leetcode总结无止境系列之链表

2014-09-03 17:01 1420人阅读 评论(0) 收藏[1] 举报

reference（*12*）

是面试中十分容易考到的题目，一般代码比较短，而且考查面试者的思维全面性和写无bug代码的能力。在写链表的题目时，建议画出示意图，并把头结点、尾节点这些特殊的结点考虑在内。

主要的技巧就是要用dummy head，就是在链表头加一个节点指向head，这样可以避免判断头指针，统一处理所有情况，最后返回dummy->next；另外有一个跟dummy head等价的方法是用指针的指针**p去处理，也能达到dummy head的效果，比dummy好的地方是，不用delete dummy head，另外如果一个节点比较大的话，用**p也比较省空间，但是逻辑上比dummy要绕一点.

```
1. //dummy head的使用示例
2. ListNode* ProcessList(ListNode* head){
3.     ListNode* dummy =  ListNode();
4.     dummy->next = head;
5.     //process the list
6.     head = dummy->next;
7.     delete dummy;//再提醒一次，请记得delete
8.     return head;
9. //指针的指针用法示例
10. ListNode* ProcessList(ListNode* head){
11.     ListNode** pNode = &head;
12.     //process the list
13.     //每次将处理得到的pCur连到新链表pNode上
14.     return head;
```

```
//dummy head的使用示例
ListNode* ProcessList(ListNode* head){
    ListNode* dummy = new ListNode();
    dummy->next = head;
    //process the list
    head = dummy->next;
    delete dummy;//再提醒一次，请记得delete
    return head;
}

//指针的指针用法示例
ListNode* ProcessList(ListNode* head){
    ListNode** pNode = &head;
    //process the list
    //每次将处理得到的pCur连到新链表pNode上
    return head;
```

}
链表常考题

、 给定单链表，检测。
思路：(快慢指针)使用两个指针p1,p2从链表头开始遍历，p1每次前进一步，p2每次前进两步。如果p2到达链表尾部，说明无环，否则p1、p2必然会在某个时刻相遇(p1==p2)，从而检测到链表中有环。

、 给定两个单链表(head1, head2)，检测两个链表是否有交点，如果有返回第一个交点。
思路：(两个单链表相交只可能为“X”型，因为单链表只有一个next指针)如果head1==head2，那么显然相交，直接返回head1。否则，分别从head1,head2开始遍历两个链表获得其长度len1与len2。假设len1>=len2，那么指针p1由head1开始向后 移动len1-len2步。指针p2=head2，下面p1、p2每次向后前进一步并比较p1p2是否相等，如果相等即返回该结点，否则说明两个链表没有 交点。

、 给定单链表(head)，如果有环的话请返回从头结点进入环的第一个节点。
思路：运用题一，我们可以检查链表中是否有环。如果有环，那么p1p2重合点p必然在环中。从p点断开环，方法为：p1=p, p2=p->next, p->next=NULL。此时，原单链表可以看作两条单链表，一条从head开始，另一条从p2开始，于是运用题二的方法，我们找到它们的第一个 交点即为所求。

、只给定单链表中某个结点p(并非最后一个结点，即p->next!=NULL)指针，该结点。
思路：办法很简单，首先是放p中数据,然后将p->next的数据copy入p中，接下来删除p->next即可。

、只给定单链表中某个结点p(非空结点)，在p前面一个结点。
思路：办法与前者类似，首先分配一个结点q，将q插入在p后，接下来将p中的数据copy入q中，然后再将要插入的数据记录在p中。

、给定单链表头结点，删除链表中倒数第k个结点。
思路：使用两个节点p1,p2，p1初始化指向头结点，p2一直指向p1后第k个节点，两个结点平行向后移动直到p2到达链表尾部(NULL),然后根据p1删除对应结点。

复杂链表复制
思路：复杂链表，其结点除了有一个m_pNext指针指向下一个结点外，还有一个m_pSibling指向链表中的任一结点或者NULL。第一步根据原始链表的每个结点N，创建对应的N’，把N’链接在N的后面；第二步是设置复制出来的链表上的结点的m_pSibling；第三步是把这个长链表拆分成两个：把奇数位置的结点链接起来就是原始链表，把偶数位置的结点链接出来就是复制出来的链表。

两个不交叉的有序链表的
思路：主要考察了merge的过程，和链表的操作，在leetcode总结无止境系列之排序[2]中有相关例题

链表翻转（包括全翻转，部分翻转，分段翻转）（递归或非递归实现）
思路：翻转过程一般需要三个结点指针：尾结点subTail（初始为待翻转序列的头结点，一步步翻转后最后成为新序列的尾结点）、待翻转结点pCur（其实就是尾结点的next结点）、头结点（每次翻转都需要更新）。具体也见下面的leetcode例题：Reverse Nodes in k-Group、Reverse Linked List II

实现的一种算法
思路：链表排序可用：归并、插入、冒泡及选择（单向链表）；归并、快排、插入、冒泡及选择（双向链表）；优先考虑归并，时间复杂度低，实现简单。

删除有序单链表中重复的元素
思路：删除重复主要有两种思路：一是考虑复制有用结点的value值，覆盖掉需删除的结点；二是直接操作链

表删除或者借助栈或set之类的容器来重组。

用链表模拟大整数加法运算
思路：要注意对进位的全面考虑

leetcode 上相关题目汇总

1.Add Two Numbers[3]

2.Swap Nodes in Pairs[4]

3.Partition List[5]

4.Remove Duplicates from Sorted List[6]

5.Remove Duplicates from Sorted List II[7]

6.Remove Nth Node From End of List[8]

7.Reverse Linked List II[9]

8.Reverse Nodes in k-Group[10]

Links

1. javascript:void(0);
2. http://cuijing.org/study/summary-of-sort-in-leetcode.html
3. https://github.com/missjing/leetcode/blob/master/Add%20Two%20Numbers.txt
4. https://github.com/missjing/leetcode/blob/master/Swap%20Nodes%20in%20Pairs.txt
5. https://github.com/missjing/leetcode/blob/master/Partition%20List.txt
6. https://github.com/missjing/leetcode/blob/master/Remove%20Duplicates%20from%20Sorted%20List.txt
7. https://github.com/missjing/leetcode/blob/master/Remove%20Duplicates%20from%20Sorted%20ListII.txt
8. https://github.com/missjing/leetcode/blob/master/Remove%20Nth%20Node%20From%20End%20of%20List.txt
9. https://github.com/missjing/leetcode/blob/master/Reverse%20Linked%20ListII.txt
10. https://github.com/missjing/leetcode/blob/master/Reverse%20Nodes%20in%20k-Group.txt