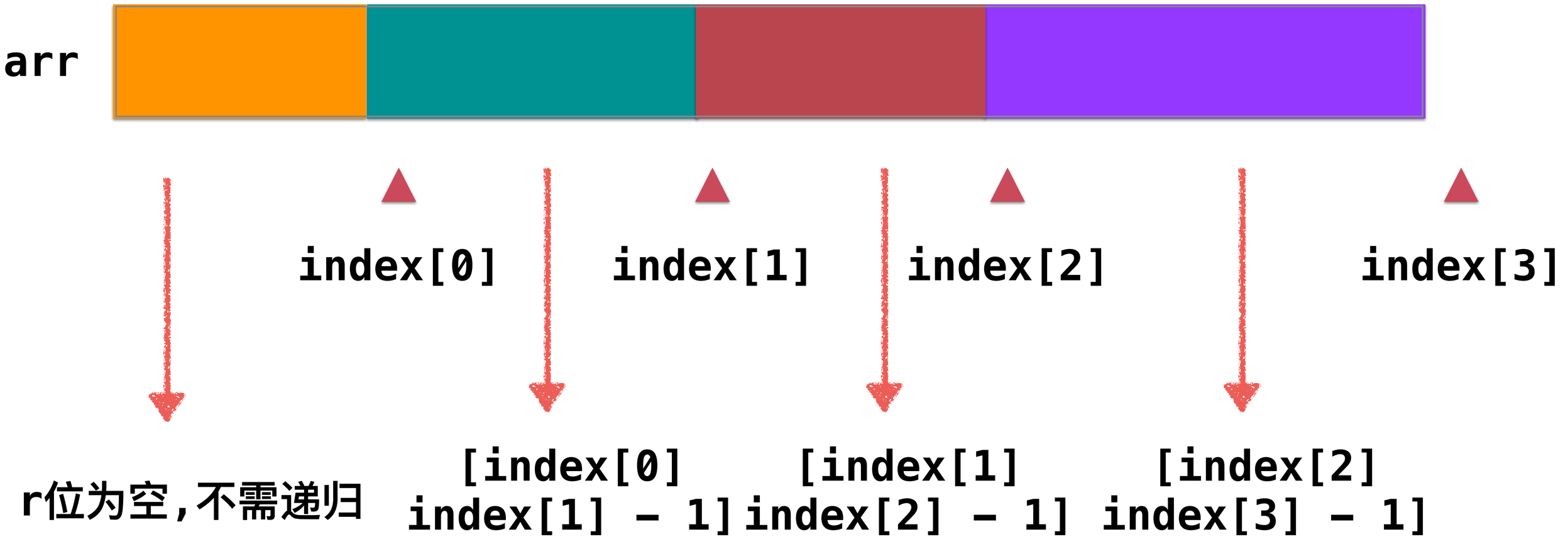
实现 MSD 基数排序



实现 MSD 基数排序



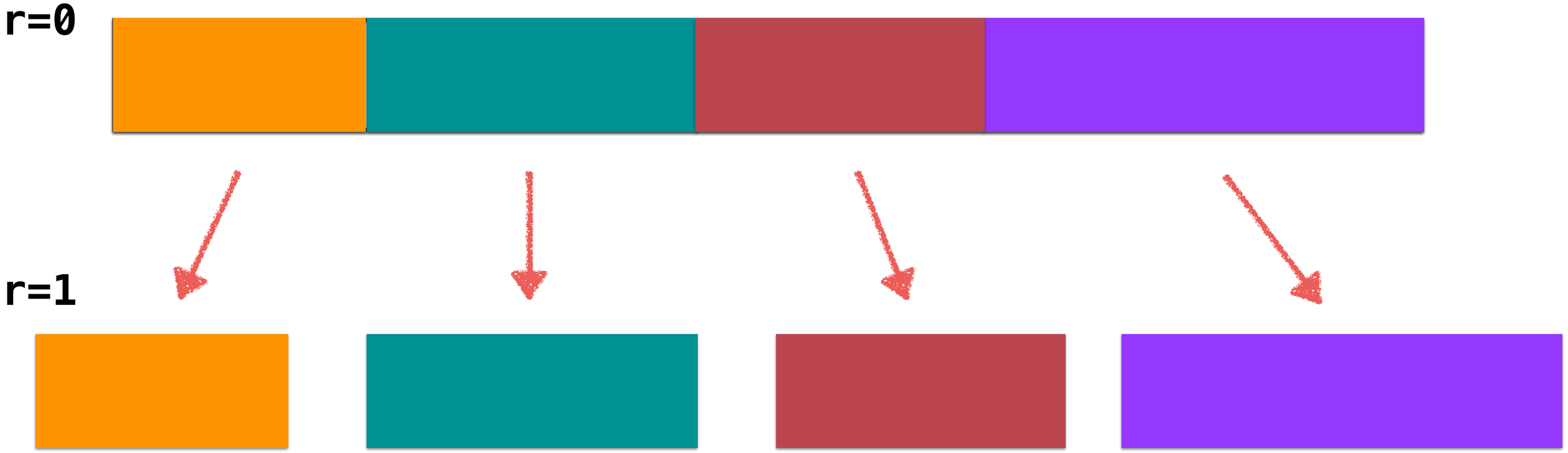
实现 MSD 基数排序


arr
index[0]
index[1]
index[2]
index[3]
r位为空,不需递归
[index[0] index[1] - 1]
[index[1] index[2] - 1]
[index[2] index[3] - 1]

实践：完成 MSD 基数排序

MSD 基数排序性能分析

MSD 基数排序性能分析



O(w*n)　　w 是最长的字符串的长度　　大量的递归提前结束

桶排序

桶排序

基数排序　　LSD　　123　　123　　效率差

45　　045

6　　006

MSD　　123　　MSD 不适用于数字

45

6

桶排序

MSD 不适用于数字

在 arr[left, right] 的范围，
根据 r 位字符，分成 256 类

让 MSD 的思想适用于数字

数字没有 r 位字符，
直接根据大小关系分成 B 份?

桶排序　　MSD：B = R = 256 + 1

桶排序

数字没有 r 位字符，
直接根据大小关系分成 B 份?

桶排序

6　　12　　32　　29　　13　　9　　30　　27　　B = 5

在 5 个桶中，元素取值范围：32 - 6 + 1 = 27

每个桶能盛放：ceil(27 / 5) = 6 个元素可能

桶排序

6　　12　　32　　29　　13　　9　　30　　27　　B = 5

每个桶能盛放：ceil(27 / 5) = 6 个元素可能

桶 0 [6, 11]：

桶 1 [12, 17]：

桶 2 [18, 23]：

桶 3 [24, 29]：

桶 4 [30, 35]：

桶排序

6　　12　　32　　29　　13　　9　　30　　27　　B = 5

每个桶能盛放：ceil(27 / 5) = 6 个元素可能

桶 0 [6, 11]：　　6　　(6 - 6) / 6 = 0

桶 1 [12, 17]：

桶 2 [18, 23]：

桶 3 [24, 29]：

桶 4 [30, 35]：

桶排序

6　　12　　32　　29　　13　　9　　30　　27　　B = 5

每个桶能盛放：ceil(27 / 5) = 6 个元素可能

桶 0 [6, 11]：　6　　(12 - 6) / 6 = 1

桶 1 [12, 17]：　12

桶 2 [18, 23]：

桶 3 [24, 29]：

桶 4 [30, 35]：

桶排序

6　　12　　32　　29　　13　　9　　30　　27　　B = 5

每个桶能盛放：ceil(27 / 5) = 6 个元素可能

桶 0 [6, 11]：　6　　(32 - 6) / 6 = 4

桶 1 [12, 17]：　12

桶 2 [18, 23]：

桶 3 [24, 29]：

桶 4 [30, 35]：　32

桶排序

6　　12　　32　　29　　13　　9　　30　　27　　B = 5

每个桶能盛放：ceil(27 / 5) = 6 个元素可能

桶 0 [6, 11]：　6　　(29 - 6) / 6 = 3

桶 1 [12, 17]：　12

桶 2 [18, 23]：

桶 3 [24, 29]：　29

桶 4 [30, 35]：　32

桶排序

6　　12　　32　　29　　13　　9　　30　　27　　B = 5

每个桶能盛放：ceil(27 / 5) = 6 个元素可能

桶 0 [6, 11]：　6　　9

桶 1 [12, 17]：　12　　13

桶 2 [18, 23]：

桶 3 [24, 29]：　29　　27

桶 4 [30, 35]：　32　　30

对每个桶，递归下去

和 MSD 的思想一样

每一轮本质也在做计数排序

不需要考虑字符空的问题

实现桶排序

实践：实现桶排序

更简单的桶排序

更简单的桶排序

MSD  桶排序

不需要递归的桶排序

回忆之前的课程，快排或者归并排序，当
数据量小的时候，转成插入排序反而更快

把数据分成若干桶，直接对每一个桶中的元素进行排序

更简单的桶排序

MSD → 桶排序

把数据分成若干桶，直接对每一个桶中的元素进行排序

arr

桶 0:

桶 1:

桶 2:

限定桶内的元素数量

通常用链表

更简单的桶排序

MSD  桶排序

把数据分成若干桶，直接对每一个桶中的元素进行排序

arr 

桶 0: 

桶 1: 

桶 2: 

限定桶内的元素数量

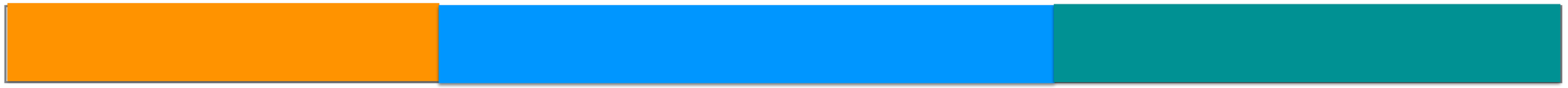
通常用链表

更简单的桶排序

MSD → 桶排序

把数据分成若干桶，直接对每一个桶中的元素进行排序

arr



桶 0:

桶 1:

桶 2:

限定桶内的元素数量

通常用链表

桶排序的性能分析

桶排序的性能分析

桶排序的性能分析

每个桶有 c 个元素

一共 n / c 个桶

每个桶的排序：c^2

整体：O(n / c * c^2) = O(cn) = O(n)

更多关于桶排序

Leetcode 164

处理浮点数?

总结

计数排序

基数排序 LSD

基数排序 MSD

桶排序

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