# 题目

给定一个带有头结点 head 的非空单链表，返回链表的中间结点。

如果有两个中间结点，则返回第二个中间结点。

**示例 1：**

输入：[1,2,3,4,5]

输出：此列表中的结点 3 (序列化形式：[3,4,5])

返回的结点值为 3 。 (测评系统对该结点序列化表述是 [3,4,5])。

注意，我们返回了一个 ListNode 类型的对象 ans，这样：

ans.val = 3, ans.next.val = 4, ans.next.next.val = 5, 以及 ans.next.next.next = NULL.

**示例 2：**

输入：[1,2,3,4,5,6]

输出：此列表中的结点 4 (序列化形式：[4,5,6])

由于该列表有两个中间结点，值分别为 3 和 4，我们返回第二个结点。

**提示：**

给定链表的结点数介于 1 和 100 之间。

# 分析

## 方法一：数组

**思路：**

链表的缺点在于不能通过下标访问对应的元素。因此我们可以考虑对链表进行遍历，同时将遍历到的元素依次放入数组 A 中。如果我们遍历到了 N 个元素，那么链表以及数组的长度也为 N，对应的中间节点即为 A[N/2]。

**代码：**

class Solution {

public:

ListNode\* middleNode(ListNode\* head) {

vector<ListNode\*> A = {head};

while (A.back()->next != NULL)

A.push\_back(A.back()->next);

return A[A.size() / 2];

}

};

复杂度分析

时间复杂度：O(N)，其中N是给定链表中的结点数目。

空间复杂度：O(N)，即数组A用去的空间。

## 方法二：单指针

**思路：**

我们可以对方法一进行空间优化，省去数组 A。

我们可以对链表进行两次遍历。第一次遍历时，我们统计链表中的元素个数 N；第二次遍历时，我们遍历到第 N/2 个元素（链表的首节点为第 0 个元素）时，将该元素返回即可。

**代码：**

class Solution {

public:

ListNode\* middleNode(ListNode\* head) {

int n = 0;

ListNode\* cur = head;

while (cur != nullptr) {

++n;

cur = cur->next;

}

int k = 0;

cur = head;

while (k < n / 2) {

++k;

cur = cur->next;

}

return cur;

}

};

复杂度分析

时间复杂度：O(N)，其中N是给定链表的结点数目。

空间复杂度：O(1)，只需要常数空间存放变量和指针。

## 方法三：双指针

**思路：**

我们可以继续优化方法二，用两个指针slow与fast一起遍历链表。slow一次走一步，fast一次走两步。那么当fast到达链表的末尾时，slow必然位于中间。

**说明：**这种思路非常好，需要掌握！

**代码：**

class Solution {

public:

ListNode\* middleNode(ListNode\* head) {

ListNode\* slow = head;

ListNode\* fast = head;

//不是head->next，快慢指针起点相同，步长不同

while (fast != NULL && fast->next != NULL) {

slow = slow->next;

fast = fast->next->next;

}

return slow;

}

};

复杂度分析

时间复杂度：O(N)，其中N是给定链表的结点数目。

空间复杂度：O(1)，只需要常数空间存放slow和fast两个指针。