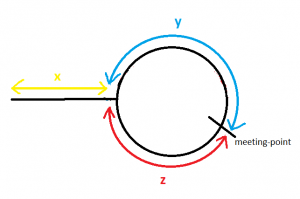
**Detecting start of a loop in singly Linked List**

**Steps to find the starting point of the loop?**

* Once we know for sure that a loop is present, by Floyd algorithm(using fast and slow pointers).
* Move the slowPointer to start of the list,(i.e headNode) and let fastPointer remain there at the meeting point
* Now move both the pointers one node at a time(Yes! You heard it right. Now even fastPointer moves at one node at a time)
* The point where both pointers will meet, is our required start of the loop.

By now it had already started itching in mind that, **Why the hell does moving slow Pointer to start of list and moving both pointer one step at a time will find the start of the loop?**

**Proof1: Assumed that only one cycle/loop to reach the meeting point:**

[](https://web.archive.org/web/20160401024212/http:/learningarsenal.info/wp-content/uploads/2015/08/loop-explanation.png)

Total no of nodes in the list n = x + y + z = x + l

where,

x – No of nodes from head to starting point of the loop

y – No of nodes from starting point of the loop to the meeting point

z – No of nodes from meeting point to starting point of the loop

l(L) – Length of cycle; l = y + z

Here we should prove that, number of nodes from the head node to starting point of the loop(x) is equal to number of nodes from the meeting point to starting point of the loop(z), i.e x =z

Distance traveled by slowPointer before meeting point= x + y

Distance traveled by fastPointer before meeting point = x + y + l

= x + y + y + z, where l =y + z

= x + 2y + z

Since fastPointer travels with double the speed of slowPointer, and time is constant for both when the reach the meeting point. So by using simple speed, time and distance relation

2(slow pointer traversals) = fast pointer traversals

2(x+y) = x+2y+z

2x+2y = x+2y+z

x=z; Hence proved

So by moving slowPointer to start of linked list, and making both slowPointer and fastPointer to move one node at a time, they both will reach at the point where the loop starts in the linked list.

**Proof2: Considering multiple cycles/loops in both slow and fast pointers:**

Proof1 was assumed only single loop for both pointers, but distance traveled by slow pointer should be (x + y + l\*i) and the distance traveled by the fast pointer should be (x + y + l\*j) where i and j are the number of cycles traversed by the slow and fast pointer, l is length of the cycle

Distance traveled by slowPointer before meeting= x + y + l\*i

Distance traveled by fastPointer before meeting = x + y + l\*j

Since fastPointer travels with double the speed of slowPointer, and time is constant for both when the reach the meeting point.

So by using simple speed, time and distance relation

2(x+y+l\*i)= x+y+l\*j

x+y = l\*j – 2\*l\*i

x+y = l(j – 2\*i); Assume j – 2i = constant k;

x+y = l\*k;

x = l\*k – y;

Here Intuition is,

Since it is a cycle or circle, it does not matter how many times it runs into circle. It will come to same point again. Example:

If k=3, After 2 times pointer will come to same point and last one rotation will be helpful to find the starting point of the loop.

If k=5, After 4 times pointer will come to same point and last one rotation will be helpful.

So we can eliminate the k-1 rotations and consider only one rotation and equation will become,

x = l – y

x = (y+z) – y, where length of cycle l = y+z

x = z; Hence proved