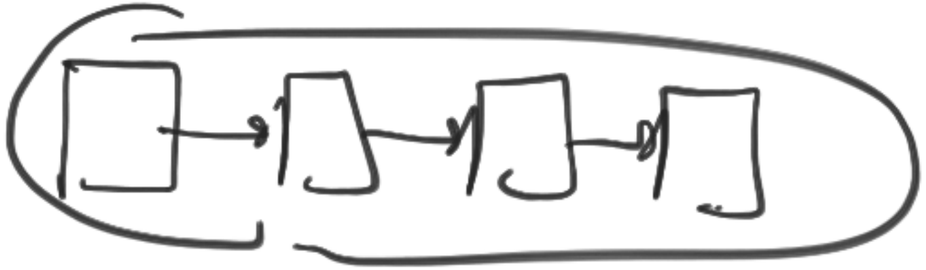
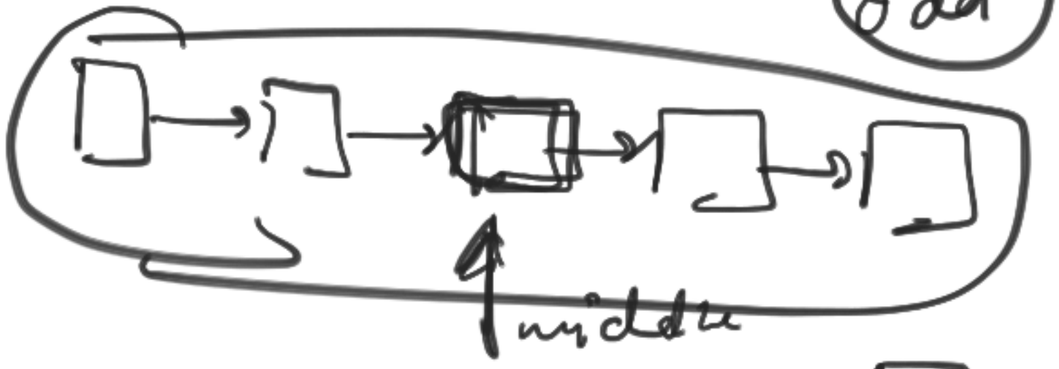
# Linked list question.



## find middle node.

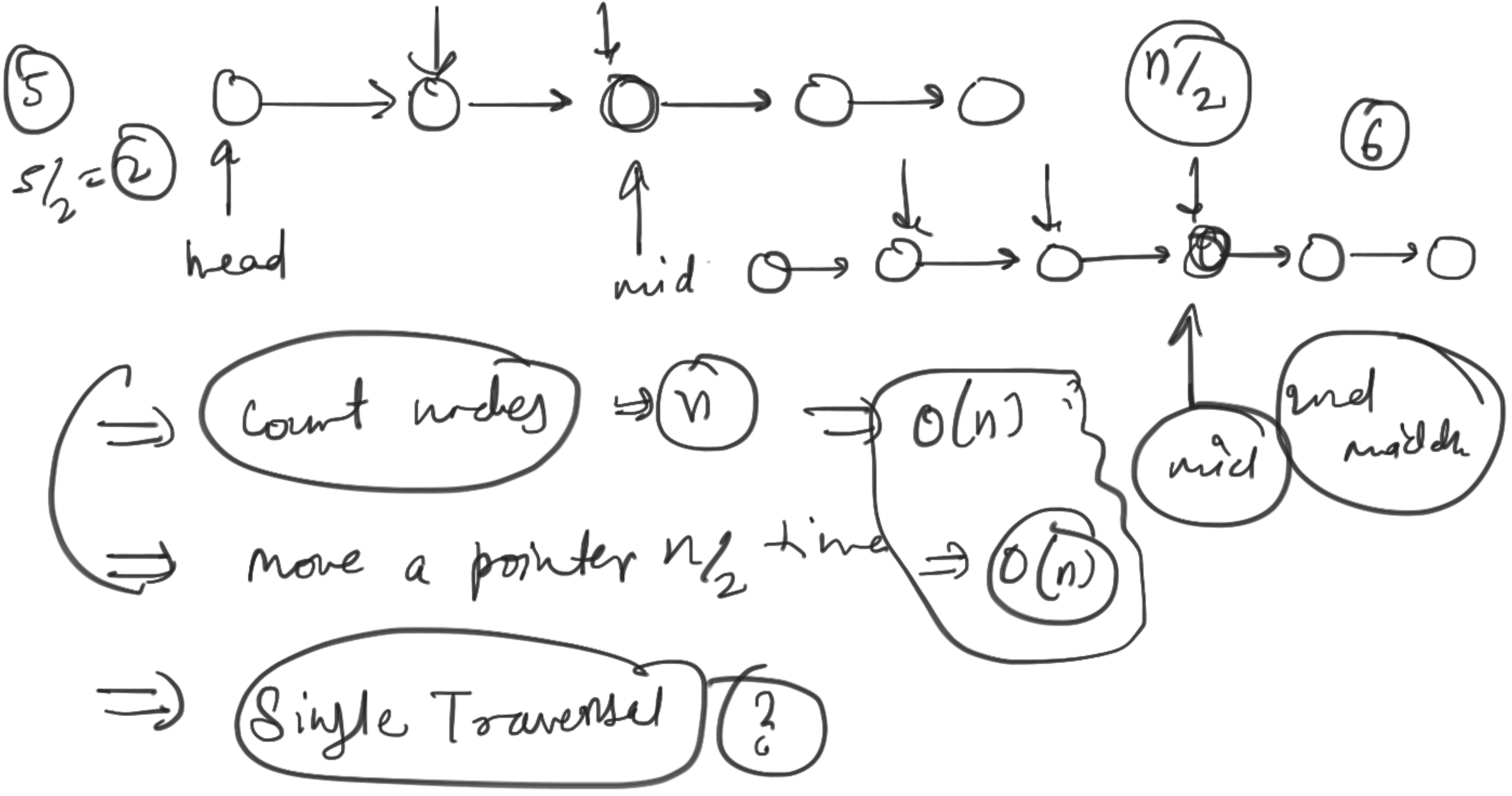
odd

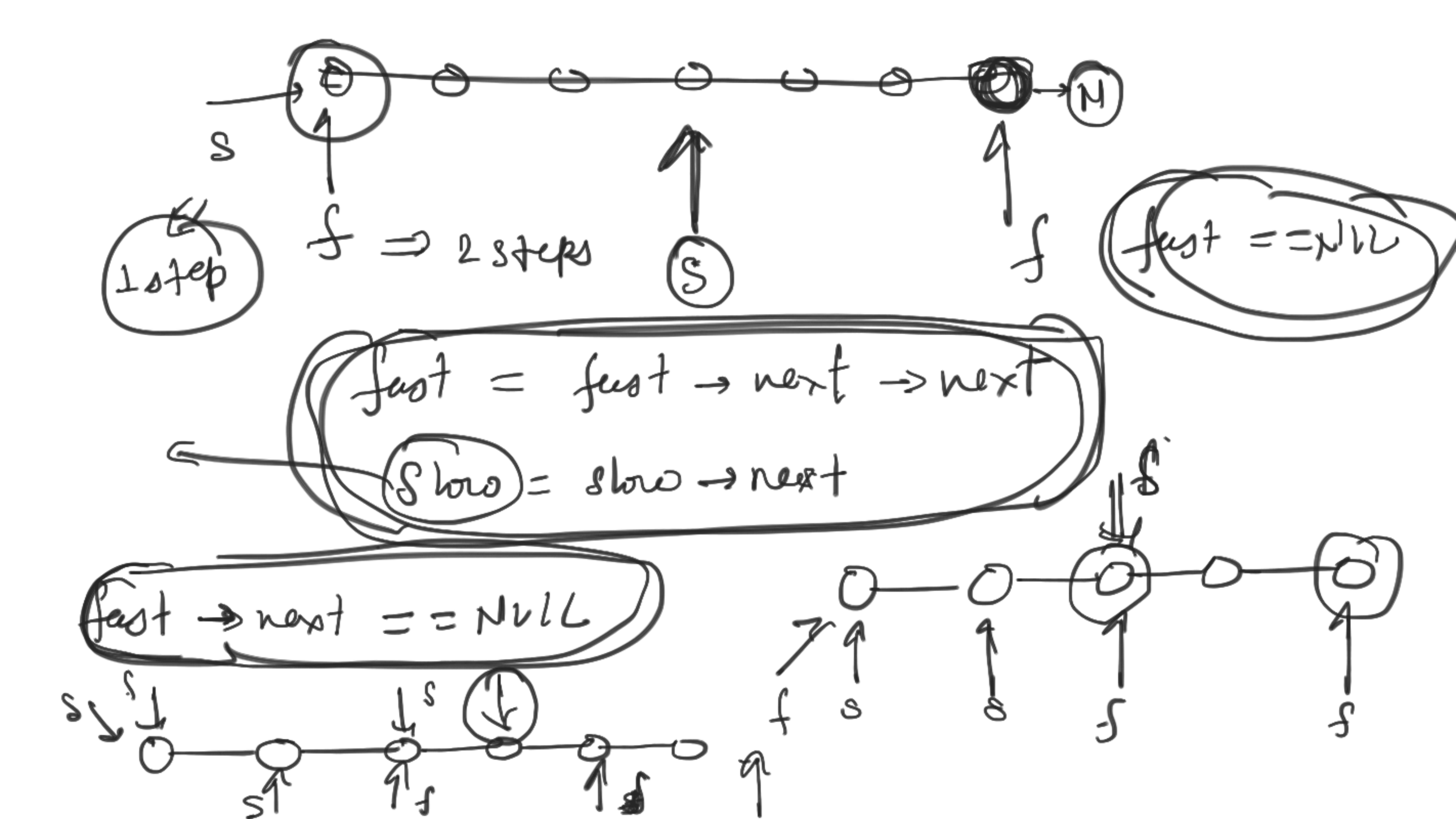



- Remove duplicates
- Reverse LL
- Partition LL
- merge LL
- Intersection

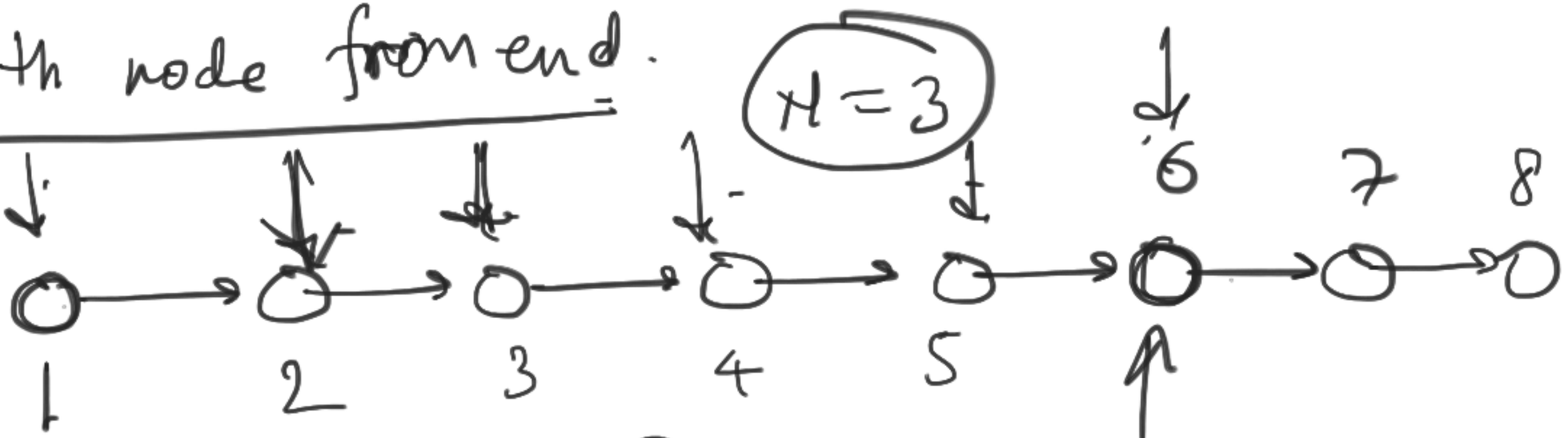
LL is a palindrom

Sort


5
n/2
6
n/2 = 2
head
mid
mid
Count nodes
n
O(n)
move a pointer n/2 time
O(n)
Single Traversed
2


s
f
N
1 step
f ⇒ 2 steps
s
f
fast == NULL
fast = fast → next → next
slow = slow → next
fast → next == NULL
s
f
s
f
s
f
f
s
s
f
f

# find Nth node from end.




count = 8 $\Rightarrow$ O(n) $\Rightarrow$ 1

M = 3

$\Rightarrow$ count - M + 1 the 3rd node from end.

$\Rightarrow$ count - M

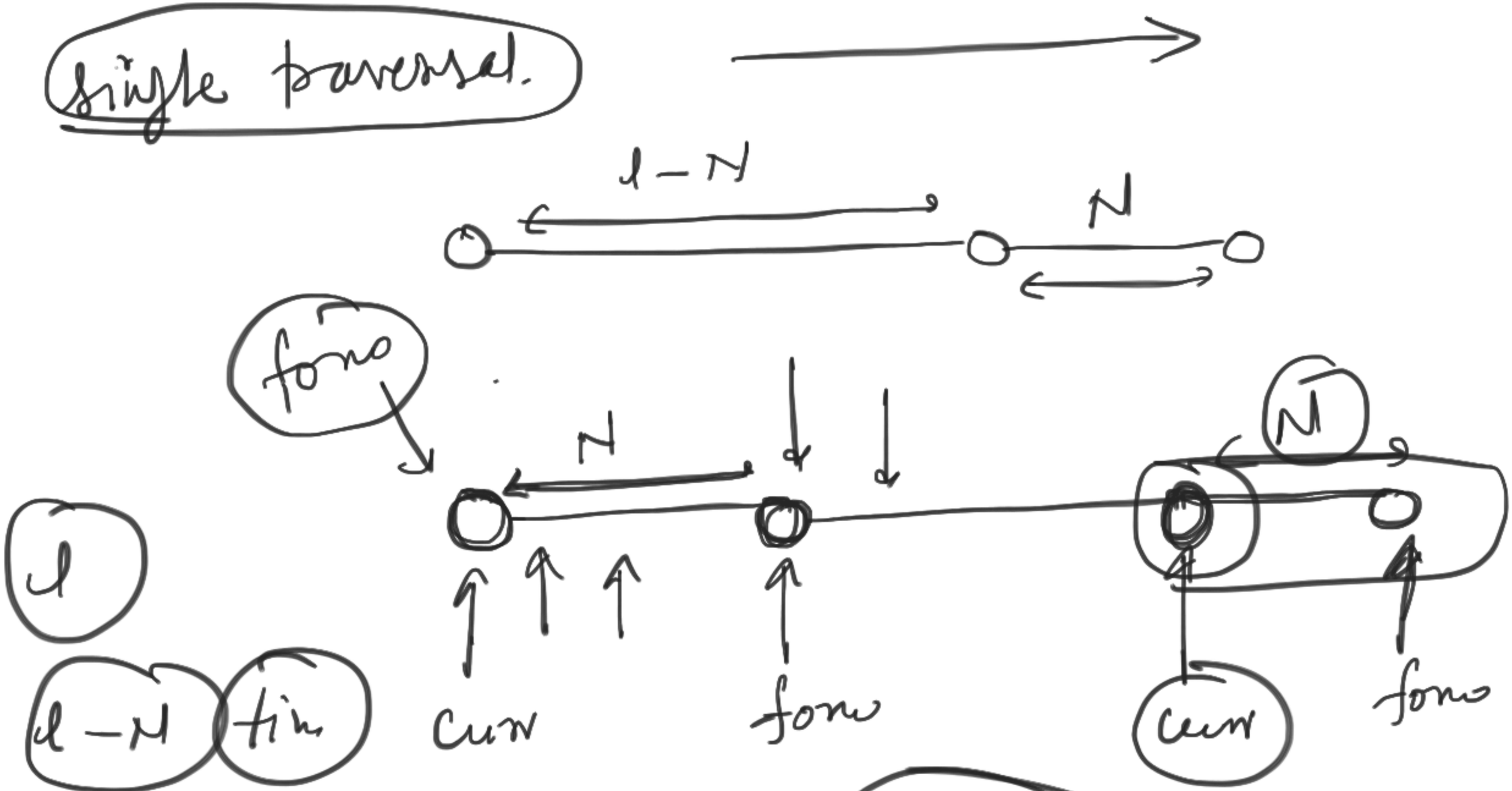
step $\Rightarrow$ 5

Double Traversal

8 - 3 + 1

1 + L - M $\Rightarrow$ 6 node from L - M

Single traversal.

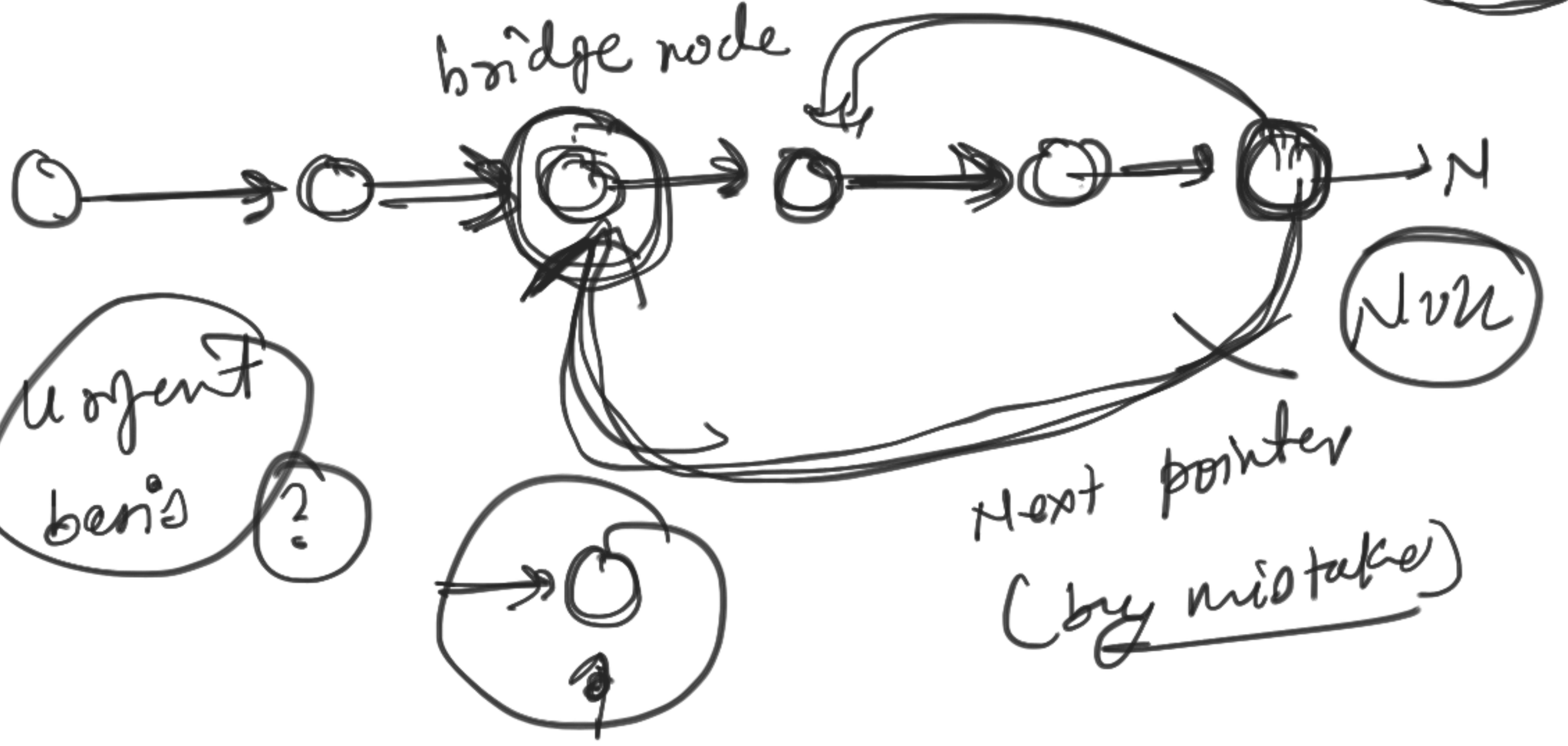



$$\Rightarrow \quad l + l - M = 2l - M$$

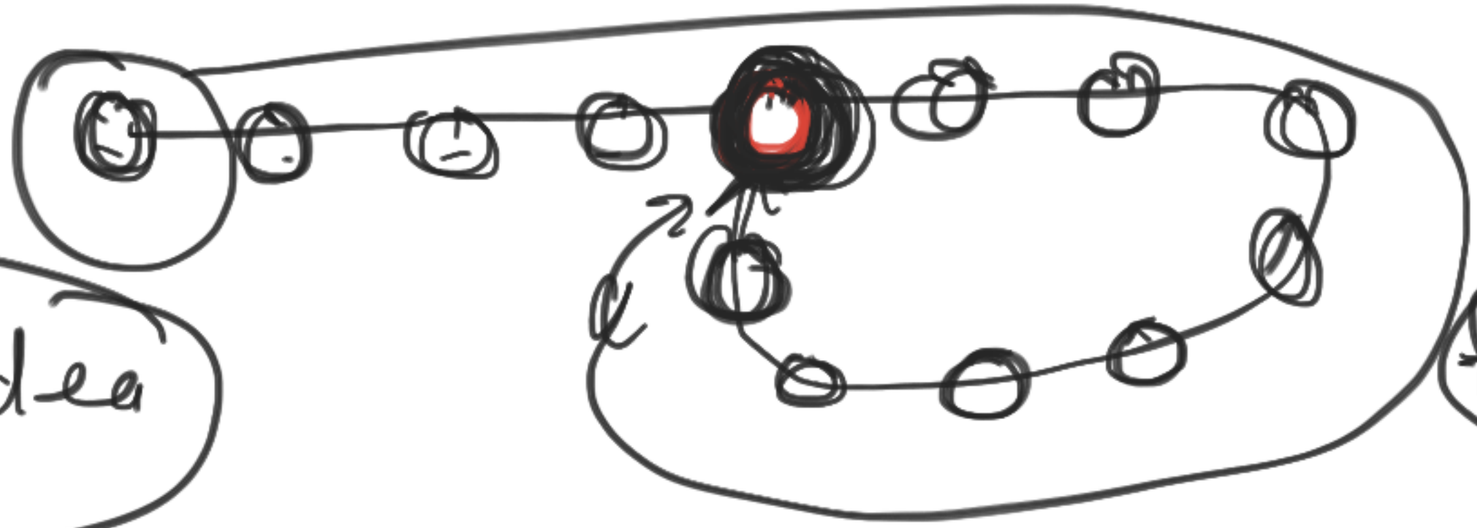
Detect loop in the LL. ⇒ true, false

infinite loop

Remove loop in the LL ⇒



Basic idea



floyd algorithm

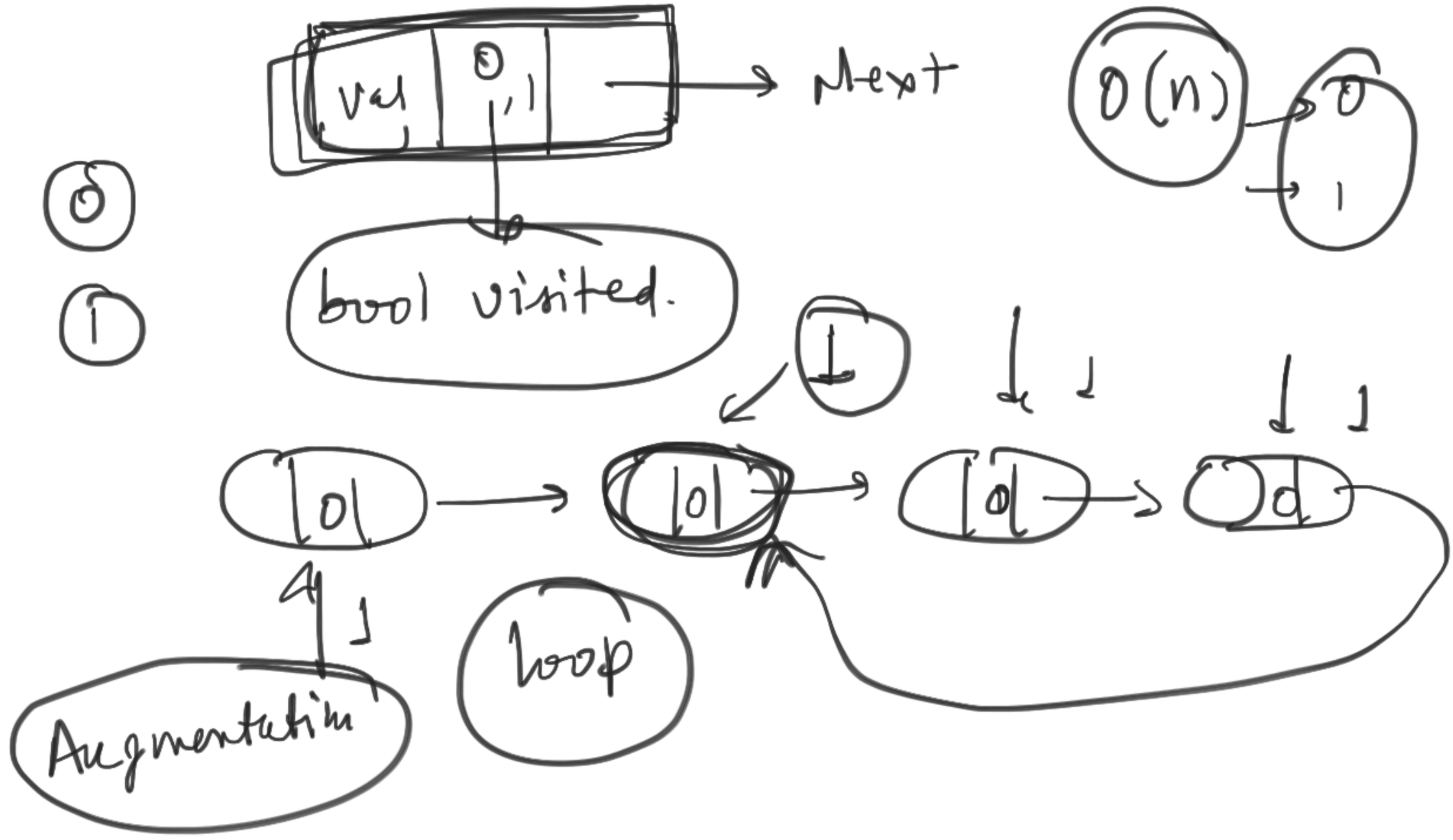
Hash map ⇒ curr, insert

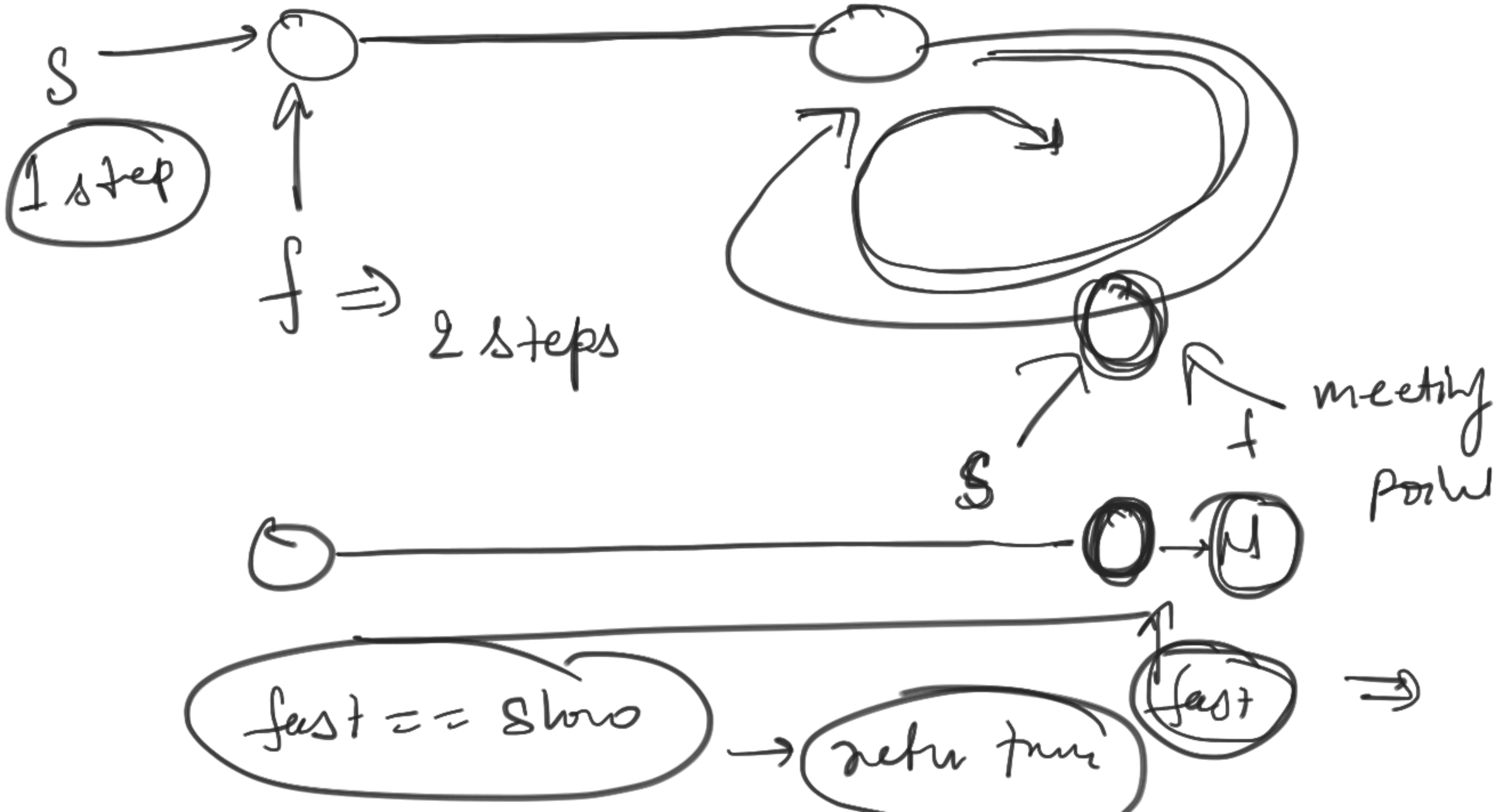
Hash map < ListNode>

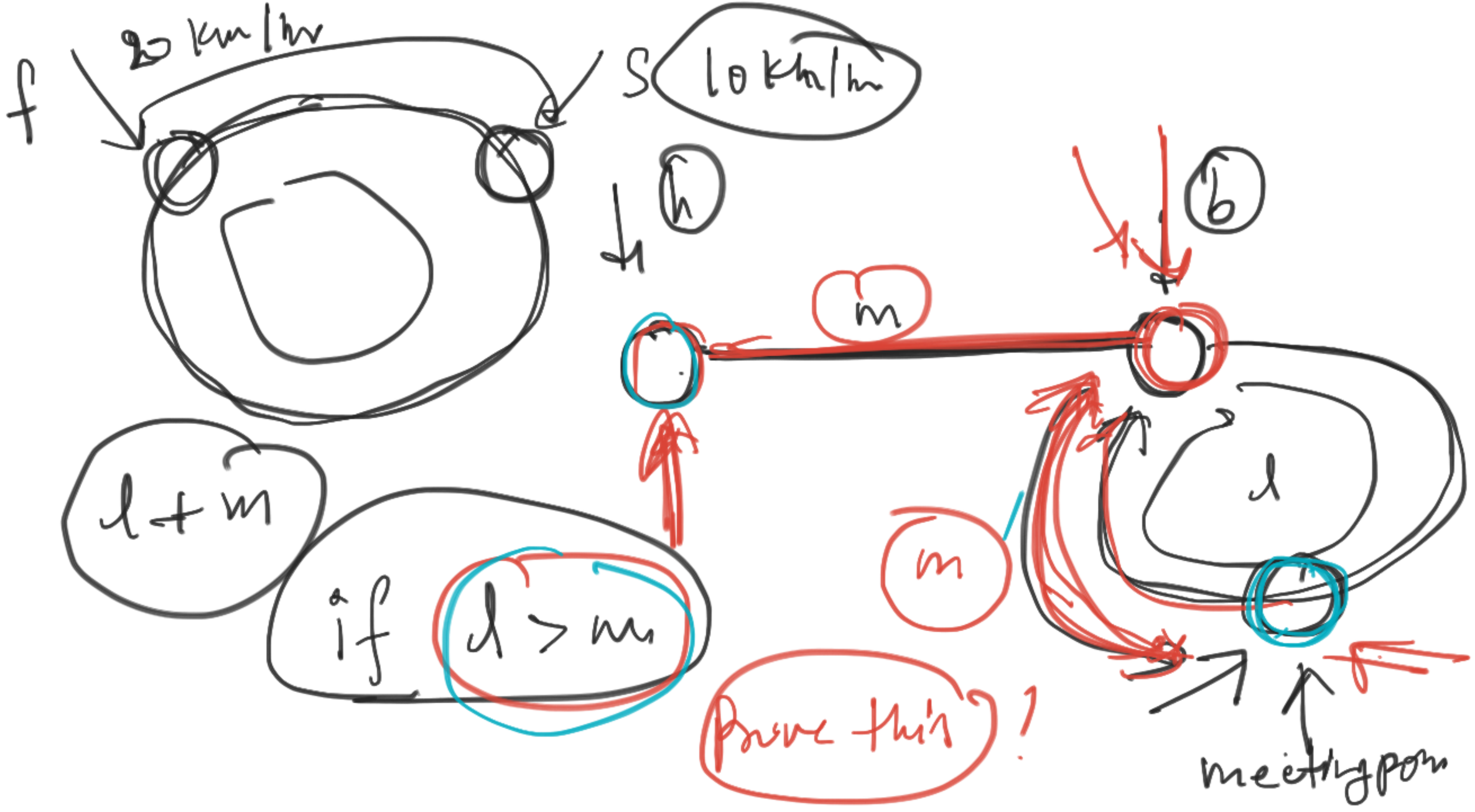
O(n)

O(n)

extra space


val
0
1
Next
O(n)
0
1
bool visited.
0
1
0
1
0
1
0
1
Augmentation
loop


S
1 step
f
2 steps
S
f
meeting point
fast == slow
return true
fast


f
20 km/hr
S
10 km/hr
h
b
m
l + m
if
l > m
m
l
Prove this!
meeting point

if (l < m)