算法题练习 -- 判断单链表中是否有环

作者：zhengliwei
原文链接：https://ld246.com/article/1602241968993
来源网站：链滴

hello，大家好，欢迎来到银之庭。我是Z，一个普通的程序员。一个假期过去了，也没更新文章，今我们来做一道简单的算法练习，热热手吧。

1. 题目

这道题是经典的程序员入门题，也是一些善良的面试官喜欢出的题目，主要是给面试者找自信用的mile 。题目为：给出一个单链表的头节点，判断这条单链表中是否有环。链表节点定义如下：

```
class Node {
    public int value;
    public Node next;
}
```

PS: 这里为了更直观地表现出算法的含义，直接把内部变量定义成了public的，工程上是肯定不会这做的。

2. 思路分析

这是个典型的快慢指针方法能解决的问题。所谓快慢指针，就是定义两个指针，都从链表头部开始向遍历，快指针一次移动两个节点，慢指针一次移动一个节点，假如链表无环，则快指针一定先移动到表尾部；假如链表有环，则快指针先进入环，然后死循环，等待慢指针进入环，这时快指针距离慢指n个节点，假设环由m个节点组成，则一定有0 <= n < m，由于快慢指针每次移动后他们之间的距离会减一，当移动n次后，他们就会重合。由上面的推导可以得出结论：如果在快指针变成null（即移到链表结尾）前，出现了 慢指针==快指针（即两个指针重合）的情况，则可以确定链表有环。

上面这段分析希望大家可以完全理解，因为快慢指针方法的代码实现不难，但为什么两个指针重合时以得出有环的结论，这个推导过程还是需要理解一下的。最后，写代码的时候别忘了考虑头节点为空只有一个节点等特殊情况哦。

3. 代码实现

话不多说，直接上代码：

```
    public boolean haveRing(Node head) {
        // 头节点为空或只有一个节点且不是自身成环的情况，直接返回false
        if (head == null || head.next == null) {
            return false;
        }

        Node fasterNode = head; // 快指针
        Node slowerNode = head; // 慢指针
        while (fasterNode != null) {
            // 考虑快指针的next节点是空的情况，即链表无环，且快指针到达最后一个节点的情况
            fasterNode = fasterNode.next != null ? fasterNode.next.next : null;
            slowerNode = slowerNode.next;
            if (fasterNode == slowerNode) { // 如果快慢指针重合，则证明有环
                return true;
            }
        }

        // 跳出循环证明快指针为null，则链表无环
        return false;
```

```
    }
```

注释已经写的比较清晰了，大家可以自己构造几种情况来测试一下代码的效果，如头节点为空；只有个节点无环；只有一个节点自己成环；共两个节点且两个节点成环；共两个节点且后一个节点成环等况。