本节内容

计数排序

Counting Sort

考研大纲对计数排序的要求 be like

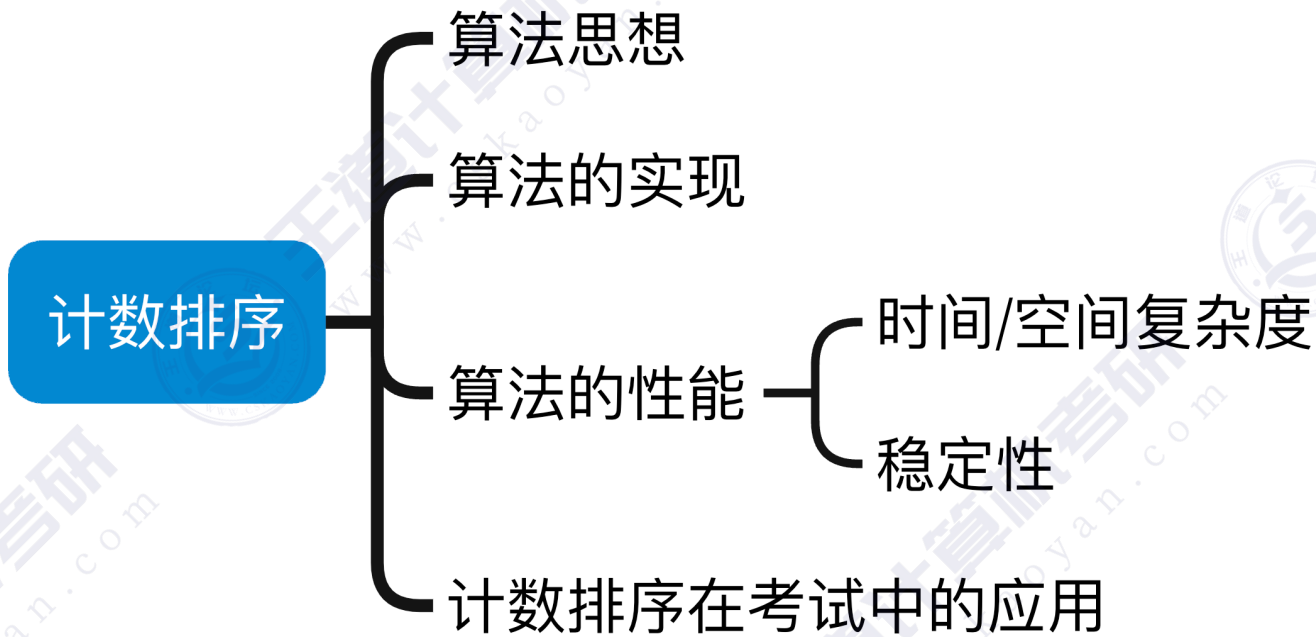
大纲明面上要求的...

考研大纲：没事哒，计数排序不用学，不要求掌握~

实际上...

考场上：计数排序这么简单的东西，想必同学们都会吧？

知识总览

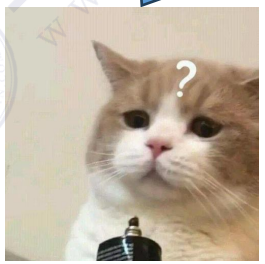


Intro：计数排序的算法思想

小学某班英语成绩：

旺财	90分
铁柱	97分
阿呆	76分
正南	83分
风间	100分
❤️卧龙	3分
狗剩	95分
咸鱼	16分
大黄	84分
❤️凤雏	12分

如何知道咸鱼的16分成绩排第几？



统计有多少个元素值 ≤ 16

共计3个元素（包含咸鱼自身）的值 ≤ 16。咸鱼应该排第3

成绩排名（倒数）：

卧龙	3分
凤雏	12分
咸鱼	16分
阿呆	76分
正南	83分
大黄	84分
旺财	90分
狗剩	95分
铁柱	97分
风间	100分

计数排序的实现

数组下标	0	1	2	3	4	5	6	7
A	2	4	3	0	2	3	0	3

使用计数排序的前提：
- 待排序元素的关键字是整数型
- 元素的取指范围是 [0, k)

此例中，待排序元素取值范围是 [0, 5)

Step 1：创建一个长度为 k 的辅助数组，统计每个元素的出现次数（辅助数组的下标对应元素值）

数组下标	0	1	2	3	4
C	2	0	2	3	1

0 出现过2次

1 出现过0次

3 出现过3次

Step 2：再次处理辅助数组，用 C[i] 统计 ≤ i 的元素个数

数组下标	0	1	2	3	4
C	2	2	4	7	8

≤ 0的元素累计出现了2个

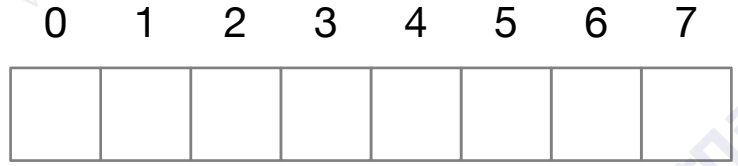
≤ 3的元素累计出现了7个

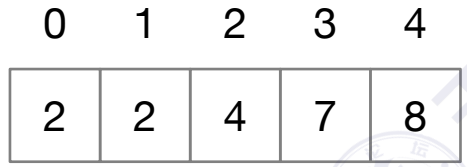
≤ 1的元素累计出现了2个

Step 3：利用辅助数组C实现排序

计数排序的实现

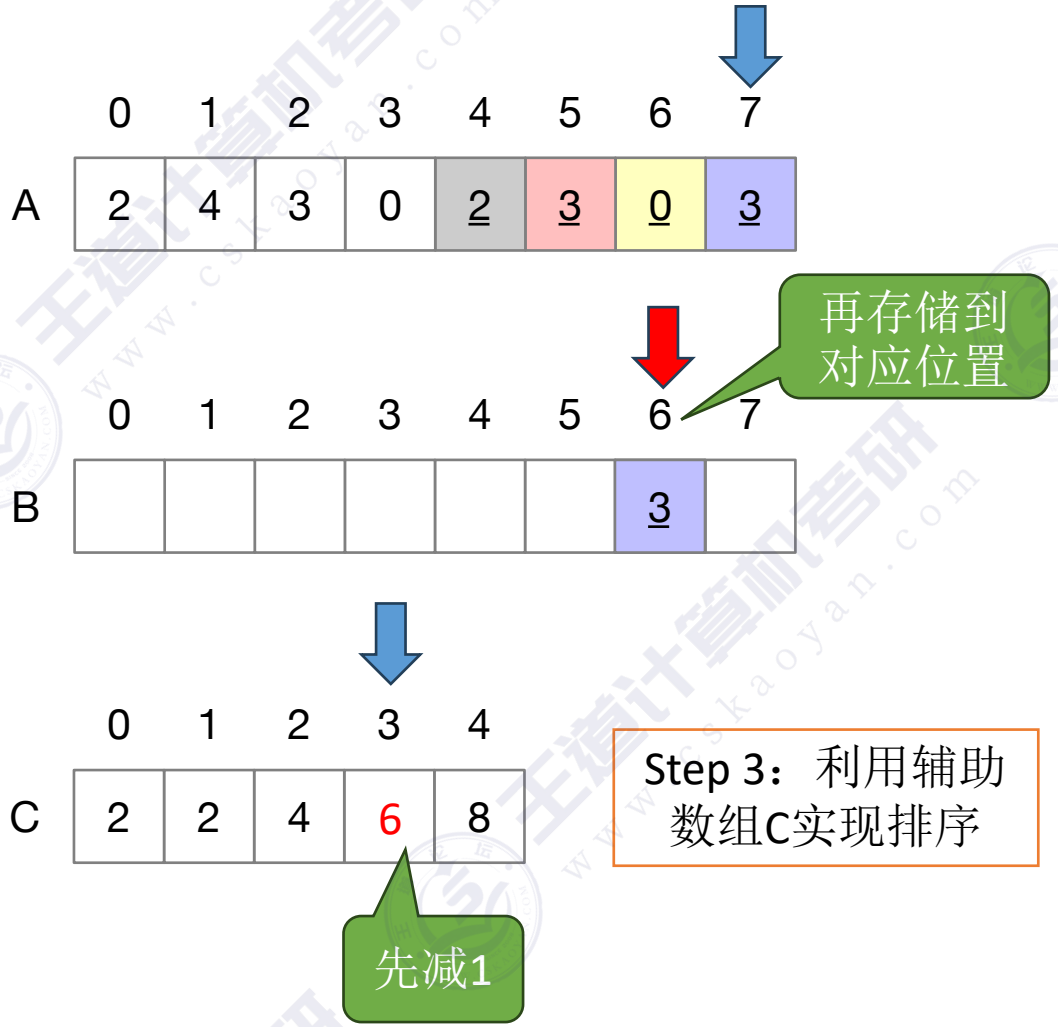
输出数组B的长度与数组A相同，用于存储排序结果



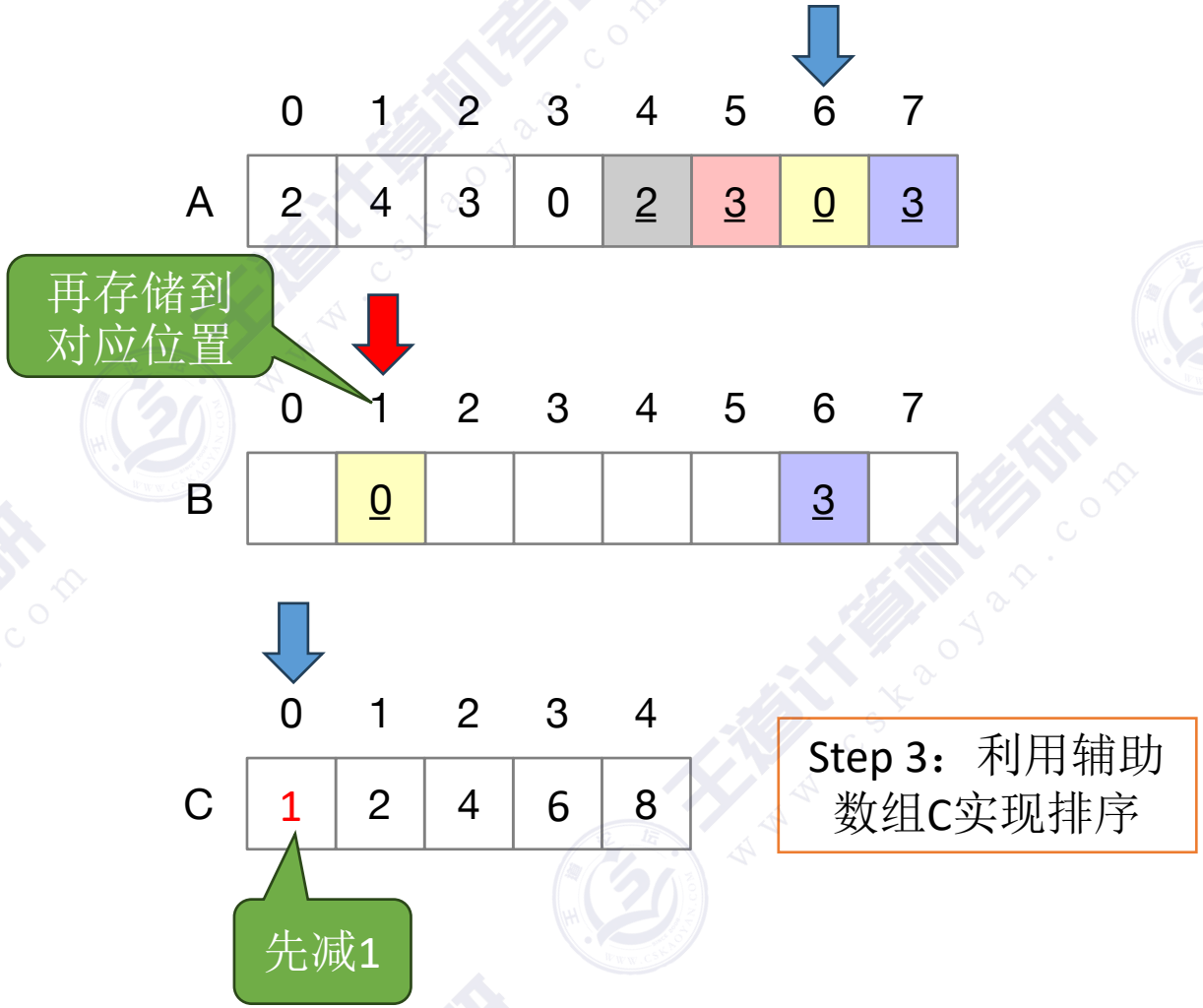


Step 3：利用辅助数组C实现排序

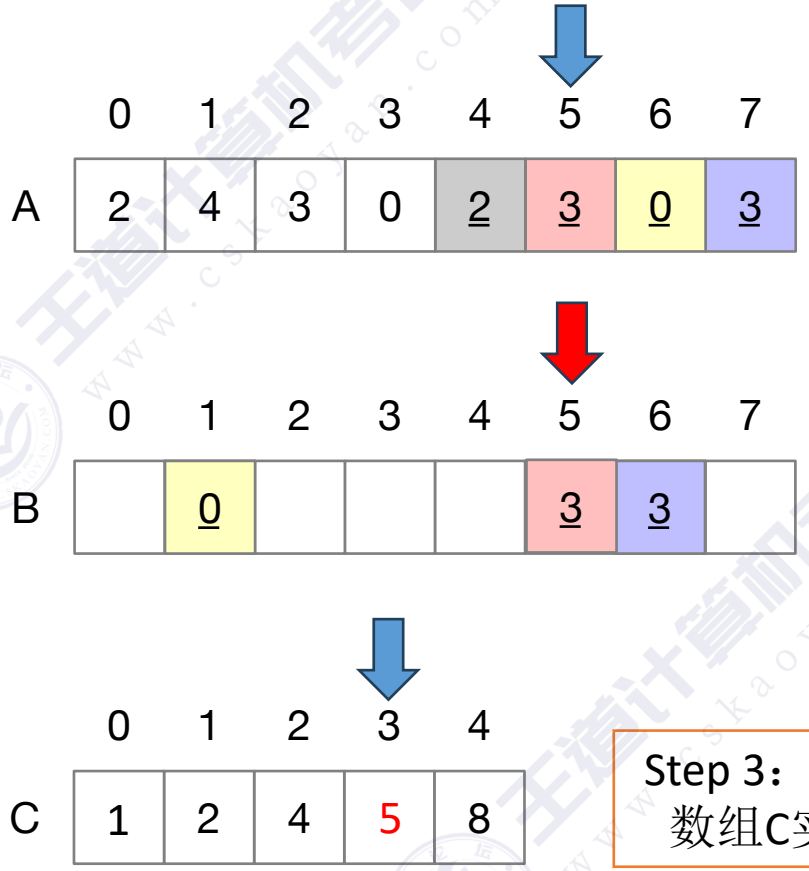
计数排序的实现



计数排序的实现

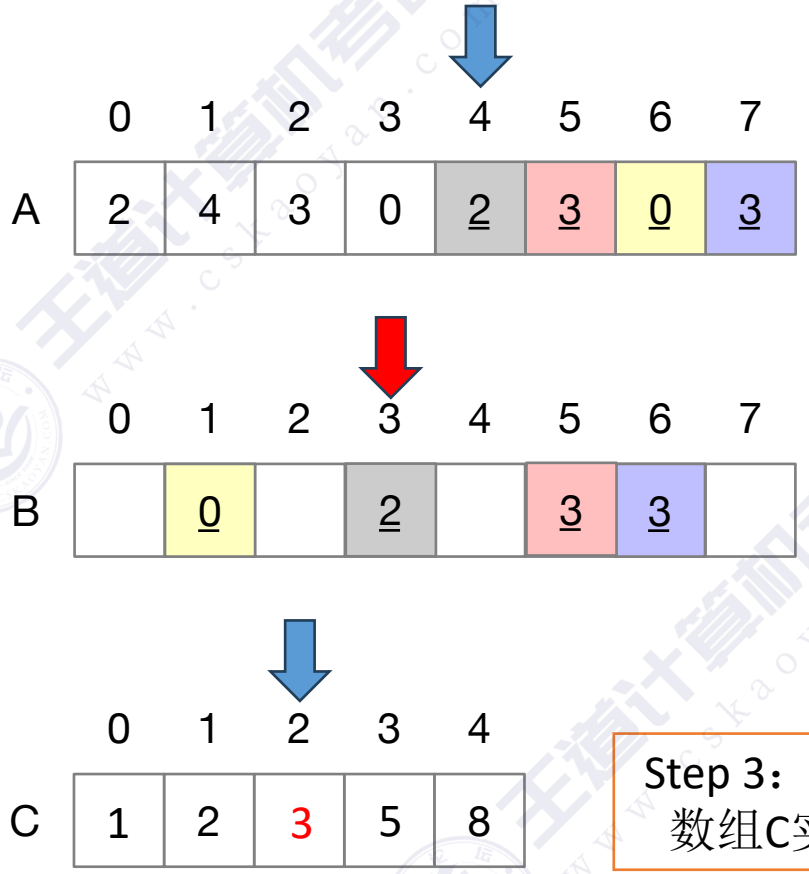


计数排序的实现



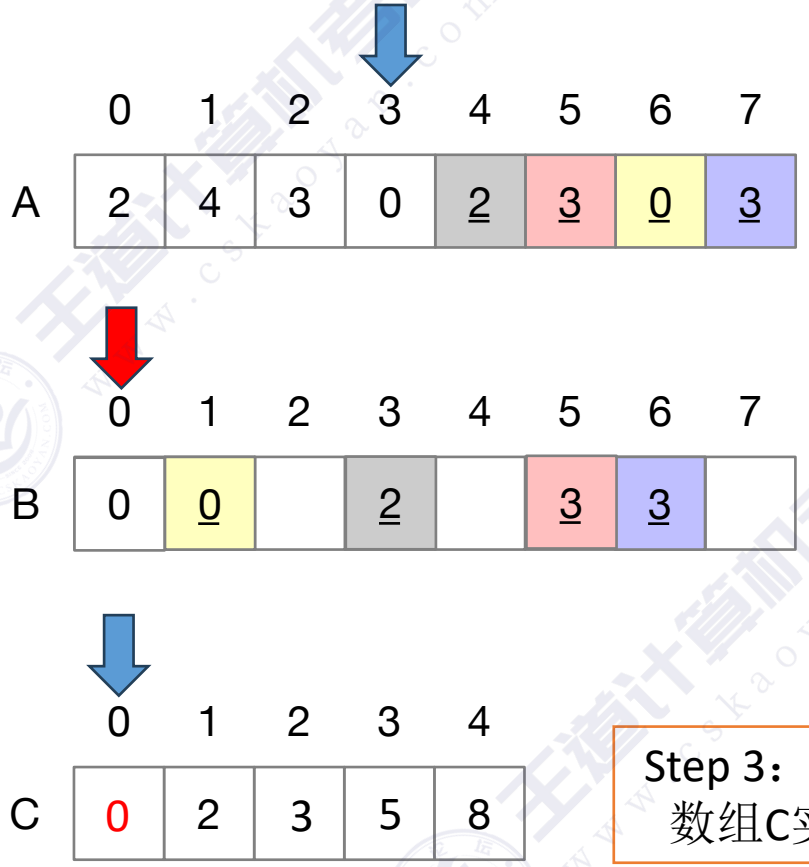
Step 3：利用辅助数组C实现排序

计数排序的实现



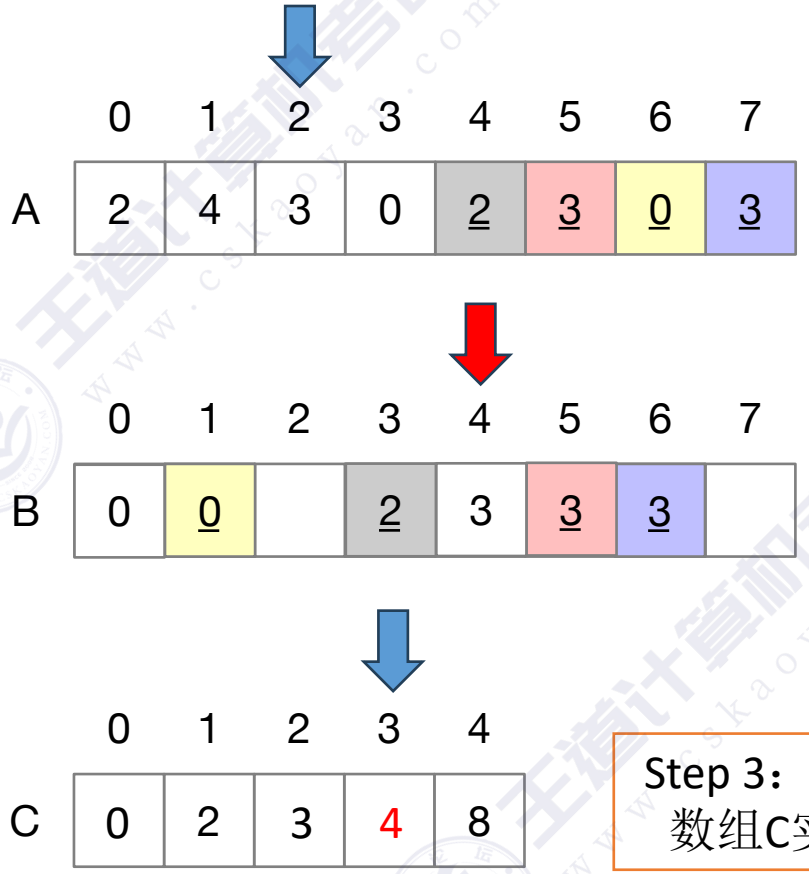
Step 3：利用辅助数组C实现排序

计数排序的实现



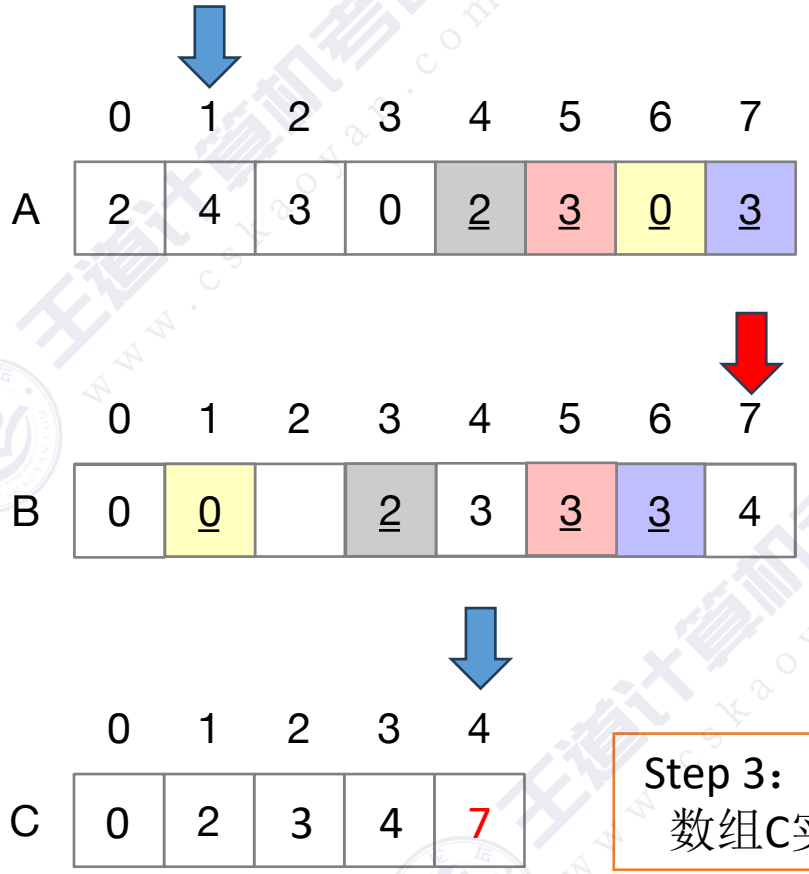
Step 3：利用辅助数组C实现排序

计数排序的实现



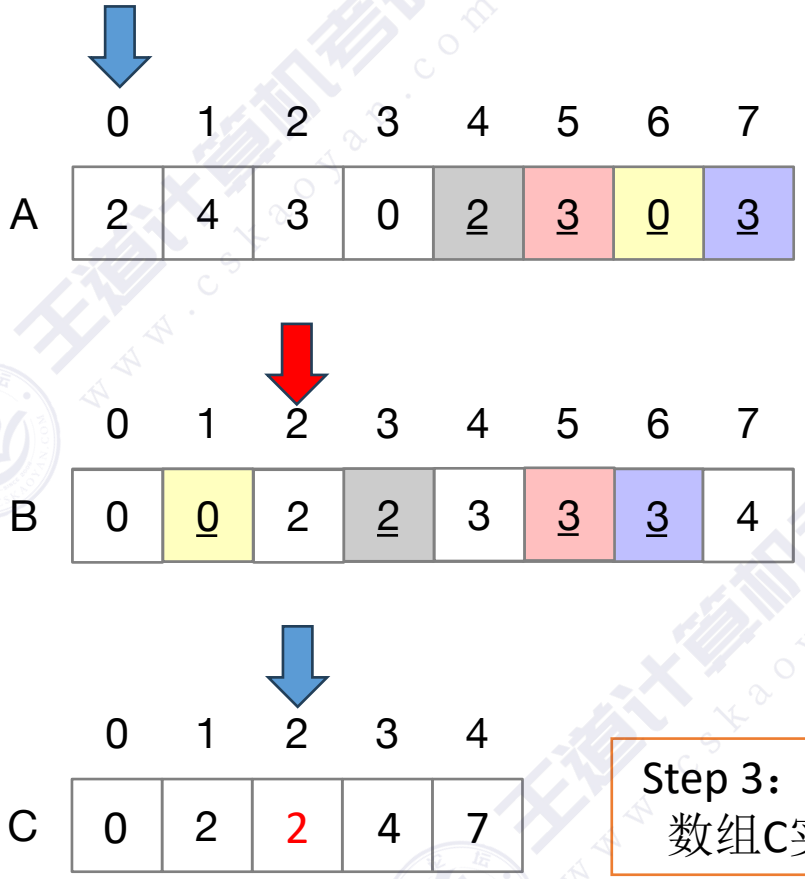
Step 3：利用辅助数组C实现排序

计数排序的实现



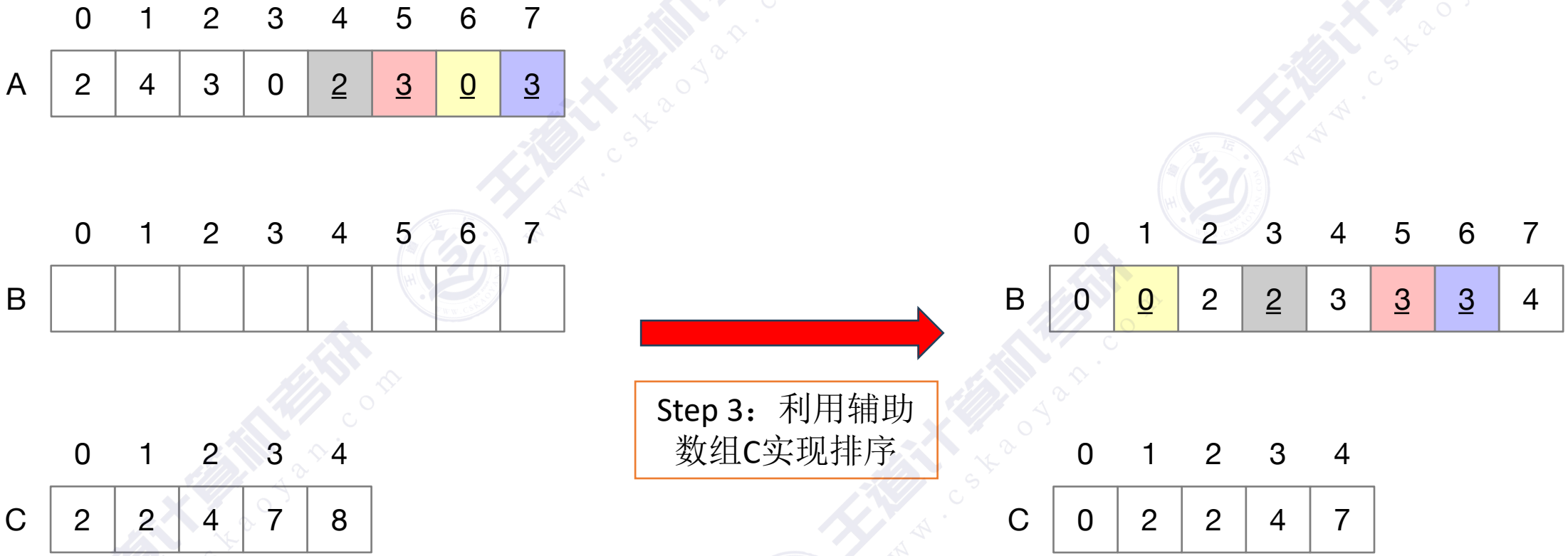
Step 3：利用辅助数组C实现排序

计数排序的实现



Step 3：利用辅助数组C实现排序

计数排序的实现



计数排序的实现（C语言）

A是输入数组（待排序数组）　B是输出数组　n是A[ ]的长度　k反映A[ ]的取值范围

```
// 计数排序
void CountSort(int A[], int B[], int n, int k){
  int i,C[k];              //辅助数组C的长度取决于待排序元素取值范围[0, k)
  for(i=0;i<k;i++)         //初始化计数数组
    C[i]=0;
  for(i=0;i<n;i++)         //Step1：遍历待排序数组，统计每个关键字的出现次数
    C[A[i]]++;
  for(i=1;i<k;i++)         //Step2：再次处理辅助数组，统计不大于 i 的元素个数
    C[i]=C[i]+C[i-1];      //C[i]保存的是小于或等于 i 的元素个数
  for(i=n-1;i>=0;i--){     //Step3：利用辅助数组C实现计数排序（从后往前处理）
    C[A[i]]=C[A[i]]-1;
    B[C[A[i]]]=A[i];   //将元素 A[i]放在输出数组 B[]的正确位置上
  }
}
```

注：这段C语言代码改写自《算法导论》，适合用于教学、考试。实践中可拓展这段代码，让计数排序更具备通用性（见计数排序代码附件）

计数排序的复杂度分析

来源于输出数组B　来源于辅助数组C

空间复杂度 $= O(n+k)$
时间复杂度 $= O(n+k)$

若 $k = O(n)$，计数排序的时间复杂度 $= O(n)$，此时计数排序的时间效率优于快速排序、堆排序等排序算法的 $O(n\log_2 n)$

若 $k > O(n\log_2 n)$，计数排序的时间复杂度 $> O(n\log_2 n)$，此时计数排序的时间效率不如快速排序、堆排序等排序算法的 $O(n\log_2 n)$

- 计数排序是典型的“空间”换“时间”思想
- 适用场景：当待排序数组的取指范围 k 较小时，可以考虑使用计数排序的思想

A是输入数组（待排序数组）　B是输出数组　n是A[ ]的长度　k反映A[ ]的取值范围

```
// 计数排序
void CountSort(int A[], int B[], int n, int k){
  int i,C[k];                          //辅助数组C的长度取决于待排序
  for(i=0;i<k;i++)                     //初始化计数数组   O(k)
    C[i]=0;
  for(i=0;i<n;i++)                     //Step1：遍历待排序数组，统   O(n)
    C[A[i]]++;
  for(i=1;i<k;i++)                     //Step2：再次处理辅助数组，   O(k)
    C[i]=C[i]+C[i-1];                  //C[i]保存的是小于或等于 i
  for(i=n-1;i>=0;i--){                 //Step3：利用辅助数组C实现计
    C[A[i]]=C[A[i]]-1;                                              O(n)
    B[C[A[i]]]=A[i];                   //将元素 A[i]放在输出数组 B[]
  }
}
```

计数排序在考试中的应用：2013年41题

41.（13 分）已知一个整数序列 $A=(a_0, a_1, \cdots, a_{n-1})$，其中 $0 \leqslant a_i < n$（$0 \leqslant i < n$）。若存在 $a_{p1}=a_{p2}=\cdots=a_{pm}=x$ 且 $m>n/2$（$0 \leqslant p_k < n$, $1 \leqslant k \leqslant m$），则称 x 为 A 的主元素。例如 $A=(0, 5, 5, 3, 5, 7, 5, 5)$，则 5 为主元素；又如 $A=(0, 5, 5, 3, 5, 1, 5, 7)$，则 A 中没有主元素。假设 A 中的 n 个元素保存在一个一维数组中，请设计一个尽可能高效的算法，找出 A 的主元素。若存在主元素，则输出该元素；否则输出−1。要求：

1）给出算法的基本设计思想。

2）根据设计思想，采用 C、C++或 Java 语言描述算法，关键之处给出注释。

3）说明你所设计算法的时间复杂度和空间复杂度。

计数排序在考试中的应用：2018年41题

41.（13 分）给定一个含 n（$n \geq 1$）个整数的数组，请设计一个在时间上尽可能高效的算法，找出数组中未出现的最小正整数。例如，数组{−5, 3, 2, 3}中未出现的最小正整数是 1；数组{1, 2, 3}中未出现的最小正整数是 4。要求：

1）给出算法的基本设计思想。

2）根据设计思想，采用 C 或 C++语言描述算法，关键之处给出注释。

3）说明你所设计算法的时间复杂度和空间复杂度。

知识回顾与重要考点

若 $O(k) > O(n\log_2 n)$，就不如用快速排序、堆排序

- 计数排序
  - 适用条件
    - 待排序元素的**关键字是整数型**
    - 待排序**元素的取值范围是 [0, k)**，且**k值较小**，k与O(n)同数量级
  - 算法实现
    - 创建一个长度为 k 的**辅助数组C[k]**
    - Step 1：遍历待排序数组A[n]，**用辅助数组C[i]统计 A[ ] 中每个元素值 i 出现的次数（元素值只可能是 0~k-1，与辅助数组下标一一对应）**
    - Step 2：再次处理辅助数组 C，最终**用 C[i] 统计 ≤ i 的元素个数**
    - Step 3：**从后往前**依次处理各个待排序元素，结合辅助数组C[ ]实现排序（注意：**从后往前处理各个元素可确保排序的稳定性**，详细过程建议自己手绘复习）
  - 算法性能
    - 时间复杂度
      - O(n+k)
      - **当 k与O(n)同数量级时，时间复杂度 = O(n)**
    - 空间复杂度
      - O(n+k)
        - O(n)由输出数组B[n]导致
        - O(k)由辅助数组C[k]导致
      - 当 k与O(n)同数量级时，**空间**复杂度 = O(n)
    - 稳定性 —— 稳！

计数排序（实战版）

注：计数排序代码（实战版）见代码pdf附件