在链表问题中，由于链表长度需要一次遍历才能知道，对于一些应用例如求中点等问题来说会多一次遍历。所以有了双指针的使用场景。

双指针，顾名思义就是使用两个指针，又称为快慢指针。通常在链表问题中，快指针一次走两个结点，慢指针一次一个。这样二者走的距离差就是链表长度的一半。

对于求中点问题：当快指针走到末尾即为空时，慢指针刚好走到中点位置。

while fast and fast.next:

fast = fast.next.next

slow = slow.next

双指针还可以求链表中的环以及环的入口。为什么在slow和fast相遇后，从0和slow出发移动到相遇的点就是环的入口。假设环长为L，从起点到环的入口步数为a，从环的入口继续走b步到达相遇位置，从相遇位置继续走c步回到环的入口，则有b+c=L。并且可知，慢指针走了a+b步，快指针走了2(a+b)步，在相遇位置，快指针比慢指针夺走了若干圈，即2(a+b)=a+b+kL，得到a=kL-b=(k-1)L+c，可知，如果从起点出发慢指针走a步到达环的入口，而此时快指针已经走了k-1圈之后又走了c步，由于从相遇位置走c步即可回到环的入口，因此

也到达入口，两个指针在入口相遇

fast = head

slow = head

step = 0

while fast and fast.next:

step += 1

fast = fast.next.next

slow = slow.next

if fast == slow:

break

# 这里是判断指针出发过没有以及是否存在环

if fast != slow or step == 0:

return None

fast = head

while fast != slow:

fast = fast.next

slow = slow.next

return fast