单链表

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

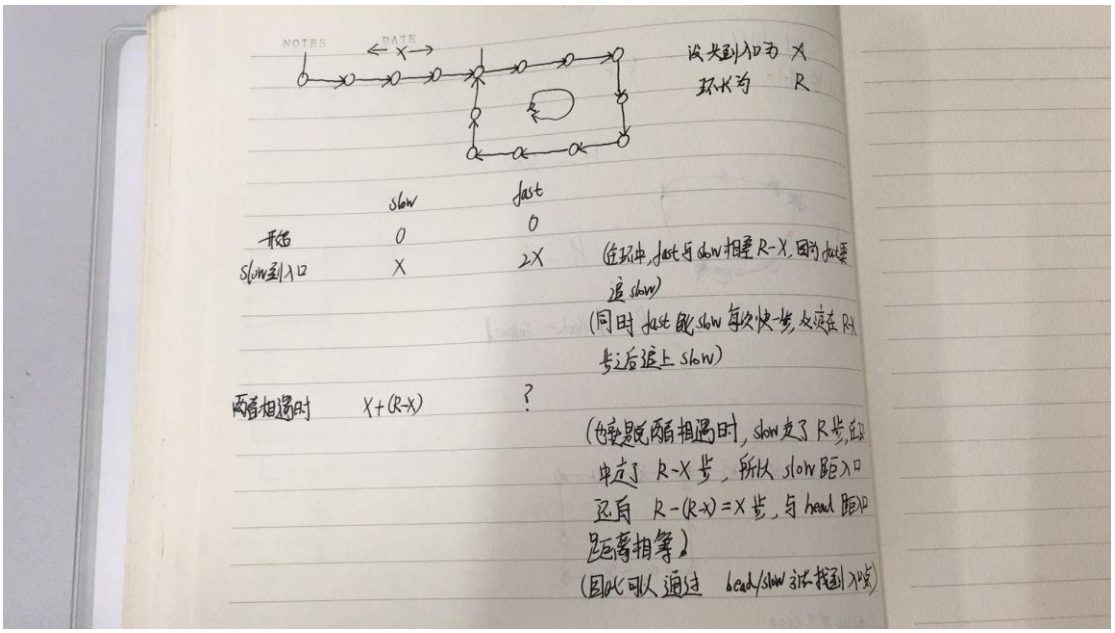
1， 给一个单链表，判断其中是否有环的存在；

```
bool exitLoop(Node *head)
{   //设置两个指针，一个快，一个慢，一开始都指向头
    Node *fast, *slow ;
    slow = fast = head ;

    //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
    while (slow != NULL && fast -> next != NULL)
    {
        slow = slow -> next ;
        fast = fast -> next -> next ;
        //如果追上，则证明有环
        if (slow == fast)
            return true ;
    }
   //如果 slow 或者 fast 遍历到了尾节点，则证明无环
    return false ;
}
```

2，如果存在环，找出环的入口点；

```
Node* findLoopStart(Node *head)
{   //同样，先判断有无环
    Node *fast, *slow ;
    slow = fast = head ;
    while (slow != NULL && fast -> next != NULL)
    {
        slow = slow -> next ;
        fast = fast -> next -> next ;
        if (slow == fast)
            break ;
    }
   //没有环，返回 NULL 值
    if (slow == NULL || fast -> next == NULL)
       return NULL ;

    //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
    Node * ptr1 = head ; //链表开始点
    Node * ptr2 = slow ; //相遇点
    //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。

```
int countLooppoint(Node *head)
{
    Node *fast, *slow ;
    slow = fast = head ;
   int loop_point = 0;
    while (1)
    {
        slow = slow -> next ;
        fast = fast -> next -> next ;
        loop_point++;
       if (slow == fast)
            break ;
    }
    Node * ptr1 = head;
   Node * ptr2 = slow;
   int dis_entrance=0;
   while (ptr1 != ptr2)
  {
      ptr1=ptr1->next;
      ptr2=ptr2->next;
      dis_entrance++;
  }

    return loop_point+dis_entrance;
}
```