4.1 队列基础概念_物联网 / 嵌入式工程师 - 慕课网

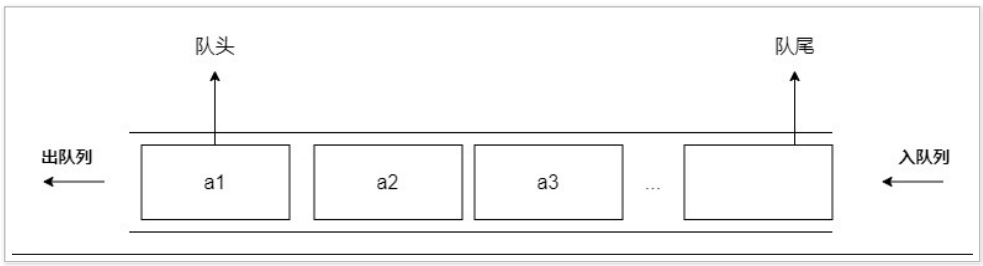
> 慕课网慕课教程 4.1 队列基础概念涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

当我们拨打联通、移动客服电话的时候，客服人员与客户相比总是在少数，在所有客服人员都占线的情况下，客户被要求等待，直到有某个客服人员空下来，才能让最先等待的客户接通电话。也就是说我们这里将所有打电话的客户进行了排队的操作。还有，过年的时候火车票非常的难买，一般去买火车站买票的时候，对会排着长长的队伍。这些都是我们常见的排队。我们数据结构中的队列，就是类似的结构。

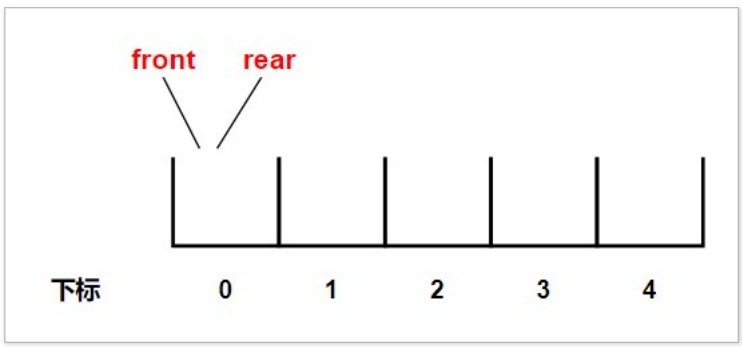
队列是一种先进先出（First In Fisr Out）的线性表，简称 FIFO，允许在一端进行插入操作的叫做**队尾**，允许删除的一端称为**队头**。

假如队列的元素为 a1,a2,...an, 那么如下图所示，a1 就是队头元素，

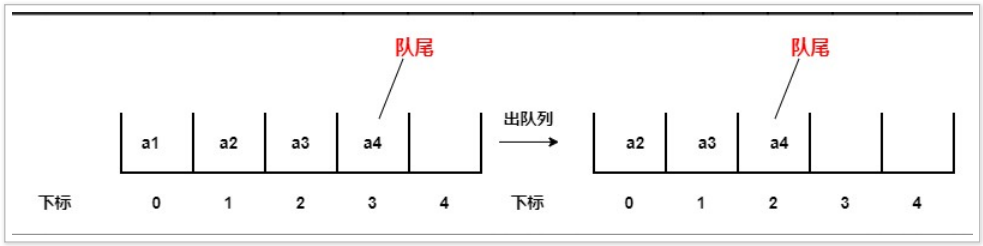
an 就是队尾元素。就像我们排队走地下通道，**第一个**进入的人肯定是**第一个**出来的人，这个就是我们所说的**先进先出**，如下图所示 **。**



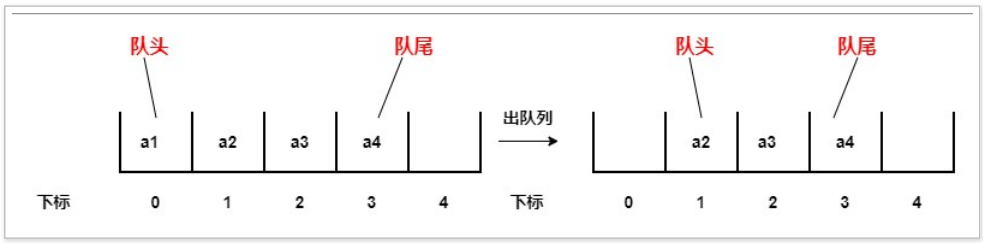
按照我们之前的学习规律，我们应该会学习队列的顺序存储和队列的链式存储。下面我们先来看看队列的顺序存储。由于我们不知道究竟队列究竟存储多少个元素，因此我们一般定义一个比较大的数组来存储我们的数据。假如我们的一个队列有 n 个元素，**一般下标为 0 的一般我们叫做队头**，所谓的入队操作，就是在数组最后一个元素后，在追加一个新的元素。前面的元素不需要移动。如下图



** 而根据我们队列的定义，我们队列的出队操作都是在队头，**** 也就是在下标为 0 的位置出队。** 那也就是说，我们队列后面的元素相对来说都需要向前移动，以保持我们下标为 0 位置不为空。如下图所示。



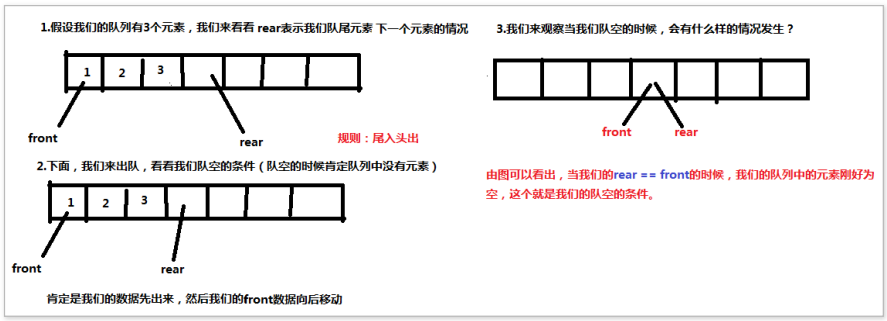
** 这样对我们来说是不是效率太低呢？** 若是我们把队头设置为可移动的，也就是说队头不需要一定在下标为 0 的位置，这样我们的效率不是大大的提高了吗?

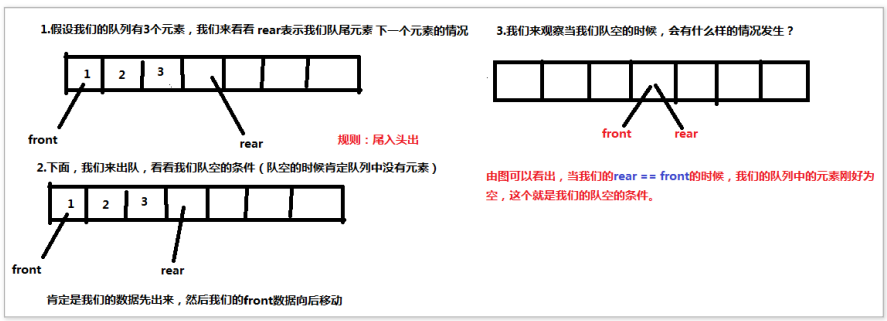


** 既然我们的数据是 **** 尾入头出，** 那么我们就来定义两个变量来表示头和尾，一个叫做 front 表示我们的队头元素的下标，一个叫做 rear 表示我们队尾元素的下一个元素下标。

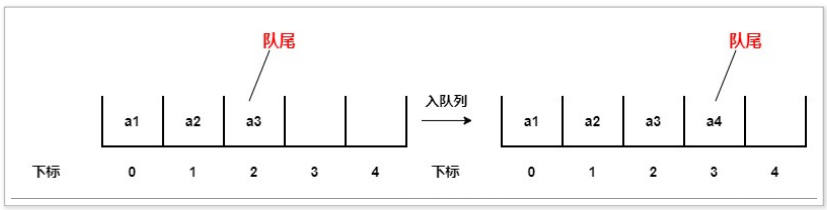
** 提问：** 为什么要 rear 要指向我们的队尾元素的下一个元素的下标，而不是直接指向我们的队尾元素的下标呢?

答案：表示队尾元素的下一个元素下标方便我们队空的操作。

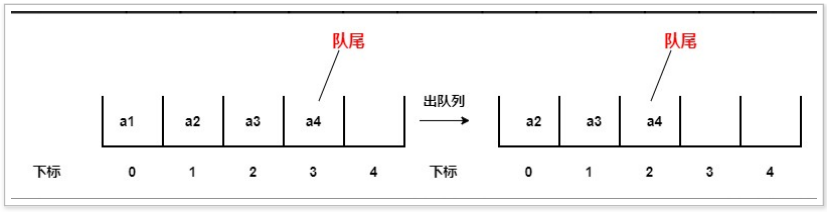




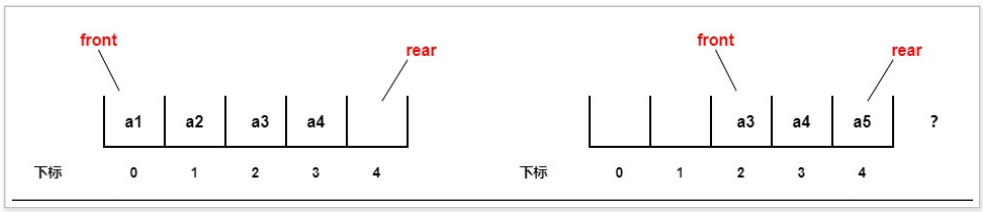
- 顺序存储的问题
  - 假设长度是我们有 int a[5] 的数组，刚开始里面没有存放任何的元 素, front 和 rear 都指向下标为 0 的位置。



  - a1,a2,a3,a4 开始入队, front 依旧指向了 0，rear 则指向了 4 的位置。



- ** 出队 a1,a2, 则 front 指向下标为 2 的位置，rear 不变。如下左图所示，然后在入队 a5, 此时 frone 位置不变，raer 的位置移动数组之外。** 是不是越界了？我们数组中只有 3 个元素竟然越界了？？？

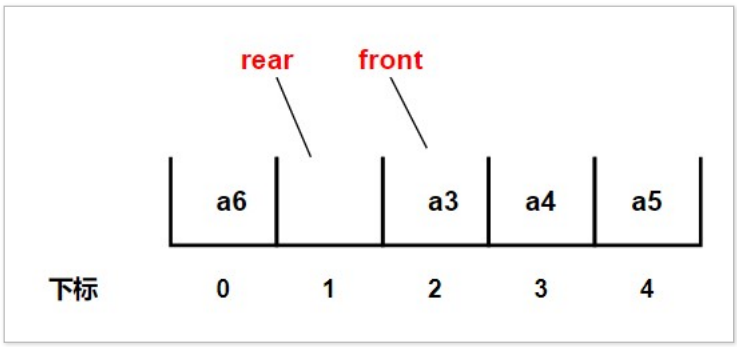


** 这种现象叫做假溢出。** 但是，我们前面的 0 和 1 的位置是空的啊！为什么不让他们放到 0 和 1 的位置呢?

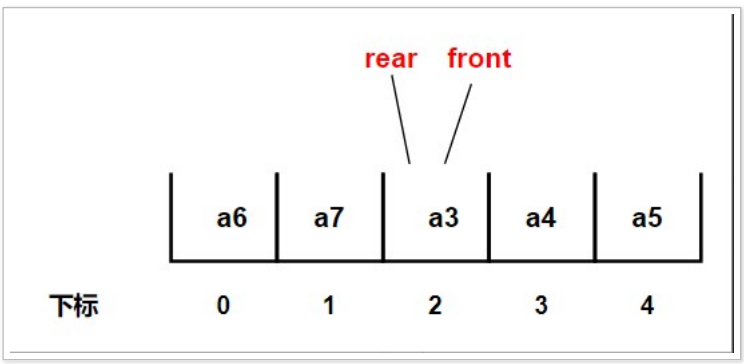
这个就是我们下面要说的循环队列的思想。

队列的这种首尾相连的顺序存储结构称之为循环队列。

若是我们允许当数组后面的数据满了的时候，我们允许 rear 把数据塞到我们下标为 0 的状态，就如下图。



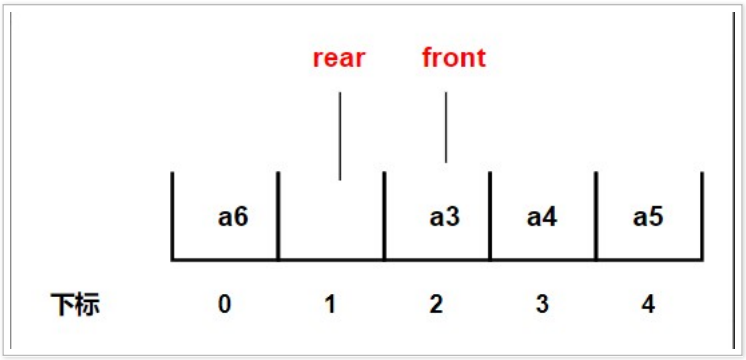
** 若是我们的在入队 a7 呢? ** 我们的 rear 和 fron 不是就重合了吗? 如下图。



解决方法:

1、设置一个标志位以区分队列空，还是队列满（用的少）

2、我们修改队列满的条件，保留一个元素的空间。也就是说，队列满的时候，我们的数组中还有一个空闲的单位！如下图



队满的条件是 " 队列 front 指向了队尾 rear 的下一个位置上 "。

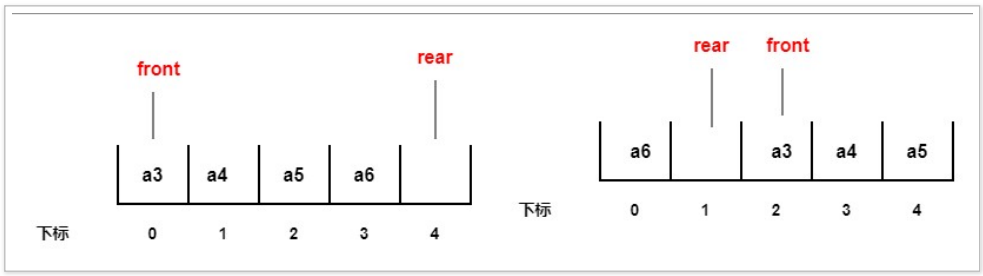
首先我们来明确一个概念假如我们的数组中有 5 个数据。

思考：1 % 5 = ? 2 % 5 = ? 3 % 5 = ? 4 % 5 = ? 5 % 5 = ?

21 % 5 = ? 22 % 5 = ? 23 % 5 = ? ...

** 总结：** 运算数 % 5 的到的结果，一定是 0~4 之间的数字.

以上两种情况，造成我们的 front 的值可能比我们 rear 的值大，或者说比我们 rear 的值小。



front---------- 队头元素的位置

rear ---------- 队尾元素下一个元素的位置

** 队空条件：**front == rear

** 队满条件：**front == (rear + 1) % MAX

更新 rear 的方法 (让 rear 指向下一个位置的方法)：

rear = (rear + 1) % MAX;

更新 front 的方法 (让 front 正确指向下一个位置的方法)：

front = (front + 1) % MAX;

全文完