Data Structures: Queues & ADT

CS 1112

Mona Diab

Queues

- Definition of a Queue
- Examples of Queues
- Design of a Queue Class
- Different Implementations of the Queue Class

Introduction to Queues

- A queue is a **waiting line**

- It’s in daily life:
  - A line of persons waiting to check out at a supermarket
  - A line of persons waiting to purchase a ticket for a film
  - A line of planes waiting to take off at an airport
  - A line of vehicles at a toll booth

Definition of a Queue

- A queue is a data structure that models/enforces the **first-come first-serve** order, or equivalently the **first-in first-out** (FIFO) order.
- That is, the element that is inserted first into the queue will be the element that will deleted first, and the element that is inserted last is deleted last.
- A waiting line is a good real-life example of a queue. (In fact, the British word for “line” is “queue”.)

Queues

- Unlike stacks in which elements are popped and pushed only at the ends of the list

Collection of data elements:

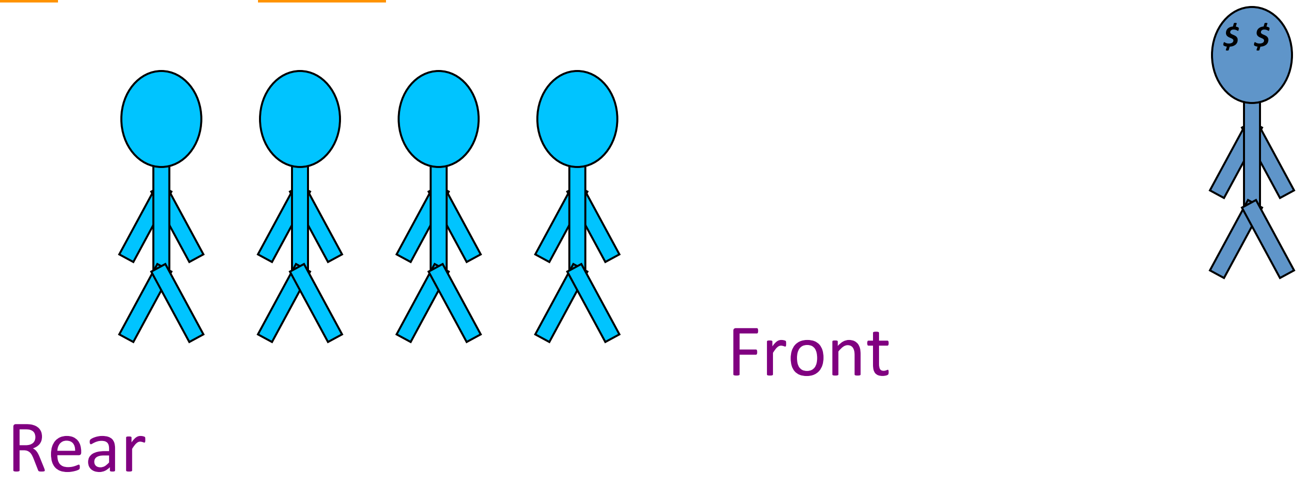
- items are removed from a queue at one end, called the **FRONT** of the queue;
- and elements are added at the other end, called the **BACK**

Introduction to Queues

- Difference between **Stack** and **Queues**:
  - Stack exhibits last-in-first-out (LIFO)
  - Queue exhibits first-in-first-out (FIFO)

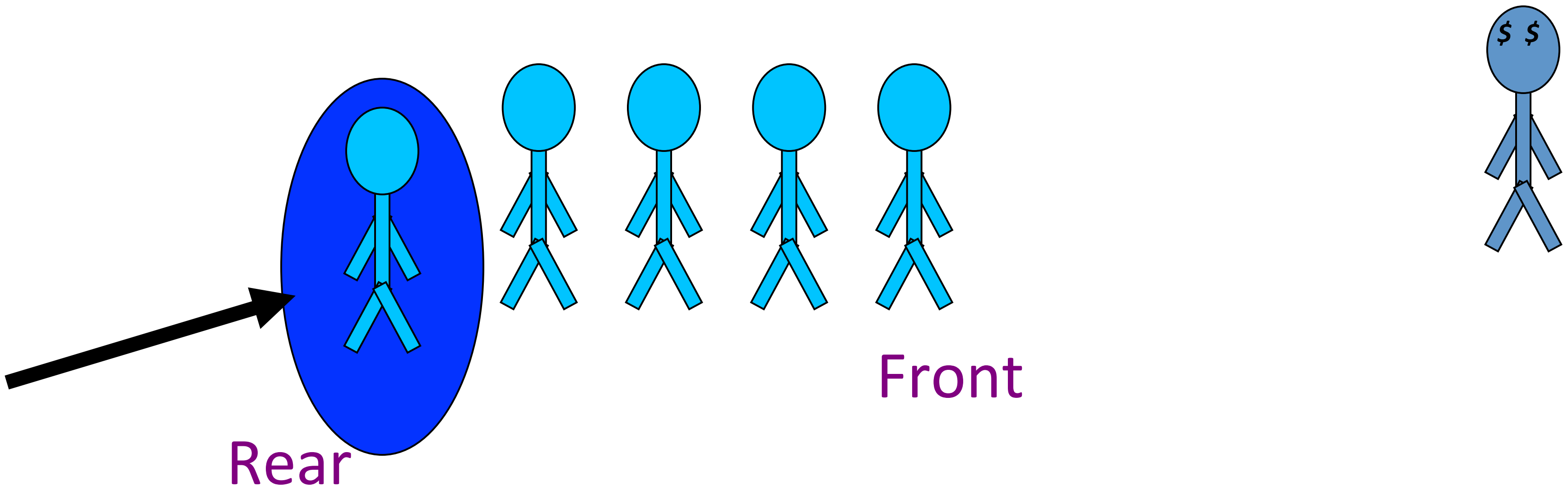
The Queue Operations

- A queue is like a line of people waiting for a bank teller. The queue has a **front** and a **rear**.



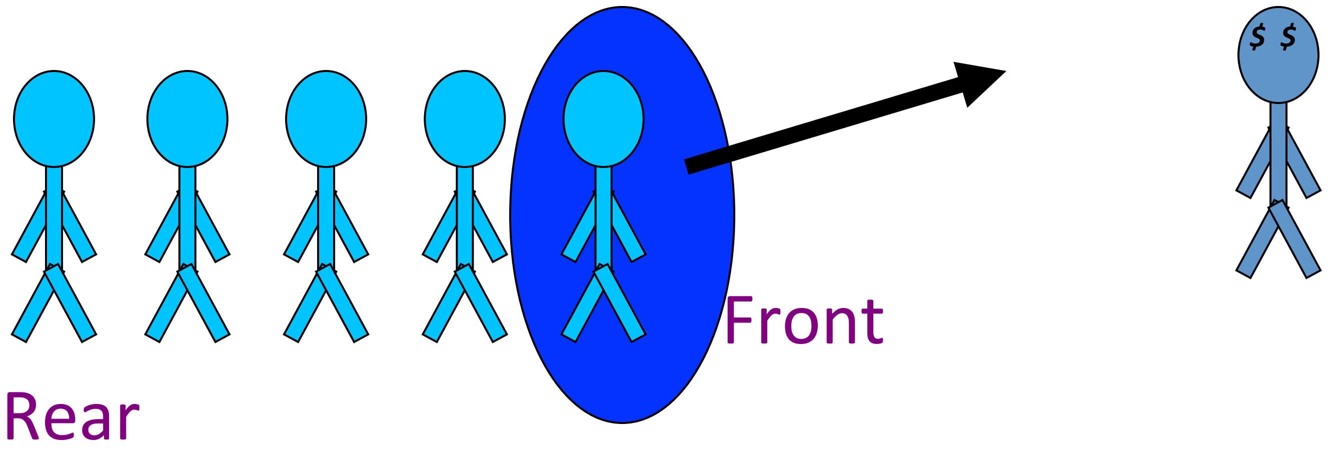
The Queue Operations

- New people must enter the queue at the rear. This is called an **enqueue** operation.



The Queue Operations

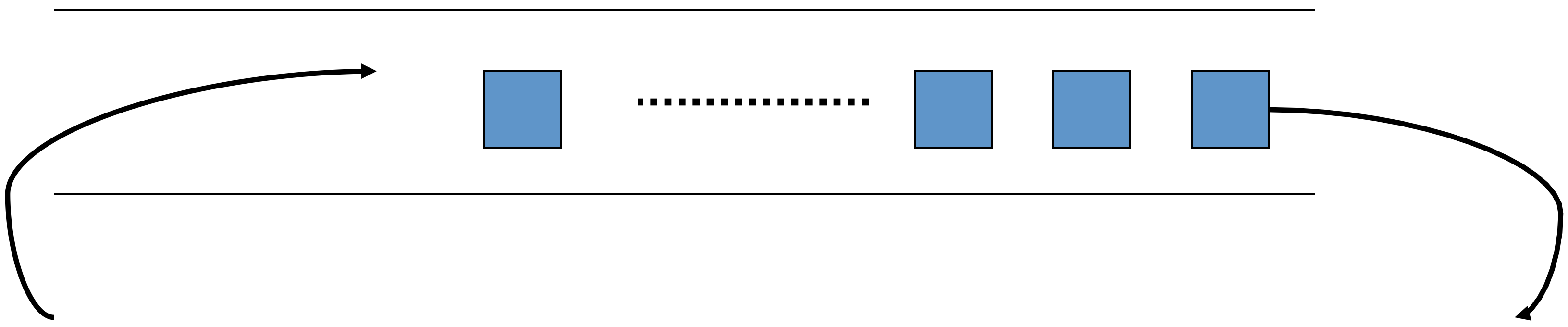
- When an item is taken from the queue, it always comes from the front. This is called a **dequeue** operation.



Queue Abstract Data Type

- Basic operations
  - Construct a queue
  - Check if empty
  - Enqueue (add element to back)
  - Front (retrieve value of element from front)
  - Dequeue (remove element from front)

A Graphic Model of a Queue



Tail:
All new items
are added on
this end

Head:
All items are
deleted from
this end

Operations on Queues

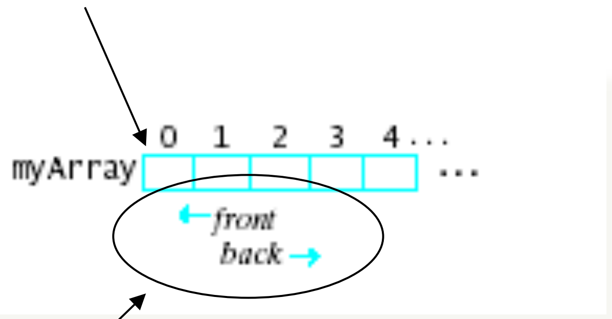
- **Insert(**item**)**: (also called enqueue)
  - It adds a new item to the tail of the queue
- **Remove( )**: (also called delete or dequeue)
  - It deletes the head item of the queue, and returns to the caller. If the queue is already empty, this operation returns NULL
- **getHead( ):**
  - Returns the value in the head element of the queue
- **getTail( ):**
  - Returns the value in the tail element of the queue
- **isEmpty( )**
  - Returns **true** if the queue has no items
- **size**( )
  - Returns the number of items in the queue

Queue as a Class

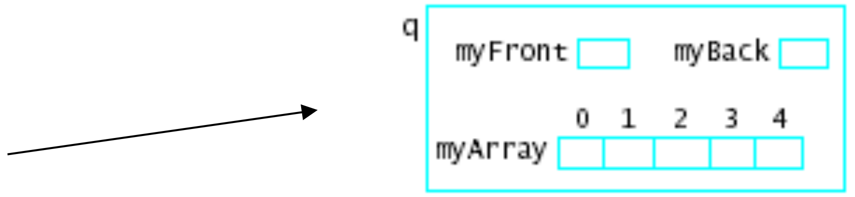
- Much like stacks and linked lists were designed and implemented as classes, a queue can be conveniently packaged as a class
- It seems natural to think of a queue as similar to a linked list, but with more basic operations, to enforce the FIFO order of insertion and deletion

Designing and Building a Queue Class Array-Based

- Consider an array in which to store a queue

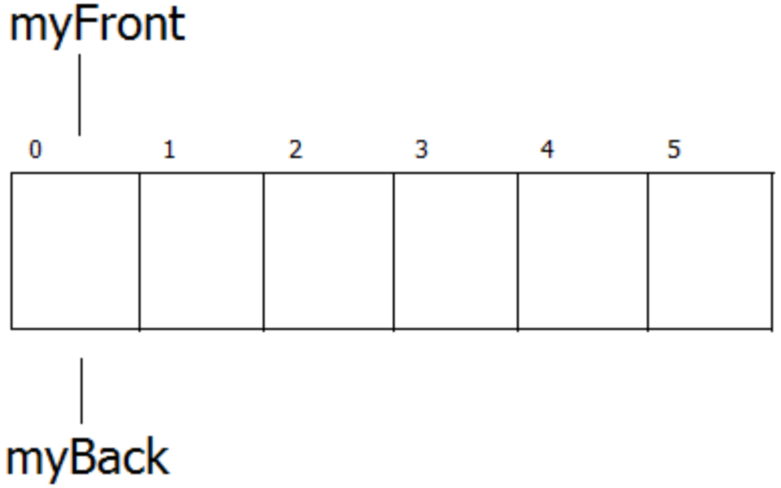


- Note additional variables needed
  - **`myFront, myBack`**
- Picture a queue object like this

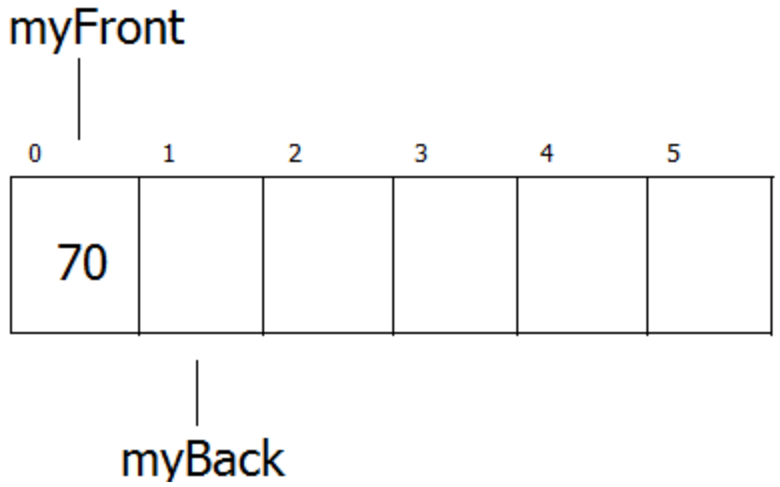


Queue Operation

- Empty Queue

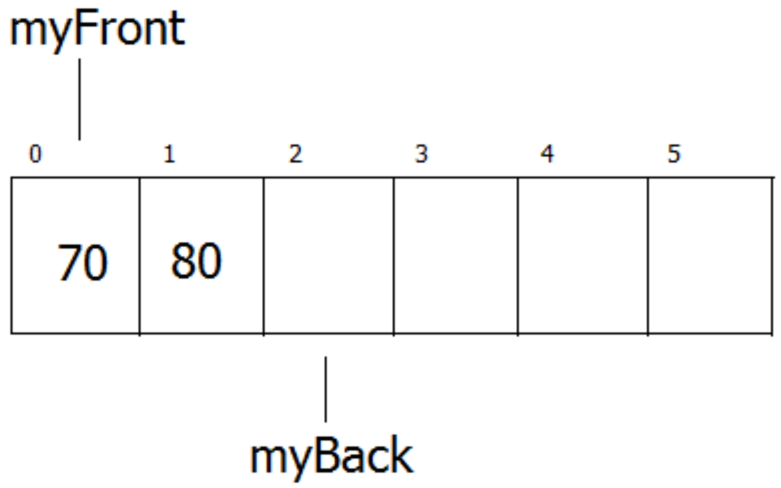


Enqueue(70)

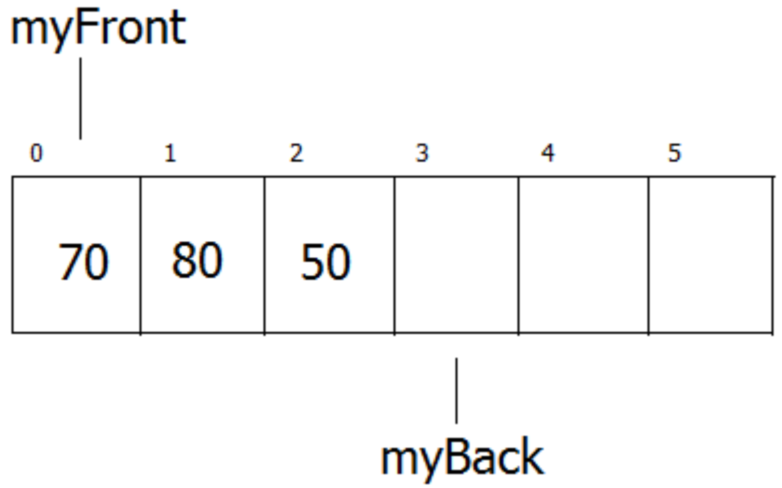


Queue Operation

- Enqueue(80)

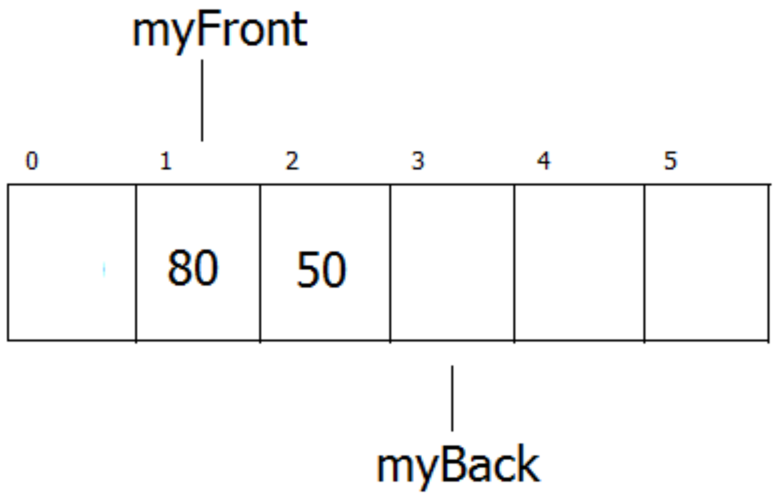


- Enqueue(50)

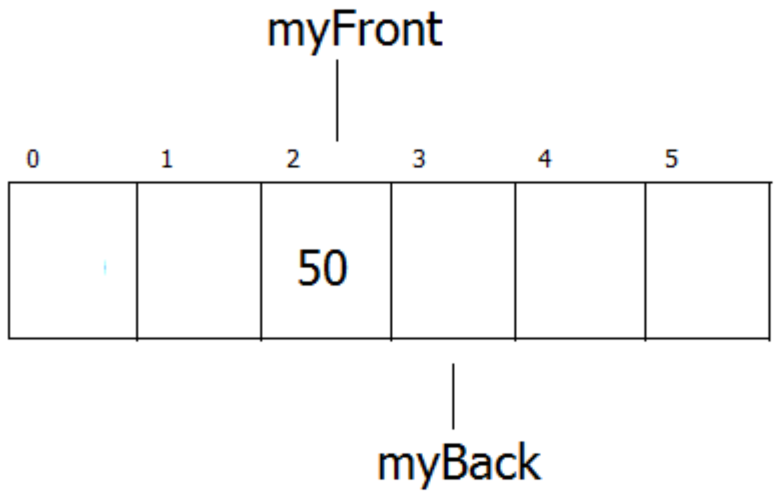


Queue Operation

- Dequeue()

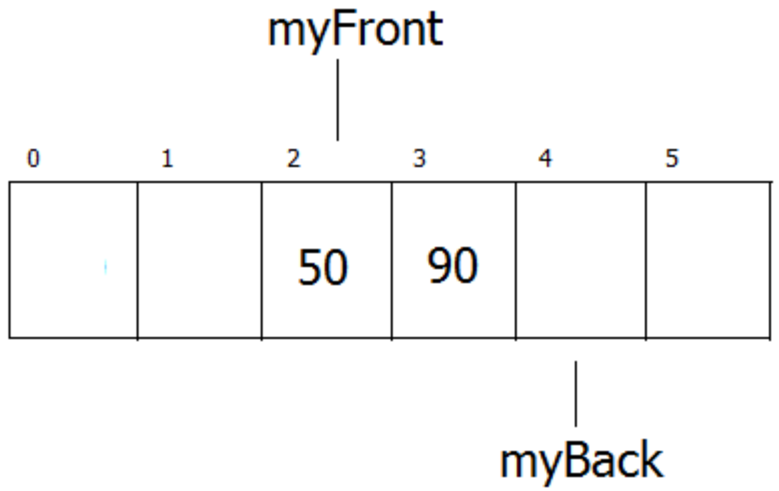


- Dequeue()

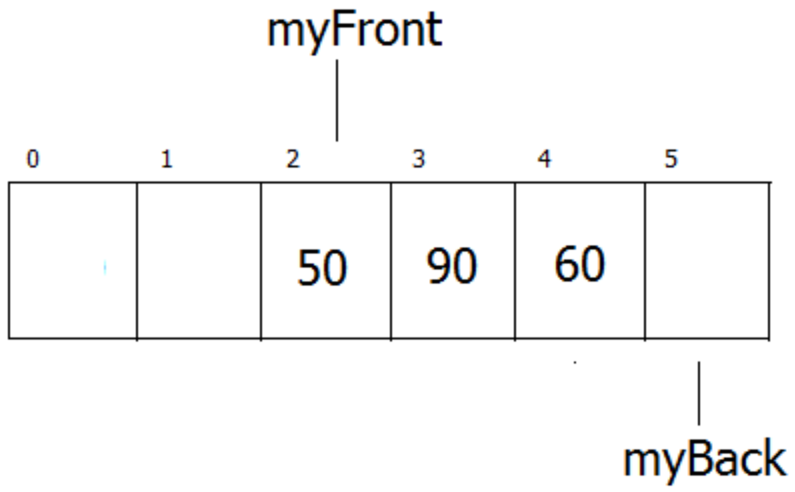


Queue Operation

- Enqueue(90)



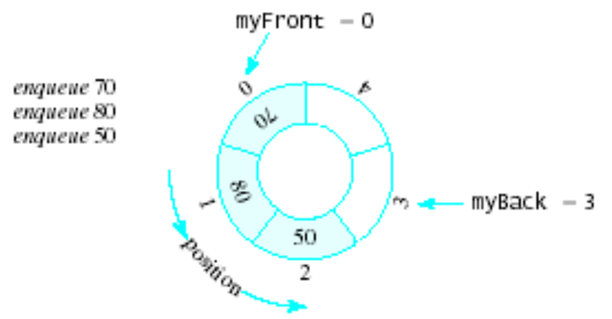
- Enqueue(60)



Circular Queue

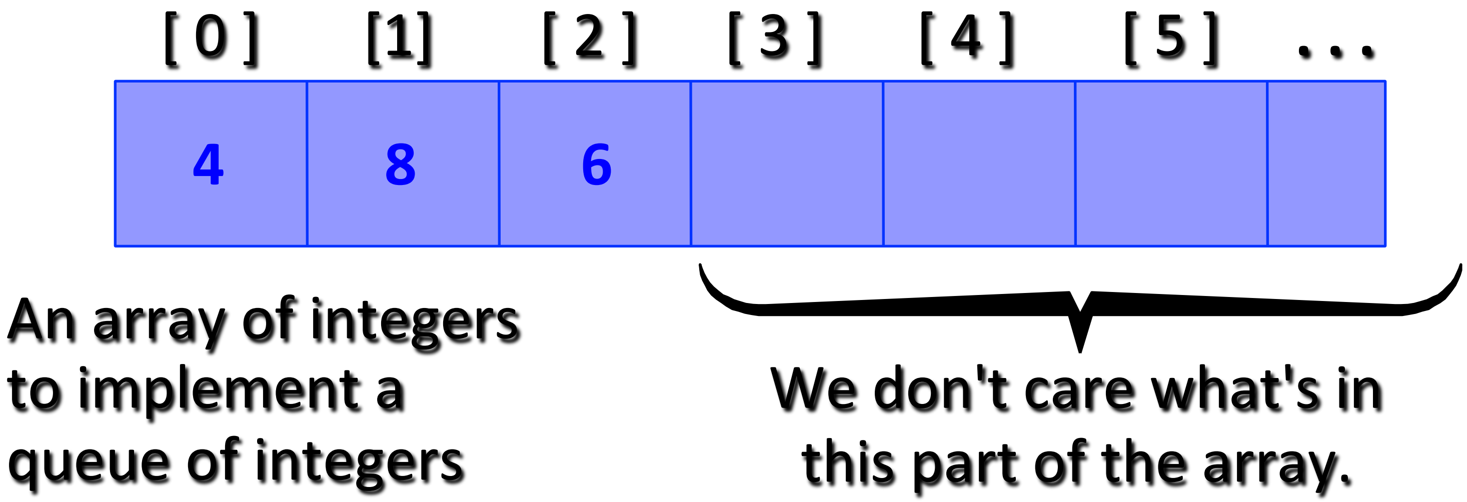
- Problems
  - We quickly "walk off the end" of the array
- Possible solutions
  - Shift array elements
  - Use a circular queue

  - Note that both empty and full queue gives `myBack == myFront`



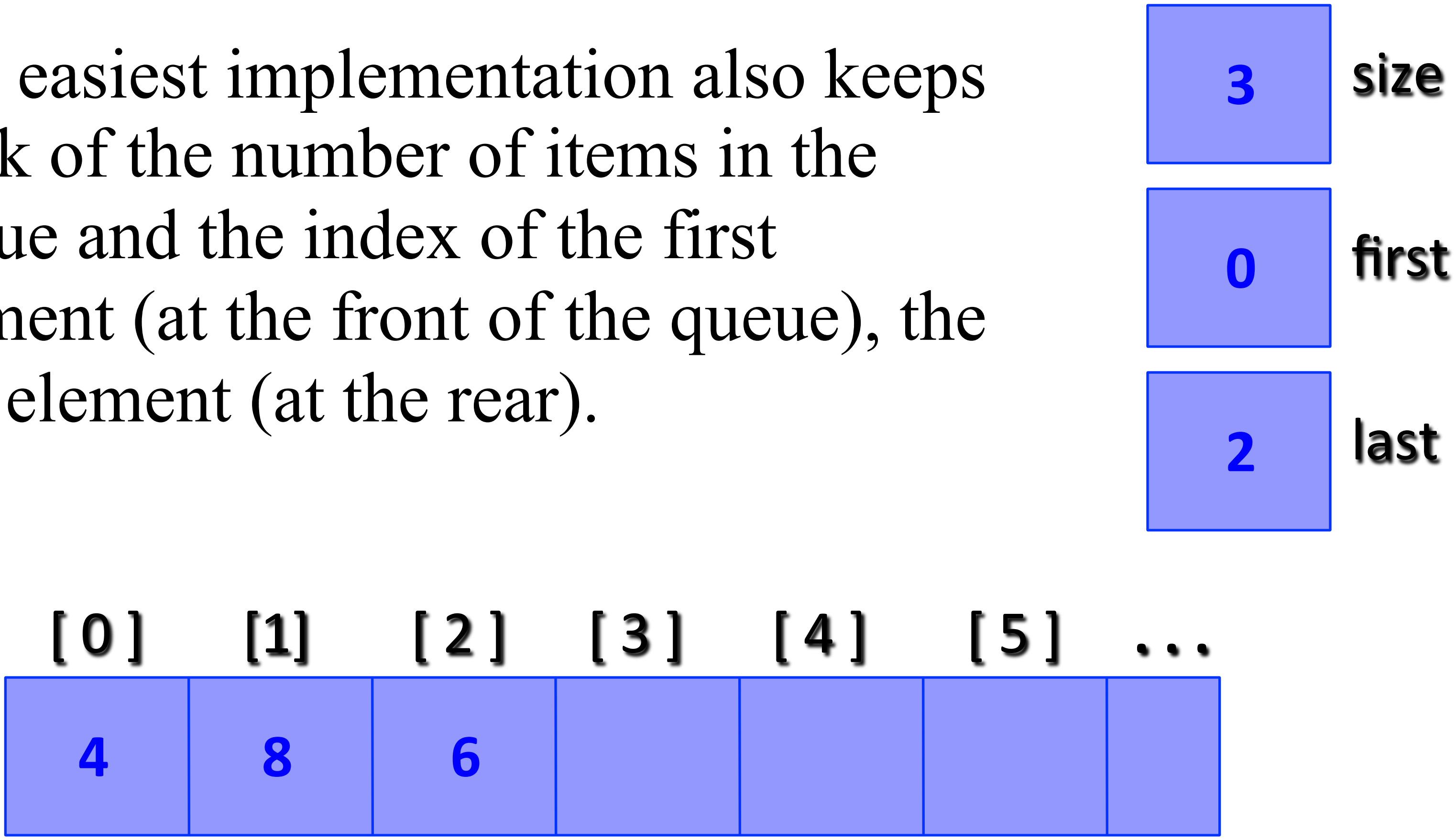
Array Implementation

- A queue can be implemented with an array, as shown here. For example, this queue contains the integers 4 (at the front), 8 and 6 (at the rear).


[0]
[1]
[2]
[3]
[4]
[5]
. . .
4
8
6
An array of integers to implement a queue of integers
We don't care what's in this part of the array.

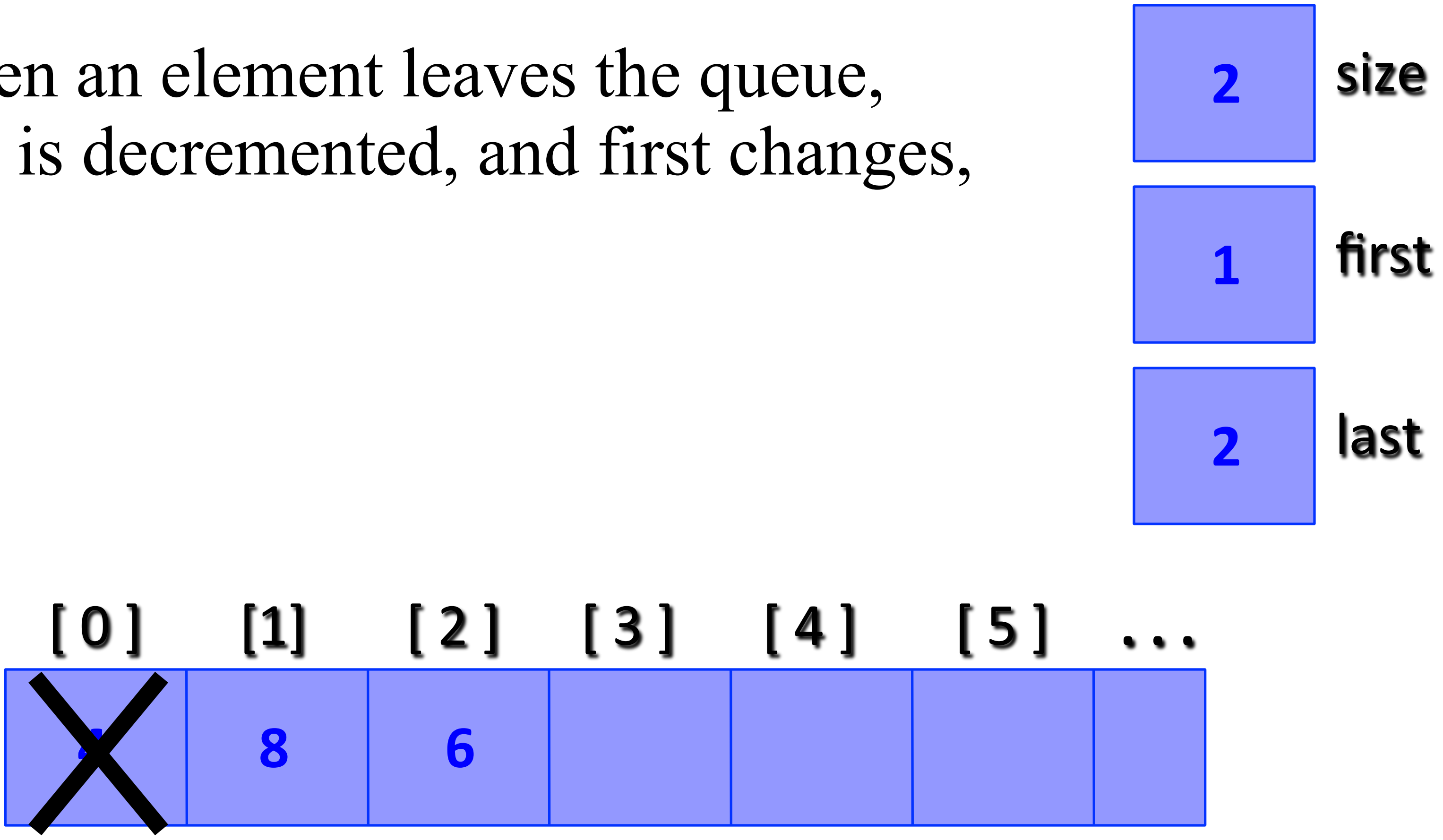
Array Implementation

- The easiest implementation also keeps track of the number of items in the queue and the index of the first element (at the front of the queue), the last element (at the rear).



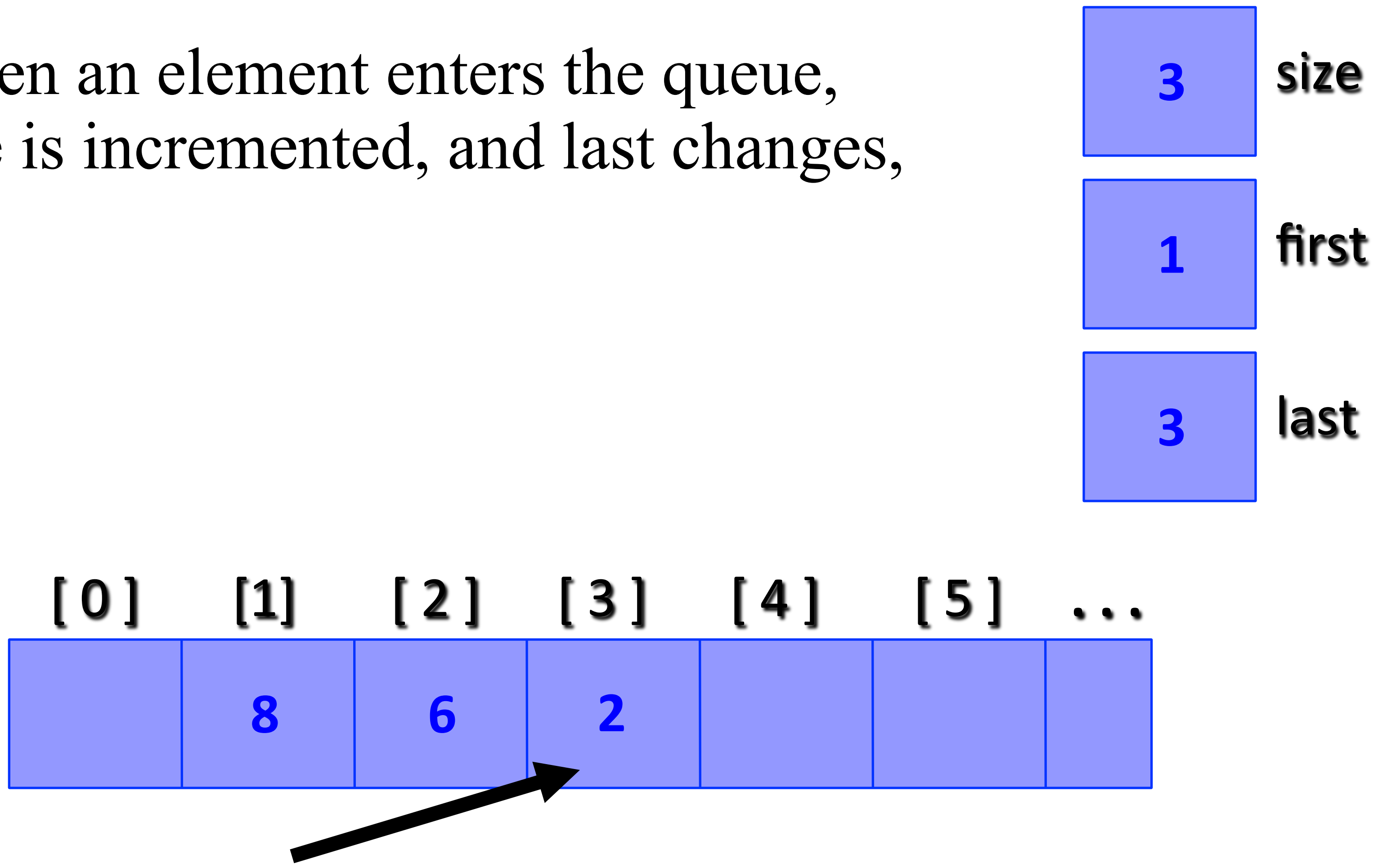
A Dequeue Operation

- When an element leaves the queue, size is decremented, and first changes, too.



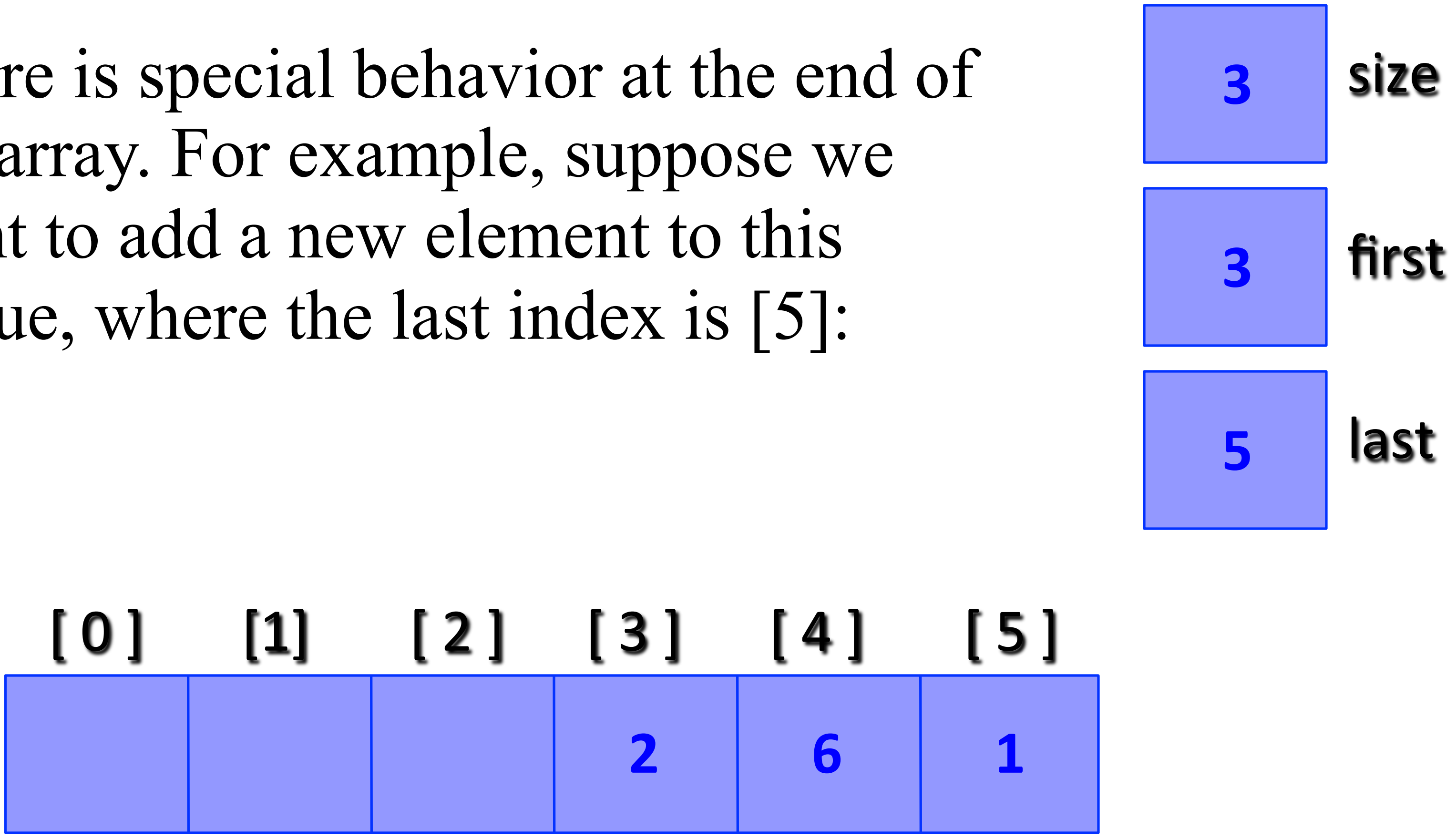
An Enqueue Operation

- When an element enters the queue, size is incremented, and last changes, too.



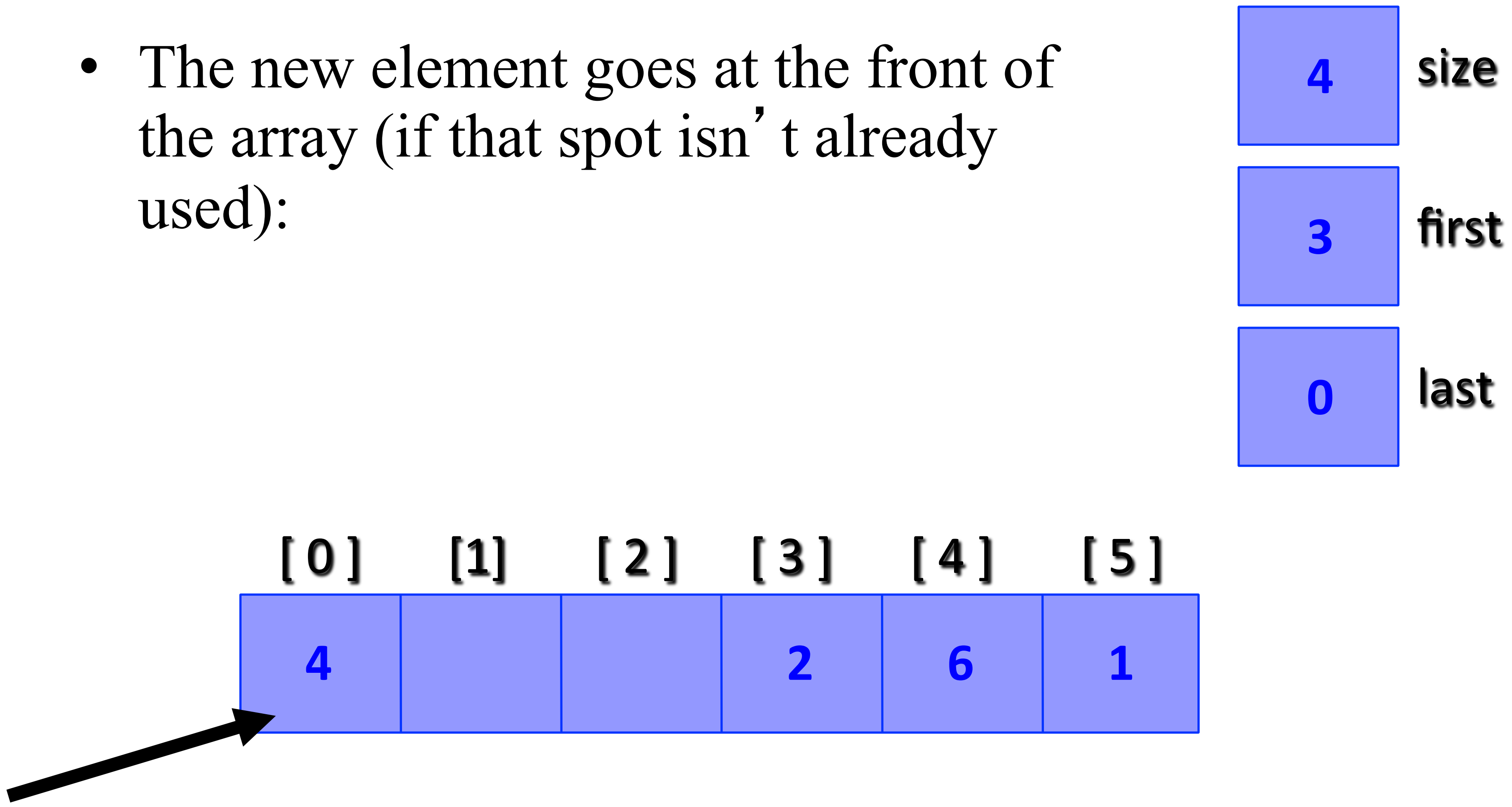
At the End of the Array

- There is special behavior at the end of the array. For example, suppose we want to add a new element to this queue, where the last index is [5]:



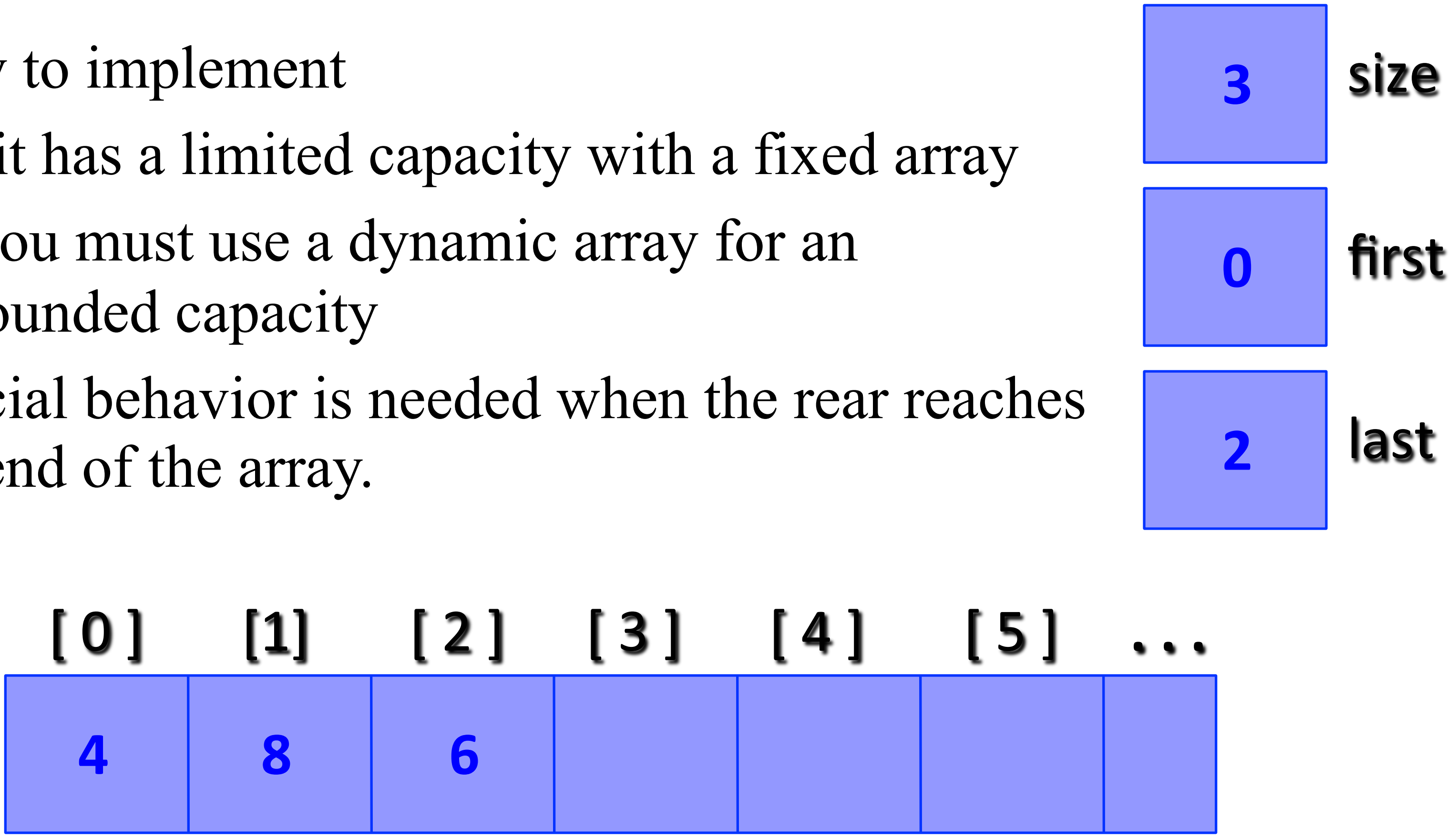
At the End of the Array

- The new element goes at the front of the array (if that spot isn't already used):



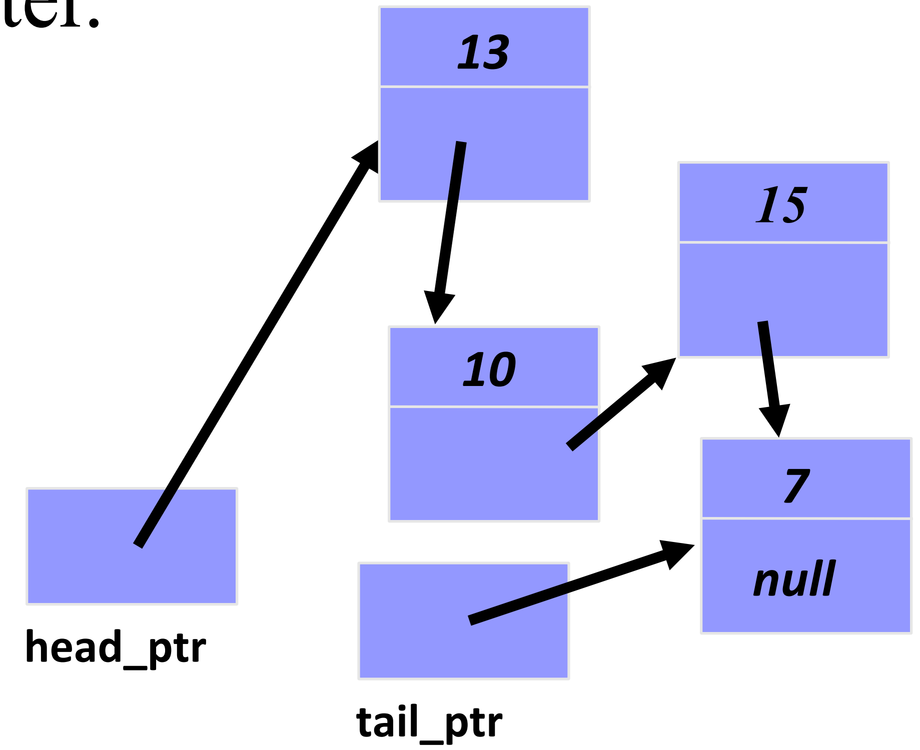
Array Implementation

- Easy to implement
- But it has a limited capacity with a fixed array
- Or you must use a dynamic array for an unbounded capacity
- Special behavior is needed when the rear reaches the end of the array.



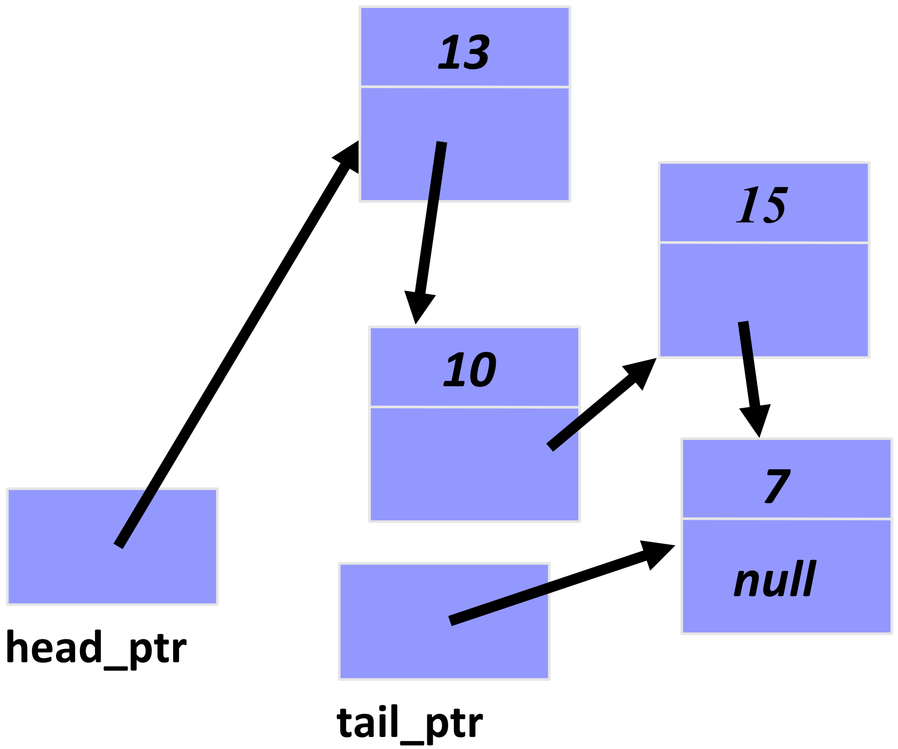
Linked List Implementation

- A queue can also be implemented with a linked list with both a head and a tail pointer.



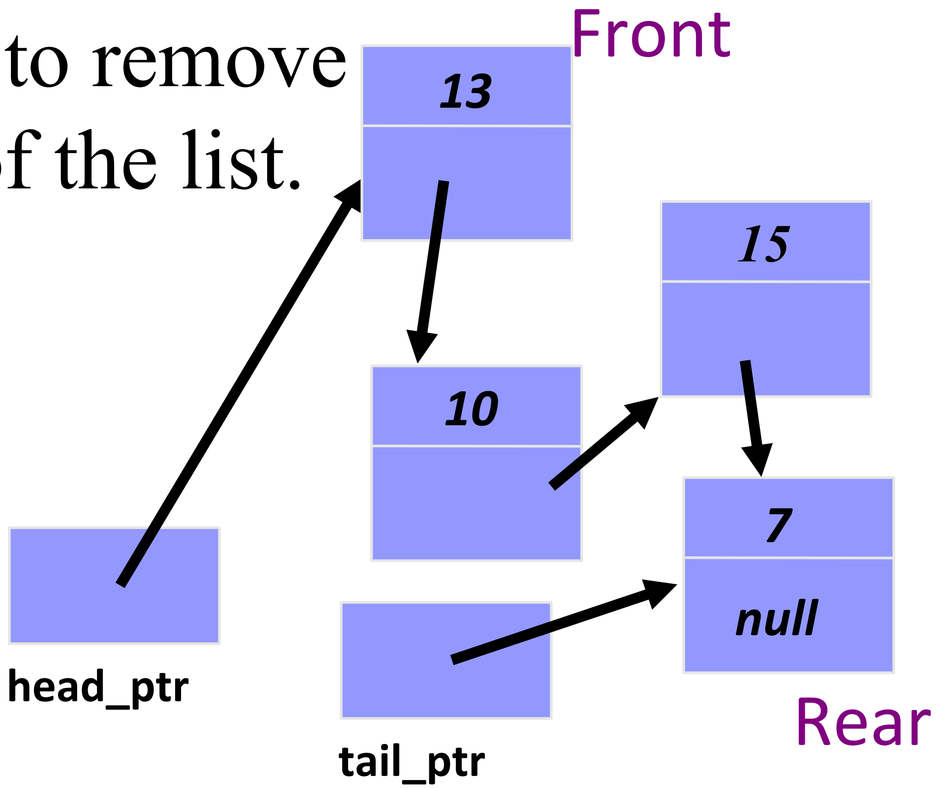
Linked List Implementation

- Which end do you think is the front of the queue? Why?



Linked List Implementation

- The head_ptr points to the front of the list.
- Because it is harder to remove items from the tail of the list.



Abstract Data Type

- Abstract Data Type as a design tool
- Concerns only on the important concept or model
- No concern on implementation details.
- Stack & Queue is an example of ADT
- An array is not ADT.

What is the difference?

- Stack & Queue vs. Array
  - Arrays are data storage structures while stacks and queues are specialized DS and used as programmer's tools.
- Stack – a container that allows push and pop
- Queue - a container that allows enqueue and dequeue
- No concern on implementation details.
- In an array any item can be accessed, while in these data structures access is restricted.
- They are more abstract than arrays.

Questions?

- Array is not ADT
- Is Linked list ADT?