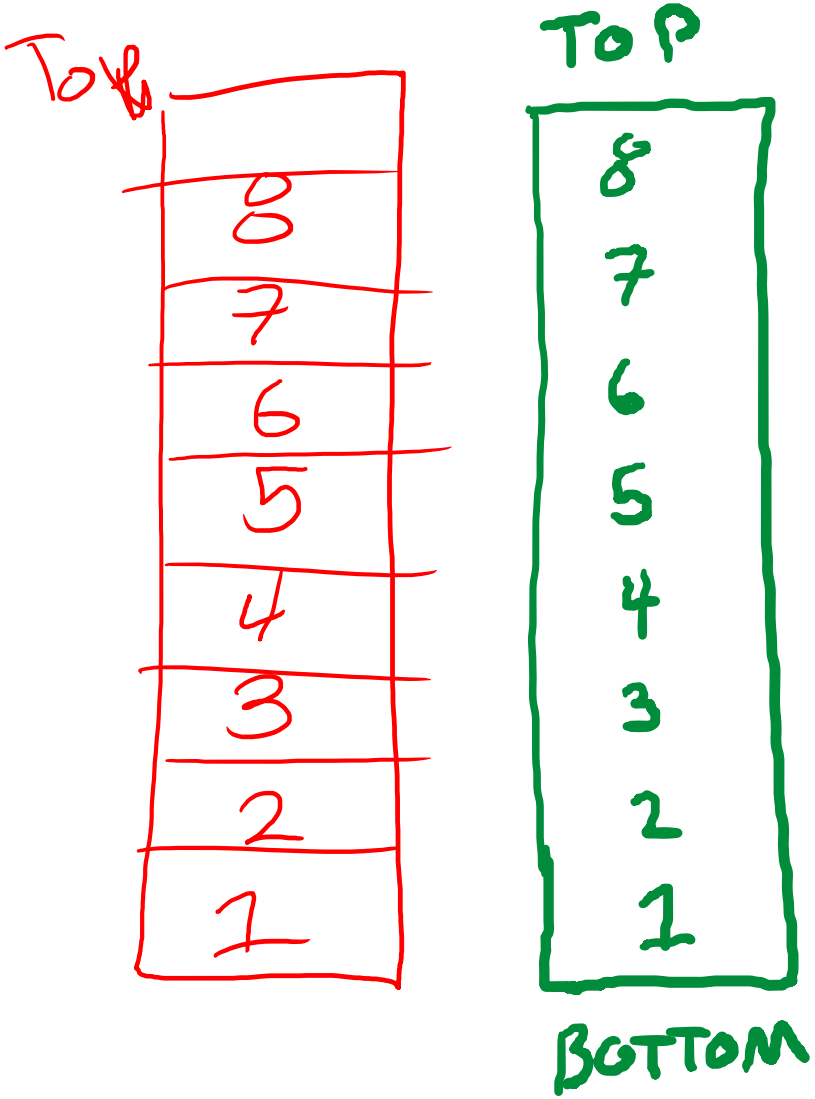
Lecture 3: Queues, Testing, and Working with Code

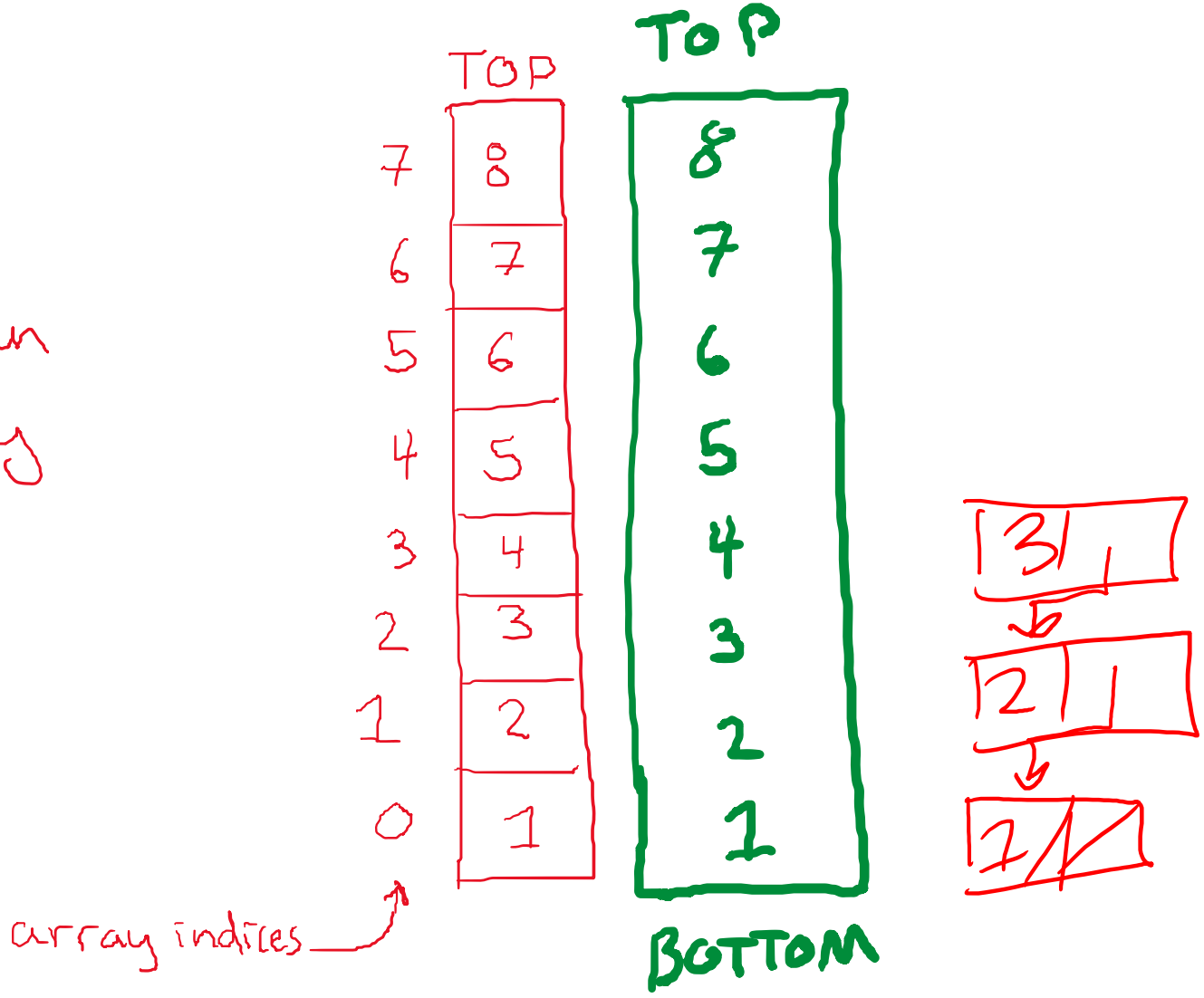
Data Structures and Algorithms

Clearing up Stacks



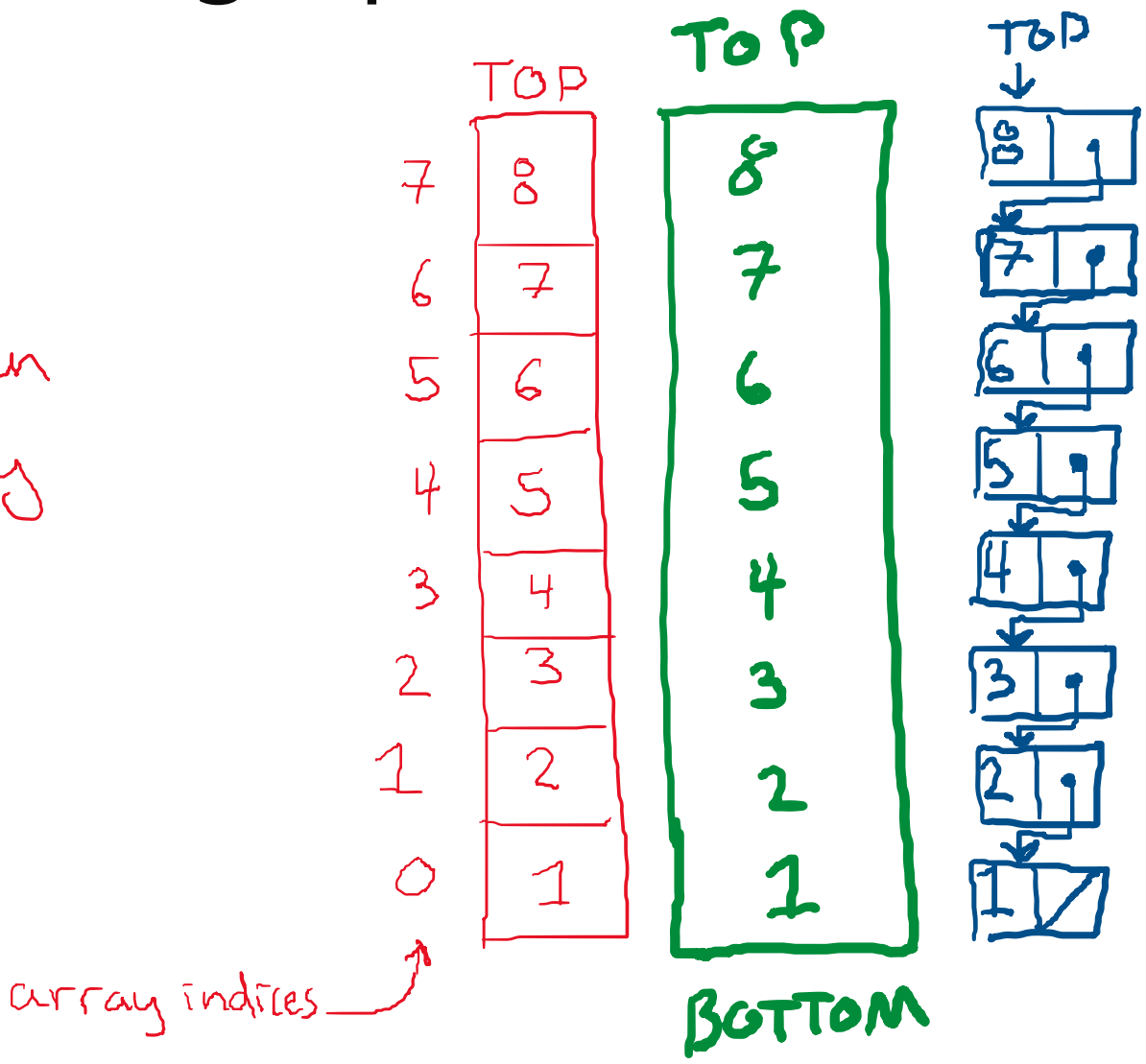
Clearing up Stacks

As an array



Clearing up Stacks

As an array

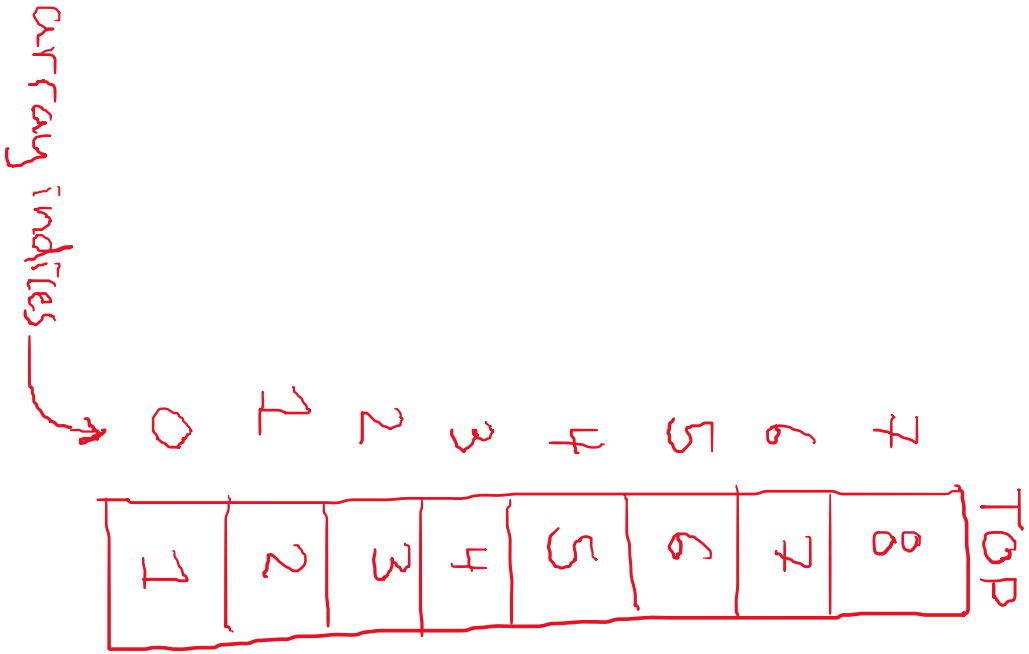


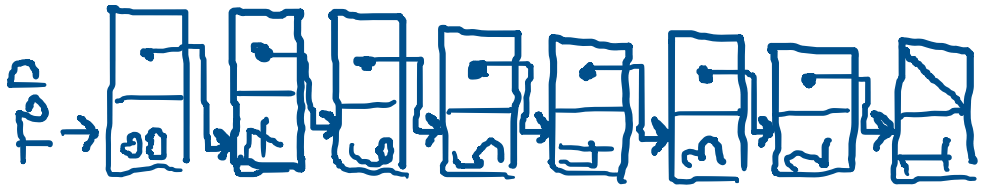
As a linked list

Clearing up Stacks (push them over)

As an array

As a linked list

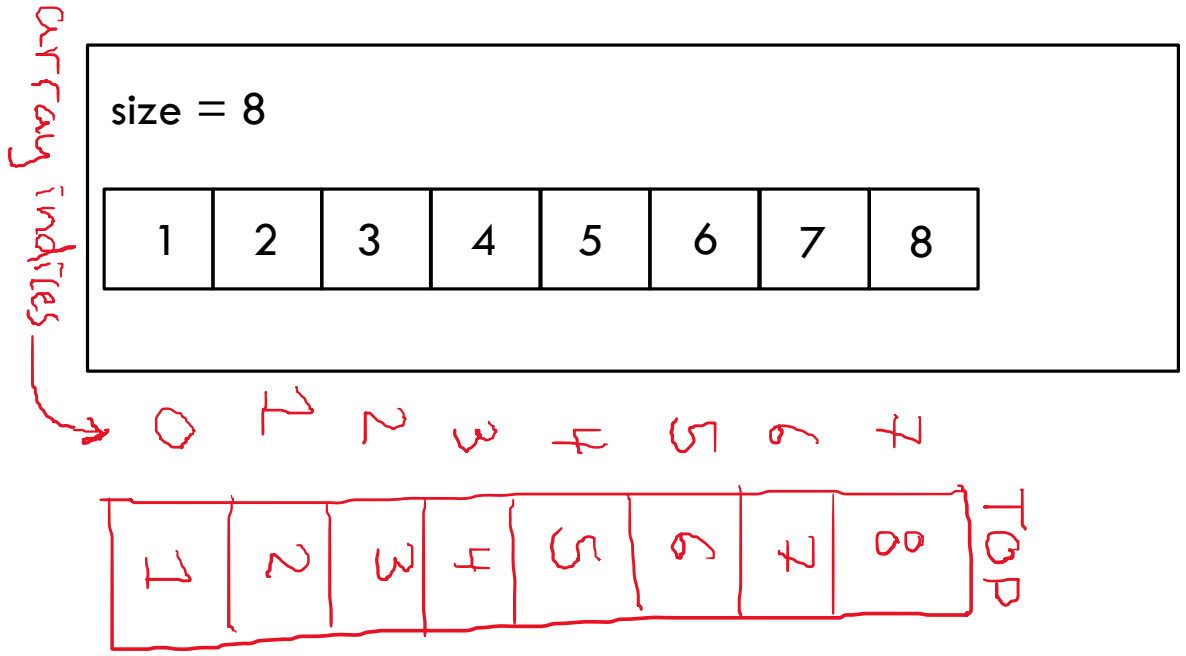




Clearing up Stacks (push them over)

As an array

As a linked list



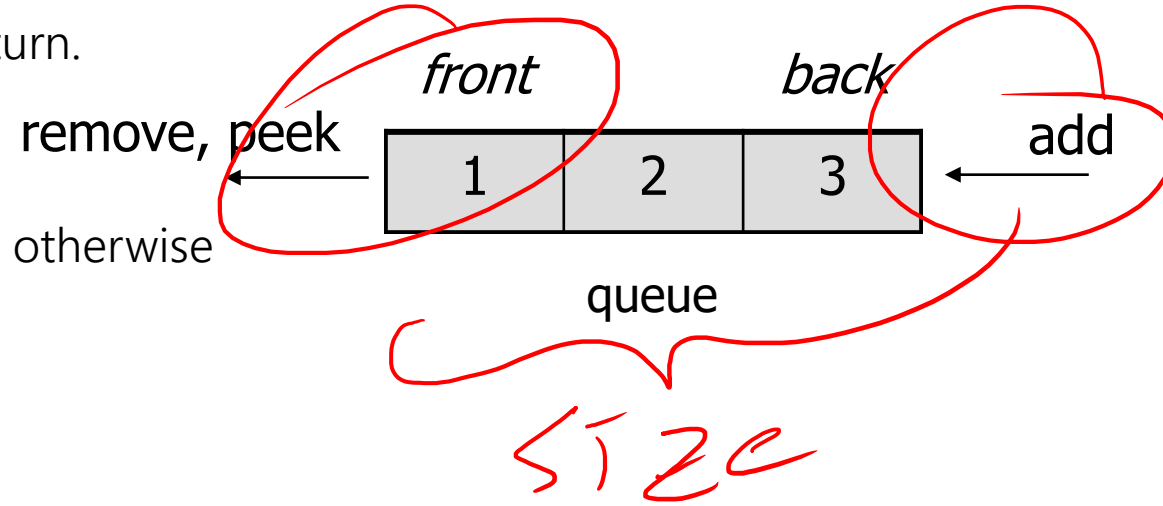
Review: What is a Queue?

queue: Retrieves elements in the order they were added.

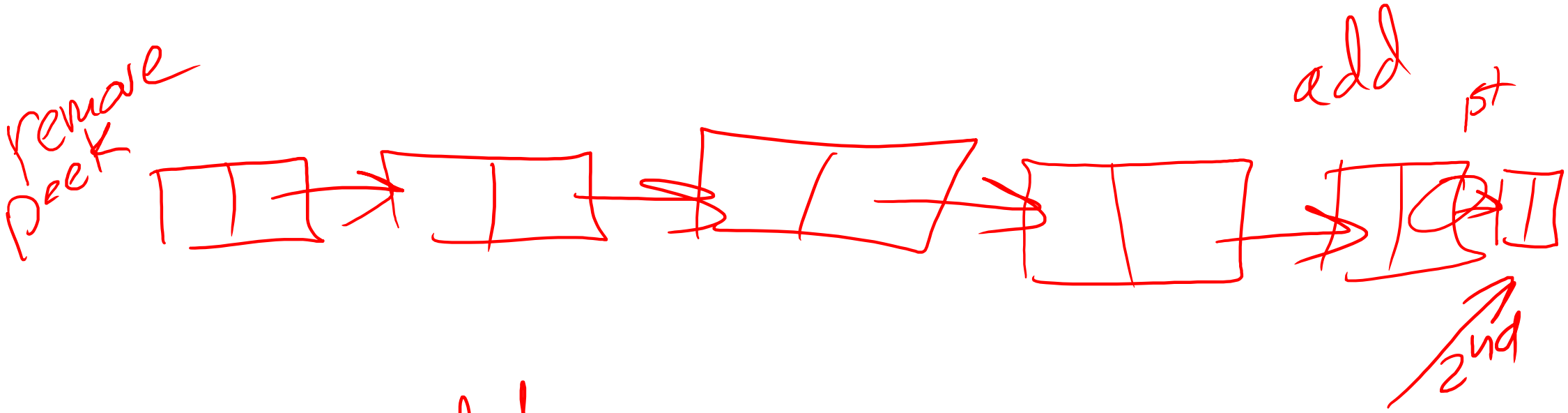
- First-In, First-Out ("FIFO")
- Elements are stored in order of insertion but don't have indexes.
- Client can only add to the end of the queue, and can only examine/remove the front of the queue.

basic queue operations:

- **add(item):** aka "enqueue" add an element to the back.
- **remove():** aka "dequeue" Remove the front element and return.
- **peek()**: Examine the front element without removing it.
- **size():** how many items are stored in the queue?
- **isEmpty():** if 1 or more items in the queue returns true, false otherwise



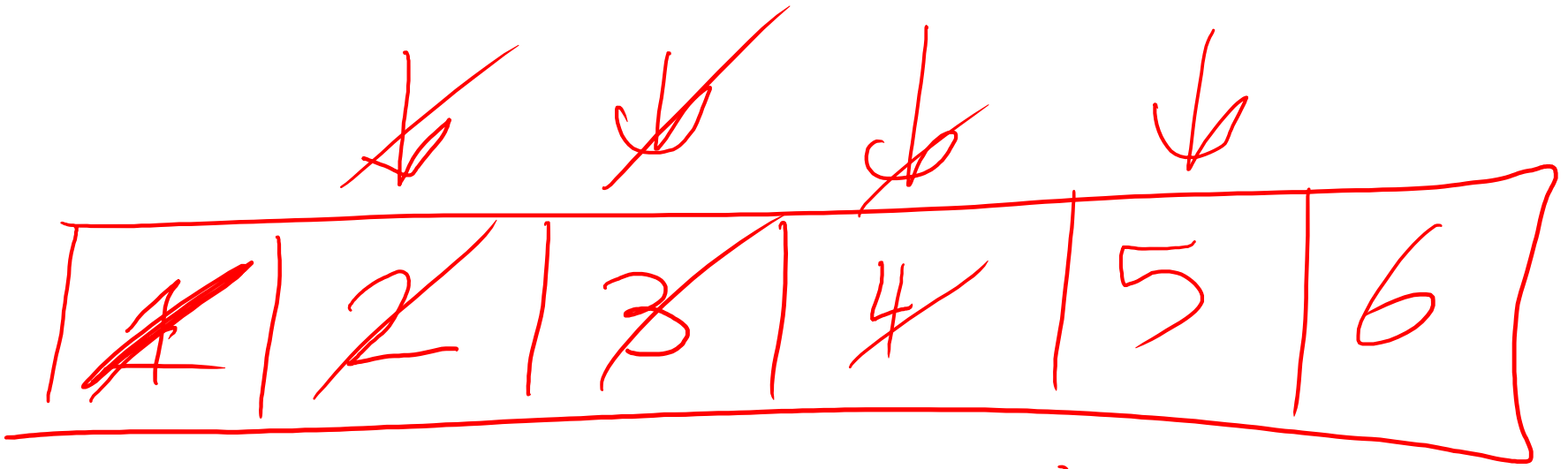
Queues as Linked Lists



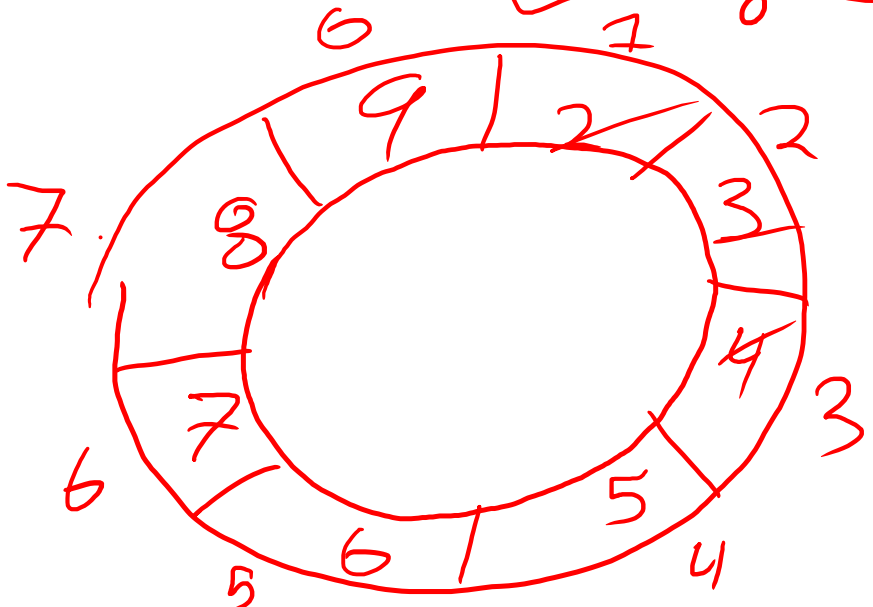
add:
if empty set
front and back to the new node
else: make new node

back

Queues an Arrays



add(1)
add(2)
remove()
add(3)



Queues an Arrays – Looping Back

Discuss with your neighbors:

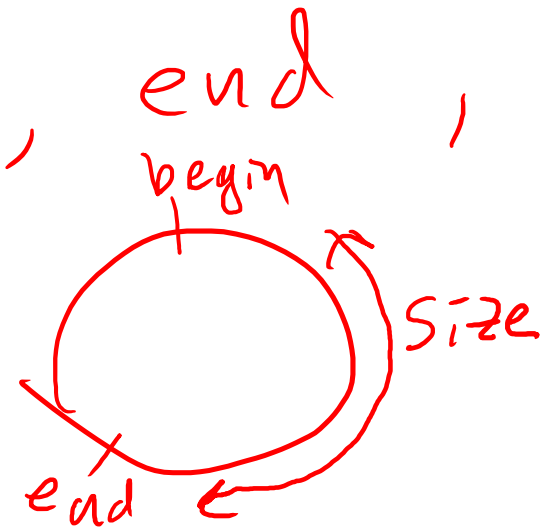
1) What data do I need to store in addition to the array itself? (variable(s) + type(s))

2) How do you calculate the next array position to fill? (equation and/or code)

3) How do you determine if the array is full? Empty? (equation and/or code)

4) How do you resize the array if needed? (description of algorithm)

beginning, end, size : indices (int)



Queues an Arrays – Looping Back

Discuss with your neighbors:

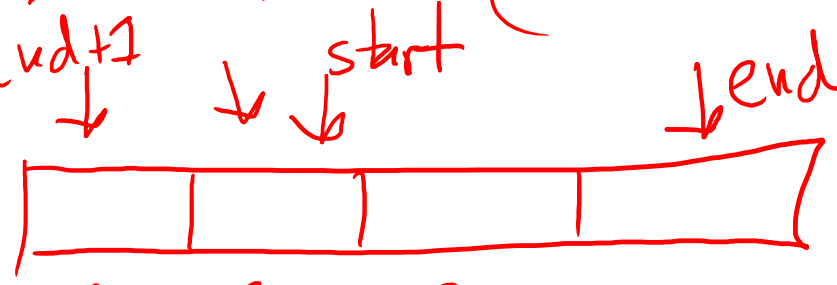
1) What data do I need to store in addition to the array itself? (variable(s) + type(s))

2) How do you calculate the next array position to fill? (equation and/or code)

3) How do you determine if the array is full? Empty? (equation and/or code)

4) How do you resize the array if needed? (description of algorithm)

indexToFill = (endingIndex + 1) % array.length



start=1 size=3

Could use a conditional

size +1

(start+size) % length = 1+3+ = 4 % 4 = 0

Queues an Arrays – Looping Back

Discuss with your neighbors:

1) What data do I need to store in addition to the array itself? (variable(s) + type(s))

2) How do you calculate the next array position to fill? (equation and/or code)

3) How do you determine if the array is full? Empty? (equation and/or code)

4) How do you resize the array if needed? (description of algorithm)

begin, end, size

is size == length of array or 0

Queues an Arrays – Looping Back

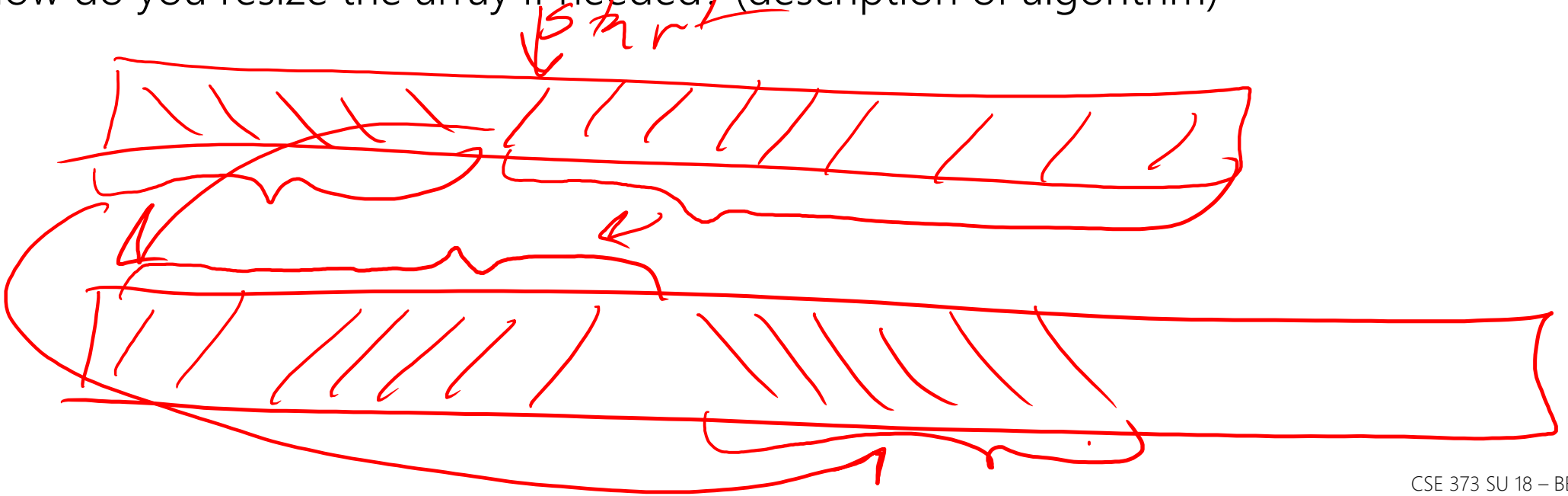
Discuss with your neighbors:

1) What data do I need to store in addition to the array itself? (variable(s) + type(s))

2) How do you calculate the next array position to fill? (equation and/or code)

3) How do you determine if the array is full? Empty? (equation and/or code)

4) How do you resize the array if needed? (description of algorithm)



Question from last time: Aren't these Lists?

Yes, a doubly-linked list can act as both a stack and a queue!

You'll build one of these in your first partner project.

Small correction from last lecture: I flipped a <= sign in the Stack implementation – I checked data.length >= size instead of size >= data.length

Announcements

TA Office Hours Posted - All in CSE 006 – Calendar on Website

Ben's Office Hours shifted to avoid section 2:10 – 4:10 T Th.

Problems and solutions posted for yesterday's section after-all.

Feedback Form on Website

Course Background Survey (required – due next Friday)

Homework 1 releases (individual – due next Friday)

Homework Demo

gitlab.cs.washington.edu

Testing

Computers don't make mistakes- people do!

"I'm almost done, I just need to make sure it works"
– Naive 14Xers

Software Test: a separate piece of code that exercises the code you are assessing by providing input to your code and finishes with an assertion of what the result should be.

1. Isolate

break your code into small modules

2. Build in increments

Make a plan from simplest to most complex cases

3. Test as you go

As your code grows, so should your tests

Types of Tests

Black Box

- Behavior only – ADT requirements
- From an outside point of view
- Does your code uphold its contracts with its users?
- Performance/efficiency

White Box

- Includes an understanding of the implementation
- Written by the author as they develop their code
- Break apart requirements into smaller steps
- "unit tests" break implementation into single assertions

What to test?

Expected behavior

- The main use case scenario
- Does your code do what it should given friendly conditions?

Forbidden Input

- What are all the ways the user can mess up?

Empty/Null

- Protect yourself!
- How do things get started?

Boundary/Edge Cases

- First
- last

Scale

- Is there a difference between 10, 100, 1000, 10000 items?

Thought Experiment

Discuss with your neighbors: Imagine you are writing an implementation of the List interface that stores integers in an Array. What are some ways you can assess your program's correctness in the following cases:

Expected Behavior

- Create a new list
- Add some amount of items to it
- Remove a couple of them

Forbidden Input

- Add a negative number
- Add duplicates
- Add extra large numbers

Empty/Null

- Call remove on an empty list
- Add to a null list
- Call size on an null list

Boundary/Edge Cases

- Add 1 item to an empty list
- Set an item at the front of the list
- Set an item at the back of the list

Scale

- Add 1000 items to the list
- Remove 100 items in a row
- Set the value of the same item 50 times

JUnit

JUnit: a testing framework that works with IDEs to give you a special GUI experience when testing your code

```
@Test
public void myTest() {
    Map<String, Integer> basicMap = new LinkedListDict<String, Integer>();
    basicMap.put("Fido", 42);
    assertEquals(42, basicMap.get("Fido"));
}
```

Assertions:

- assertEquals(item1, item2)
- assertTrue(Boolean expression)
- assertFalse(bollean expression)
- assertNotNull(item)

More: https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html

Traversing Data

Array

```
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}
```

List

```
for (int i = 0; i < myList.size(); i++) {
    System.out.println(myList.get(i));
}
```

```
for (T item : list) {
    System.out.println(item);
}
```

Iterator!

Implement a CircularQueue (ArrayQueue)

- What order should we implement in?
- Test as we go!
- The debugger is our friend!

Iterators

iterator: a Java interface that dictates how a collection of data should be traversed.

Behaviors:

hasNext() – returns true if the iteration has more elements

next() – returns the next element in the iteration

```
while (iterator.hasNext()) {
    T item = iterator.next();
}
```

Implementing an Iterator

TODO list

Homework 1 is live! Set up your development environment.

- Individual Assignment
- **Try logging into gitlab.cs.washington.edu ASAP so we can help you early!**
- Due 6/29 at 11:59pm

Start looking for partners for Homework 2!

- Will be assigned on 6/29
- Sign up for Piazza – there's a "Search for Teammates" pinned post

Fill out the pre-class survey

- Link is on the website