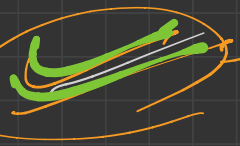
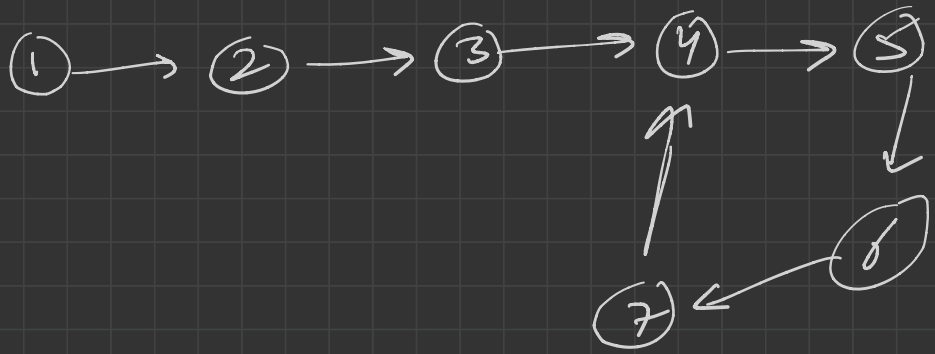
Linked List Questions

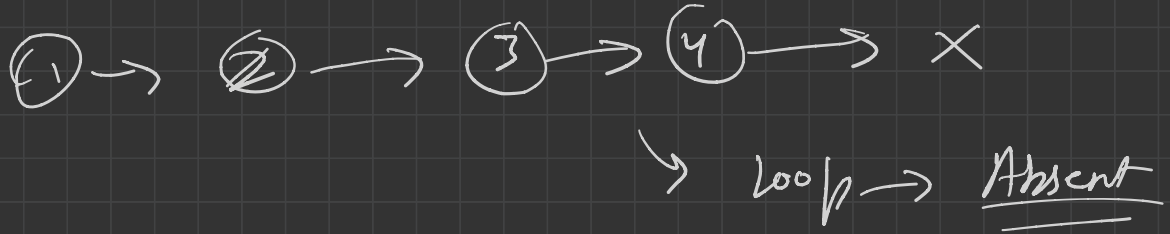
→ Detect cycle in LL  → FCD

→ Remove Cycle from LL

→ Beginning / start node of loop in LL

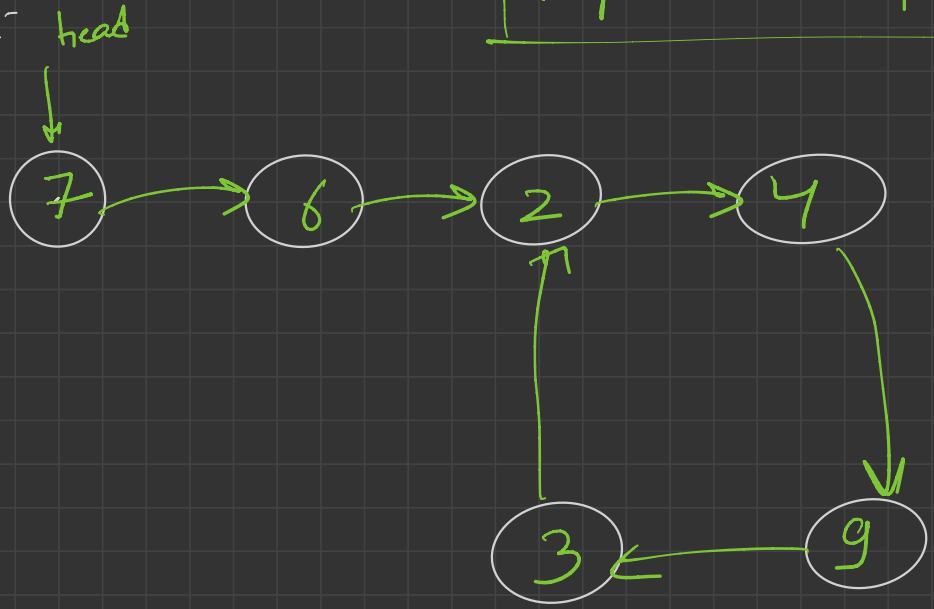
→ FCD


1
2
3
4
5
6
7


1
2
3
4
X
Loop
Absent

Approach:-

map < Node*, bool > visited;



① detect cycle → T/F

map

Approach

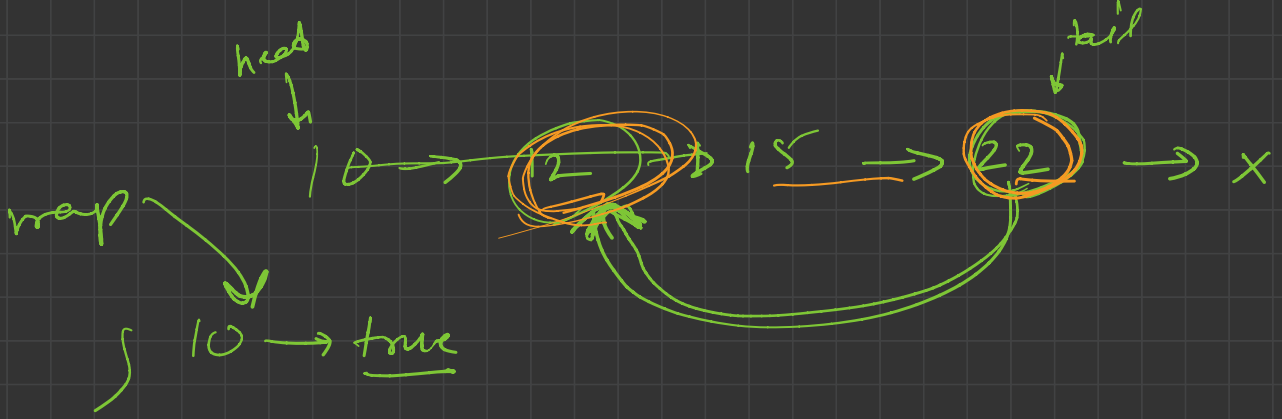
3	"str"

4	"babbar"

7	"love"

m C++ STL

map → ?



12 → true
15 → true
22 → true

12 → tu toh pehle se hi true

↓

Loop is present

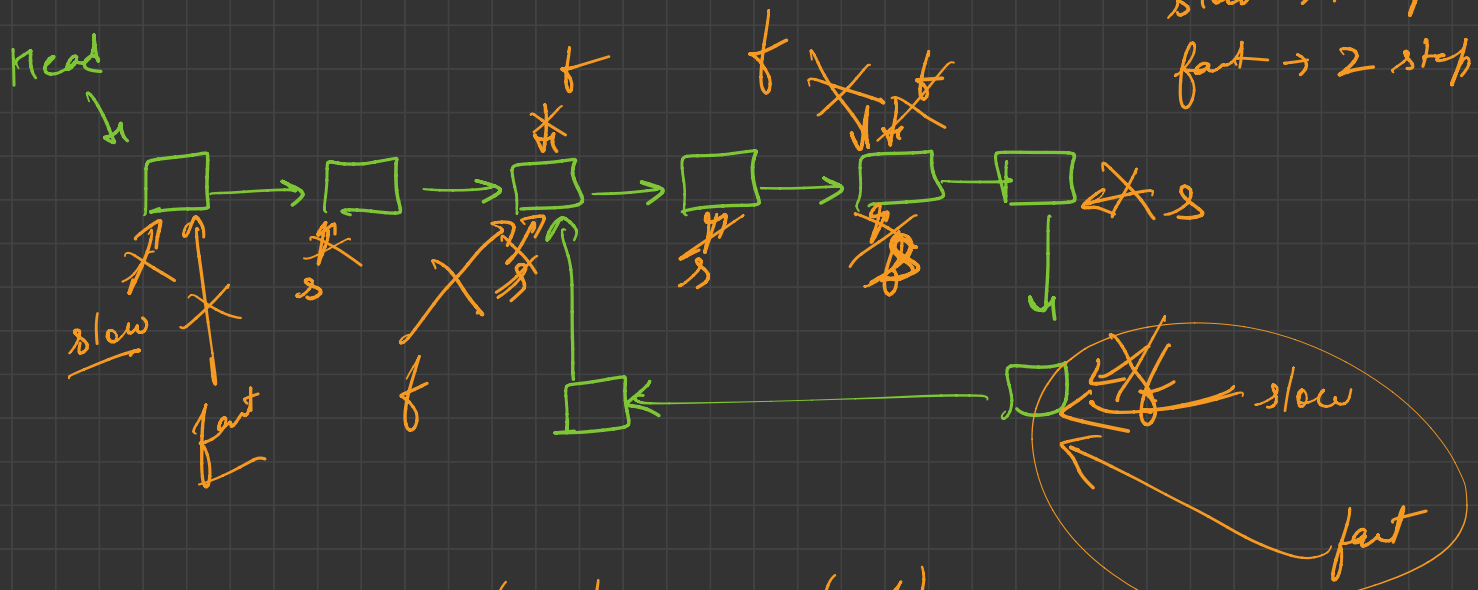
S.C → Map → O(n)

T.C → O(n)

→ O(1)

① Floyd's cycle detection algo:

slow → 1 step
fast → 2 step

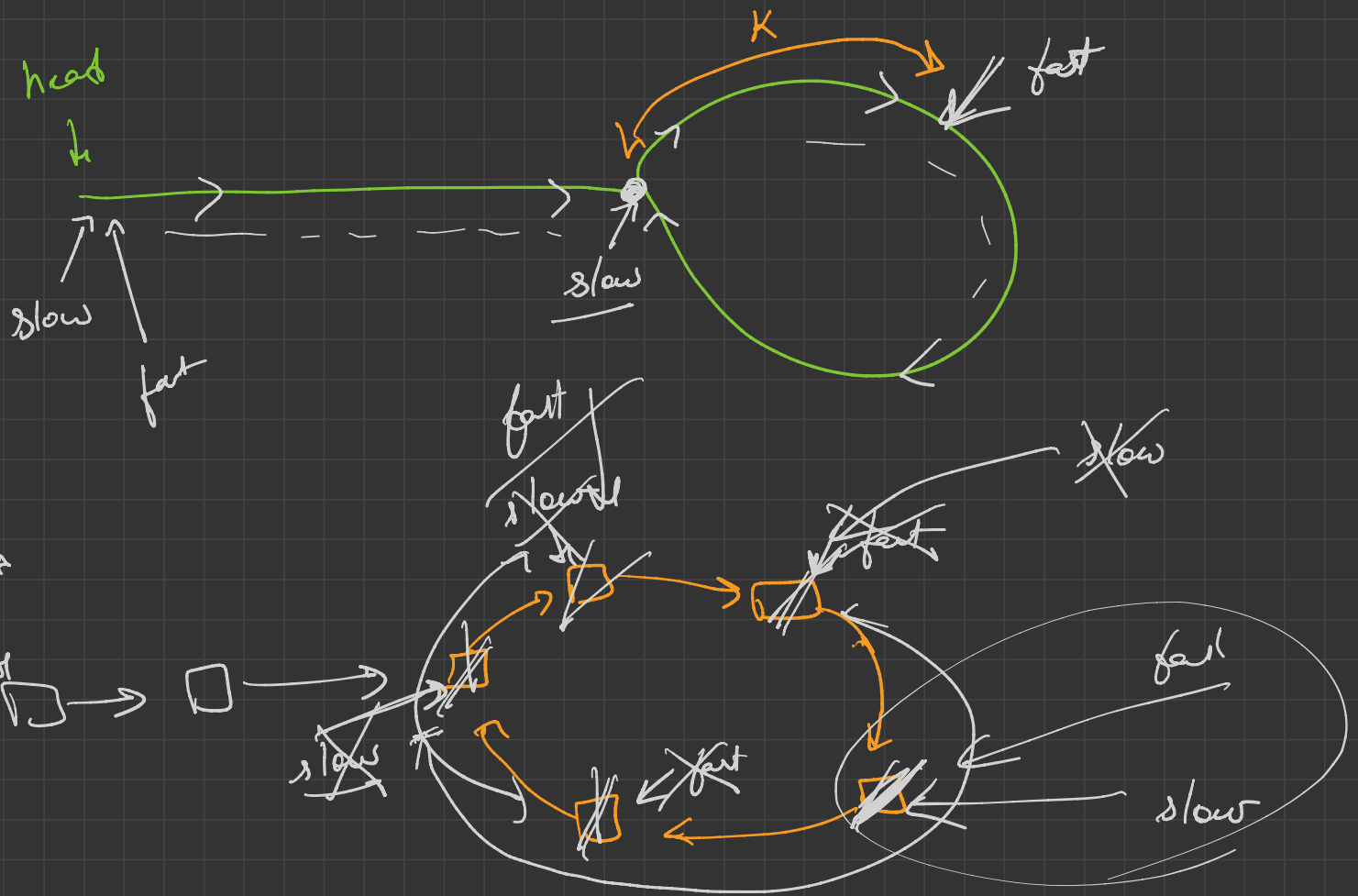


with `if (slow == fast)` → loop is present

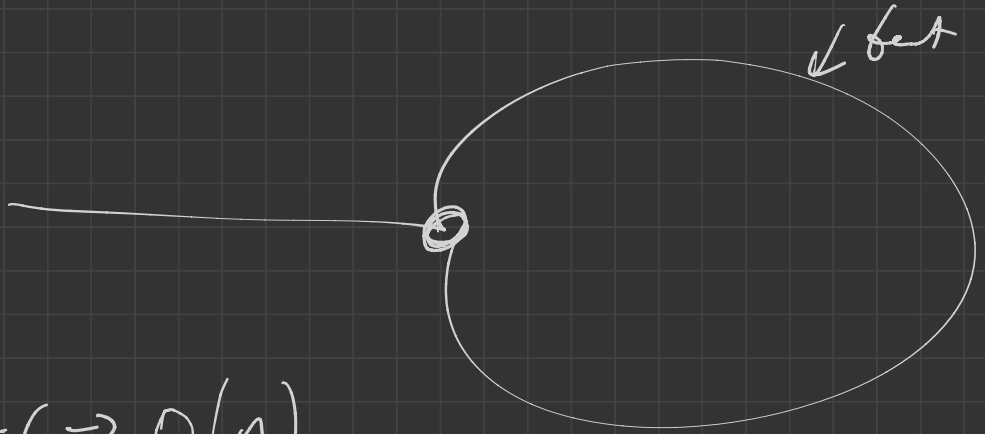
(2) fast = NULL → No Loop

T.C → O(n)

S.C → O(1)


head
h
slow
fast
K
h
fast
slow
fast
slow
head
slow
slow
fast
fast
slow
fast
slow

Distance: 4, 3, 2 1



iteration

distance

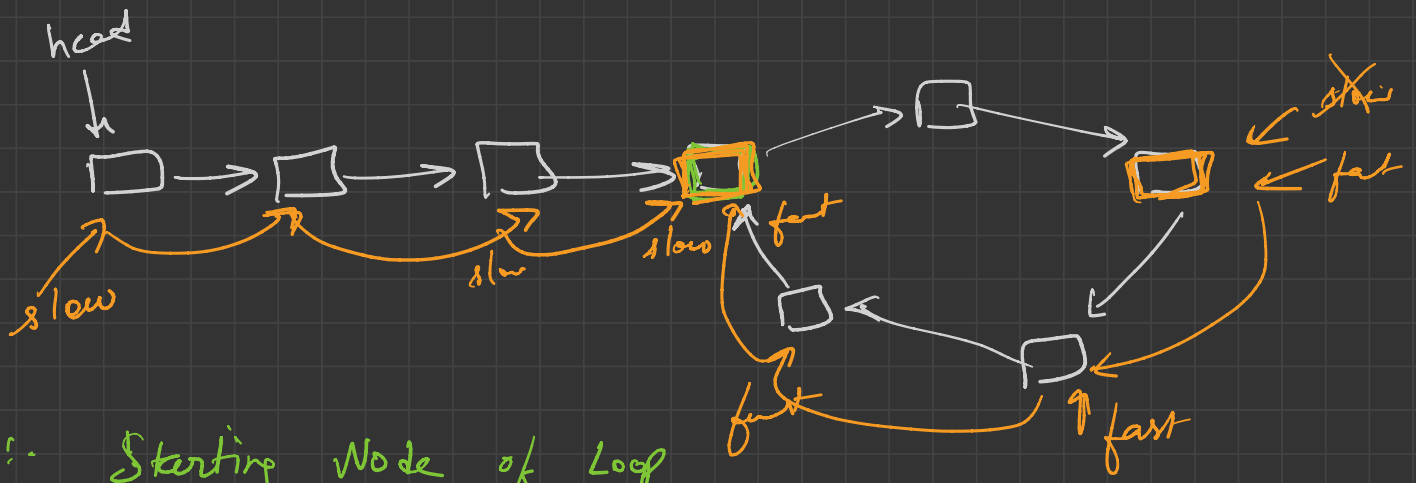
-1

at one point

meet

T.C → O(n)

S.C → O(1)



Qus:- Starting Node of Loop

Approach:-

I → FCD Algo → Point of Intersection

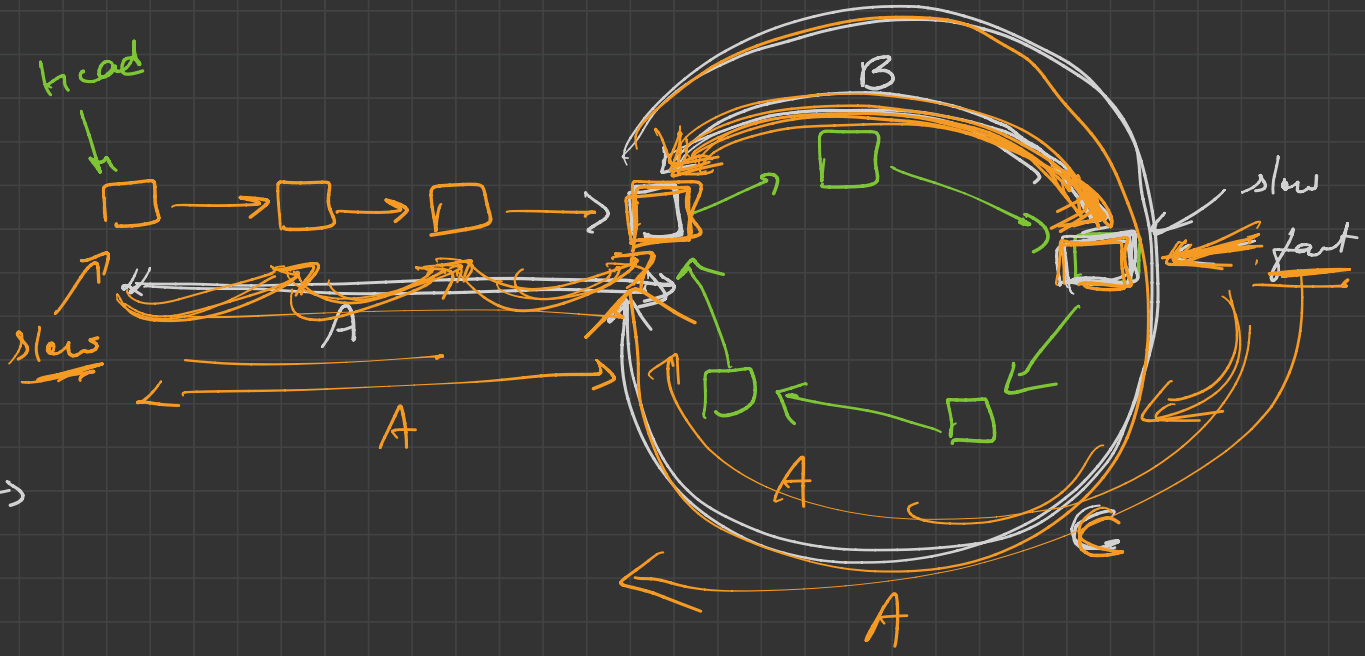
II → slow = head

↳ slow, fast → same pace

1 1/1

when(slow == fast)

↳ starting point of Loop



A →

Distance by fast pointer = 2 * Distance by slow pointer

$$(A + x * C + B) = 2 * (A + y * C + B)$$

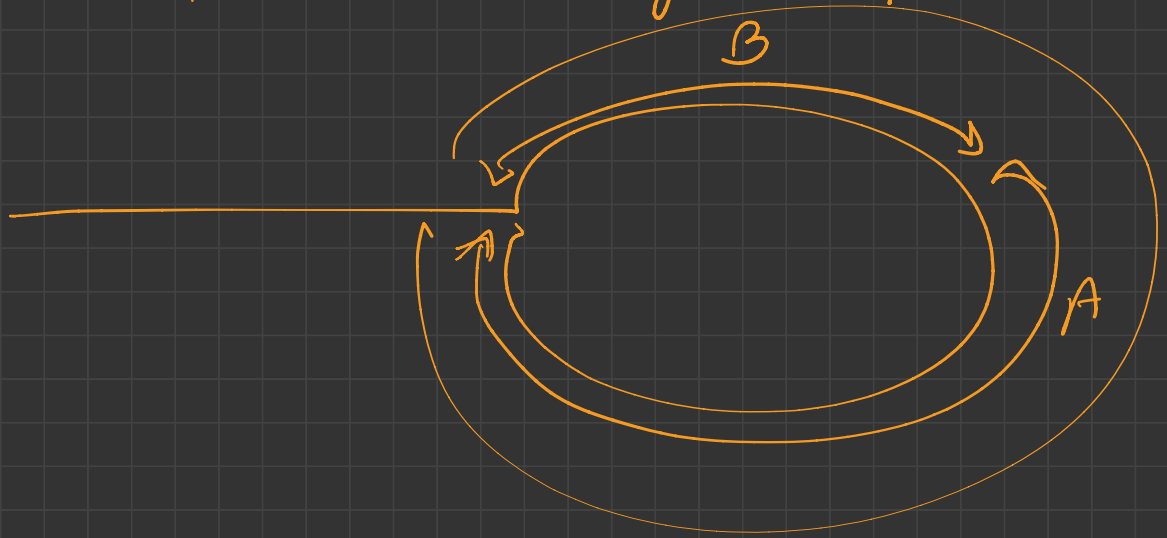
$$A + xC + B = 2A + 2yC + 2B$$

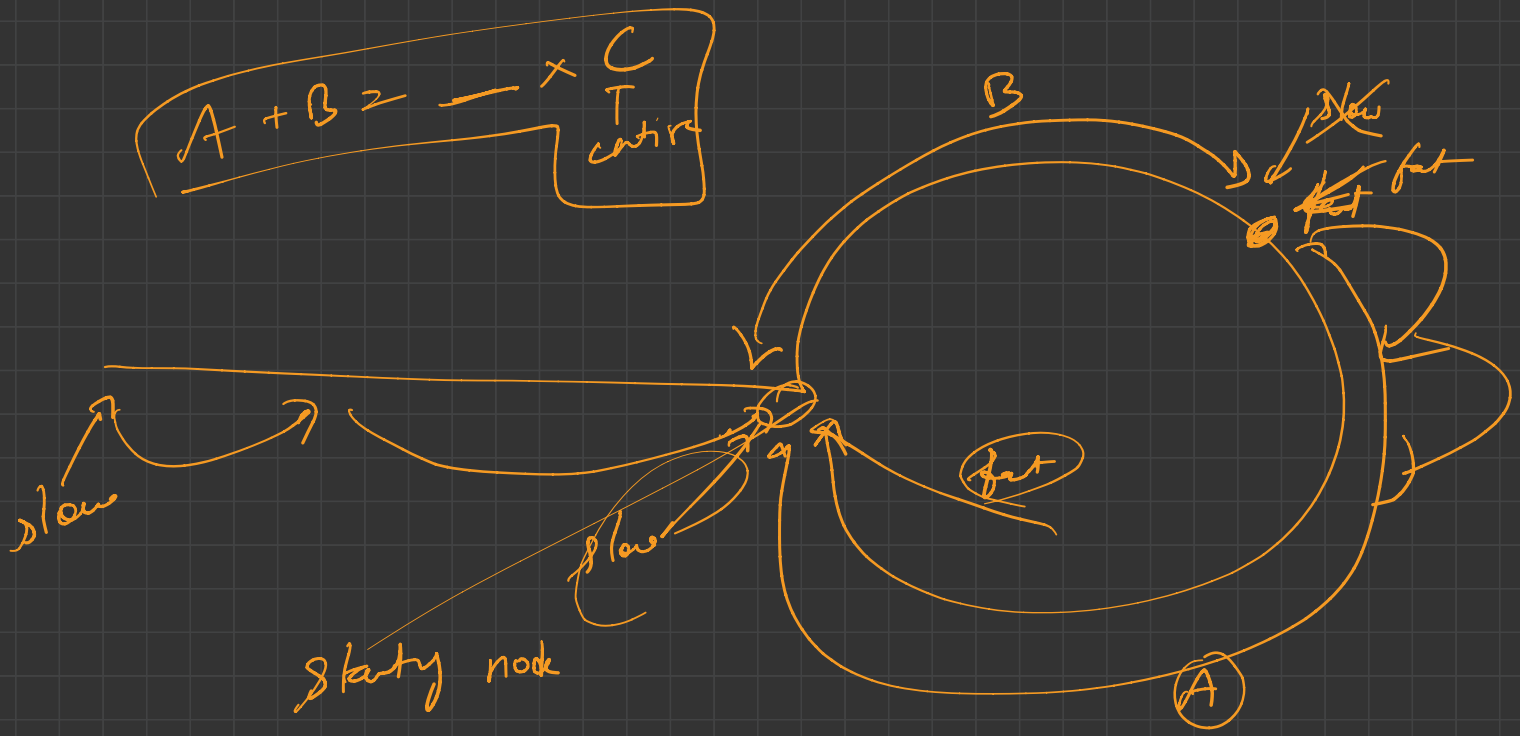
$$C(x - 2y) = A + B$$

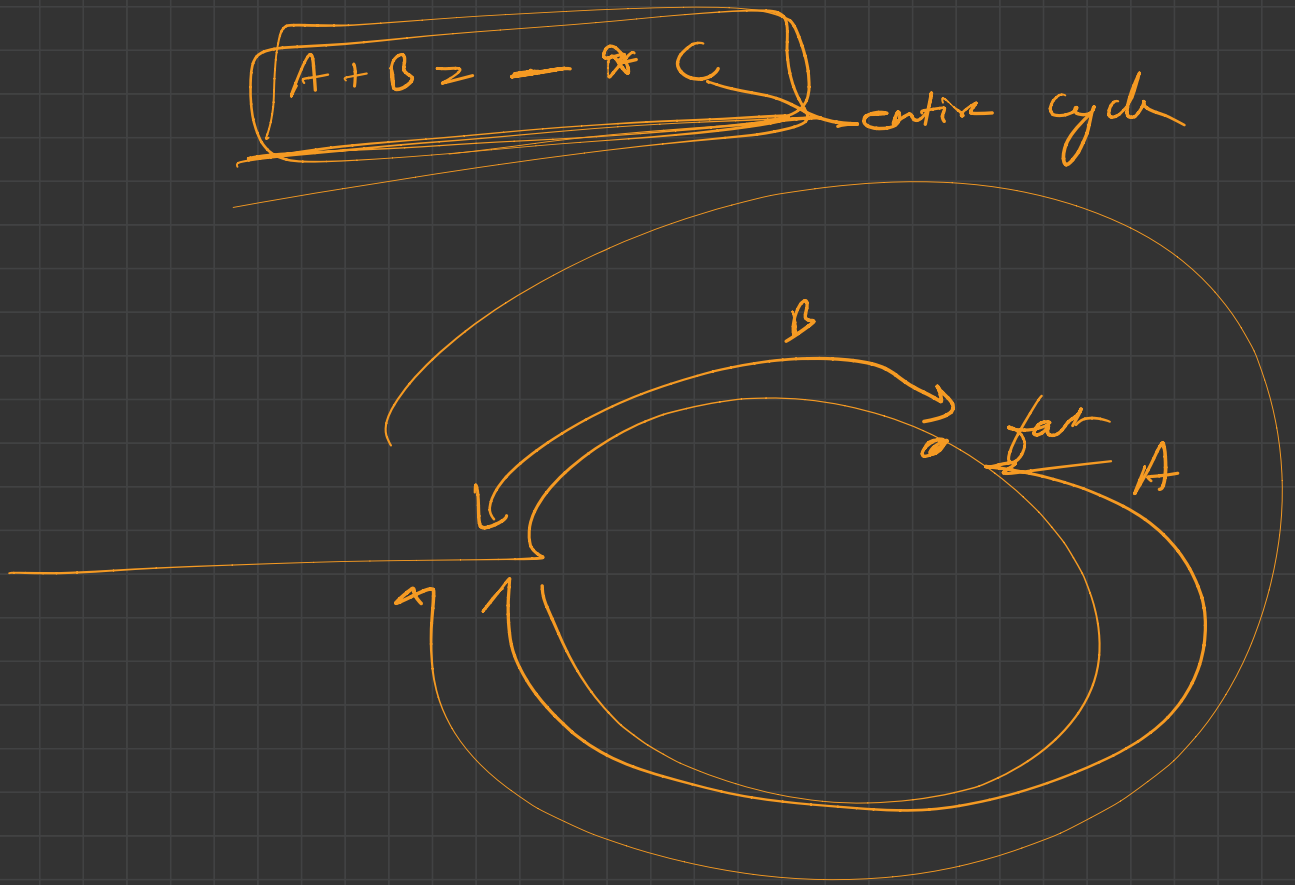
$$\underbrace{\quad}_{K}$$

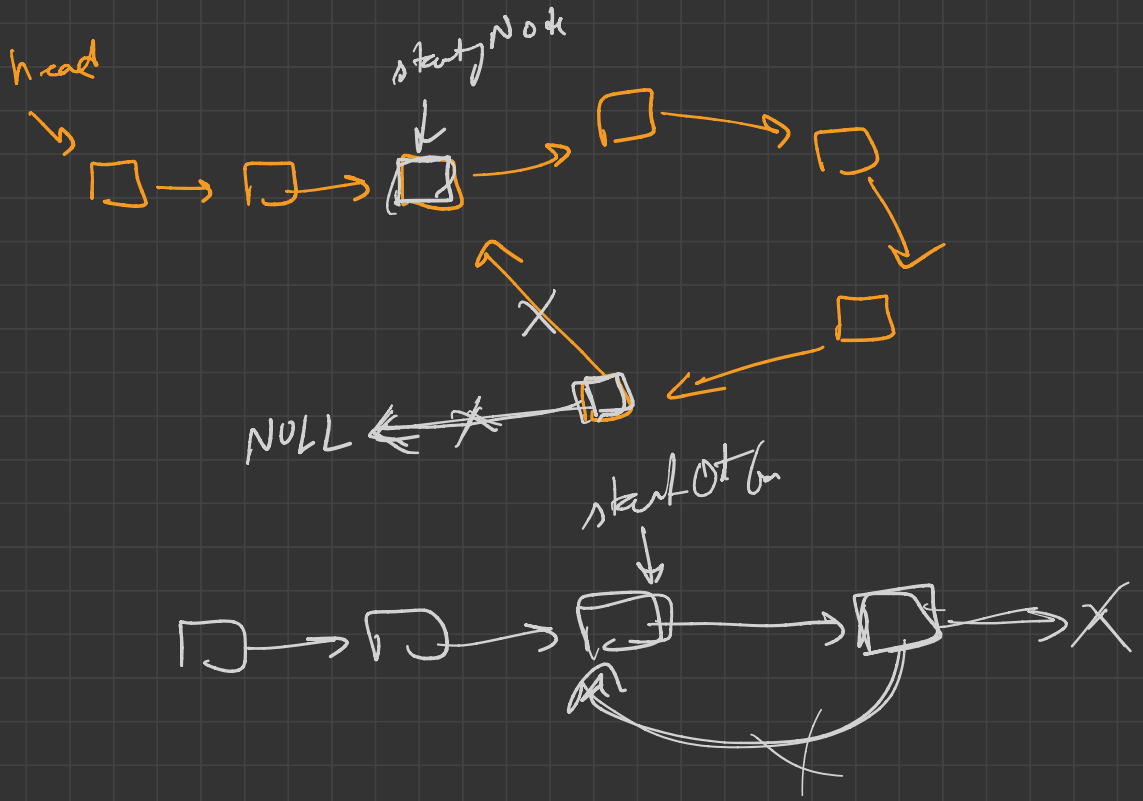
$A + B = K$ times C

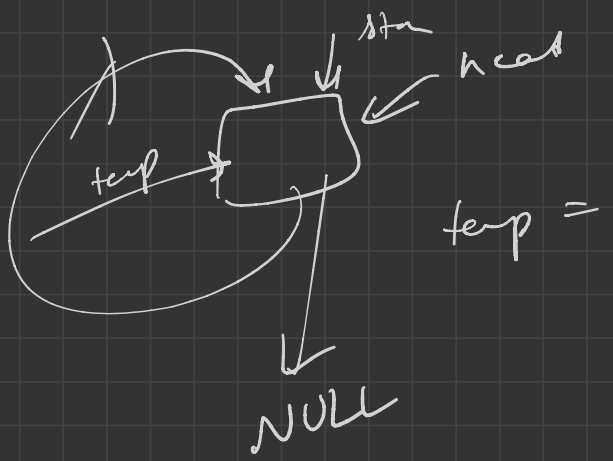
⇒ A + B ⟶ Cycle complete Kardi

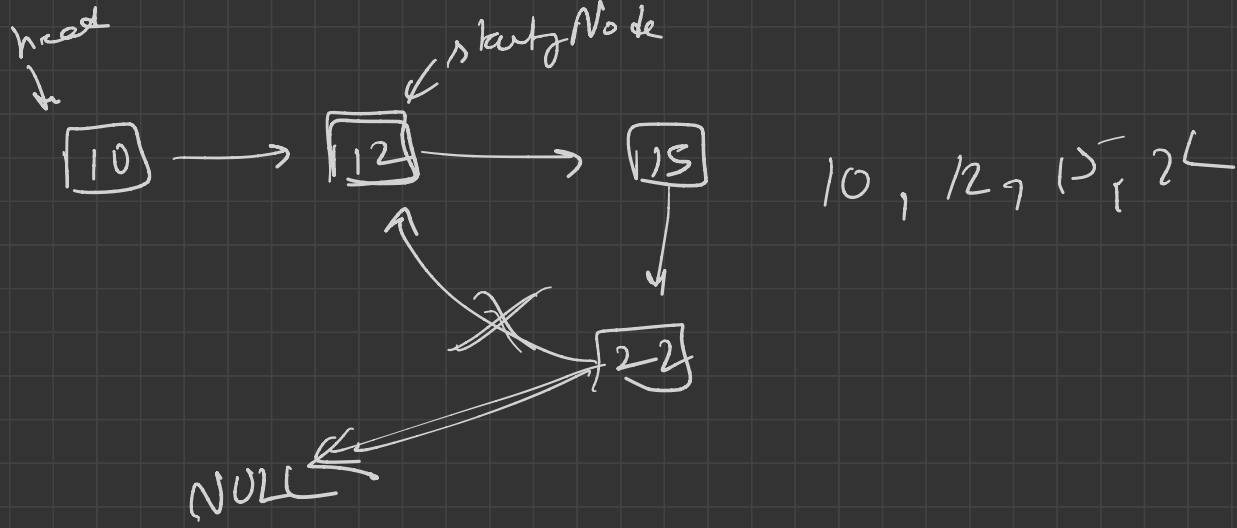



A + B =
C
T
entire
B
slow
slow
fast
fast
Starting node
A


A + B ⇄ C
entire cycle
B
fan
A


head
startNode
NULL
startOfLoop

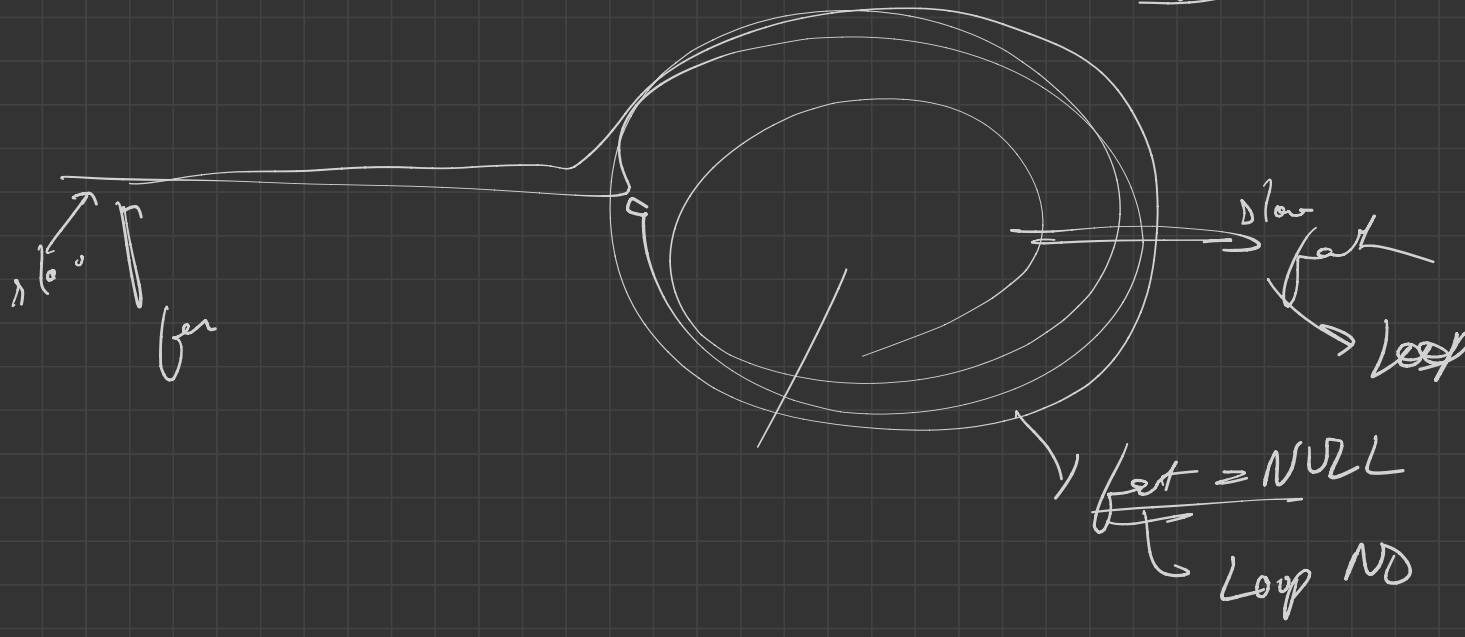

start
head
temp
temp =
NULL


head
startingNode
10
12
15
22
NULL
10, 12, 15, 22

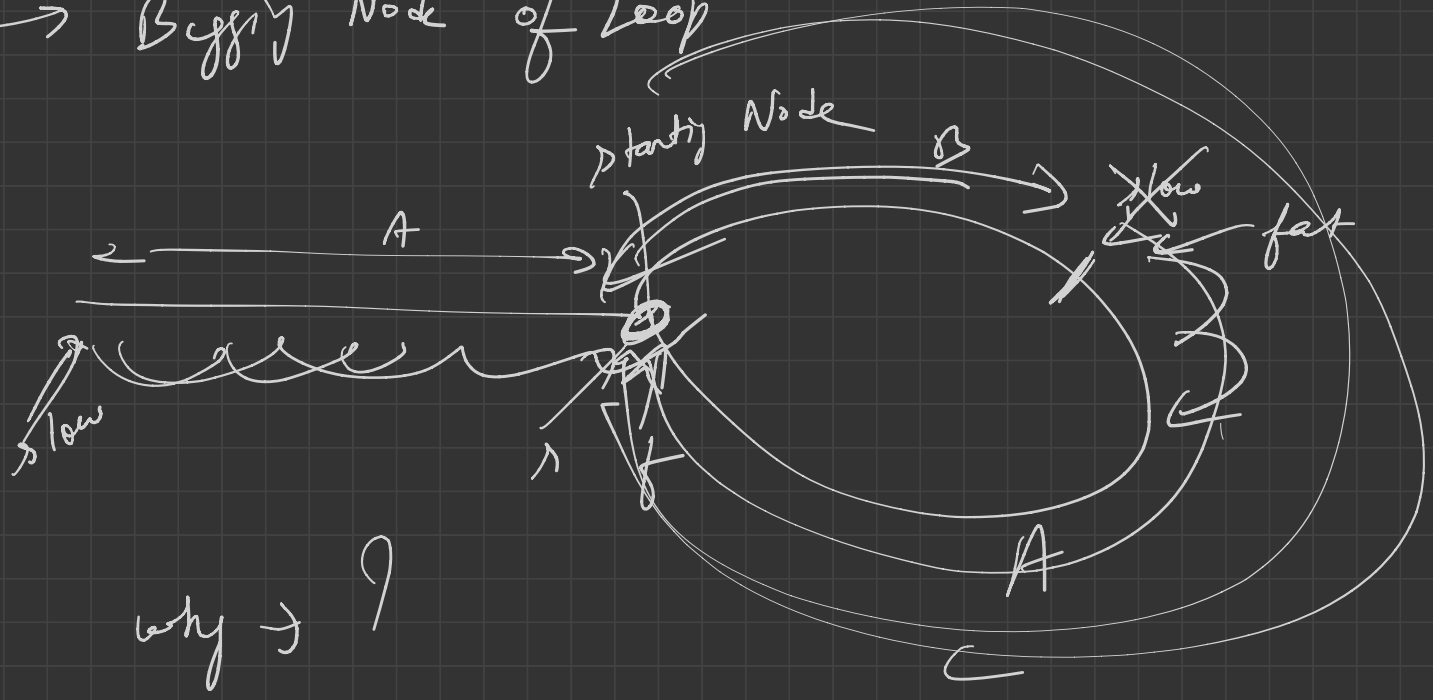
→ Detect Cycle → FCD

↳ map → O(n)
O(n)

↳ FCD → slow
fast

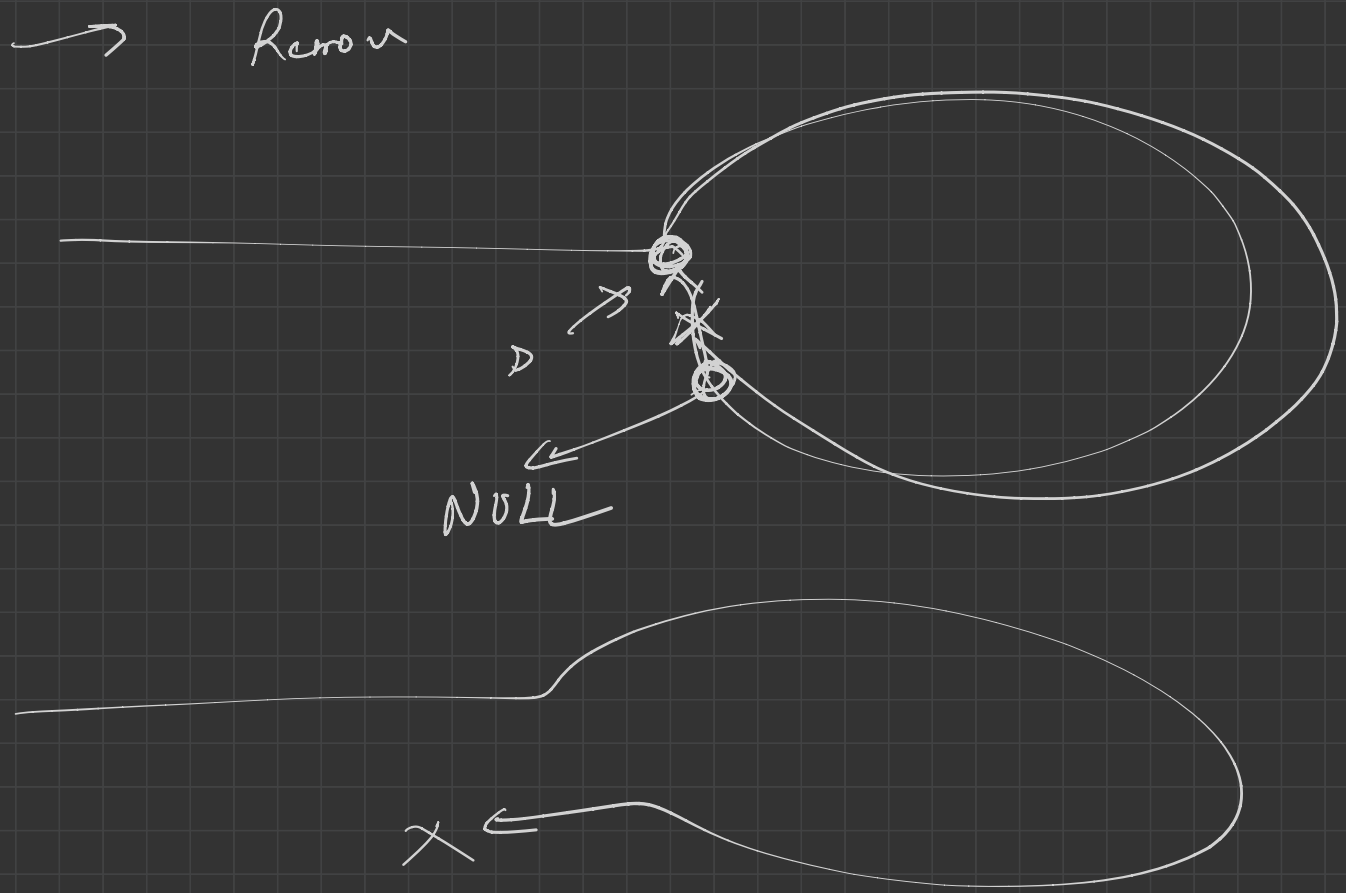


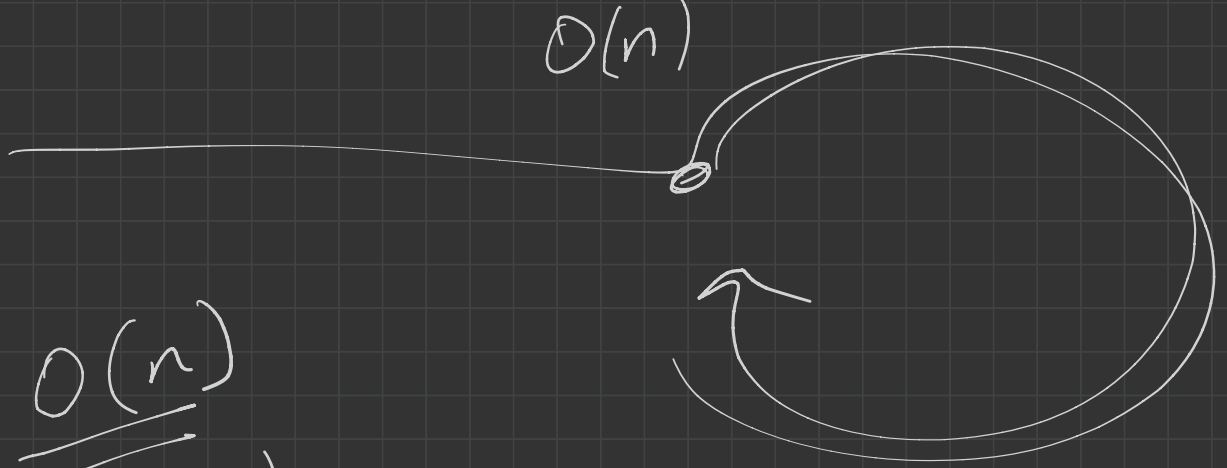
→ Buggy Node of Loop



why → ?

$$A + B = K \times C$$


Return
D
NULL
X



O(n)

O(1)

H/w

Remove → 5 ways

loop

Detect cycle

3 ways