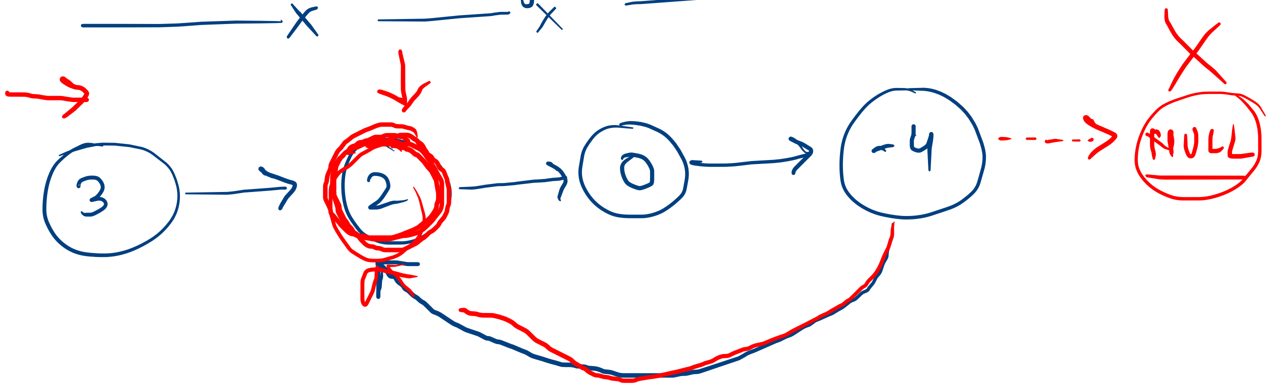
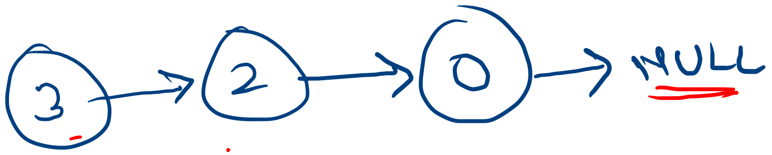
Linked List Cycle II

I/p:



O/p: 2

I/p:

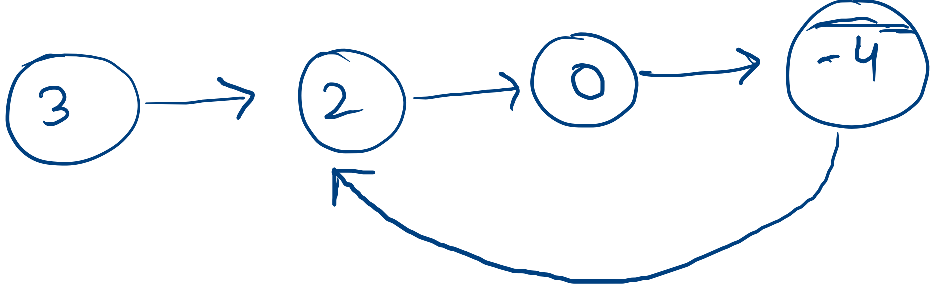


O/p: NULL

NAIVE Sol^n^ :

Basic Approach

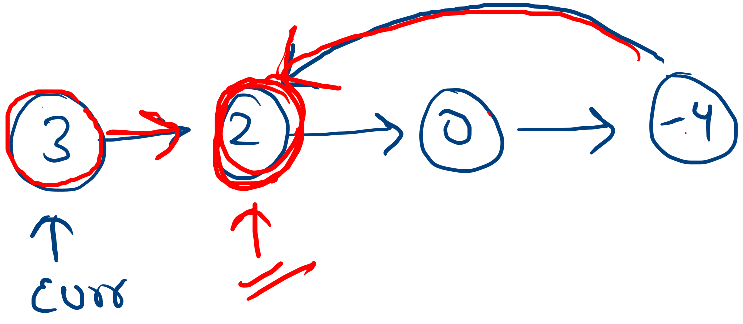
I/p :



```
unordered_set<Node*> set;

i) if curr is present
       return curr

   else
       insert curr;
```

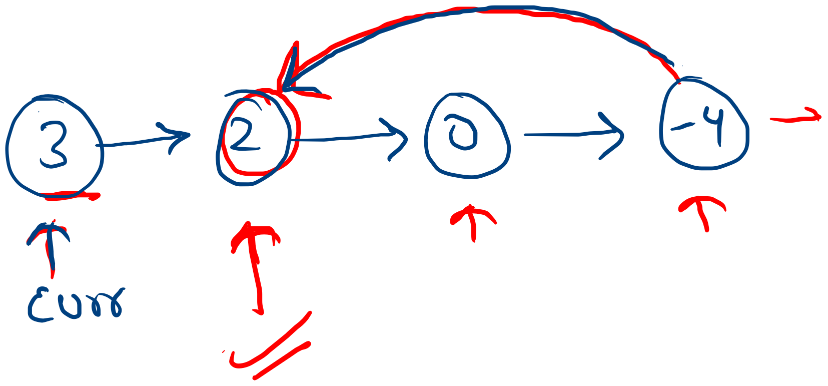


```
unordered_set <Node*> set;

i) if curr is present
       return curr

   else
       insert curr;
```

{3, 2, 0, -4}

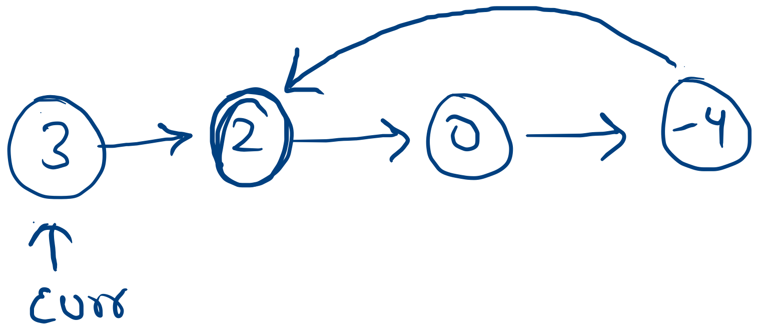


o/p:

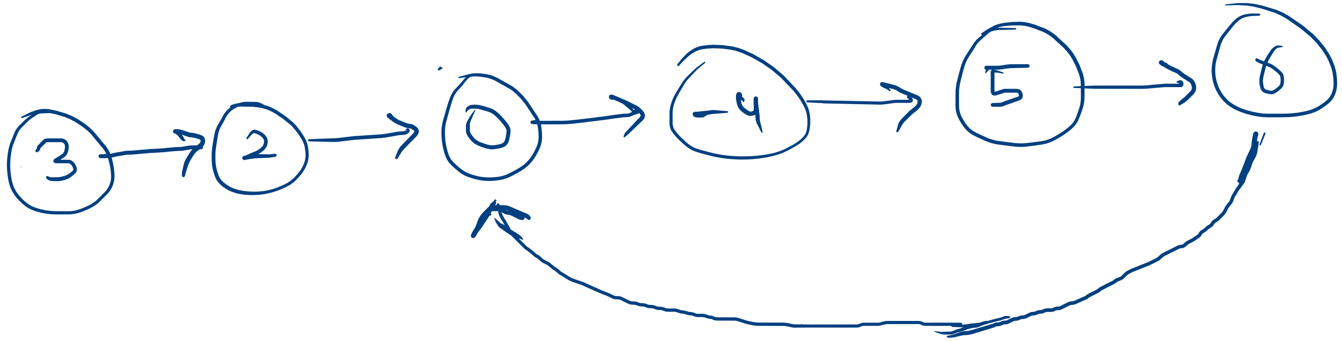
Space O(N)

Best Approach

I/p :

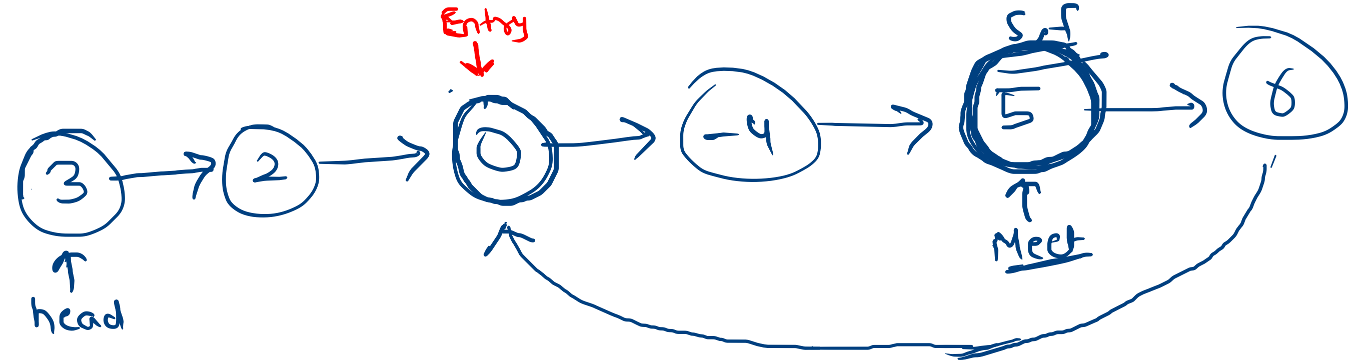


Understand Concept :

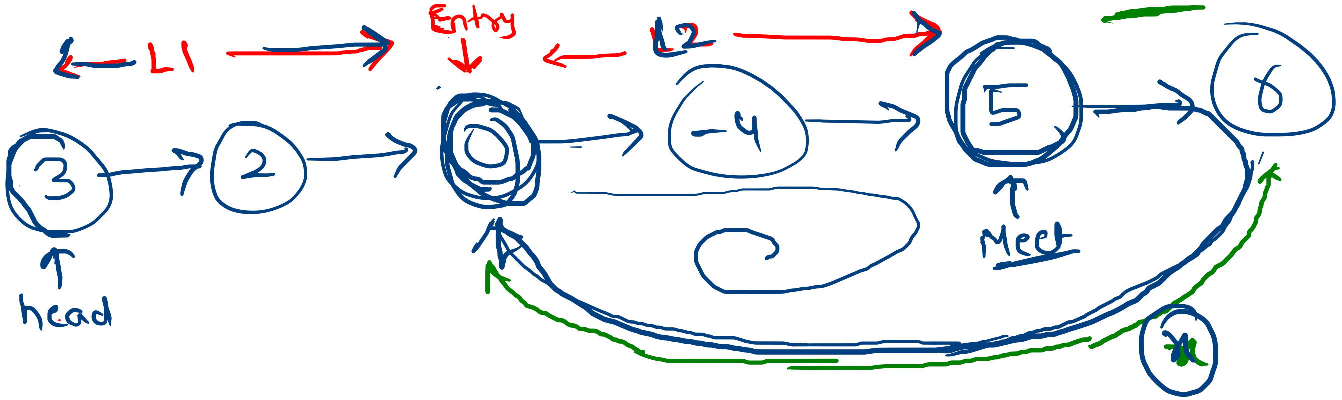


Understand

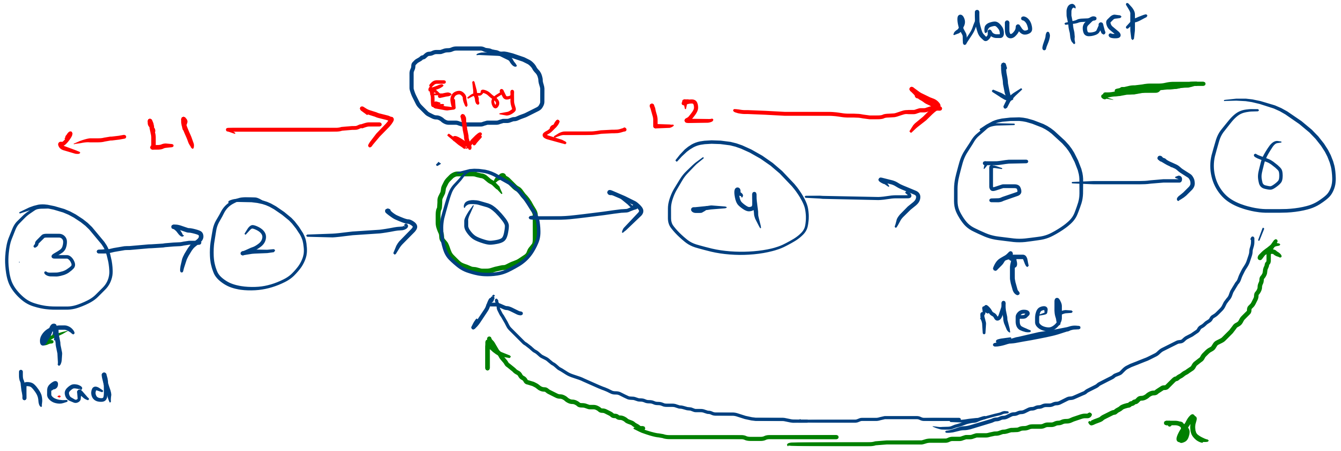
Concept:



⇒



I/p:



if we can find L1 then we can find Entry point

✓ distance travelled by slow = $\boxed{L1 + L2}$

✓ distance travelled by fast = $L1 + L2 + n*(L2+x)$

(where n is no. of cycle rotation of Linked list cycle)

$\Rightarrow$ (fast has double speed than slow)

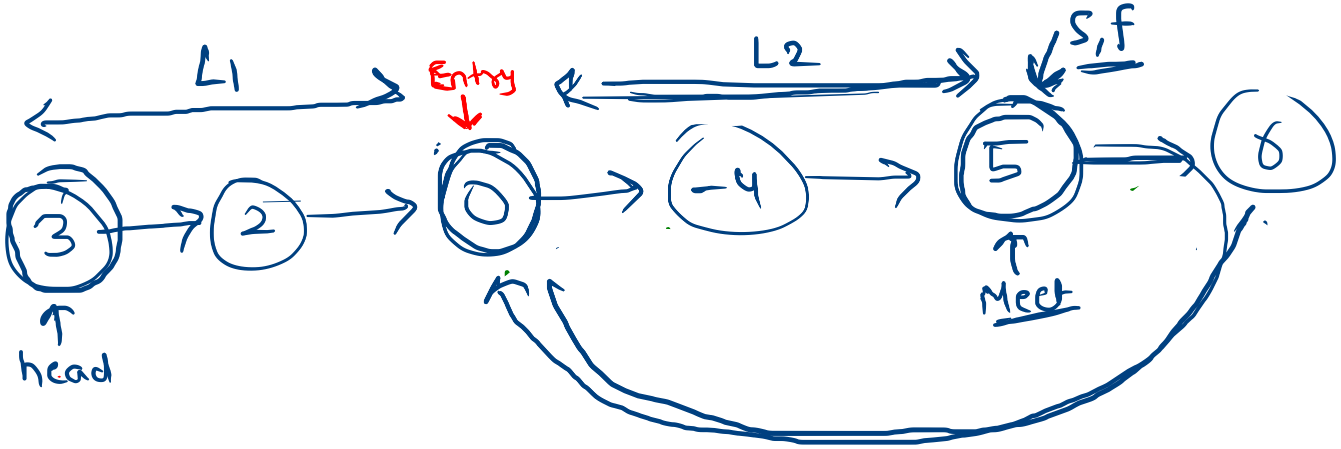
$\Rightarrow$ $2(L1 + L2) = L1 + L2 + n*(L2+x)$

$\Rightarrow$ $2L1 + 2L2 = L1 + L2 + nL2 + nx$

$\Rightarrow$ $\boxed{x = L1}$

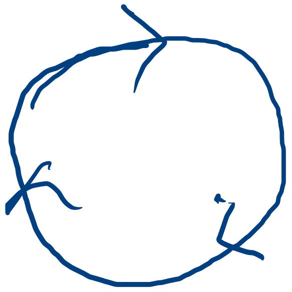
For once let's suppose $\boxed{n=1}$

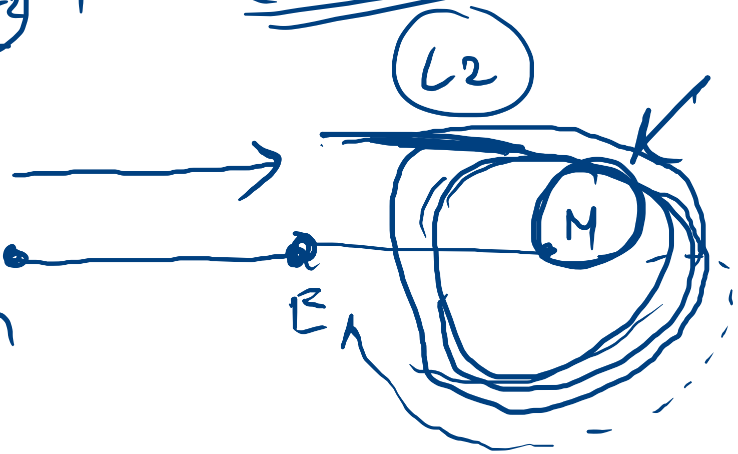
Ill:

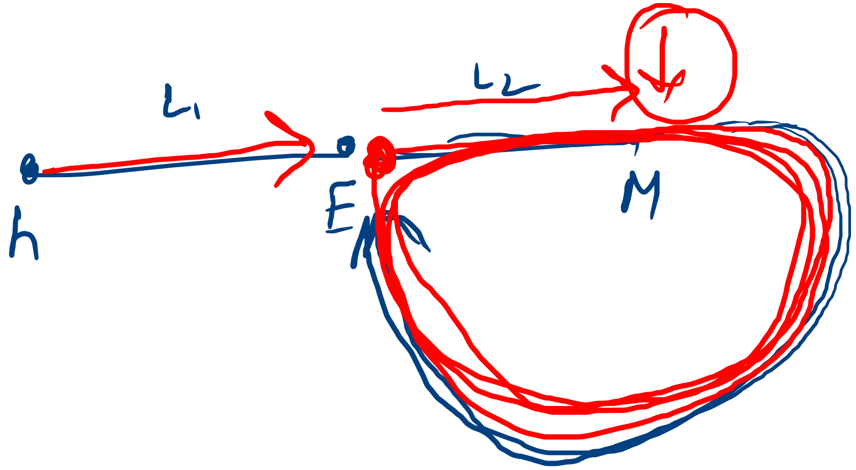


$(S = l_1 + l_2)$

$f = \underline{\boxed{L_1 + L_2}} + n\underline{(L_2 + x)}$





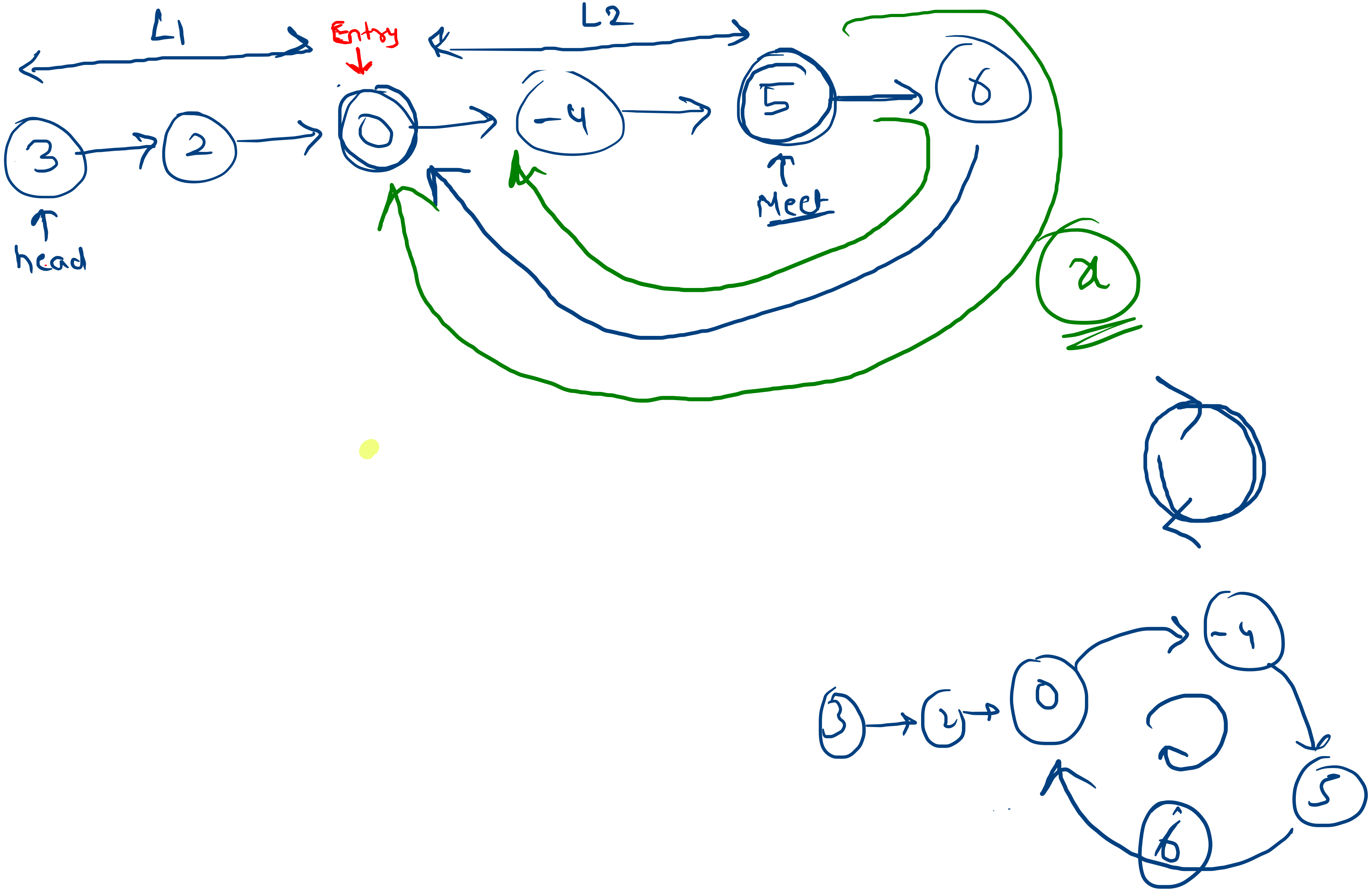


$$L_1 + n(L_2+x) + L_2$$

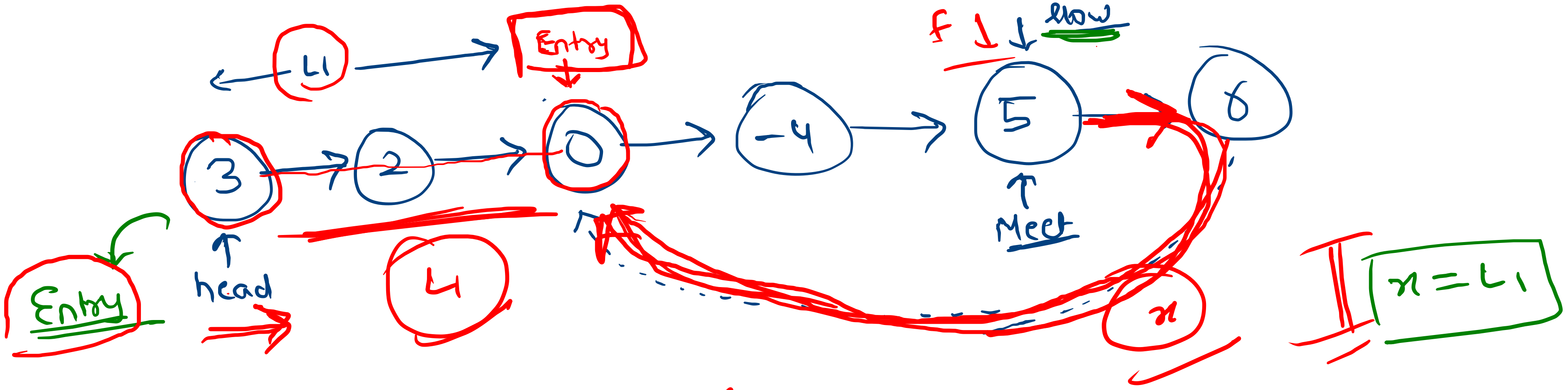
$$\boxed{L_1 + L_2 + n(L_2+x)}$$

Rough calculation

For distance



Rough Calculation

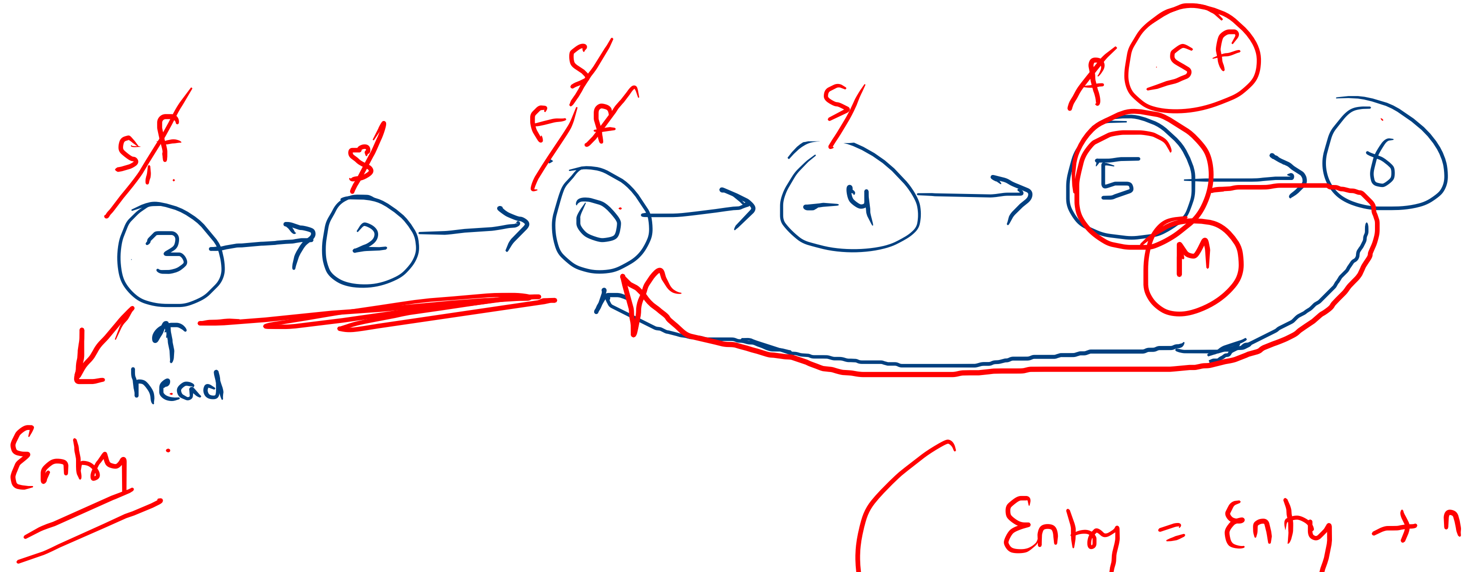


$x = L_1$

(Both run at same speed)

place they meet is Entry point

Final traversal



```
Entry = Entry → next
slow = slow → next

(Entry != slow)
```