**Two-Pointer in Linked List**

[Report Issue](https://github.com/LeetCode-Feedback/LeetCode-Feedback/issues)

Let's start with a classic problem:

Given a linked list, determine if it has a cycle in it.

You might have come up with the solution using the hash table. But there is a more efficient solution using the two-pointer technique. Try to think it over by yourself before reading the remaining content.

Imagine there are two runners with different speed. If they are running on a straight path, the fast runner will first arrive at the destination. However, if they are running on a circular track, the fast runner will catch up with the slow runner if they keep running.

That's exactly what we will come across using two pointers with different speed in a linked list:

1. If there is no cycle, the fast pointer will stop at the end of the linked list.
2. If there is a cycle, the fast pointer will eventually meet with the slow pointer.

So the only remaining problem is:

What should be the proper speed for the two pointers?

It is a safe choice to move the slow pointer one step at a time while moving the fast pointer two steps at a time. For each iteration, the fast pointer will move one extra step. If the length of the cycle is *M*, after *M* iterations, the fast pointer will definitely move one more cycle and catch up with the slow pointer.

What about other choices? Do they work? Would they be more efficient?