# 题目

给定一个链表和一个特定值 x，对链表进行分隔，使得所有小于 x 的节点都在大于或等于 x 的节点之前。

你应当保留两个分区中每个节点的初始相对位置。

**示例:**

输入: head = 1->4->3->2->5->2, x = 3

输出: 1->2->2->4->3->5

# 分析

## 方法一：双指针

## 方法二：两链表拼接

**思路：**（推荐这种做法）

直观来说我们只需维护两个链表small和large即可，small链表按顺序存储所有小于x的节点，large链表按顺序存储所有大于等于x的节点。遍历完原链表后，我们只要将small链表尾节点指向large链表的头节点即能完成对链表的分隔。

为了实现上述思路，我们设smallHead和largeHead分别为两个链表的哑节点，即它们的next指针指向链表的头节点，这样做的目的是为了更方便地处理头节点为空的边界条件。同时设 small和large节点指向当前链表的末尾节点。开始时smallHead=small，largeHead=large。随后，从前往后遍历链表，判断当前链表的节点值是否小于x，如果小于就将small的next指针指向该节点，否则将large的next指针指向该节点。

遍历结束后，我们将large的next指针置空，这是因为当前节点复用的是原链表的节点，而其next指针可能指向一个小于x的节点，我们需要切断这个引用。同时将small的next 指针指向largeHead的next指针指向的节点，即真正意义上的large链表的头节点。最后返回smallHead的next指针即为我们要求的答案。

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

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

\* };

\*/

class Solution {

public:

ListNode\* partition(ListNode\* head, int x) {

ListNode \*less = new ListNode(-1);

ListNode \*more = new ListNode(-1);

ListNode \*tmpLess = less;

ListNode \*tmpMore = more;

while(head)

{

if(head->val < x)

{

less->next = head;

less = less->next;

}

else

{

more->next = head;

more = more->next;

}

head = head->next;

}

more->next = NULL; //需要单独处理一下尾指针

less->next = tmpMore->next;

return tmpLess->next; //返回前面暂存的头指针

}

};

**复杂度分析：**

时间复杂度：O(n)，其中n是原链表的长度。我们对该链表进行了一次遍历。

空间复杂度：O(1)。