题目：



解题思路一：

使用计数器，记录两个单向链表的长度分别为num1,num2;使长链表与短链表长度相同，即是减去长链表多余的部分，取num1-num2的绝对值，就是长链表长的部分，使指针走到这部分即可使两个链表长度一样。如果有交点，那么一定有headA=headB的过程，有则返回交点，无则返回nullptr

代码：

#include<iostream>

using namespace std;

struct ListNode {

int val;

ListNode \*next;

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

};

class Solution {

public:

ListNode\* getIntersectionNode(ListNode\* headA, ListNode\* headB) {

ListNode\* temp1 = headA;

ListNode\* temp2 = headB;

int num1 = 0,num2=0;//记录两个链表的长度

while(temp1)

{

num1++;

temp1 = temp1->next;

}

while (temp2)

{

num2++;

temp2 = temp2->next;

}

//使两个链表相等

if (num1 > num2)

{

int count = num1 - num2;

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

{

headA= headA->next;

}

while (headA != headB)

{

headA = headA->next;

headB = headB->next;

}

if (headA == headB)

return headA;

return nullptr;

}

else

{

int count = num2 - num1;

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

{

headB = headB->next;

}

while (headA != headB)

{

headA = headA->next;

headB = headB->next;

}

if (headA == headB)

return headA;

return nullptr;

}

}

};

解题思路2：

公式法：设链表headA的长度是a,链表headB的长度是b,它们的公共部分长度是c，那么到交点前，链表headA的长度是a-c,链表headB的长度是b-c;我们可以使用两个指针来进行遍历，指针A先遍历完链表headA再从头开始遍历headB，走过的距离是a+(b-c);指针B先遍历headB再遍历headA,到交点时，走过的距离是b+(a-c);这两个距离一定是相同的。如果c>0,那么返回的是相交的起点，如果C<0,那么没有交点，返回空指针

class Solution {

public:

ListNode \*getIntersectionNode(ListNode \*headA, ListNode \*headB) {

ListNode \*A = headA, \*B = headB;

while (A != B) {

A = A != nullptr ? A->next : headB;//先遍历A再从B开始遍历

B = B != nullptr ? B->next : headA;//先遍历B再开始从A开始遍历

}

return A;

}

};

解题思路三：

使用辅助栈：循环遍历链表存入栈，每次从栈顶元素开始比较，使用一个空指针ans存贮返回节点，循环比较栈顶元素，如果相等，当前栈顶赋值给ans，两个链表出栈，循环结束的条件是一个链表为空链表或者两个链表的栈顶元素不相等

代码：

class Solution {

public:

ListNode\* getIntersectionNode(ListNode\* headA, ListNode\* headB) {

stack<ListNode\*> s1, s2;

for (ListNode\* p = headA; p != nullptr; p = p->next) {

s1.push(p);

}//链表A入栈

for (ListNode\* p = headB; p != nullptr; p = p->next) {

s2.push(p);

}//链表B入栈

ListNode\* ans = nullptr;

while ( !s1.empty() && !s2.empty() && s1.top() == s2.top())

{

ans = s1.top();

s1.pop();

s2.pop();

}

return ans;

}

};