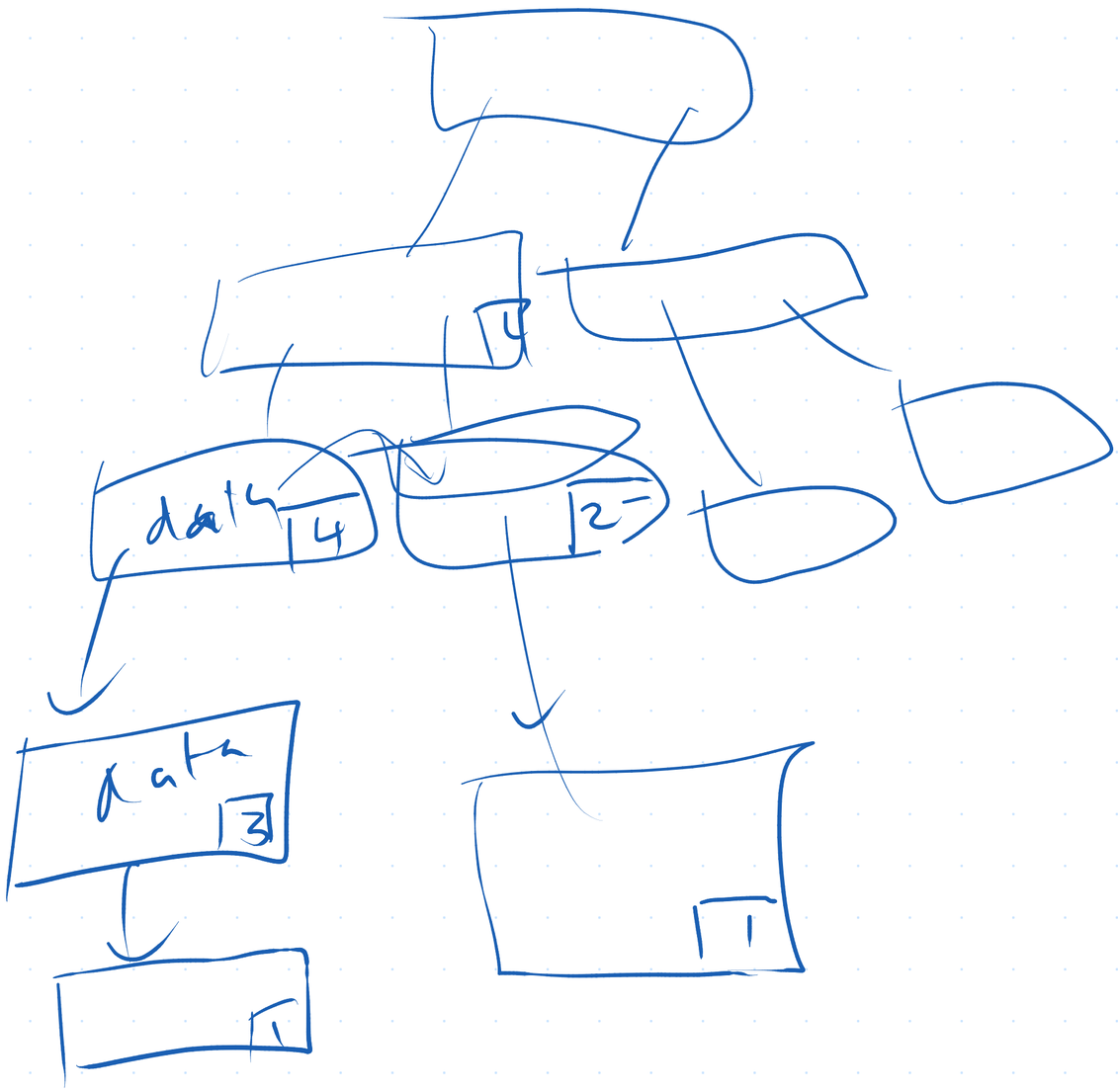
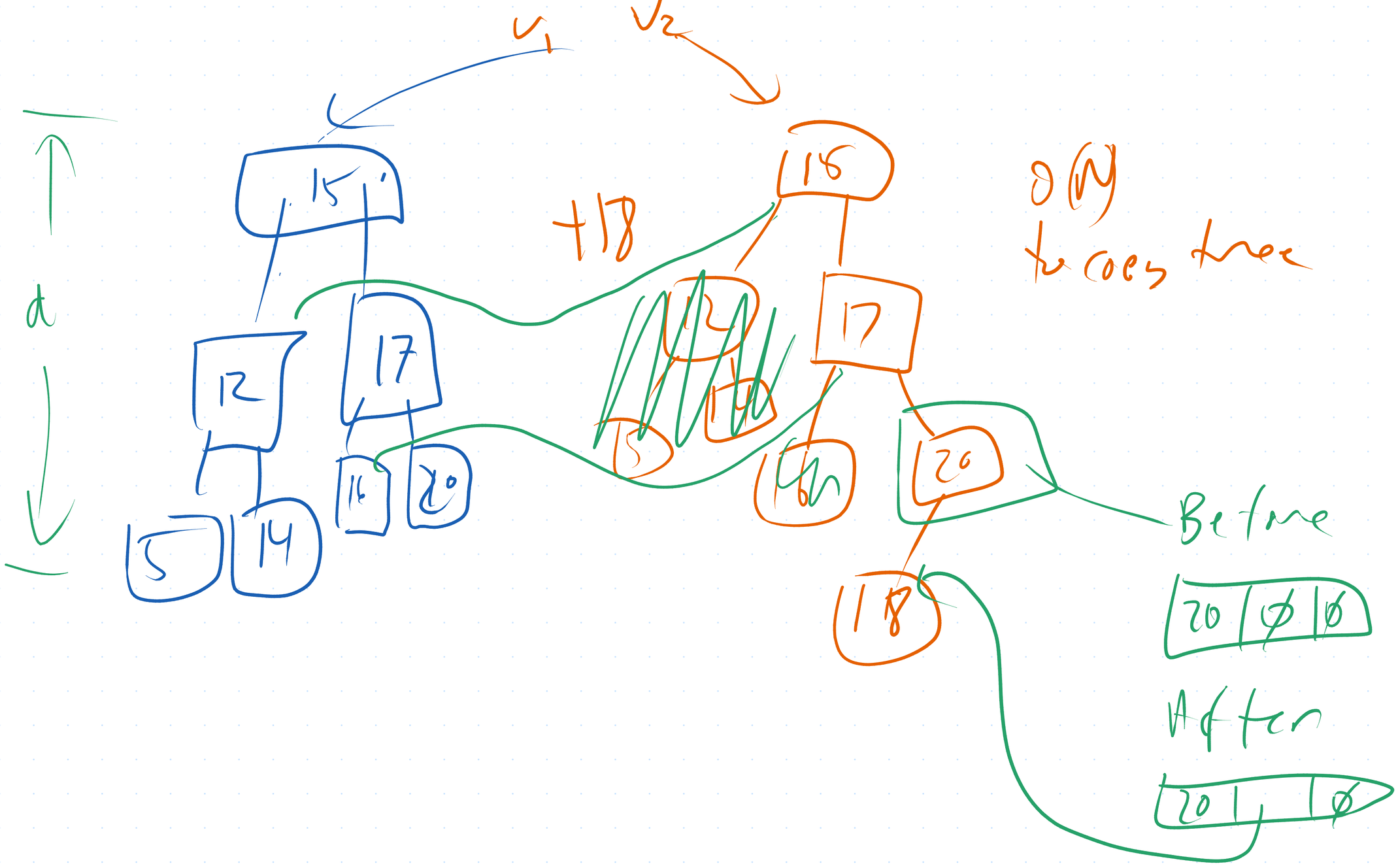
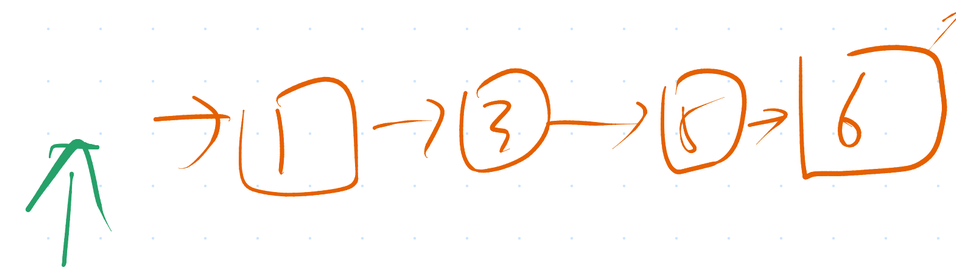



```
let foo = ___

let mut foo = ___
```

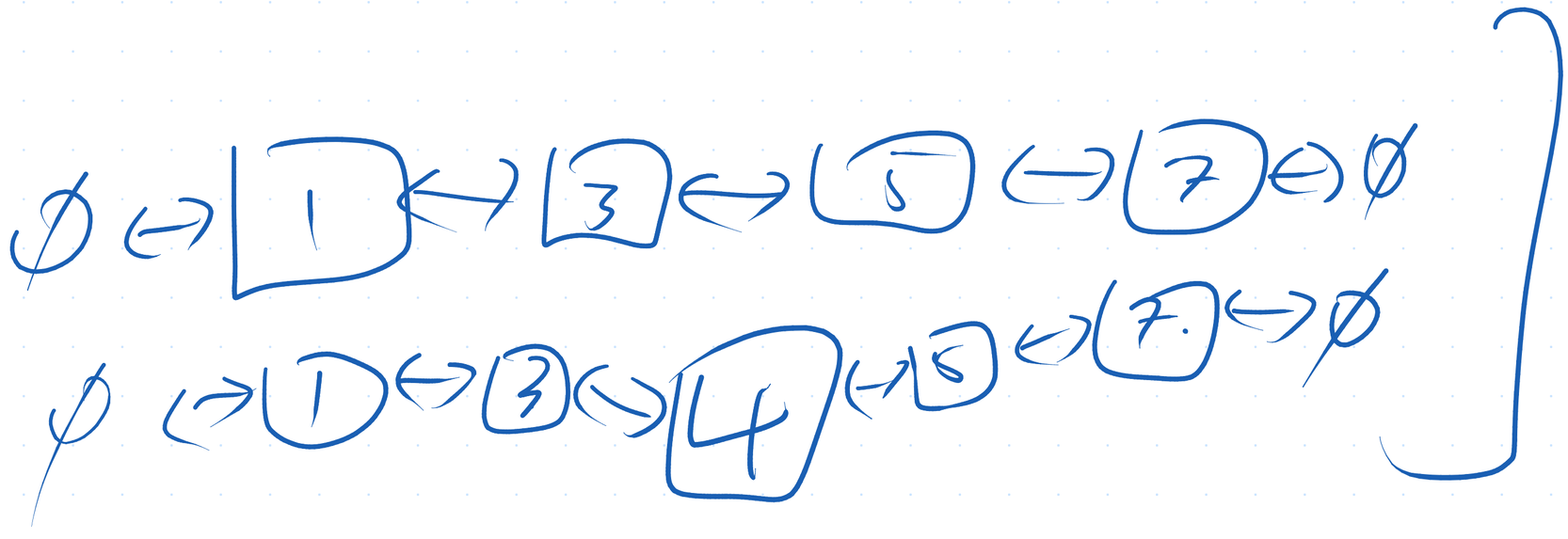

v_1
v_2
d
15
12
17
15
14
16
20
+18
15
12
17
15
16
20
18
O(N)
to copy tree
Before
20 | ∅ | ∅
After
20 | 1 | ∅

Immutable Linked List

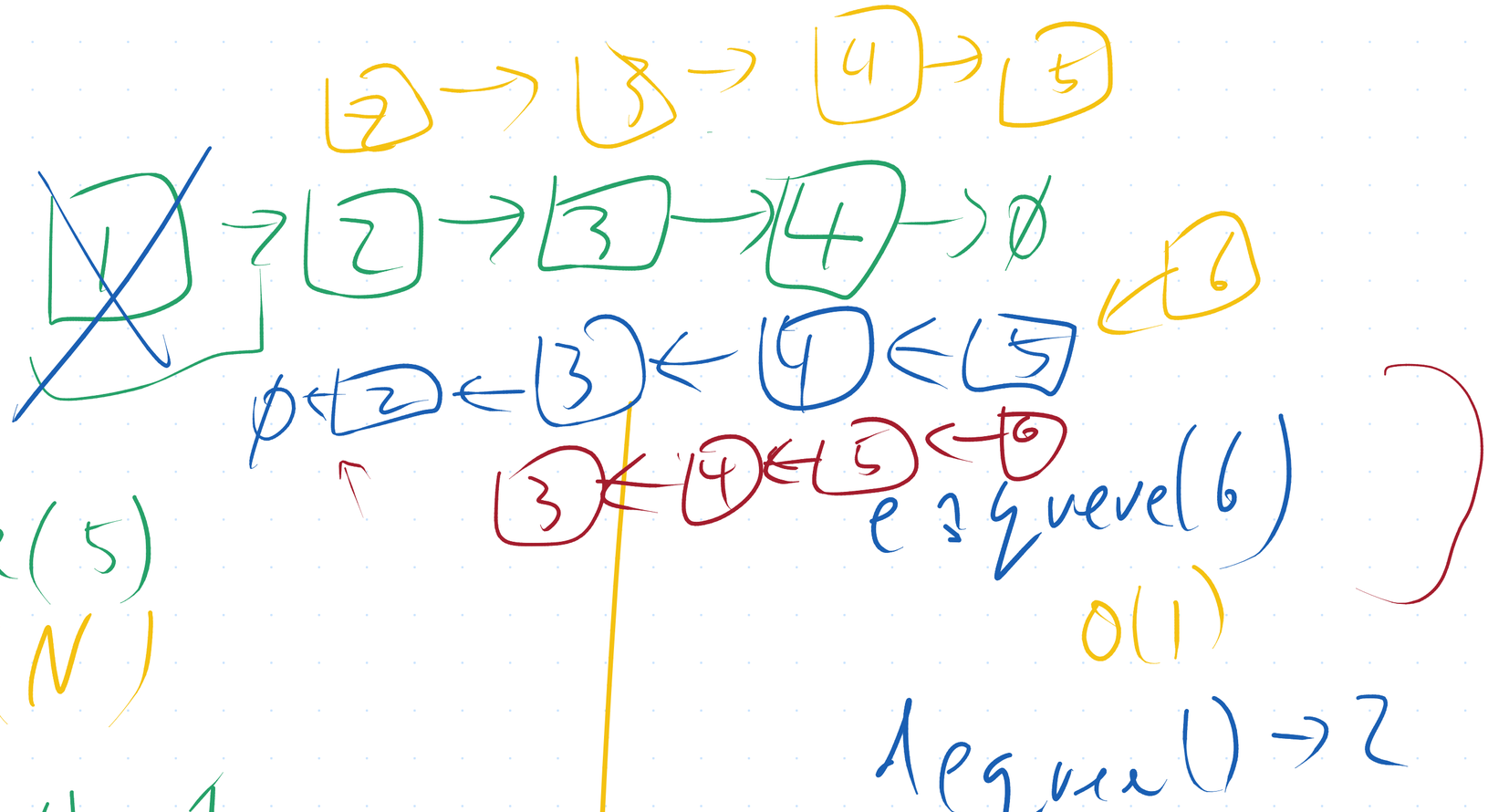


O(1) Front

O(N) copies back



All updates O(N)



enqueue(5)

O(N)

dequeue() → 1

O(1)

enqueue at back

dequeue from front

enqueue(6)

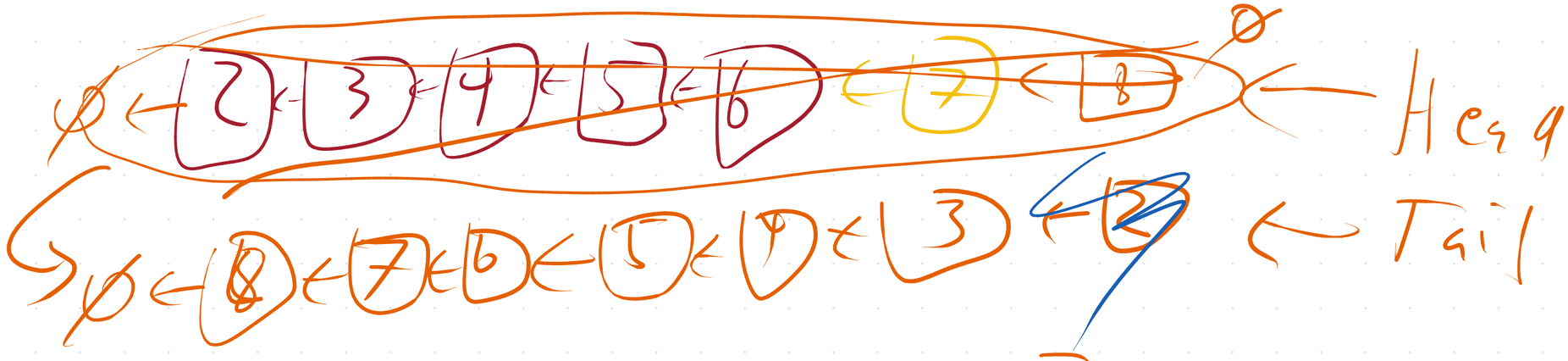
O(1)

dequeue() → 2

O(N)

enqueue at front/front

dequeue at back

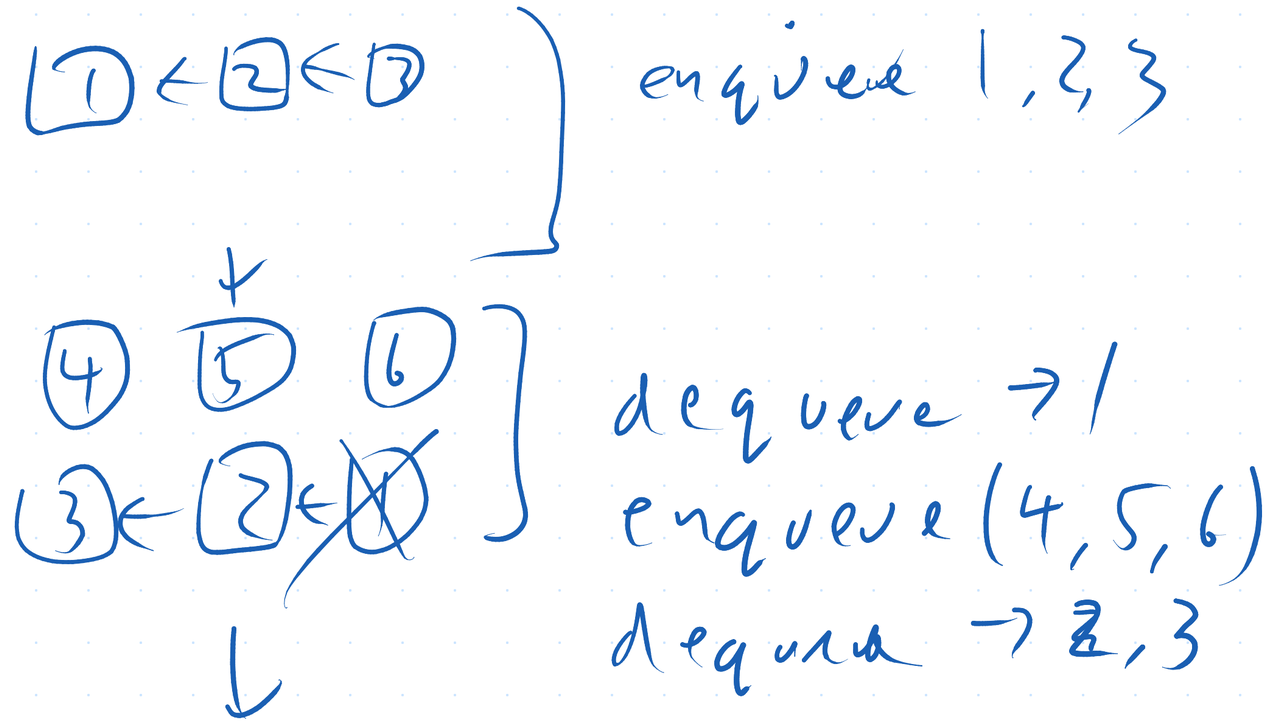


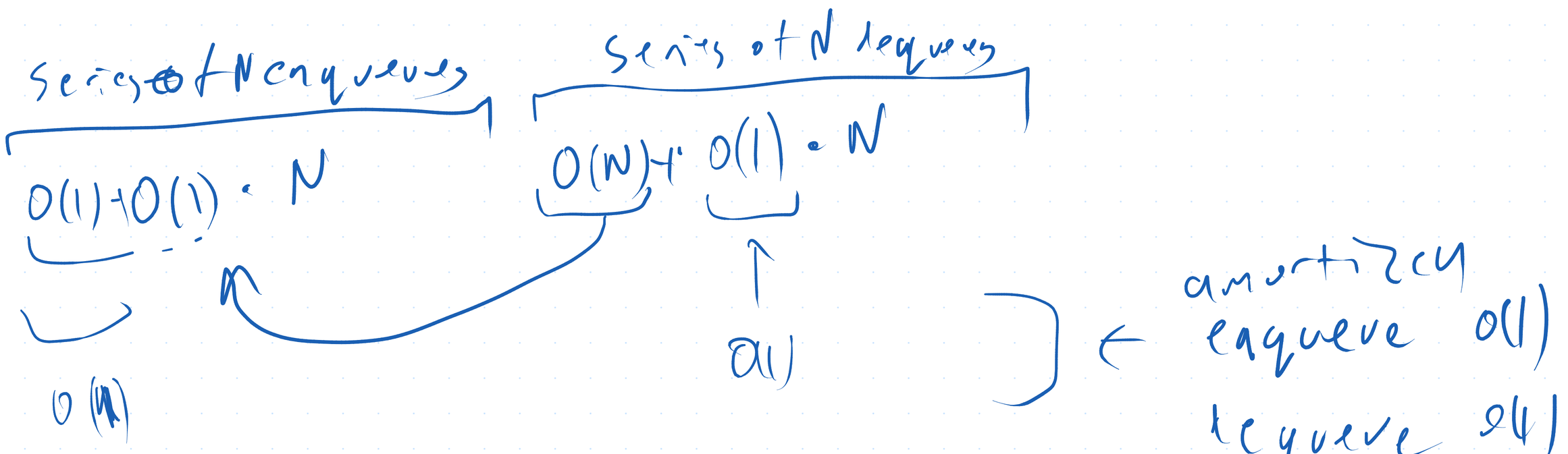
enque
- add to Front of Head Linked List] O(1)

dequeue
- If tail empty
  - reverse head queue
  - this is the new tail queue] O(N)
- remove from tail] O(1)

{ if tail empty O(N) + O(1)
{ else O(1) + O(1)

pay-it-forward savings


1 ← 2 ← 3
enqueue 1, 2, 3
4 5 6
3 ← 2 ← X X
dequeue → 1
enqueue (4, 5, 6)
dequeue → 2, 3


Series of N enqueues
O(1) + O(1) · N
O(N)
Series of N dequeues
O(N) + O(1) · N
O(1)
amortized
enqueue O(1)
dequeue O(1)