# 题目

给定一个链表，删除链表的倒数第n个节点，并且返回链表的头结点。

**示例：**

给定一个链表: 1->2->3->4->5, 和 n = 2.

当删除了倒数第二个节点后，链表变为 1->2->3->5.

**说明：**

给定的n保证是有效的。

**进阶：**

你能尝试使用一趟扫描实现吗？

# 分析

## 方法一：快慢指针

思路：

代码：

class Solution {

public:

ListNode\* removeNthFromEnd(ListNode\* head, int n) {

if(!head || !head->next)

return NULL;

ListNode \*dummyNode = new ListNode(0);

dummyNode ->next = head;

ListNode \*fast = dummyNode, \* slow = dummyNode;

for(int i = 0; i <= n; i++)

fast = fast ->next;

while(f)

{

fast = fast ->next;

slow = slow->next;

}

slow ->next = slow ->next->next;

return dummyNode ->next;

}

};

或：

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution {

public:

ListNode\* removeNthFromEnd(ListNode\* head, int n) {

ListNode \*dummy = new ListNode(INT\_MAX); //哑结点

dummy->next = head;

ListNode \*p = dummy;

ListNode \*q = dummy; //快慢指针

ListNode \*tmp;

for(int i=0;i<n;i++)

{

p = p->next;

}

while(p->next)

{

p = p->next;

q = q->next;

}

tmp = q->next;

q->next = tmp->next;

delete tmp;

return dummy->next;

}

};

**复杂度分析：**

时间复杂度：O(L)，其中 L 是链表的长度。

空间复杂度：O(1)。