单链表

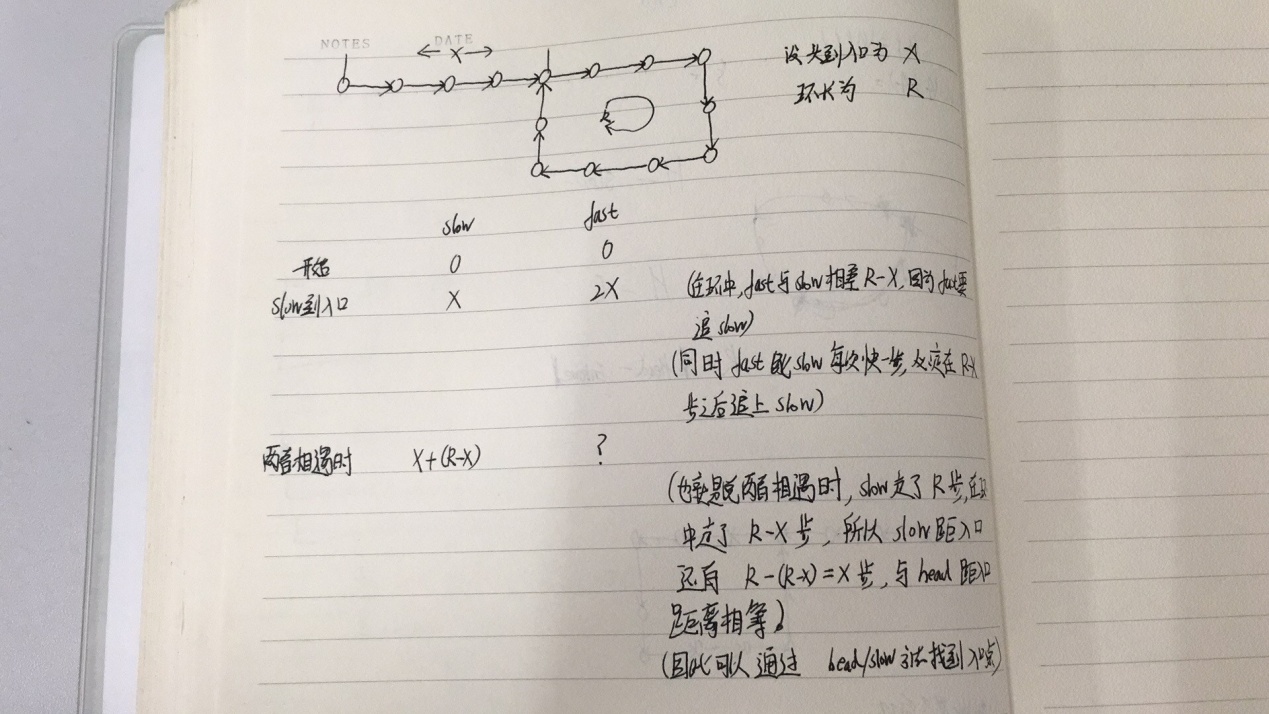
（<https://www.cnblogs.com/dancingrain/p/3405197.html>）

1，**给一个单链表，判断其中是否有环的存在；**

1. bool exitLoop(Node \*head)
2. {   //设置两个指针，一个快，一个慢，一开始都指向头
3. Node \*fast, \*slow ;
4. slow = fast = head ;
6. //快指针一次走两步，慢指针一次走一步，如果有环，快指针一定会追上慢指针
7. **while** (slow != NULL && fast -> next != NULL)
8. {
9. slow = slow -> next ;
10. fast = fast -> next -> next ;
11. //如果追上，则证明有环
12. **if** (slow == fast)
13. **return** **true** ;
14. }
15. //如果slow或者fast遍历到了尾节点，则证明无环
16. **return** **false** ;
17. }

**2，如果存在环，找出环的入口点；**

1. Node\* findLoopStart(Node \*head)
2. {   //同样，先判断有无环
3. Node \*fast, \*slow ;
4. slow = fast = head ;
5. **while** (slow != NULL && fast -> next != NULL)
6. {
7. slow = slow -> next ;
8. fast = fast -> next -> next ;
9. **if** (slow == fast)
10. **break** ;
11. }
12. //没有环，返回NULL值
13. **if** (slow == NULL || fast -> next == NULL)
14. **return** NULL ;
16. //找寻入口点，重新设置两个指针，一个指向头结点，一个指向slow节点
17. Node \* ptr1 = head ; //链表开始点
18. Node \* ptr2 = slow ; //相遇点
19. //当两个节点相遇时，即使环的入口节点
20. **while** (ptr1 != ptr2)
21. {
22. ptr1 = ptr1 -> next ;
23. ptr2 = ptr2 -> next ;
24. }
25. **return** ptr1 ; //找到入口点
26. }



**3，如果存在环，求出环上节点的个数；**

由上图可知，当slow和fast第一次相遇时，slow就已经走了环的长度，我们只需要统计一下slow的步数即可。

1. int countLooppoint(Node \*head)
2. {   //同样，先判断有无环
3. Node \*fast, \*slow ;
4. slow = fast = head ;
5. int loop\_point = 0;
6. **while** (slow != NULL && fast -> next != NULL)
7. {
8. slow = slow -> next ;
9. fast = fast -> next -> next ;
10. loop\_point++;
11. **if** (slow == fast)
12. **break** ;
13. }
14. //没有环，返回NULL值
15. **if** (slow == NULL || fast -> next == NULL)
16. **return** 0 ;
17. return loop\_point;
18. }

**4，如果存在环，求出链表的长度；**

先求出环的长度，在求出到入口的长度，和便是链表长度。

1. int countLooppoint(Node \*head)
2. {
3. Node \*fast, \*slow ;
4. slow = fast = head ;
5. int loop\_point = 0;
6. **while** (1)
7. {
8. slow = slow -> next ;
9. fast = fast -> next -> next ;
10. loop\_point++;
11. **if** (slow == fast)
12. **break** ;
13. }
14. Node \* ptr1 = head;
15. Node \* ptr2 = slow;
16. int dis\_entrance=0;
17. **while** (ptr1 != ptr2)
18. {
19. ptr1=ptr1->next;
20. ptr2=ptr2->next;
21. dis\_entrance++;
22. }
23. return loop\_point+dis\_entrance;
24. }