开课吧

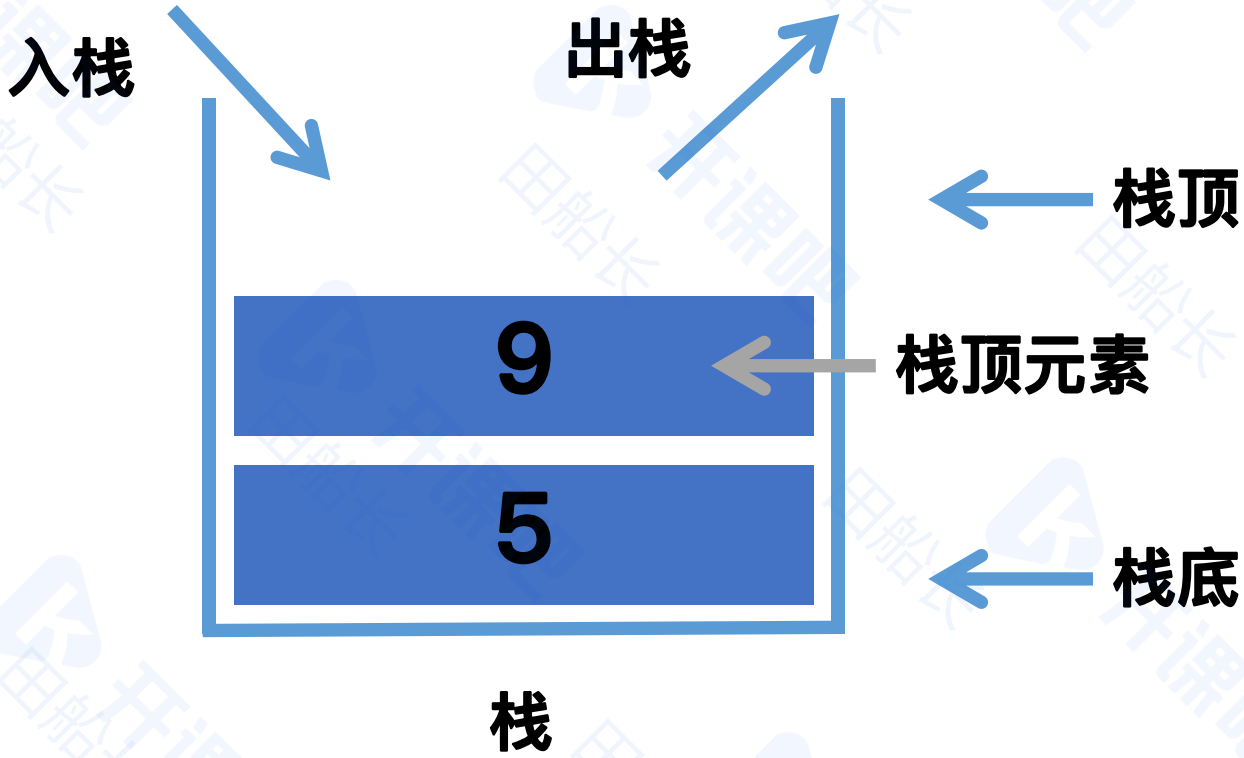
栈与队列

田船长

栈

栈（Stack）是一种只允许在一端进行插入或删除的线性表，它是一种先进后出（FILO）的数据结构

在栈不为空时，我们只能看到栈顶元素



栈的结构定义

```
#define MaxCnt 50 //栈当中元素的最大个数
struct Stack { //栈的定义
    EleType data[MaxCnt]; //栈的数据域
    int top; //栈顶元素位置
};
```

与顺序栈相对应的，还有一种“链栈”，顾名思义，是使用类似于链表的结构实现的

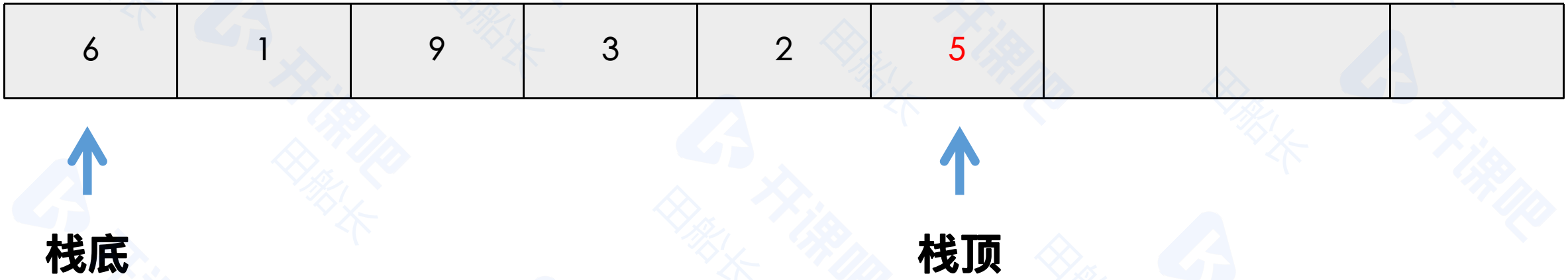
还有一种“共享栈”（也称为对顶栈）：两个栈共享同一片内存空间

入栈操作

size = 5

将元素5入栈

入栈操作



size = 6

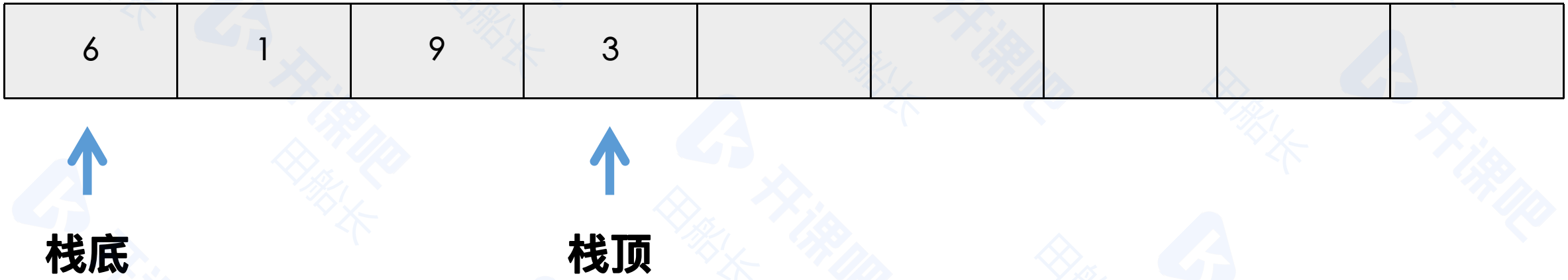
直接入栈即可

出栈操作

size = 5

将栈顶元素出栈

出栈操作



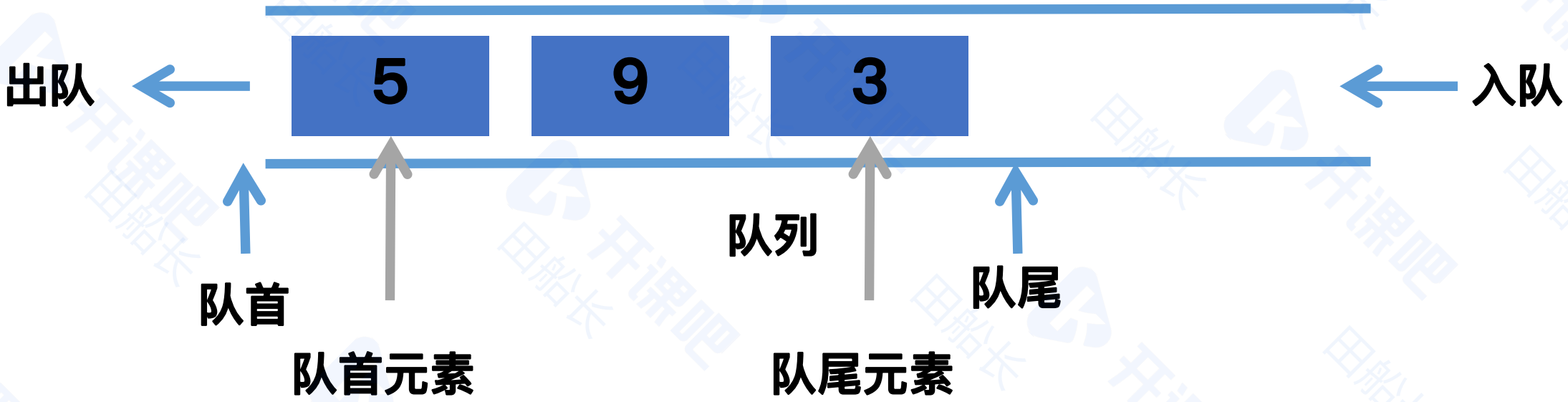
size = 4

直接出栈即可

队列

队列（Queue）是一种只允许在一端进行插入，在另一端进行删除的线性表，它是一种先进先出（FIFO）的数据结构

在队列不为空时，（一般的）我们只能看到队首元素

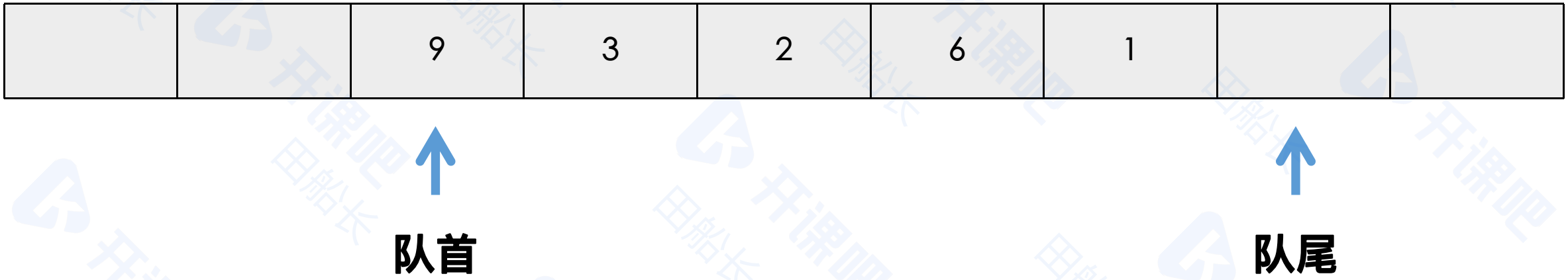


队列的结构定义

```
#define MaxCnt 50 //队列当中元素的最大个数
struct Queue { //队列的定义
    EleType data[MaxCnt]; //队列的数据域
    int front, back; //队首元素位置与队尾元素位置
};
```

由于普通的队列在入队出队的过程中
指针指的位置可能超过数据范围上限
所以一般在实现时使用循环队列：
当指针指向上限时将指针移动到最开始
与栈类似，也可以用链表的结构实现链式队列

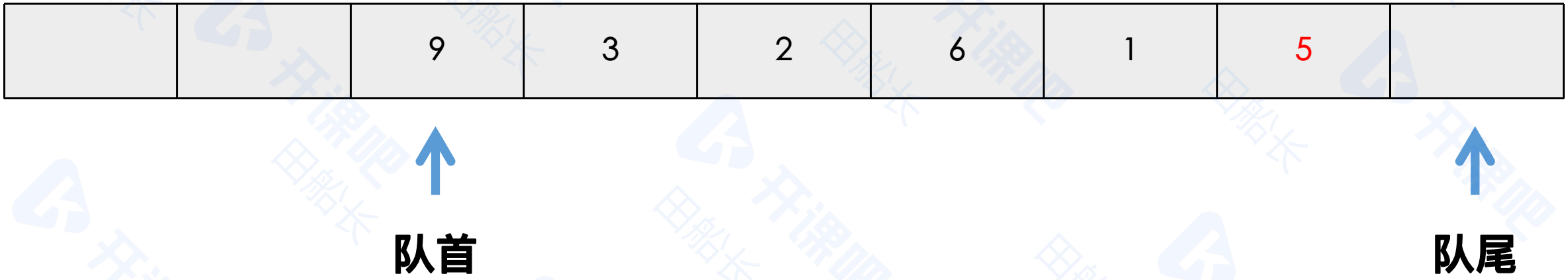
入队操作



size = 5

将元素5入队

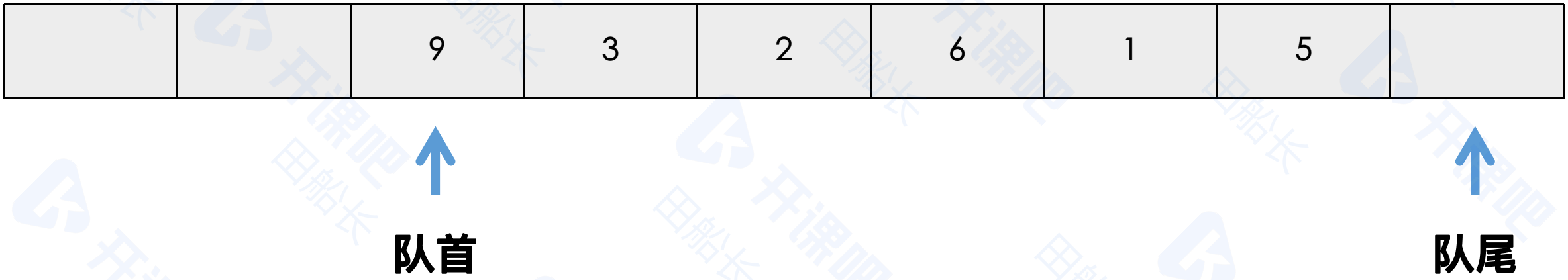
入队操作



size = 6

直接入队即可

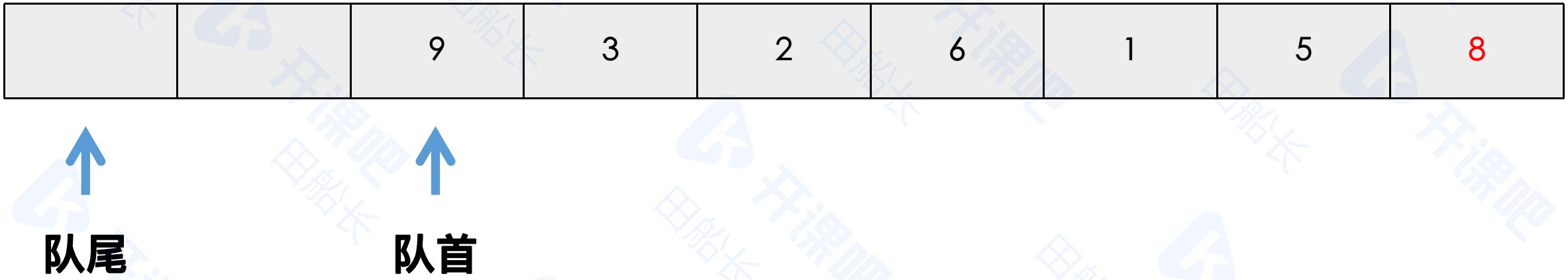
入队操作



size = 6

继续将元素8入队

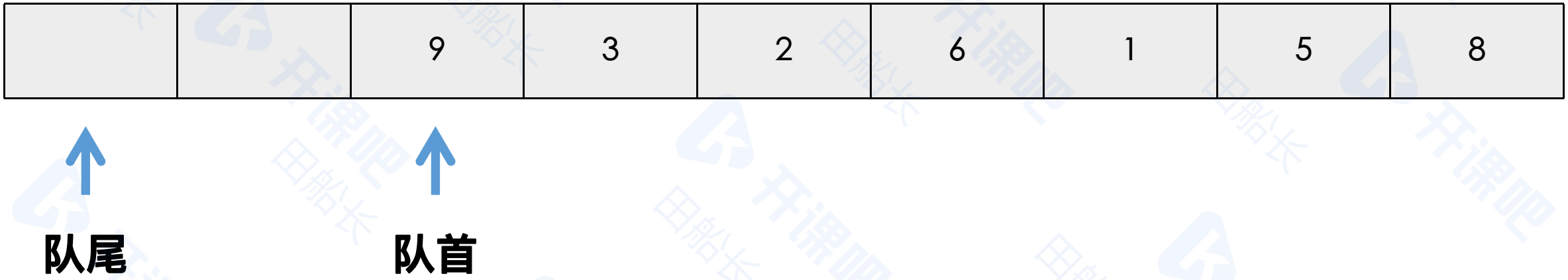
入队操作



size = 7

循环队列，队尾在最后后移时，移动到起始位

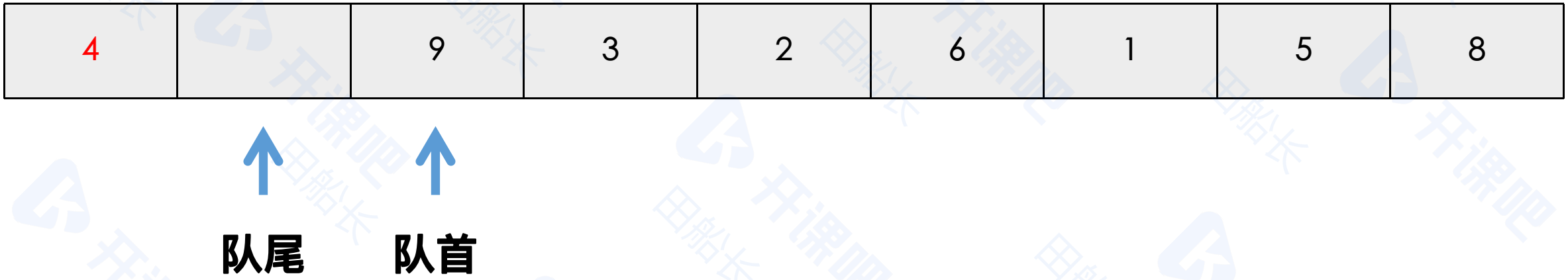
入队操作



size = 7

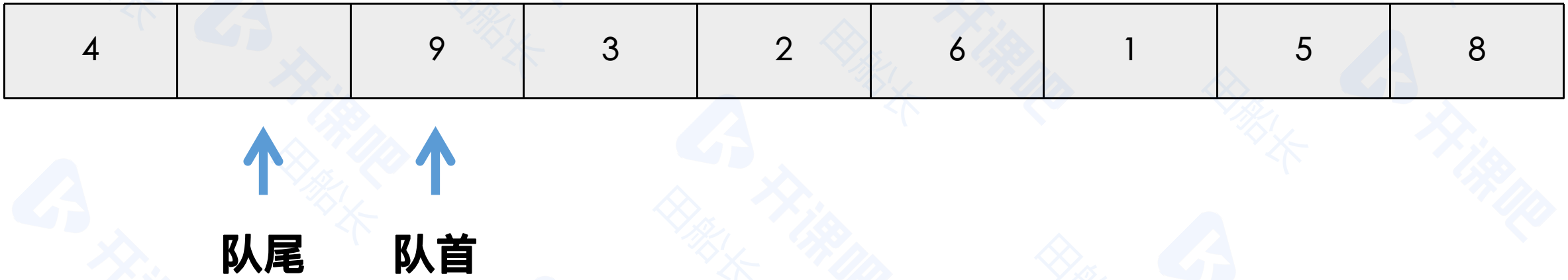
继续将4入队

入队操作



size = 8　直接入队即可，此时队尾+1为队首，说明循环队列已满

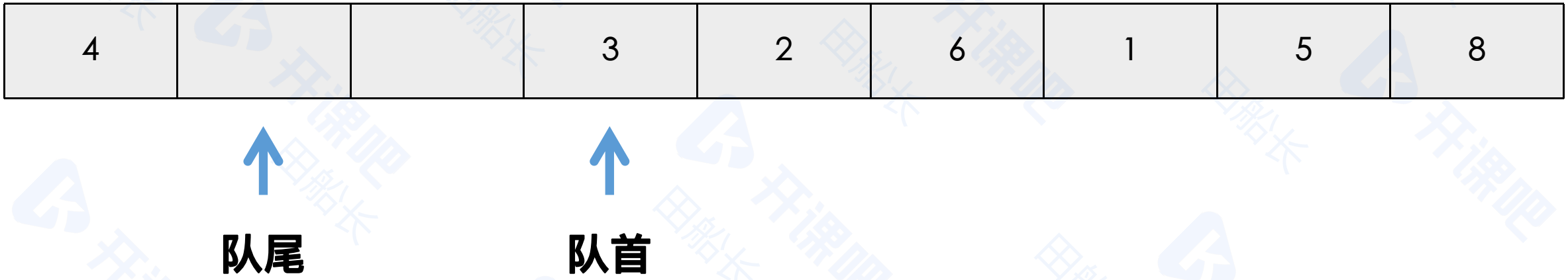
出队操作



size = 8

将队首元素出队

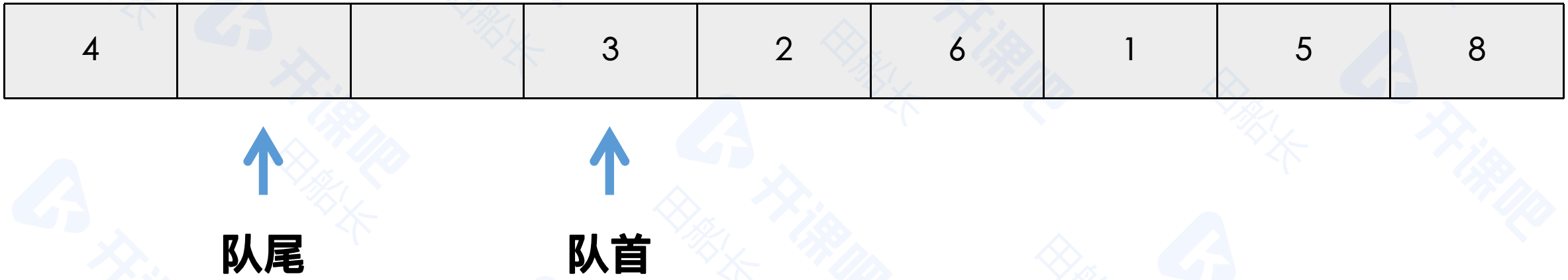
出队操作



size = 7

直接出队即可

出队操作



size = 7

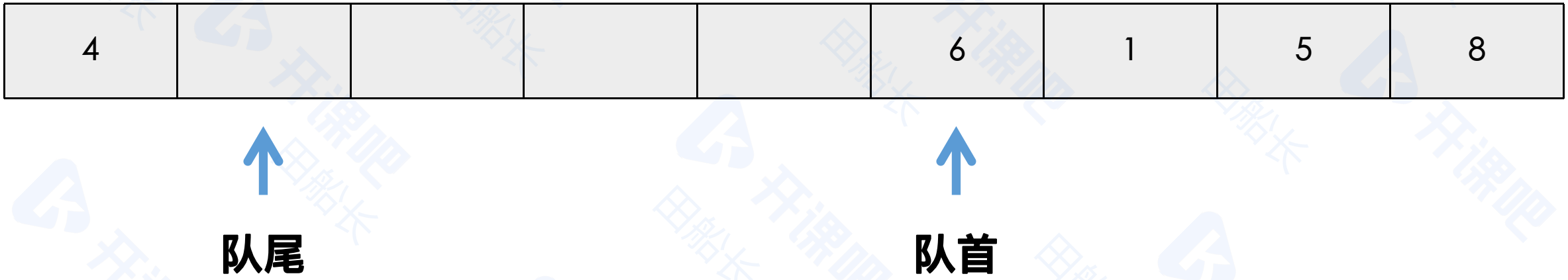
持续出队

出队操作

size = 6

持续出队

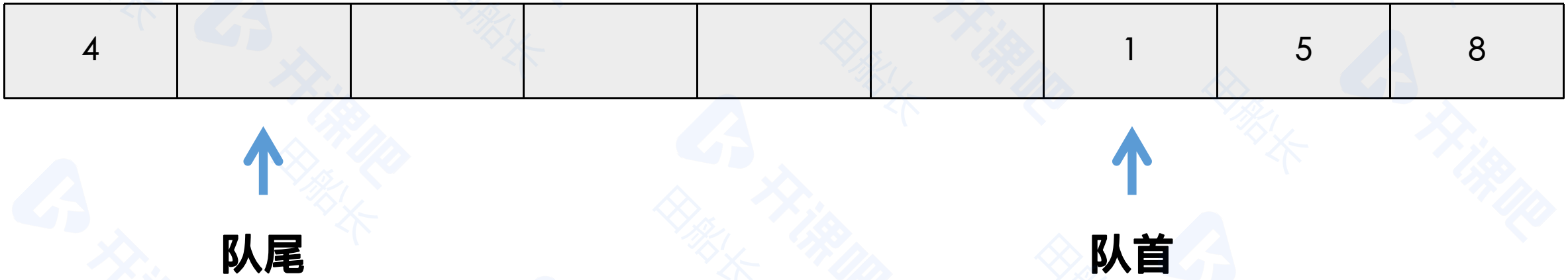
出队操作



size = 5

持续出队

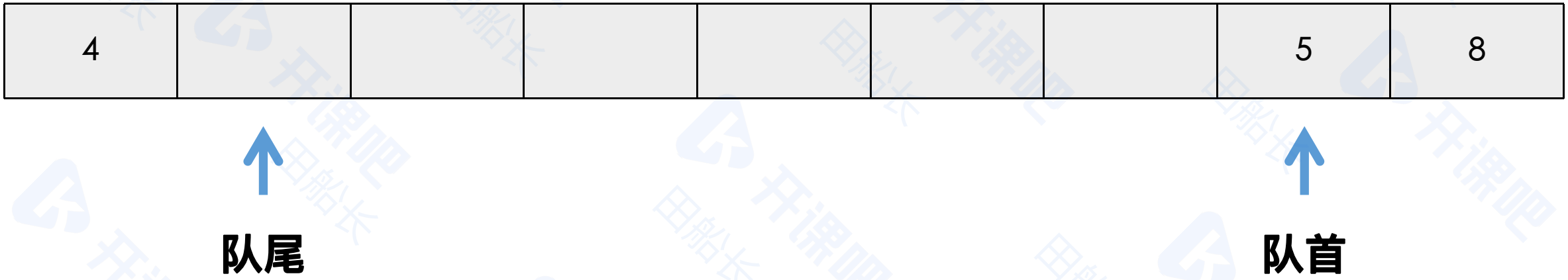
出队操作



size = 4

持续出队

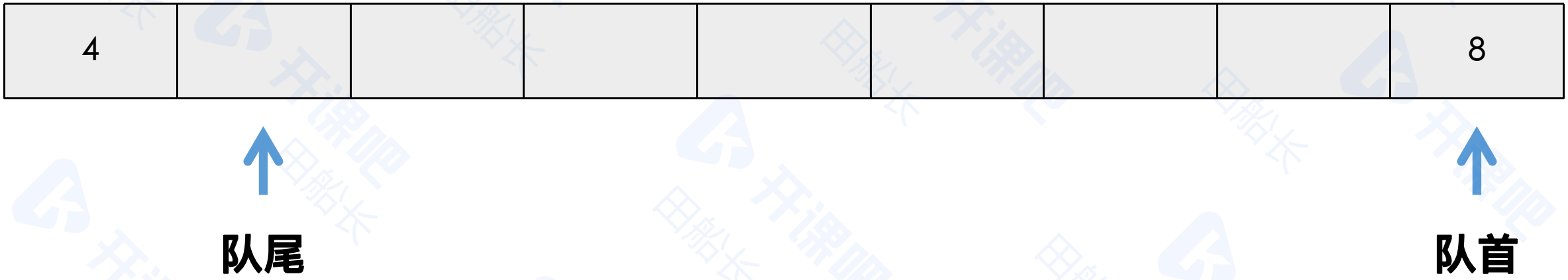
出队操作



size = 3

持续出队

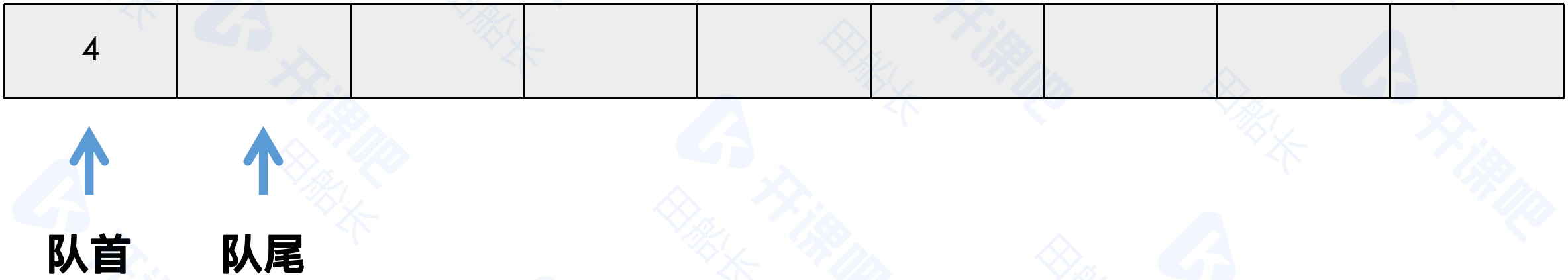
出队操作



size = 2

持续出队

出队操作

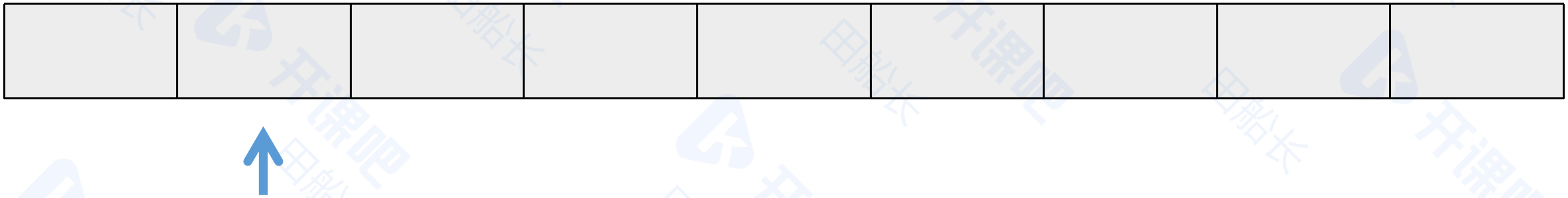


循环队列，队首在最后后移时，移动到起始位

持续出队

size = 1

出队操作



队尾
队首

size = 0

队首与队尾重叠，队列为空