Topic 16
Queues

"FISH queue: n.

[acronym, by analogy with FIFO (First In, First Out)] 'First In, Still Here'. A joking way of pointing out that processing of a particular sequence of events or requests has stopped dead. Also FISH mode and FISHnet; the latter may be applied to any network that is running really slowly or exhibiting extreme flakiness."

-The Jargon File 4.4.7

Queues

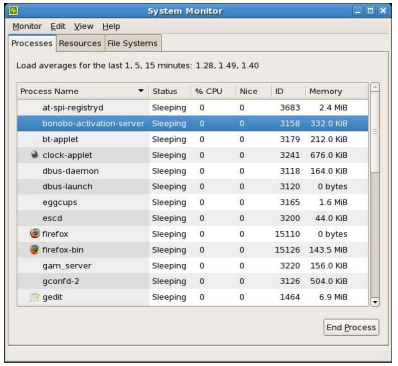
- A sharp tool, like stacks
- A line
  - In England people don't "get in line" they "queue up".

Queue Properties

- Queues are a first in first out data structure
  - FIFO (or LILO, but I guess that sounds a bit silly)
- Add items to the end of the queue
- Access and remove from the front
  - Access to the element that has been in the structure the ***longest*** amount of time
- Used extensively in operating systems
  - Queues of processes, I/O requests, and much more

Queues in Operating Systems

- On a computer with N cores on the CPU, but more than N processes, how many processes can actually be executing at one time?
- One job of OS, schedule the processes for the CPU



Queue operations

- `void enqueue(E item)`
  - a.k.a. `add(E item)`
- `E front()`
  - a.k.a. `E peek()`
- `E dequeue()`
  - a.k.a. `E remove()`
- `boolean isEmpty()`
- Specify methods in an interface, allow multiple implementations.

Queue interface, version 1

```
public interface Queue314<E> {
      //place item at back of this queue
      public void enqueue(E item);

      //access item at front of this queue
      //pre: !isEmpty()
      public E front();

      //remove item at front of this queue
      //pre: !isEmpty()
      public E dequeue();

      public boolean isEmpty();
}
```


Implementing a Queue

- Given the internal storage container and choice for front and back of queue what are the Big O of the queue operations?

	ArrayList	LinkedList (Singly Linked)	LinkedList (Doubly Linked)
enqueue			
front			
dequeue			
isEmpty			

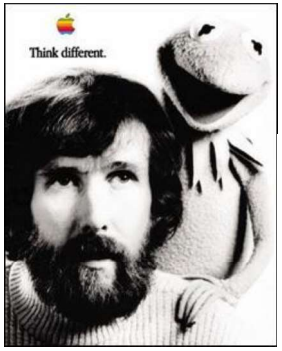
Clicker 1

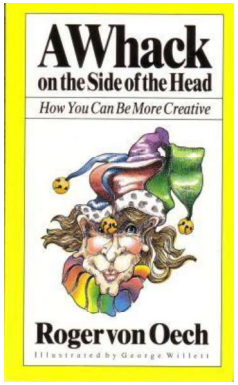
- If implementing a queue with a singly linked list with references to the first and last nodes (head and tail) which end of the list should be the front of the queue in order to have all queue operations O(1)?

A. The front of the list should be the front of the queue.
B. The back of the list should be the front of the queue.
C. Either end will work to make all ops O(1).
D. Neither end will allow all ops to be O(1).

Alternate Implementation

- How about implementing a Queue with a native array?
  - Seems like a step backwards





Application of Queues

- Radix Sort
  - radix is a synonym for *base*. base 10, base 2
- Multi pass sorting algorithm that ***only*** looks at individual digits during each pass
- Use queues as *buckets* to store elements
- Create an array of 10 queues
- Starting with the least significant digit place value in queue that matches digit
- empty queues back into array
- repeat, moving to next least significant digit

Radix Sort in Action: 1s place

- original values in array

  9, 113, 70, 86, 12, 93, 37, 40, 252, 7, 79, 12
- Look at ones place

  9, 113, 70, 86, 12, 93, 37, 40, 252, 7, 79, 12
- Array of Queues (all empty initially):

0	5
1	6
2	7
3	8
4	9

Radix Sort in Action: 1s

- original values in array

  9, 113, 70, 86, 12, 93, 37, 40, 252, 7, 79, 12
- Look at ones place

  9, 113, 70, 86, 12, 93, 37, 40, 252, 7, 79, 12
- Queues:

0 70, 40	5
1	6 86
2 12, 252, 12	7 37, 7
3 113, 93	8
4	9 9, 79

Radix Sort in Action: 10s

- Empty queues in order from 0 to 9 back into array

  70, 40, 12, 252, 12, 113, 93, 86, 37, 7, 9, 79

- Now look at 10's place

  70, 40, 12, 252, 12, 113, 93, 86, 37, _7, _9, 79

- Queues:

0 _7, _9	5 252
1 12, 12, 113	6
2	7 70, 79
3 37	8 86
4 40	9 93

Radix Sort in Action: 100s

- Empty queues in order from 0 to 9 back into array

  7, 9, 12, 12, 113, 37, 40, 252, 70, 79, 86, 93

- Now look at 100's place

  __7, __9, _12, _12, 113, _37, _40, 252, _70, _79, _86, _93

- Queues:

0 _7, _9, _12, _12, _37, _40, _70, _79, _86, _93	5
1 113	6
2 252	7
3	8
4	9

Radix Sort in Action: Final Step

- Empty queues in order from 0 to 9 back into array

  7, 9, 12, 12, 40, 70, 79, 86, 93, 113, 252

Radix Sort Code

```
public static void sort(int[] data){
      ArrayList<Queue<Integer>> queues
           = new ArrayList<Queue<Integer>>();
      for(int i = 0; i < 10; i++)
          queues.add(new LinkedList<Integer>());
      int passes = numDigits(getMax(data));// helper methods

      for(int i = 0; i < passes; i++){
          for(int j = 0; j < data.length; j++) {
              int digit = valueOfDigit(data[j], i);
              queues.get(digit).add(data[j]);
          }
          int pos = 0;
          for(Queue<Integer> q : queues){
              while(!q.isEmpty())
                  data[pos++] = q.remove();
          }
      }
}
```