leetcode Question: Linked List Cycle II

Linked List Cycle II

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

Follow up:
Can you solve it without using extra space?

Analysis:

This problem can be viewed as two major steps:
(1) Detect whether the loop exists in the linked list.
(2) Find the loop start node if loop exists.

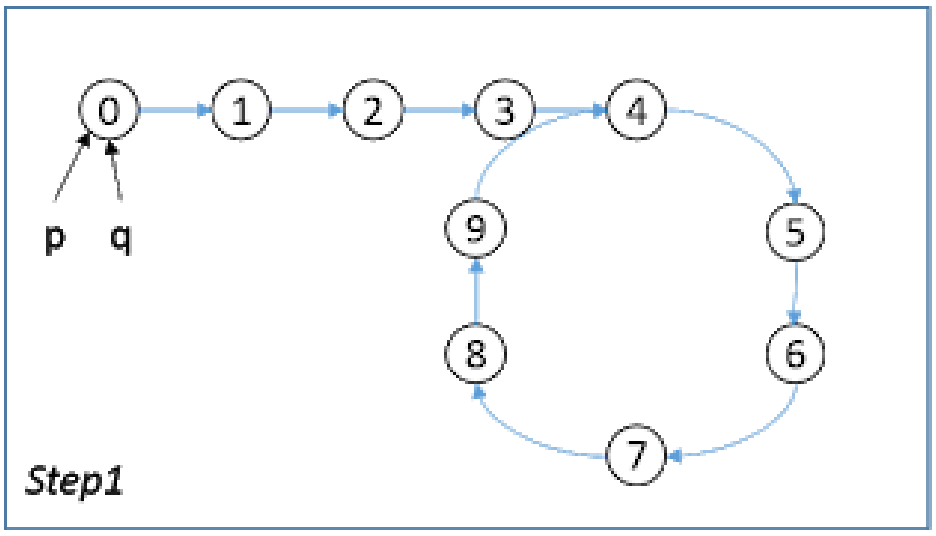
The (1) step can be easily solved using the slow&fast pointers (see Linked List Cycle)
How to deal with step (2)?

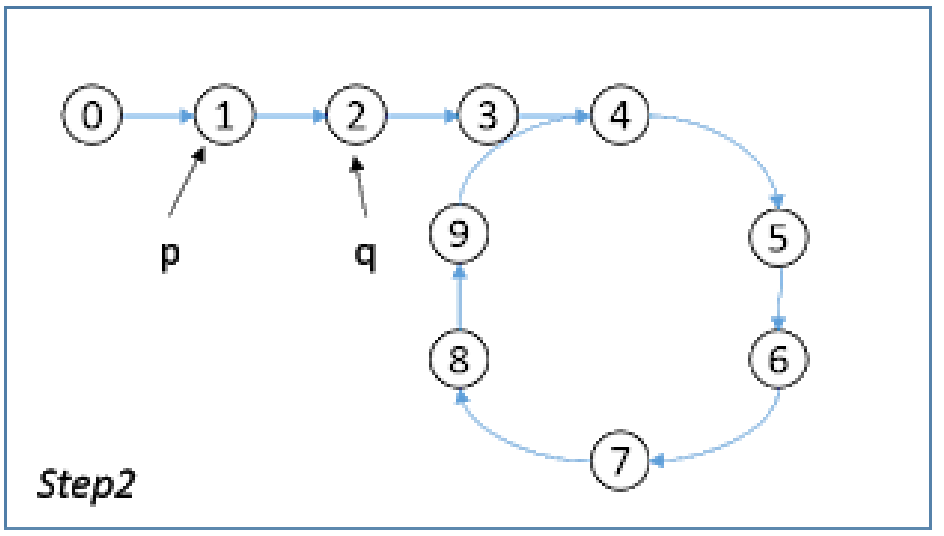
Firstly let us assume the slow pointer (S) and fast pointer (F) start at the same place in a **n node** circle. S run **t steps** while F can run **2t steps,** we want to know what is t (where they meet) , then just solve: **t mod n = 2t mod n**, it is not hard to know that when t = n, they meet, that is the start of the circle.

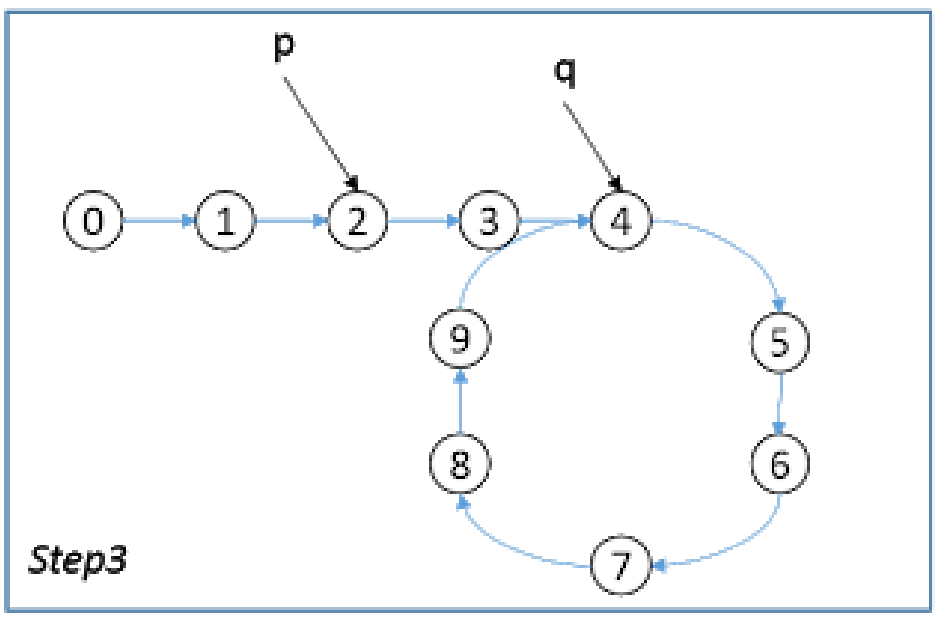
For our problem, we can consider the time when S enter the loop for the 1st time, which we assume k step from the head. At this time, the F is already in k step ahead in the loop. When will they meet next time?
Still solve the function: **t mod n = (k + 2t) mod n**
Finally, we can find out: **t = (n-k)**, S and F will meet, this is k steps before the start of the loop.

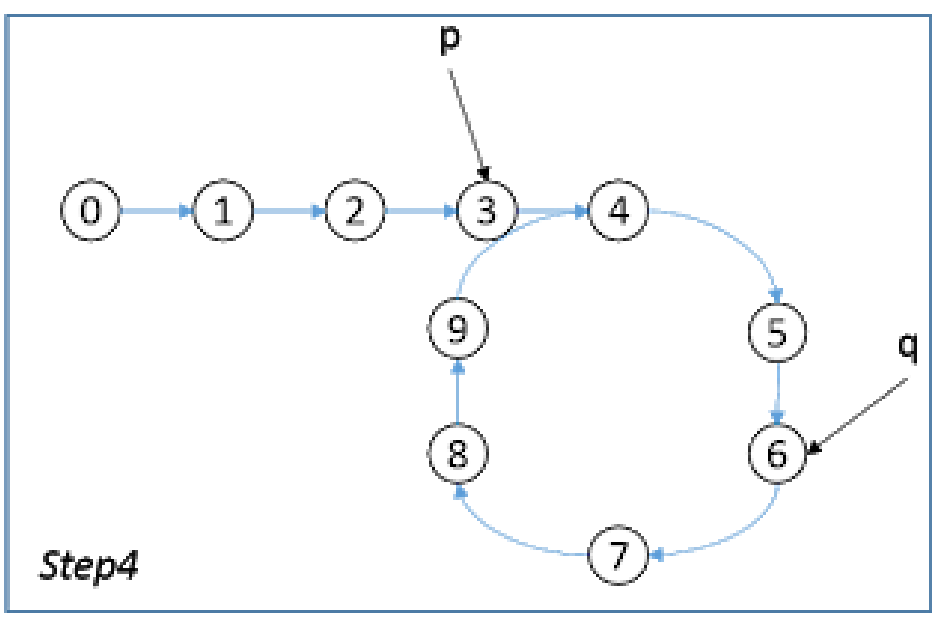
So, the Step (2) can be easily solved after Step(1):
Note that here S and F are not slow or fast pointers, but only regular pointers.
1. Set S to the head.
2. S = S -> next, F = F->next
3. output either one of the pointer until they meet.

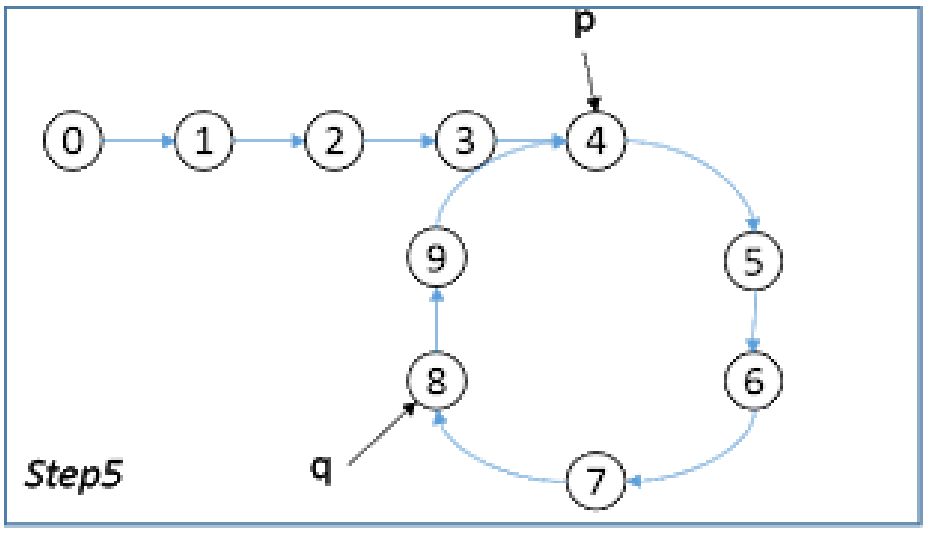
Here I show some figure for better illustration:
Note that, from step 5, p (slow pointer) firstly enter the loop, it will go (n-k) steps, and meet with fast pointer q. The rest of step is n-(n-k) = k, steps from current meet point, back to the start of the loop.

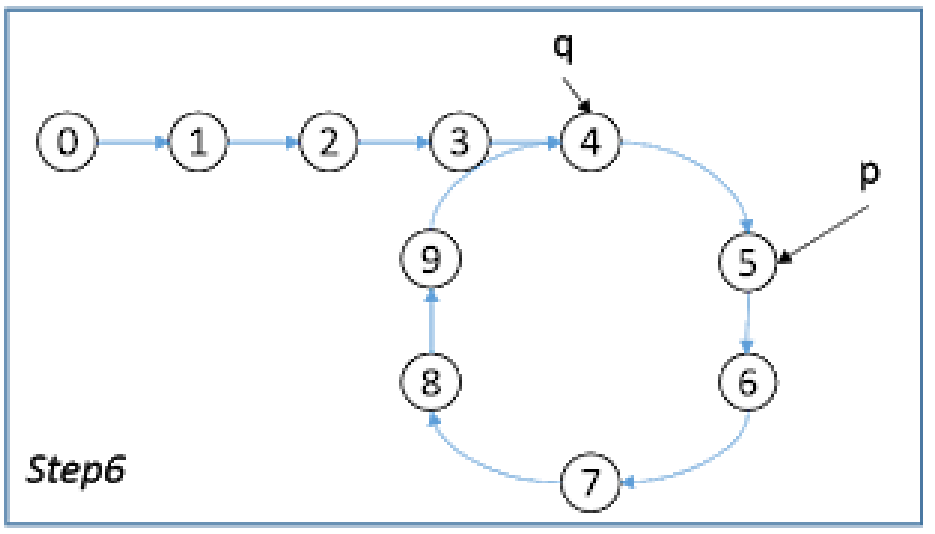

0
1
2
3
4
5
6
7
8
9
p
q
Step1

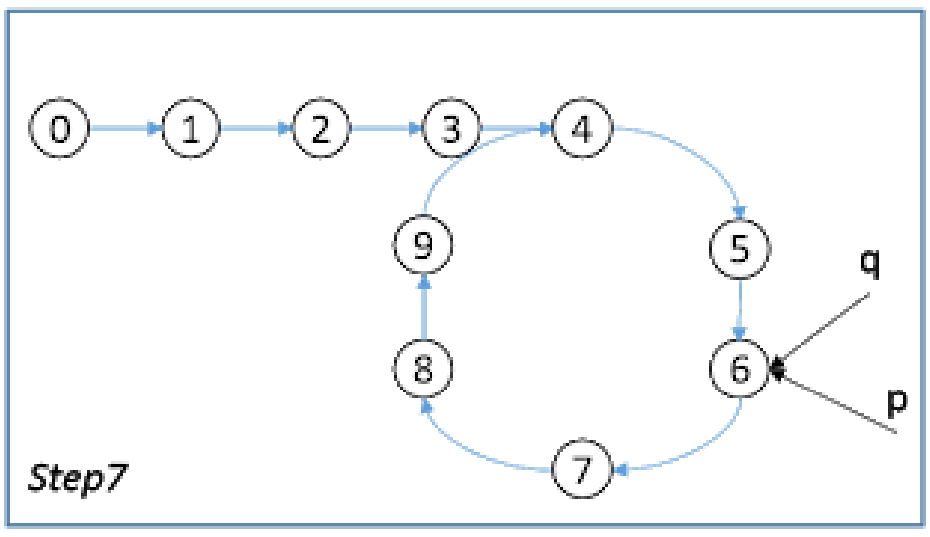

0
1
2
3
4
5
6
7
8
9
p
q
Step2

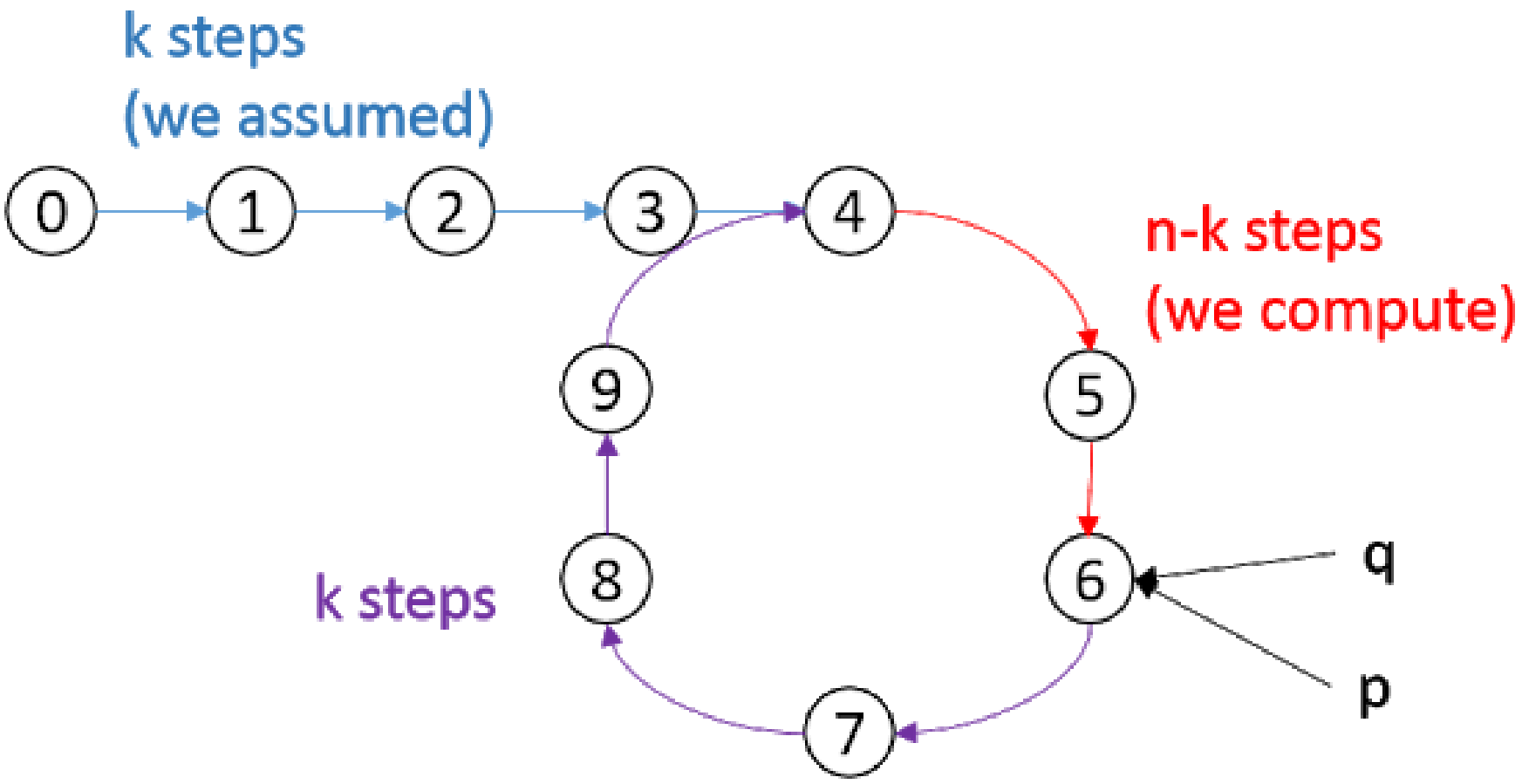

0
1
2
3
4
5
6
7
8
9
p
q
Step3

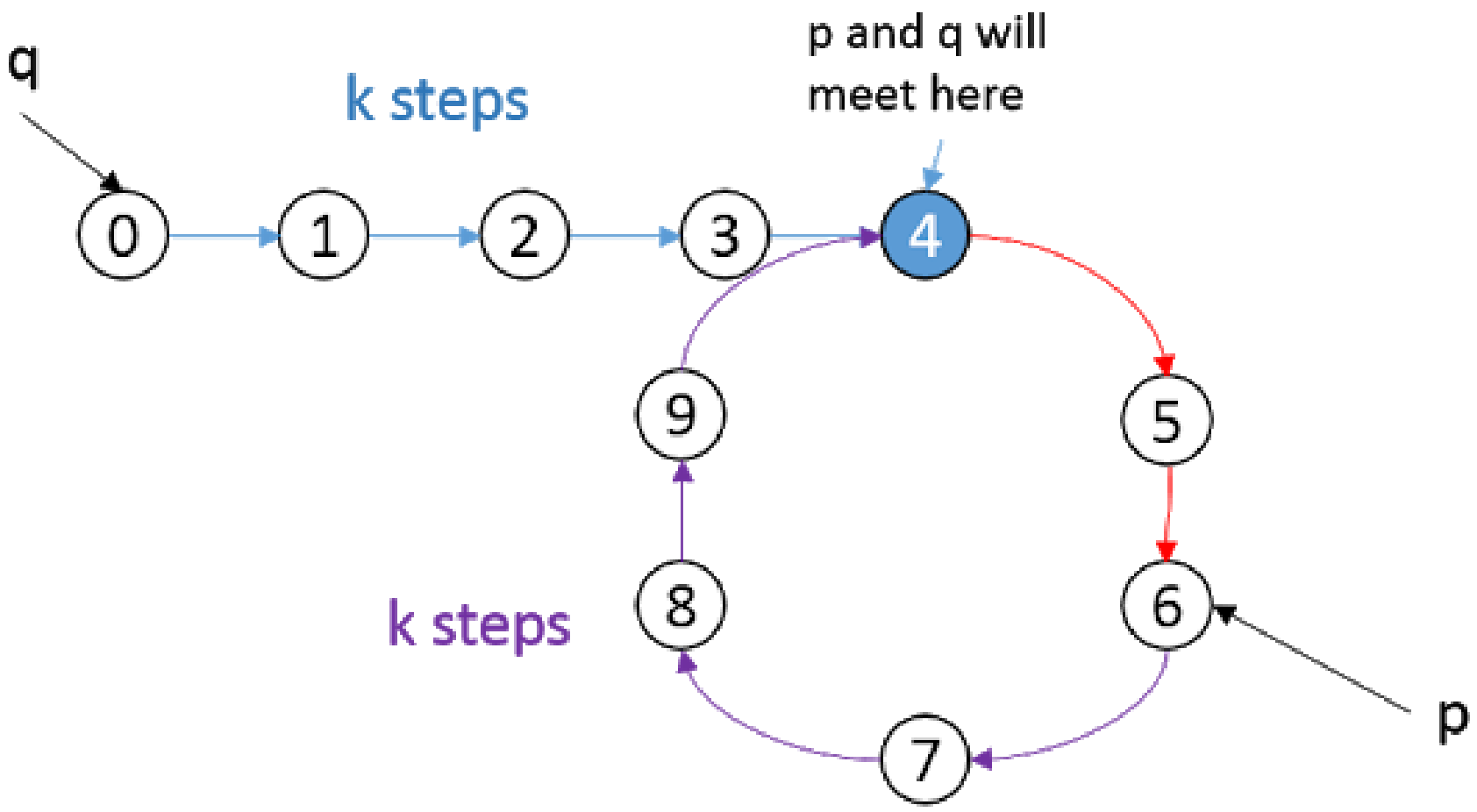

0
1
2
3
4
5
6
7
8
9
p
q
Step4


0
1
2
3
4
5
6
7
8
9
p
q
Step5


0
1
2
3
4
5
6
7
8
9
q
p
Step6


0
1
2
3
4
5
6
7
8
9
q
p
Step7


k steps
(we assumed)
0
1
2
3
4
n-k steps
(we compute)
9
5
k steps
8
6
q
7
p


q
k steps
p and q will
meet here
0
1
2
3
4
9
5
k steps
8
6
p
7