**关于有环链表的问题**

2011-05-12 16:46:00|  分类： [数据结构](http://churuimin425.blog.163.com/blog/#m=0&t=1&c=fks_084065083083085067081083080095087082080065093094081068) |  标签：[链表](http://churuimin425.blog.163.com/blog/#m=0&t=3&c=链表)  [环](http://churuimin425.blog.163.com/blog/#m=0&t=3&c=环)   |举报 |字号大中小 订阅

 【摘要】有一个单链表，其中可能有一个环，也就是某个节点的next指向的是链表中在它之前的节点，这样在链表的尾部形成一环。1、如何判断一个链表是不是这类链表？2、如果链表为存在环，如果找到环的入口点？扩展：判断两个单链表是否相交，如果相交，给出相交的第一个点。

有一个单链表，其中可能有一个环，也就是某个节点的next指向的是链表中在它之前的节点，这样在链表的尾部形成一环。

问题：

1、如何判断一个链表是不是这类链表？  
2、如果链表为存在环，如果找到环的入口点？

解答：

一、判断链表是否存在环，办法为：

设置两个指针(fast, slow)，初始值都指向头，slow每次前进一步，fast每次前进二步，如果链表存在环，则fast必定先进入环，而slow后进入环，两个指针必定相遇。(当然，fast先行头到尾部为NULL，则为无环链表)程序如下：

bool IsExitsLoop(slist \*head)  
{  
    slist \*slow = head, \*fast = head;  
  
    while ( fast && fast->next )   
    {  
        slow = slow->next;  
        fast = fast->next->next;  
        if ( slow == fast ) break;  
    }  
  
    return !(fast == NULL || fast->next == NULL);  
}

二、找到环的入口点

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，于是我们从链表头、与相遇点分别设一个指针，每次各走一步，两个指针必定相遇，且相遇第一点为环入口点。程序描述如下：

slist\* FindLoopPort(slist \*head)  
{  
    slist \*slow = head, \*fast = head;//特别要注意 以 head 作为共同的起点 否则会造成死循环  
  
    while ( fast && fast->next )   
    {  
        slow = slow->next;  
        fast = fast->next->next;  
        if ( slow == fast ) break;  
    }  
  
    if (fast == NULL || fast->next == NULL)  
        return NULL;  
  
    slow = head;  
    while (slow != fast)  
    {  
         slow = slow->next;  
         fast = fast->next;  
    }  
  
    return slow;  
}

扩展问题：

判断两个单链表是否相交，如果相交，给出相交的第一个点（两个链表都不存在环）。

比较好的方法有两个：

一、将其中一个链表首尾相连，检测另外一个链表是否存在环，如果存在，则两个链表相交，而检测出来的依赖环入口即为相交的第一个点。

二、如果两个链表相交，那个两个链表从相交点到链表结束都是相同的节点，我们可以先遍历一个链表，直到尾部，再遍历另外一个链表，如果也可以走到同样的结尾点，则两个链表相交。

这时我们记下两个链表length，再遍历一次，长链表节点先出发前进(lengthMax-lengthMin)步，之后两个链表同时前进，每次一步，相遇的第一点即为两个链表相交的第一个点。