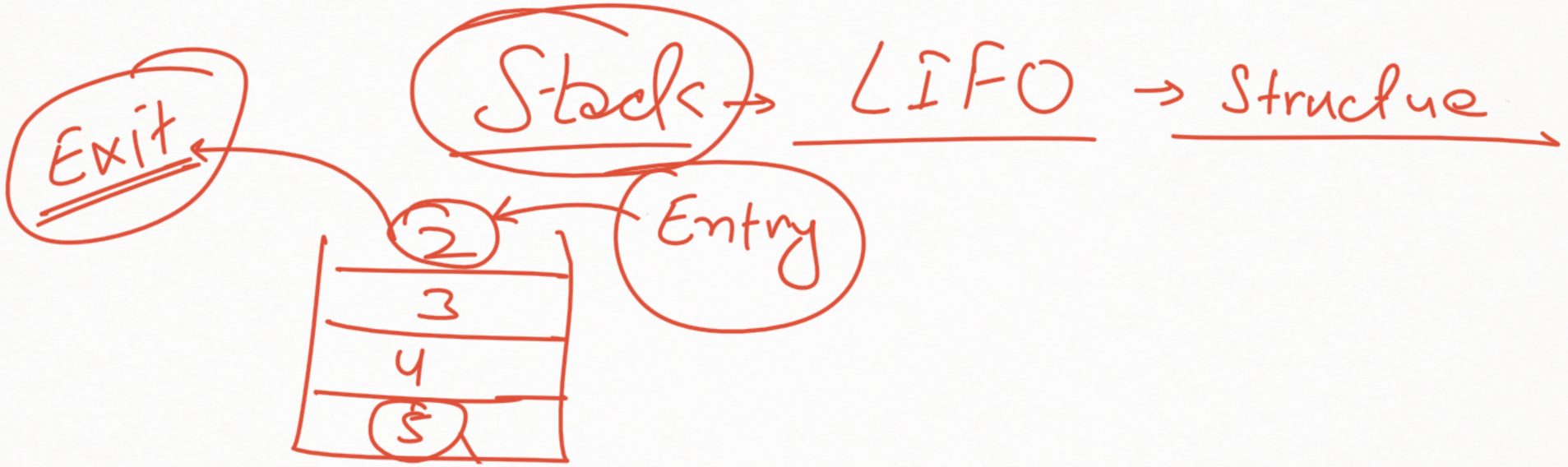
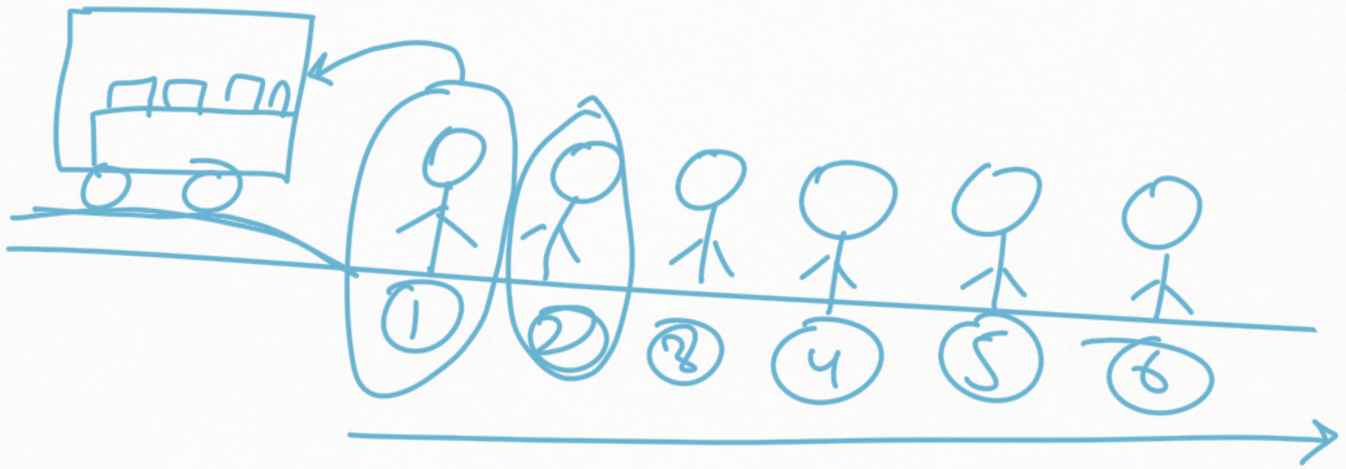
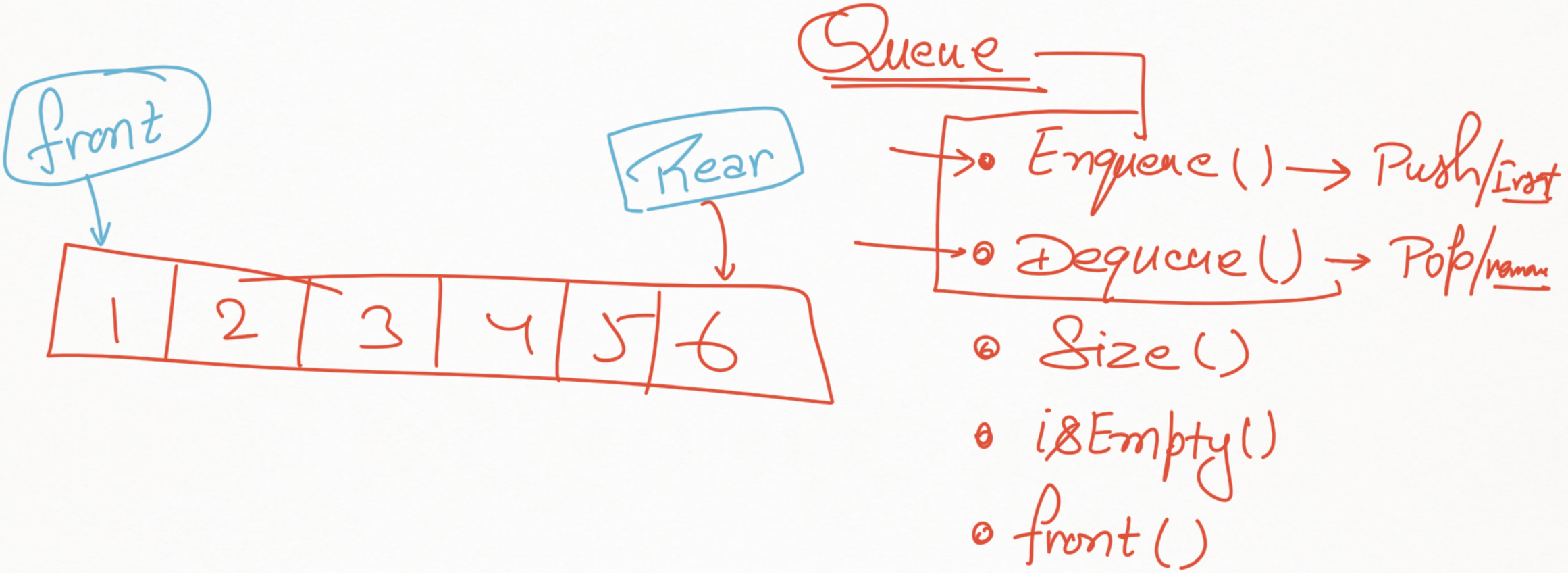
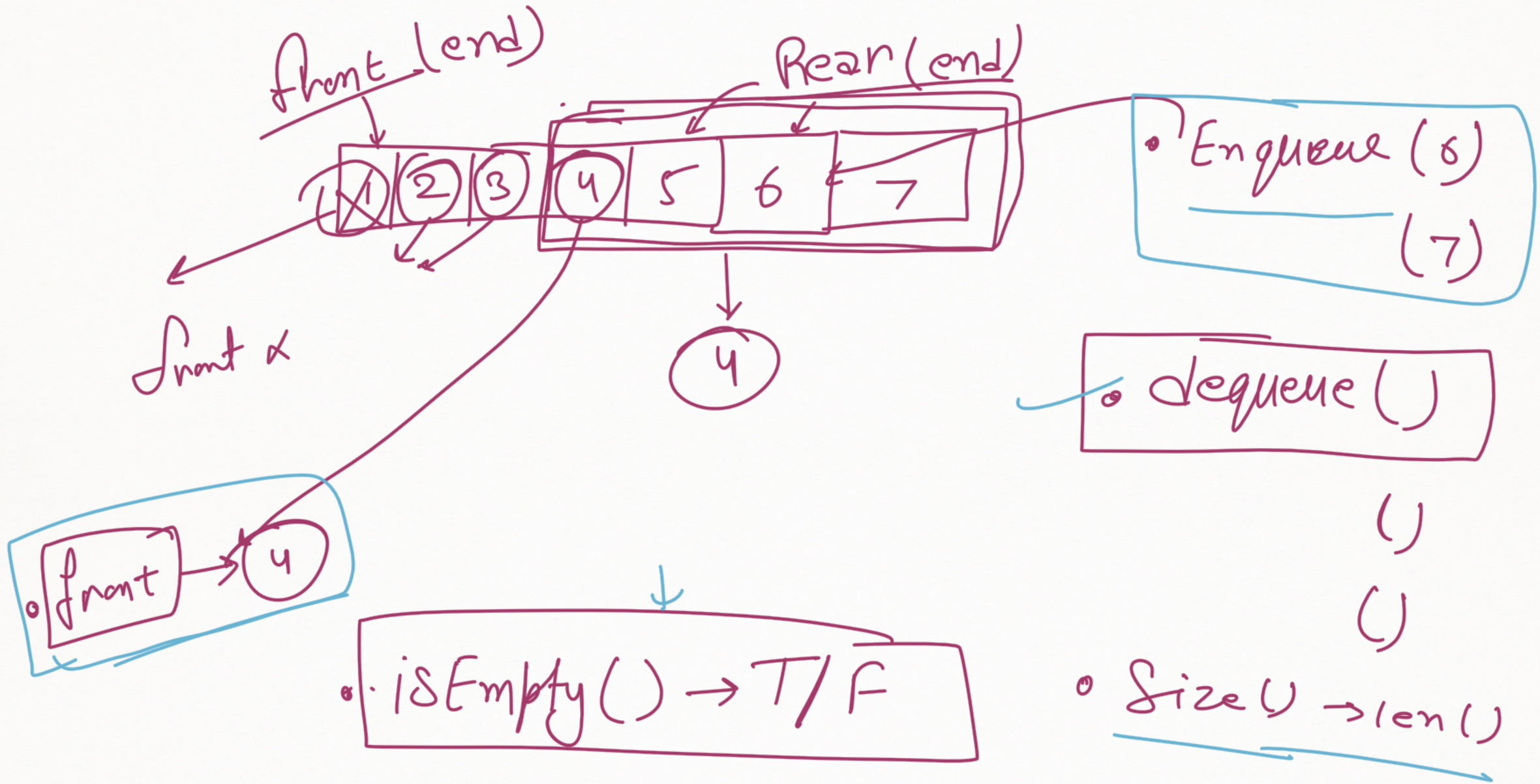

Stack → LIFO → Structue
Exit
Entry
2
3
4
5

Queue :→ FIFO → Structure

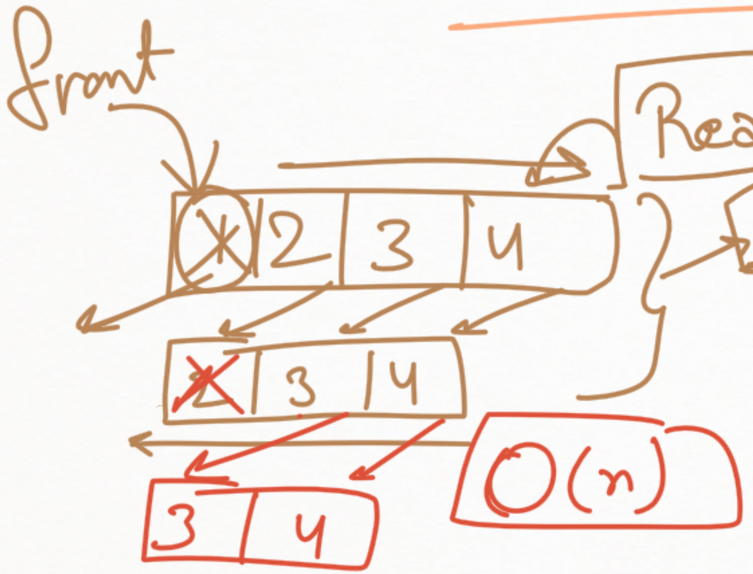
- ArrayList
- LinkedList
- Stacks
- Queue




front
Rear
1
2
3
4
5
6
Queue
Enqueue() → Push/Insert
Dequeue() → Pop/Remove
Size()
isEmpty()
front()


front (end)
Rear (end)
1
2
3
4
5
6
7
front x
4
Enqueue (6)
(7)
dequeue ()
front
4
isEmpty () → T/F
()
()
Size () → len ()

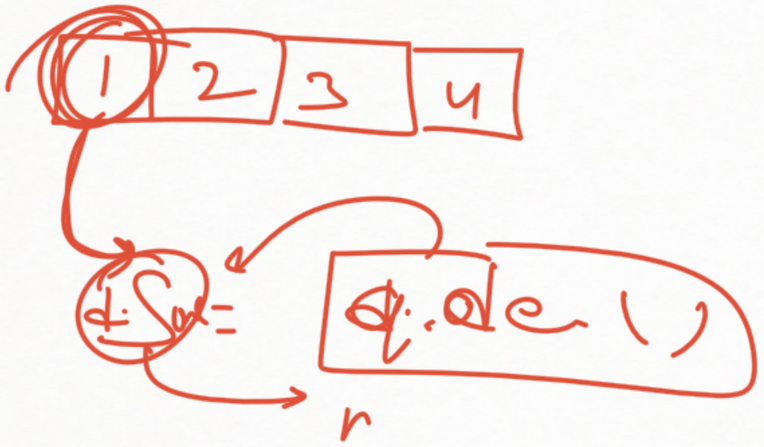
Queue Using Array :-



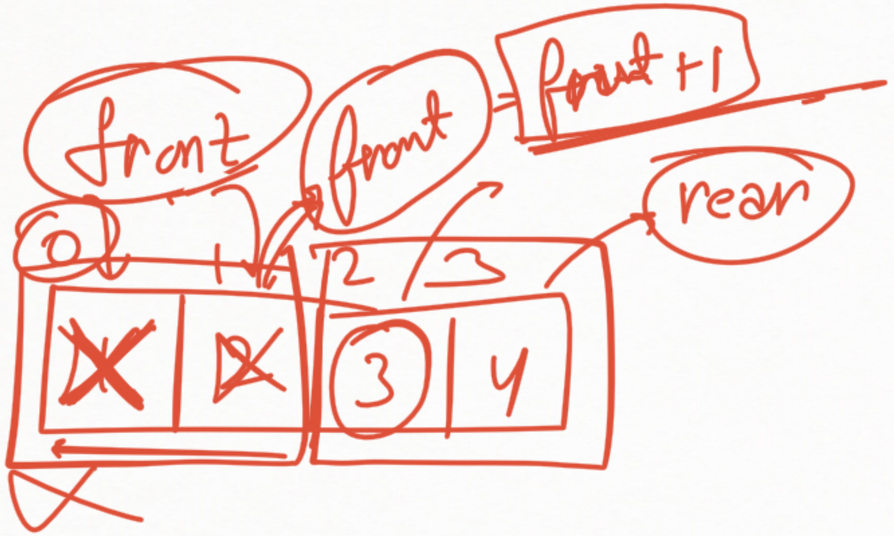
operations

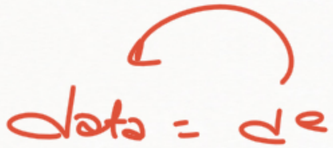
- Enqueue() → O(1)
- dequeue() → O(1)
- Size()
- isEmpty()
- front()

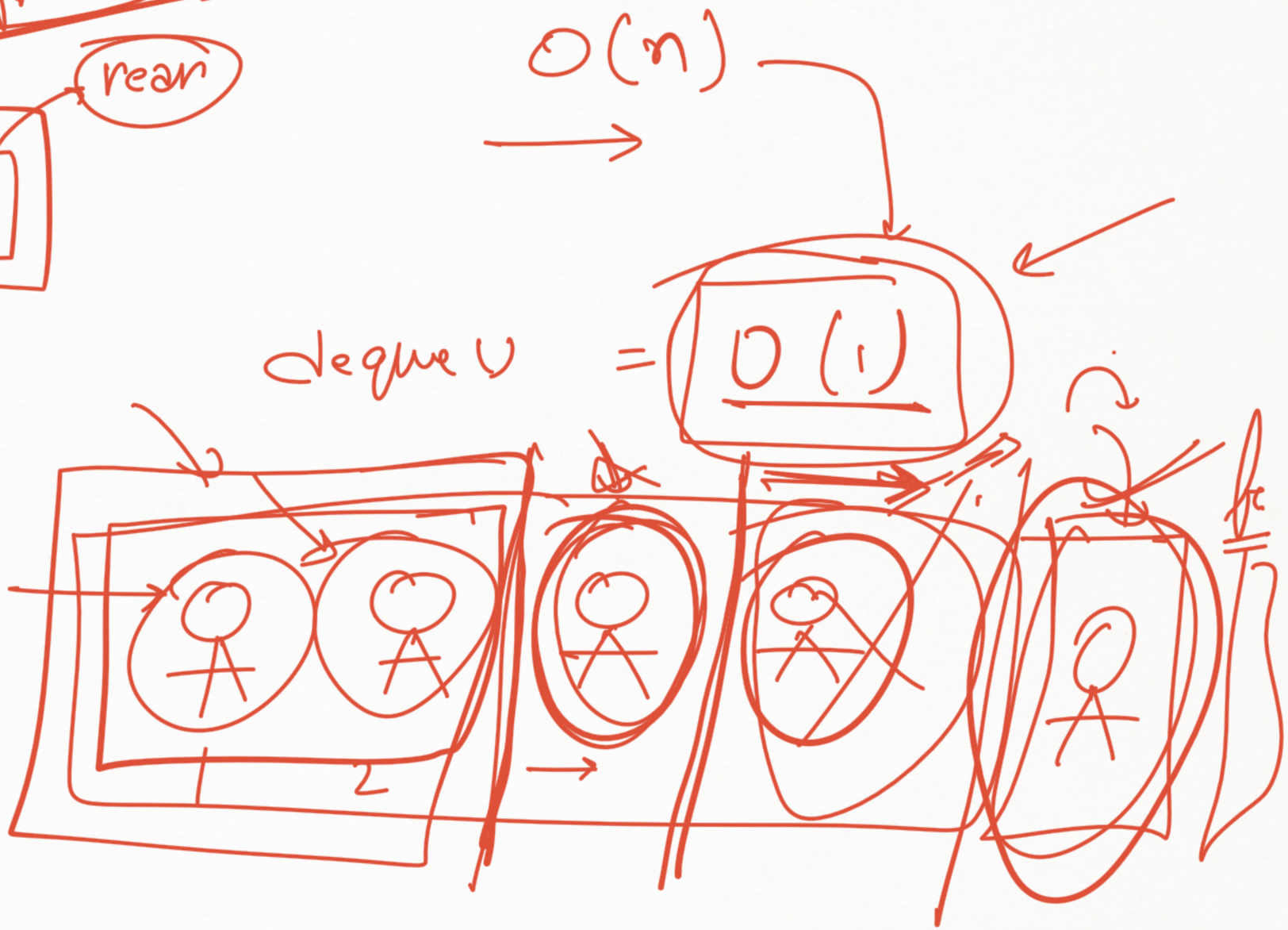
q.enqueue(1)
(2)
(3)
(4)



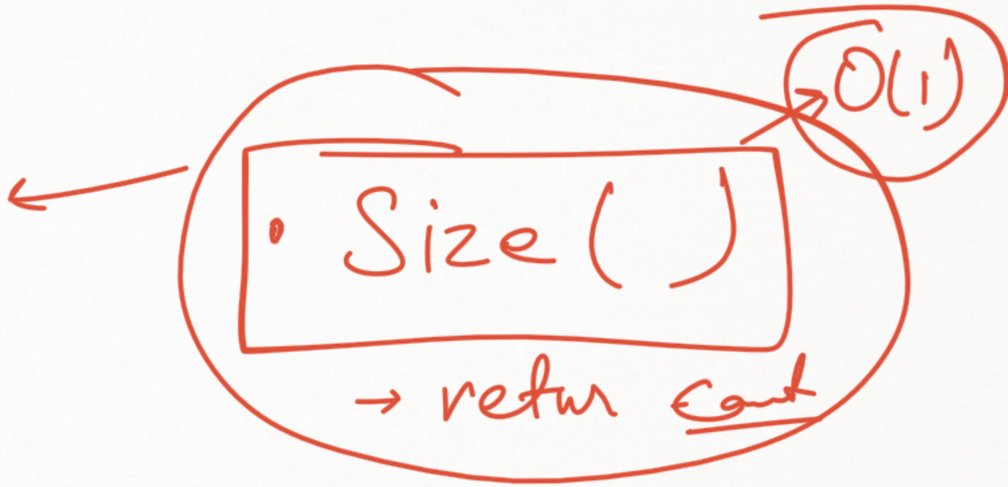
q.dequeue()
1

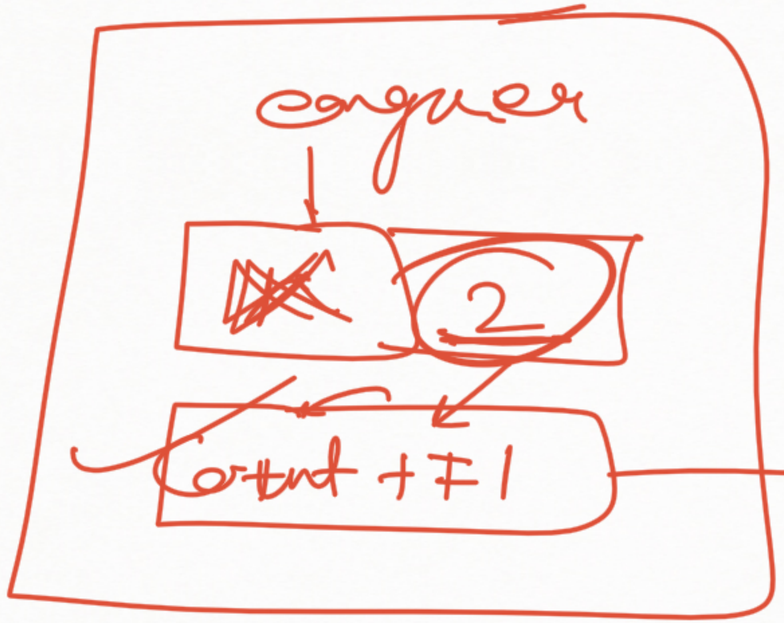

front
front
front+1
rear
0
1
2
3
3
4

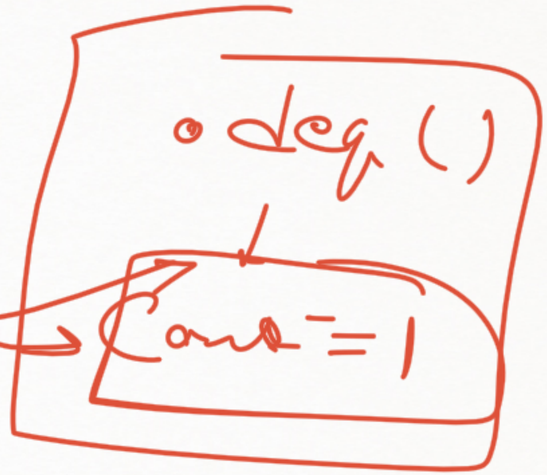

data = de


O(n)
dequeu
=
O(1)
2

- arr = [ ]
- front = 0
- Count = 0







- array = [ ]
- front = 0
- count = 0

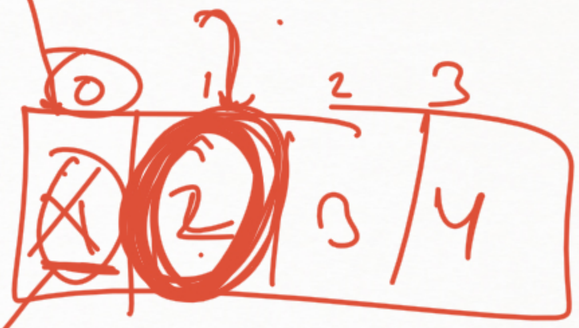
→ O(1) operation

for

element = arr [front]

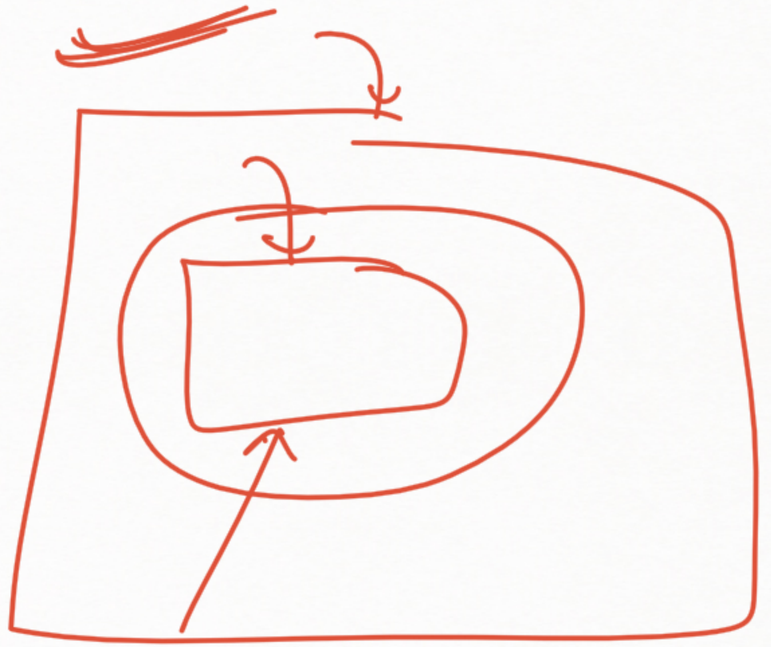
front = front + 1

return element

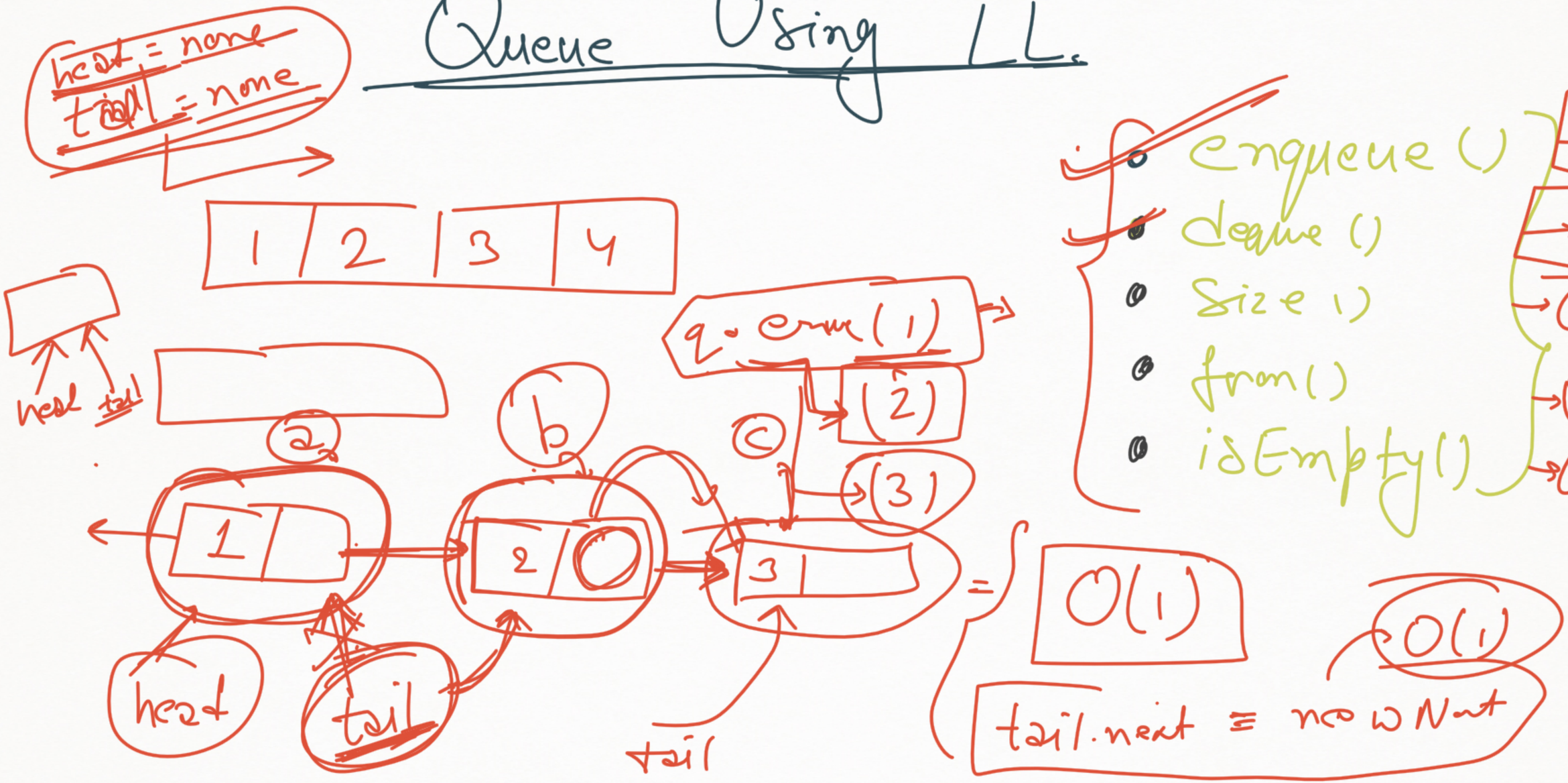


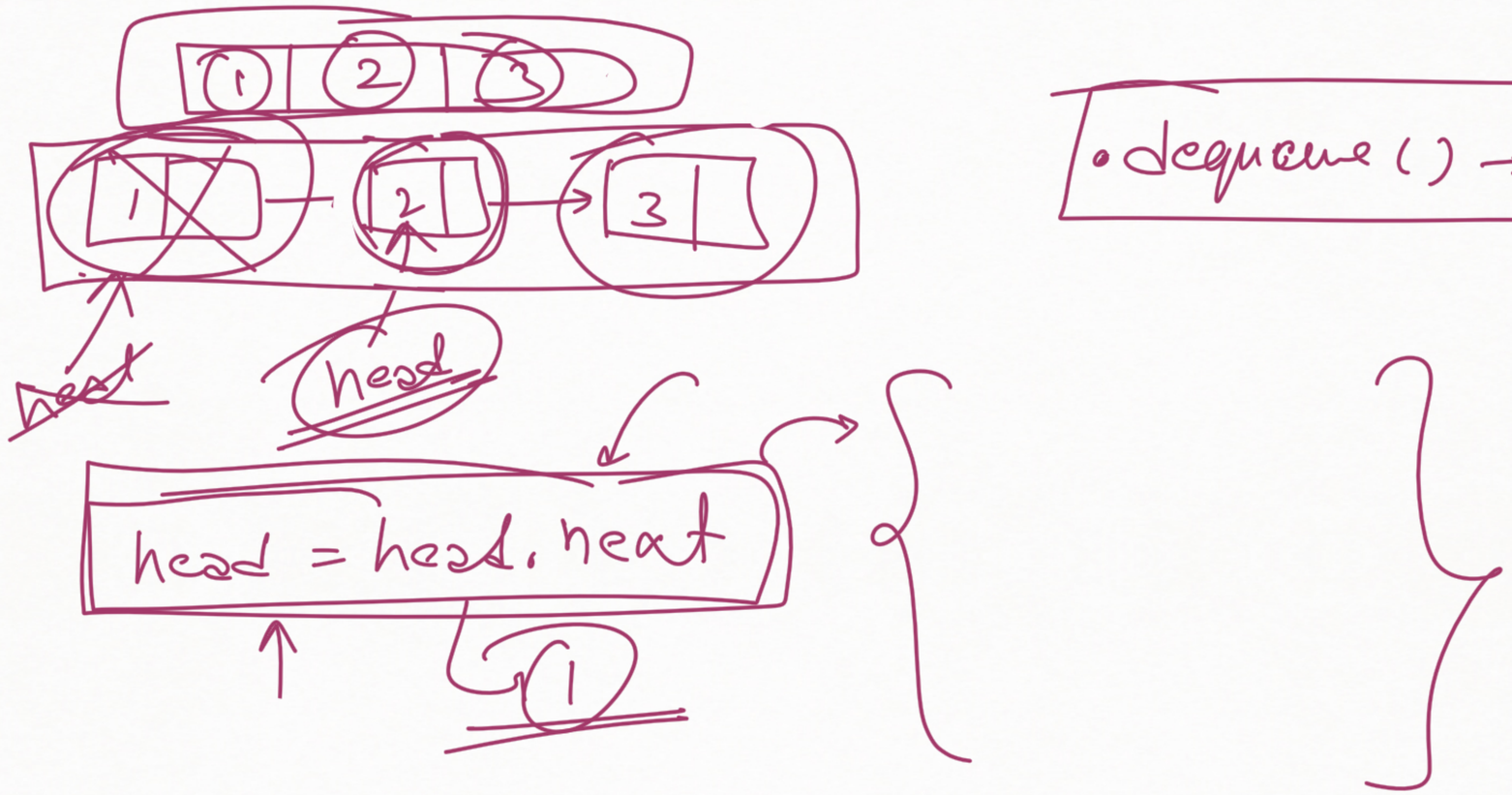
0 ⇒ Count == 0

• isEmpty = Size == 0
return ↑



Queue Using LL.





- dequeue() → O(1)

Count = +1

−1