	0	1	2	3	4
0	2	3	8	6	7
1	1	2	3	0	4
2	5	0	6	7	8
3	9	10	11	12	13

now we have two hashset

rowHashset = { }
col Hashset = { }

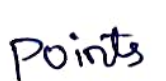
after loop the ~~the~~ matrix we will have like this

rowHashset = { 1, 3 }
col Hashset = {1, 2 }

Then loop the matrix again make the elements zero accordingly

Linked list

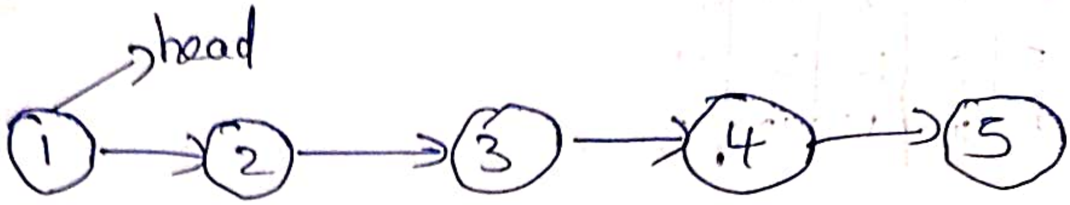
Linked list is linear data structure. It contains value is reference to other node.


reference to other node

singly linked list

each node points to other node where last node points to null

Starting node of linked list is a head.



Doubly linked list

1) upgraged version of singly linked list
2) where it has two pointer to a node

    1) prev pointer
    2) next pointer

Linked list reversal

we are given with linked list we need to reverse the linked list in-place

Step by step approach

1) we can reverse linked list by copying the linked ~~to~~ list to array. then creating new linked list in reverse order
but this ~~approach~~ approch is take additional space complexity

we can solve ~~to~~ this problem in-place.

2) take two addition pointer curNode, prevNode.

3)

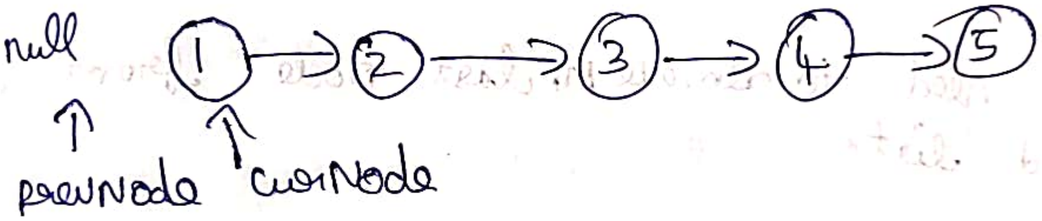
4) ~~each~~ prevNode is used to track previous Node & curNode is used to track curr Node

5) we need to loop the linked list till reaching last Node

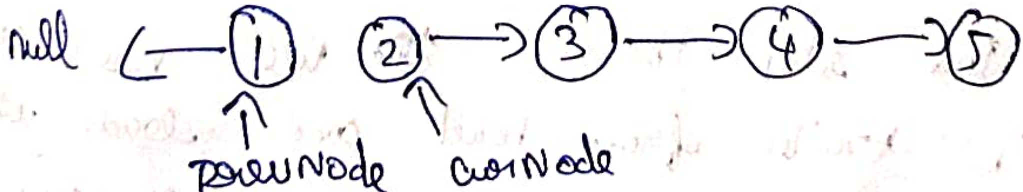
6) at first we should store next of Node in a variable lets name it as nextNode

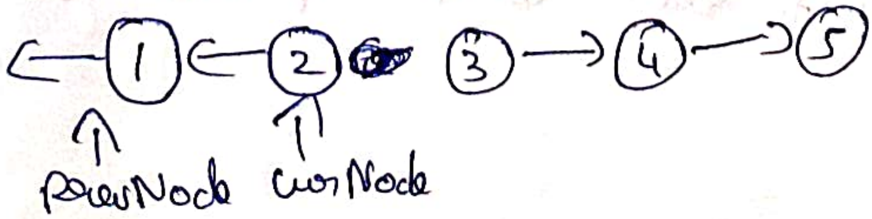
7) Then currNode next point should point to prevNode

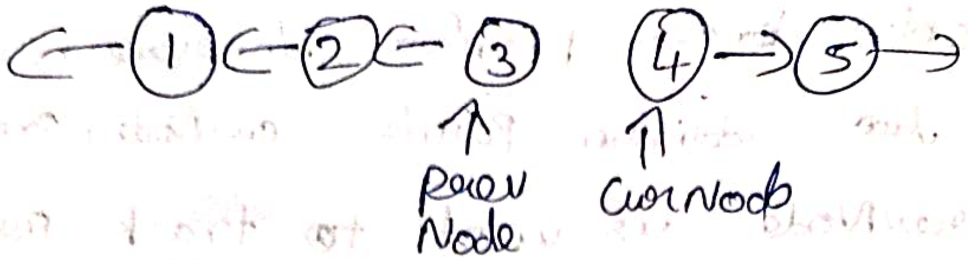
8) move forward both prevNode & curNode

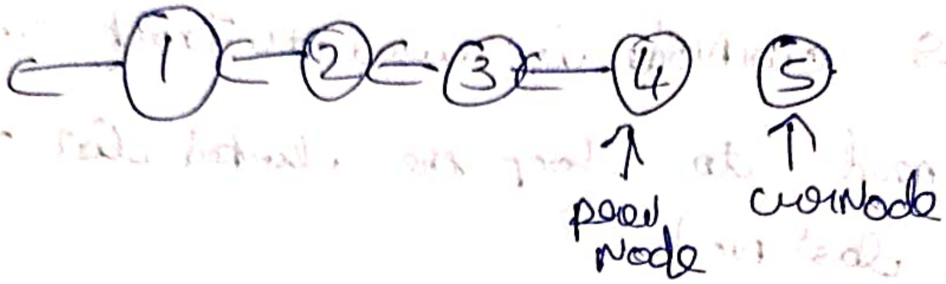


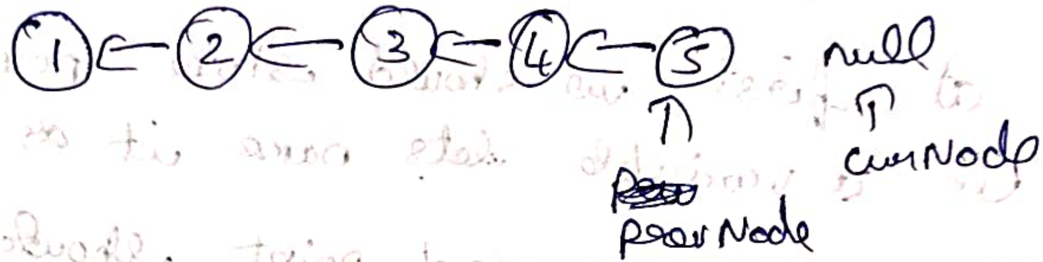
```
var nextNode = currNode.next
currNode.next = prevNode
prev Node = currNode
currNode = nextNode
```











return the prev Node as new head

Remove Kth last node from a linked list

* We need to remove Kth last node from a linked list

Step by step apporach

1) we need 2 variable named leader & trailer
2) leader move K step forward in linked list
3) then stop there. we need start the trailer from head and leader need to start at the position it stopped

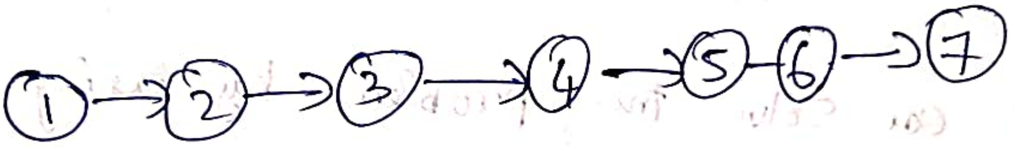
3) move both leader & trailer one position slowly

4) when leader reaches the last node. It means trailer is at position one before ~~last~~ kth last node

5) simply set next node one before node of kth last node to next node of kth last node

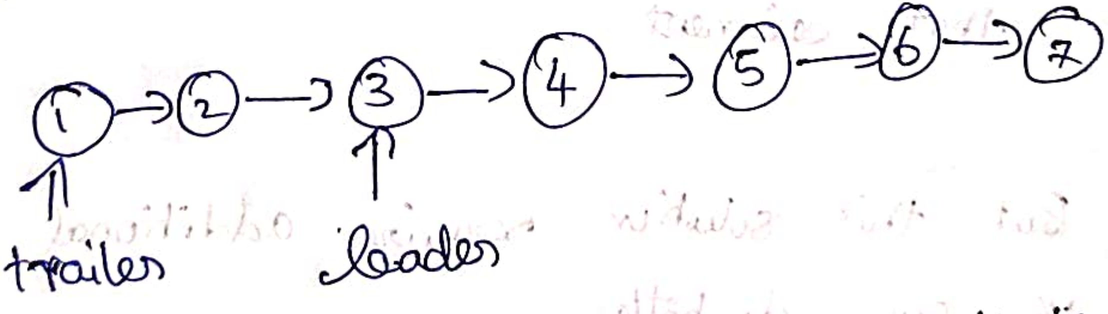
~~new~~Node.

trailer.next = trailer.next.next

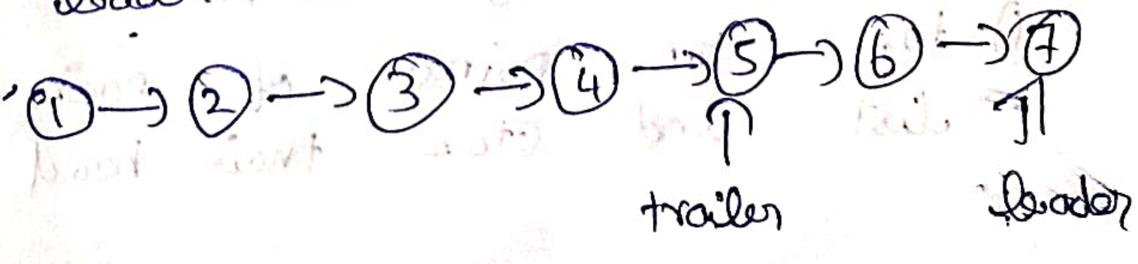


lets say k = 2.

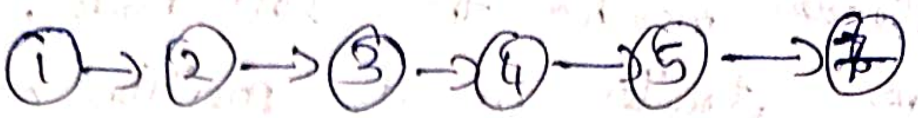
so we need to move leader 2 positon from head



move both one position forward till ~~it~~ reach the last node ~~reach~~ leader ~~the~~

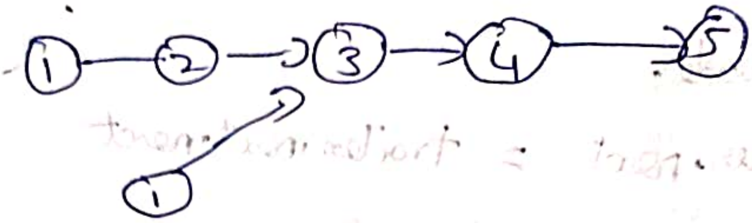


now delete the kth last node



Intersection of linked list

We need to find a intersection point of linked list for given 2 linked list



we can solve this problem by using hashset

Where we iterate over a one linked list and store each element in hashset. now iterate over other linked list. when we have first match found in hashset return that element

But this solution require additional hashset we can do better.

Step by step approch

1) take two pointer of each linked list and store their head

2) iterate loop until they are same.

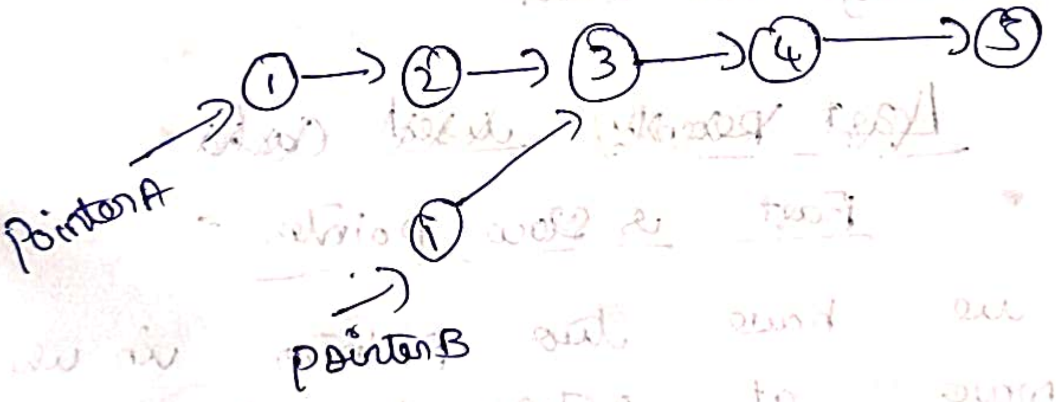
3) inside loop

1) if PointerA is null reset the pointer to head of second linked list else move one step forward

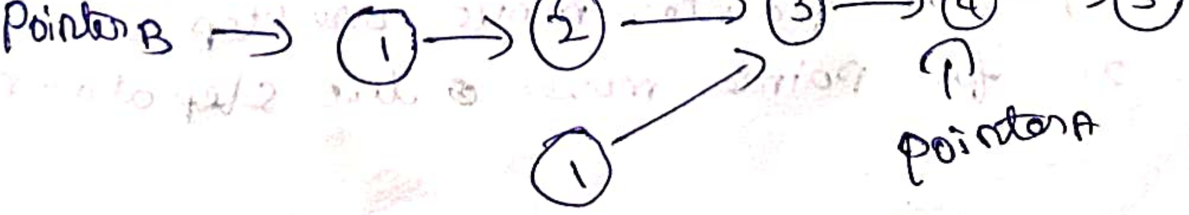
2) if PointerB is null, reset the pointer to head of first linked list else move one step forward

4) if loop break the we have intersection

5) we can return any one pointer

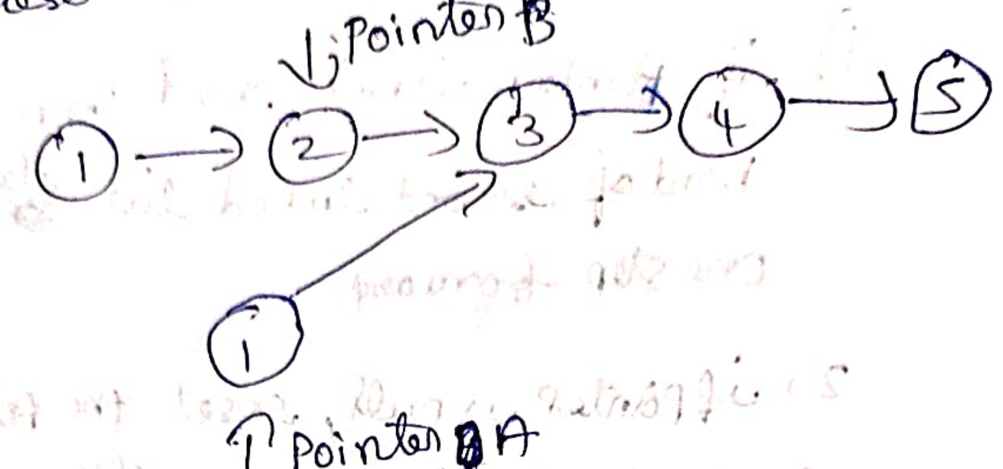


when we move foorward one by one pointer B will reach nulle so we reset it head Of A



now Pointer A will reach the

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 vs Slow pointer

. here we have two pointer in which one moves at 2 steps at a time & 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