# 题目

给定链表 head 和两个整数 m 和 n. 遍历该链表并按照如下方式删除节点:

开始时以头节点作为当前节点.

保留以当前节点开始的前 m 个节点.

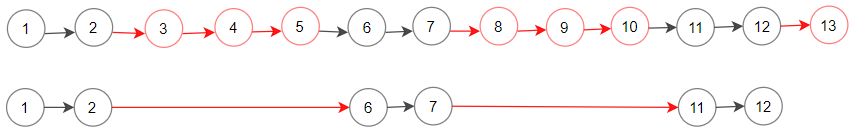
删除接下来的 n 个节点.

重复步骤 2 和 3, 直到到达链表结尾.

在删除了指定结点之后, 返回修改过后的链表的头节点.

进阶问题: 你能通过就地修改链表的方式解决这个问题吗?

示例 1:



输入: head = [1,2,3,4,5,6,7,8,9,10,11,12,13], m = 2, n = 3

输出: [1,2,6,7,11,12]

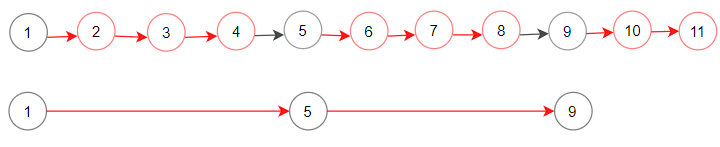
解析: 保留前(m = 2)个结点, 也就是以黑色节点表示的从链表头结点开始的结点(1 ->2).

删除接下来的(n = 3)个结点(3 -> 4 -> 5), 在图中以红色结点表示.

继续相同的操作, 直到链表的末尾.

返回删除结点之后的链表的头结点.

示例 2:



输入: head = [1,2,3,4,5,6,7,8,9,10,11], m = 1, n = 3

输出: [1,5,9]

解析: 返回删除结点之后的链表的头结点.

示例 3:

输入: head = [1,2,3,4,5,6,7,8,9,10,11], m = 3, n = 1

输出: [1,2,3,5,6,7,9,10,11]

示例 4:

输入: head = [9,3,7,7,9,10,8,2], m = 1, n = 2

输出: [9,7,8]

提示:

 1 <= 链表结点数 <= 10^4

[1 <= 链表的每一个结点值 <=10^6]

1 <= m,n <= 1000

# 分析

解题思路

思路其实很简单, 就从虚拟头节点开始，走m步，删 n个。主要问题是在于走m步的时候可能会走到尾，和删n个的时候可能还没删完就到尾了，所以要在走和删的时候判断是不是走到链表尾了。而且dummy head有个常见框架, 每次需要用dummy head的时候,我就先把这个写出来：

ListNode dummy = ListNode(0);

dummy.next = head;

ListNode \*cur = &dummy;

/\*

\* 中间一堆代码

\*/

return dummy.next;

**代码：**

class Solution {

public:

ListNode\* deleteNodes(ListNode\* head, int m, int n) {

ListNode dummy = ListNode(0);

dummy.next = head;

ListNode \*cur = &dummy;

while(cur->next != nullptr) {

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

// 没到 m 步走不动了就 break

if (cur != nullptr) cur = cur->next;

else break;

}

// 如果上面 break 了, 那后面也没什么好删的了, 直接 break

if (cur == nullptr) break;

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

// 没删到 n 个的时候没东西删了, 还是 break

if (cur->next != nullptr) cur->next = cur->next->next;

else break;

}

}

return dummy.next;

}

};

另一种写法：

基本就是创建一个dummy指针，令其next指向head。从dummy开始，往前找m个位置，如果存在，则逐个删除node，最多删除n个，有可能剩余不足n个，那么就删除所有剩余node。

class Solution {

public:

ListNode\* deleteNodes(ListNode\* head, int m, int n) {

ListNode\* dummy = new ListNode();

dummy->next = head;

auto cur = dummy;

while (cur) {

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

if (cur == nullptr) {

break;

}

cur = cur->next;

}

if (cur == nullptr) {

break;

}

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

if (cur->next == nullptr) {

break; // nothing left to be deleted

}

auto next =cur->next;

cur->next = cur->next->next;

delete next;

}

}

delete dummy;

return head;

}

};