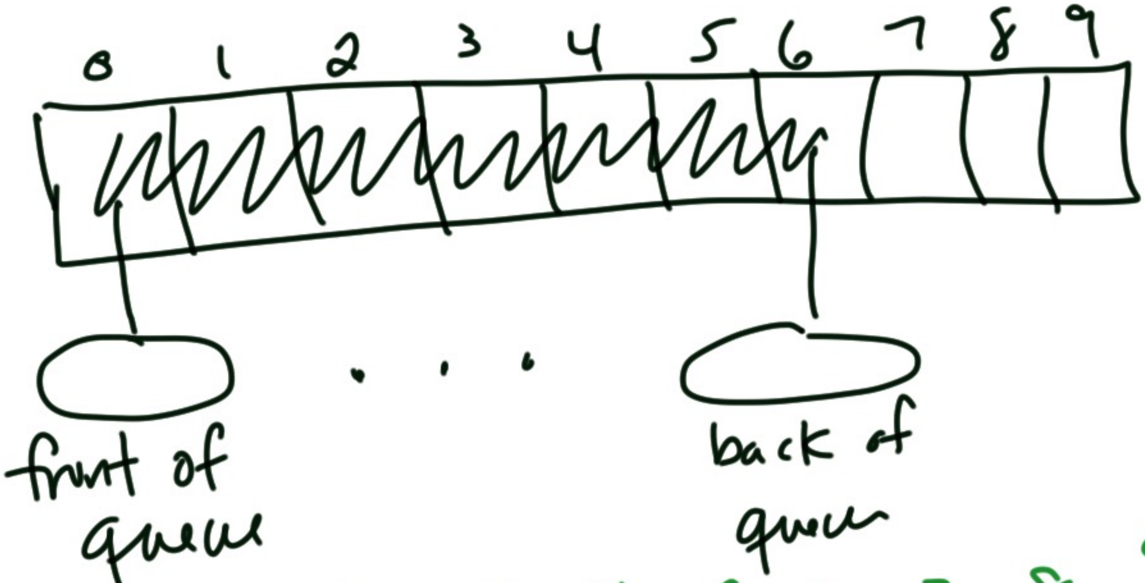
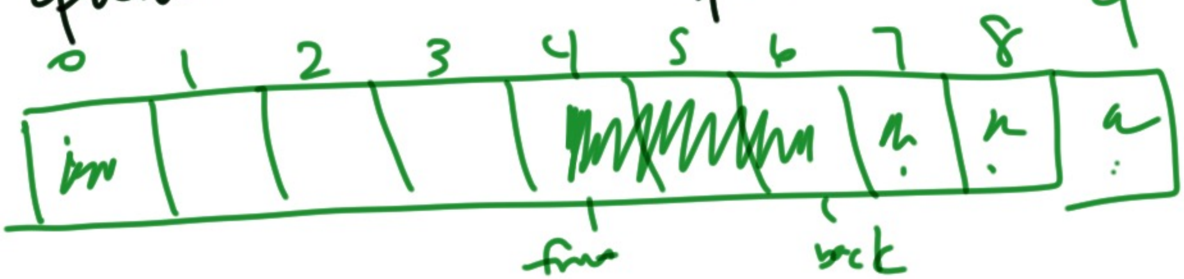
Implementing a queue w/an array

enqueue 7 items results in



what if dequeue 4 times?

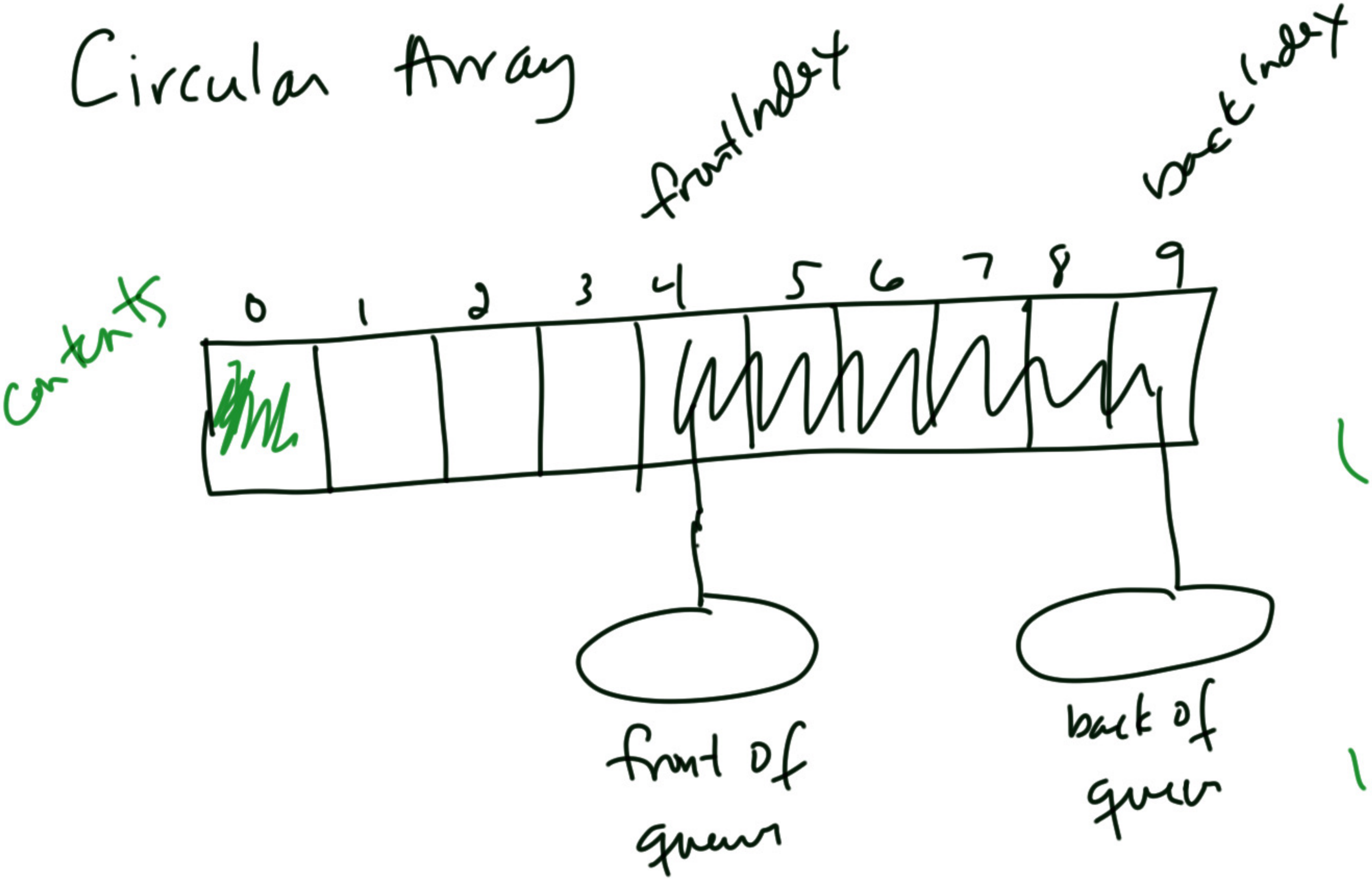


what if enqueue 4 more items?

wrap around

backIndex = (backIndex + 1) % contents.length

Circular Array



$(9+1) \% 10$

$10 \% 10$

$1 \text{ R } 0$

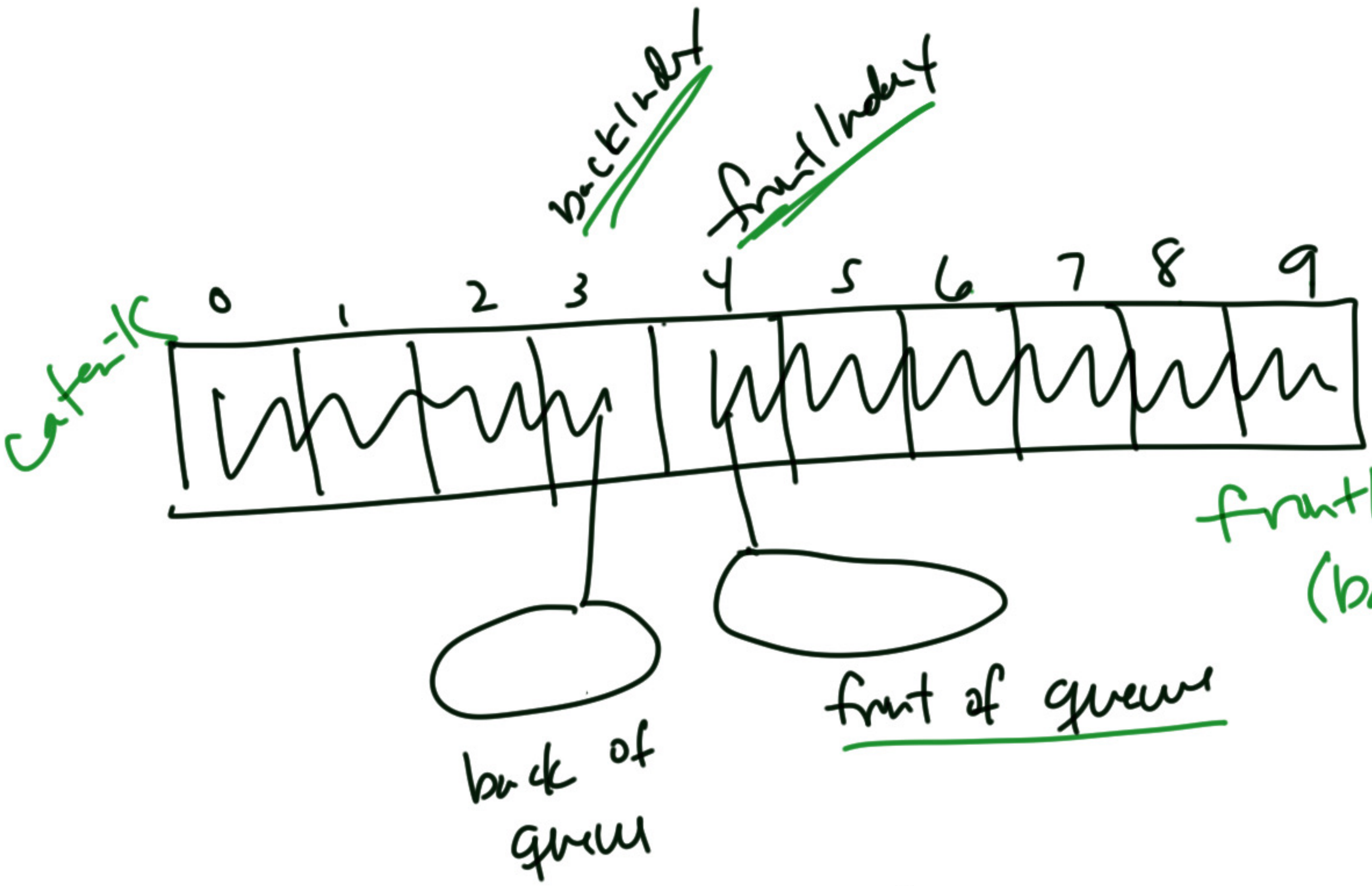
$10 \overline{)10}$

$\underline{10}$

0

What if we enqueue?

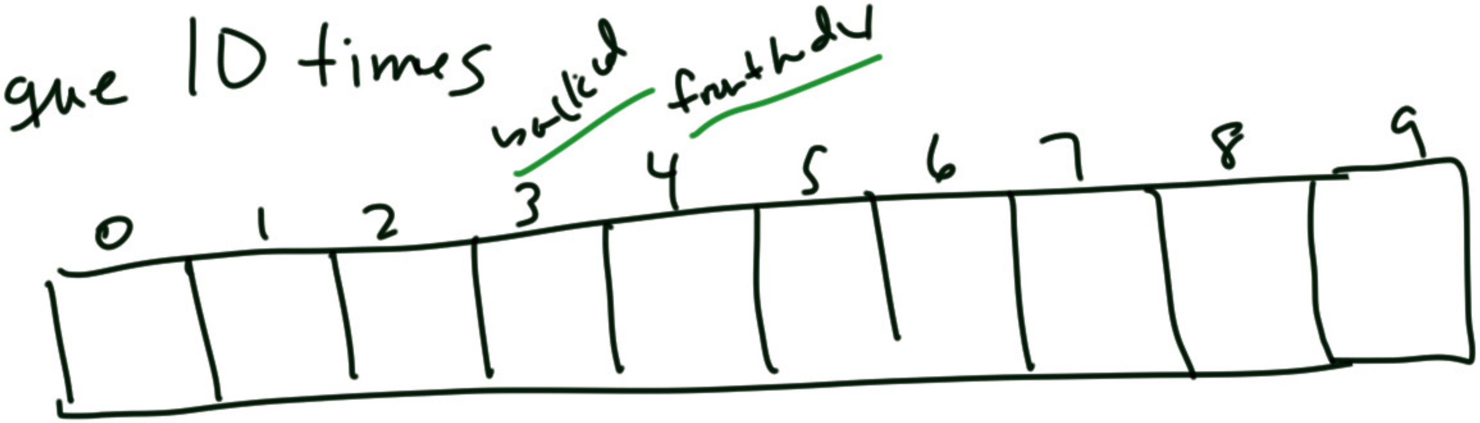
$\text{backIndex} = (\text{backIndex} + 1) \% \text{contents.length}$



FULL

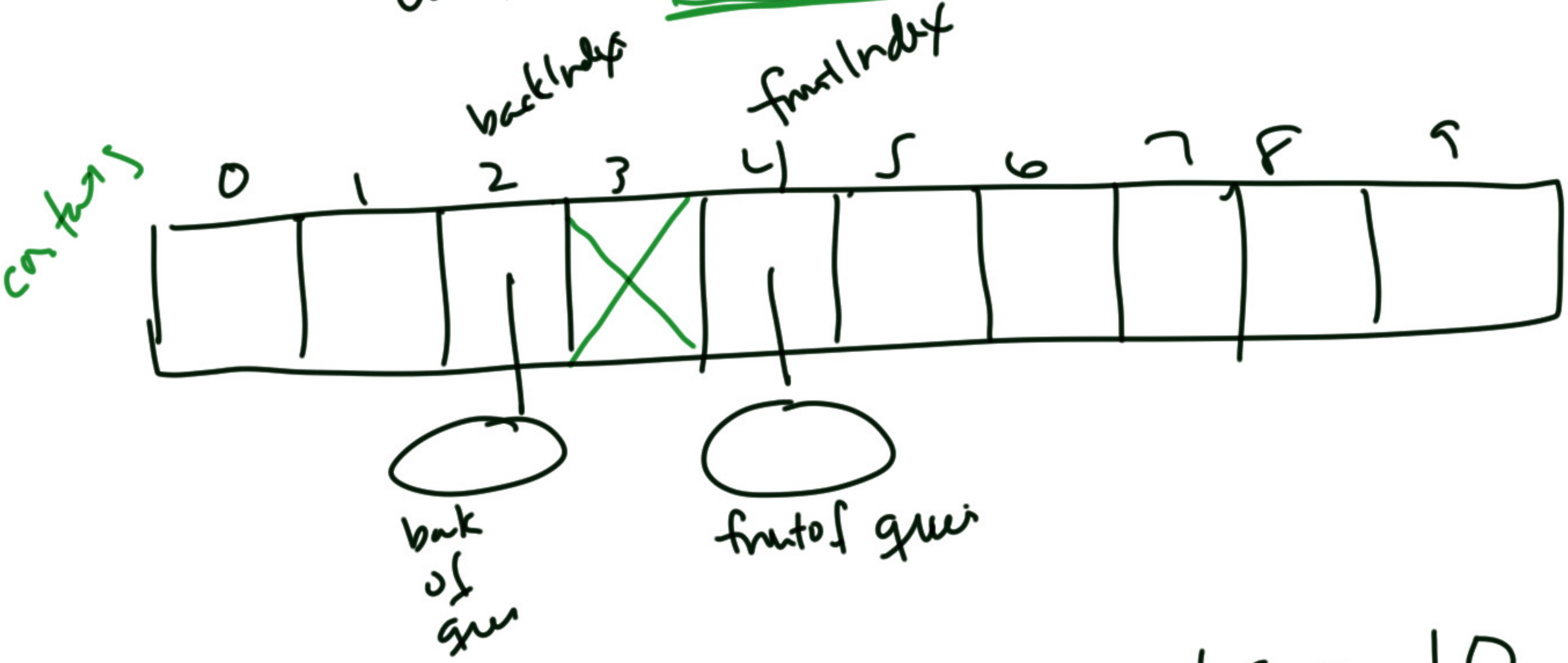
frontIndex = (backIndex + 1) % contents.length

deque 10 times



EMPTY

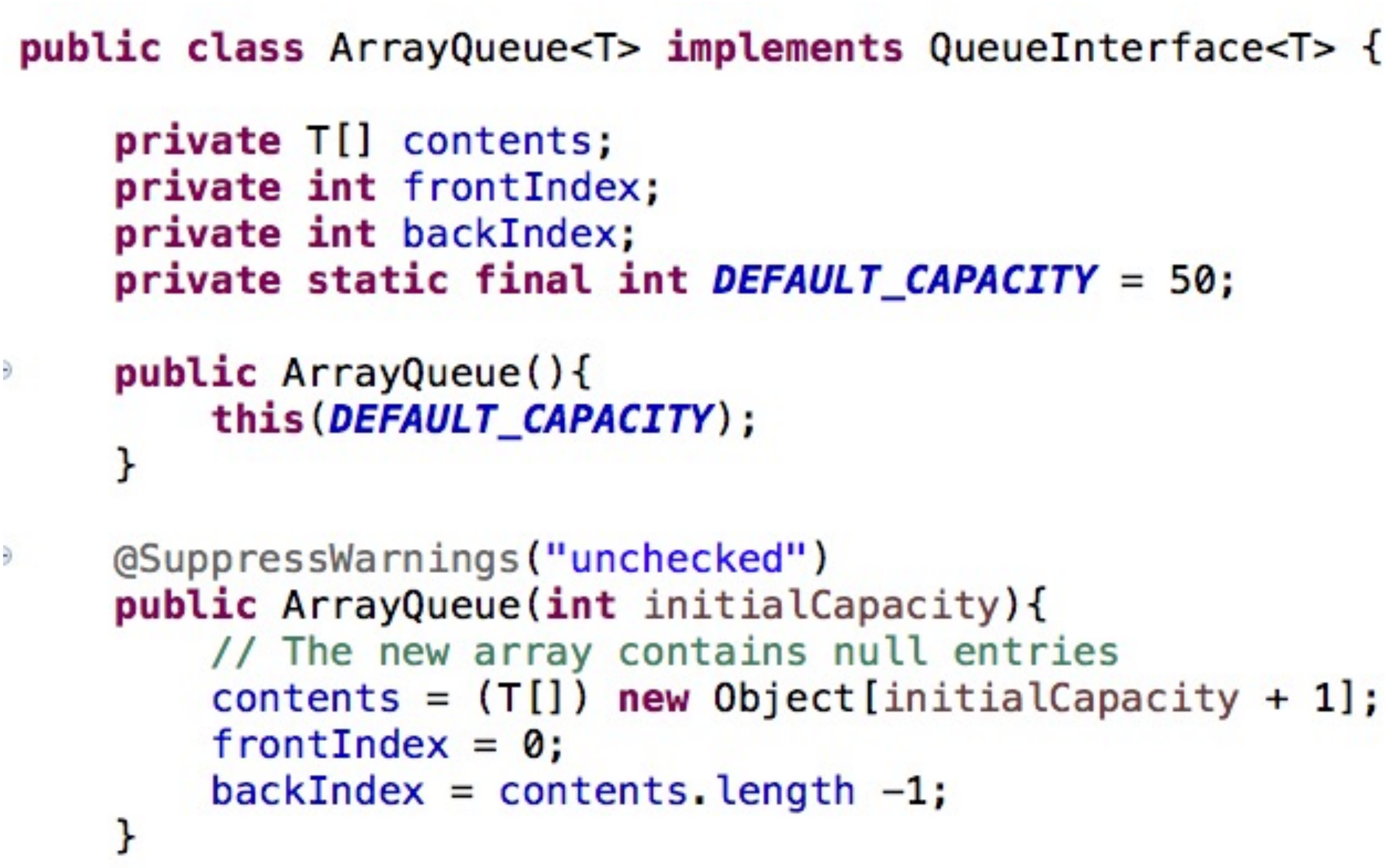
Solution: Circular Array with one unused location



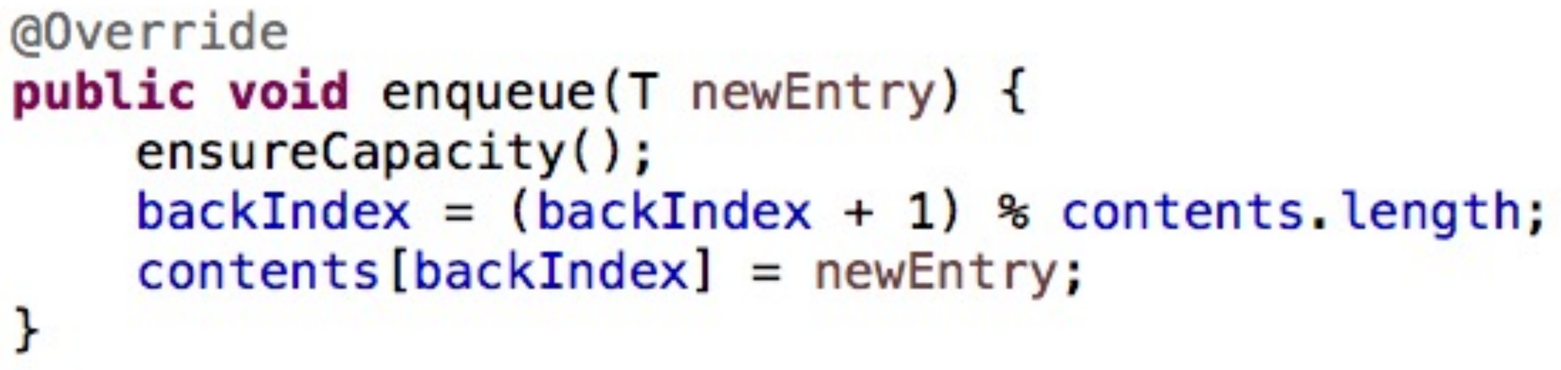
length of contents = 10

capacity of queue = 9 10-1

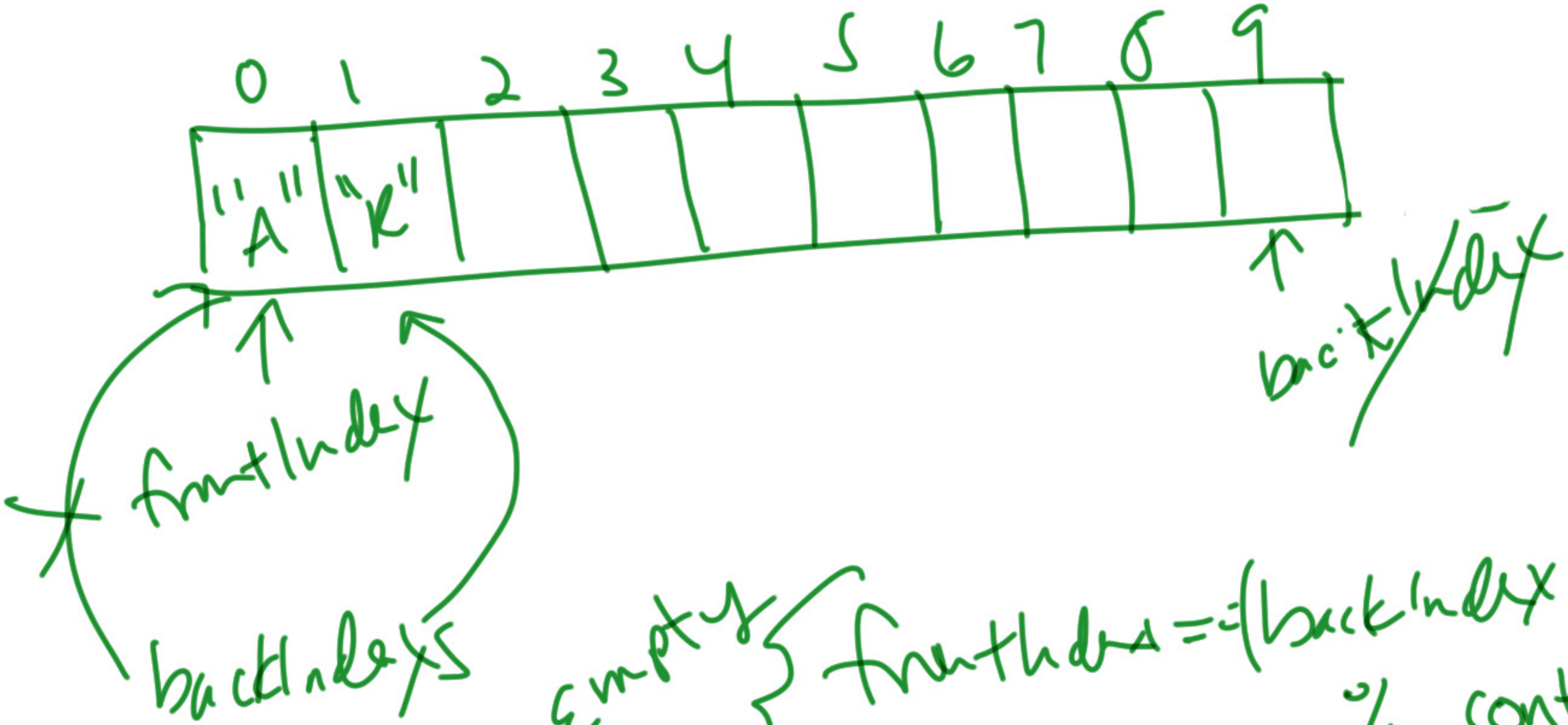
size = # of items in queue (0-9)



```
public class ArrayQueue<T> implements QueueInterface<T> {

    private T[] contents;
    private int frontIndex;
    private int backIndex;
    private static final int DEFAULT_CAPACITY = 50;

    public ArrayQueue(){
        this(DEFAULT_CAPACITY);
    }

    @SuppressWarnings("unchecked")
    public ArrayQueue(int initialCapacity){
        // The new array contains null entries
        contents = (T[]) new Object[initialCapacity + 1];
        frontIndex = 0;
        backIndex = contents.length -1;
    }
```



```java
@Override
public void enqueue(T newEntry) {
    ensureCapacity();
    backIndex = (backIndex + 1) % contents.length;
    contents[backIndex] = newEntry;
}
```



empty { frontIndex == (backIndex + 1) % contents.length

1 elemt { frontIndex == backIndex

frontIndex == (backInd + 2) % contents.lgth

Wrapping Around

backIndex = (backIndex + 1) % queue.length

Cases when queue.length is 10:

Initial backIndex	Incremented backIndex
0	1
1	2
5	6
8	9
9	0