# 题目

给定单向链表的头指针和一个要删除的节点的值，定义一个函数删除该节点。

返回删除后的链表的头节点。

注意：此题对比原题有改动

**示例 1:**

**输入:** head = [4,5,1,9], val = 5

**输出:** [4,1,9]

**解释:** 给定你链表中值为 5 的第二个节点，那么在调用了你的函数之后，该链表应变为 4 -> 1 -> 9.

**示例 2:**

**输入:** head = [4,5,1,9], val = 1

**输出:** [4,5,9]

**解释:** 给定你链表中值为 1 的第三个节点，那么在调用了你的函数之后，该链表应变为 4 -> 5 -> 9.

**说明：**

题目保证链表中节点的值互不相同。

若使用C或C++语言，你不需要free或delete被删除的节点。

# 分析

**思路：**

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

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

\* };

\*/

class Solution {

public:

ListNode\* deleteNode(ListNode\* head, int val) {

ListNode\*pre=head;

ListNode\*curr=head->next;

if(head->val==val)return curr;

while(curr!=NULL){

if(curr->val==val){

pre->next=curr->next;

return head;

}

else{

curr=curr->next; //pre和cur指针向后移动

pre=pre->next;

}

}

return head;

}

};

另一种写法：

class Solution {

public:

ListNode\* deleteNode(ListNode\* head, int val) {

ListNode\* dummy=new ListNode(-1,head);

ListNode\* cur=dummy;

while(cur->next)//由于即使head为空，cur初始值也不为空，所以该循环不会报错。

{

if(cur->next->val==val)

{

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

/\*找到需要删除的节点后，进行常规删除操作。

\*此时cur->next需要被删除，所以只需改变cur节点的next指针即可。\*/

break;

//只删除一个节点，找到删除该节点之后，无需再遍历，节省时间

}

cur=cur->next;//cur节点要不断往后走，才能进行链表的遍历。

}

return dummy->next;

}

};

**算法复杂度：**  
     由于只进行了链表的一次遍历

时间复杂度为：O(n)

空间复杂度为：O(1)。