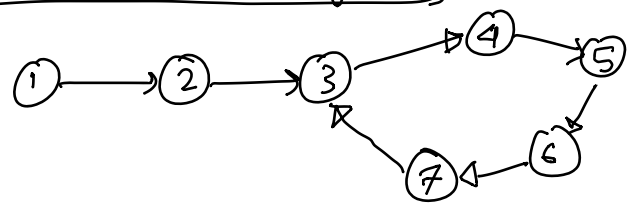
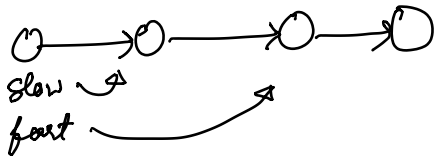
Linked List

Find Linked List Cycle

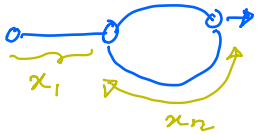
i/p :





* if there's a cycle they'll meet for sure.

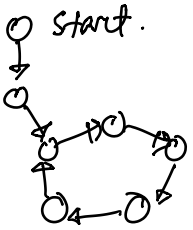
✓ To find the starting position of the cycle.



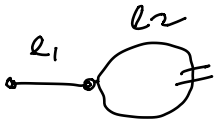
→ here slow & fast first meet.

$x_1 = x_2$

i.e. $l_1 = k - l_2$



lets assume a 5 node cycle and 2 node before that.



* at this position fast & slow meet.

$l_1 + l_2 + n_1 k$ = slow's travel.

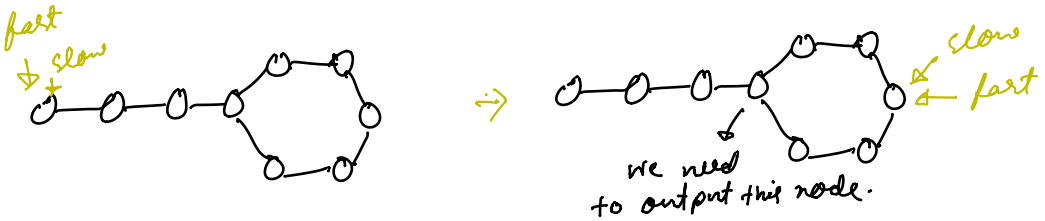
$l_1 + l_2 + n_2 k$ = fast's travel.

k → size of loop no. of nodes in loop.

$(l_1 + l_2 + n_1 k)\,2 = l_1 + l_2 + n_2 k$.

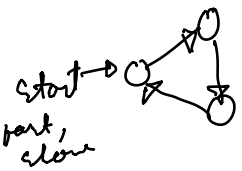
$l_1 + l_2 + 2n_1 k = n_2 k$.

if $n_1 = 0$ $\quad l_1 + l_2 = nk$.



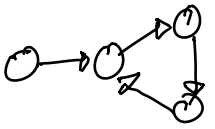
* slow & temp will meet for sure;

return temp.

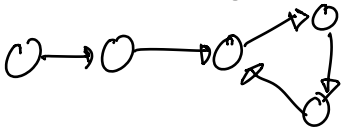


$k = 3 \quad l_1 = 0 \quad l_2 = 3$

slow 3 fast 6



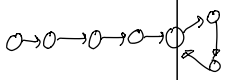
slow $1 + 2$ fast $((1+2) \times 2 - 1) \% 3$



slow $2 + k$ fast $((2+k) \times 2 - 2) \% 3$

$k=1$	3	1 ✓
$k=2$	4	$6 \% 3 = 0$ ✗
$k=3$	5	$10 - 2 \% 3 = 8 \% 3 = 2$ ✗
$k=4$	6	$12 - 2 \% 3 = 1$ ✓

$k = 1$



slow $4 + k$ fast $((4+k) \times 2 - 4) \% 3$

$k=1$ 5	$10 - 4 \% 3 = 0$ ✗
$k=2$ 6	$12 - 4 \% 3 = 2$ ✓
$k=5$ 9	$18 - 4 \% 3 = 2$ ✓

$5 + k$ $((5+k) \times 2 - 5) \% 3 = k$

$k=1$ 6	$12 - 5 \% 3 = 1$ ✓
$k=2$ 7	$14 - 5 \% 3 = 0$ ✗
$k=3$ 8	$16 - 5 \% 3 = 2$ ✗
$k=4$ 9	$18 - 5 \% 3 = 1$ ✓

$6 + k$ $((6+k) \times 2 - 6) \% 3 = k$

$\Rightarrow k = (6+k) \times 2 \% 3$

k = multiple of 3

Remove Nodes from LL :-

i/p: 5 → 3 → 13 → 3 → 8 o/p: 13 → 8

Approach :-

1) using Stack : space $O(N)$.

2) Reverse LL : 8 → 3 → 13 → 3 → 5 → 8 → 13

then return reverse of ans.

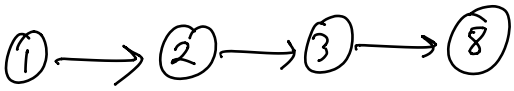
3) Simple Recursion :

5 → 3 → 13 → 3 → 8

recursive Remove (head → next) will return.

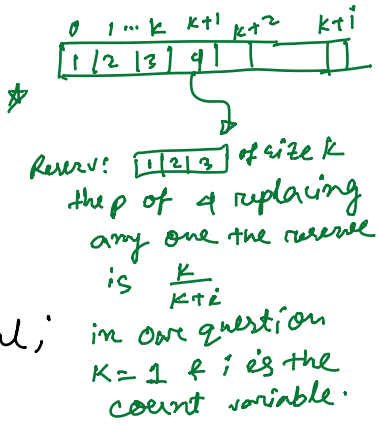
13 → 8

Reservoir Sampling :-



function getRandom() generate a random number from LL.

```
ListNode * Result ;
count = 0 ; temp = head ;
while ( temp != null ){
    if ( rand() % count < 1/count )
        result = temp -> val ;
    count ++
    temp = temp -> next.
```



#Math_algo_numberTheory/Reservoir sampling

Merge 2 Sorted List

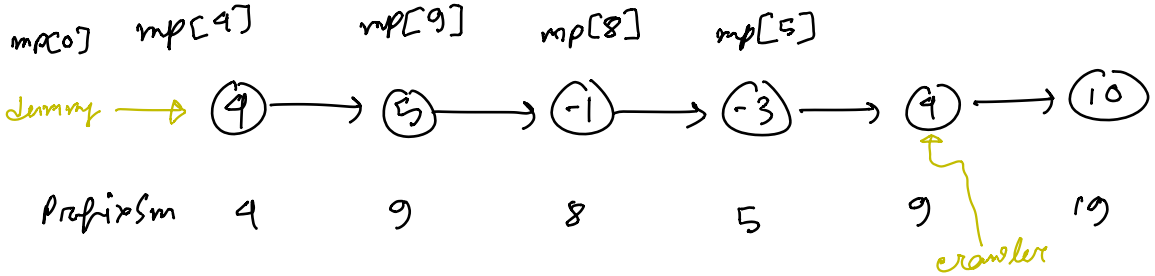
the usual merge ; using recursion -

Delete Zero Sum !:-

i/p : 4 → 5 → -1 → -3 → 4 → 10 (-1 → -3 → 4 = 0)

o/p : 4 → 5 → 10

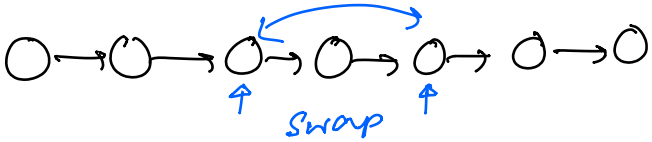
use map to store prefix sum ; unordered_map < int , ListNode * >
(int : PrefixSum ; ListNode : Node)



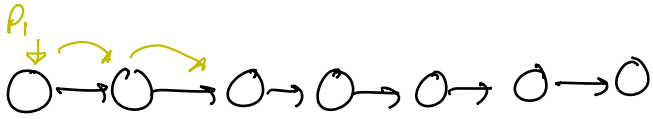

```
if ( mp.find(prefixSum) != mp.end())
    mp[prefixSum] -> next = crawler -> next.
```

Swapping K^{th} Node from the beginning & end

K = 3

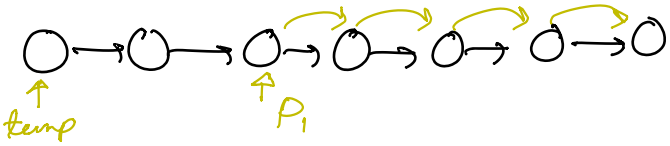


#Trick



move k-1 time

now, take a temp variable at head.



move p_1 until reaches the end,

* when p_1 reaches end temp reaches k^{th} position from end.

Sum Linked List

i/p : 7 → 8 → 2 → 4

5 → 3 → 8

o/p : 8 → 3 → 6 → 2

The sum

$$\begin{array}{r} 7\ 8\ 2\ 4 \\ +\quad 5\ 3\ 8 \\ \hline 8\ 3\ 6\ 2 \end{array}$$

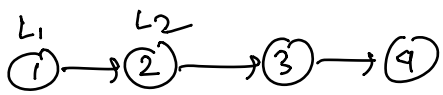
Reverse the 2 i/p LL then add & store in another temp LL (2 → 6 → 3 → 8)

then return the reverse of temp.

Swap Nodes

#algorithm/recursion

i/p: 1 → 2 → 3 → 4 o/p: 2 → 1 → 5 → 3



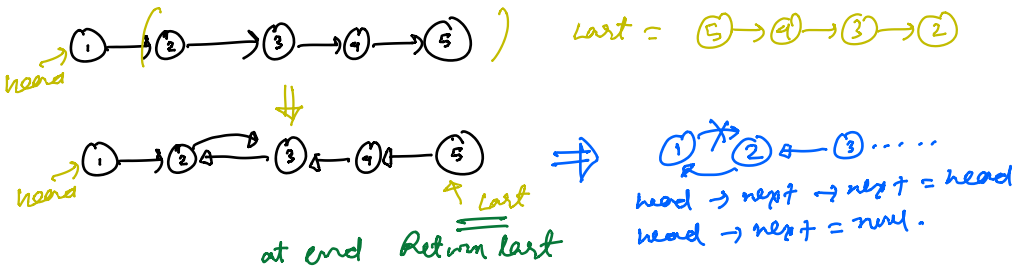
(use recursion)

#algorithm/Linked List

$L_1 \rightarrow next$, $L_2 \rightarrow next$ = solver ($L_2 \rightarrow next$) , L_1

Reverse Linked List :-

```
ListNode* reverse(ListNode* head) {
    if(!head || !head->next) return head;
    ListNode* last = reverse(head->next);
    head->next->next = head
    head->next       = NULL
    return last.
```



Reverse Linked List II

using iterative method without recursion.

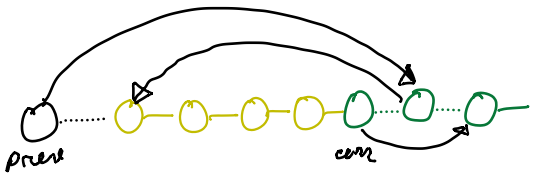
i) dummy → 1 → 2 → 3 → ... in case head is also changed, return dummy->next.

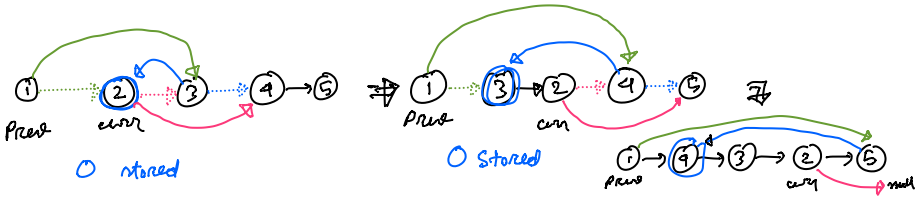
ii) assign prev & curr pointer

Prev points to the starting of reverse can be dummy as well if reversal start from head.

iii) ListNode* temp = prev->next

- prev->next = curr->next
- curr->next = curr->next->next
- prev->next->next = temp





1→5→4→3→2

#⭐, #LeetCode/Hard, #Revision