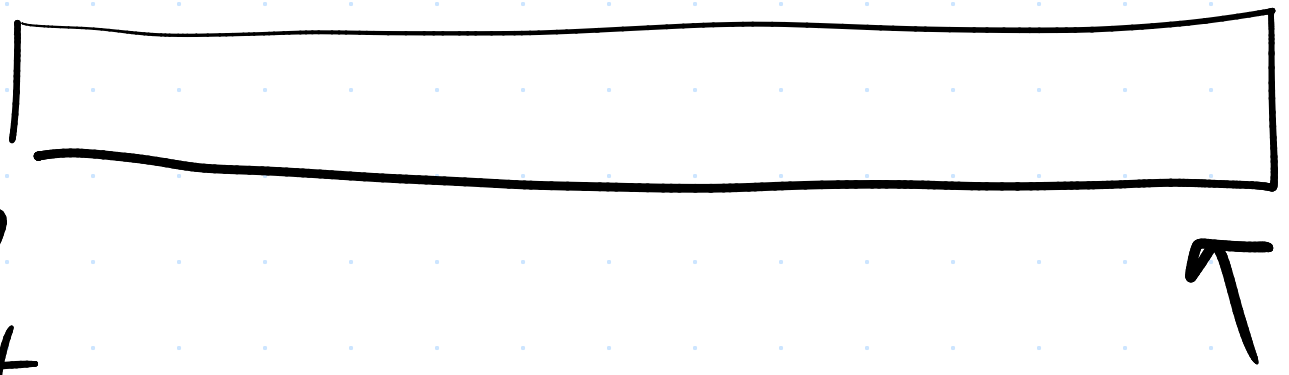
Queues

- First In, First Out (FIFO)

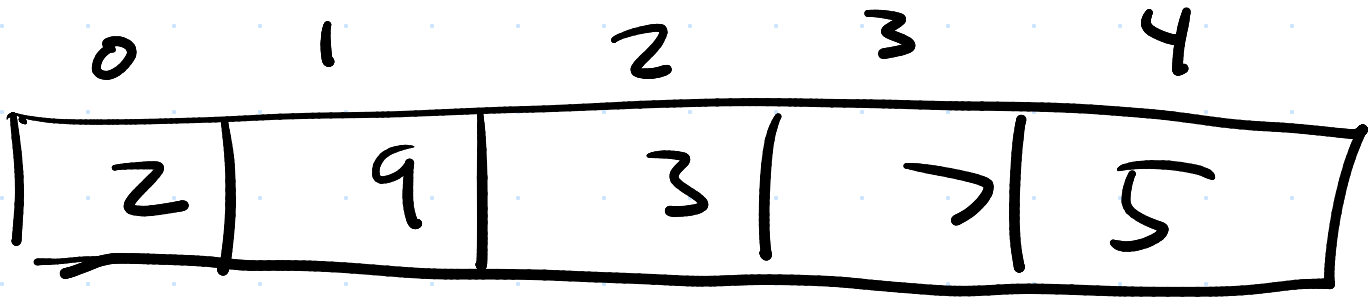
Stacks

- Last In, First Out (LIFO)



Implement strategies
- array
- list
  - linked chain]

With a list, same as a stack (almost)



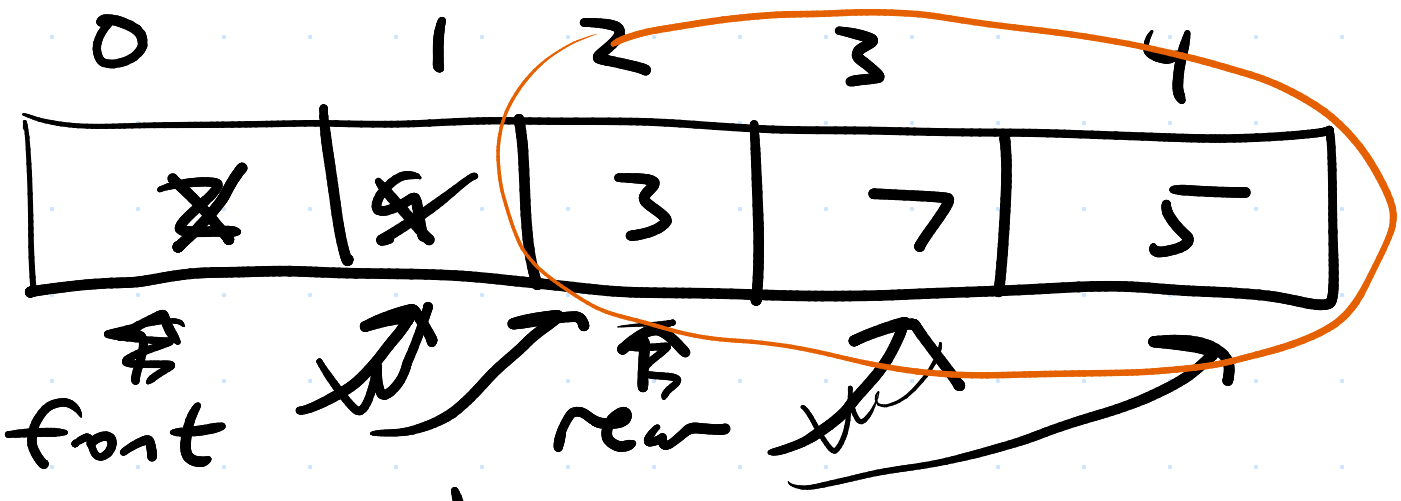
enqueue

2
9
3
7
5

Naive approach

- add on end — O(1)
  list.add
- remove from location 0
  list.remove(0) — O(n)

Smater approach: "snake"



	front	rear
enqueue(2)	~~0~~	~~0~~
enqueue(9)	~~1~~	~~1~~
enqueue(3)	~~2~~	~~2~~
enqueue(7)	2	4

dequeue() ⇒ 2

enqueue(5)

dequeue() ⇒ 9

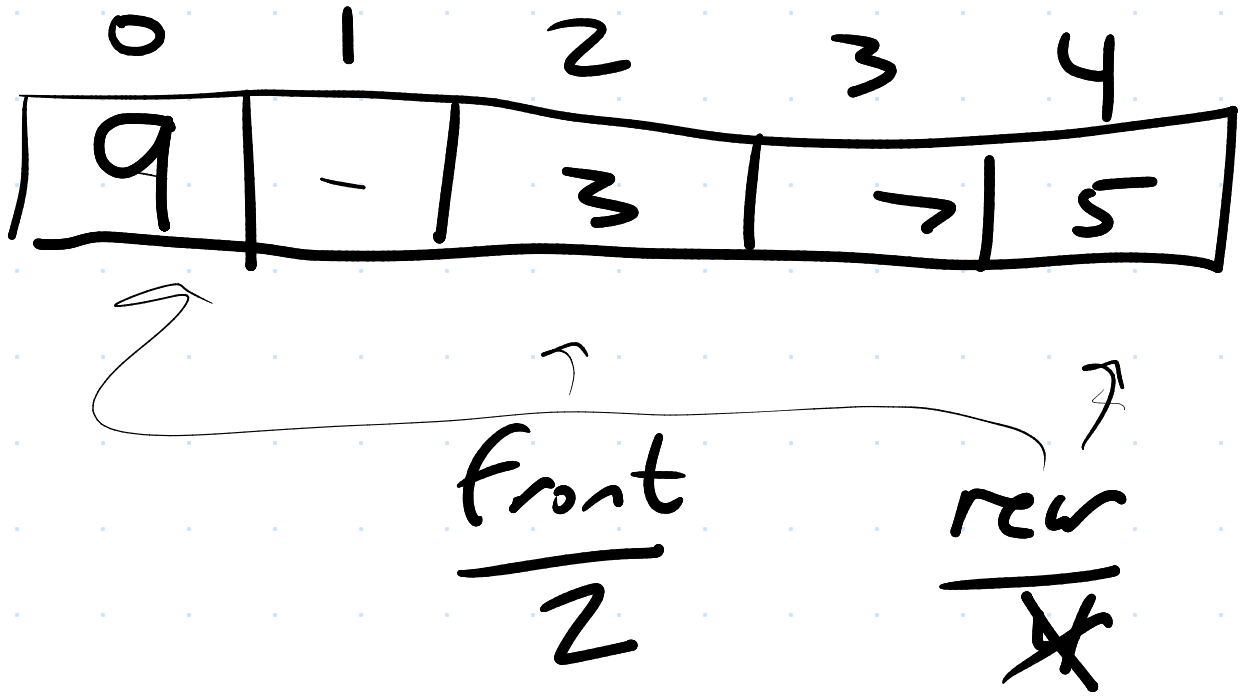
O(1) for everything

Empty memory at front just piles up

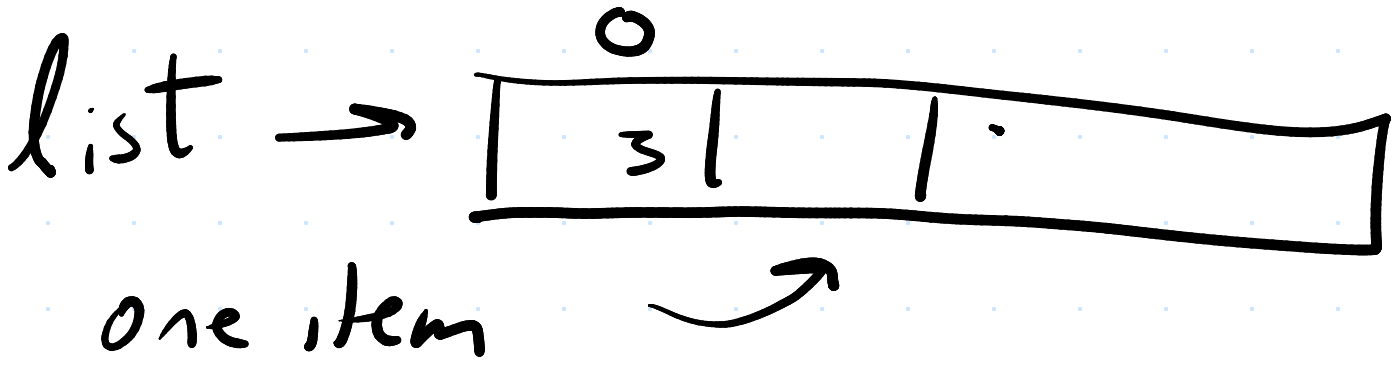
- eventually eat up all the memory you have

Fix: slide everything down
? every now and then
... ok... O(n) when it
happens

FIX! Wrap around the
end of the list



enqueue(9)



one item

front = 0
rear = 0

} front and rear same means one item

In this simple version,

front = -1
rear = -1

empty queue

For adding 2 possibilities

room in list

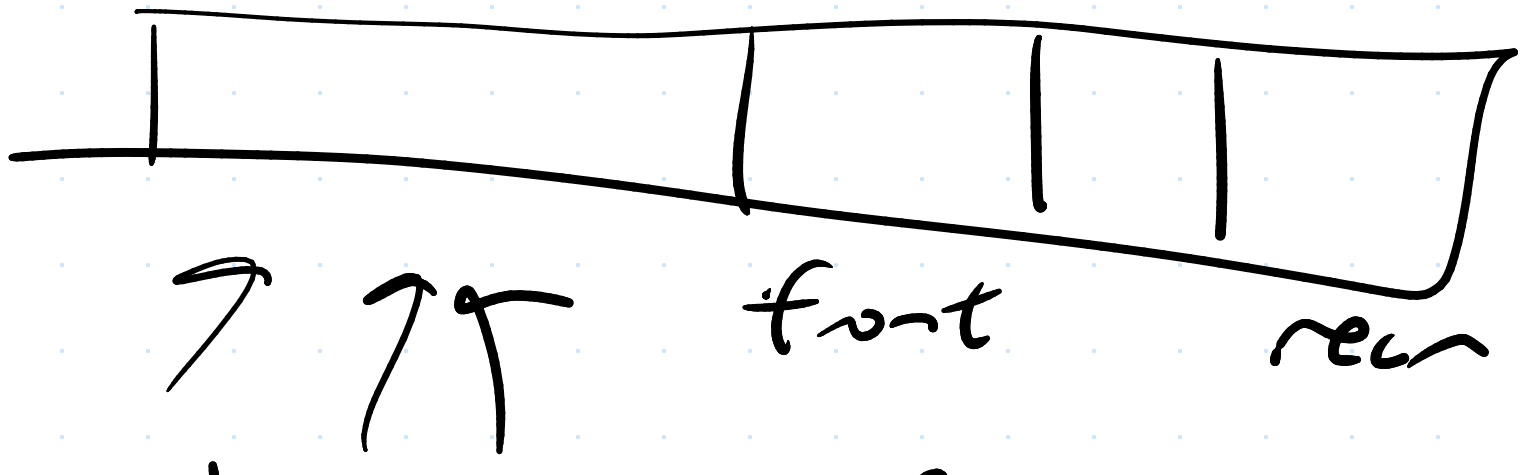
list[rear] = item

Post end of list

rear

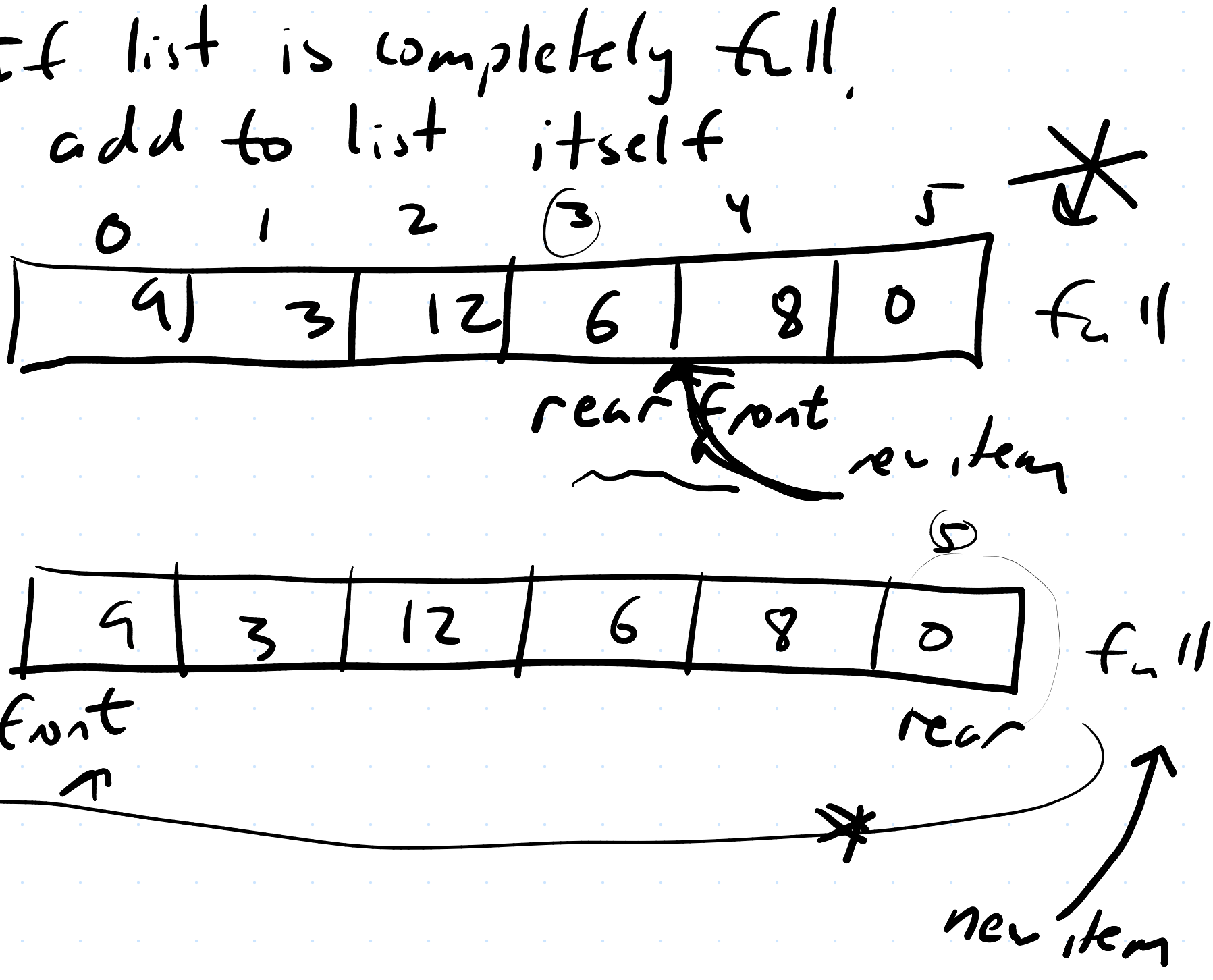
list.add(item)

Fix enqueue to instead wrap around end of list if you are at the end of list and there is room at beginning



What is condition for a full list?

If list is completely full, add to list itself



have room