Week06

✨ **Fun Friday!** ✨
this week sure *flew* by, didn't it?

- deque
- ~~Python vs. Java vs. C philosophy of data structures~~
- fun friday

WARMUP
what does **random access** (direct access) mean in the context of data structures?

TODO: record lecture

93

review: queue

94

95

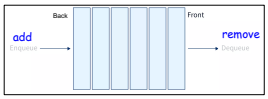
queue interface

```
- // Add (enqueue) a new element to the back of the queue.
  void add(ElementType element);

- // Remove (dequeue) the front
  // element and return it.
  ElementType remove();

- // Peek (look) at the front element (without removing it)
  // and return it.
  ElementType peek();

- // Returns the number of elements currently in the queue.
  int size();
```



96

deque
(double-ended queue)

97

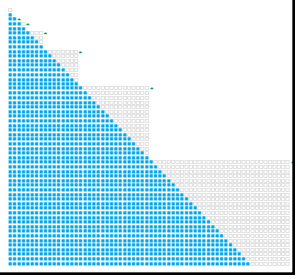
deque

98

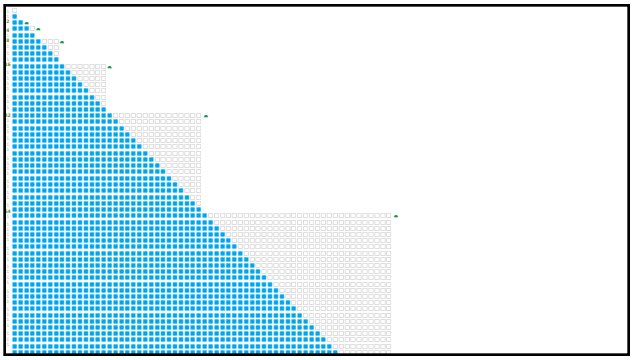
deque interface

- `addFront(...)`
- `removeFront()`
- `addBack(...)`
- `removeBack()`

- what is the amortized runtime of these functions if we were to implement the deque interface using an array list?
  - `O(1) addBack(...)` and `removeBack()` 😀
  - `O(n) addFront(...)` and `removeFront()` 😀
- can we do better?



99



100

neat idea: store the queue in the *middle*

```
BACK [ null null null null null null null ] FRONT
BACK [ null null null 1111 null null null ] FRONT
BACK [ null null 2222 1111 null null null ] FRONT
BACK [ null 3333 2222 1111 null null null ] FRONT
BACK [ null 3333 2222 null null null null ] FRONT
BACK [ null 3333 2222 4444 null null null ] FRONT
BACK [ null 3333 2222 4444 5555 null null ] FRONT
BACK [ null null 2222 4444 5555 null null ] FRONT
```

```
// addBack(1111)
// addBack(2222)
// addBack(3333)
// removeFront()
// addFront(4444)
// addFront(5555)
// removeBack()
```

101

neat idea: store the queue in the *middle*

does this waste space?
(in the worst case)
(relative to an array list)

can we fix this?

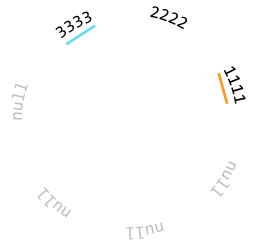
```
BACK [ null 3333 2222 1111 null null null ] FRONT
```

`int back;` `int front;`

note: many possible implementations;
key point: you have to keep track of back and front somehow

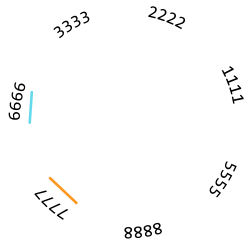
102

extension: treat the array as circular



103

extension: treat the array as circular



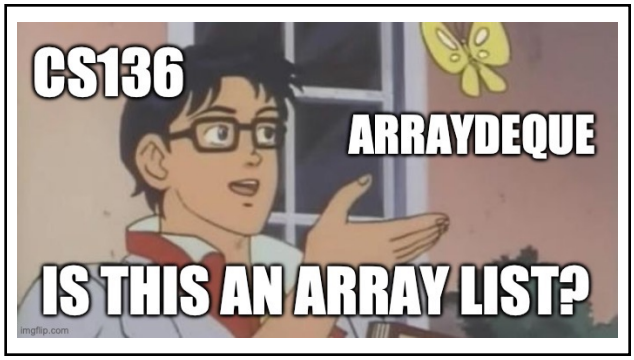
104

big picture lesson

105

once you get past CS136,
data structures are acually kind of
rich and complicated and messy

106



107

Java vs. C
philosophy
of data structures

108

[scavenger hunt]

109

can i call `get(...)` or `set(...)` on
Java's `ArrayDeque`?

why or why not?

110