CS151 Intro to Data Structures

Stacks

Junit

Queues

Announcements

HW2 due Sunday

Stacks - FILO

- First In Last Out

- stack of plates in the dining hall

- Operations:
 - push
 - pop
 - peek
 - isEmpty

Stack Example - Browser History

Let's implement Stack

```
public interface Stack<E> {
  int size();
  boolean isEmpty();
  E pop();
  E peek(); //does not modify the stack
  void push(E element); //pushes to top of stack
}
```

Array Stack Performance

Space complexity is

• O(n)

Runtime Complexity:

- push:
 - · O(1)
 - what if we had an expandable array? O(n)
- Pop:
 - O(1)
- Peek:
 - O(1)

Now let's implement stack with a linked list!

Linked List Stack Performance

Space complexity is

• O(n)

Runtime Complexity:

- push:
 - · O(1)
- Pop:
 - O(1)
- Peek:
 - · O(1)

Queues

FIFO Stacks

Stack Property

First-in Last-out (FILO)

Where might a FILO stack not make sense?

Line for the cash register

Printer Queue

FIFO: First-in First-out

The first item in, is the first item out

Add-to the back, remove from the front

This is a Queue

Inserting - "enqueue"

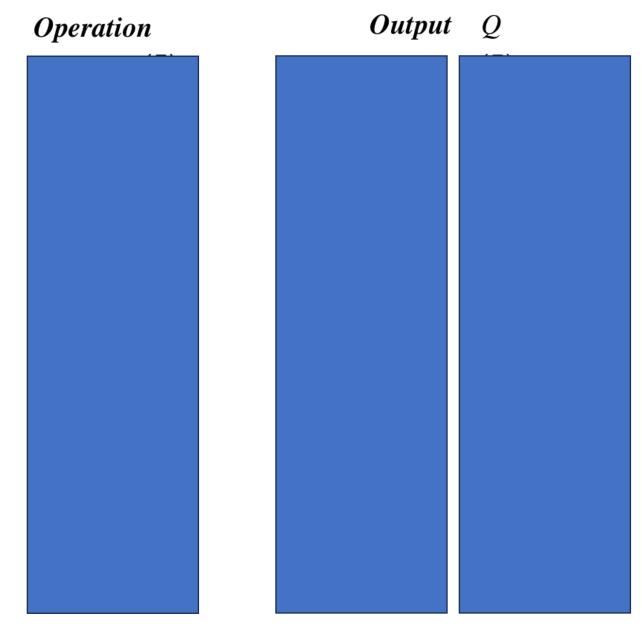
Removing - "dequeue"

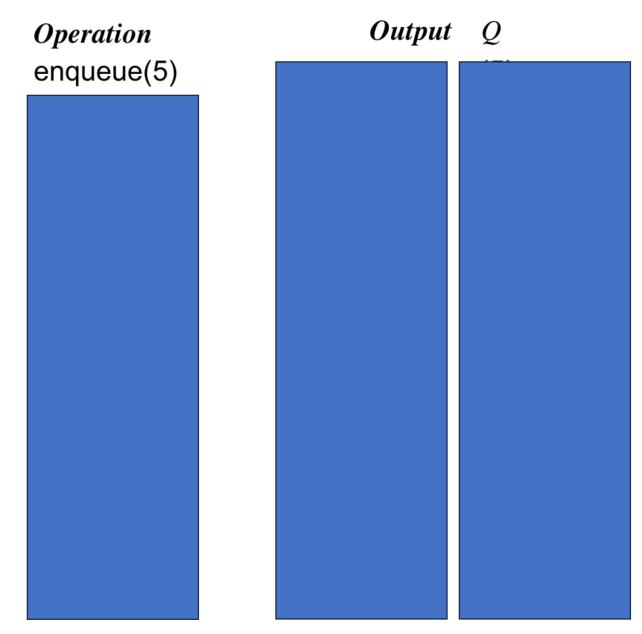
Queue Interface

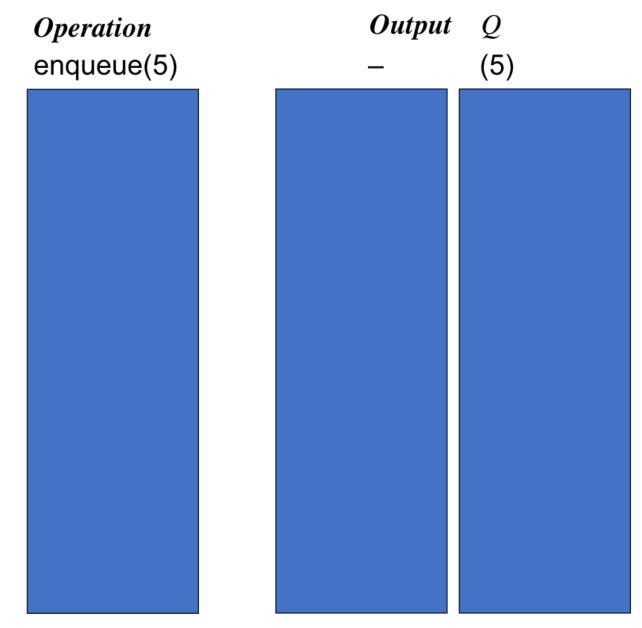
```
public interface Queue<E> {
  int size();
  boolean isEmpty();
  E first();
  void enqueue (E e);
  E dequeue();
                           null is returned from
                            dequeue() and first()
                            when queue is empty
```

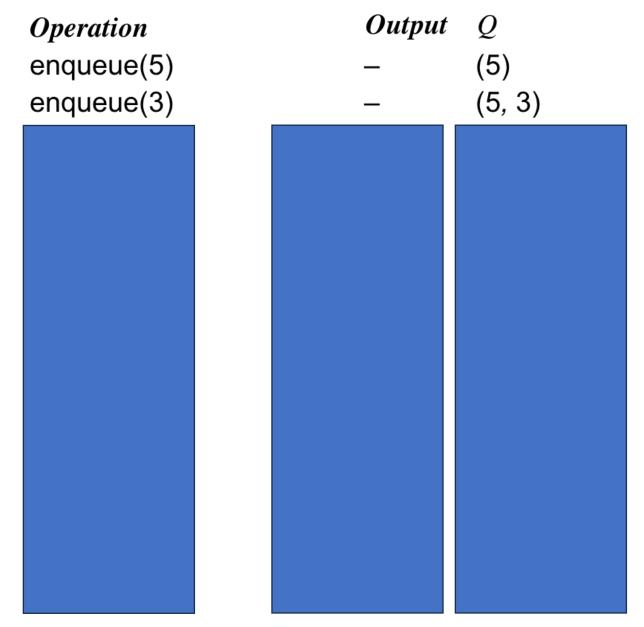
Queue Example

Cash register code









```
Output
Operation
enqueue(5)
                                  (5)
enqueue(3)
                                  (5, 3)
dequeue()
enqueue(7)
dequeue()
first()
dequeue()
dequeue()
isEmpty()
enqueue(9)
enqueue(7)
size()
enqueue(3)
enqueue(5)
dequeue()
```

```
Output
Operation
enqueue(5)
                                     (5)
                                     (5, 3)
enqueue(3)
dequeue()
                           5
                                     (3)
                                     (3, 7)
enqueue(7)
dequeue()
                                     (7)
first()
                                     (7)
dequeue()
dequeue()
                           null
isEmpty()
                           true
                                     (9)
enqueue(9)
enqueue(7)
                                     (9, 7)
size()
                                     (9, 7)
enqueue(3)
                                     (9, 7, 3)
enqueue(5)
                                     (9, 7, 3, 5)
dequeue()
                           9
                                    (7, 3, 5)
```

Amortized Analysis

Amortized Analysis

average time complexity

Array insertion:

- worst case?
- best case?
- average case? (explanation on next slide)

Amortized Analysis

Amortized cost per operation for a sequence of k operations is the total cost of the operations divided by k

Similar to an average

O(1)