142. Linked List Cycle II

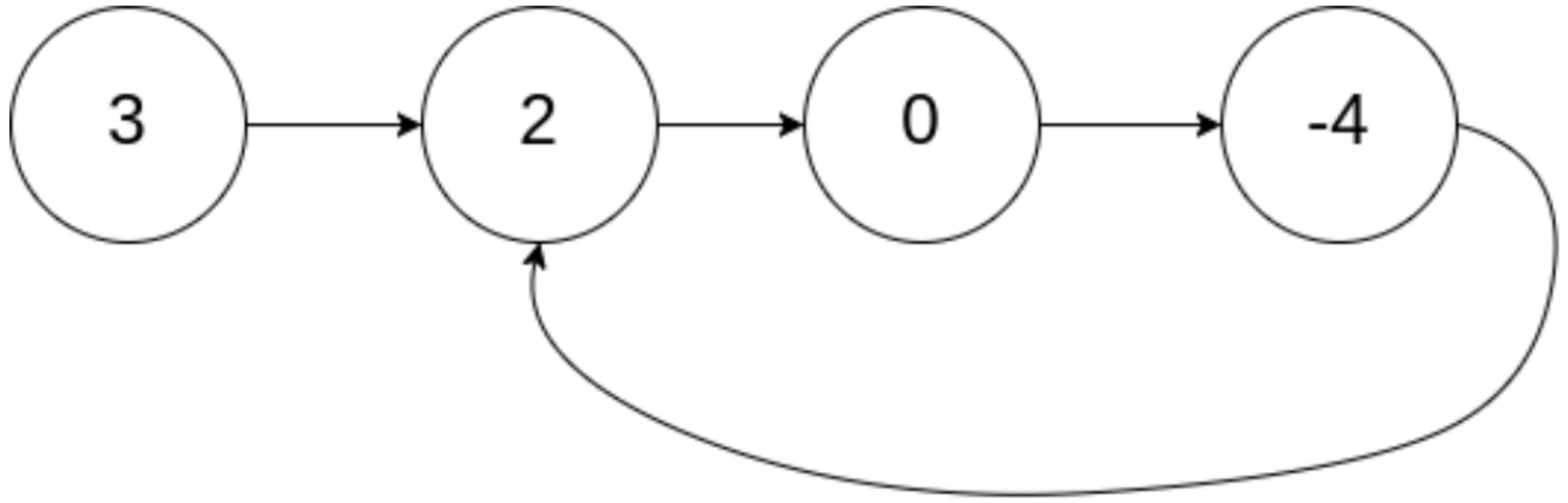
Medium 7245 476 Add to List Share

Given the `head` of a linked list, return *the node where the cycle begins. If there is no cycle, return* `null`.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to (**0-indexed**). It is `-1` if there is no cycle. **Note that** `pos` **is not passed as a parameter**.

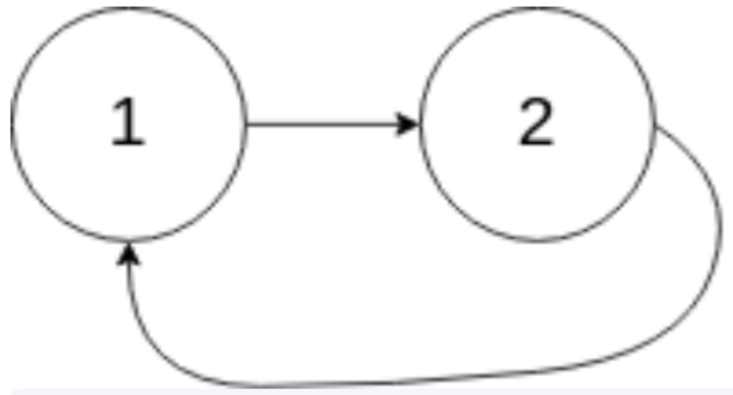
Do not modify the linked list.

Example 1:



```
Input: head = [3,2,0,-4], pos = 1
Output: tail connects to node index 1
Explanation: There is a cycle in the linked list, where tail connects to the
second node.
```

Example 2:



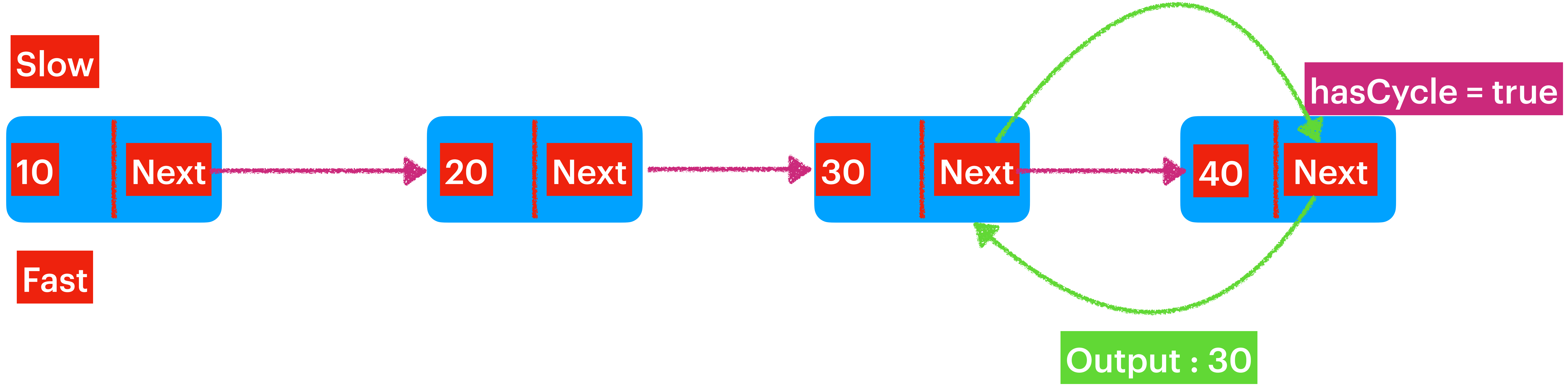
```
Input: head = [1,2], pos = 0
Output: tail connects to node index 0
Explanation: There is a cycle in the linked list, where tail connects to the
first node.
```

Example 3:

```
Input: head = [1], pos = -1
Output: no cycle
Explanation: There is no cycle in the linked list.
```

Constraints:

- The number of the nodes in the list is in the range $[0, 10^4]$.
- $-10^5 <= \text{Node.val} <= 10^5$
- `pos` is `-1` or a **valid index** in the linked-list.



Slow

10 Next

20 Next

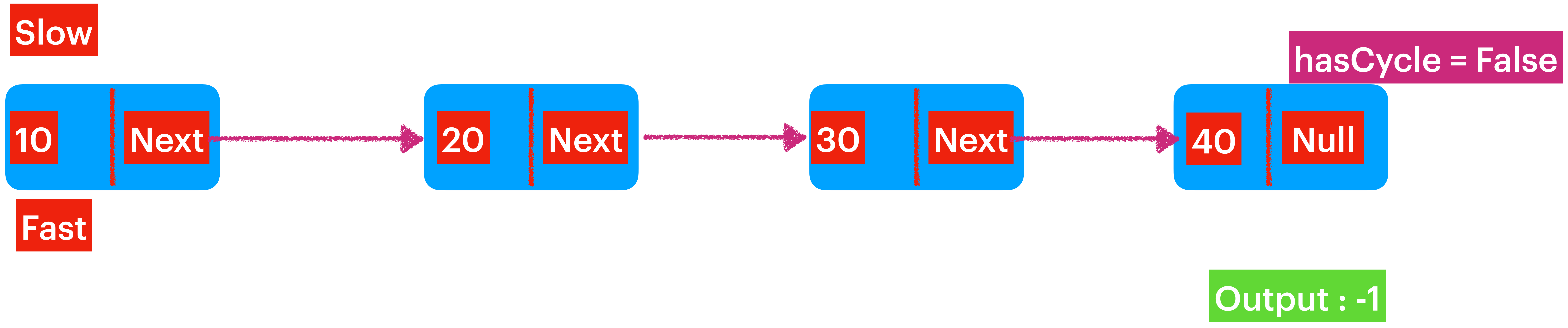
30 Next

40 Next

hasCycle = true

Fast

Output : 30



Slow

10 Next

20 Next

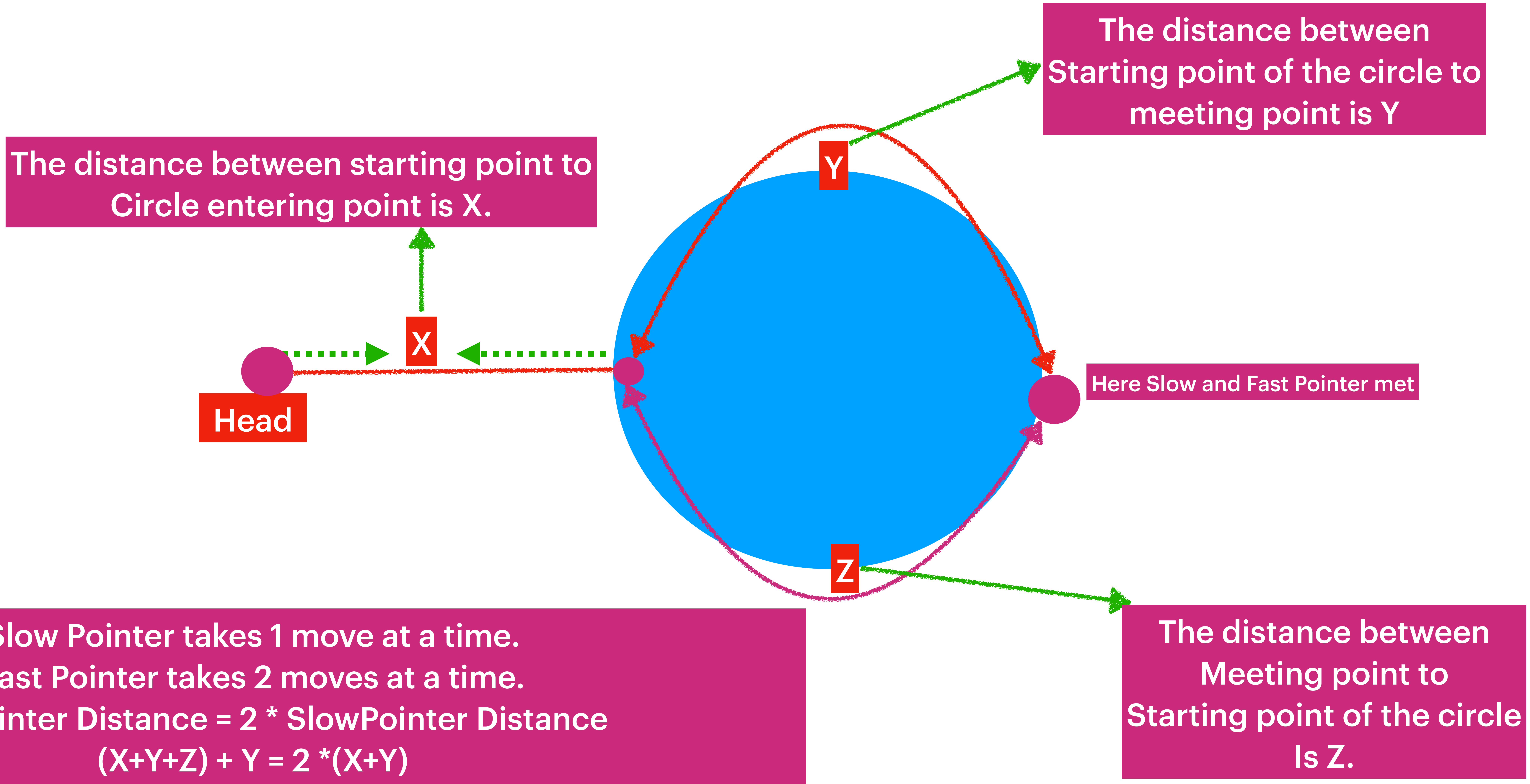
30 Next

40 Null

hasCycle = False

Fast

Output : -1



The distance between starting point to Circle entering point is X.

The distance between Starting point of the circle to meeting point is Y

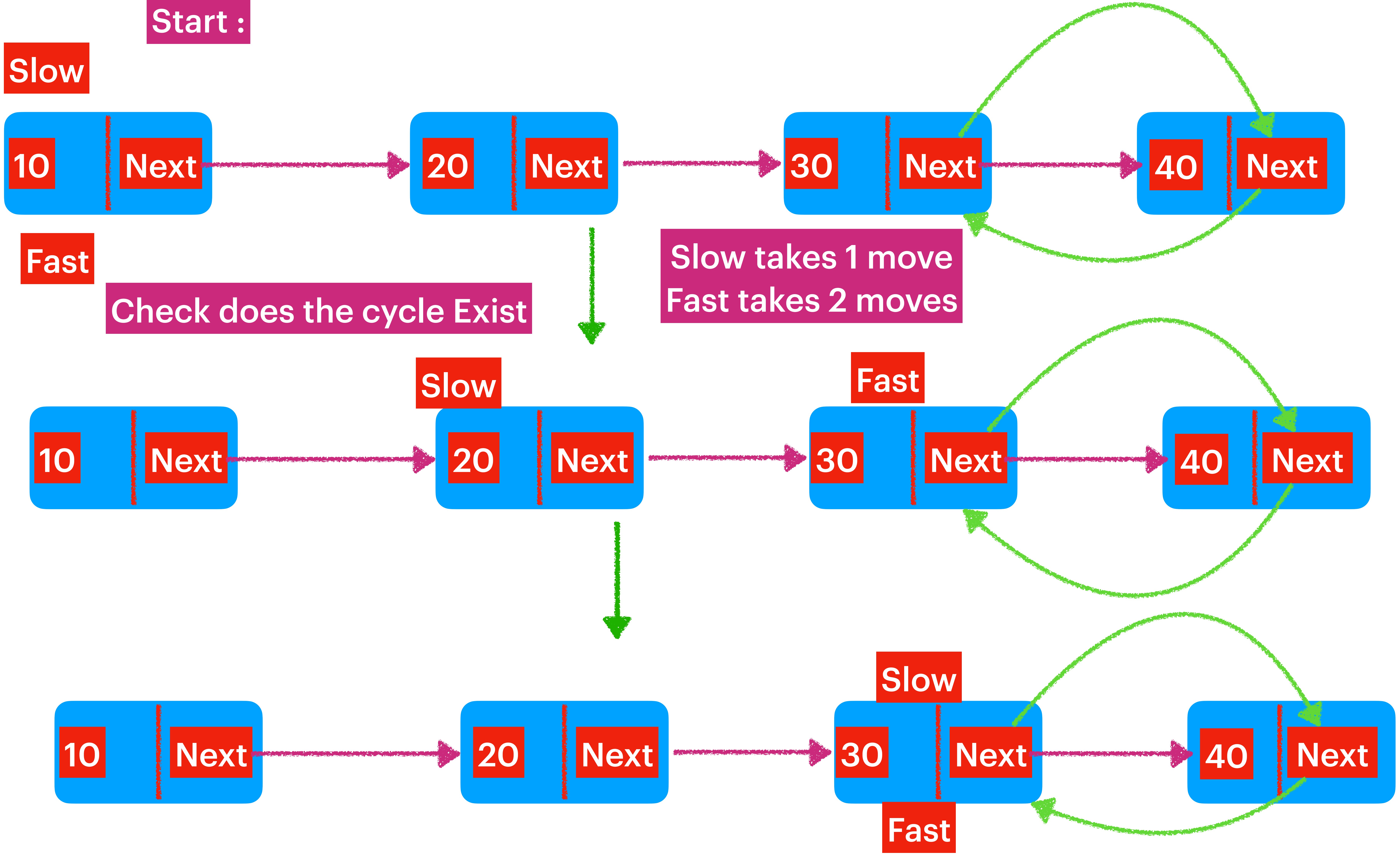
Here Slow and Fast Pointer met

The distance between Meeting point to Starting point of the circle Is Z.

Slow Pointer takes 1 move at a time.
Fast Pointer takes 2 moves at a time.
FastPointer Distance = 2 * SlowPointer Distance
$(X+Y+Z) + Y = 2 *(X+Y)$
$X+2Y+Z = 2X+2Y$

$Z = 2X+2Y -X-2Y$
$Z = X$
The distance between Meeting point to circle starting point [Z] is equals Head to circle starting point[X].

Start :



Check does the cycle Exist

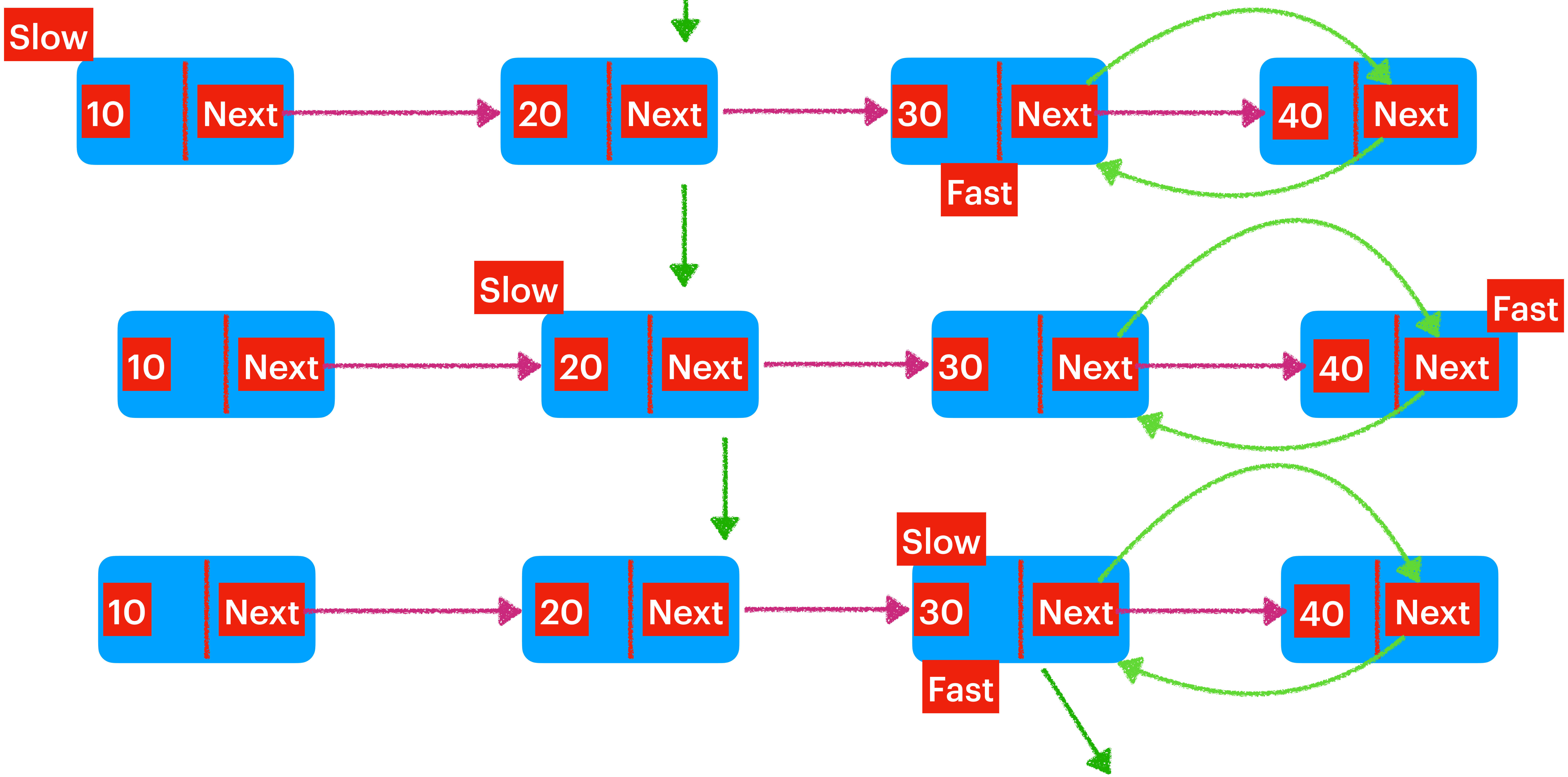
Slow takes 1 move
Fast takes 2 moves

There is a loop :

There is a loop , As per the equation we know X=Z
Keep fast in same position,
Point slow pointer to head ,
Then move both the pointers 1 move at a time.

Time Complexity : O(n)
Space Complexity : O(1)

The point where they meet is the starting point of the loop.



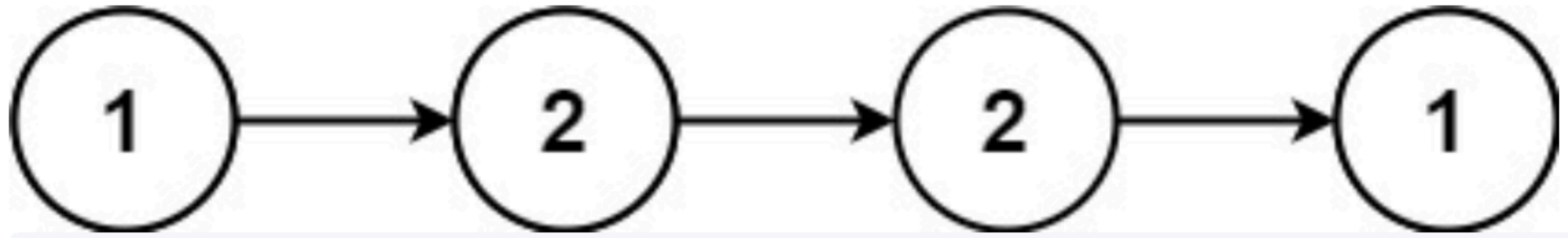
This is loop starting or interception point.

234. Palindrome Linked List

Easy 👍 8859 👎 540 ♡ Add to List Share

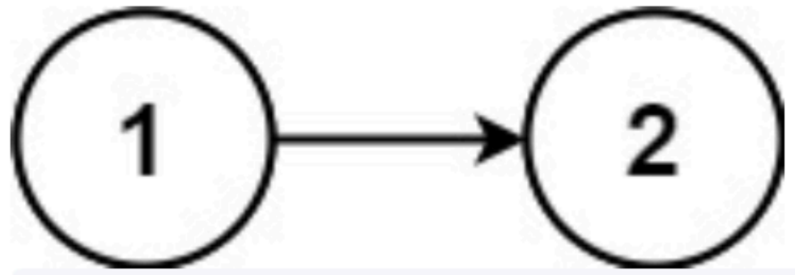
Given the `head` of a singly linked list, return `true` if it is a palindrome.

Example 1:



```
Input: head = [1,2,2,1]
Output: true
```

Example 2:

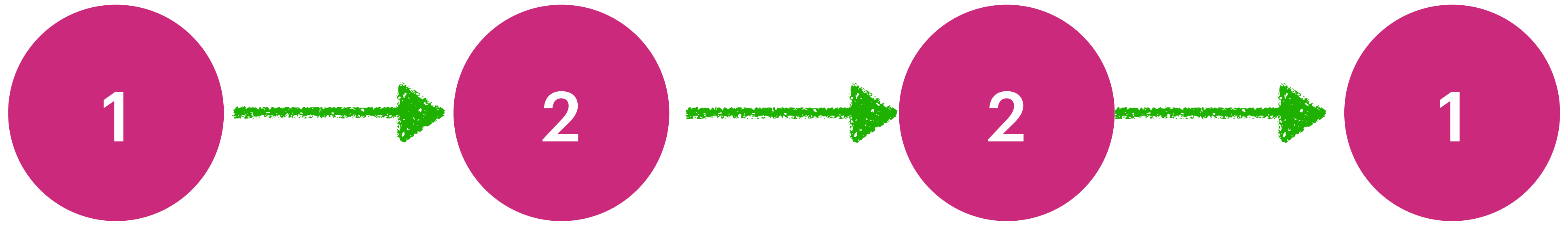


```
Input: head = [1,2]
Output: false
```

Constraints:

- The number of nodes in the list is in the range `[1, 10^5]`.
- `0 <= Node.val <= 9`

Follow up: Could you do it in `O(n)` time and `O(1)` space?



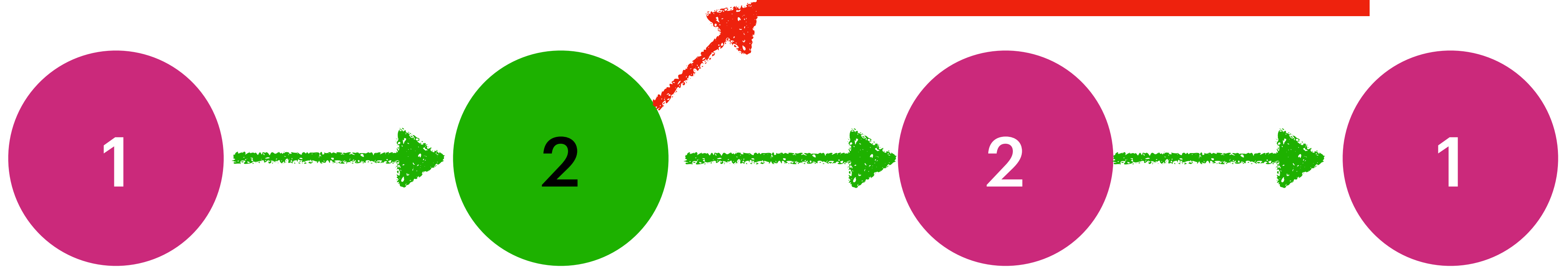
Time Complexity : O(n)
Space Complexity : O(n)

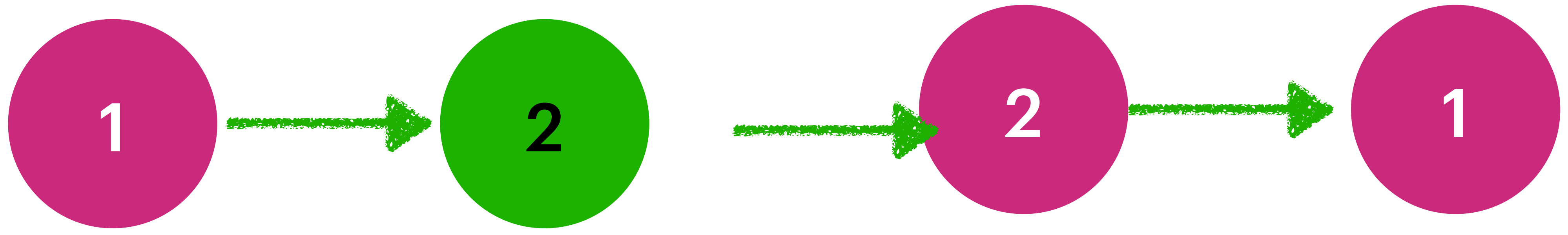
Copy all the node values to the ArrayList : [1,2,2,1]
Check ArrayList is Palindrome or not.

Lets Solve this in Constant Space

Time Complexity : O(n)
Space Complexity : O(1)

Find the mid Point

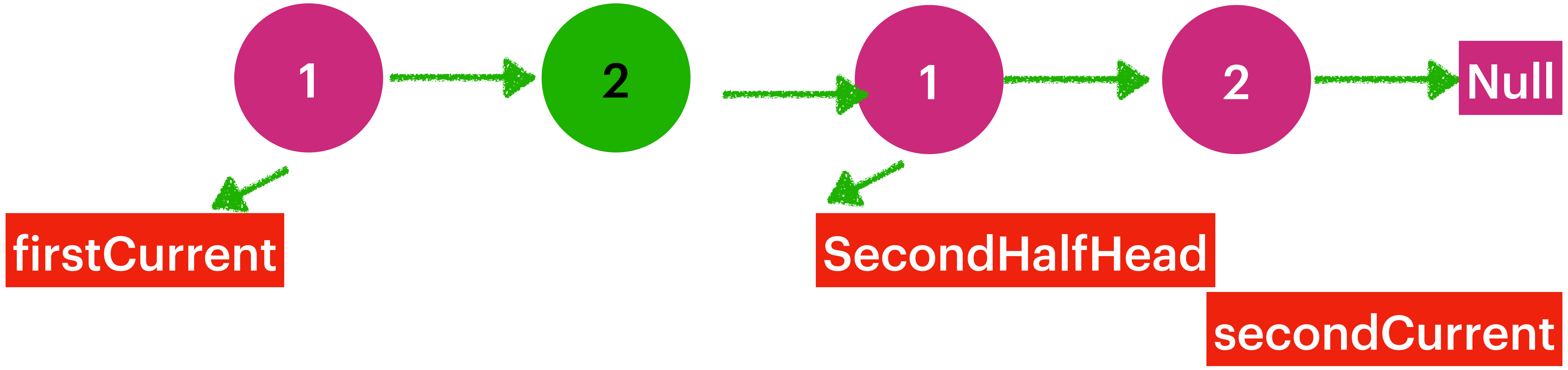




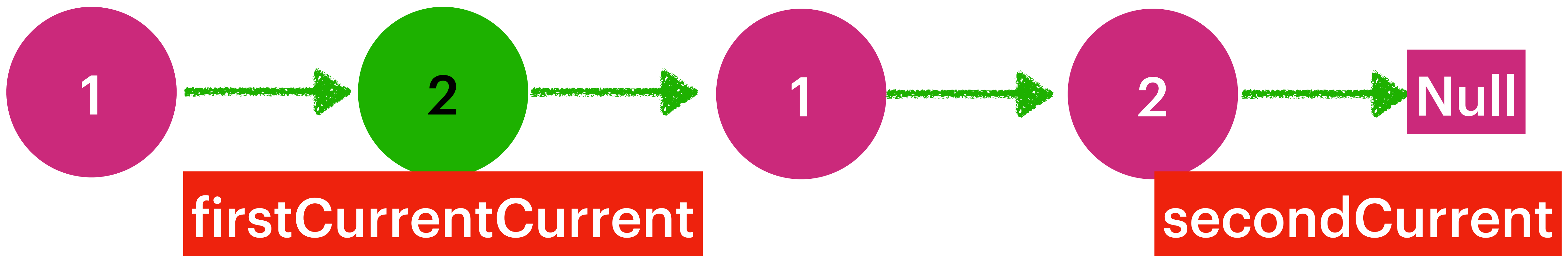
First Half

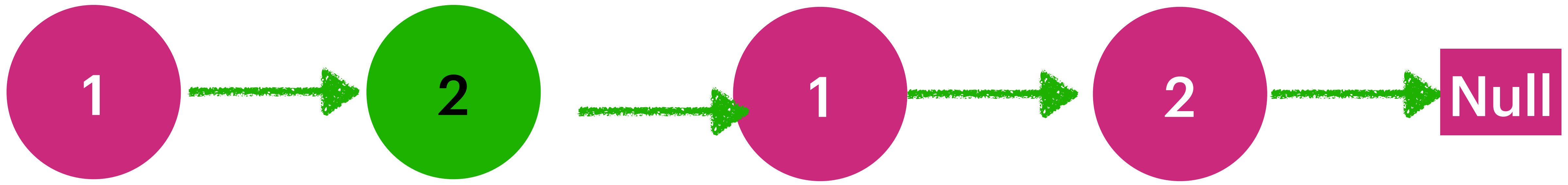
Second Half

Reverse the 2nd Half



At Each move verify does the first current is equals to 2ndcurrent





We identified LinkedList is Palindrome

As we modified list , do the reverse operation from mid.next