数据结构与算法（三）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

第3章　栈与队列

- 栈
- 栈的应用
  - 递归到非递归的转换
- 队列

队列的定义

- **先进先出（First In First Out）**
  - 限制访问点的线性表
    - 按照到达的顺序来释放元素
    - 所有的插入在表的一端进行，所有的删除都在表的另一端进行
- **主要元素**
  - 队头（front）
  - 队尾（rear）

队列的主要操作

- 入队列（enQueue）
- 出队列（deQueue）
- 取队首元素（getFront）
- 判断队列是否为空（isEmpty）

队列的抽象数据类型

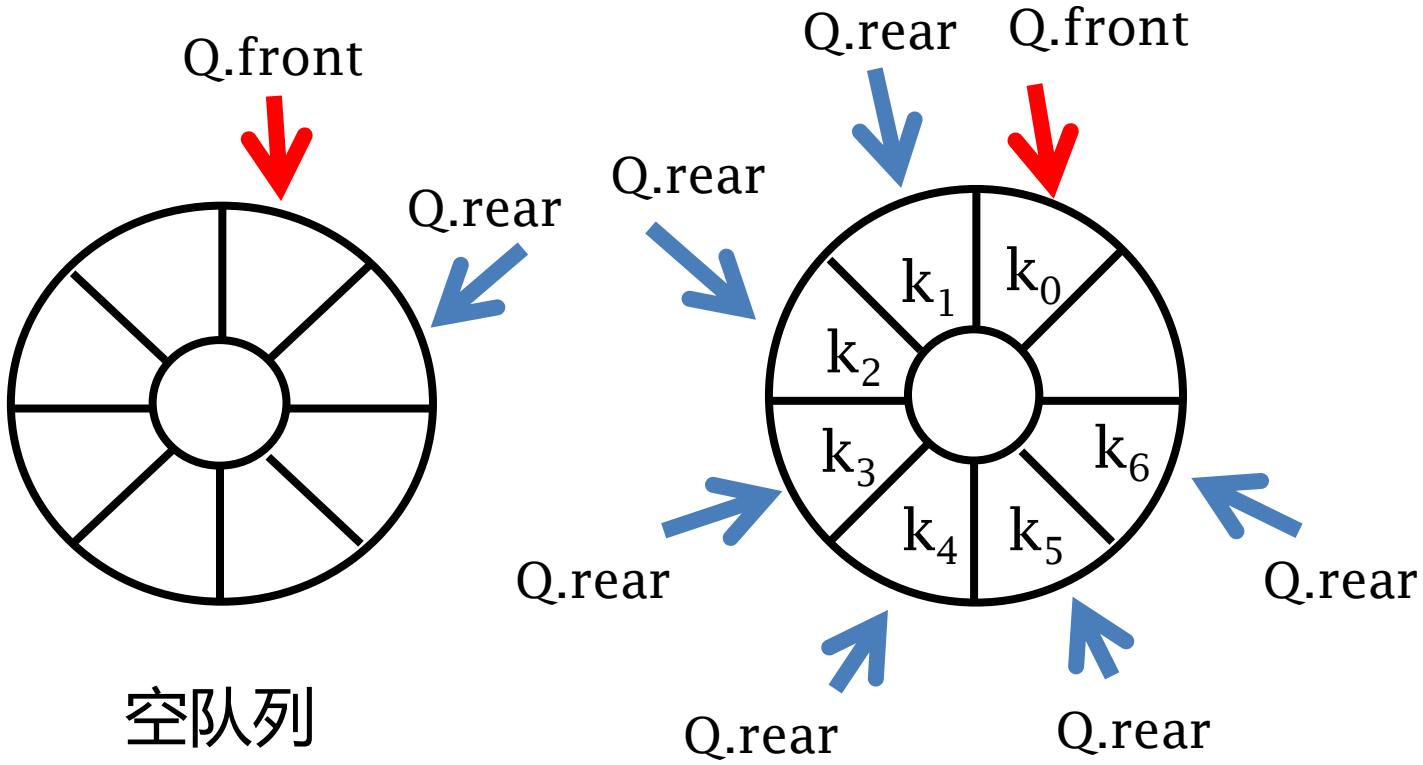
```
template <class T> class Queue {
public:                  // 队列的运算集
    void clear();        // 变为空队列
    bool enQueue(const T item);
                         // 将item插入队尾，成功则返回真，否则返回假
    bool deQueue(T & item) ;
                         // 返回队头元素并将其从队列中删除，成功则返回真
    bool getFront(T & item);
                         // 返回队头元素，但不删除，成功则返回真
    bool isEmpty(); // 返回真，若队列已空
    bool isFull();  // 返回真，若队列已满
};
```

队列的实现方式

- **顺序队列**
  - **关键**是如何防止假溢出

- **链式队列**
  - 用单链表方式存储，队列中每个元素对于链表中的一个结点

队列示意：环形(实指)

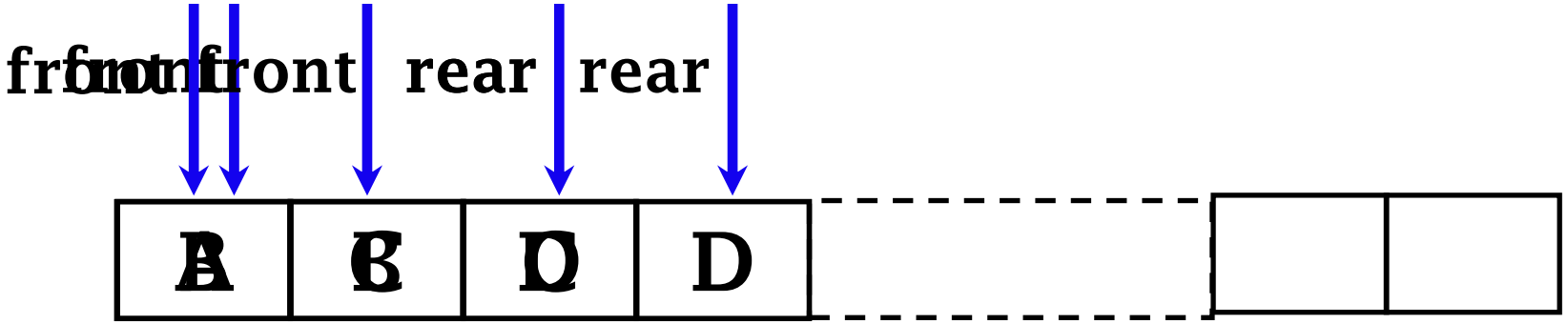

Q.front
Q.rear
空队列
Q.rear
Q.front
Q.rear
Q.rear
Q.rear
Q.rear
Q.rear
k_1
k_0
k_2
k_3
k_6
k_4
k_5

顺序队列的类定义

```
class arrQueue: public Queue<T> {
private:
    int mSize;              // 存放队列的数组的大小
    int front;              // 表示队头所在位置的下标
    int rear;               // 表示队尾所在位置的下标
    T * qu;                 // 存放类型为T的队列元素的数组
public:                     // 队列的运算集
    arrQueue(int size);     // 创建队列的实例
    ~arrQueue();            // 消除该实例，并释放其空间
}
```

顺序队列的维护

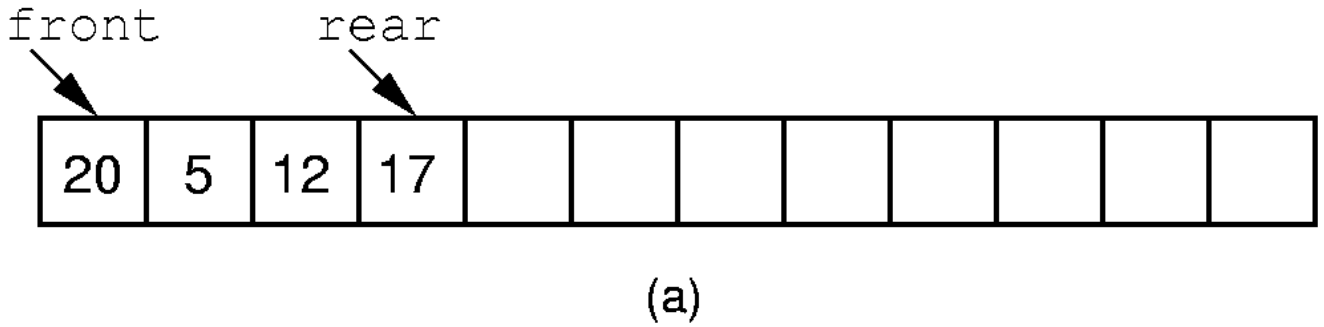
- rear实指



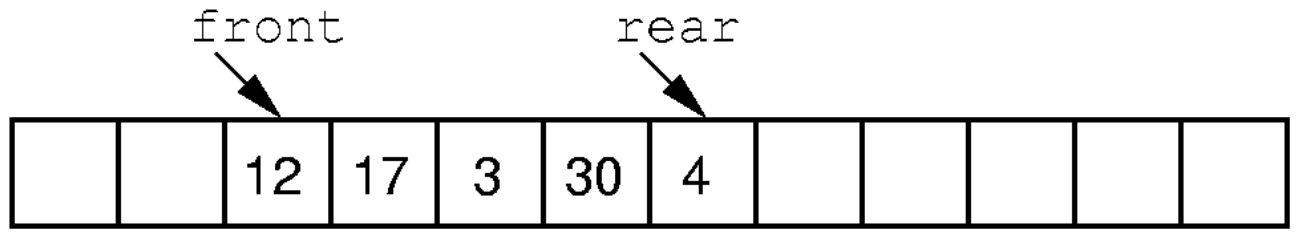
删除

顺序队列的维护

- front和rear都实指



(a)



(b)

顺序队列代码实现

```
template <class Elem> class Aqueue : public Queue<Elem> {
private:
    int size;                       // 队列的最大容量
    int front;                      // 队首元素指针
    int rear;                       // 队尾元素指针
    Elem *listArray;                // 存储元素的数组
public:
    AQueue(int sz=DefaultListSize) {
        // 让存储元素的数组多预留一个空位
        size = sz+1;                // size数组长，sz队列最大长度
        rear = 0; front = 1;        // 也可以rear=-1; front=0
        listArray = new Elem[size];
    }
    ~AQueue() { delete [] listArray; }
    void clear() { front = rear+1; }
```

顺序队列代码实现

```
bool enqueue(const Elem& it) {
    if (((rear+2) % size) == front) return false;
                                // 还剩一个空位，就要报满
    rear = (rear+1) % size; // 因为实指，需要先移动到下一个空位
    listArray[rear] = it;
    return true;
}
bool dequeue(Elem& it) {
    if (length() == 0) return false;
                                    // 队列为空
    it = listArray[front];          // 先出队，再移动front下标
    front = (front+1) % size;       // 环形增加
    return true;
}
```

顺序队列代码实现

```
bool frontValue(Elem& it) const {
    if (length() == 0)
        return false;        // 队列为空
    it = listArray[front]; return true;
}
int length() const {
    return (size +(rear - front + 1)) % size;
}
```

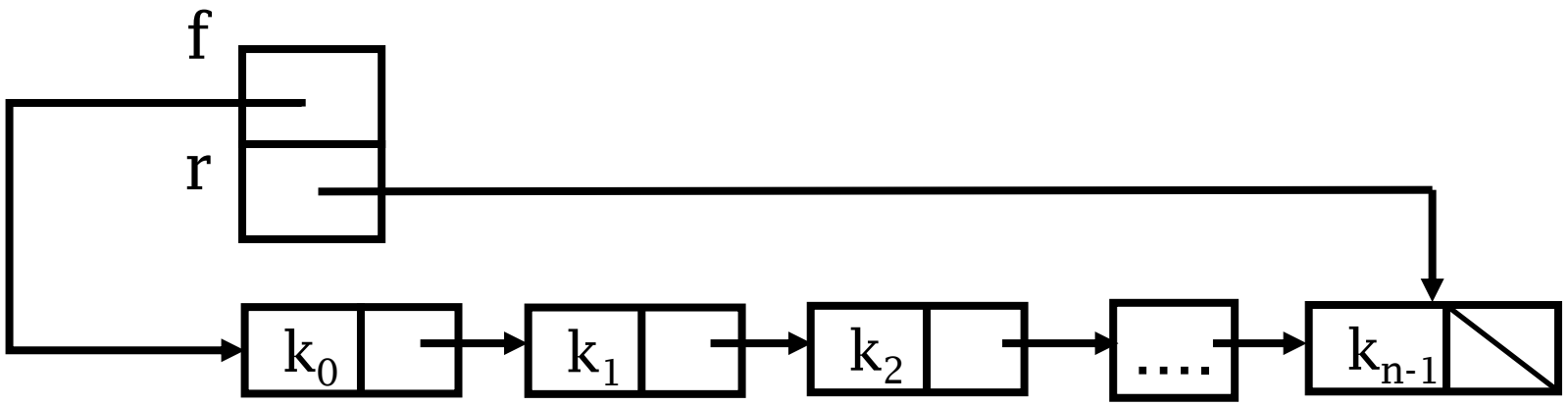
思考

- 1. 只是用 front, rear 两个变量，长度为 mSize = n 的队列，可以容纳的最大元素个数为多少？ 请给出详细的推导过程。

- 2. 如果不愿意浪费队列的存储单元，还可以采用什么方法？

链式队列的表示

- 单链表队列
- 链接指针的方向是从队列的前端向尾端链接



链式队列的类定义

```
template <class T>
class lnkQueue: public Queue<T> {
private:
    int size;                       // 队列中当前元素的个数
    Link<T>* front;                 // 表示队头的指针
    Link<T>* rear;                  // 表示队尾的指针
public:                             // 队列的运算集
    lnkQueue(int size);             // 创建队列的实例
    ~lnkQueue();                    // 消除该实例，并释放其空间
}
```

链式队列代码实现

```
bool enQueue(const T item) { //  item入队，插入队尾
    if (rear == NULL) {       // 空队列
        front = rear = new Link<T> (item, NULL);
    }
    else  {                   // 添加新的元素
        rear-> next = new Link<T> (item, NULL);
        rear = rear ->next;
    }
    size++;
    return true;
}
```

链式队列代码实现

```
bool deQueue(T* item)  {              // 返回队头元素并从队列中删除
    Link<T> *tmp;
    if (size == 0)  {                 // 队列为空，没有元素可出队
        cout << "队列为空" << endl;
        return false;
    }
    *item = front->data;
    tmp = front;
    front = front -> next;
    delete tmp;
    if (front == NULL)
        rear = NULL;
    size--;
    return true;
}
```

顺序队列与链式队列的比较

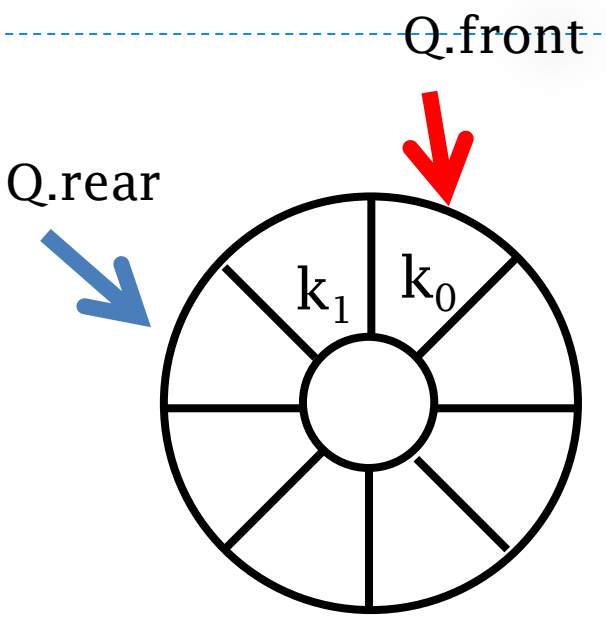
- **顺序队列**
  - 固定的存储空间
- **链式队列**
  - 可以满足大小无法估计的情况

都不允许访问队列内部元素

队列的应用

- 只要满足先来先服务特性的应用均可采用队列作为其数据组织方式或中间数据结构
- 调度或缓冲
  - 消息缓冲器
  - 邮件缓冲器
  - 计算机硬设备之间的通信也需要队列作为数据缓冲
  - 操作系统的资源管理
- 宽度优先搜索

思考



- 链式队列往往用单链表。
  为什么不用双链表来实现？

- 若采用虚指方法实现队尾指针(rear指向队尾元素后一个元素，和实指相比后移一位)，在具体实现上有何异同？哪一种更好？

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材