86. Partition List ★

Question | Editorial Solution | My Submissions (/problems/partition-list/submissions/)

Total Accepted: **77206** Total Submissions: **250942** Difficulty: **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.

For example,

Given `1->4->3->2->5->2` and *x* = 3,

return `1->2->2->4->3->5` .

Subscribe (/subscribe/) to see which companies asked this question

Show Tags

Have you met this question in a real interview? Yes No

Discuss (https://leetcode.com/discuss/questions/oj/partition-list)

Pick One (/problems/random-one-question/)

C++

```
/**
 * 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* leftRoot = NULL, *rightRoot = NULL;
        ListNode* leftCurNode = NULL, *rightCurNode = NULL;
        ListNode* curNode = head;
        while (curNode != NULL) {
            if (curNode->val < x) {
                if (leftRoot) {
                    leftCurNode->next = curNode;
                    leftCurNode = curNode;
                }
                else {
                    leftRoot = curNode;
                    leftCurNode = leftRoot;
                }
```

```
            }
            else {
                if (rightRoot) {
                    rightCurNode->next = curNode;
                    rightCurNode = curNode;
                }
                else {
                    rightRoot = curNode;
                    rightCurNode = rightRoot;
                }
            }
            curNode = curNode->next;
        }
        if (leftRoot) {
            head = leftRoot;
            leftCurNode->next = rightRoot;
            if(rightRoot)
                rightCurNode->next = NULL;
        }
        else if(rightRoot){
            head = rightRoot;
            rightCurNode->next = NULL;
        }
        else {
            head = NULL;
        }
        return head;
    }
};
```

Notes

Custom Testcase

Run Code | Submit Solution