

本节内容

基数排序

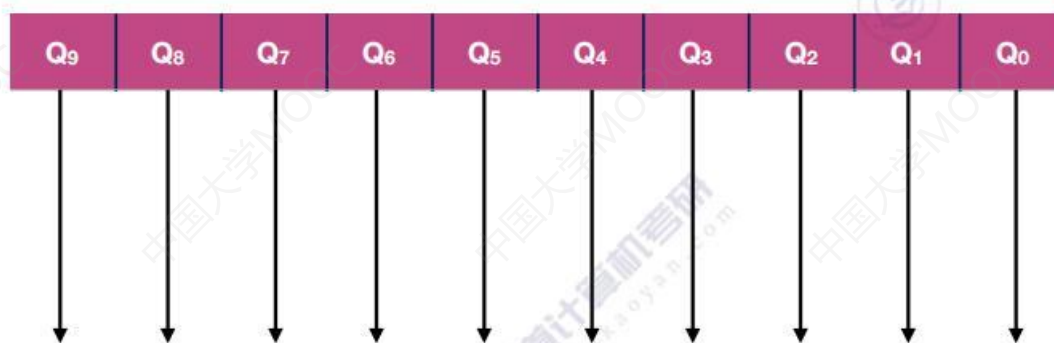
(Radix Sort)

王道考研/CSKAOYAN.COM

基数排序

520 → 211 → 438 → 888 → 007 → 111 → 985 → 666 → 996 → 233 → 168

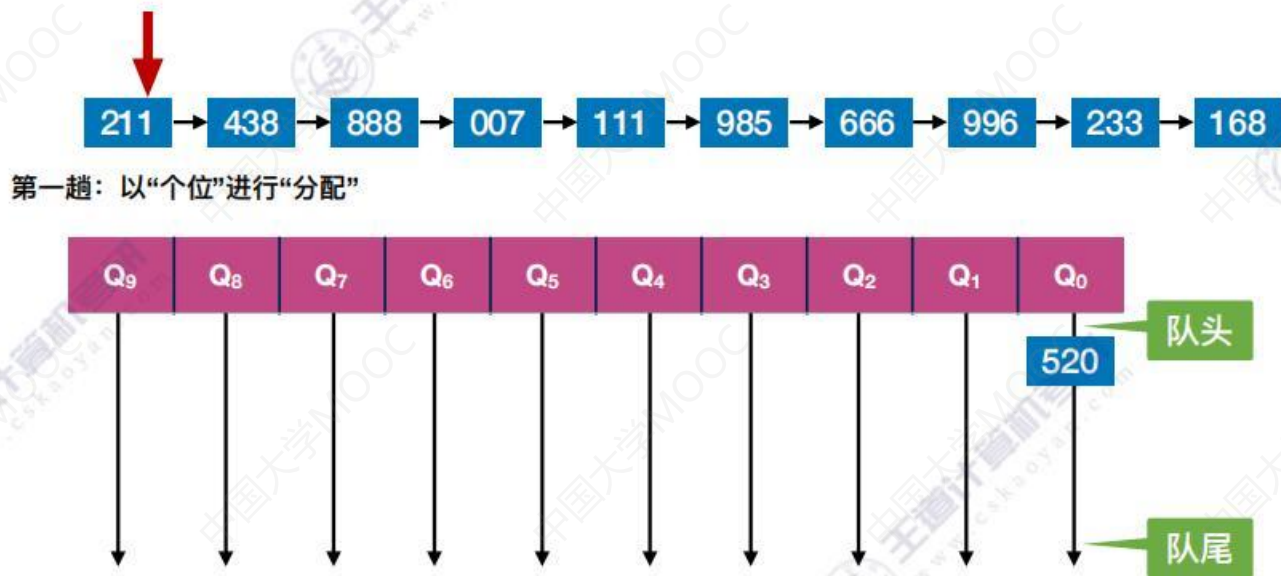
第一趟：以“个位”进行“分配”



要求：得到按关键字“递减”的有序序列。

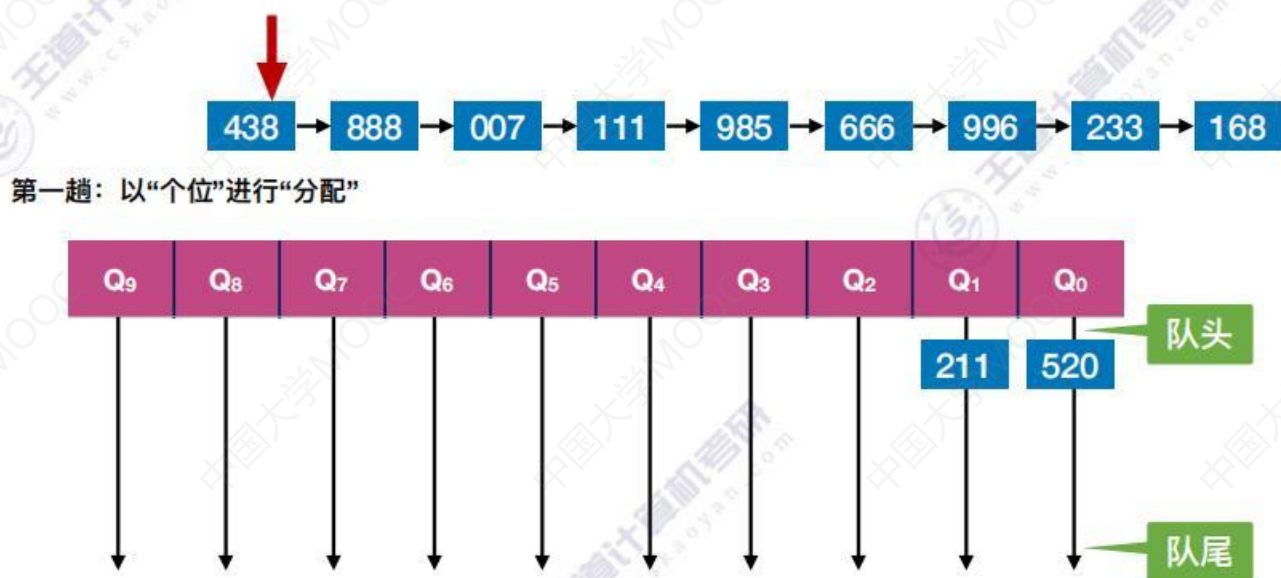
王道考研/CSKAOYAN.COM

基数排序



王道考研/CSKAOYAN.COM

基数排序

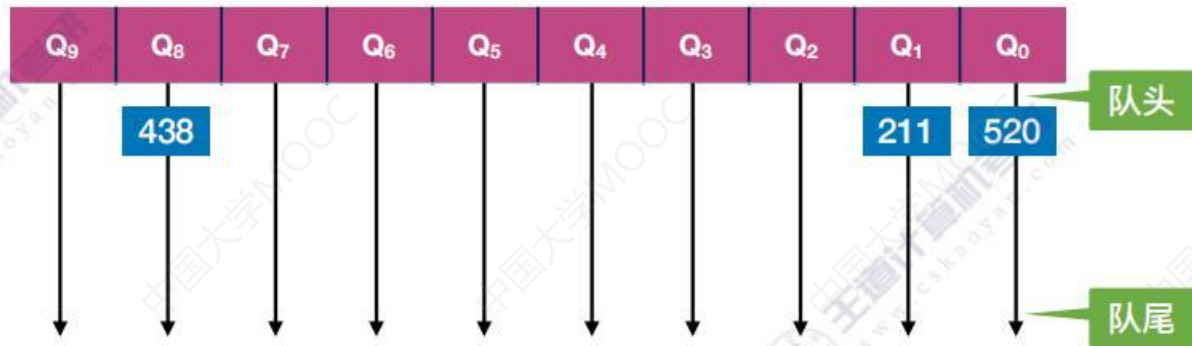


王道考研/CSKAOYAN.COM

基数排序

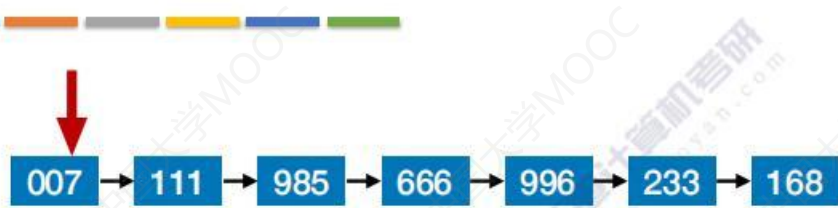


第一趟：以“个位”进行“分配”

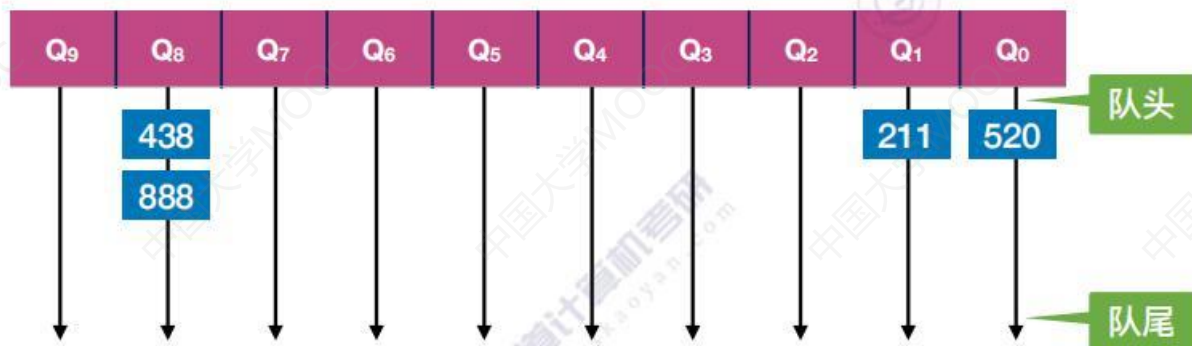


王道考研/CSKAOYAN.COM

基数排序

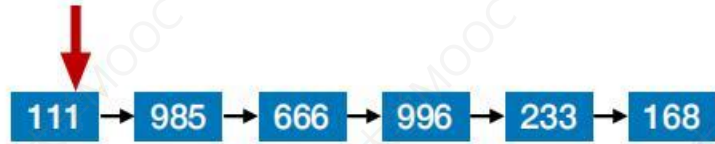


第一趟：以“个位”进行“分配”

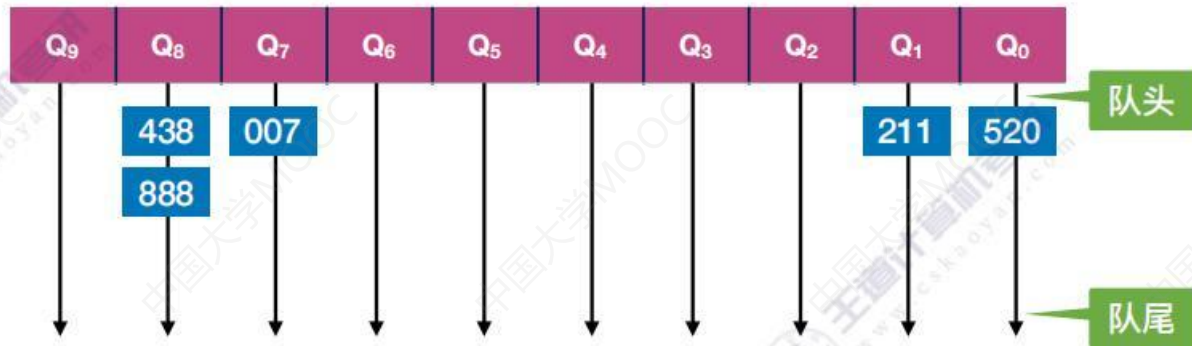


王道考研/CSKAOYAN.COM

基数排序



第一趟：以“个位”进行“分配”

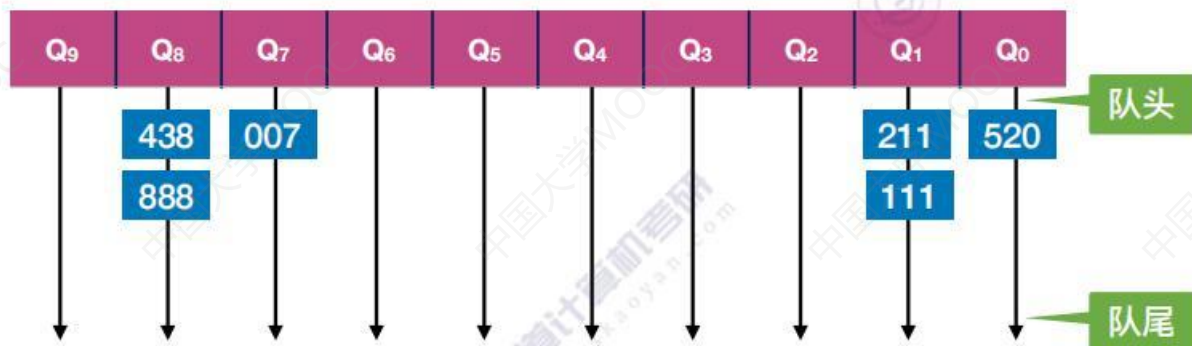


王道考研/CSKAOYAN.COM

基数排序



第一趟：以“个位”进行“分配”

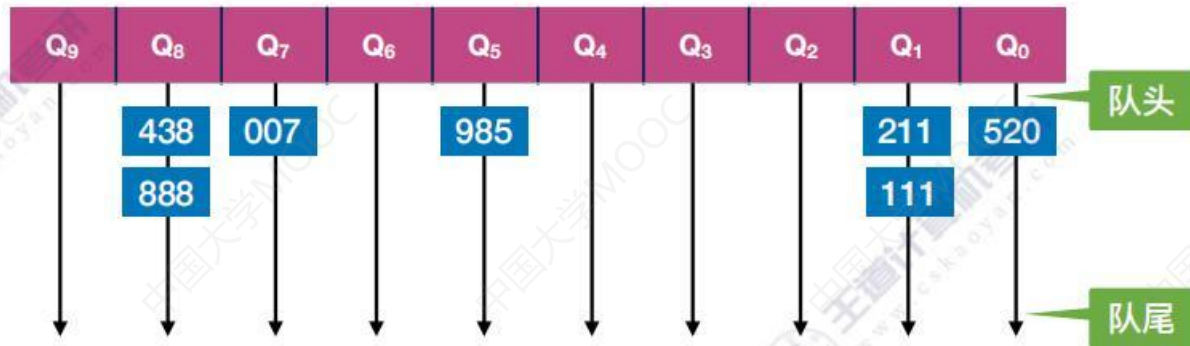


王道考研/CSKAOYAN.COM

基数排序



第一趟：以“个位”进行“分配”

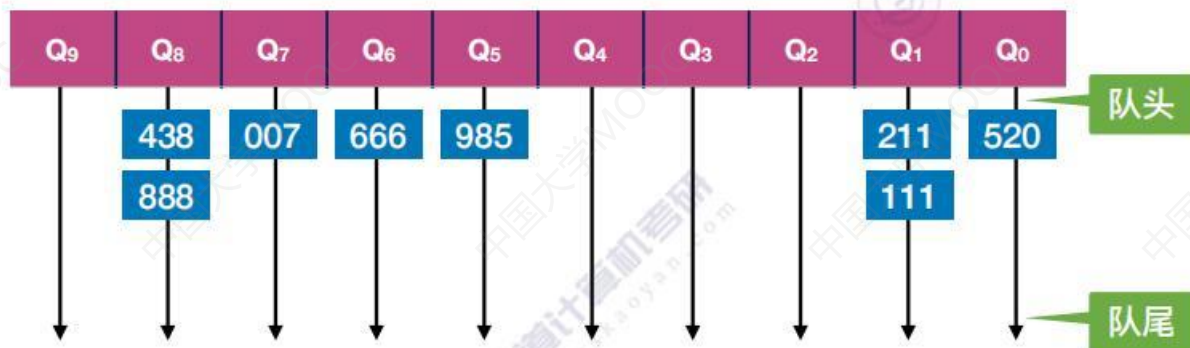


王道考研/CSKAOYAN.COM

基数排序



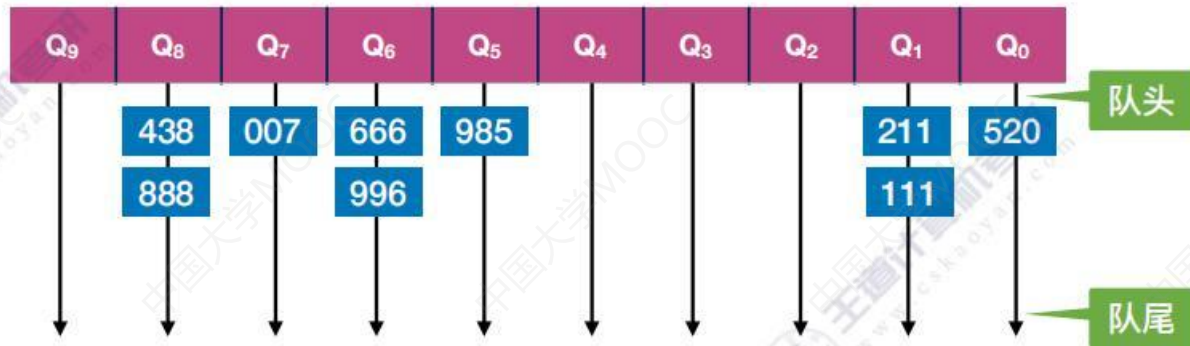
第一趟：以“个位”进行“分配”



王道考研/CSKAOYAN.COM

基数排序

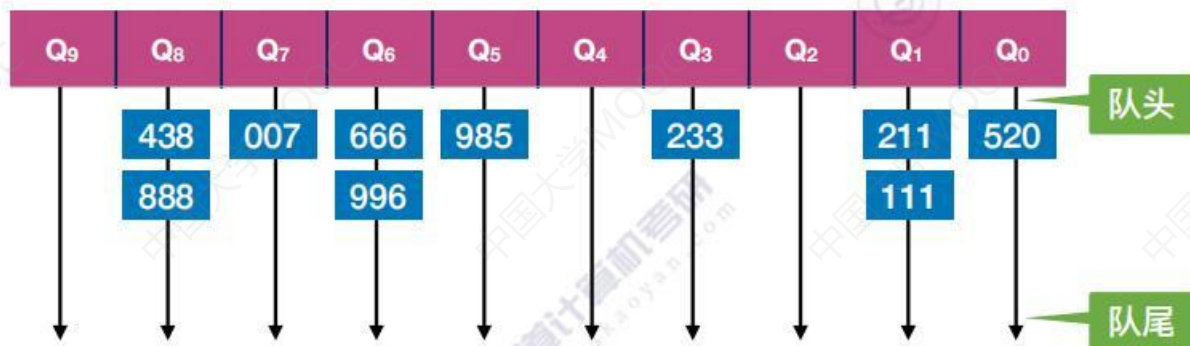
第一趟：以“个位”进行“分配”



王道考研/CSKAOYAN.COM

基数排序

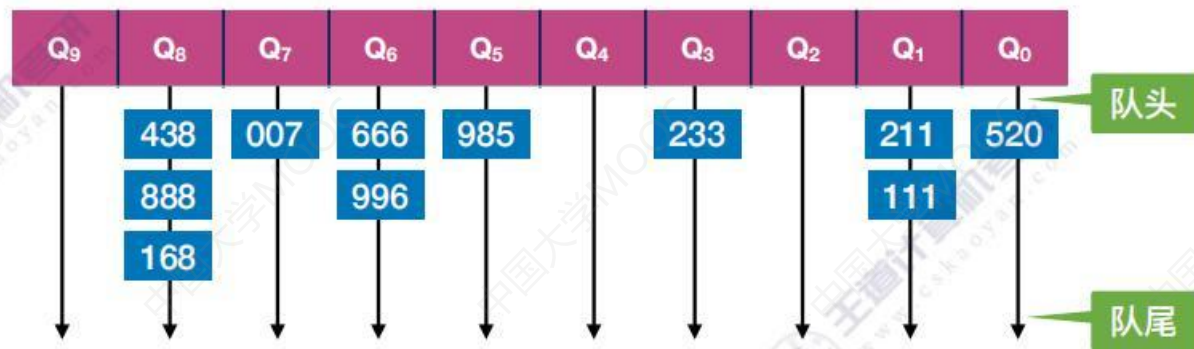
第一趟：以“个位”进行“分配”



王道考研/CSKAOYAN.COM

基数排序

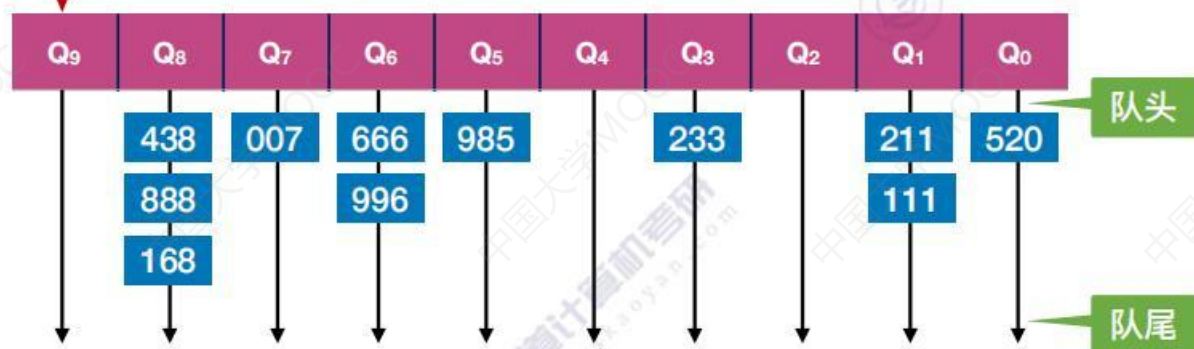
第一趟“分配”结束



王道考研/CSKAOYAN.COM

基数排序

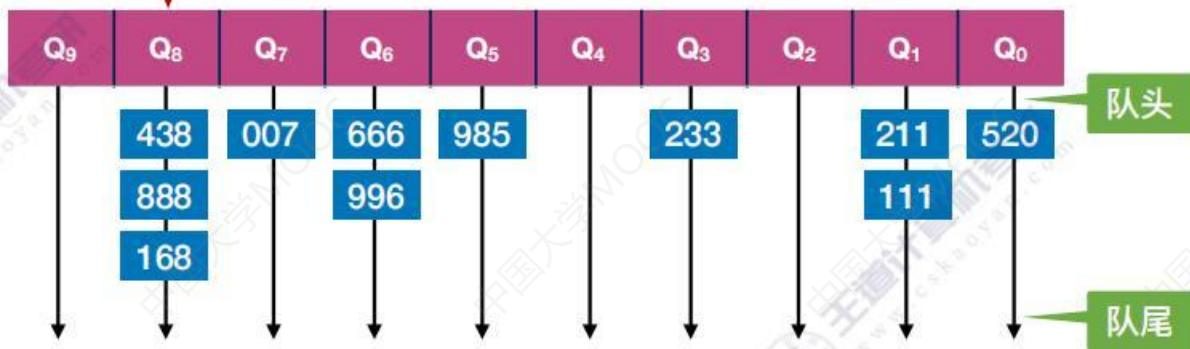
第一趟“收集”



王道考研/CSKAOYAN.COM

基数排序

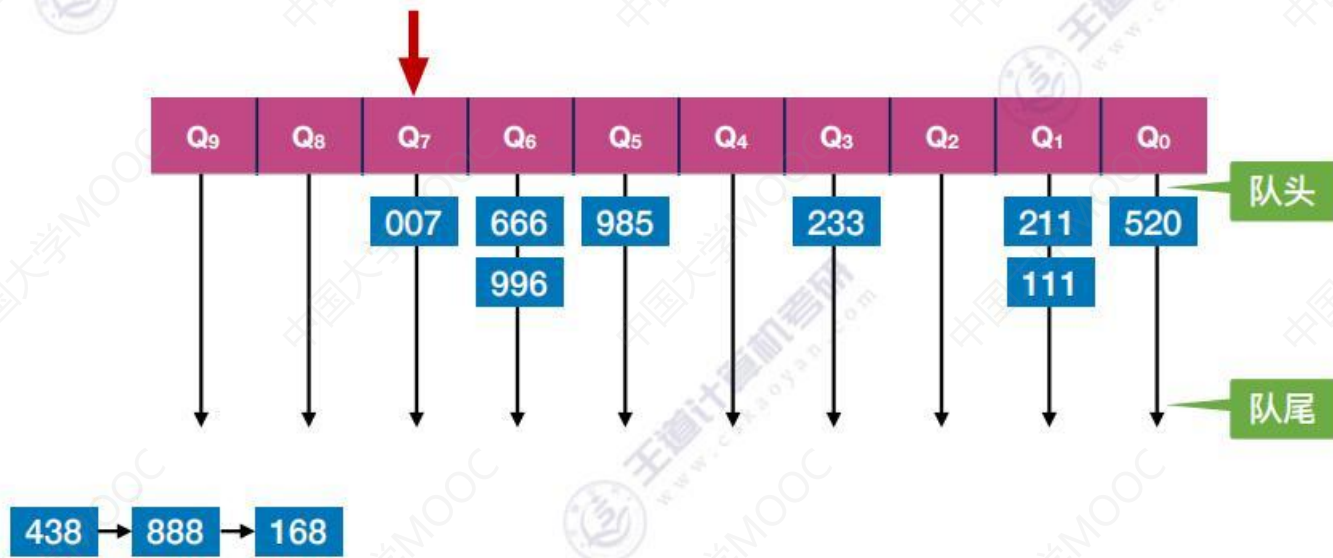
第一趟“收集”



王道考研/CSKAOYAN.COM

基数排序

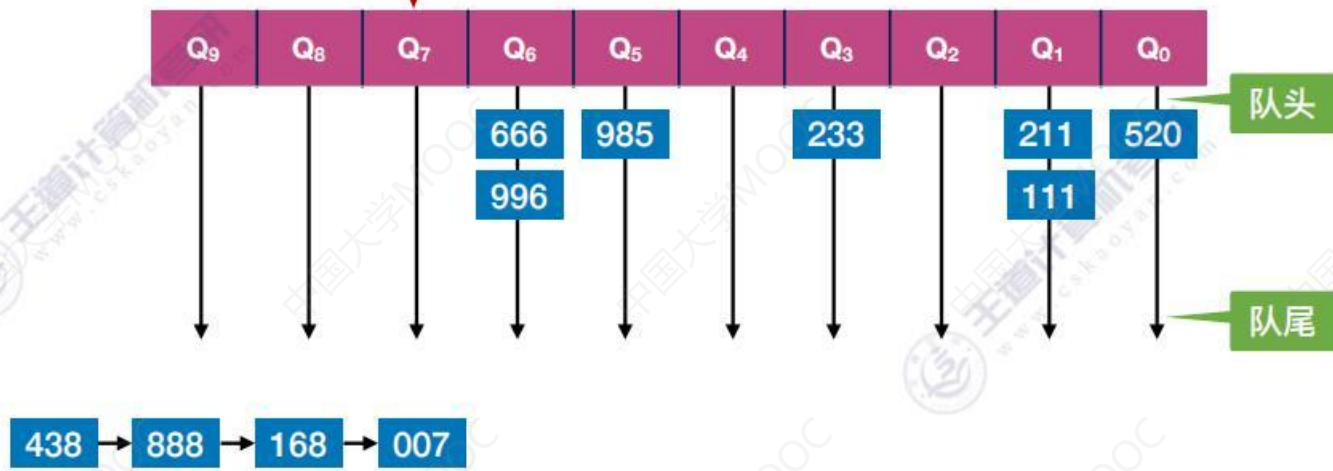
第一趟“收集”



王道考研/CSKAOYAN.COM

基数排序

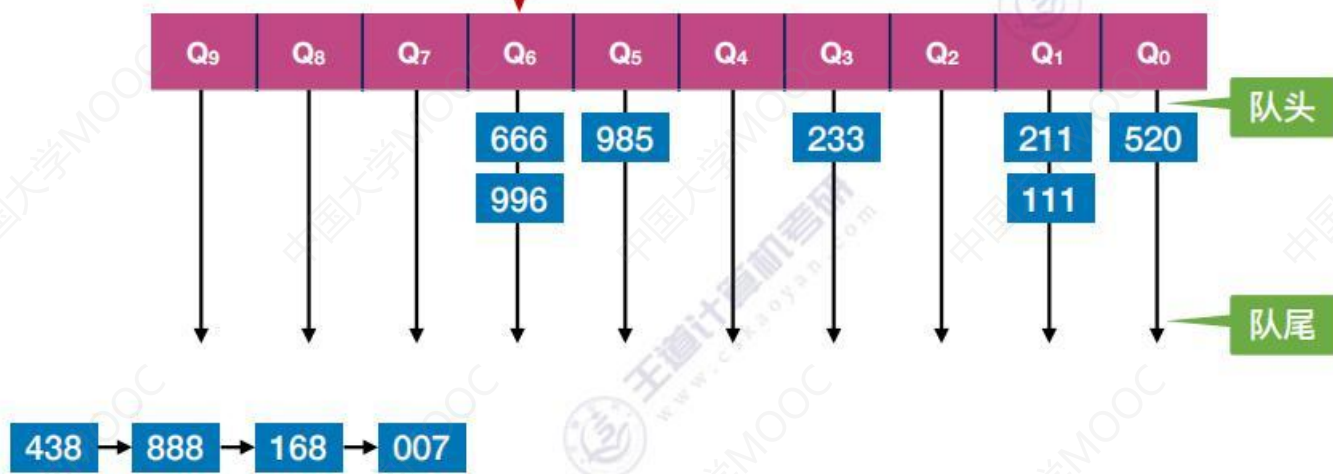
第一趟“收集”



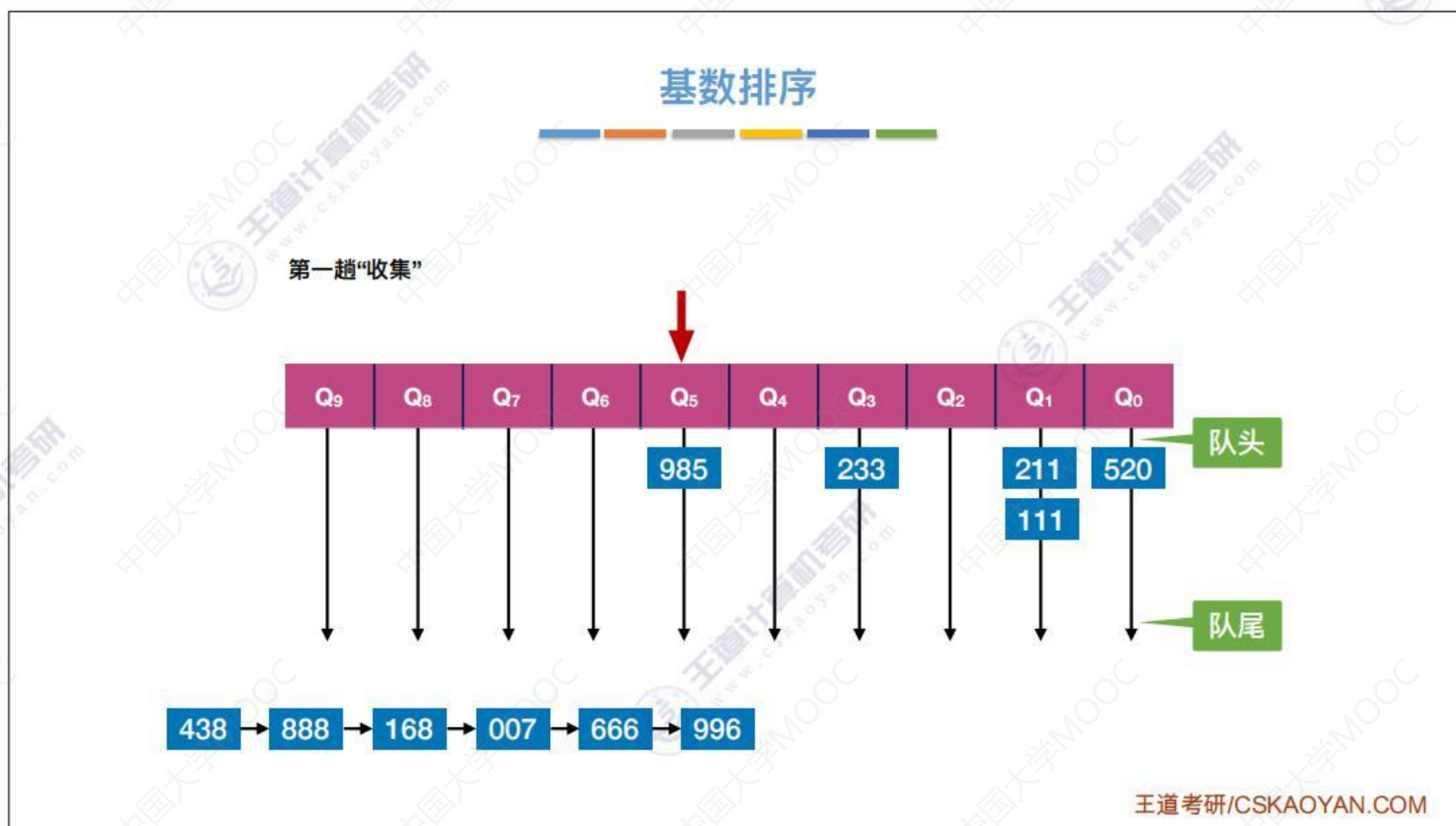
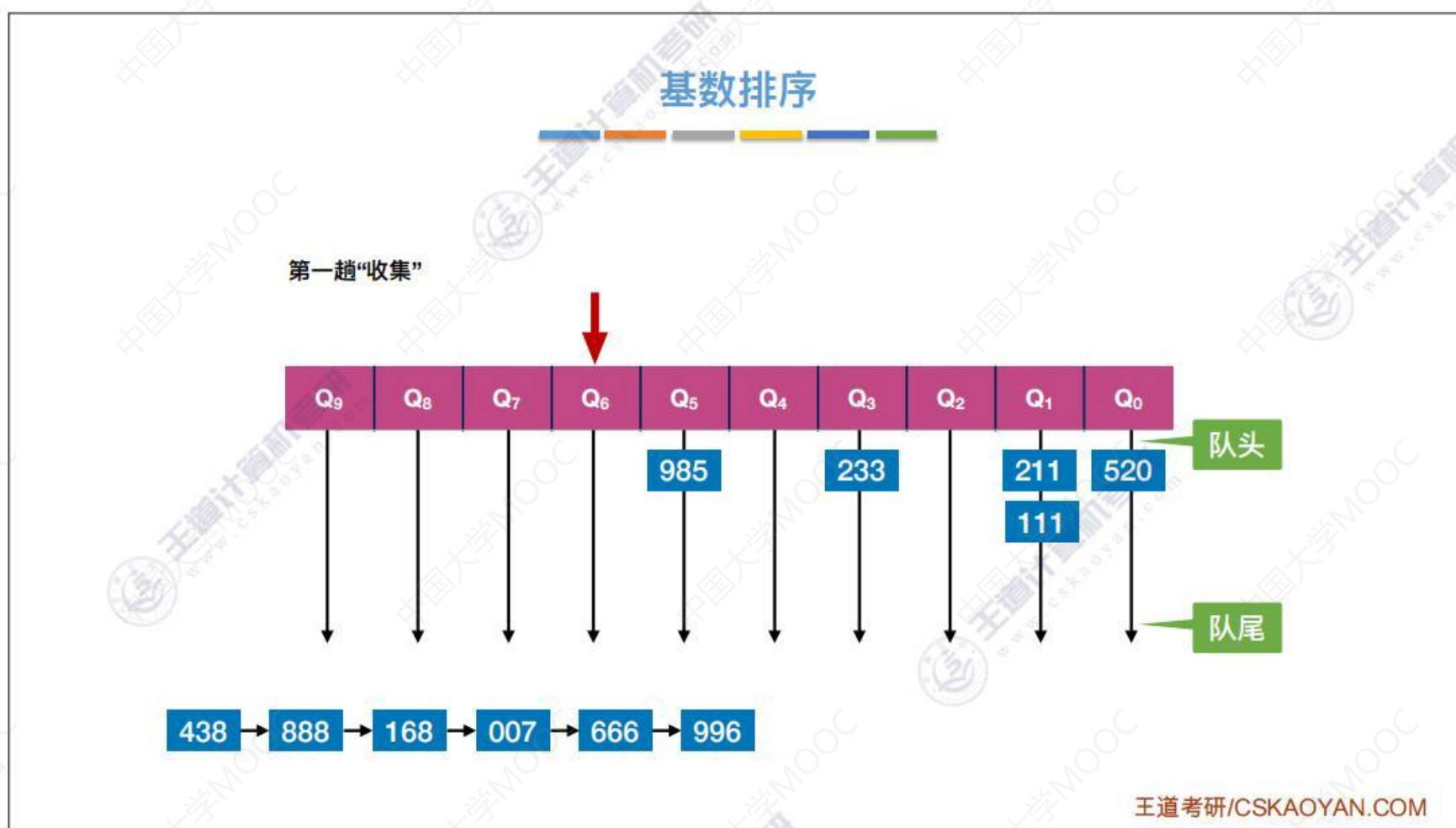
王道考研/CSKAOYAN.COM

基数排序

第一趟“收集”

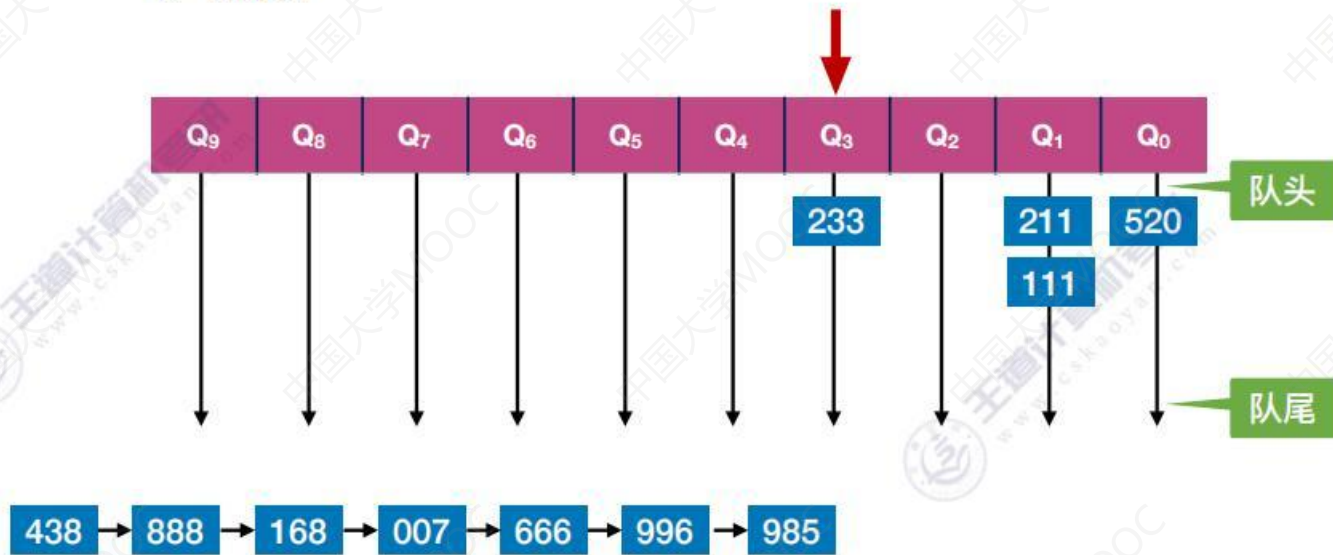


王道考研/CSKAOYAN.COM



基数排序

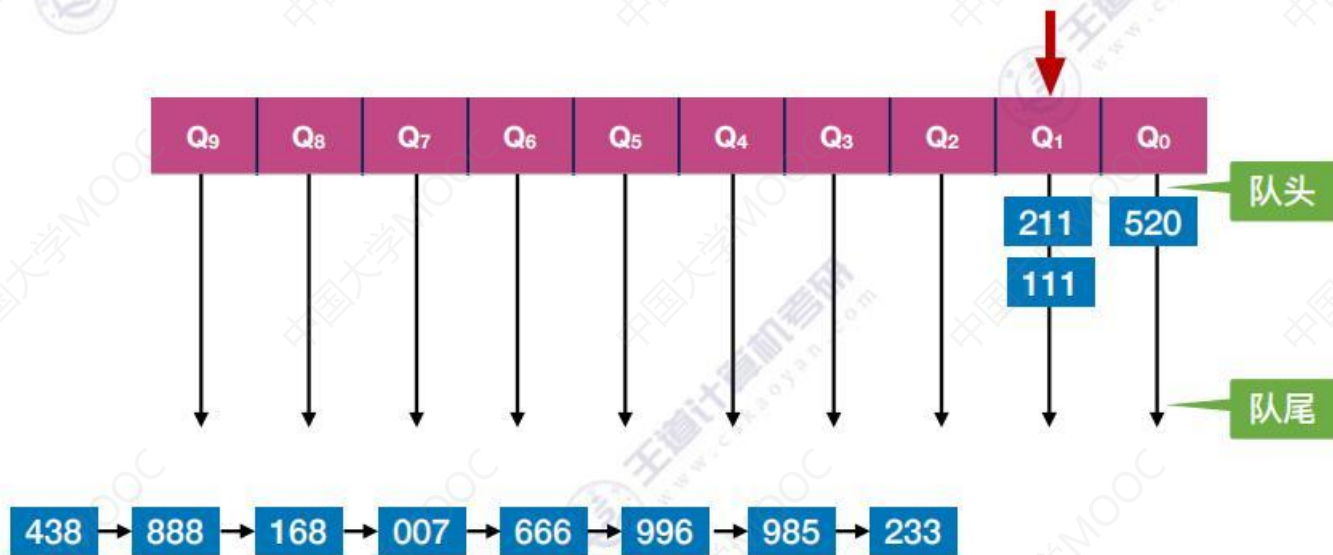
第一趟“收集”



王道考研/CSKAOYAN.COM

基数排序

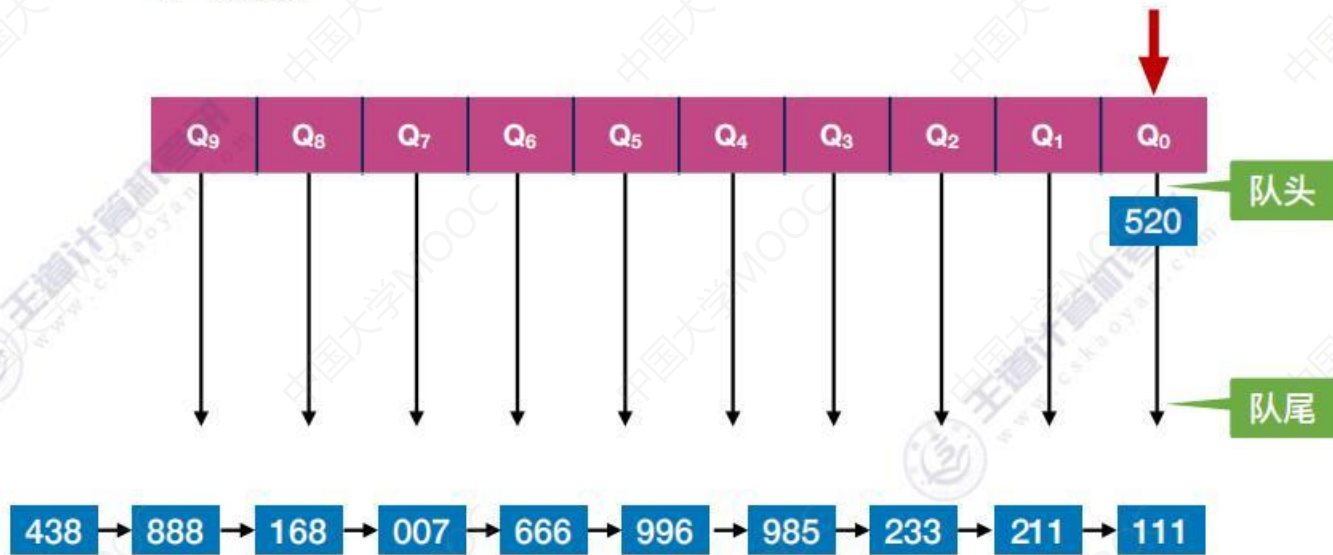
第一趟“收集”



王道考研/CSKAOYAN.COM

基数排序

第一趟“收集”



王道考研/CSKAOYAN.COM

基数排序

第一趟“收集”结束：得到按“个位”递减排序的序列

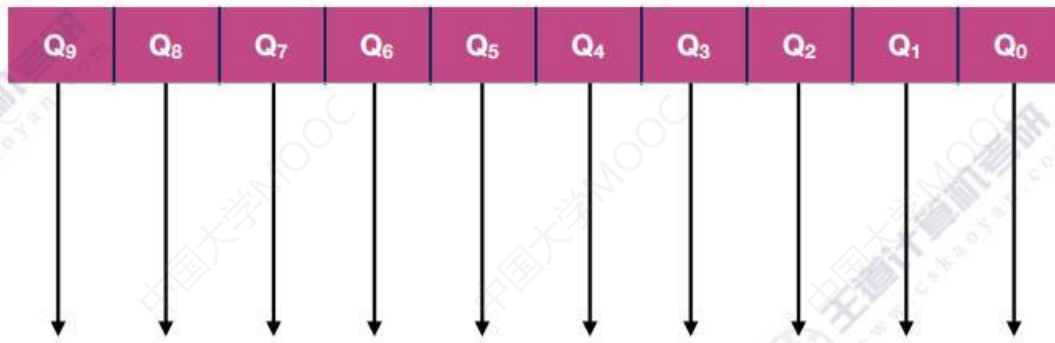


王道考研/CSKAOYAN.COM

基数排序



第二趟：以“十位”进行“分配”

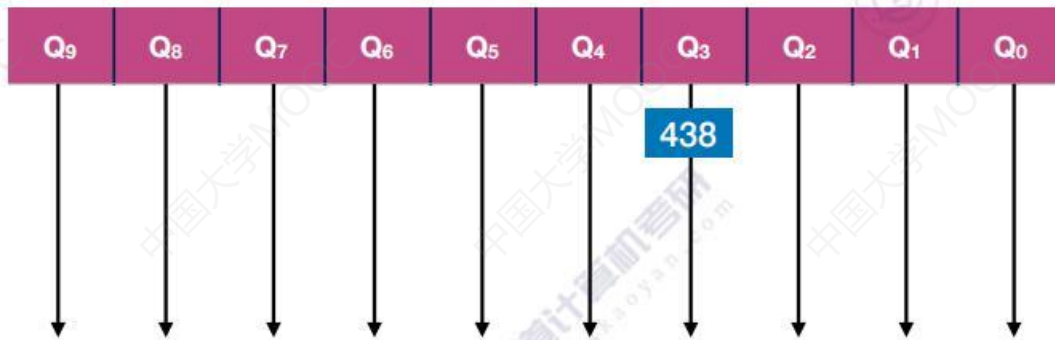


王道考研/CSKAOYAN.COM

基数排序



第二趟：以“十位”进行“分配”

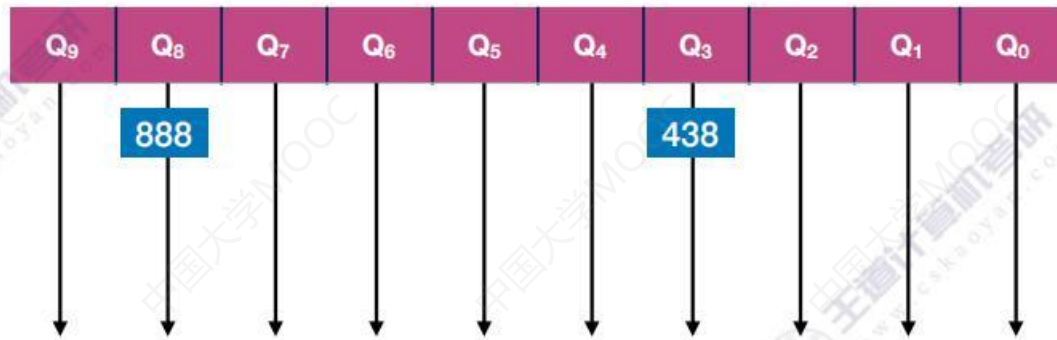


王道考研/CSKAOYAN.COM

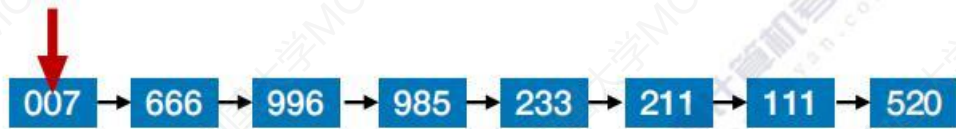
基数排序



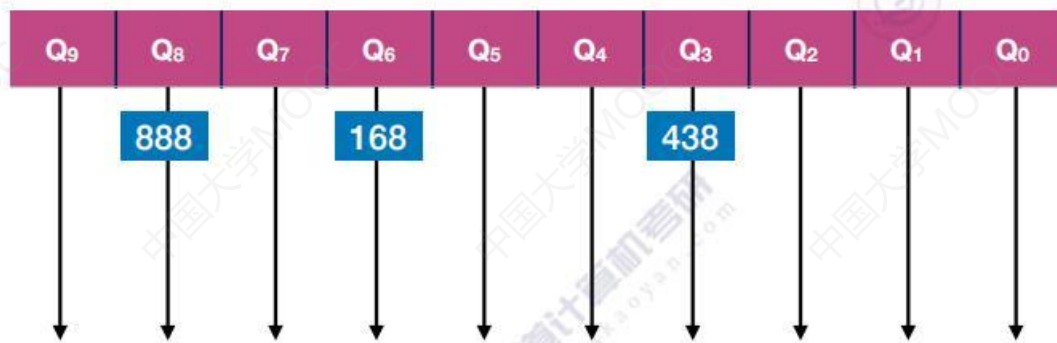
第二趟：以“十位”进行“分配”

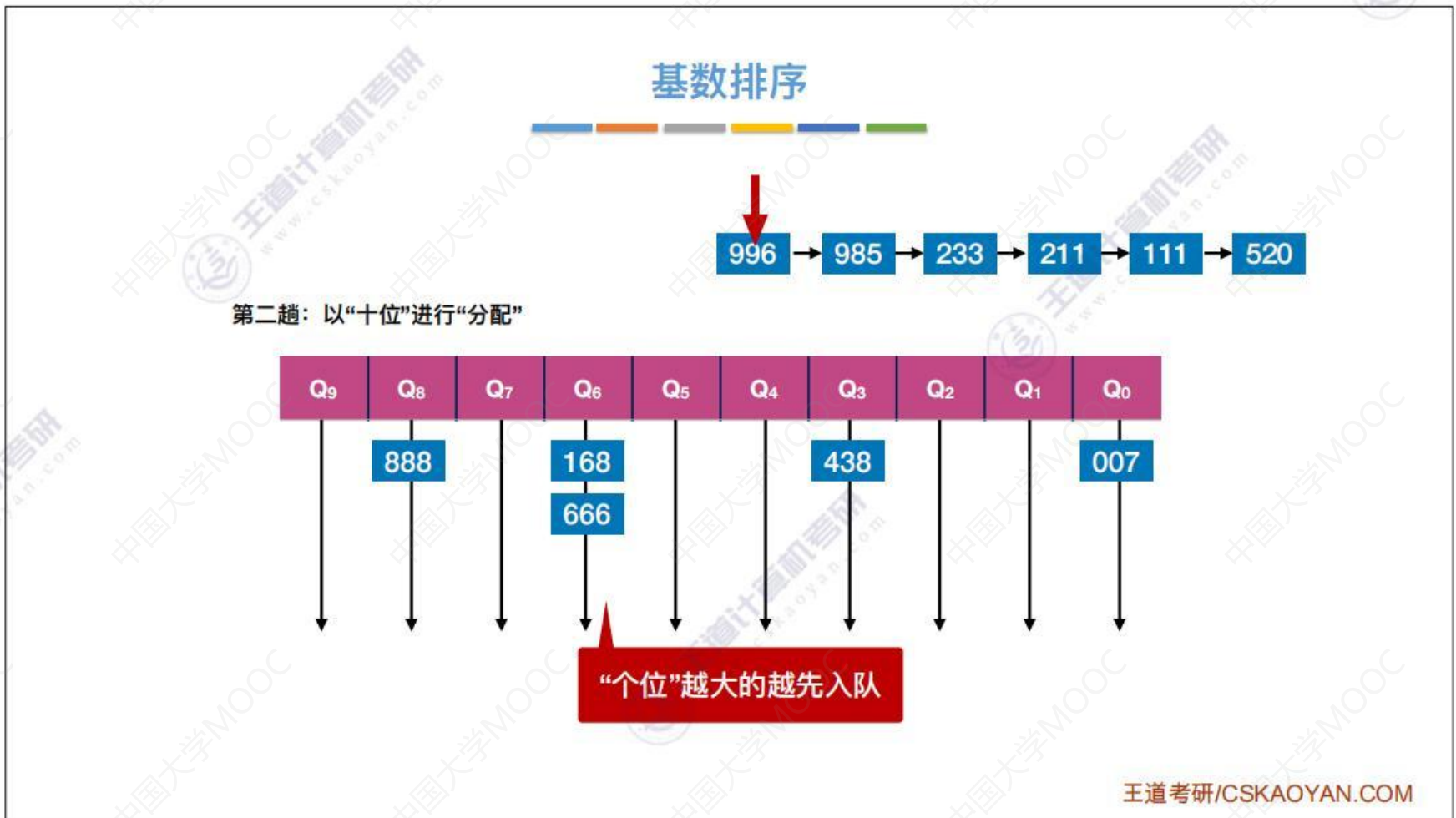
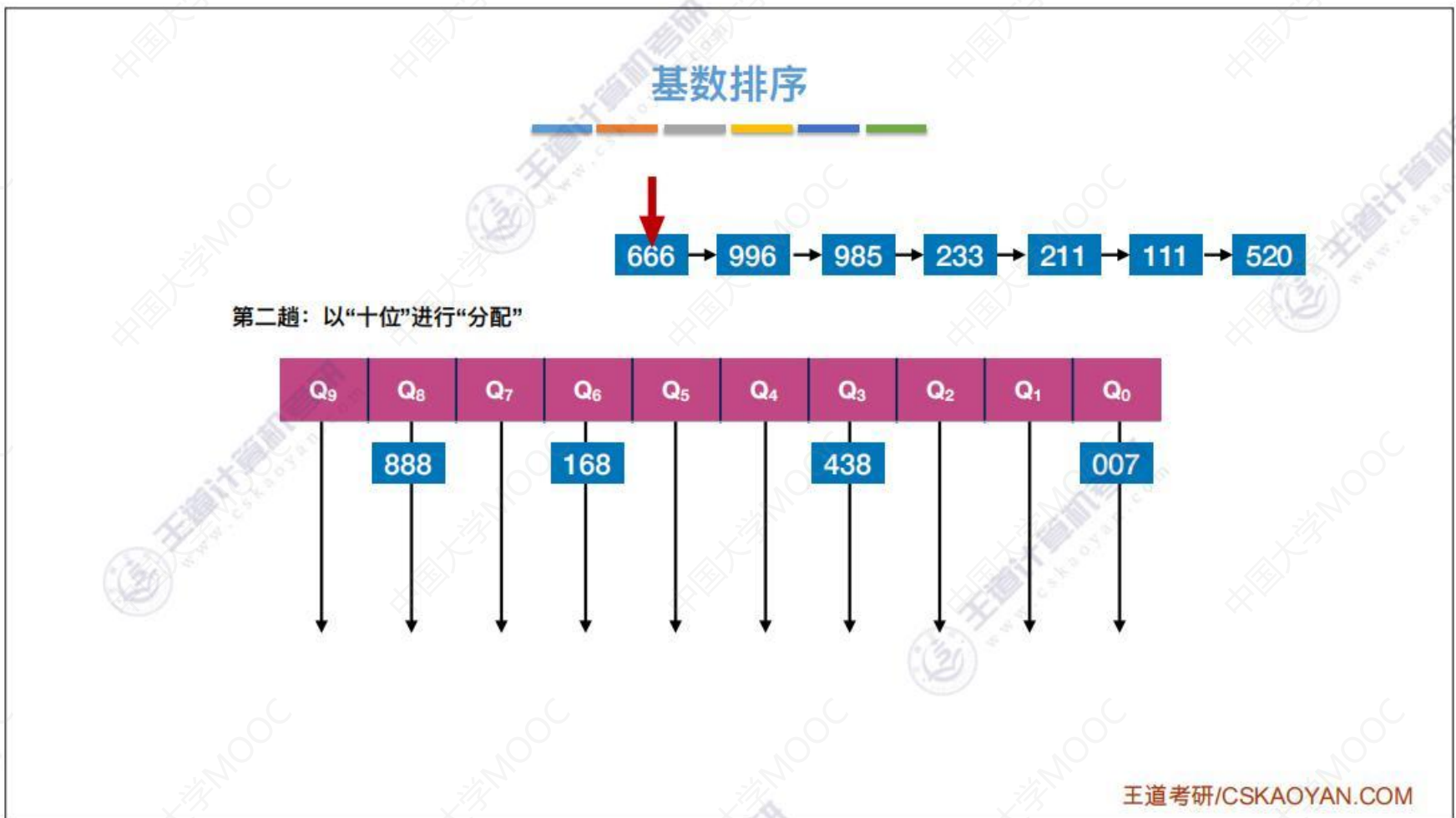


基数排序



第二趟：以“十位”进行“分配”

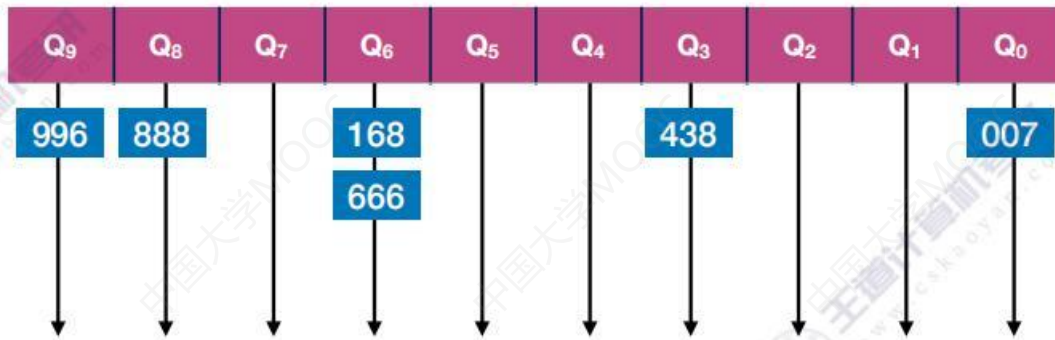




基数排序



第二趟：以“十位”进行“分配”

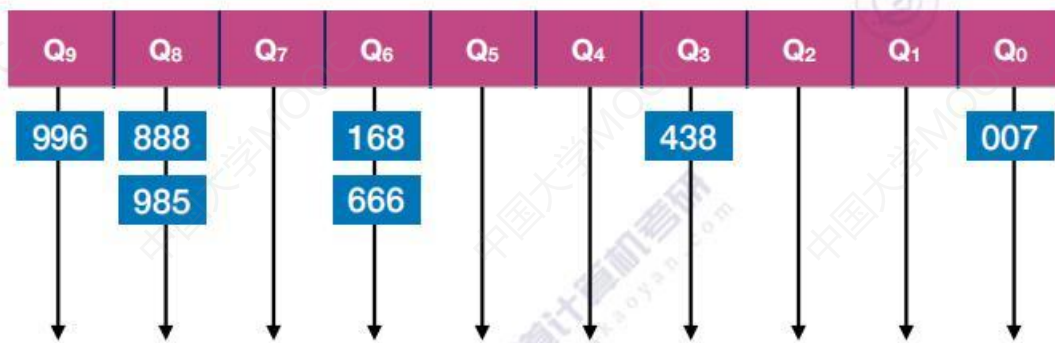


王道考研/CSKAOYAN.COM

基数排序

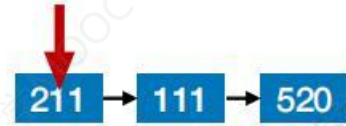


第二趟：以“十位”进行“分配”

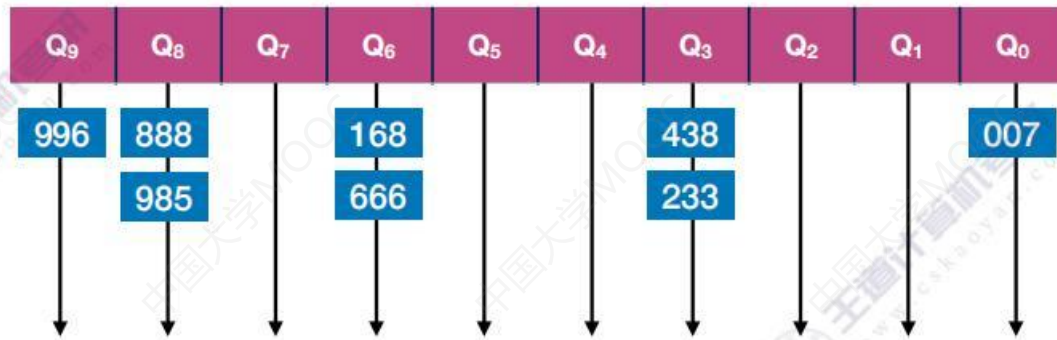


王道考研/CSKAOYAN.COM

基数排序



第二趟：以“十位”进行“分配”

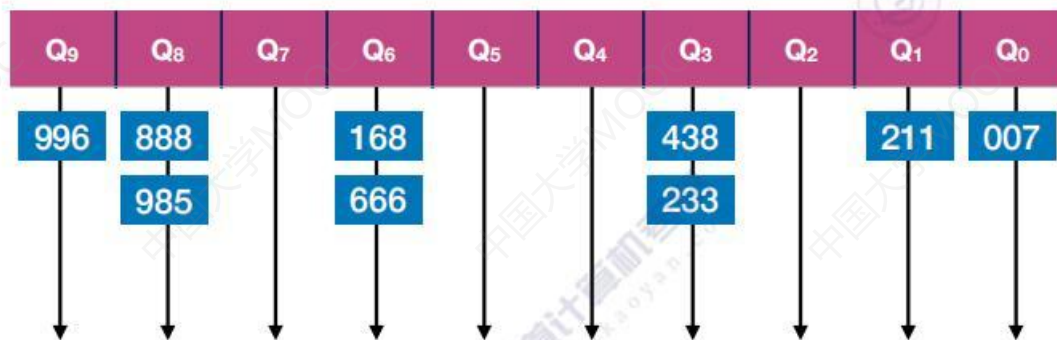


王道考研/CSKAOYAN.COM

基数排序



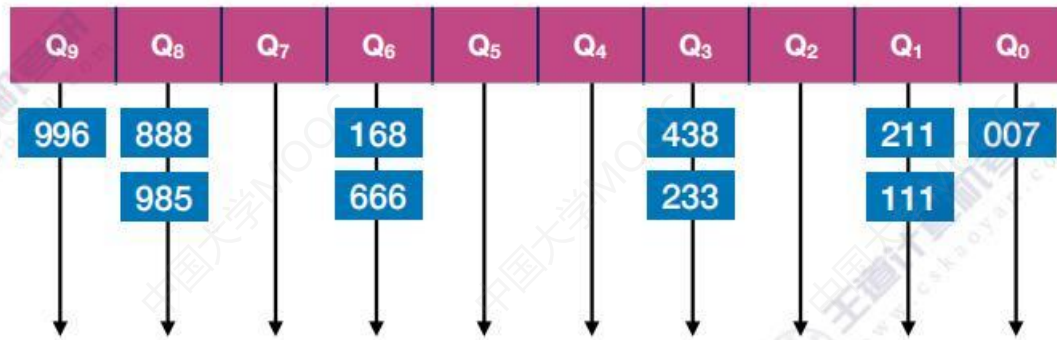
第二趟：以“十位”进行“分配”



王道考研/CSKAOYAN.COM

基数排序

第二趟：以“十位”进行“分配”

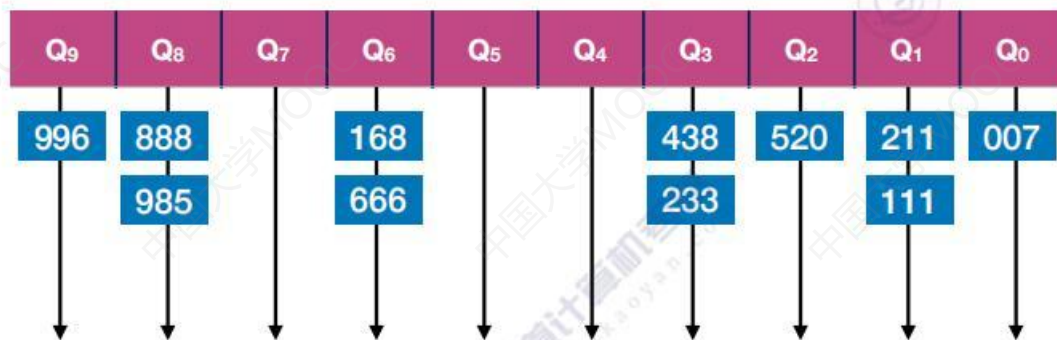


520

王道考研/CSKAOYAN.COM

基数排序

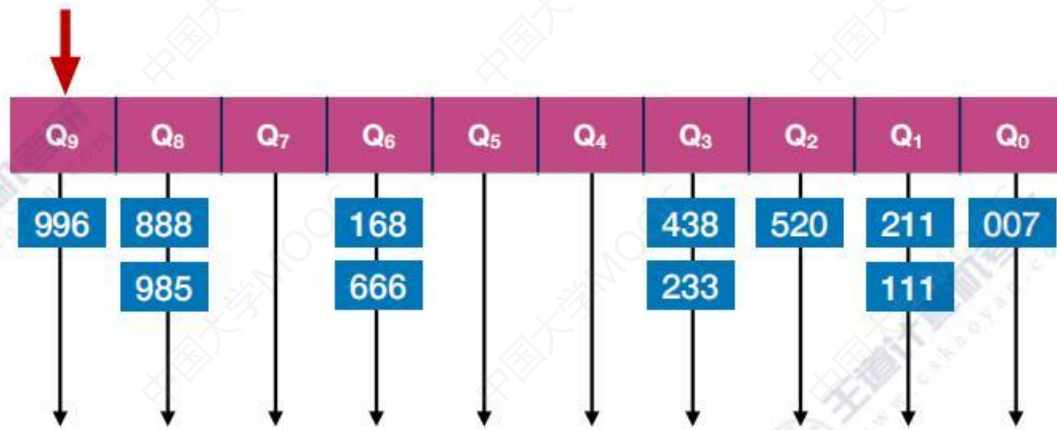
第二趟“分配”结束



王道考研/CSKAOYAN.COM

基数排序

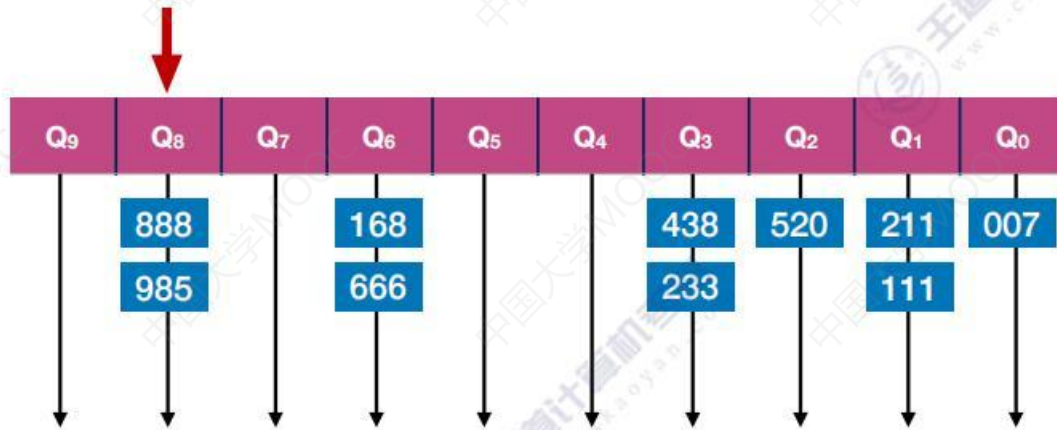
第二趟“收集”:



王道考研/CSKAOYAN.COM

基数排序

第二趟“收集”:

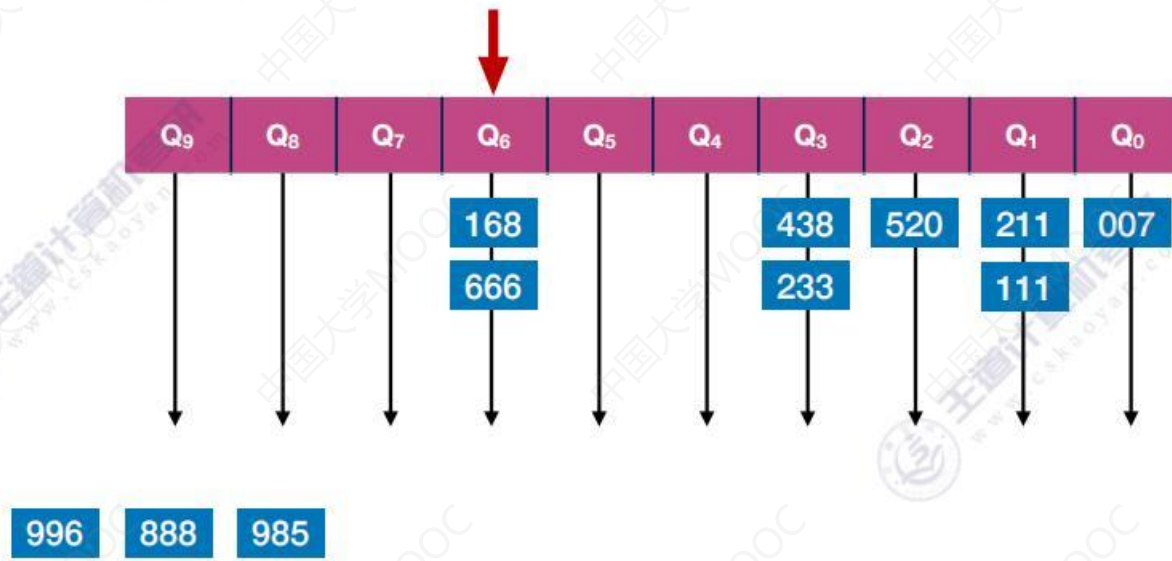


996

王道考研/CSKAOYAN.COM

基数排序

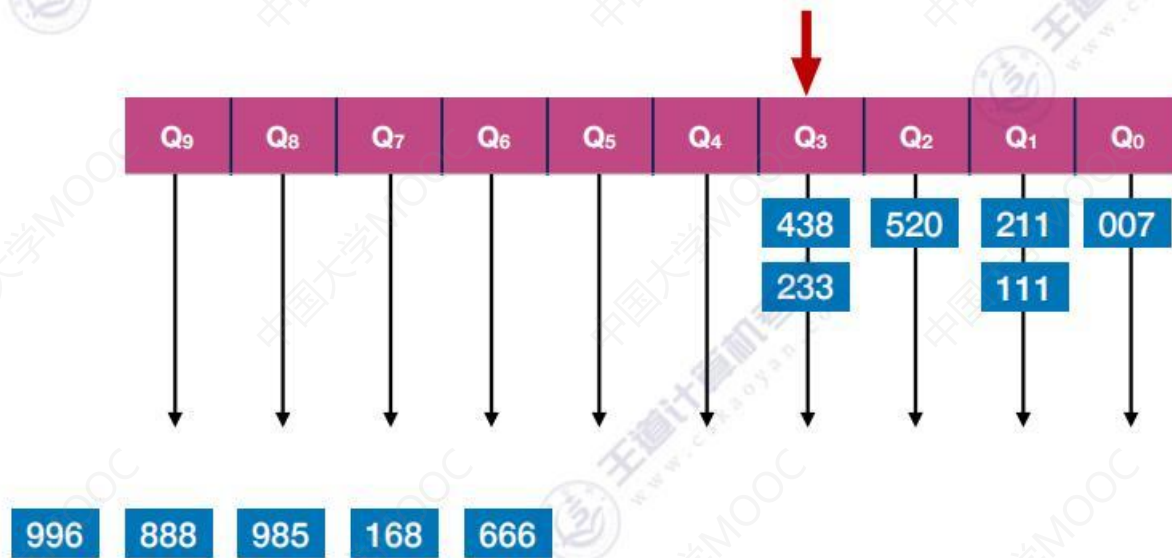
第二趟“收集”:



王道考研/CSKAOYAN.COM

基数排序

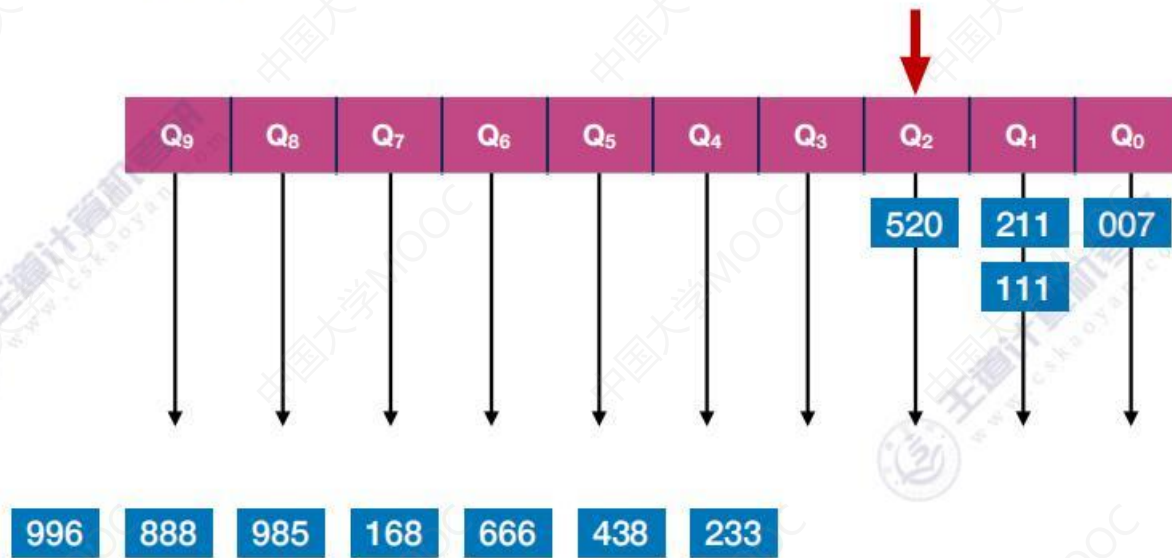
第二趟“收集”:



王道考研/CSKAOYAN.COM

基数排序

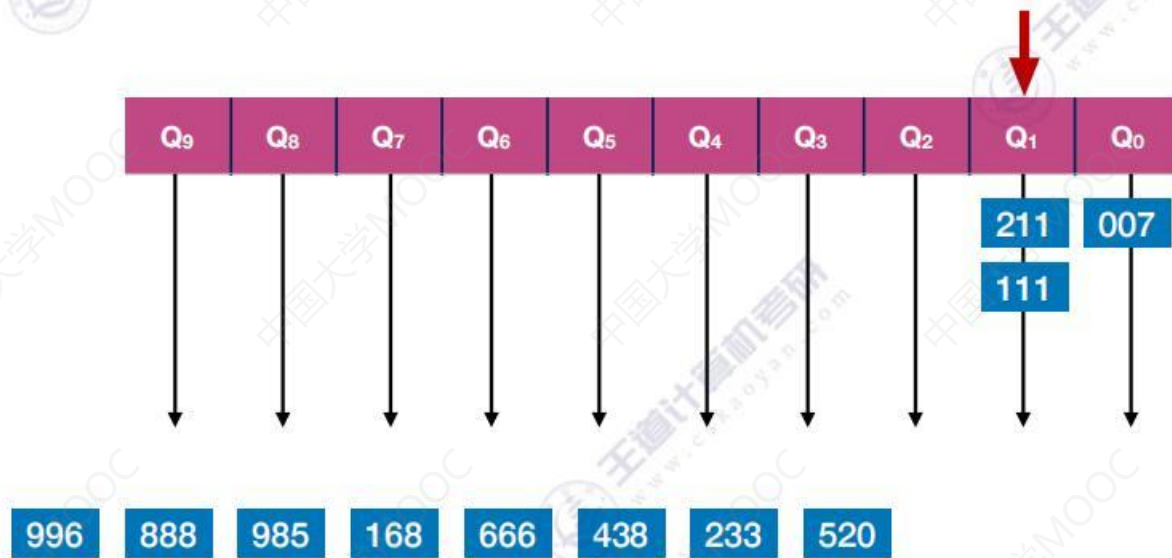
第二趟“收集”:



王道考研/CSKAOYAN.COM

基数排序

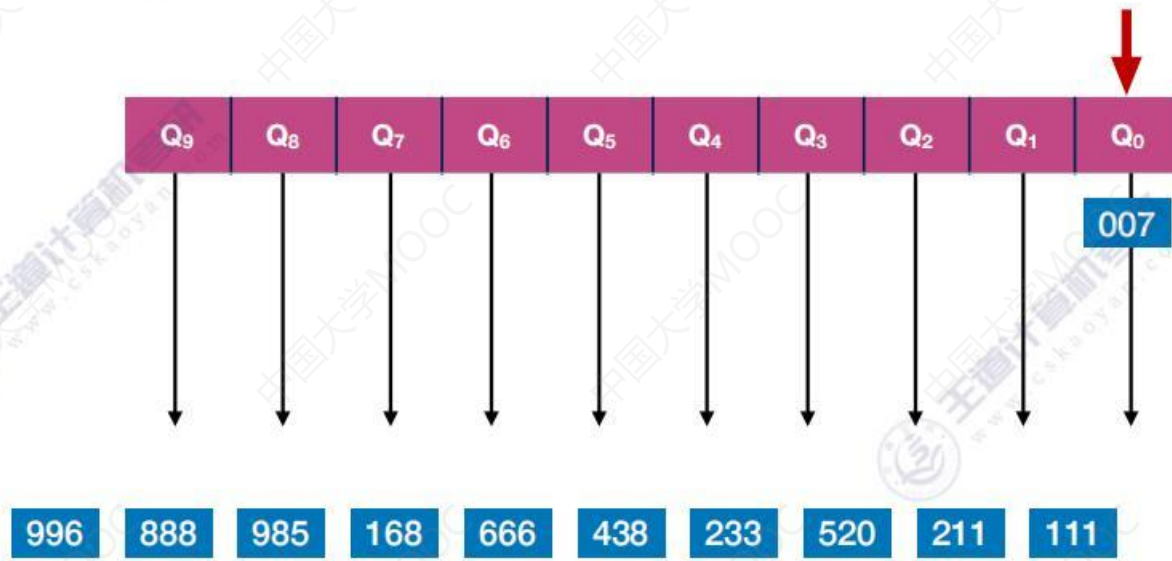
第二趟“收集”:



王道考研/CSKAOYAN.COM

基数排序

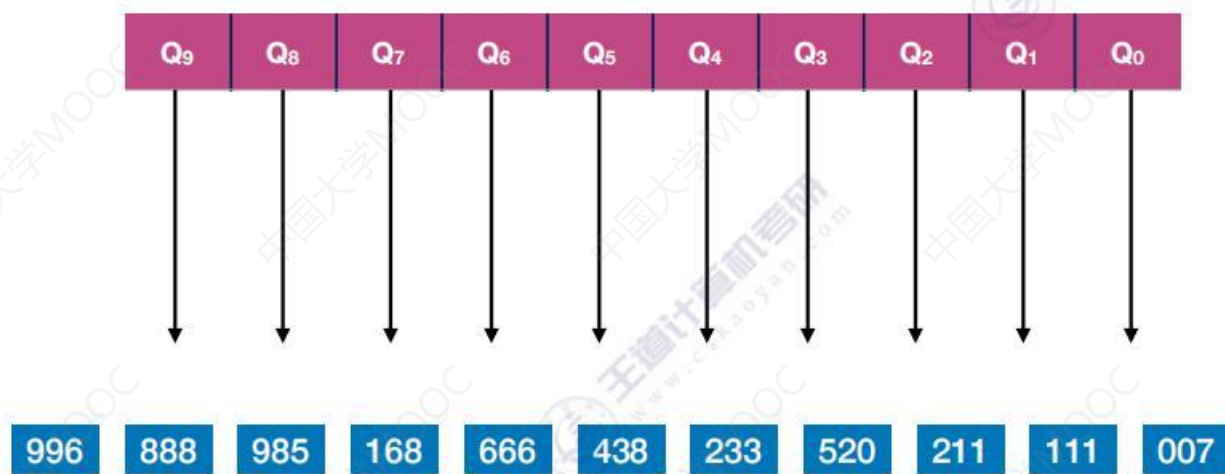
第二趟“收集”:



王道考研/CSKAOYAN.COM

基数排序

第二趟“收集”:



王道考研/CSKAOYAN.COM

基数排序

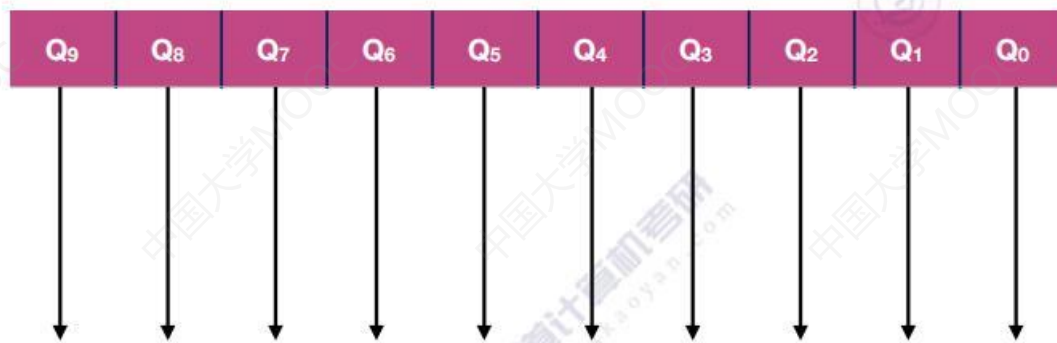


王道考研/CSKAOYAN.COM

基数排序



第三趟：以“百位”进行“分配”

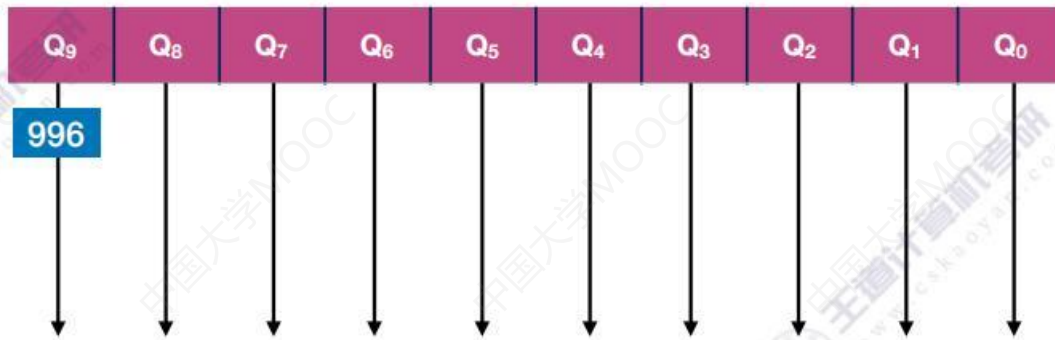


王道考研/CSKAOYAN.COM

基数排序



第三趟：以“百位”进行“分配”

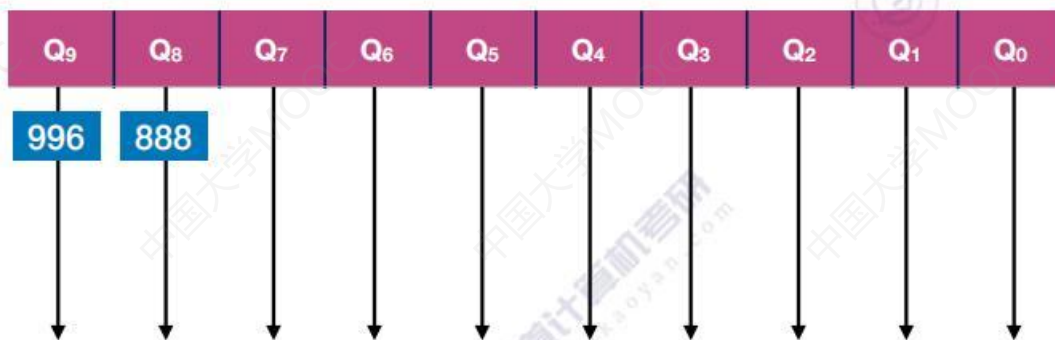


王道考研/CSKAOYAN.COM

基数排序

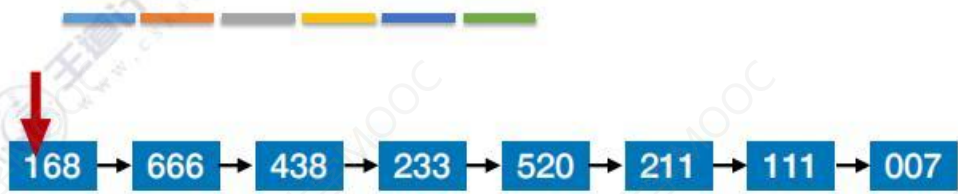


第三趟：以“百位”进行“分配”

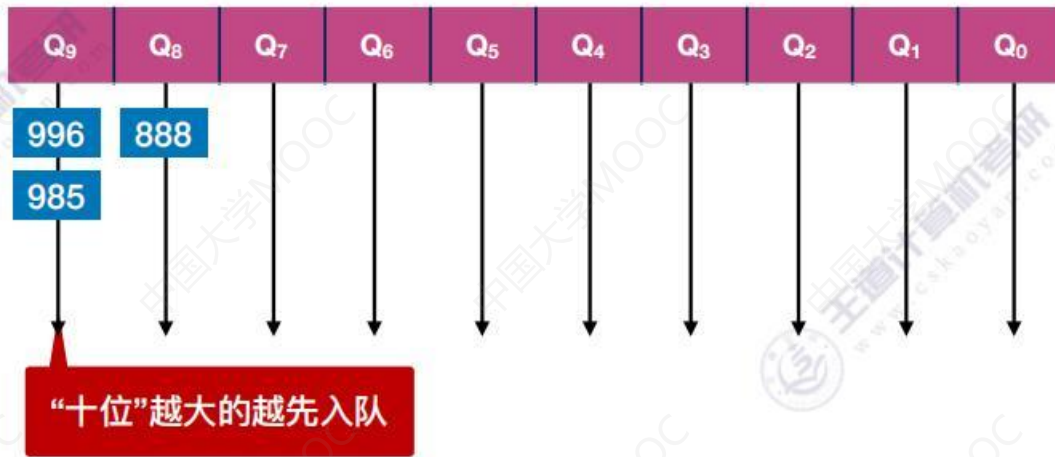


王道考研/CSKAOYAN.COM

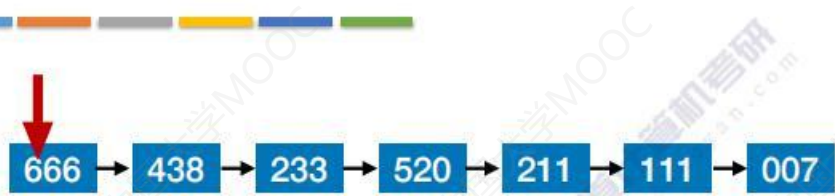
基数排序



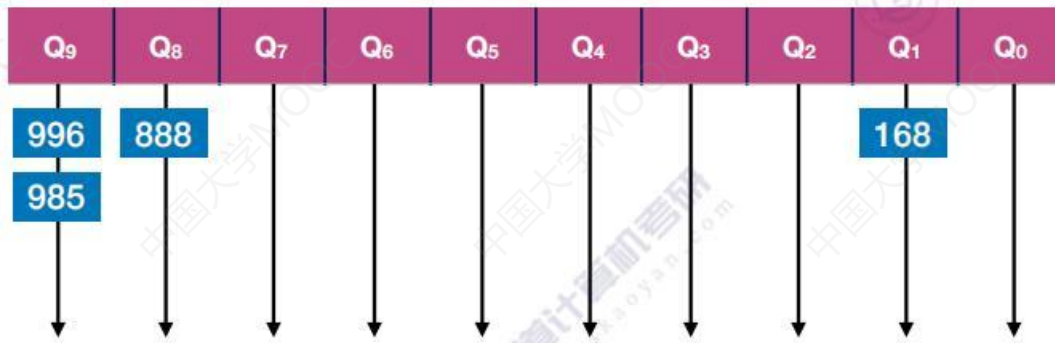
第三趟：以“百位”进行“分配”



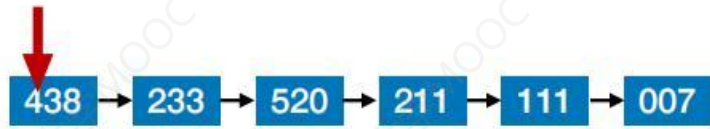
基数排序



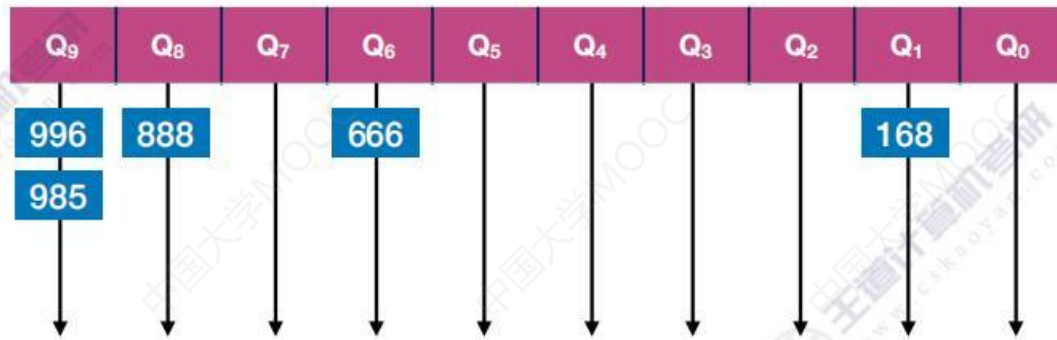
第三趟：以“百位”进行“分配”



基数排序



第三趟：以“百位”进行“分配”

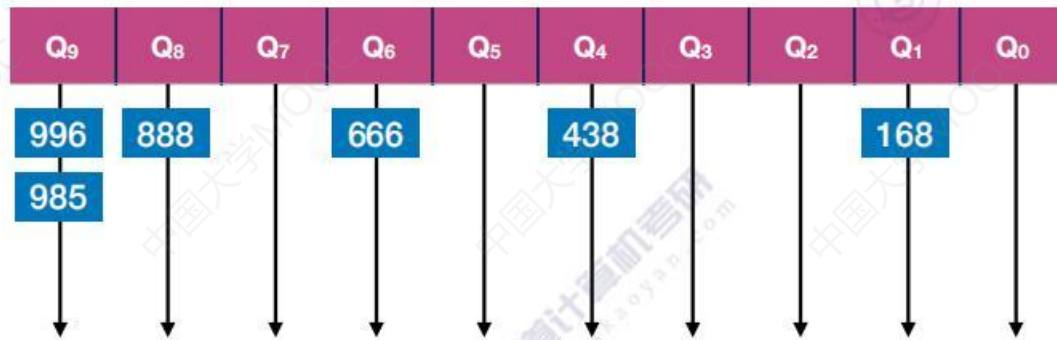


王道考研/CSKAOYAN.COM

基数排序



第三趟：以“百位”进行“分配”

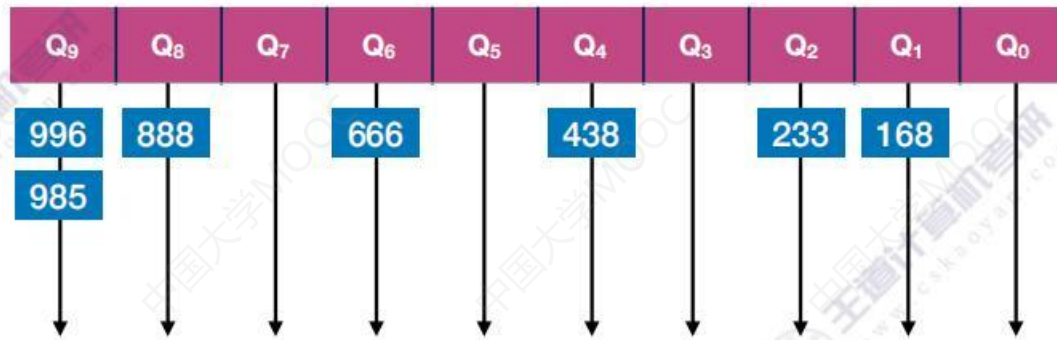


王道考研/CSKAOYAN.COM

基数排序



第三趟：以“百位”进行“分配”

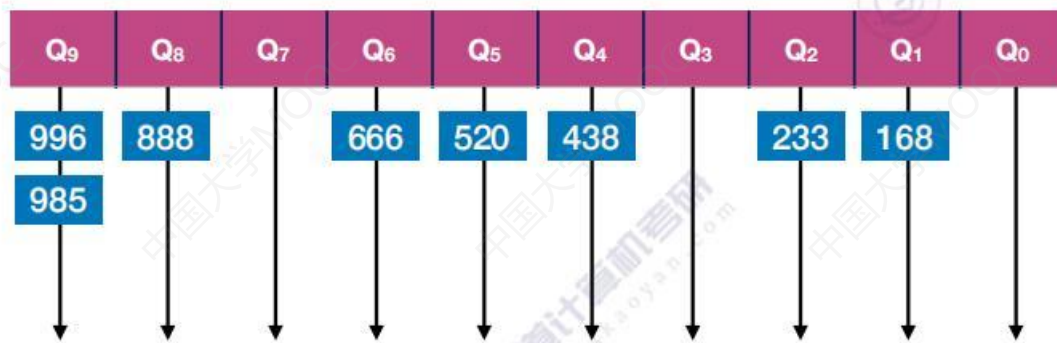


王道考研/CSKAOYAN.COM

基数排序

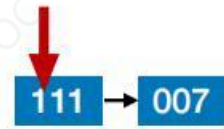


第三趟：以“百位”进行“分配”

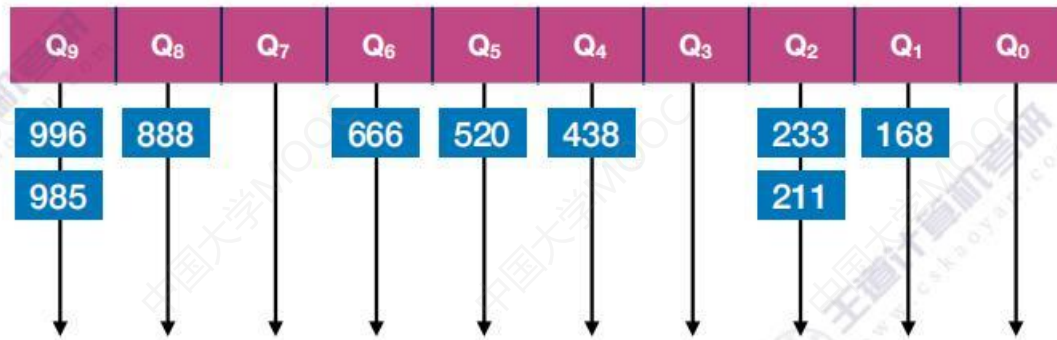


王道考研/CSKAOYAN.COM

基数排序



第三趟：以“百位”进行“分配”

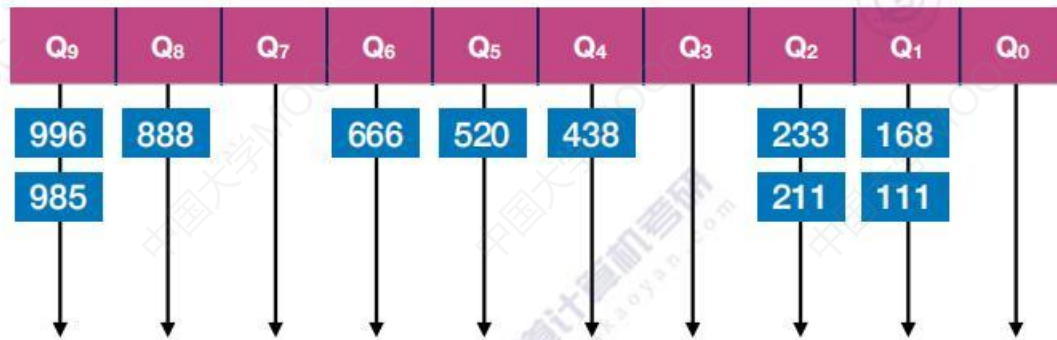


王道考研/CSKAOYAN.COM

基数排序



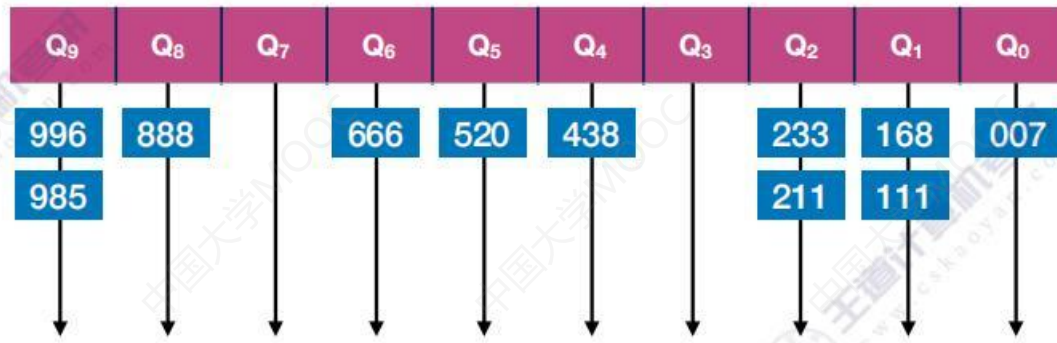
第三趟：以“百位”进行“分配”



王道考研/CSKAOYAN.COM

基数排序

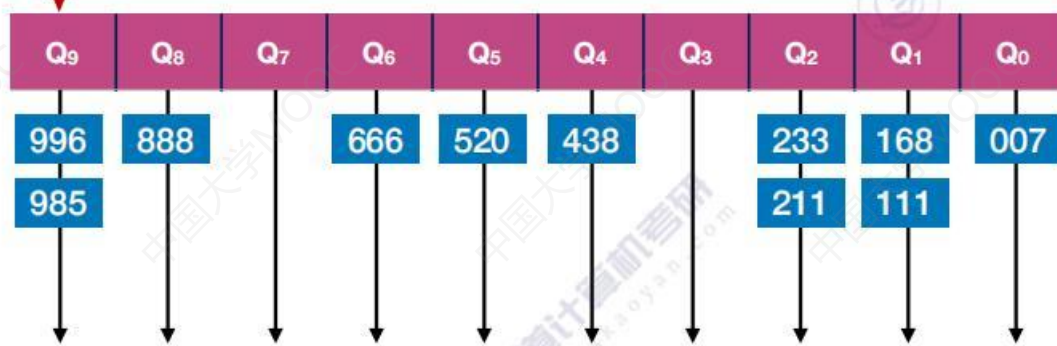
第三趟：以“百位”进行“分配”



王道考研/CSKAOYAN.COM

基数排序

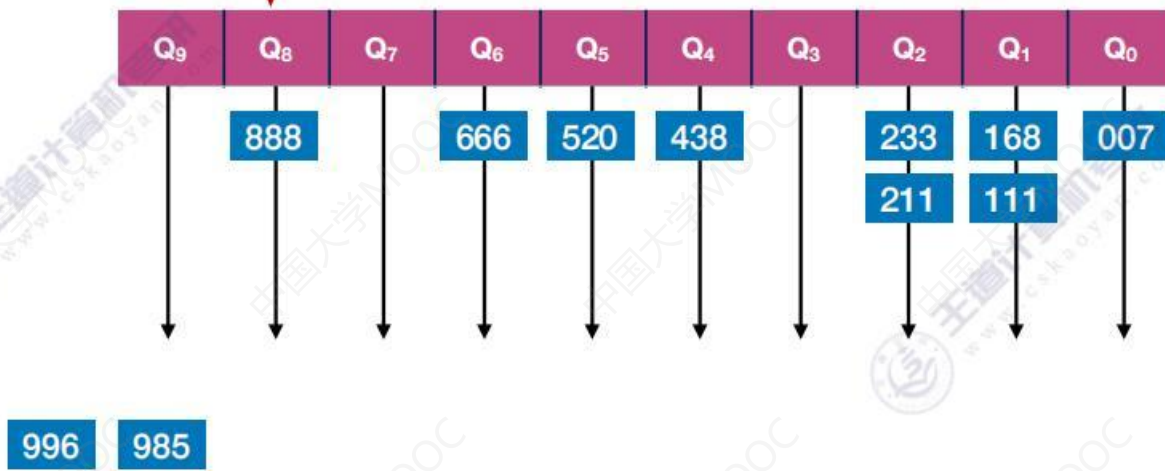
第三趟“收集”



王道考研/CSKAOYAN.COM

基数排序

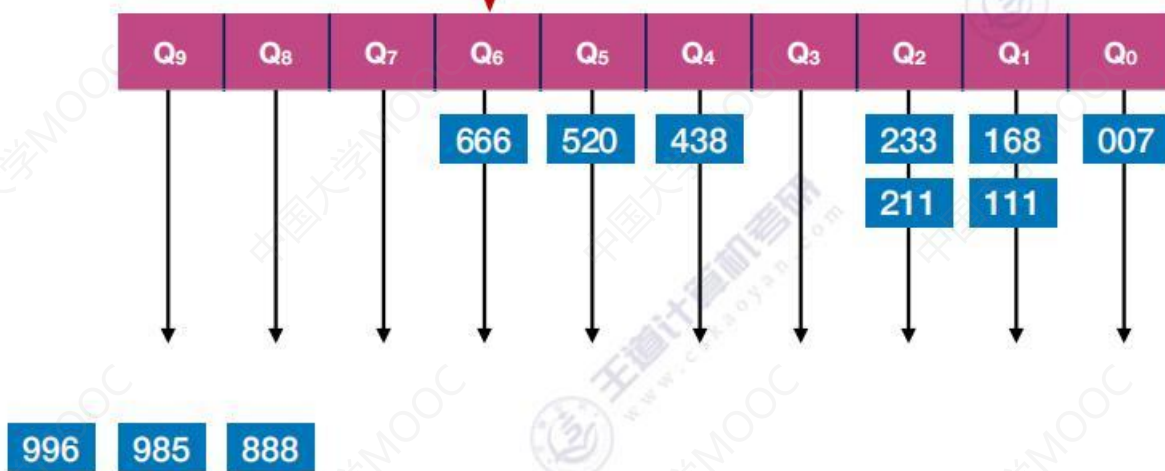
第三趟“收集”



王道考研/CSKAOYAN.COM

基数排序

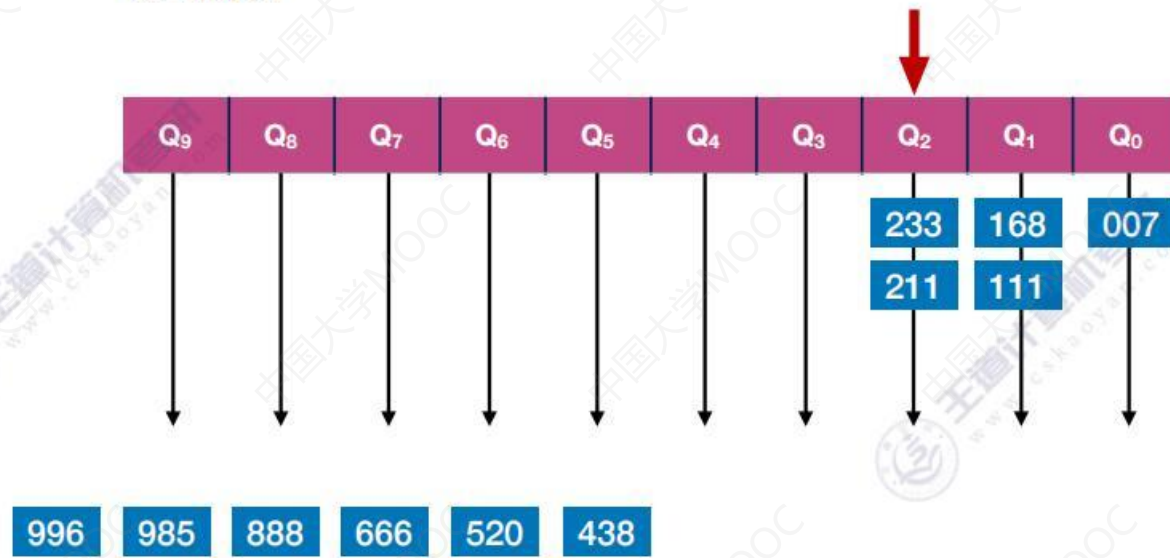
第三趟“收集”



王道考研/CSKAOYAN.COM

基数排序

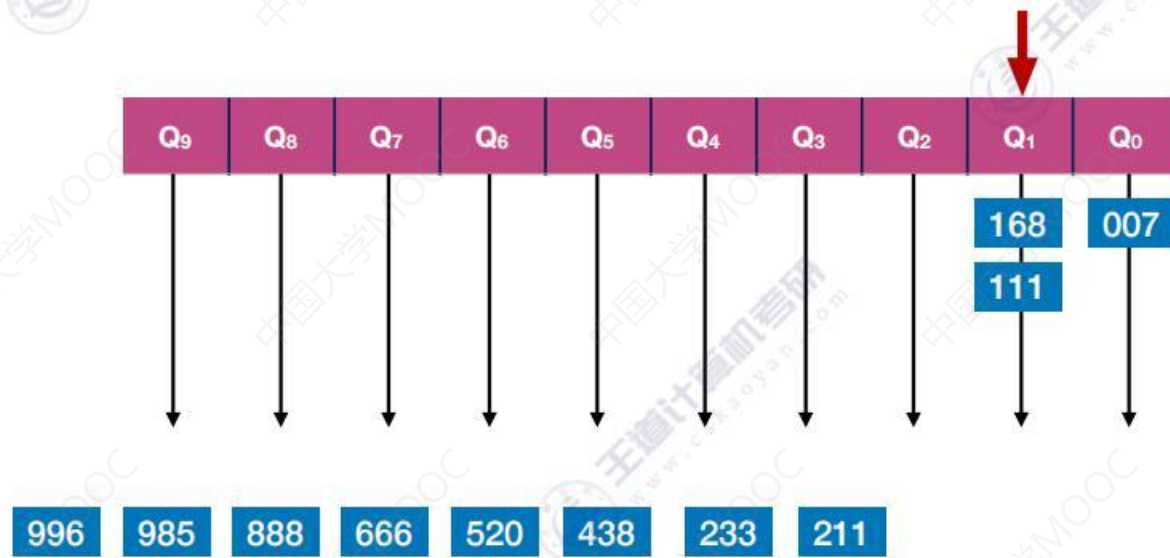
第三趟“收集”



王道考研/CSKAOYAN.COM

基数排序

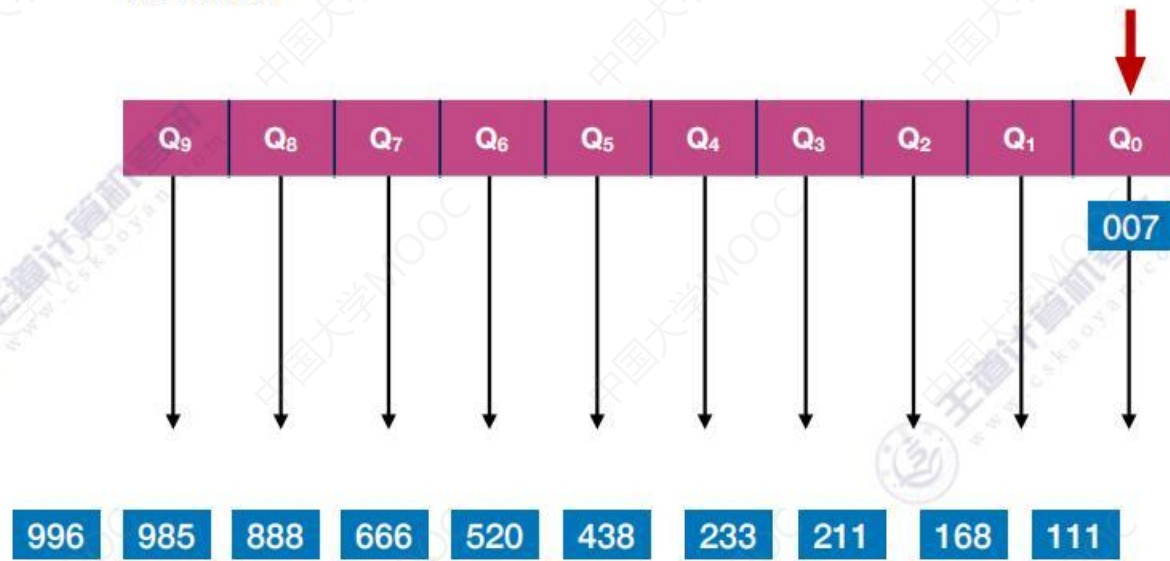
第三趟“收集”



王道考研/CSKAOYAN.COM

基数排序

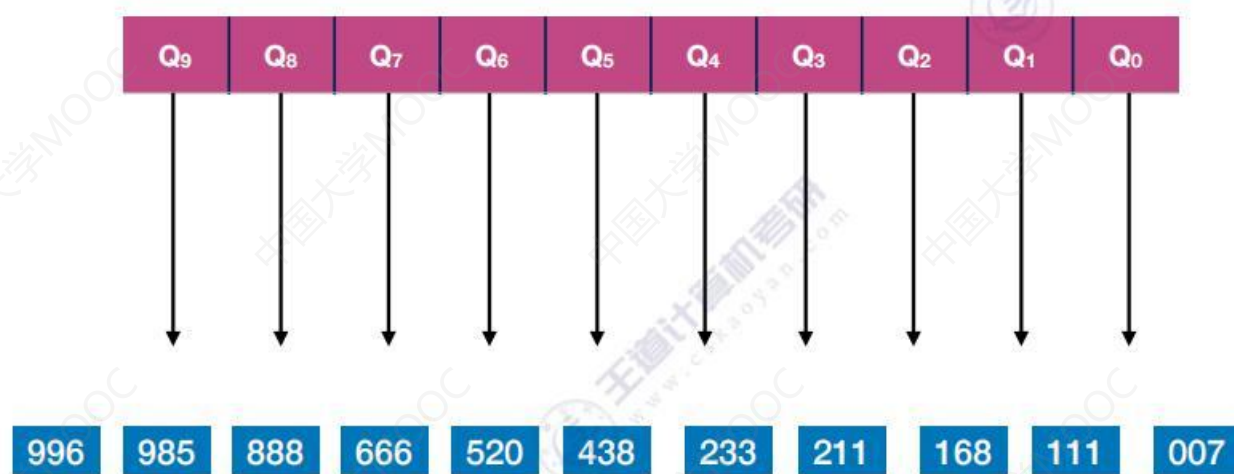
第三趟“收集”



王道考研/CSKAOYAN.COM

基数排序

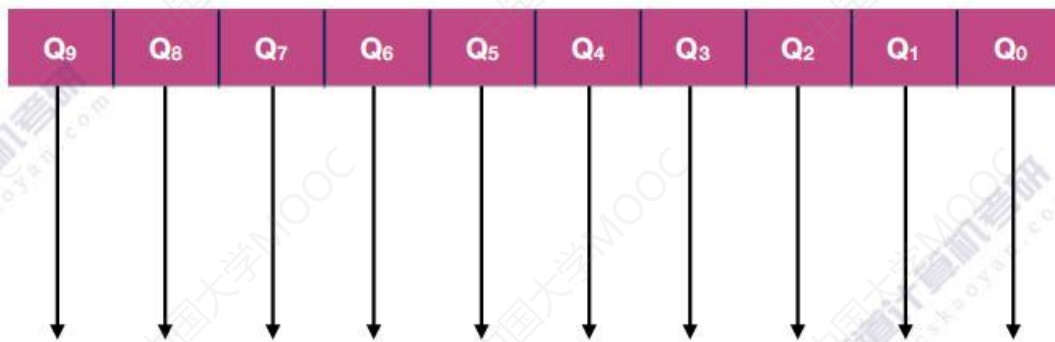
第三趟“收集”



王道考研/CSKAOYAN.COM

基数排序

第三趟“收集”



第三趟按“百位”分配、收集：得到一个按“百位”递减排列的序列，若“百位”相同则按“十位”递减排列，若“十位”还相同则按“个位”递减排列

996 → 985 → 888 → 666 → 520 → 438 → 233 → 211 → 168 → 111 → 007

王道考研/CSKAOYAN.COM

基数排序

初始序列:

520 → 211 → 438 → 888 → 007 → 111 → 985 → 666 → 996 → 233 → 168

第一趟按“个位”分配、收集：得到按“个位”递减排序的序列

438 → 888 → 168 → 007 → 666 → 996 → 985 → 233 → 211 → 111 → 520

第二趟按“十位”分配、收集：得到按“十位”递减排序的序列，“十位”相同的按“个位”递减排序

996 → 888 → 985 → 168 → 666 → 438 → 233 → 520 → 211 → 111 → 007

第三趟按“百位”分配、收集：得到一个按“百位”递减排列的序列，若“百位”相同则按“十位”递减排列，若“十位”还相同则按“个位”递减排列

996 → 985 → 888 → 666 → 520 → 438 → 233 → 211 → 168 → 111 → 007

王道考研/CSKAOYAN.COM

基数排序

初始序列:

520 → 211 → 438 → 888 → 007 → 111 → 985 → 666 → 996 → 233 → 168

最高位关键字
(最主位关键字)

最低位关键字
(最次位关键字)

假设长度为 n 的线性表中每个结点 a_j 的关键字由 d 元组 $(k_j^{d-1}, k_j^{d-2}, k_j^{d-3}, \dots, k_j^1, k_j^0)$ 组成

其中, $0 \leq k_j^i \leq r - 1$ ($0 \leq j < n, 0 \leq i < d - 1$), r 称为“基数”

基数排序得到**递减**序列的过程如下,

初始化: 设置 r 个空队列, $Q_{r-1}, Q_{r-2}, \dots, Q_0$

按照各个 **关键字位 权重递增的次序** (个、十、百), 对 d 个关键字位分别做“分配”和“收集”

分配: 顺序扫描各个元素, 若当前处理的关键字位= x , 则将元素插入 Q_x 队尾

收集: 把 $Q_{r-1}, Q_{r-2}, \dots, Q_0$ 各个队列中的结点依次出队并链接

基数排序不是基于
“比较”的排序算法

王道考研/CSKAOYAN.COM

基数排序

初始序列:

520 → 211 → 438 → 888 → 007 → 111 → 985 → 666 → 996 → 233 → 168

最高位关键字
(最主位关键字)

最低位关键字
(最次位关键字)

假设长度为 n 的线性表中每个结点 a_j 的关键字由 d 元组 $(k_j^{d-1}, k_j^{d-2}, k_j^{d-3}, \dots, k_j^1, k_j^0)$ 组成

其中, $0 \leq k_j^i \leq r - 1$ ($0 \leq j < n, 0 \leq i < d - 1$), r 称为“基数”

基数排序得到**递增**序列的过程如下,

初始化: 设置 r 个空队列, Q_0, Q_1, \dots, Q_{r-1}

按照各个 **关键字位 权重递增的次序** (个、十、百), 对 d 个关键字位分别做“分配”和“收集”

分配: 顺序扫描各个元素, 若当前处理的关键字位= x , 则将元素插入 Q_x 队尾

收集: 把 Q_0, Q_1, \dots, Q_{r-1} 各个队列中的结点依次出队并链接

基数排序不是基于
“比较”的排序算法

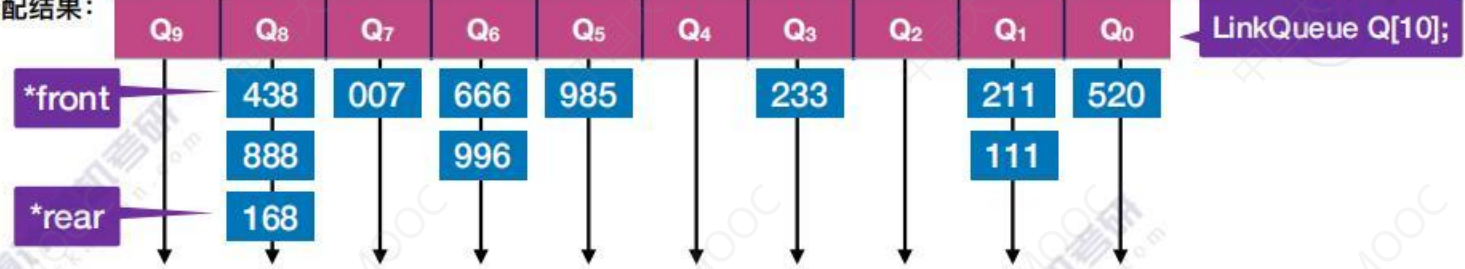
王道考研/CSKAOYAN.COM

算法效率分析

初始序列:



第一趟分配结果:



基数排序通常基于链式存储实现

```
typedef struct LinkNode{
    ElemType data;
    struct LinkNode *next;
}LinkNode, *LinkList;
```

```
typedef struct{
    LinkNode *front,*rear; //链式队列 //队列的队头和队尾指针
}LinkQueue;
```

需要 r 个辅助队列, 空间复杂度 = $O(r)$

一趟分配 $O(n)$, 一趟收集 $O(r)$, 总共 d 趟分配、收集, 总的时间复杂度 = $O(d(n+r))$

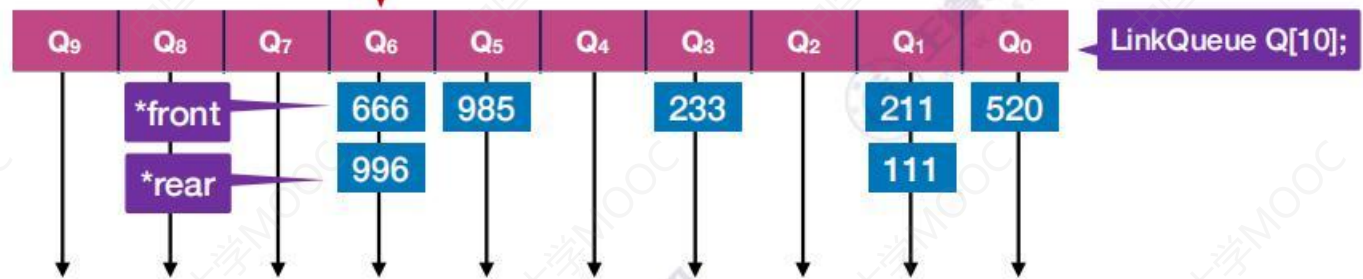
把关键字拆为 d 个部分

每个部分可能取得 r 个值

王道考研/CSKAOYAN.COM

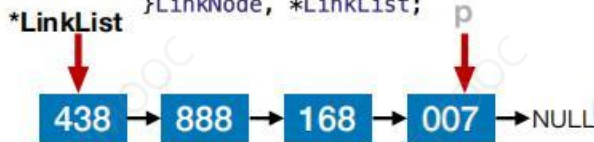
算法效率分析

“收集”各个队列内元素



```
typedef struct LinkNode{
    ElemType data;
    struct LinkNode *next;
}LinkNode, *LinkList;
```

```
typedef struct{
    LinkNode *front,*rear; //链式队列 //队列的队头和队尾指针
}LinkQueue;
```



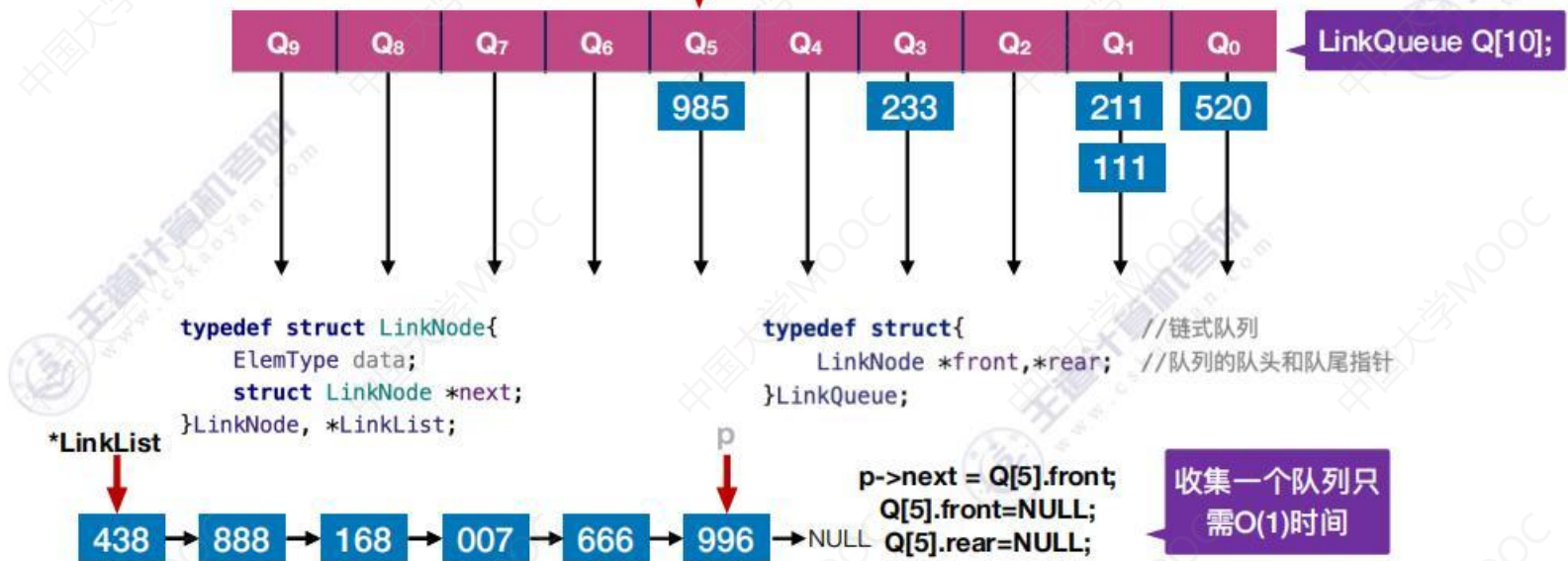
```
p->next = Q[6].front;
Q[6].front=NULL;
Q[6].rear=NULL;
```

收集一个队列只需 $O(1)$ 时间

王道考研/CSKAOYAN.COM

算法效率分析

“收集”各个队列内元素

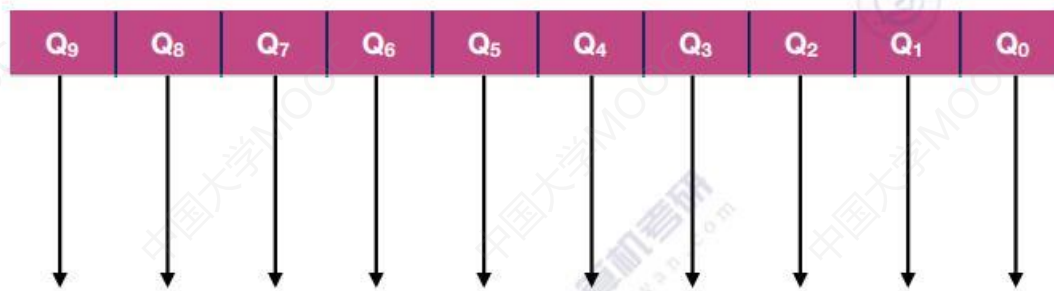


王道考研/CSKAOYAN.COM

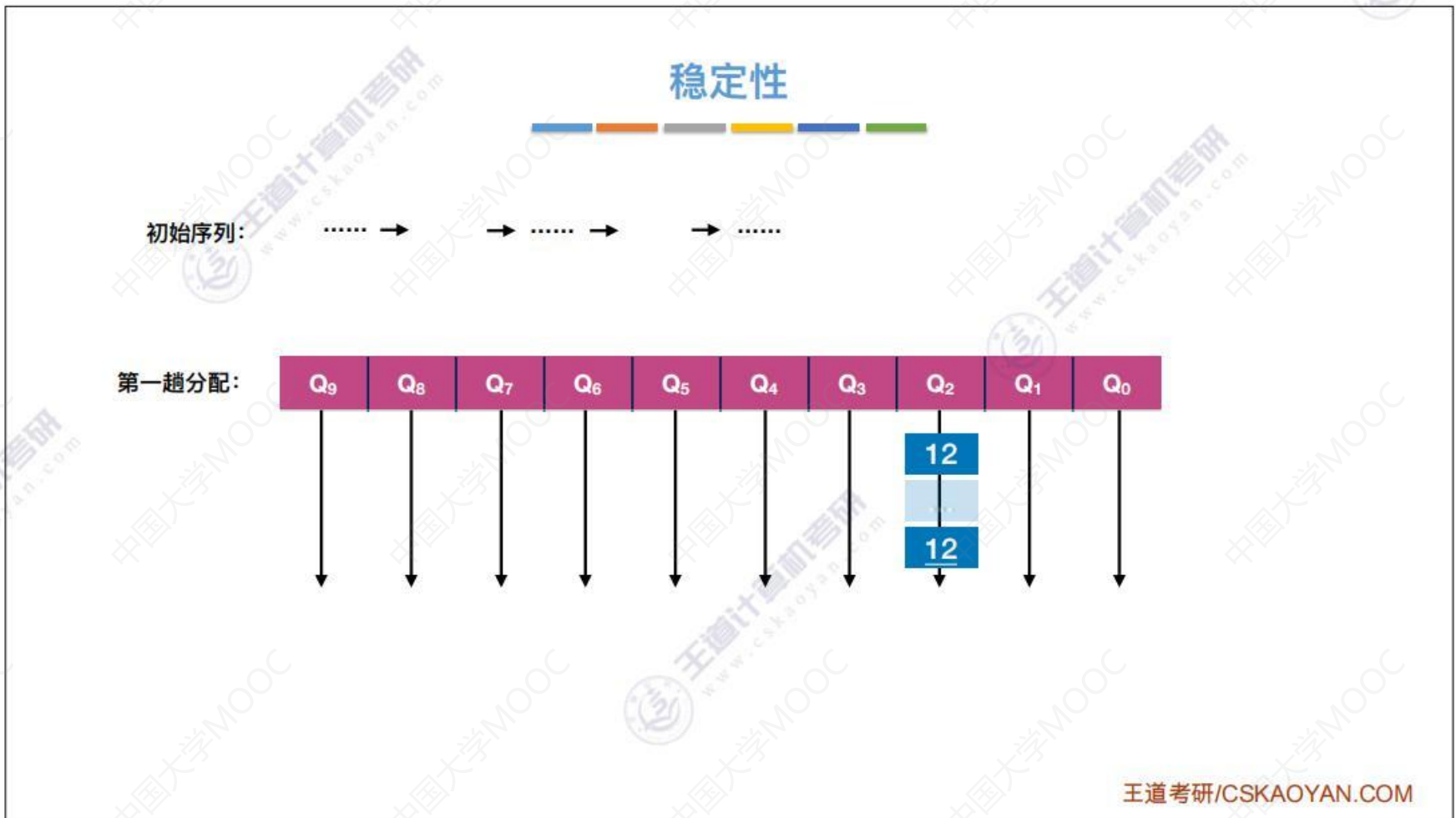
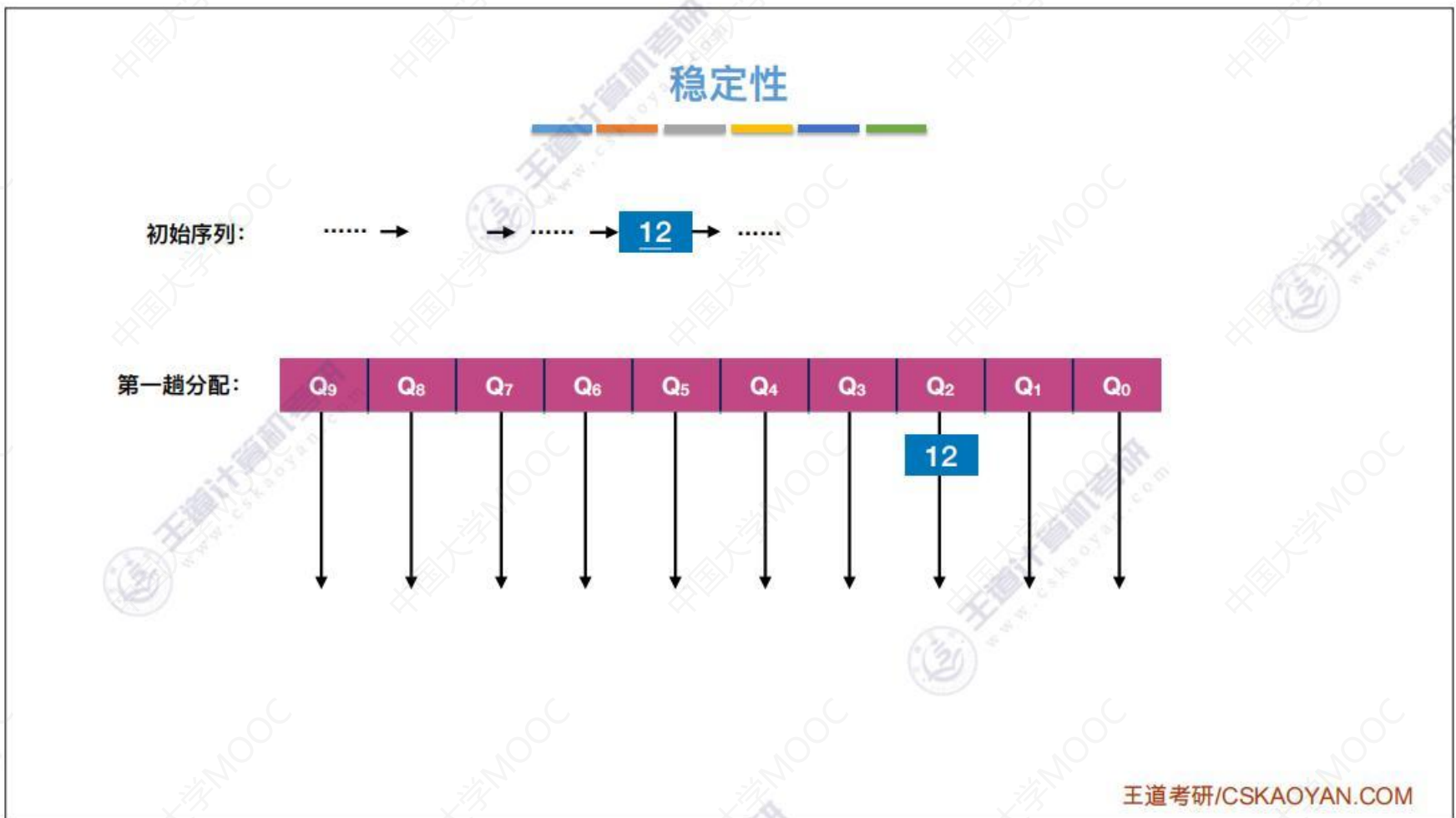
稳定性

初始序列: → 12 → → 12 →

第一趟分配:



王道考研/CSKAOYAN.COM

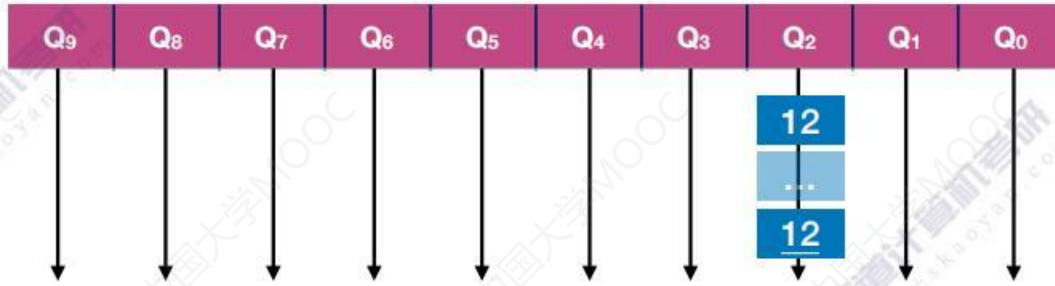


稳定性

初始序列:



第一趟分配:



第一趟收集:



基数排序是稳定的



王道考研/CSKAOYAN.COM

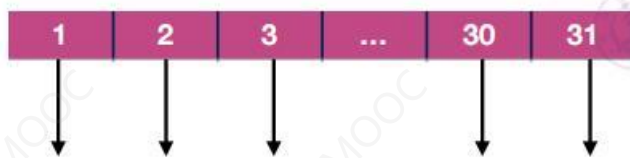
基数排序的应用

某学校有 10000 学生, 将学生信息按年龄递减排序

生日可拆分为三组关键字: 年(1991~2005)、月(1~12)、日(1~31)

权重: 年 > 月 > 日

第一趟分配、收集 (按“日”递增):



年、月、日越大, 年龄越小

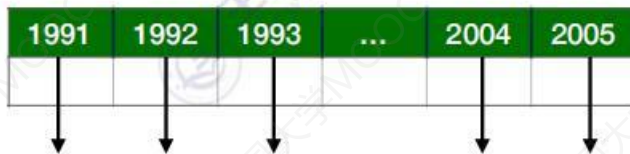
第二趟分配、收集 (按“月”递增):



基数排序, 时间复杂度 = $O(d(n+r))$
 $\approx O(30000)$

若采用 $O(n^2)$ 的排序, $\approx O(10^8)$
若采用 $O(n \log_2 n)$ 的排序, $\approx O(140000)$

第三趟分配、收集 (按“年”递增):



王道考研/CSKAOYAN.COM

基数排序的应用

某学校有 10000 学生，将学生信息按**年龄递减**排序

生日可拆分为三组关键字：年(1991~2005)、月(1~12)、日(1~31)

基数排序，时间复杂度 = $O(d(n+r))$
 $\approx O(30000)$

若采用 $O(n^2)$ 的排序， $\approx O(10^8)$
若采用 $O(n\log_2 n)$ 的排序， $\approx O(140000)$

基数排序擅长解决的问题：

- ①数据元素的关键字可以方便地拆分为 d 组，且 d 较小
- ②每组关键字的取值范围不大，即 r 较小
- ③数据元素个数 n 较大

王道考研/CSKAOYAN.COM

基数排序的应用

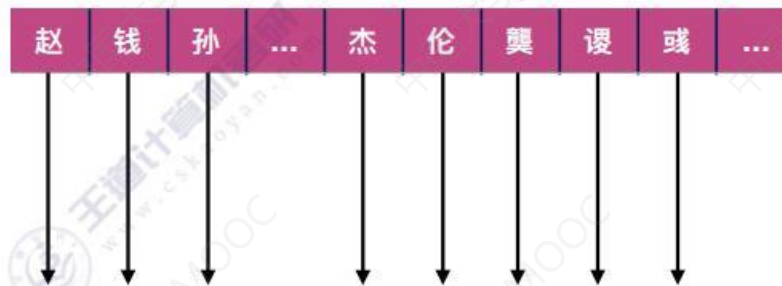
基数排序，时间复杂度 = $O(d(n+r))$

基数排序擅长解决的问题：

- ①数据元素的关键字可以方便地拆分为 d 组，且 d 较小 反例：给5个人的身份证号排序
- ②每组关键字的取值范围不大，即 r 较小 反例：给中文人名排序
- ③数据元素个数 n 较大 擅长：给十亿人的身份证号排序

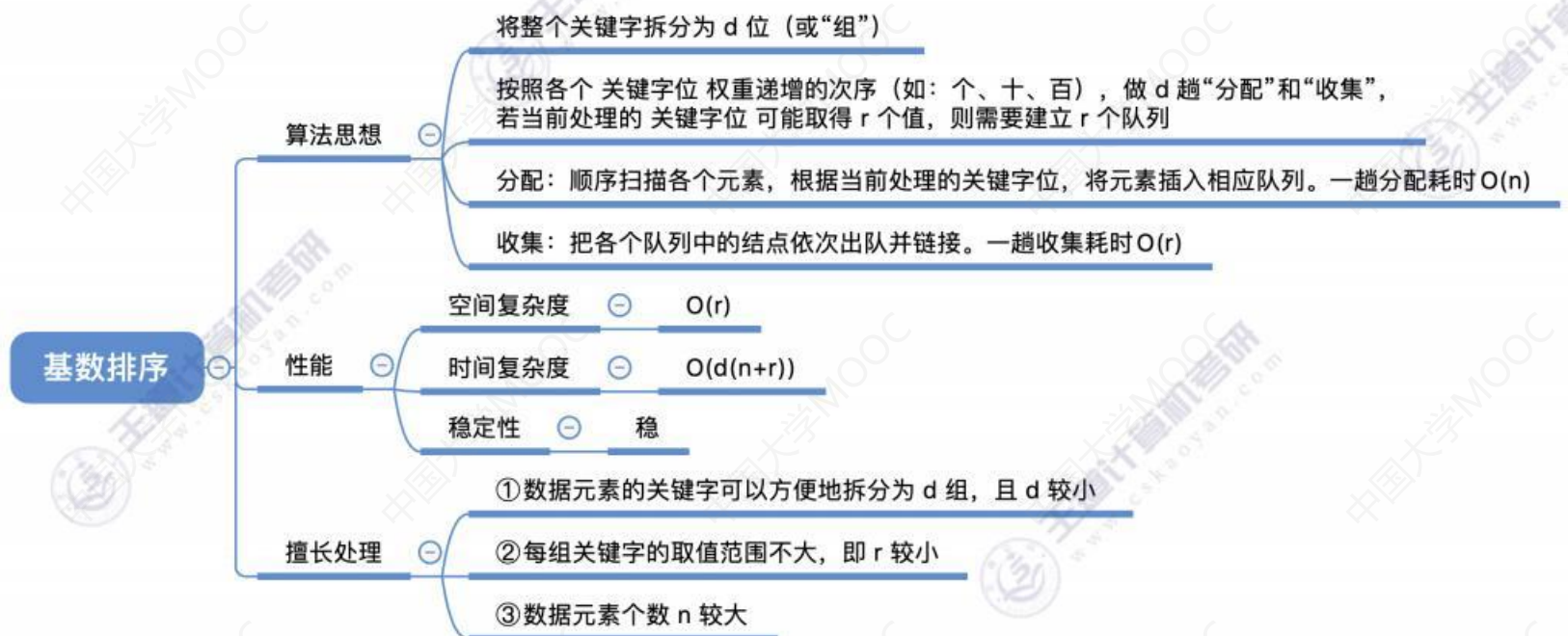
身份证号 XXXXXXXXXXXXXXXXXX

18位身份证号需要
分配、回收18趟



王道考研/CSKAOYAN.COM

知识回顾与重要考点



王道考研/CSKAOYAN.COM



@王道论坛



@王道计算机考研备考



@王道咸鱼老师-计算机考研

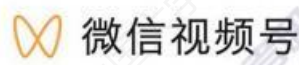
@王道楼楼老师-计算机考研



@王道计算机考研



@王道计算机考研



@王道计算机考研



微信公众平台

@王道在线