链表划分Leetcode\_86. PartitionList\_Medium

# 分割链表Leetcode\_86. PartitionList\_Medium

## 题目介绍

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Leetcode\_86\_PartitionList\_Medium\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\* 题目介绍：Leetcode\_86\_PartitionList\_Medium

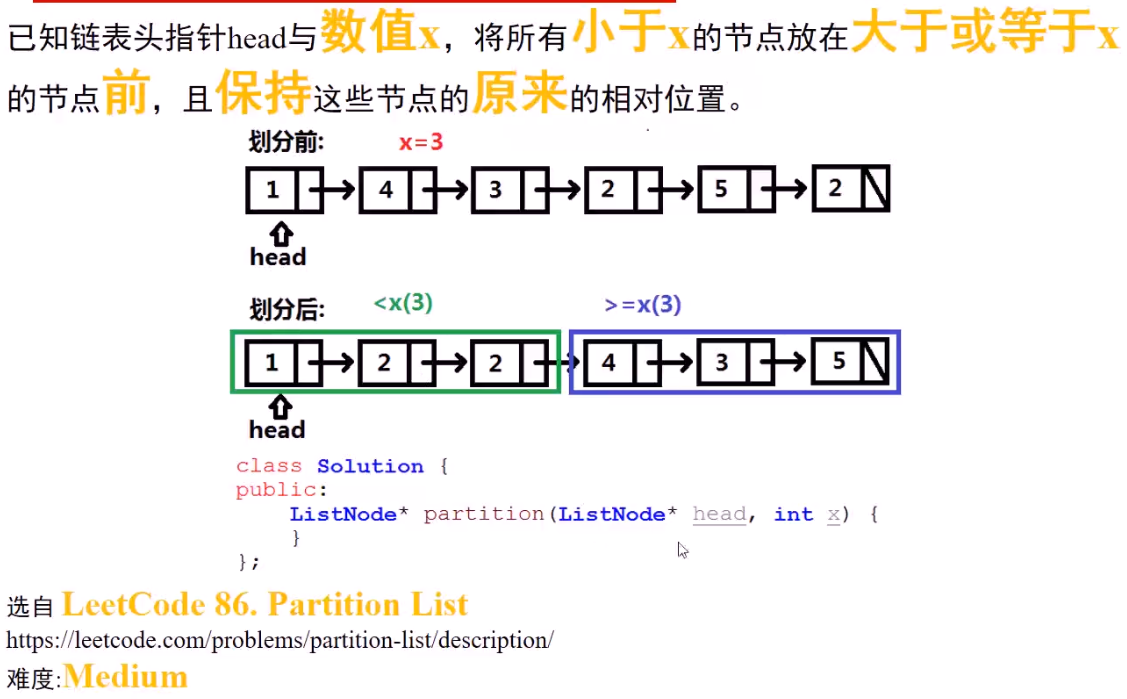
\* 难度：Medium

\* Given a linked list and a value x, partition it such that all nodes

\* less than x come before nodes greater than or equal to x.

\* You should preserve the original relative order of the nodes

\* in each of the two partitions.



## 思路分析

\* 思路分析： 将原链表根据x大小分割成两个子链表smaller和bigger；

\* 然后将smaller的尾节点指向bigger的头节点，

\* 将bigger的尾节点指向null即可。

\* 这里需要注意：smaller和bigger子链表的头节点需要手动初始化节点。

技巧：巧用临时头节点。

## Java代码

// Definition for singly-linked list.

public class ListNode {

int val;

ListNode next;

ListNode(int x) {

val = x;

}

}

/\*\*

\* partition方法

\*/

public ListNode partition(ListNode head, int x) {

if (head == null) return null;

ListNode temp = head;//保护head引用

ListNode smallerHead = new ListNode(0);

ListNode smallerTemp = smallerHead;

ListNode biggerHead = new ListNode(0);

ListNode biggerTemp = biggerHead;

while (temp != null) {//遍历原链表

if (temp.val < x) {

smallerTemp.next = temp;

smallerTemp = smallerTemp.next;

} else {

biggerTemp.next = temp;

biggerTemp = biggerTemp.next;

}

temp = temp.next;//前进一步

}

smallerTemp.next = biggerHead.next;//连接两个子链表

biggerTemp.next = null;//将尾节点置空

return smallerHead.next;

}

