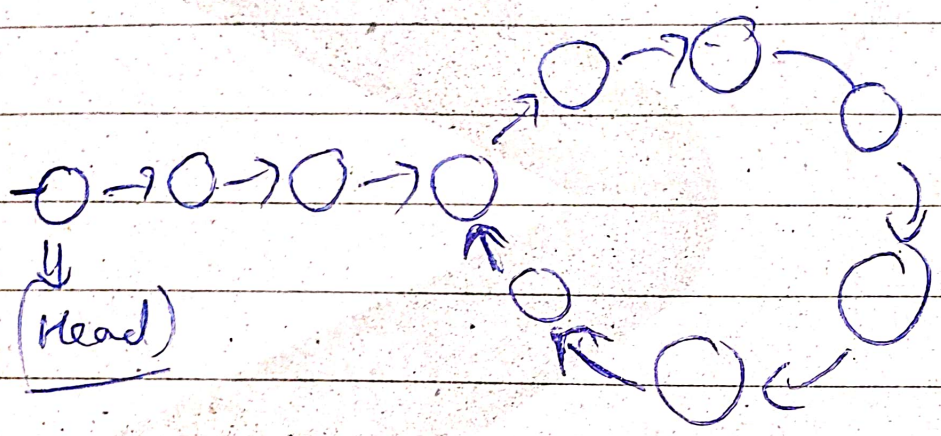
(Finding a cycle in the LinkedList)

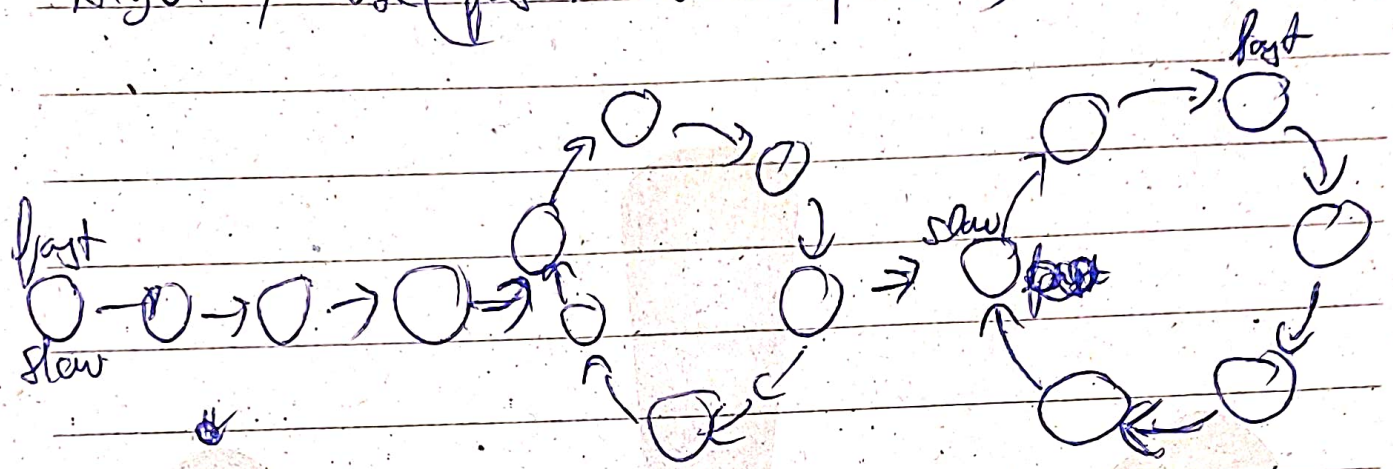


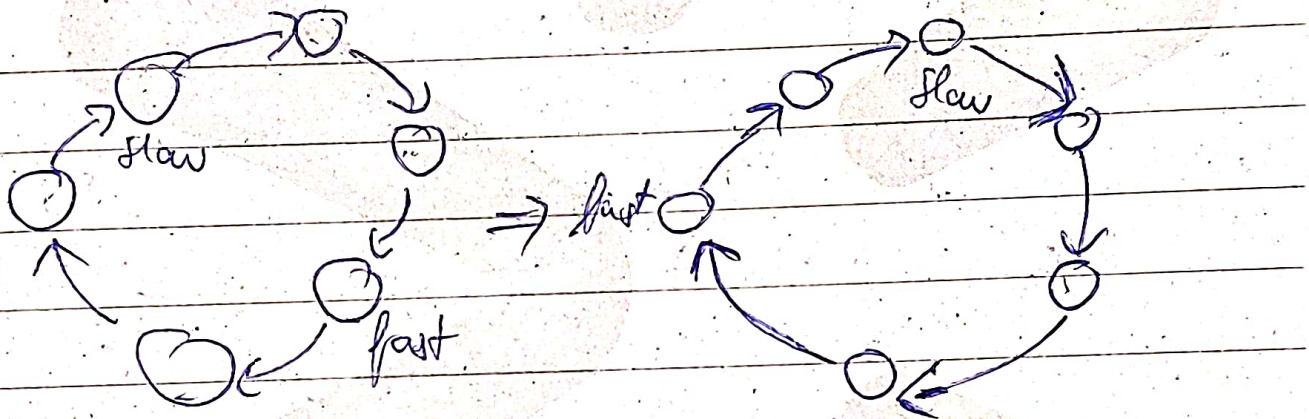
Naive approach is to start at the head pointer and maintain a HashMap to a (node → visited) map.

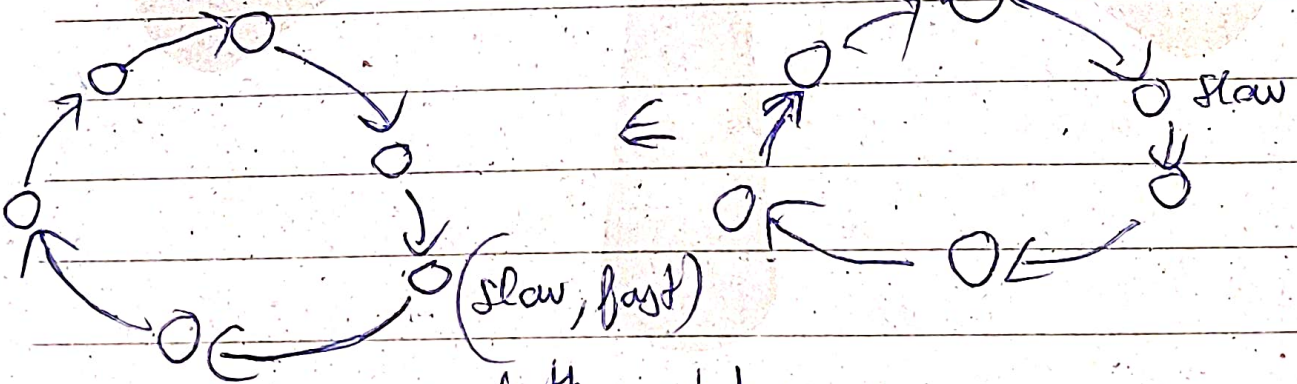
While travelling through the nodes check if its in the Hashmap, if a node is getting visited twice, that means a cycle exists.

Without using extra space

Algo, Use (fast & slow pointers)





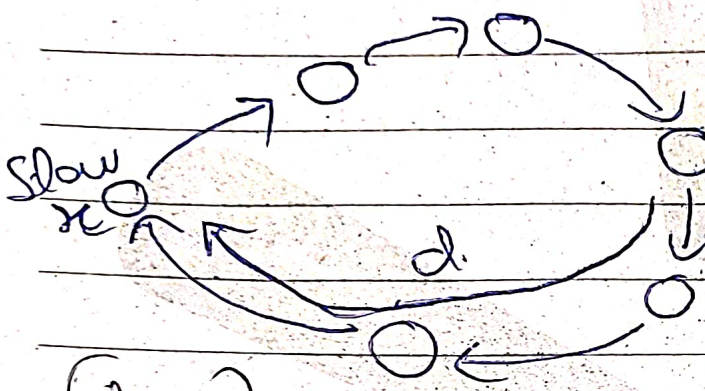


both meet here.

If we use fast and slow pointer, they will always meet, what is the intuition behind it?

- Why will they always meet

[Let's say slow is at x, and fast is at y,]



At the current moment the distance b/w them is d.

(d=3)

Both are moving in clockwise direction,

if fast is moving towards slow with a speed of 2,
if slow is moving away from fast at a speed of 1.

Relative speed = (2-1) = 1
Current distance b/w them = d.
Then after (d/1) moves they are bound to meet

Thats the intuition.