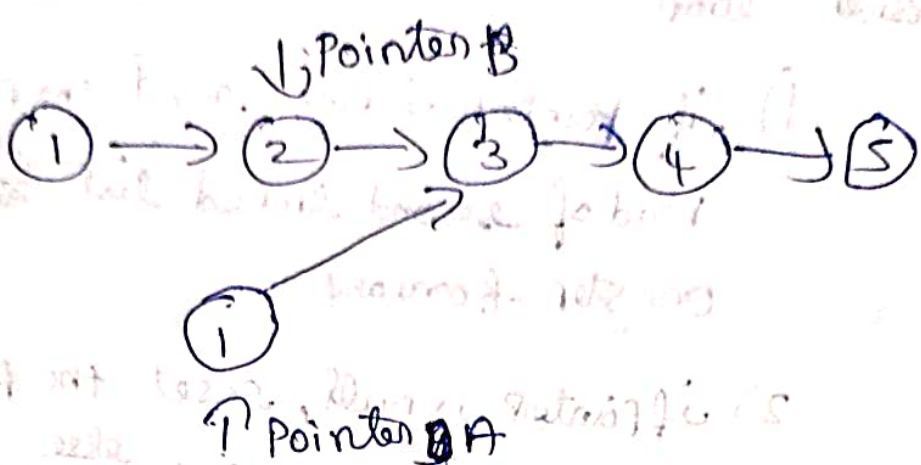
Now Pointer A will reach the end

So reset it head of B




- Now both pointer have same length to traverse. So they will reach intersection point at the same time. we will break through the loop

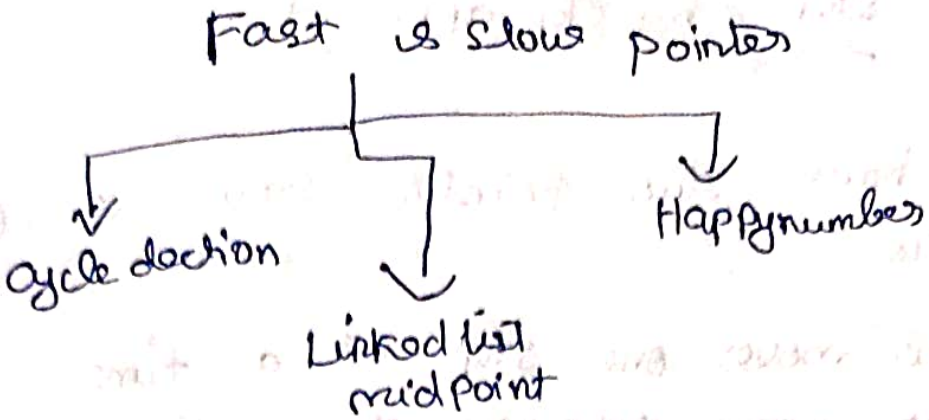
## Least recently used Cache

### Fast is slow pointer

- here we have two pointer in which one moves at 2 steps at a time is other moves at a speed of one step at a time.

usually we will have like this

1) slow pointer move one step at a time
2) fast pointer moves two step at a time




## Linked List cycle

Given a linked list we need to detect a cycle in a given linked list

To Solve this problem we can use hash set (or) fast & slow pointer tachique

In hash set solution. we need iterate over Linked list. store it in a hash set. for each element in hashset if same nodes is again visted return it as cycle detected else no cycles found

This solution need extra space O(n). we can solve this problem by O(1) space complexity

# Step by step approach

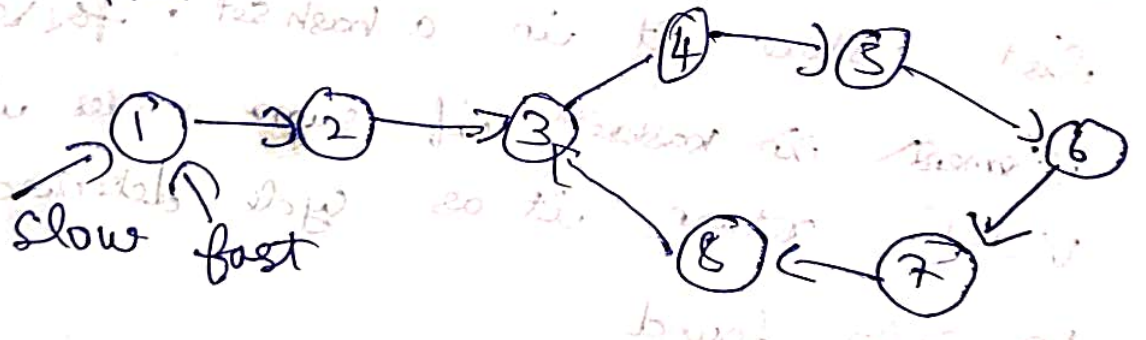
1) we have two pointer slow & fast pointer

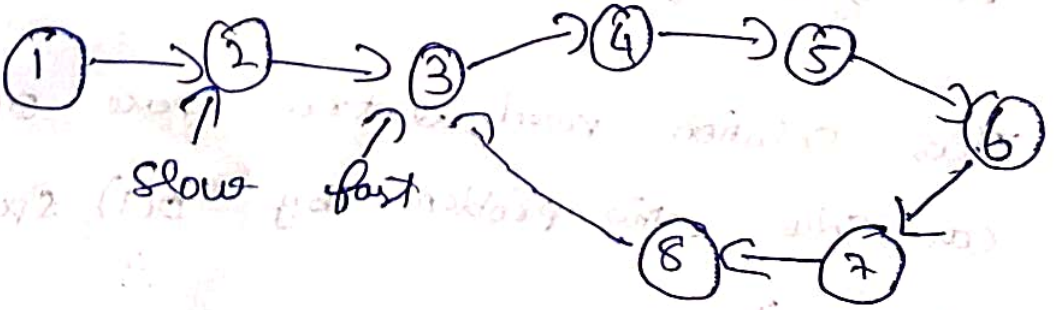
2) slow moves one step at a time
Fast moves two step at a time

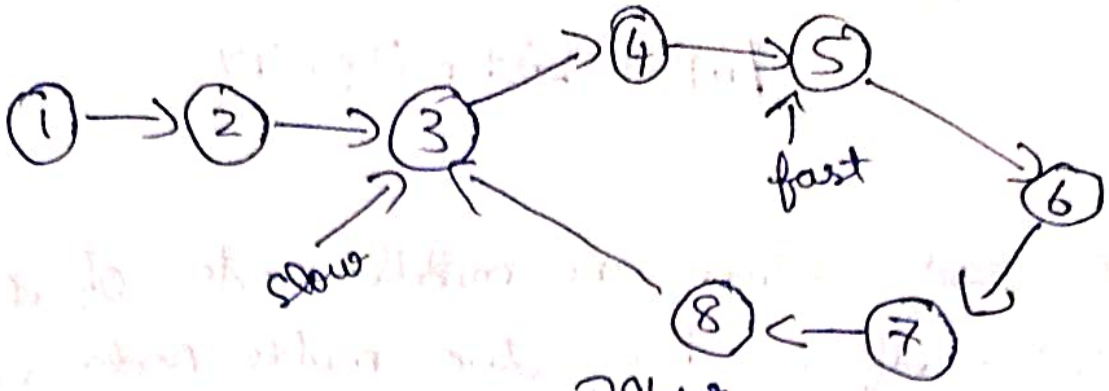
3) if linked list has no cycle then fast pointer will reach the end of linked list since its is moving at a speed of 2

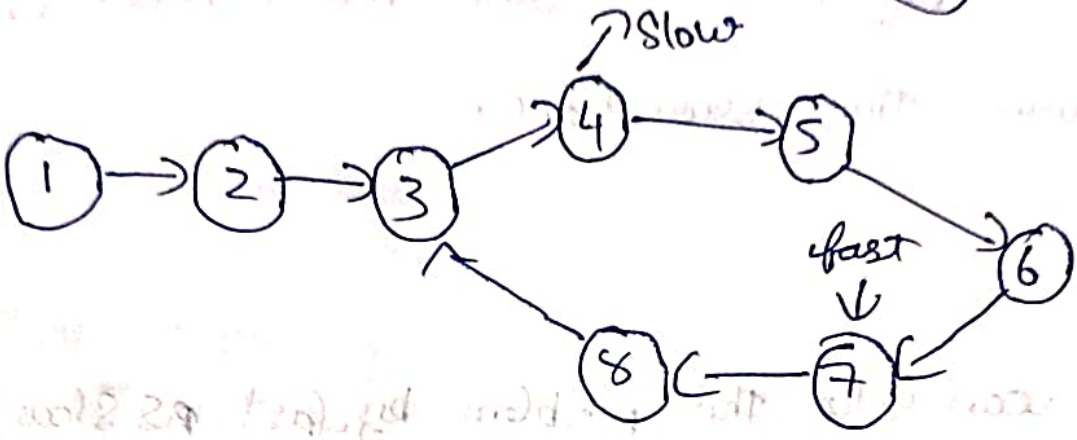
4) if linked list has cycle then there is some point at which fast pointer meet slow pointer again inside the cycle

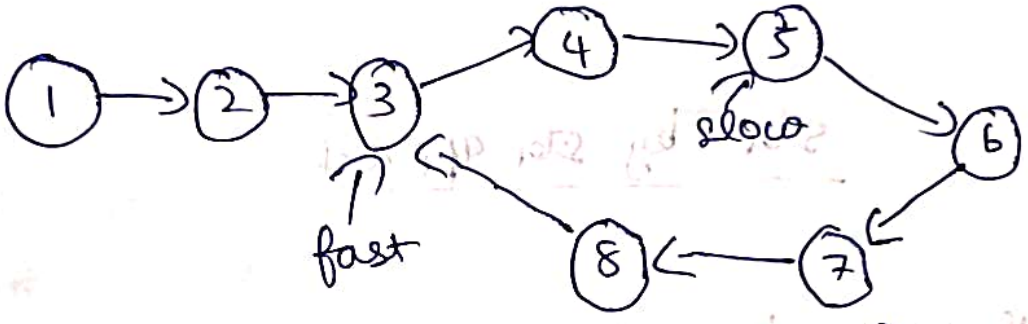
5) once after they meet return as true

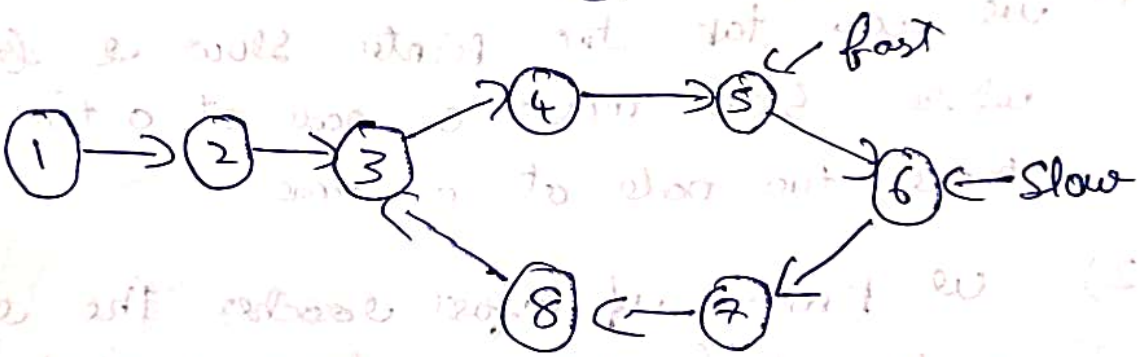


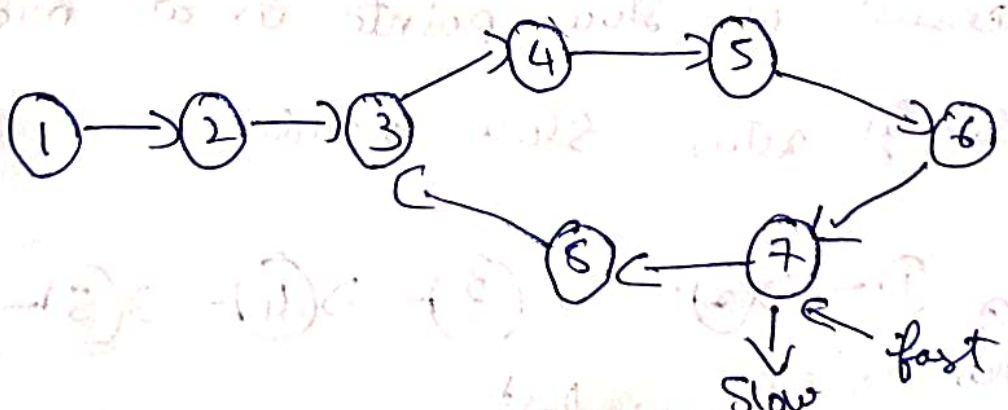






we have found point at which slow & fast met so return as true

# Linked List midpoint

We need return the middle node of linked list. If we have two middle node we need return the second one.

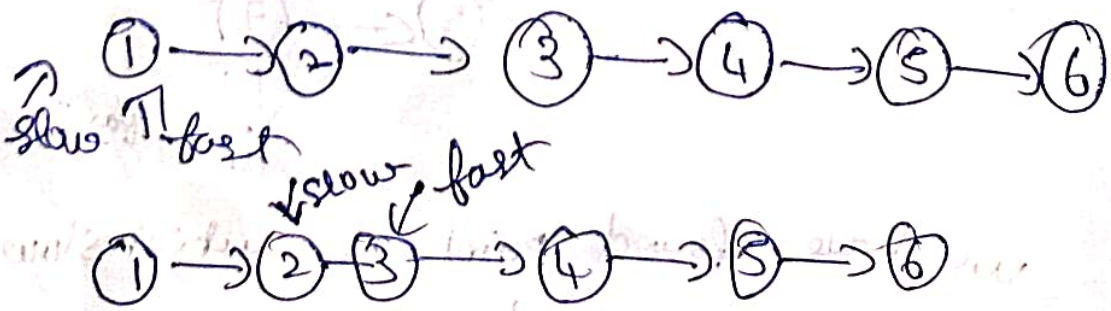
We can solve this problem by fast & slow pointer

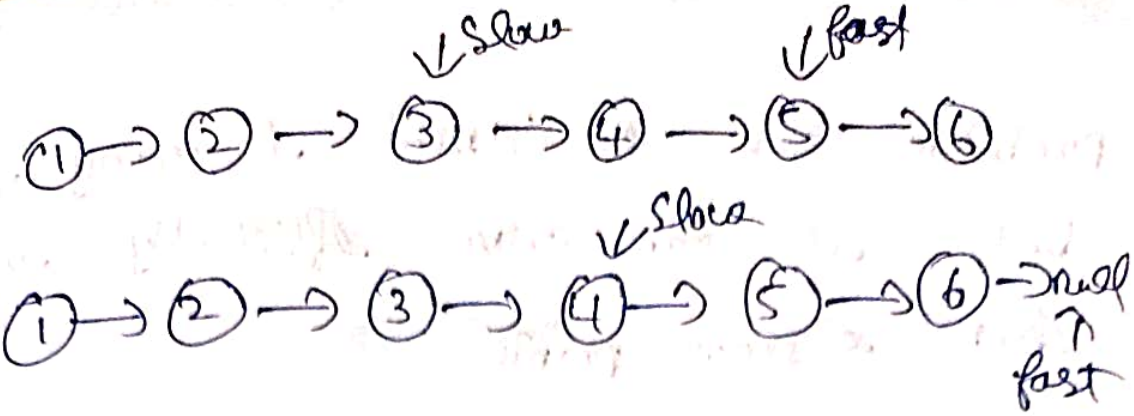
## Step by Step approach

1) We can take two pointer slow & fast where slow moves one node at a time. Fast moves two node at a time

2) We know if fast reaches the end of node which means we have reached the answer i.e slow pointer is at middle node

3) Simply return slow pointer






now return slow pointer

Time complexity → O(n)

## Happy numbers

we need find a number is happy number or not

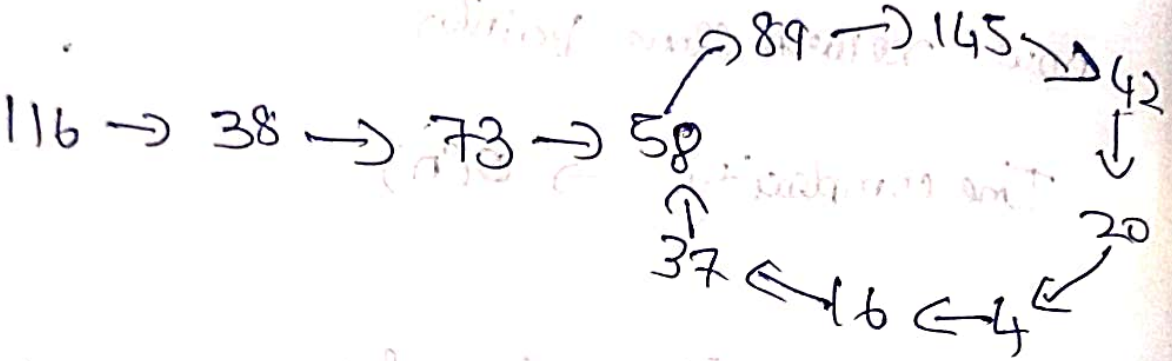
$23 \Rightarrow 2^2 + 3^2 \Rightarrow 13 \Rightarrow 1^2 + 3^2 \Rightarrow 10$

$1^2 + 0^2 \Rightarrow 1$

if we Square each digit and sum those Square eventually it lead to 1 Then it Happy number else unhappy number

Problem can be solved in many way but we can solve efficiently using fast & slow pointer

23 → 13 → 10 → 1




Step by step approach

1) we need split a number and square each digit. Then add square together to get the sum

2) if it lead to sum of one then it happy number if same number repeated again then we are in loop which mean unhappy number

3) again we are going to take two pointer where slow get one sum at a time. whereas

fast get two sum at a time.

4) if fast meets slow then there is a loop so return as unhappy number.

5) if fast == 1 then return as happy number

## Sliding window

Sliding window is technique which is subset of two pointers. window is a sub-component in a data structure where slides in array to search a particular sub component (sub array or substring)

In sliding window if we need to expand the window we have to move right pointer