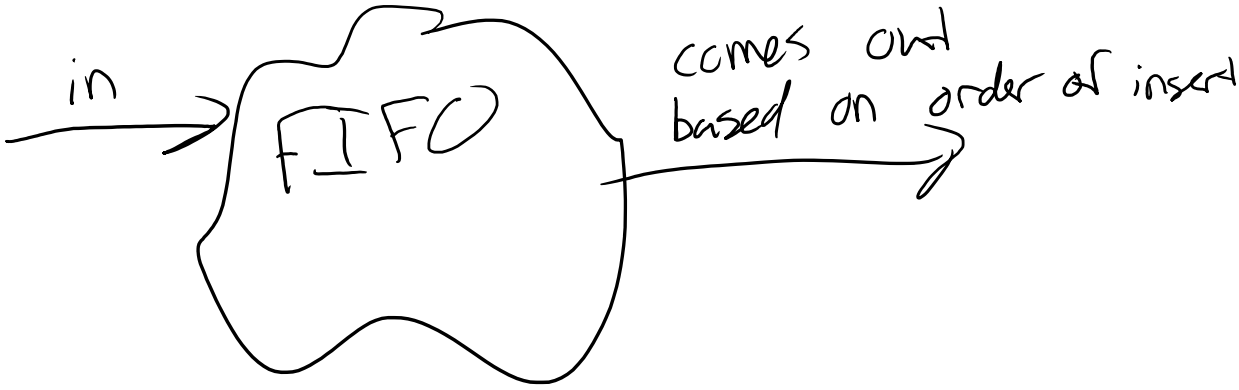
2018-09-11 Completing Stacks & Queues

Tuesday, September 11, 2018 2:59 PM

What is a queue

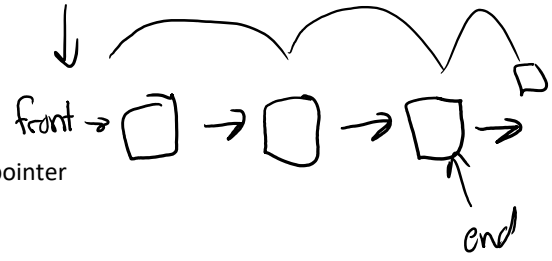


How would you write your own queue?

Idea #1: Use a vector

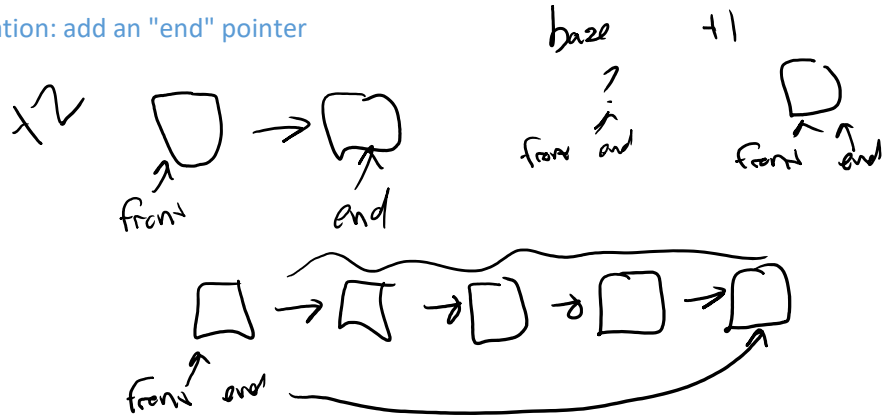
- "push back" new items. Means that items that arrived early are in the front and that items that arrived late are in the back.
  - Assuming that we have enough space, O(1) behavior
- Dequeue (or C++ pop) - Remove the front item, shift everything over (left, or by -1) by 1.
  - Requires FOR loop. O(N) behavior

Idea #2: Use a linked list



- Enqueue: add next item to the end of the list
  - O(N) complexity
- Dequeue: remove item from front of list, alter pointer
  - O(1) complexity

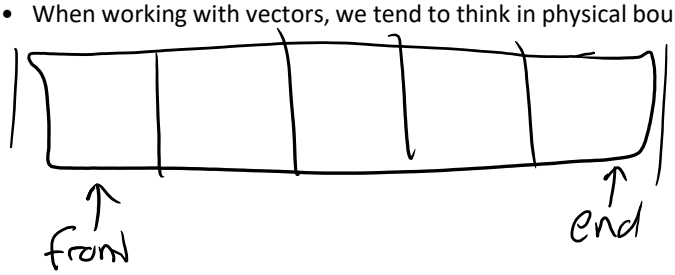
Observation: add an "end" pointer



- Using an "end" pointer allows us to add items to the end of the linked list in O(1) time
- Thus, enqueue become O(1) on a linked list

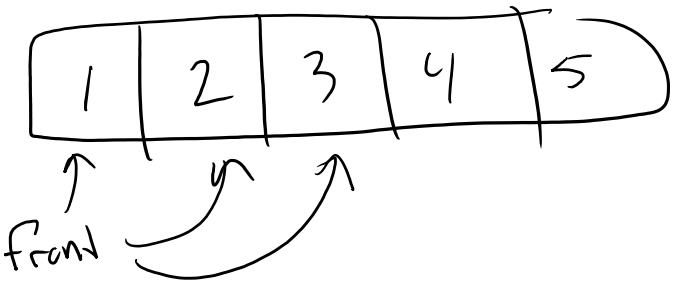
Class Exercise: Improving the Vector-based implementation

- When working with vectors, we tend to think in physical boundaries



  - Instead, consider logical or "floating" boundaries. Consider the following code:

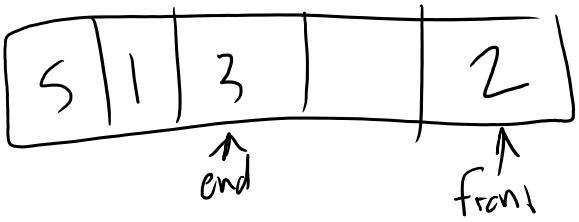
```
int front = 0;
int value = data[front];
front++;
return value;
```



- Downside: empty space at the front of the vector. Once we dequeue from a slot, we never use that space again. Thus if we do 1000 enqueues, 1000 dequeues, and 1000 more enqueues, our vector would be at least 2000 big.
- Solution: is to make the end of the vector a logical pointer. Consider:

```
Int end = vector.size() - 1;
Data[end] = value
End = (end + 1) % vector.size();
```

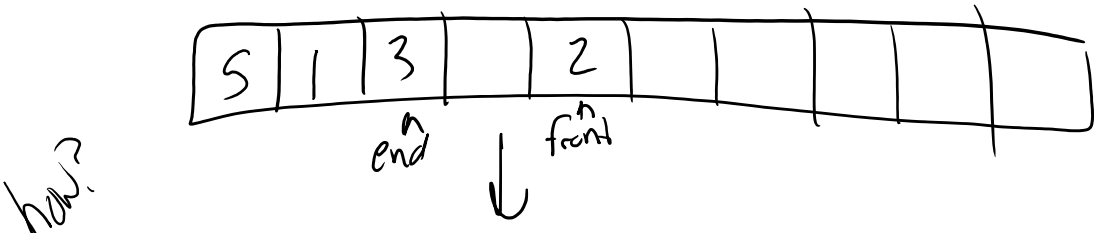
- Doing so can result in a situation in which the "front" of the queue is ahead of the "end" of the queue

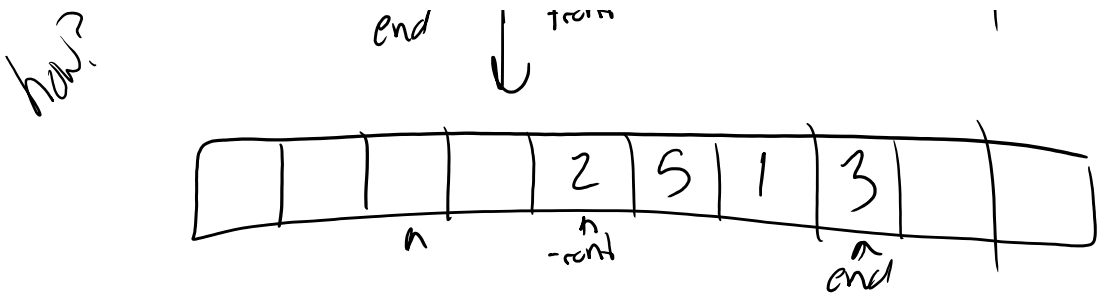


  - This implementation is called a Circular Queue. By using logical front and ends on a vector, a circular queue achieves O(1) enqueue and dequeue.

"Unwinding" a circular queue when expanding

- Double the size of our data






```
Int size = data.size();
Data.resize(size * 2)
If(front > end)
{
  Int counter = size;
  For(int I = 0; I <= end; i++)
  {
    data[counter] = data[i];
    counter++;
  }
  End = counter;
}
```