查阅更多的题解，请点击

Problem

19. Remove Nth Node From End of List(Medium)

Given a linked list, remove the n-th node from the end of list and return its head.

Example:

```
Given linked list: 1->2->3->4->5, and n = 2.

After removing the second node from the end, the linked list becomes 1->2->3->5.
```

Note:

Given n will always be valid.

Follow up:

Could you do this in one pass?

Solution

注意，题目要求在一次遍历，完成节点删除操作

那么意味着我们需要在一次遍历过程中找到需删除节点的前驱节点，有了该前驱节点，相当于找到了：

- 需删除节点
- 需删除节点的后驱

不妨设链表长度为l，则需要删除的节点为从前到后第l-n个节点
special case：

- 需删除的结点为头结点

O(n) time, O(n) space

最直观的想法，遍历链表时，将每一个节点存入数组中，那么在一次遍历结束后，可以很容易的在O(1)时间访问任一节点，这样就可以在一次遍历完成删除操作

```
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        vector<ListNode*> nodes;
        auto node=head;
        int length=0;
        while(node!=nullptr)
        {
            nodes.emplace_back(node);
            node=node->next;
            length++;
        }
        int pos=length-n;
        if(pos==0)
            return head->next;
        nodes[pos-1]->next=nodes[pos]->next;
        return head;
    }
};
```

O(n) time, O(1) space

在上述解法中，显然存储空间有极大的浪费。我们希望的是直接找到从前到后第l-n个节点，不妨设置slow和fast两个指向头结点的指针，fast先走n步，步长为1

- 此时可知fast还有l-n步走到链表尾部

然后slow和fast同时向后遍历，步长均为1，当fast到达尾部时，slow刚好指向第l-n个结点

由于希望找到第l-n个结点的前驱结点，则不妨创建一个新的结点newHead，newHead->next=head。初始化时slow=newHead, fast=head。这样就可以在一次遍历删除从后到前第n个结点

```
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        ListNode* newHead=new ListNode(-1);
        newHead->next=head;
        auto slow=newHead, fast=head;
        while(n--)
            fast=fast->next;
        while(fast!=nullptr)
        {
            slow=slow->next;
            fast=fast->next;
        }
        slow->next=slow->next->next;
        return newHead->next;
    }
};
```

GitHub传送门