Partition List

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.

For example,
Given `1->4->3->2->5->2` and $x = 3$,
return `1->2->2->4->3->5`.

Solution 1

```
ListNode *partition(ListNode *head, int x) {
    ListNode node1(0), node2(0);
    ListNode *p1 = &node1, *p2 = &node2;
    while (head) {
        if (head->val < x)
            p1 = p1->next = head;
        else
            p2 = p2->next = head;
        head = head->next;
    }
    p2->next = NULL;
    p1->next = node2.next;
    return node1.next;
}
```

written by shichaotan original link here

Solution 2

the basic idea is to maintain two queues, the first one stores all nodes with val less than x , and the second queue stores all the rest nodes. Then concat these two queues. Remember to set the tail of second queue a null next, or u will get TLE.

```
public ListNode partition(ListNode head, int x) {
    ListNode dummy1 = new ListNode(0), dummy2 = new ListNode(0);  //dummy heads o
f the 1st and 2nd queues
    ListNode curr1 = dummy1, curr2 = dummy2;      //current tails of the two queu
es;
    while (head!=null){
        if (head.val<x) {
            curr1.next = head;
            curr1 = head;
        }else {
            curr2.next = head;
            curr2 = head;
        }
        head = head.next;
    }
    curr2.next = null;          //important! avoid cycle in linked list. otherwis
e u will get TLE.
    curr1.next = dummy2.next;
    return dummy1.next;
}
```

written by cbmbbz original link here

Solution 3

```
class Solution {
public:
    ListNode* partition(ListNode* head, int x) {
        ListNode left(0), right(0);
        ListNode *l = &left, *r = &right;

        while(head){
            ListNode* & ref = head->val < x ? l : r;
            ref->next = head;
            ref = ref->next;

            head = head->next;
        }
        l->next = right.next;
        r->next = NULL;
        return left.next;
    }
};
```

written by yeze322 original link here