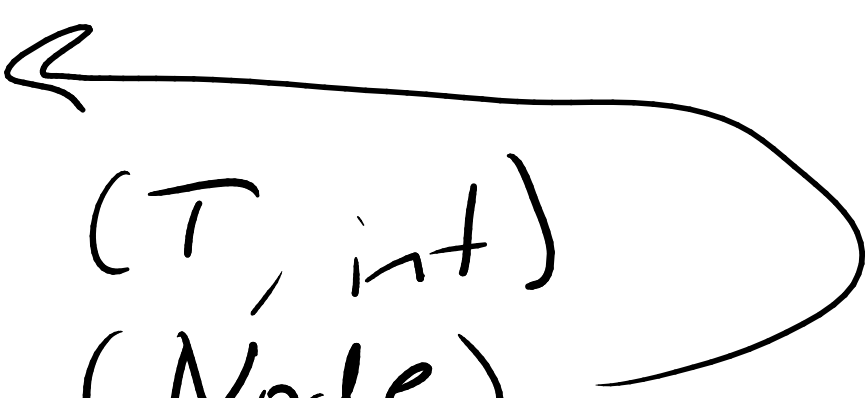



Recursion - when a function itself
OR when a data structure refers
to itself

Node 
value (T, int)
next (Node)

Base case: easy

Recursive ~~case~~ case: believe something
untrue

believe: function is already
written

 but only for cases closer
to the base case

What's happening in memory?

fact(3)

6

$n \rightarrow 3$
if $n == 1$ X
else
return $3 * \overset{2}{\text{fact}(2)}$

$n \rightarrow 2$
if $n == 1$ X
else
return $2 * \overset{1}{\text{fact}(1)}$

$n \rightarrow 1$
if $n == 1$
return 1

Pros and cons of recursion vs loops (iteration)

Recursion might use more memory

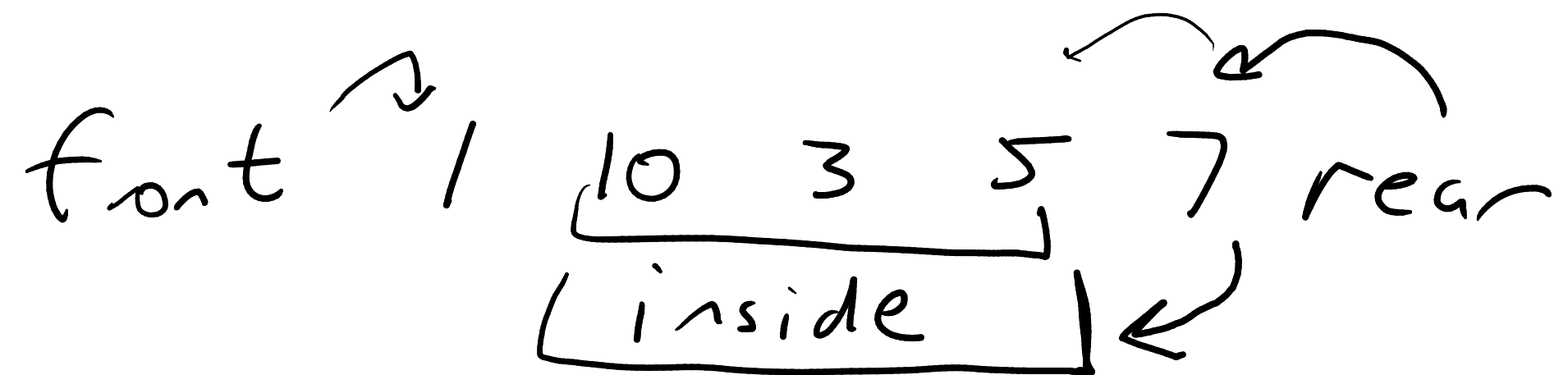
- sometimes approx same
- recursion will never be less

There are many cases where recursion is much simpler

- code might be much shorter
- easier to maintain
- etc

→ if this is a case where the memory price isn't significant, then it's a win.

Recursive queue



~~enqueue~~ enqueue(12)

