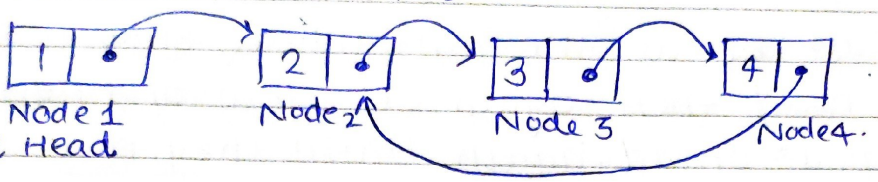
Linked List

* Detect A Cycle in a LinkedList and a Node From where the Linked List's cycle started:

Example :-

Consider a Linked List :-

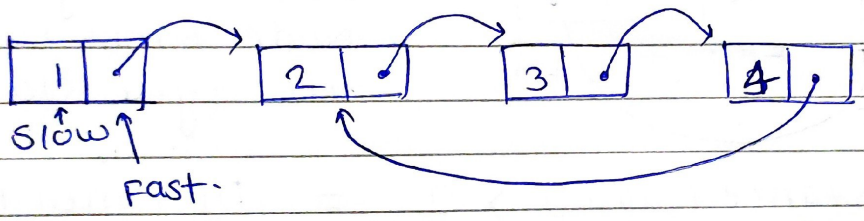


$O(n)$ - time complexity of Floyd's Algo

Here the cycle is started at Node 2

Approach to find the Cycle and the Node (Floyd's Algorithm).

① Take two pointers i.e Nodes (say: Slow, Fast). initialize them to head and then let them traverse in a way such that slow moves one node ahead whereas Fast moves two node ahead.

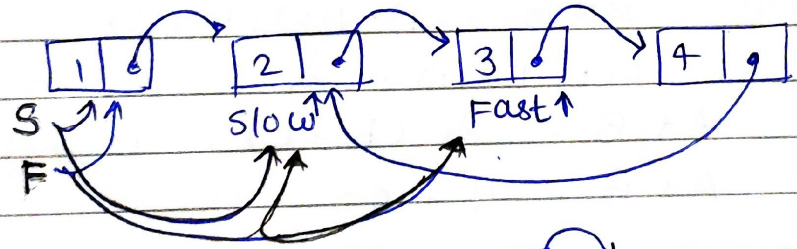


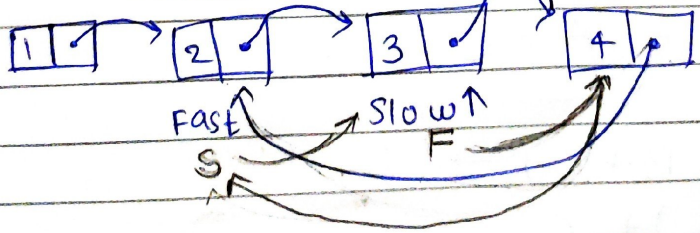
Slow = Slow.next Fast = Fast.next.next.

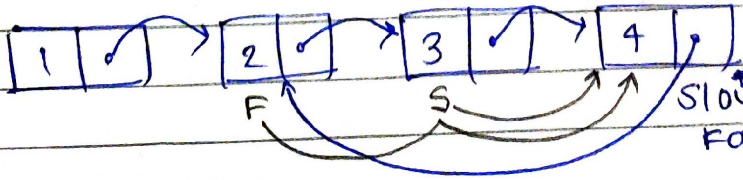
IF cycle EXist it is For the sure that Slow & Fast will meet at one point.

S - initial Place of Slow

F - initial place of Fast.





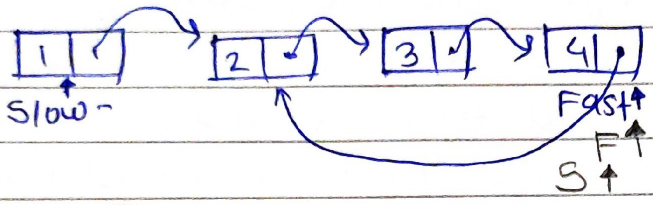


Now Here is the condition Slow == Fast

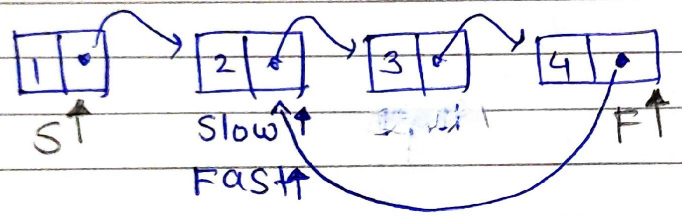
Proof of

Now, AS we detected the loop now it's forsure that we need to detect the Node From where the Cycle started.

So, Let the Fast be at the Point where Slow & Fast meet and set slow = head then again Let slow and Fast traverse one node ahead again Untill they meet eachother at a point i.e slow = slow.next ; Fast = Fast.next;

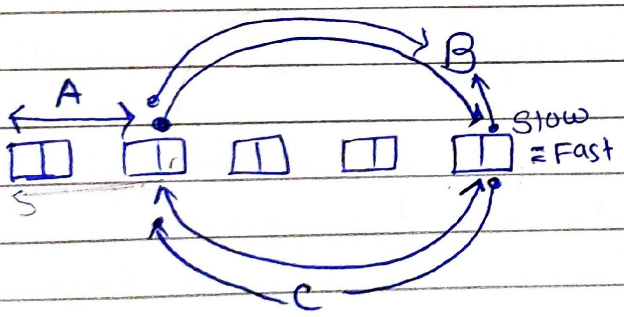


S, F deriotes the Past Place of Slow & Fast



// So here we got the Node From where the cycle started.

Proof of : Distance (Slow) x 2 = Distance(Fast)



$A + (m(B+C)) + B = A + n(B+C) + B$

$A + B = (n-m)(B+C)$.

Let $n-m = \lambda$.

$A + B = \lambda (B+C)$ if $\lambda = 1$;

$A + B = B + C$.

i.e $A = C$

if $\lambda = 2$

$A + B = 2(B+C)$

$\therefore A = (B+C) + C$.

Slow = Fast