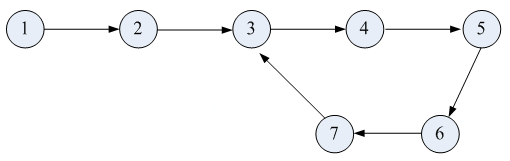
如何判断单链表是否存在环

<https://www.cnblogs.com/kkdd-2013/p/5335031.html>

## 题目描述：

单链表有环的定义：链表的尾节点指向了链接中间的某个节点。

如下图所示，如果有环，则遍历到结点7时，又重新回到结点3，结点3就是环的入口结点。



## 思路介绍：

采用快慢指针的思想，设两个工作指针，一个快一个慢，如果链表有环，它们必然会在某个结点处相遇。

## C代码

/\* 判断链表是否有环 \*/

int hasLoop(Node \*head)

{

Node \*p1,\*p2;

if(head == NULL || head->next == NULL) //链表为空，或是单结点链表直接返回头结点

return 0;

p1 = p2 = head;

while(p1->next != NULL && p1->next->next != NULL)

{

p1 = p1->next->next;

p2 = p2->next;

if(p1 == p2)

return 1;

}

return 0;

}

**这里需要注意一个问题，为什么快慢指针相遇就说明链表有环呢？该如何分析？？**

（以下内容来自：http://www.cnblogs.com/youxin/p/3303172.html）

1.链表中是否有环的判断

可以设置两个指针(fast,slow)，初始值均指向头，slow每次向前一步，fast每次向前两步；

如果链表中有环，则fast先进入环中，而slow后进入环中，两个指针在环中必定相遇;

如果fast遍历到尾部为NULL，则无环

2.链表有环，判断环的入口点

当fast若与slow相遇时，slow肯定没有走遍历完链表，而fast已经在环内循环了n圈(1<=n)。假设slow走了s步，则fast走了2s步（fast步数还等于s 加上在环上多转的n圈），设环长为r，则：

2s = s + nr

s= nr

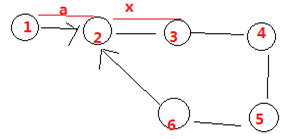
设整个链表长L，入口环与相遇点距离为x，起点到环入口点的距离为a。

a + x = nr

a + x = (n – 1)r +r = (n-1)r + L - a

a = (n-1)r + (L – a – x)

(L – a – x)为相遇点到环入口点的距离，由此可知，从链表头到环入口点等于(n-1)循环内环+相遇点到环入口点



（L-a-x为相遇点到环入口点的距离，怎么理解，比如上面的，我们假设slow和fast在点3相遇，

启动为1，环入口点 为2，相遇点为3，走了（L-a-x）长的距离后就回到了2点。

我们在起点和环相遇点各设置一个指针，每次各走一步，必定相遇。相遇一定在2点，为什么，

）

因而，可以在链表头，相遇点分别设定一个指针，每次各走一步，两个指针必定相遇，则相遇第一点为环入口点。