CSE 332

JUNE 19– COURSE INTRODUCTIONS; ADTS; STACKS AND QUEUES

WELCOME!

- Administrative Minutiae
- Course Objectives
- Review of Stacks and Queues
- Abstract Data Types (ADT)

COURSE INFO

- **Evan McCarty (ejmcc@uw.edu)**
- **Office hours (CSE 214)**
  - Mondays: 11:00 – 12:00
  - Wednesdays: 11:00 – 12:00
  - By appointment or over email

COURSE STAFF

- **TAs**
  - Jefferson – Section Thur 9:40-10:40
  - Alon – Grading and Office hours

HOMEWORK

- **Homework will be assigned throughout the quarter.**

HOMEWORK

- **Homework will be assigned throughout the quarter.**
- **There will be lots of assignments, all with different due dates, but they should all be fairly simple.**

HOMEWORK

- **Homework will be assigned throughout the quarter.**
- **There will be lots of assignments, all with different due dates, but they should all be fairly simple.**
- **Some are more difficult than others, however, so it's good to look at them in advance**

HOMEWORK

- **This course focuses not only on programming and implementation, but also on theory and understanding**

HOMEWORK

- This course focuses not only on programming and implementation, but also on theory and understanding
- Homework and Projects are likely to reflect this

HOMEWORK

- **Academic honesty**
  - High level discussion
  - Fully understand submission

- **Reasonable effort and office hours**

PROJECTS

- In addition to exercises which will go out throughout the quarter, there will also be three projects, which you will complete in teams of two (or three if the class has an odd size)

PROJECTS

- In addition to exercises which will go out throughout the quarter, there will also be three projects, which you will complete in teams of two (or three if the class has an odd size)
- First project will be out before class on Wednesday, so start thinking about teams and work schedules now.

LECTURES

- **Lecture slides will be posted online after class**

LECTURES

- Lecture slides will be posted online after class
- Questions are strongly encouraged

LECTURES

- Lecture slides will be posted online after class
- Questions are strongly encouraged
- All material fair game for exams

LECTURES

- Lecture slides will be posted online after class
- Questions are strongly encouraged
- All material fair game for exams
- Weiss textbook

LECTURES

- **Lecture slides will be posted online after class**
- **Questions are strongly encouraged**
- **All material fair game for exams**
- **Weiss textbook**
  - Mostly a reference, but it's good for when an explanation isn't good enough from me or the TAs

LECTURES

- **Because summer quarter has lectures that are 10 minutes longer, we will often use those 10 minute periods for practice problems and project check-ins**

LECTURES

- Because summer quarter has lectures that are 10 minutes longer, we will often use those 10 minute periods for practice problems and project check-ins
- Attendance is mandatory on days where a project check-in is scheduled

SECTION

- Conducted by Jefferson

SECTION

- Conducted by Jefferson
- QuickCheck

SECTION

- **Conducted by Jefferson**
- **QuickCheck**
  - Start section with a 5-10 minute problem for you to work on alone

SECTION

- **Conducted by Jefferson**
- **QuickCheck**
  - Start section with a 5-10 minute problem for you to work on alone
  - Good indication of the type of problem we expect you to have learned from that week and the speed we expect you to be able to solve it

SECTION

- **Conducted by Jefferson**
- **QuickCheck**
  - Start section with a 5-10 minute problem for you to work on alone
  - Good indication of the type of problem we expect you to have learned from that week and the speed we expect you to be able to solve it
  - Ungraded

EXAMS (TENTATIVE)

- **Midterm exam**
  - 9:40– 10:40; Friday, July 21
- **Final Exam**
  - 9:40 – 10:40; Friday, Aug 18

EXAMS (TENTATIVE)

- **Midterm exam**
  - 9:40– 10:40; Friday, July 21
- **Final Exam**
  - 9:40 – 10:40; Friday, Aug 18
- **Both conducted in lecture session.**

BEFORE HW COMES OUT

- **Make sure you've properly set up the JDK**

BEFORE HW COMES OUT

- Make sure you've properly set up the JDK
- Also, get and install the latest version of eclipse, it is pivotal for the version control you will use on your projects

DATA STRUCTURES AND PARALLELISM

- Understand and recognize behavior of key data structures

DATA STRUCTURES AND PARALLELISM

- Understand and recognize behavior of key data structures
- Understand and solve common data structure problems

DATA STRUCTURES AND PARALLELISM

- Understand and recognize behavior of key data structures
- Understand and solve common data structure problems
- Analyze operations and algorithms

DATA STRUCTURES AND PARALLELISM

- Understand and recognize behavior of key data structures
- Understand and solve common data structure problems
- Analyze operations and algorithms
- Implement data structures and understand design trade-offs

DATA STRUCTURES AND PARALLELISM

- Understand and recognize behavior of key data structures
- Understand and solve common data structure problems
- Analyze operations and algorithms
- Implement data structures and understand design trade-offs
- Understand concurrency and parallelism and how those impact outcomes and decisions

CSE 143

- **Object-oriented Programming**
  - Classes and Inheritance
  - Methods, variables and conditions
  - Loops and recursion
  - Linked lists and simple trees
  - Basic Sorting and Searching
  - Concepts of Analysis O(n) v $O(n^2)$
  - Client v. Implementer

CSE 332

- Design decisions
- Critical thinking
- Implementations
- Debugging and Testing
- Abstract Data Types

ABSTRACTION

- **Software engineering**
  v. Computer Science
- **Applicable across languages and implementations**
- **Behavior focus**
  - How can you recognize an ADT?

STACK?

- What is a stack?

STACK?

- **What is a stack?**
  - Outside of CS?

STACK?

- **What is a stack?**
  - Outside of CS?
  - From 143?

DEFINITIONS

- **Abstract Data Type (ADT)**
  - Operations and expected behavior
- **Data Structure**
  - Specific organization of data
  - Can be analyzed
- **Implementation**
  - Language specific application

DESIGN DECISIONS

- **Between an ADT and its implementation, there are design decisions**
- **Constraints of the problem**
  - Memory v. Speed
  - One function v. another
  - Generality v. Specificity

DESIGN DECISIONS

- **Linked List v Array**

DESIGN DECISIONS

- **Linked List v Array**
  - Overhead
  - Memory use
  - Adding to middle
  - Traversal
  - Insertion

DESIGN DECISIONS

- **Shopping list?**

DESIGN DECISIONS

- **Shopping list?**
  - What sorts of behavior do shoppers exhibit?
  - What constraints are there on a shopper?
  - What improvements would make a better shopping list?

DESIGN DECISIONS

- **Shopping list?**
- **Stack?**

DESIGN DECISIONS

- **Shopping list?**
- **Stack?**
  - What sorts of behavior does the ‘stack’ support?
  - What constraints are there on a stack user?

    (Is there a change in certainty?)
  - What improvements would make a better stack?

    (What problems might arise in a stack?)

STACK ADT

- **Important to know *exactly* what we expect from a stack.**

STACK ADT

- **Important to know *exactly* what we expect from a stack.**
  - Push(Object a) returns null; *(other options?)*
  - Pop() returns Object a: where a is the element on ‘top’ of the stack; also removes a from the stack
  - Top() returns Object a: where a is the element on ‘top’ of the stack without removing that element from the stack

STACK ADT

- **Important to know *exactly* what we expect from a stack.**
  - Push(Object a) returns null; *(other options?)*
  - Pop() returns Object a: where a is the element on ‘top’ of the stack; also removes a from the stack
  - Top() returns Object a: where a is the element on ‘top’ of the stack without removing that element from the stack
  - How long will these operations take?

STACK ADT

- **Important to know *exactly* what we expect from a stack.**
  - Push(Object a) returns null; *(other options?)*
  - Pop() returns Object a: where a is the element on ‘top’ of the stack; also removes a from the stack
  - Top() returns Object a: where a is the element on ‘top’ of the stack without removing that element from the stack
  - How long will these operations take?

That depends on the Data Structure and Implementation

STACK ADT

- Array implementation
- Unique problems?

STACK ADT

- Array implementation
- Unique problems?

What if the array is full?

STACK ADT

- Array implementation
- Unique problems?

What if the array is full?

What if we alternate push() and pop()?

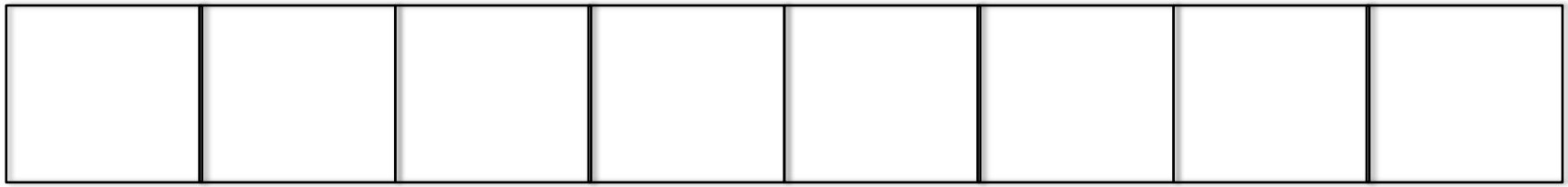
STACK ADT

- **Array implementation**
- **Unique problems?**
  - End of Array
- **Unique solutions?**

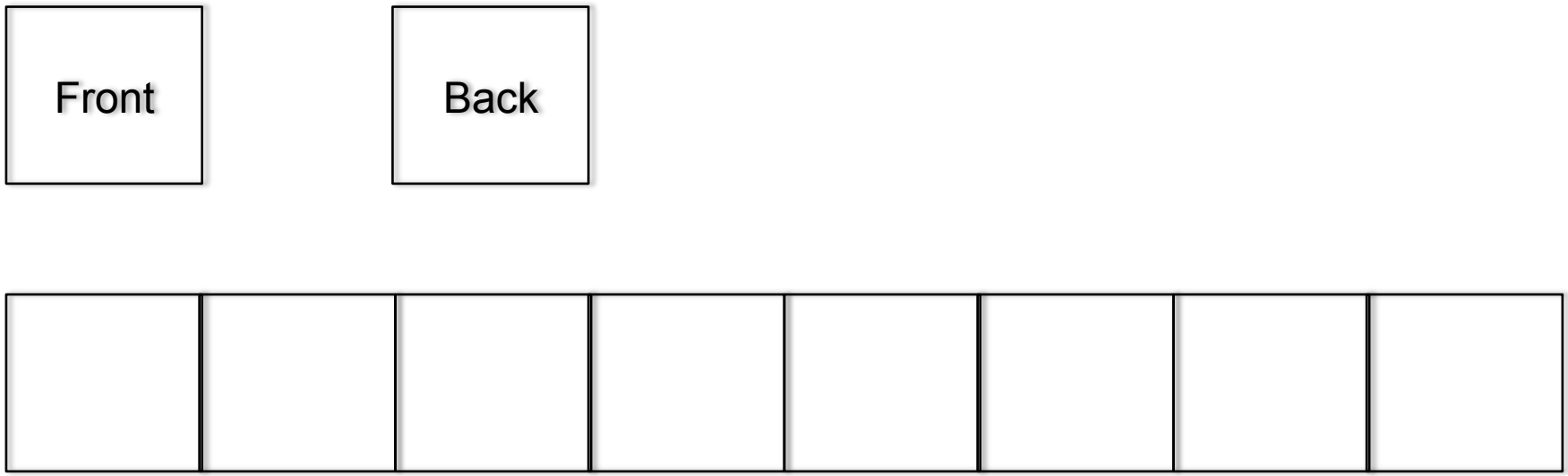
STACK ADT

- **Array implementation**
- **Unique problems?**
  - End of Array
- **Unique solutions?**
  - Resizing (costly!)
  - Circular Array (?)

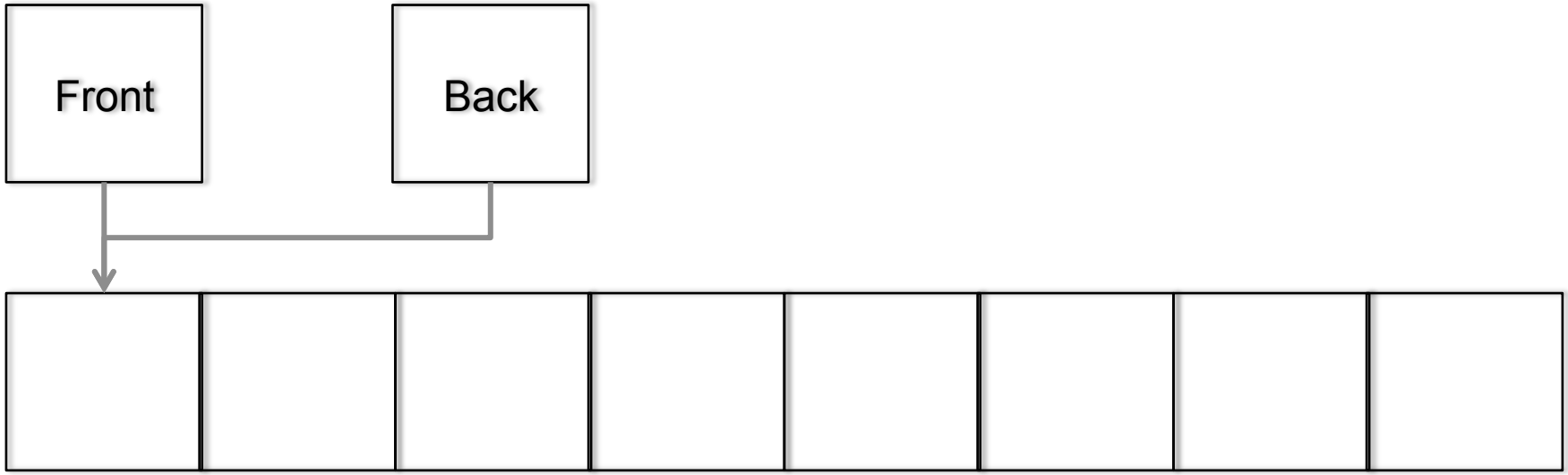
CIRCULAR QUEUES



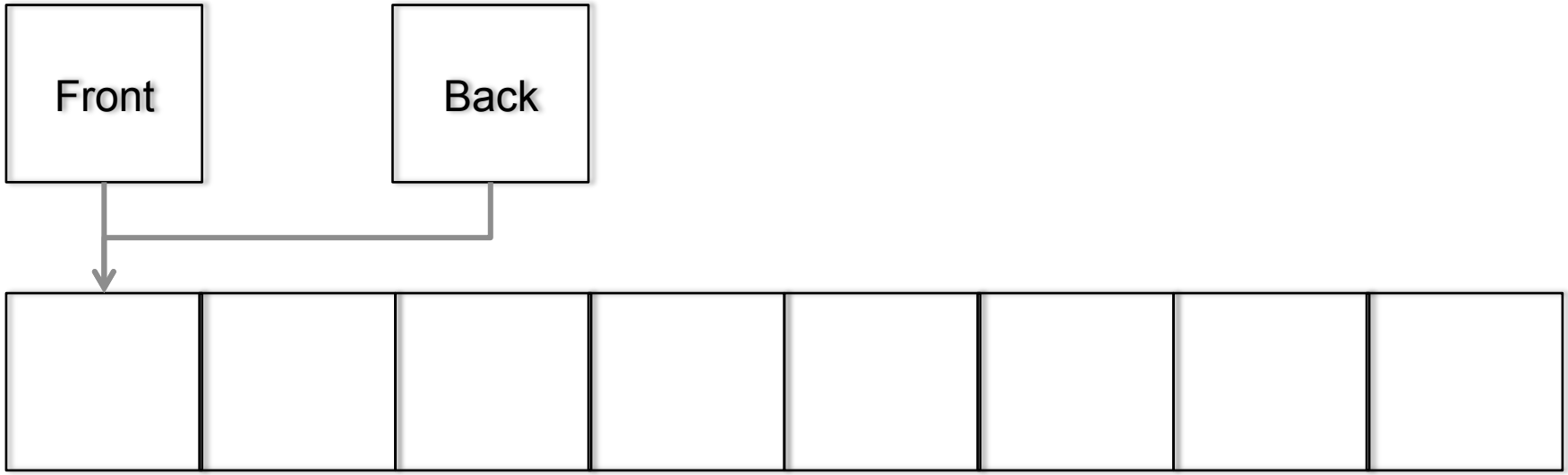
CIRCULAR QUEUES



CIRCULAR QUEUES



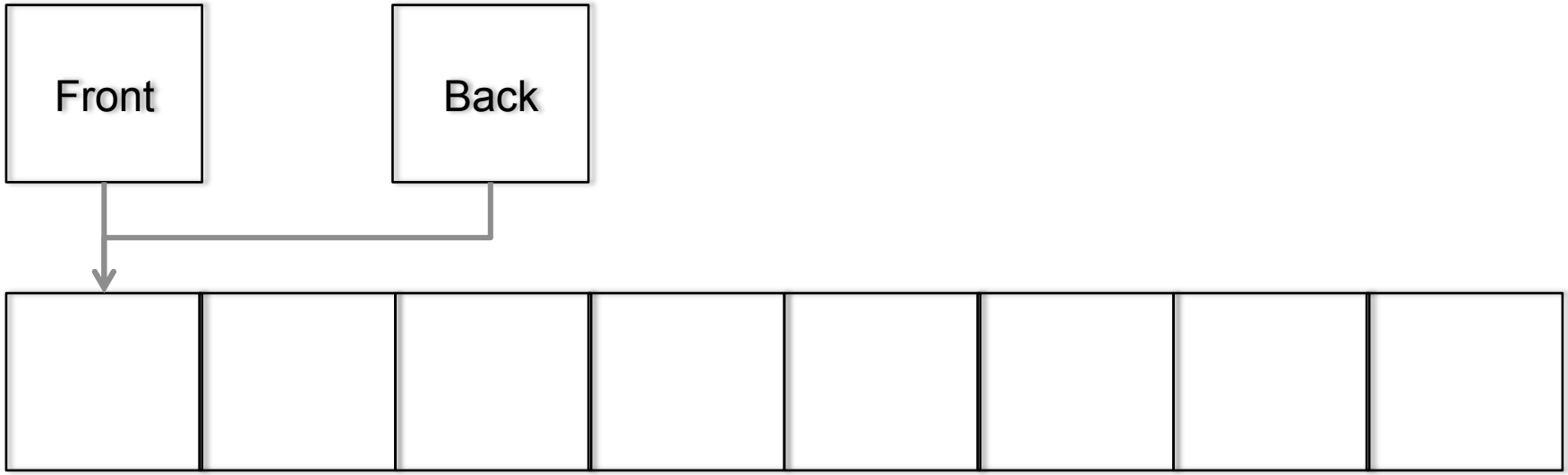
CIRCULAR QUEUES



Why this way?

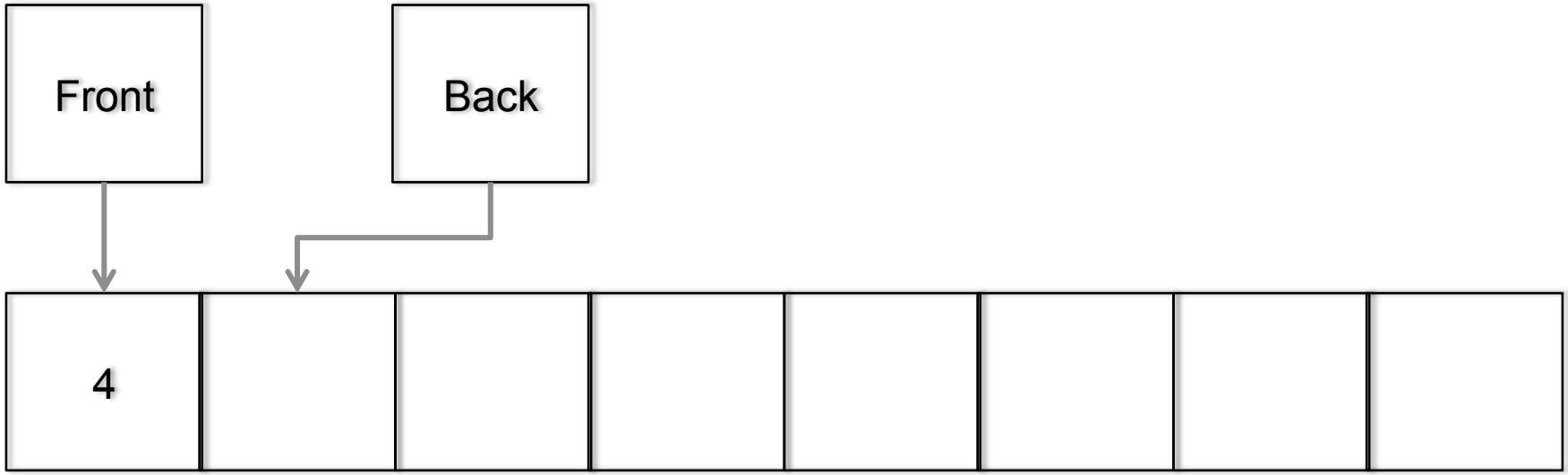
What function to front and back serve?

CIRCULAR QUEUES



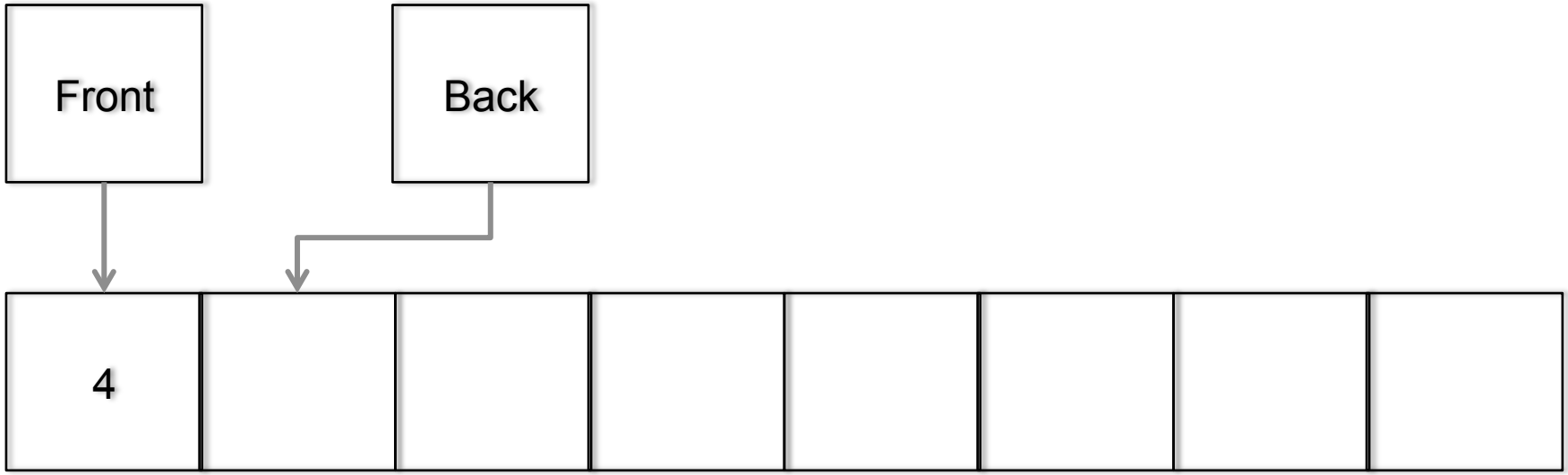
enqueue(4)

CIRCULAR QUEUES



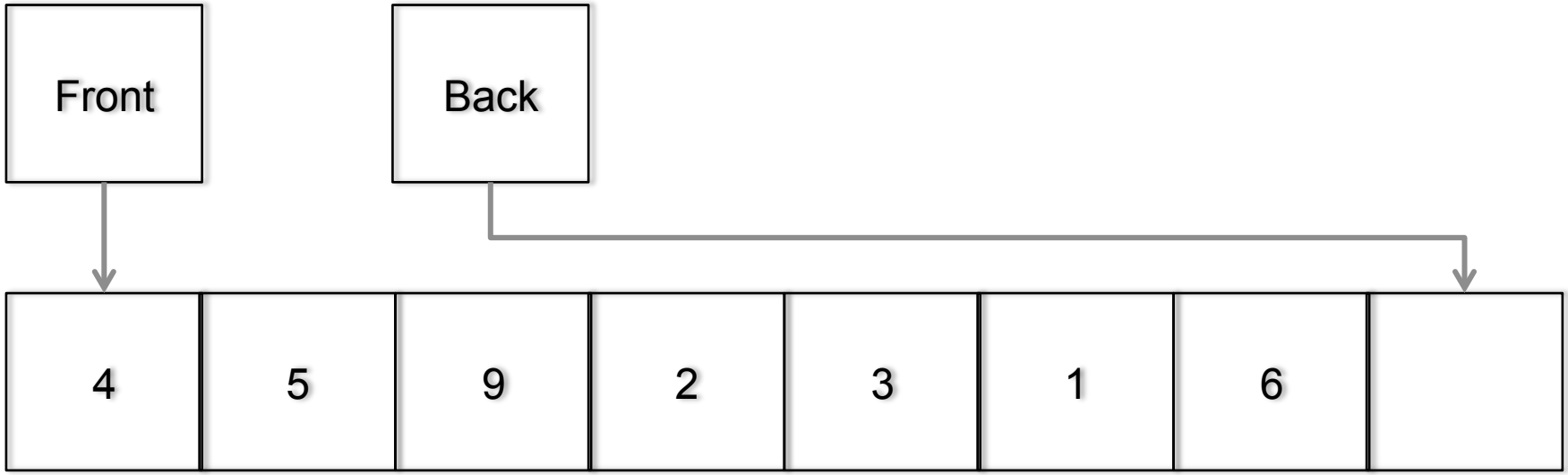
Which operations will move what pointers?

CIRCULAR QUEUES



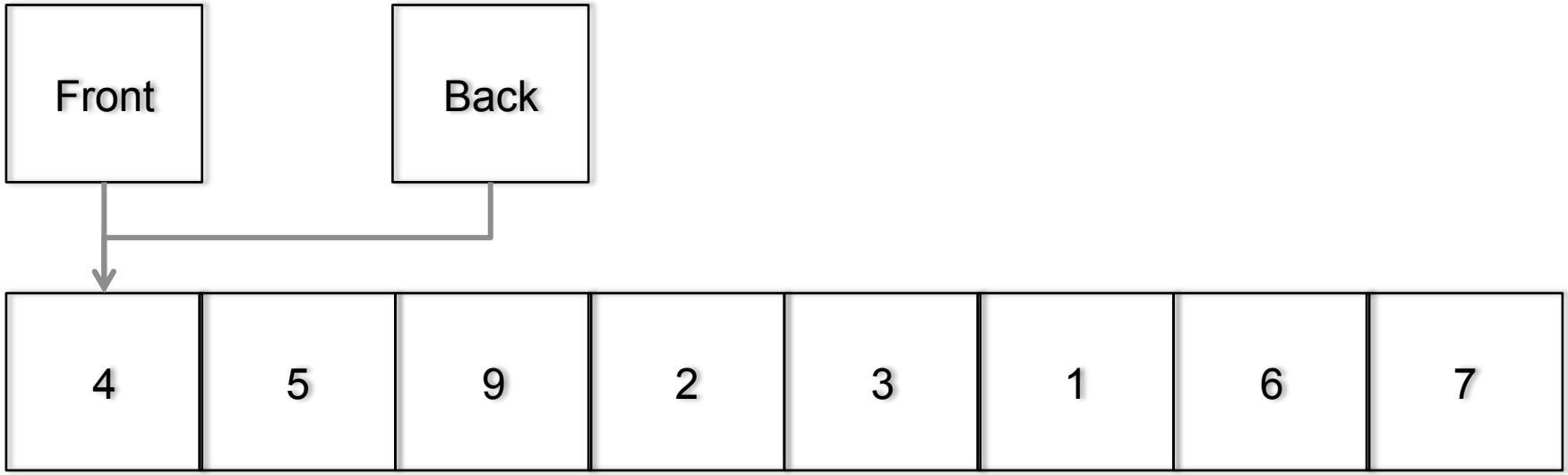
Let’s do several enqueues

CIRCULAR QUEUES



What happens now, on enqueue(7)?

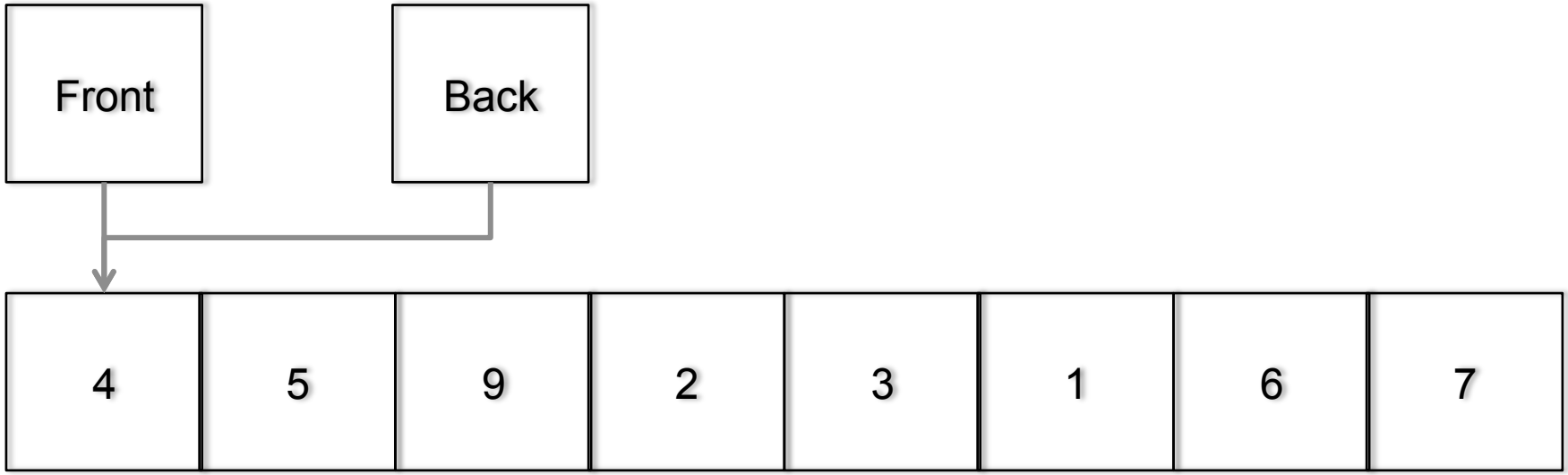
CIRCULAR QUEUES



Problems here?

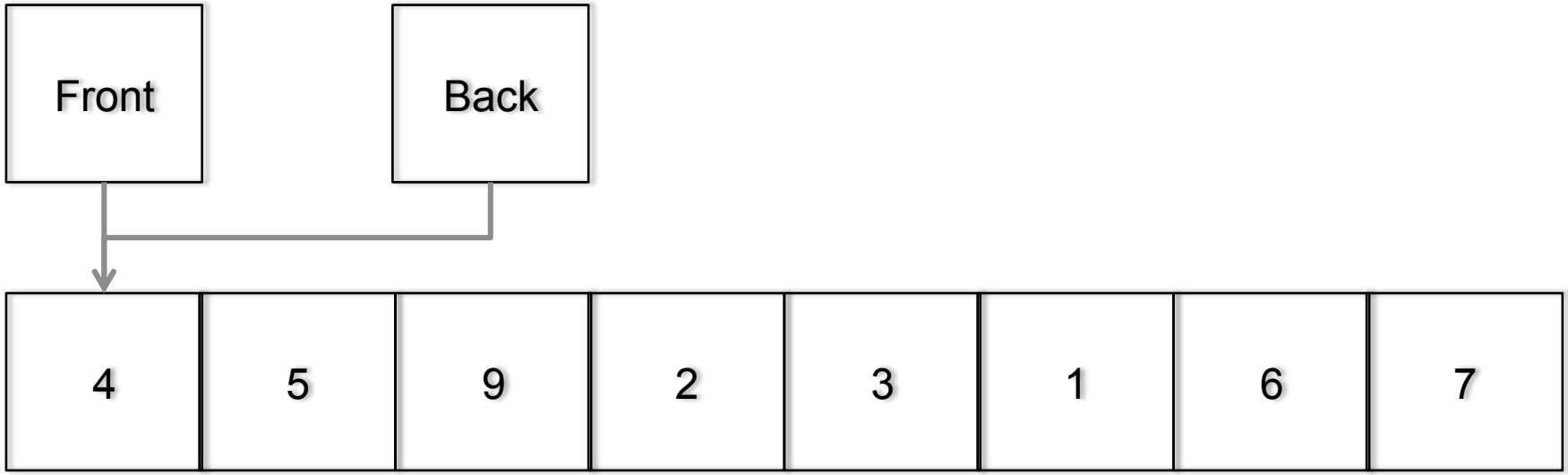
How to implement?

CIRCULAR QUEUES



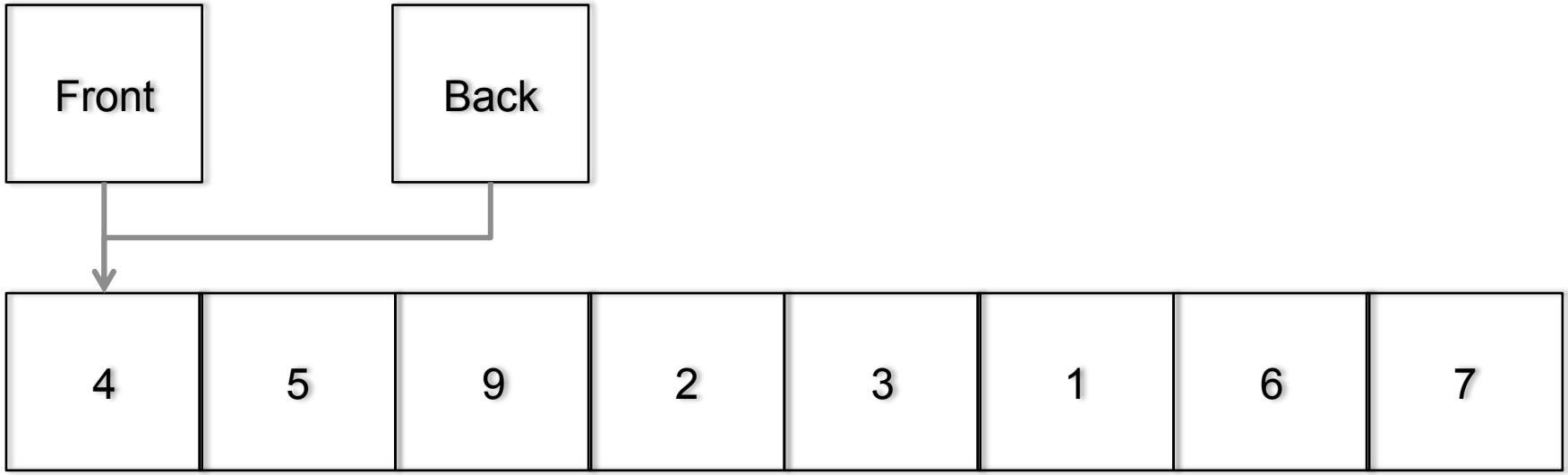
The queue is full, but it is the same situation (front == back) as when the queue is empty. This is a boundary condition.

CIRCULAR QUEUES



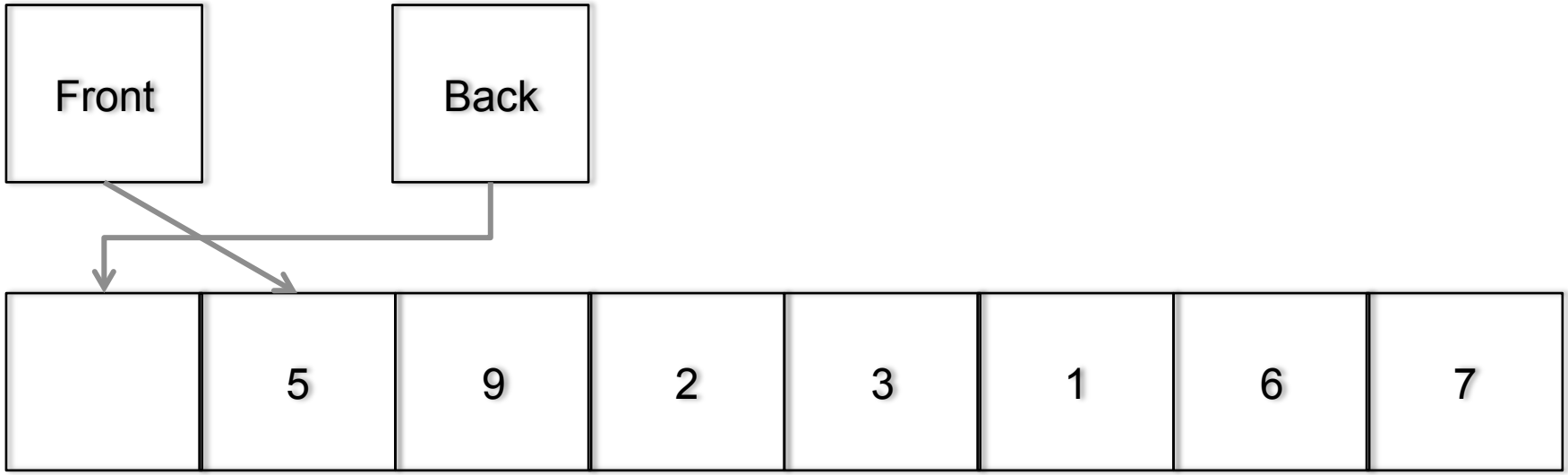
We have to resize the list (or deny the add) if we get another enqueue.

CIRCULAR QUEUES



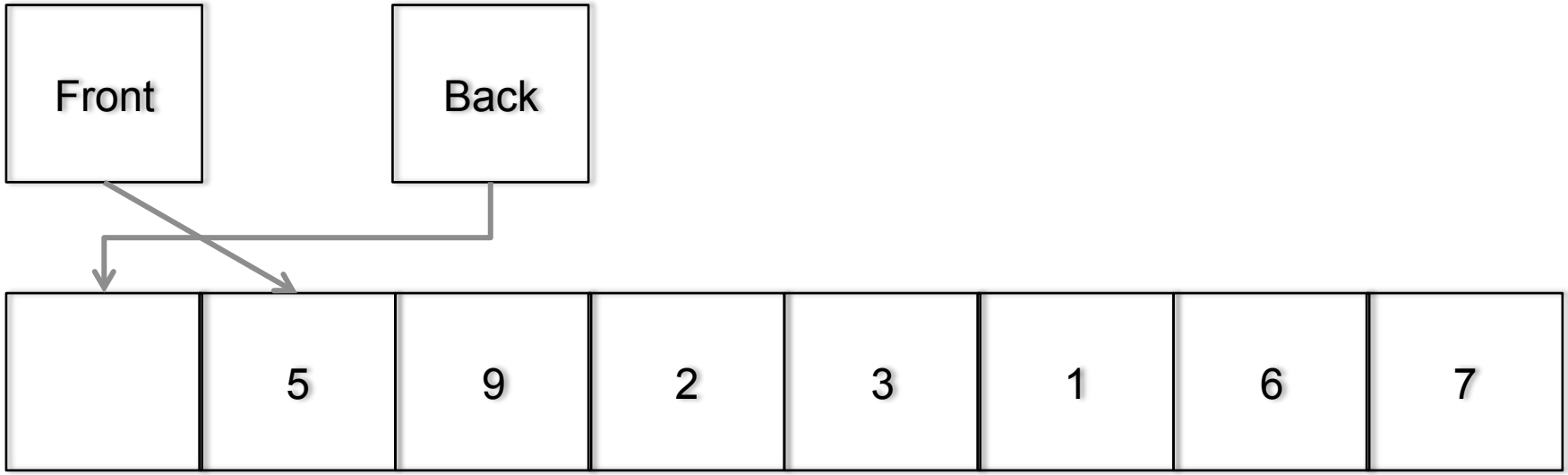
What if we dequeue some items?

CIRCULAR QUEUES



Dequeue() outputs 4

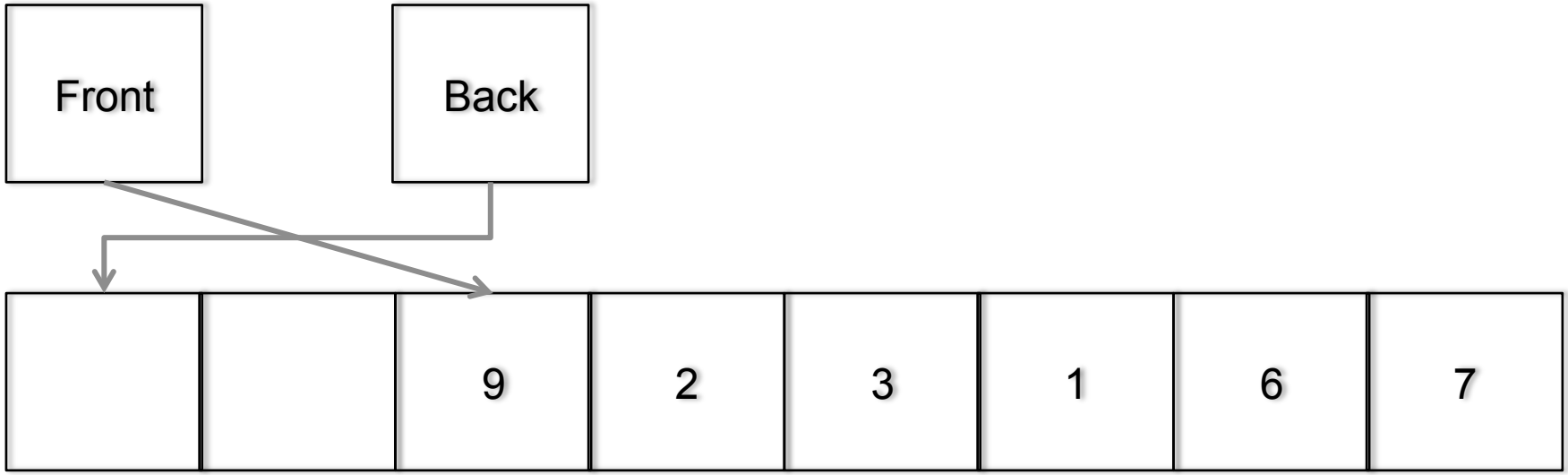
CIRCULAR QUEUES



Dequeue() outputs 4

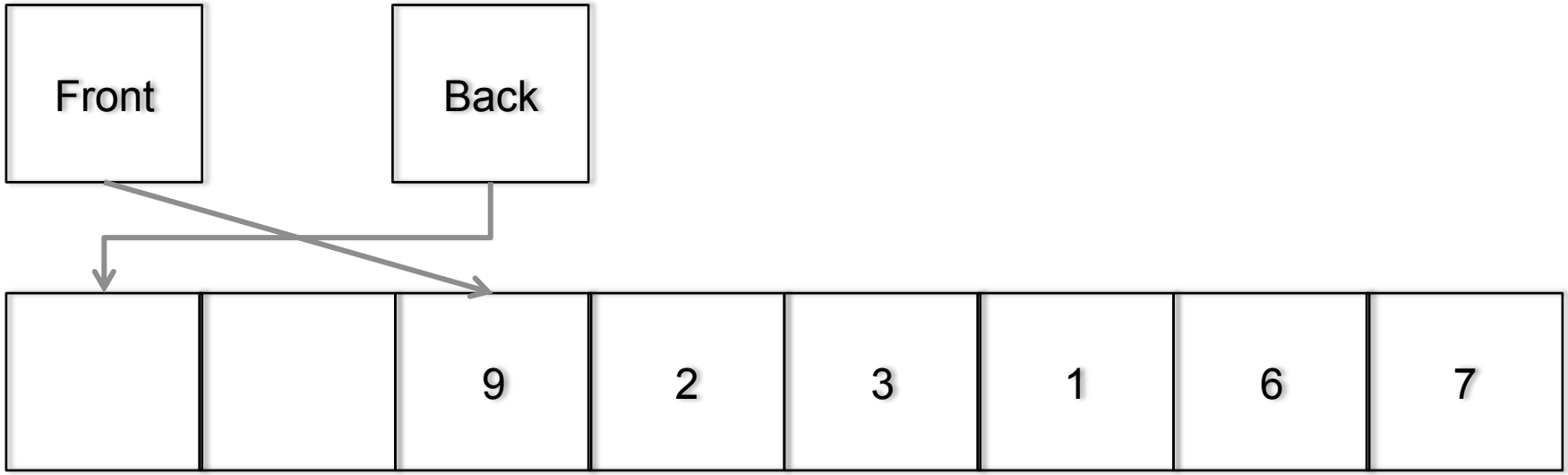
Is the 4 really “deleted”?

CIRCULAR QUEUES



Output 5

CIRCULAR QUEUES



Now we've freed up some space and can enqueue more

CIRCULAR QUEUES

- By moving the front and back pointers, we can utilize all of the space in the array
- Advantages over a linked list?

CIRCULAR QUEUES

- **By moving the front and back pointers, we can utilize all of the space in the array**
- **Advantages over a linked list?**
  - Fixed number of items
  - Small data (Memory efficiency)
- **BONUS: What is the memory overhead of the linked list?**