# 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：

1、哑结点（Dummy Node）：创建一个哑结点指向头结点，简化删除头结点的特殊情况处理。

2、快指针先行：快指针先移动n步，使得快慢指针之间保持n个节点的间隔。

3、同步移动：同时移动快慢指针，直到快指针到达链表末尾。此时慢指针指向待删除节点的前驱。

4、删除节点：调整指针指向，跳过待删除节点。

代码：

class Solution {

public:

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

ListNode\* dummy = new ListNode(0); // 类似暂存头节点

dummy->next = head;

ListNode\* fast = dummy;

ListNode\* slow = dummy;

// 快指针先移动n步

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

fast = fast->next;

}

// 同步移动，直到快指针到达末尾

while (fast->next != nullptr) {

fast = fast->next;

slow = slow->next;

}

// 删除倒数第n个节点

ListNode\* toDelete = slow->next;

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

delete toDelete; // 需要先初始化slow->next才能删除，不能颠倒顺序！

ListNode\* newHead = dummy->next;

delete dummy; // 删除临时构造的头节点，释放内存

return newHead;

}

};

或：

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(fast)

{

fast = fast ->next;

slow = slow->next;

}

slow ->next = slow ->next->next; // 这里应该待删除节点会内存泄漏？

return dummyNode ->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)。