算法与数据结构体系课程

liuyubobobo

计数排序和基数排序

非比较排序

非比较排序

不是元素之间不能比较，而是不借助 compareTo

主要应用于字符串排序

计数排序

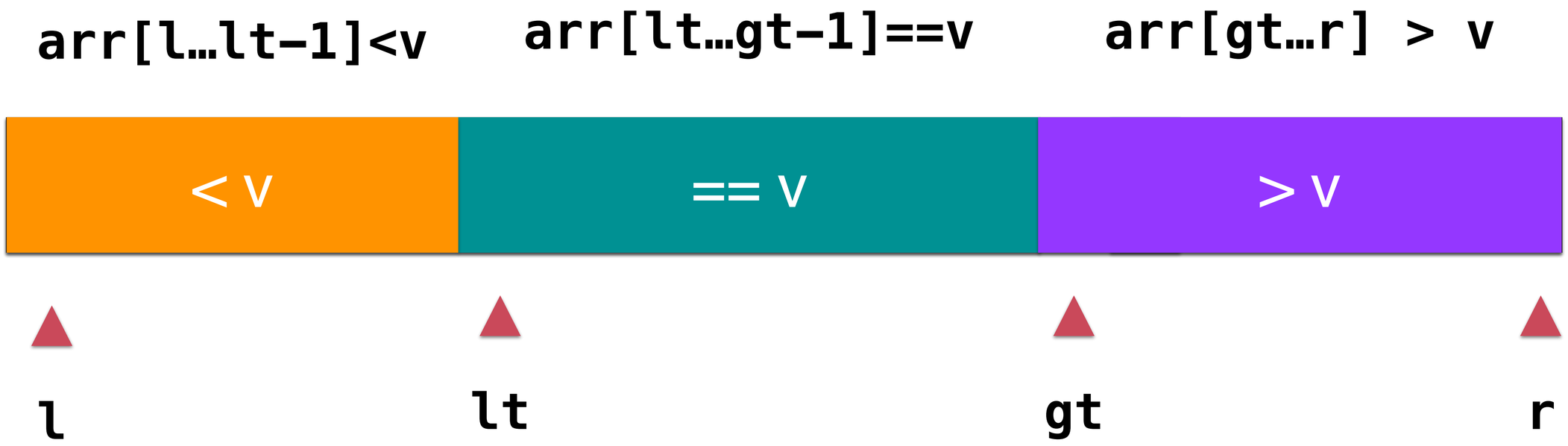
计数排序

Leetcode 75 : 颜色分类

https://leetcode-cn.com/problems/sort-colors/

给定一个数组，只包含 0，1，2，对数组排序

三路快排



计数排序

Leetcode 75：颜色分类　　给定一个数组，只包含 0，1，2，对数组排序

遍历数组，看数组中一共有多少个 0，多少个 1，多少个 2

假设有 cnt0 个 0；cnt1 个 1；cnt2 个 2

计数排序

Leetcode 75：颜色分类　　给定一个数组，只包含 0，1，2，对数组排序

假设有 cnt0 个 0；cnt1 个 1；cnt2 个 2

计数排序

l　　r1　r2

[l, r1]　r1 - l + 1 个元素

[l, r2)　r2 - l 个元素

实践：计数排序的思路实现 LC 75

更一般的计数排序

更一般的计数排序

```
int[] cnt = new int[3];
for(int num: nums)
    cnt[num] ++;
```

```
for(int i = 0; i < cnt[0]; i ++)
    nums[i] = 0;

for(int i = cnt[0]; i < cnt[0] + cnt[1]; i ++)
    nums[i] = 1;

for(int i = cnt[0] + cnt[1]; i < cnt[0] + cnt[1] + cnt[2]; i ++)
    nums[i] = 2;
```

如果数字的可能范围是 [0, R)

int[] cnt = new int[R]

计数排序需要的空间：O(R)

只适用于小数据范围

如果数字的可能范围是 [L, R]

int[] cnt = new int[R - L + 1]

cnt[num - L] ++

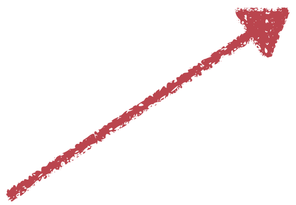
更一般的计数排序

```
int[] cnt = new int[3];
for(int num: nums)
    cnt[num] ++;
```

```
for(int i = 0; i < cnt[0]; i ++)
    nums[i] = 0;

for(int i = cnt[0]; i < cnt[0] + cnt[1]; i ++)
    nums[i] = 1;

for(int i = cnt[0] + cnt[1]; i < cnt[0] + cnt[1] + cnt[2]; i ++)
    nums[i] = 2;
```



如果取值范围是 [0, 101),

不可能写 101 个 for 循环

cnt -> index

0 : [0, cnt[0])

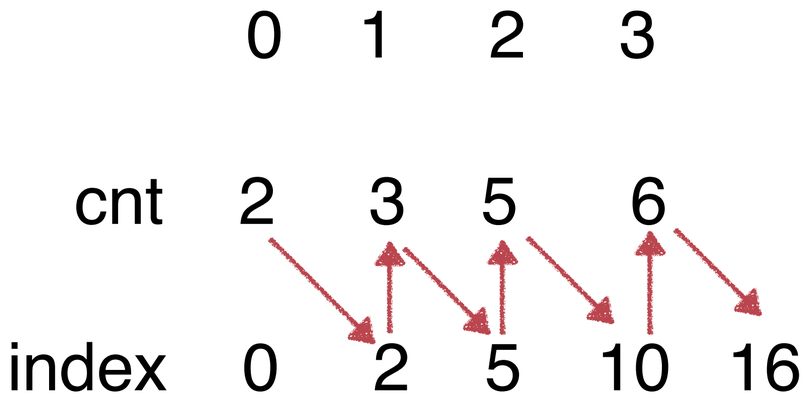
1 : [cnt[0], cnt[0] + cnt[1])

2 : [cnt[0] + cnt[1], cnt[0] + cnt[1] + cnt[2])

3 : [cnt[0] + cnt[1] + cnt[2],
cnt[0] + cnt[1] + cnt[2] + cnt[3])

更一般的计数排序

cnt -> index



0 : [0, cnt[0])

1 : [cnt[0], cnt[0] + cnt[1])

2 : [cnt[0] + cnt[1], cnt[0] + cnt[1] + cnt[2])

3 : [cnt[0] + cnt[1] + cnt[2],

cnt[0] + cnt[1] + cnt[2] + cnt[3])

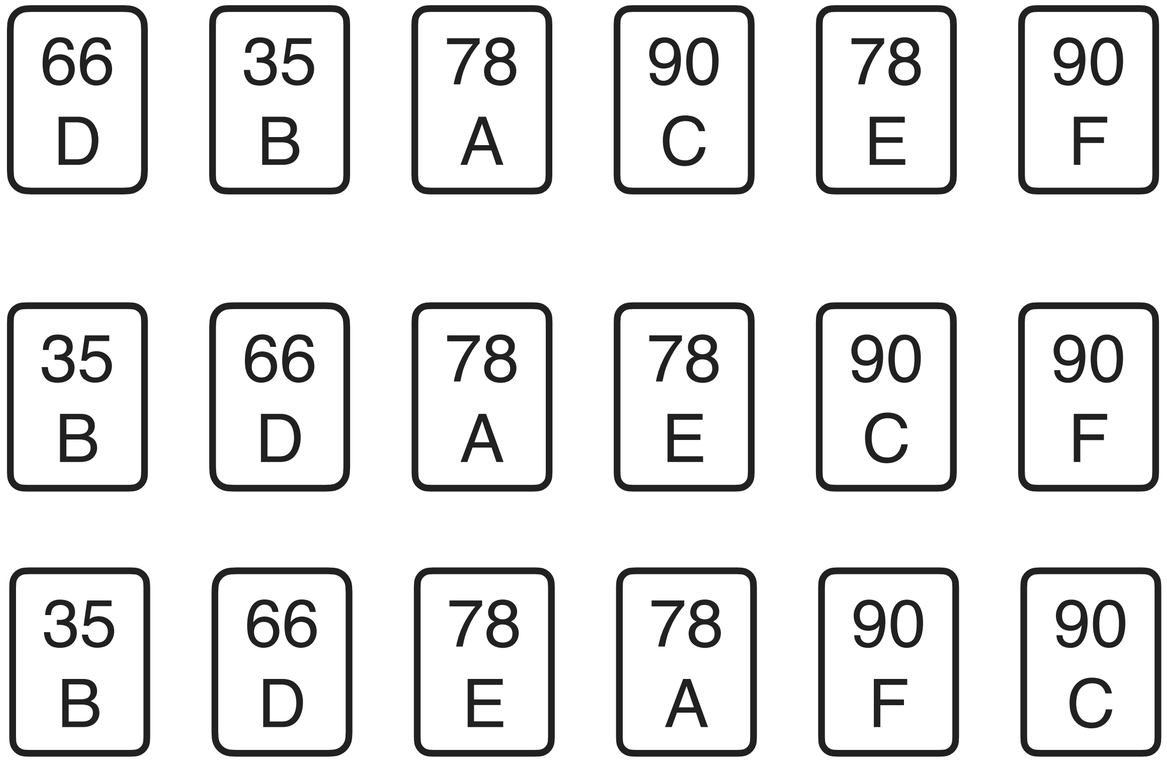
[index[i], index[i + 1]) 区间的值为 i

实践：更一般的计数排序

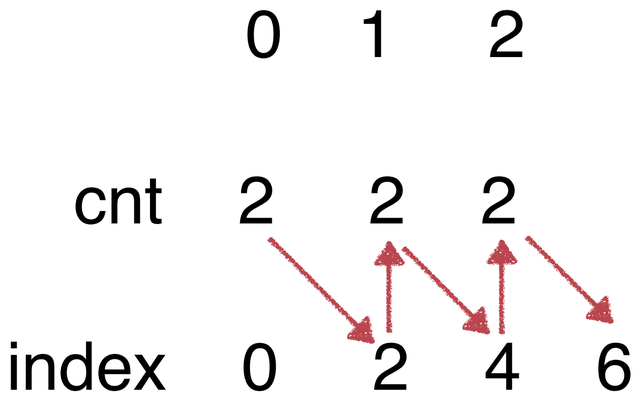
计数排序的重要性质：稳定性

计数排序的稳定性

排序的稳定性：排序前相等的两个元素，排序后相对位置不变

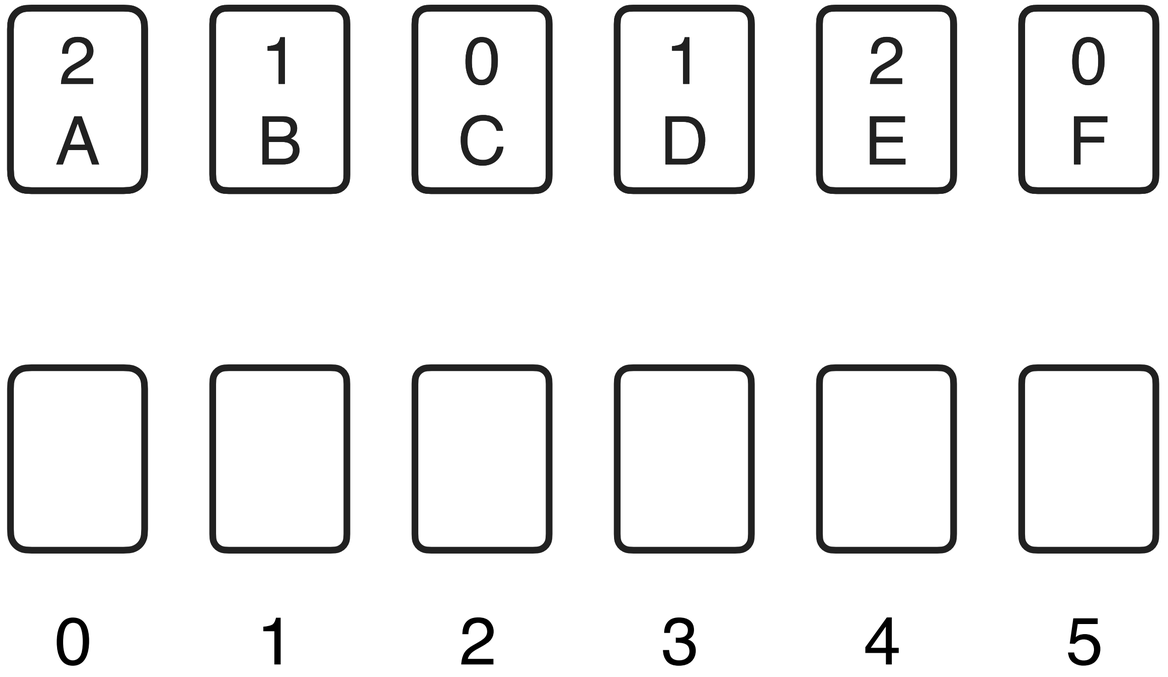


计数排序的稳定性



index[i] 是 i 的起始索引

[index[i], index[i + 1]) 区间的值为 i

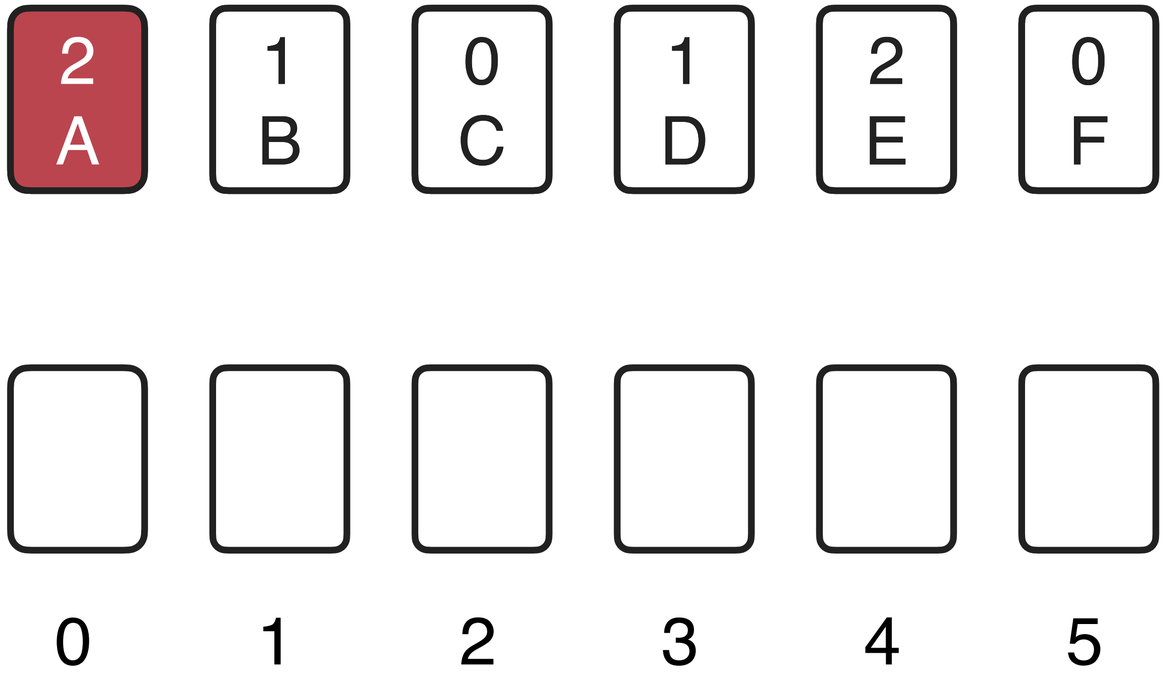


计数排序的稳定性

	0	1	2
cnt	2	2	2

	0	2	4	6
index				

index[i] 是 i 的起始索引

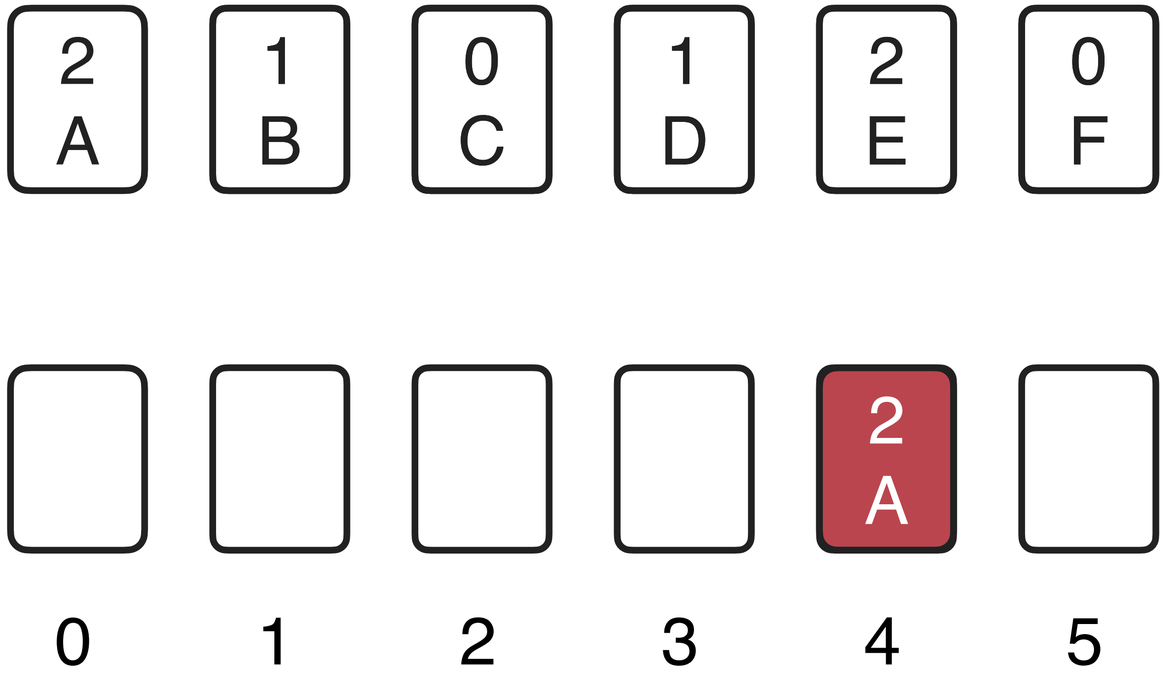


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	0	2	4	6

index[i] 是 i 的起始索引

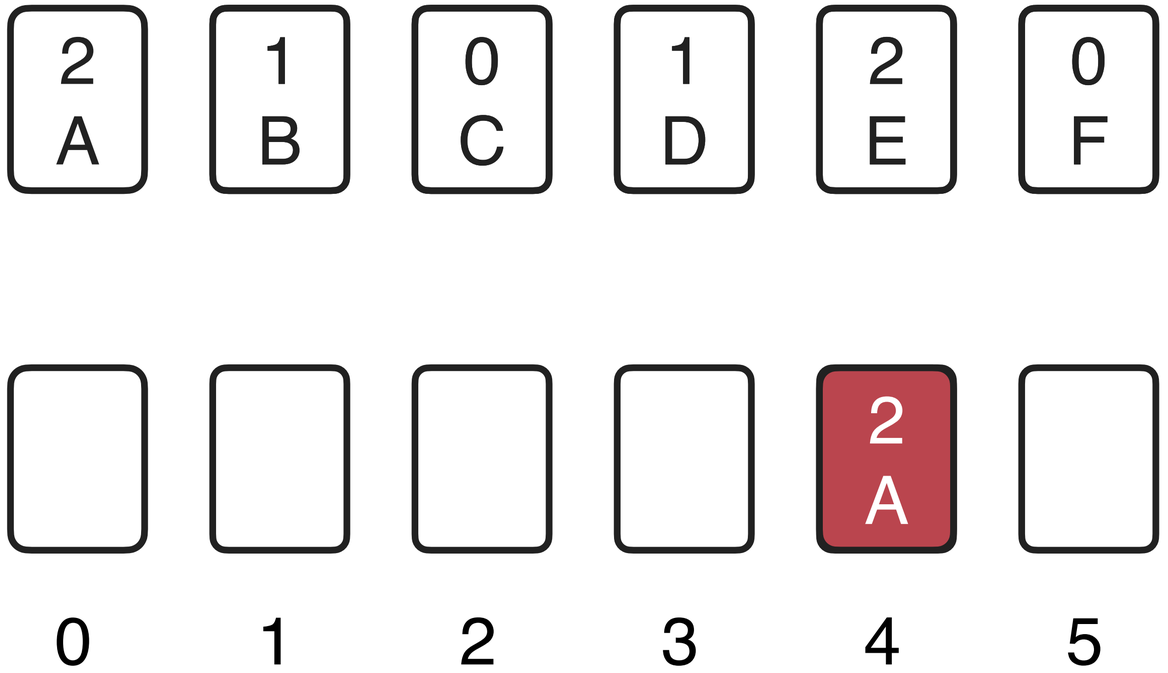


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	0	2	4	6

index[i] 是 i 的起始索引

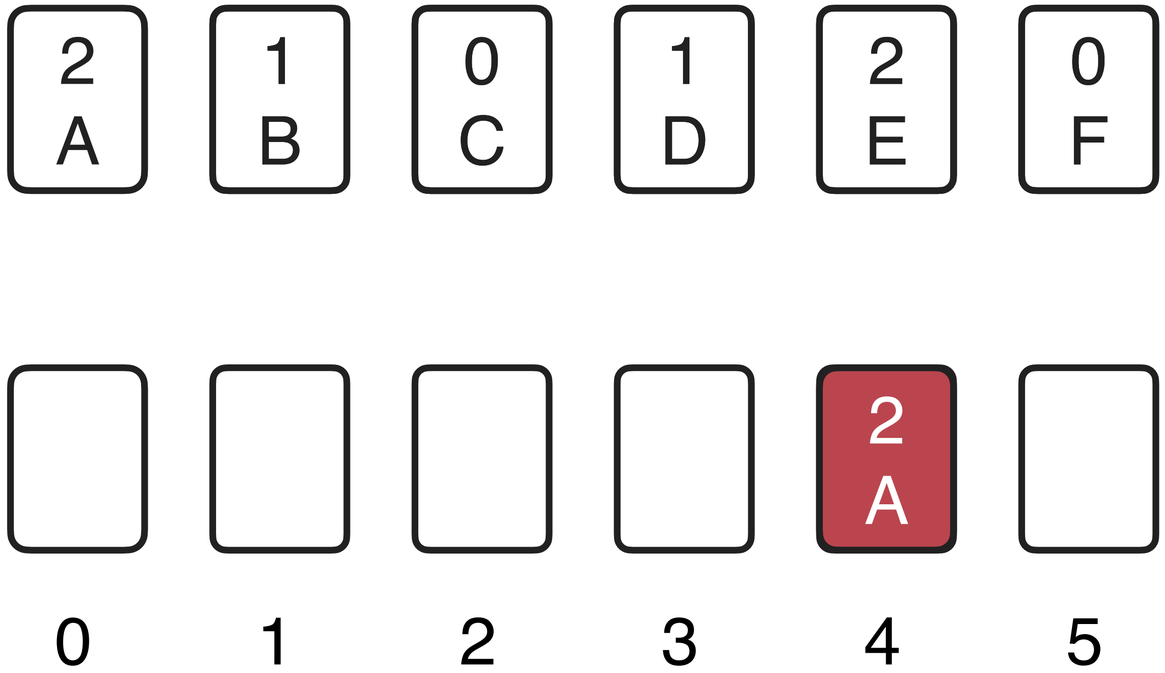


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	0	2	5	6

index[i] 是 i 的起始索引

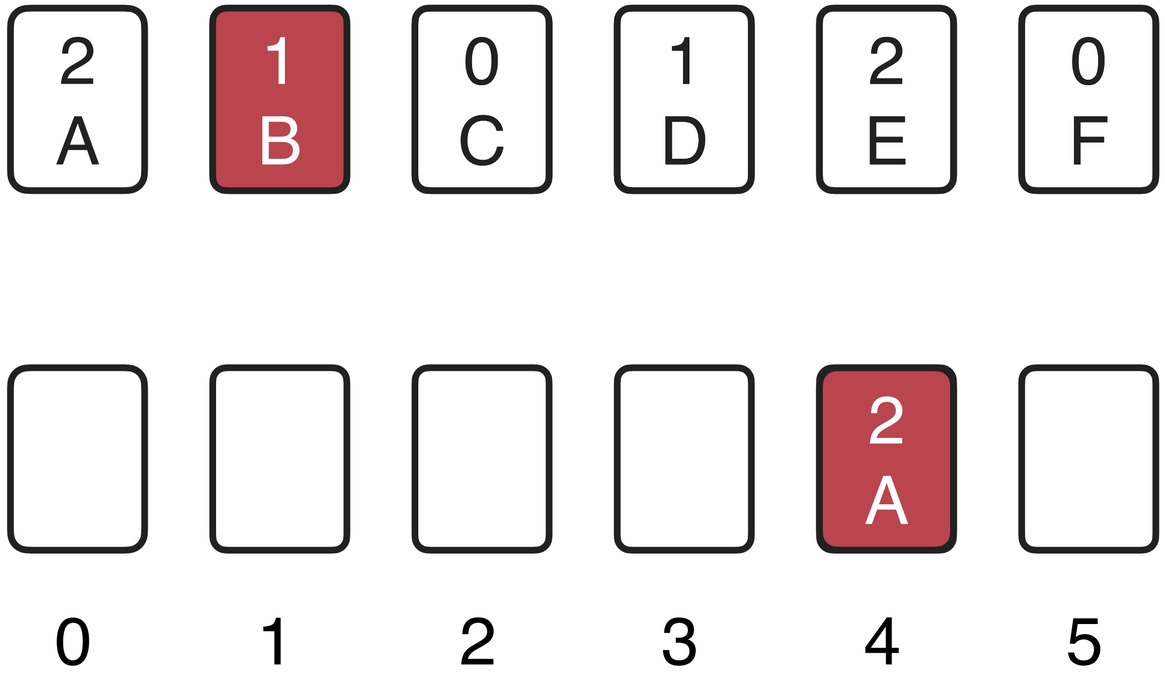


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	0	2	5	6

index[i] 是 i 的起始索引

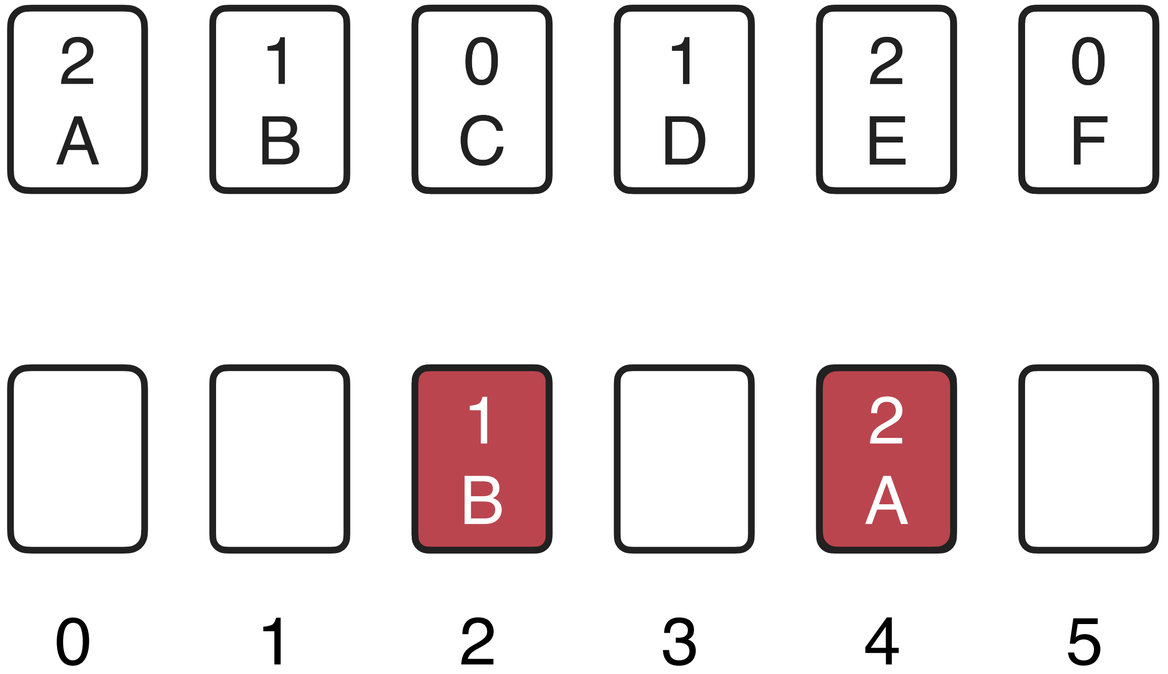


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	0	2	5	6

index[i] 是 i 的起始索引

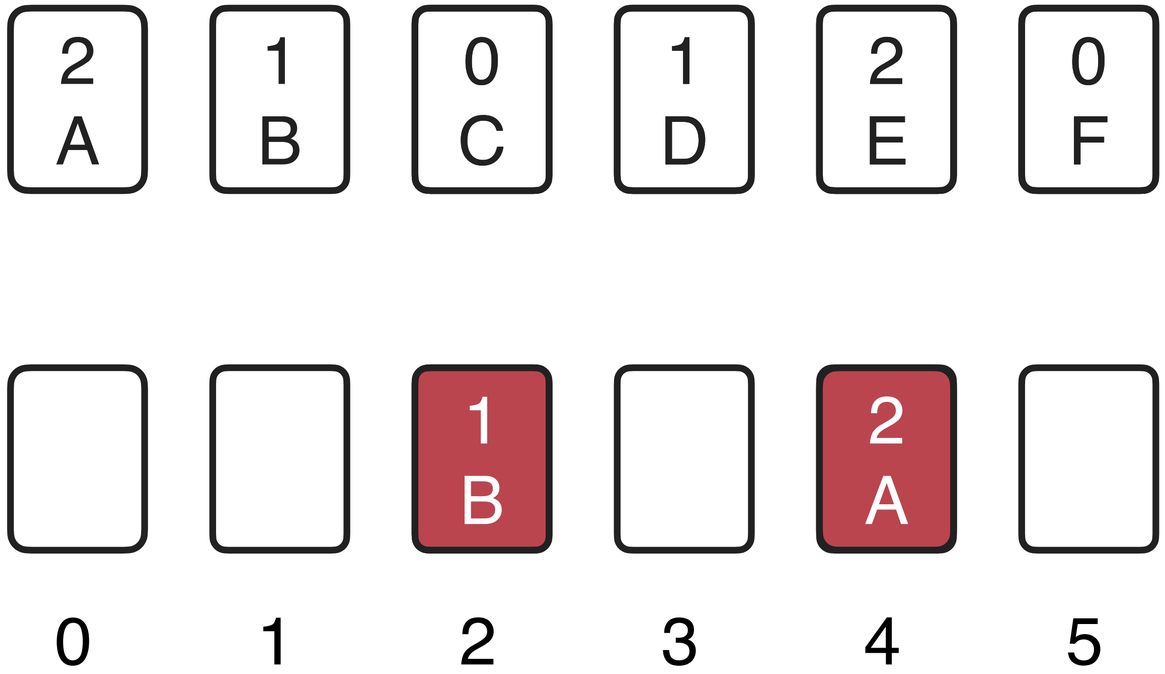


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	0	3	5	6

index[i] 是 i 的起始索引

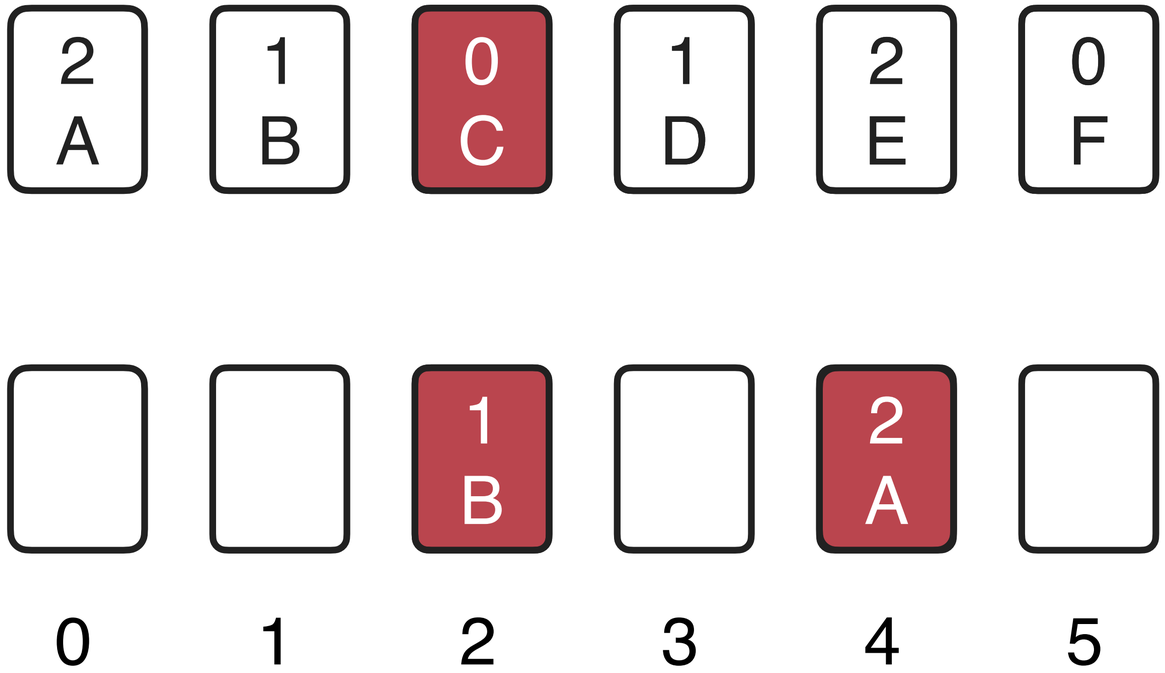


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	0	3	5	6

index[i] 是 i 的起始索引

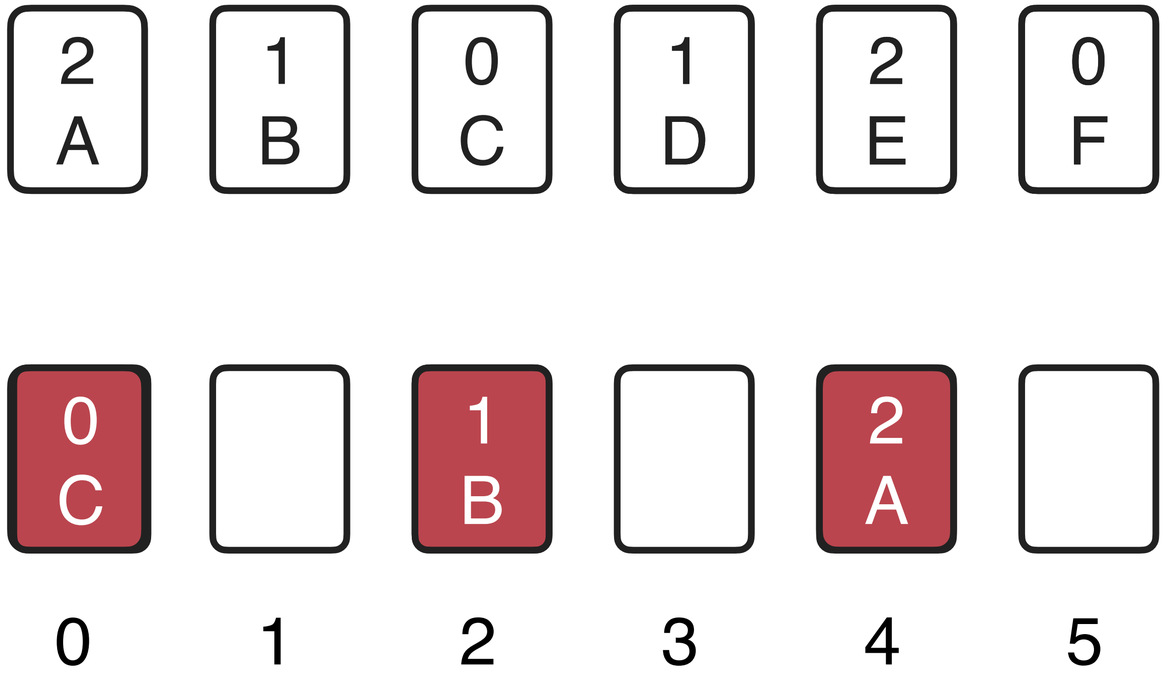


计数排序的稳定性

	0	1	2
cnt	2	2	2

	0	3	5	6
index				

index[i] 是 i 的起始索引

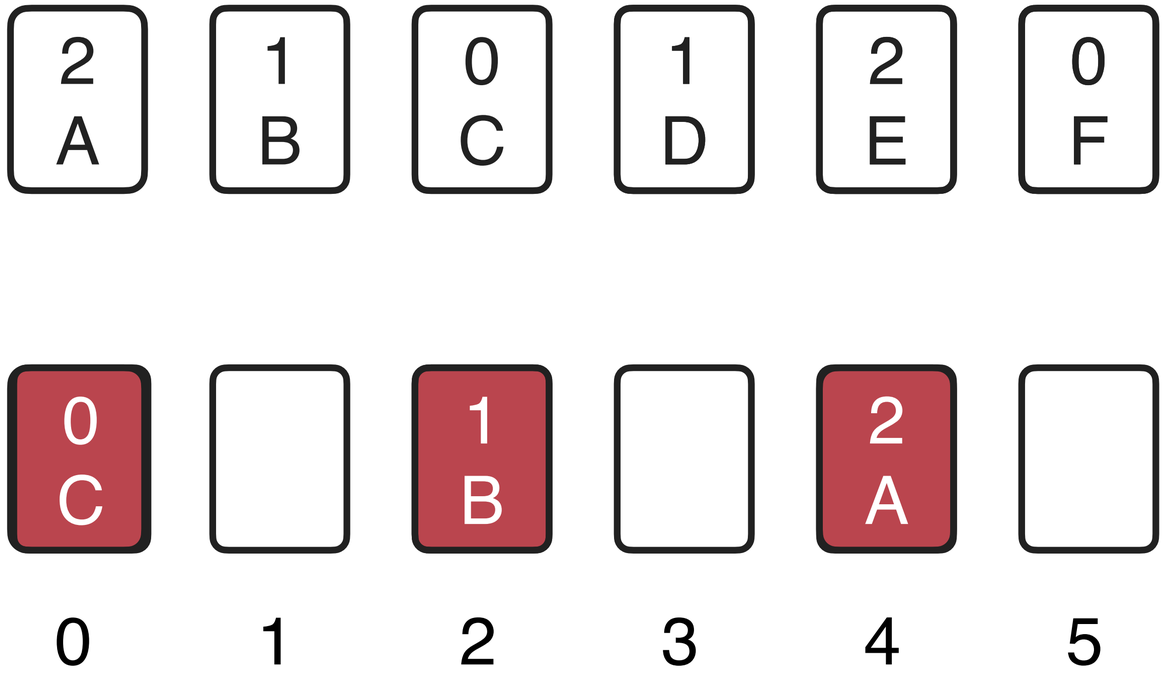


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	1	3	5	6

index[i] 是 i 的起始索引

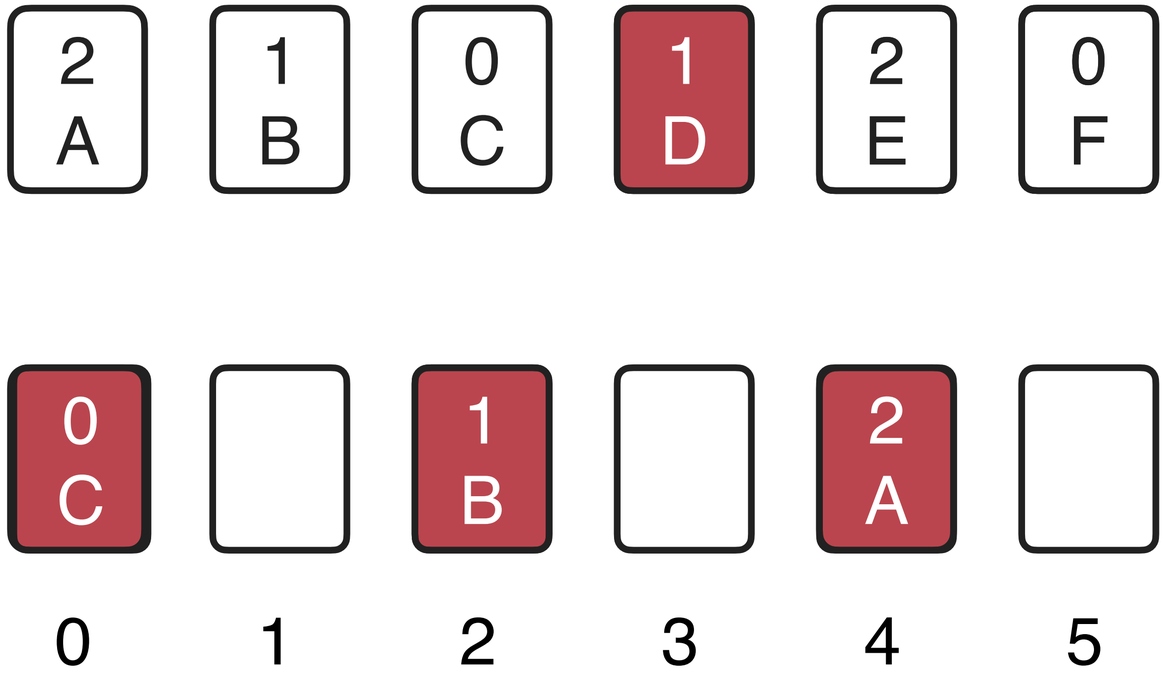


计数排序的稳定性

	0	1	2
cnt	2	2	2

	1	3	5	6
index				

index[i] 是 i 的起始索引

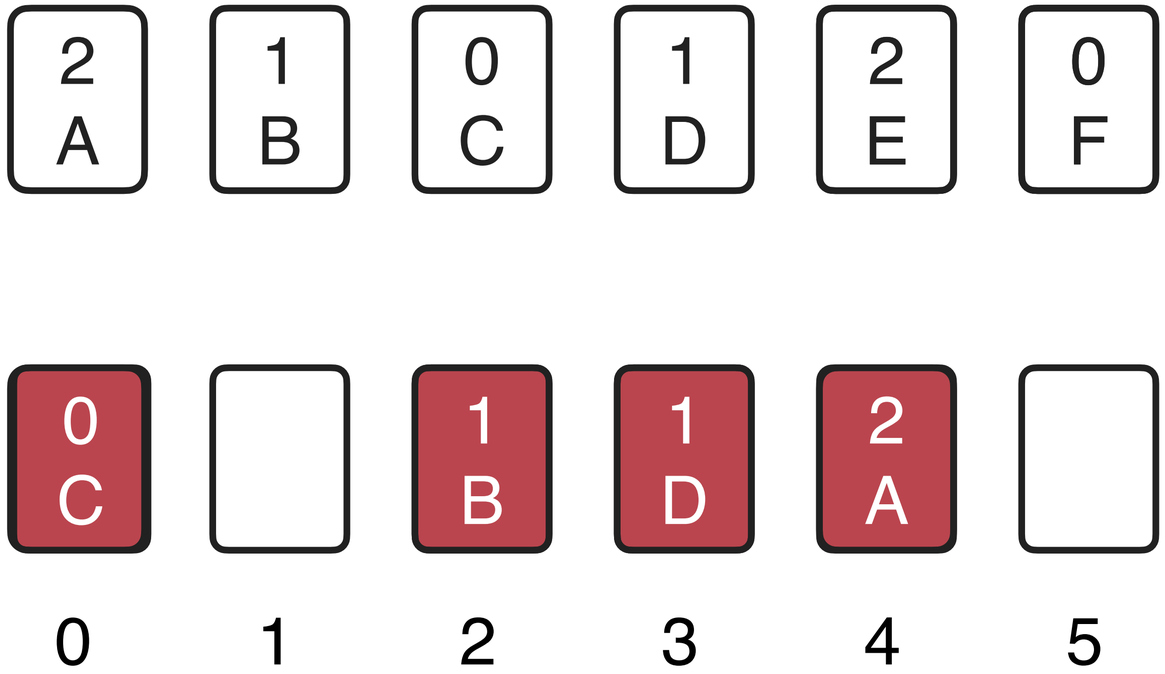


计数排序的稳定性

	0	1	2
cnt	2	2	2

	1	3	5	6
index				

index[i] 是 i 的起始索引

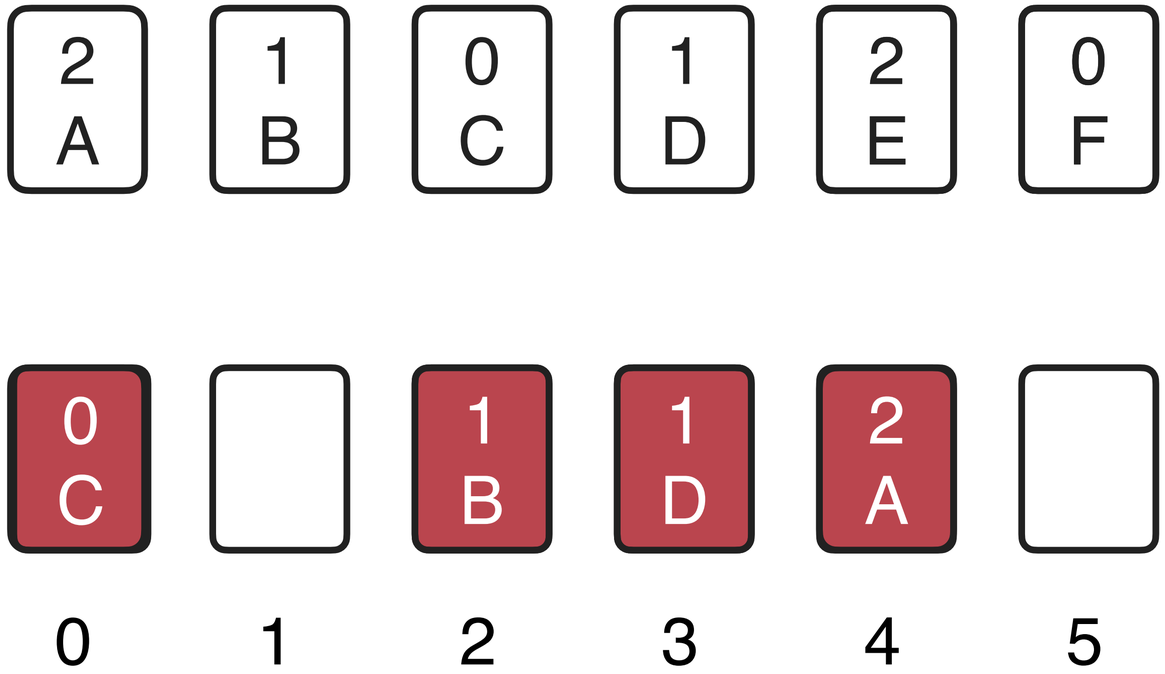


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	1	4	5	6

index[i] 是 i 的起始索引

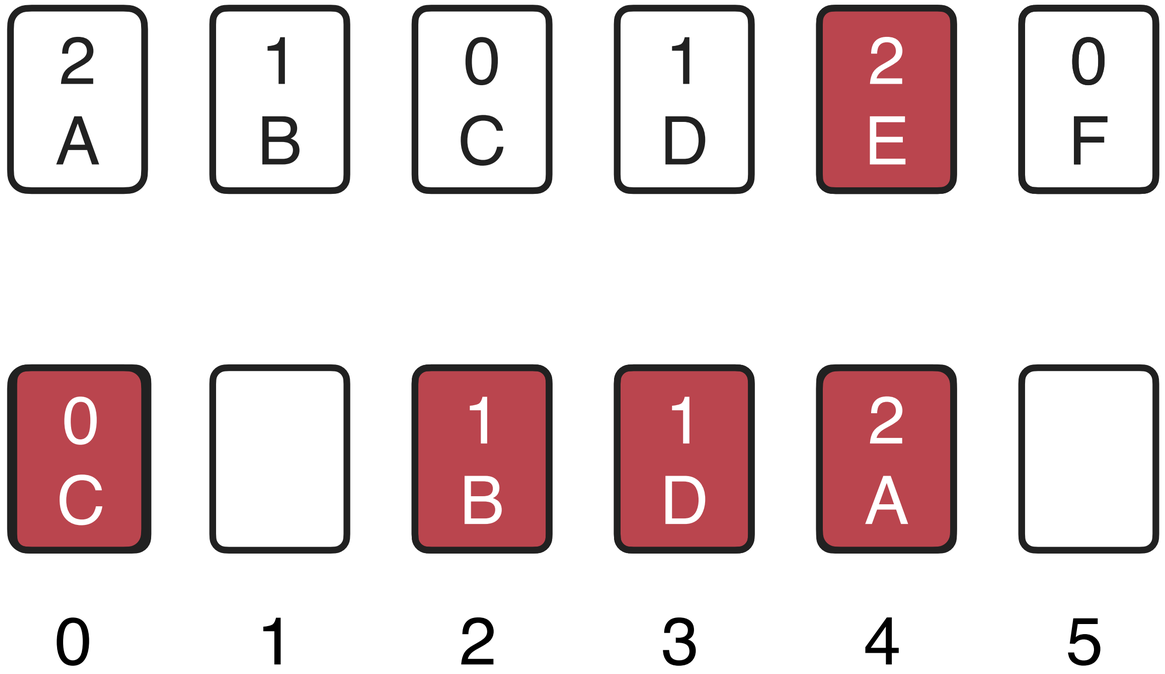


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	1	4	5	6

index[i] 是 i 的起始索引

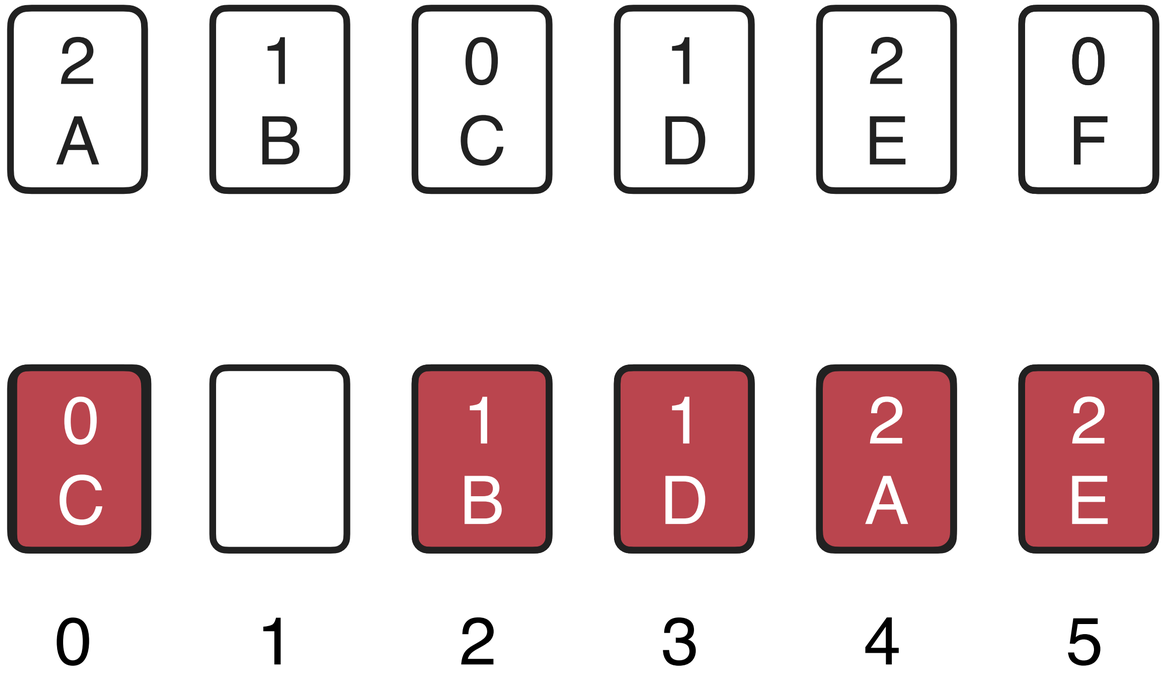


计数排序的稳定性

	0	1	2
cnt	2	2	2

	1	4	5	6
index				

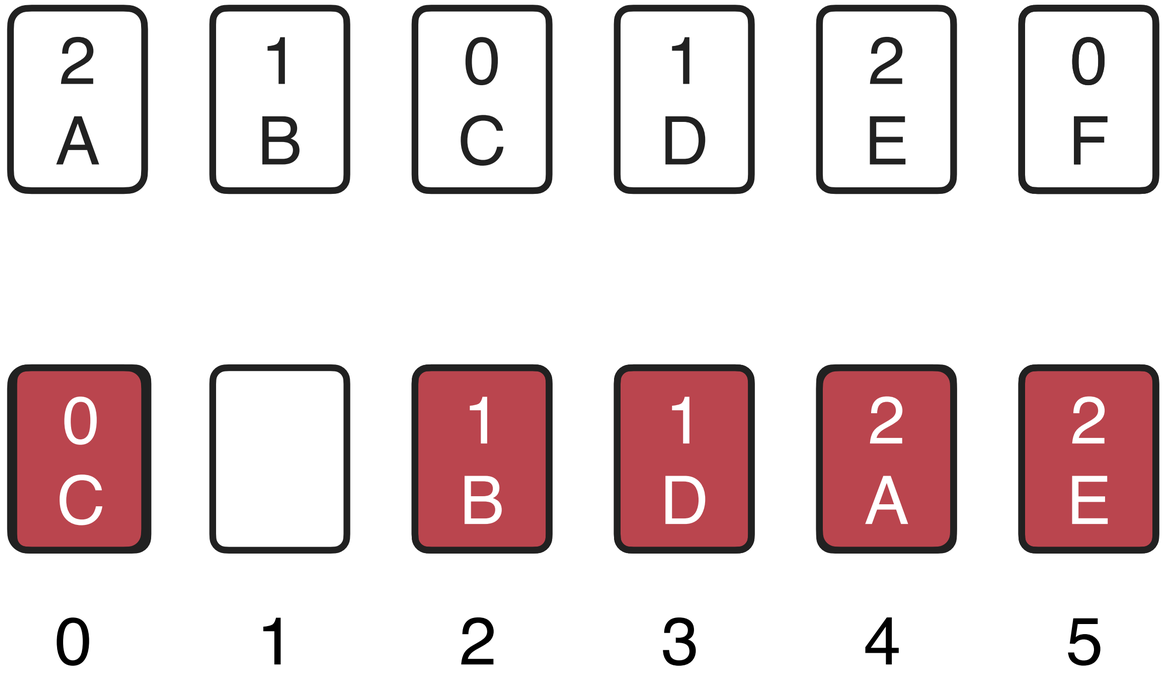
index[i] 是 i 的起始索引



计数排序的稳定性

	0	1	2
cnt	2	2	2
index	1	4	6	6

index[i] 是 i 的起始索引

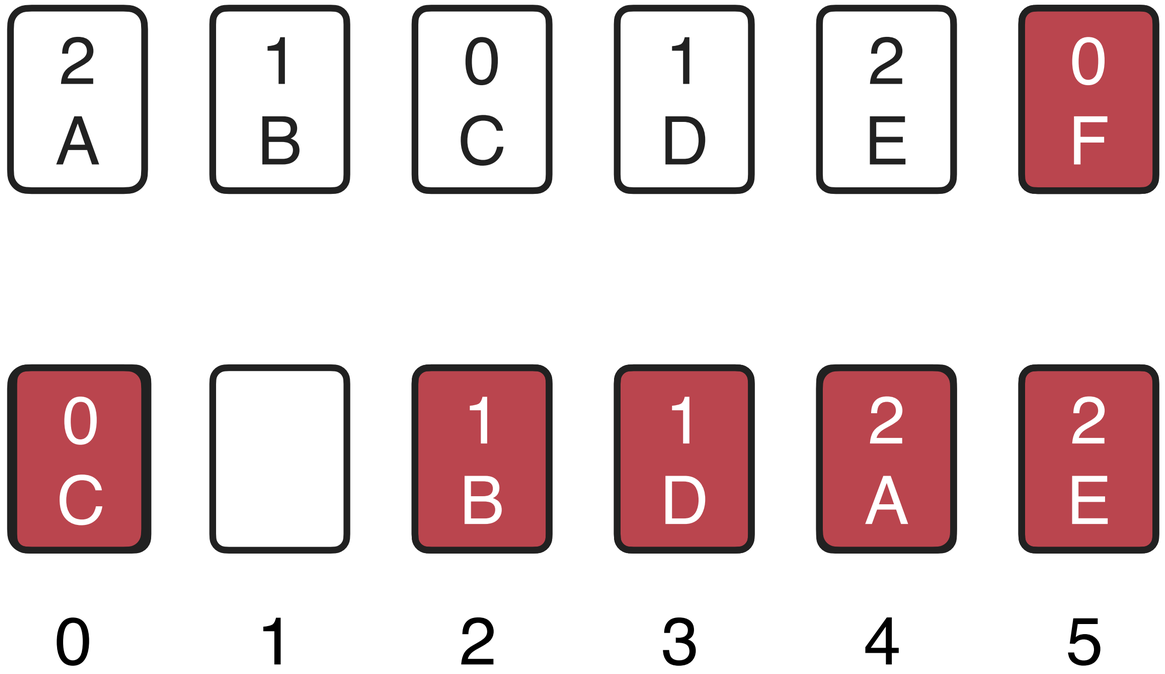


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	1	4	6	6

index[i] 是 i 的起始索引

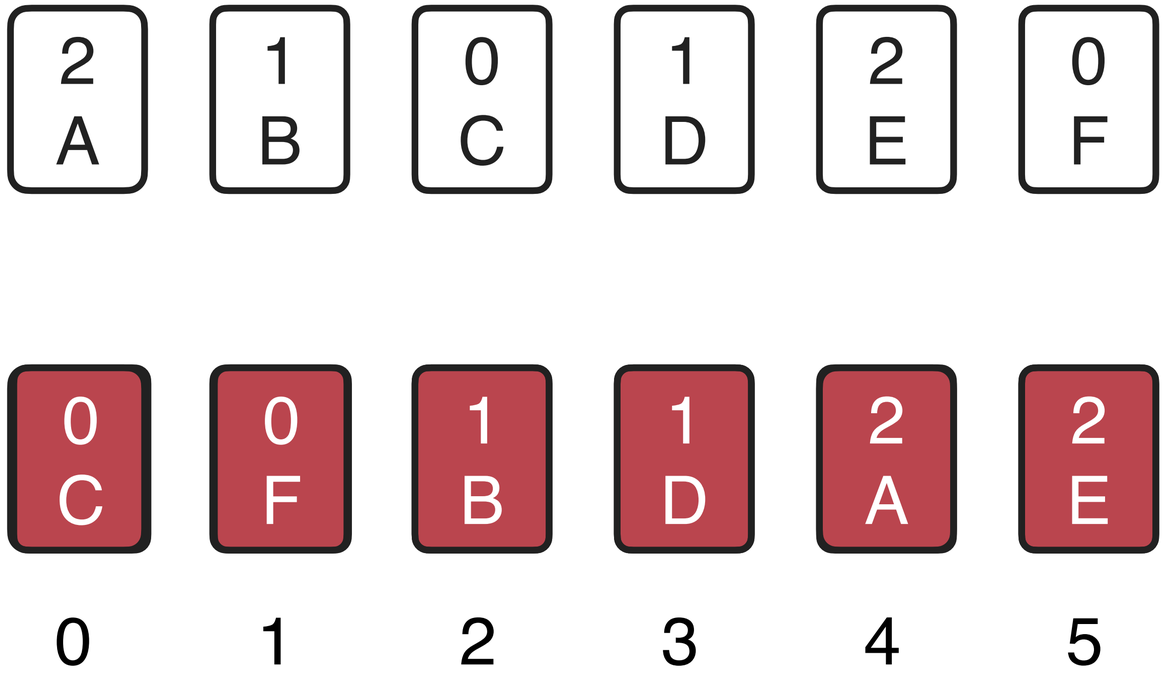


计数排序的稳定性

	0	1	2
cnt	2	2	2

	1	2	3	4
index	1	4	6	6

index[i] 是 i 的起始索引

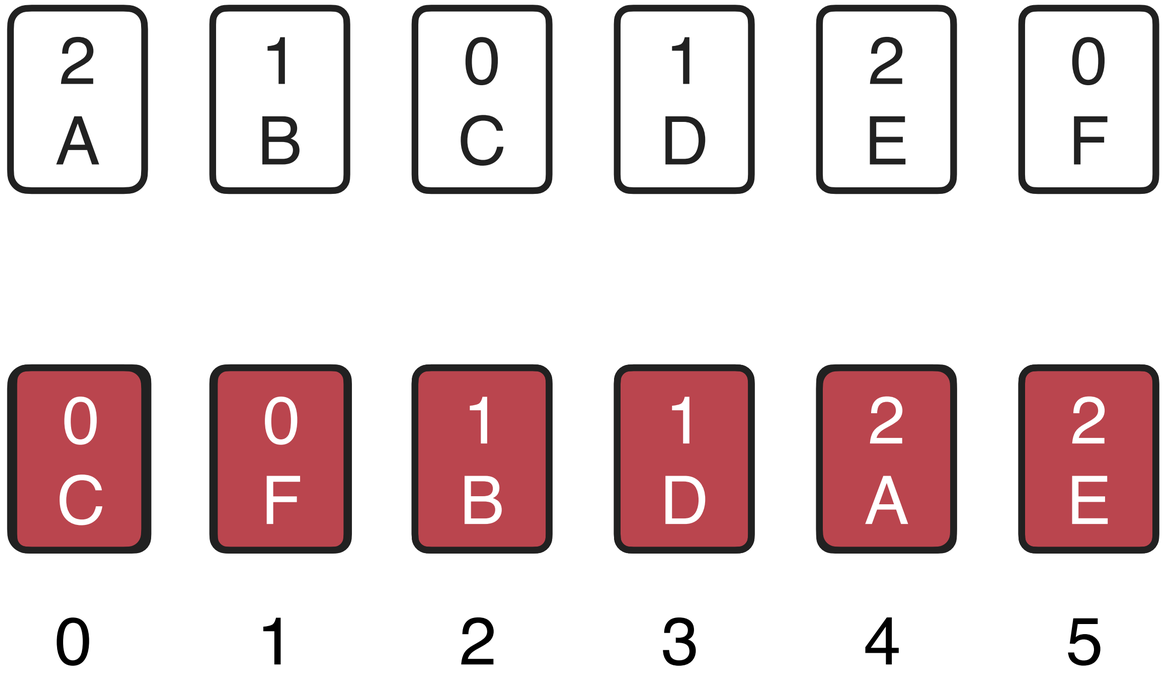


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	2	4	6	6

index[i] 是 i 的起始索引

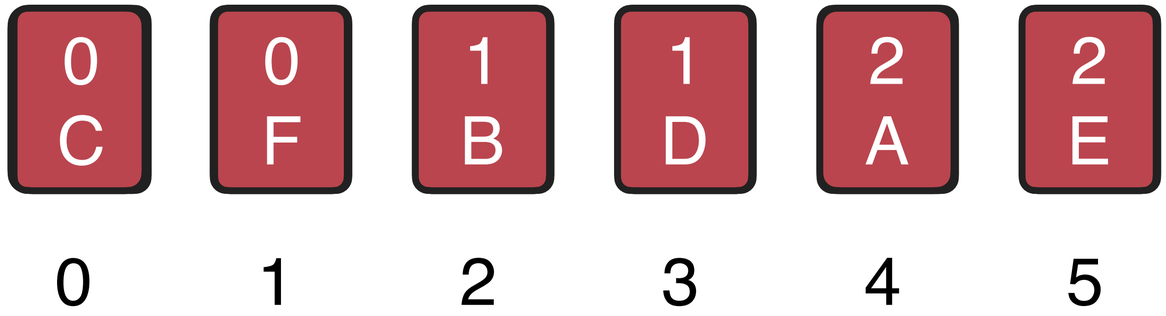


计数排序的稳定性

	0	1	2
cnt	2	2	2

index	2	4	6	6

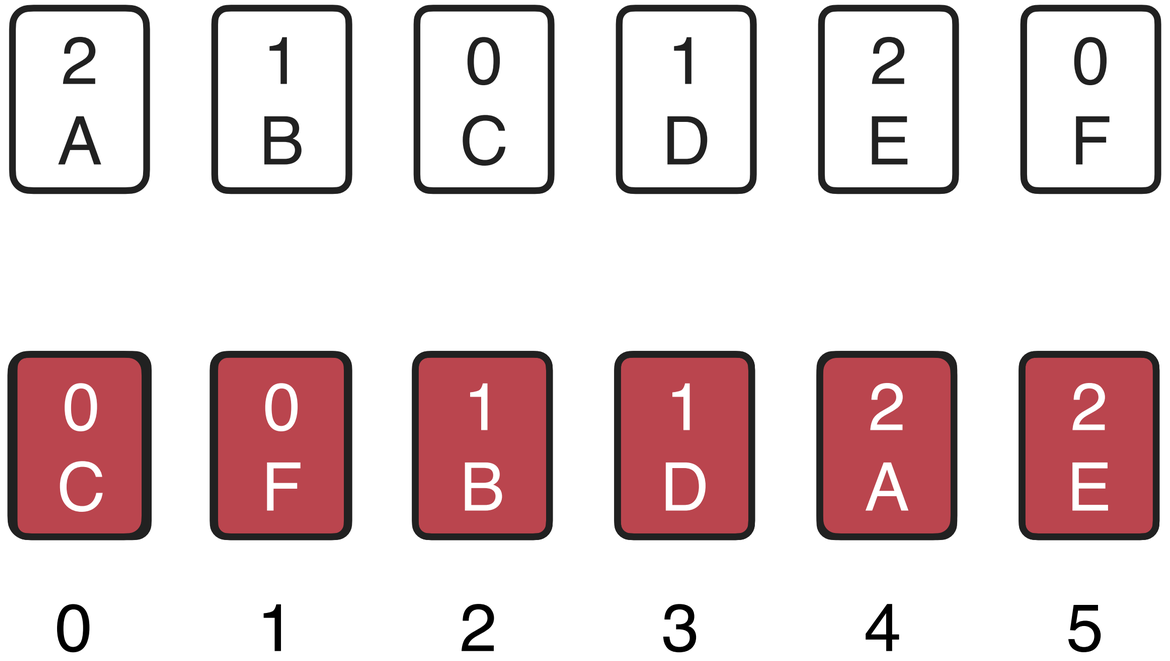
index[i] 是 i 的起始索引



计数排序的稳定性

	0	1	2	
cnt	2	2	2	
index	2	4	6	6

index[i] 是 i 的起始索引



计数排序是稳定的!

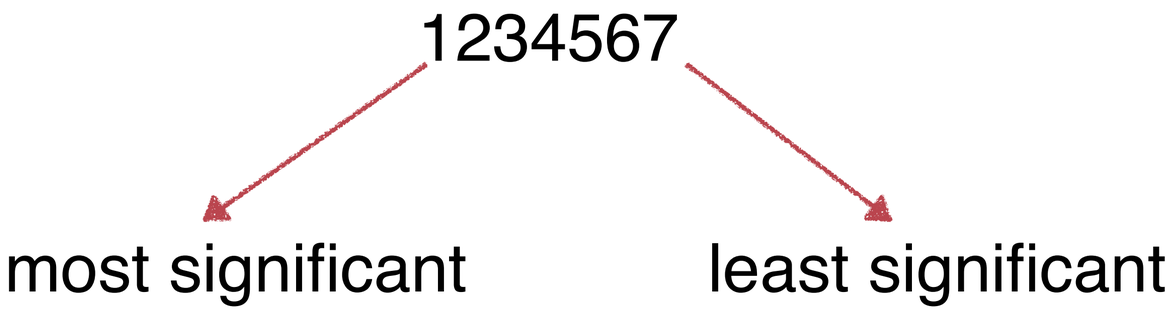
实践：编写计数排序算法

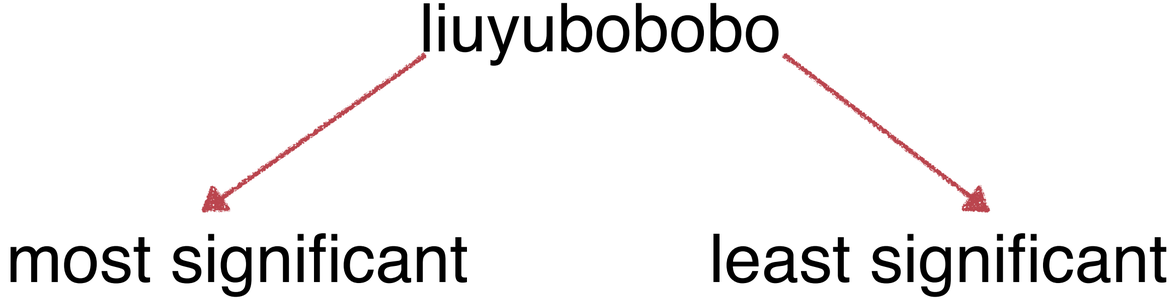
LSD 字符串排序算法

LSD 字符串排序算法

LSD　　Least Significant Digit

从最末位向前排序　　只适用于等长字符串





LSD 字符串排序算法

LSD　　Least Significant Digit

只适用于等长字符串

车牌号；电话号码；…

从最末位向前排序　　每一轮使用计数排序

LSD 字符串排序算法

BCA

CAB

ACB

BAC

ABC

CBA

LSD 字符串排序算法

BCA

CAB

ACB

BAC

ABC

CBA

LSD 字符串排序算法

BCA	BCA
CAB	CBA
ACB	CAB
BAC	ACB
ABC	BAC
CBA	ABC

LSD 字符串排序算法

BCA	BCA
CAB	CBA
ACB	CAB
BAC	ACB
ABC	BAC
CBA	ABC

LSD 字符串排序算法

BCA	BCA
CAB	CBA
ACB	CAB
BAC	ACB
ABC	BAC
CBA	ABC

LSD 字符串排序算法

BCA	BCA	CAB
CAB	CBA	BAC
ACB	CAB	CBA
BAC	ACB	ABC
ABC	BAC	BCA
CBA	ABC	ACB

LSD 字符串排序算法

BCA	BCA	CAB
CAB	CBA	BAC
ACB	CAB	CBA
BAC	ACB	ABC
ABC	BAC	BCA
CBA	ABC	ACB

LSD 字符串排序算法

BCA	BCA	CAB
CAB	CBA	BAC
ACB	CAB	CBA
BAC	ACB	ABC
ABC	BAC	BCA
CBA	ABC	ACB

LSD 字符串排序算法

BCA	BCA	CAB	ABC
CAB	CBA	BAC	ACB
ACB	CAB	CBA	BAC
BAC	ACB	ABC	BCA
ABC	BAC	BCA	CAB
CBA	ABC	ACB	CBA

计数排序的稳定性是 LSD 算法正确的关键！

实现 LSD 字符串排序算法

实践：实现 LSD

LSD 字符串排序算法的性能分析

实践：LSD 字符串排序算法的性能分析

更多关于 LSD 字符串排序算法

更多关于 LSD 字符串排序算法

计数排序　　稳定　非原地　　非比较排序

O(n)

LSD 字符串排序算法　　稳定　非原地　　在一般情况下，实际性能一般

更多关于 LSD 字符串排序算法

LSD 字符串排序算法

等长字符串

可以应用于数字　　123　　456　　789　　66

把数字看做是字符串

更多关于 LSD 字符串排序算法

LSD 字符串排序算法

等长字符串

可以应用于数字　　123　　456　　789　　066

把数字看做是字符串

可以应用于不等长的数字：补 0

基数排序

更多关于 LSD 字符串排序算法

LSD 字符串排序算法　　基数排序

等长字符串

可以应用于数字　　123　　456　　789　　066

把数字看做是字符串

可以应用于不等长的数字：补 0

同样的思路不可以应用于不等长的字符串

更多关于 LSD 字符串排序算法

LSD 字符串排序算法　　基数排序

等长字符串

可以应用于数字　　123　　456　　789　　066

把数字看做是字符串

可以应用于不等长的数字：补 0

同样的思路不可以应用于不等长的字符串

更多关于 LSD 字符串排序算法

LSD 字符串排序算法　　基数排序

等长字符串

可以应用于数字　　可以应用于不等长的数字：补 0

同样的思路不可以应用于不等长的字符串

补空格？　　补 0 对应的字符？　　\0

另一种基数排序：MSD

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