Description | Editorial | Solutions (4K) | Submissions

86. Partition List

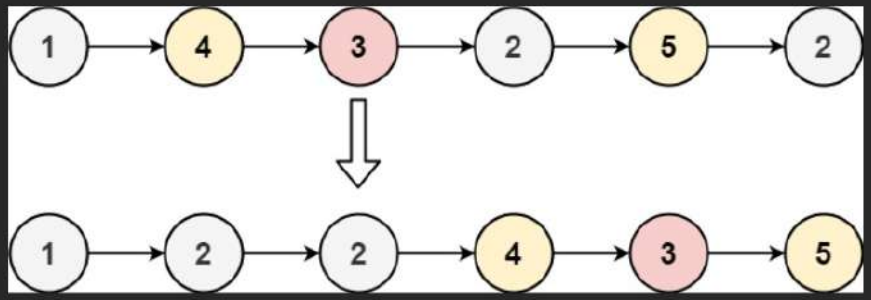
Medium 6.4K 713

Companies

Given the `head` of 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.

Example 1:




```
Input: head = [1,4,3,2,5,2], x = 3
Output: [1,2,2,4,3,5]
```

Example 2:

```
Input: head = [2,1], x = 2
Output: [1,2]
```

Java | Auto

```java
public class Solution {
    public ListNode partition(ListNode head, int x) {
        ListNode before = new ListNode(0);
        ListNode after = new ListNode(0);
        ListNode before_curr = before;
        ListNode after_curr = after;

        while(head != null) {
            if(head.val < x) {
                before_curr.next = head;
                before_curr = before_curr.next;
            } else {
                after_curr.next = head;
                after_curr = after_curr.next;
            }
            head = head.next;
```

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

head =
[1,4,3,2,5,2]

x =
3

Output

[1,2,2,4,3,5]

Console | Run | Submit

Description | Editorial | Solutions (5.8K) | Submissions

38. Count and Say

Hint

Medium 3.4K 7.1K

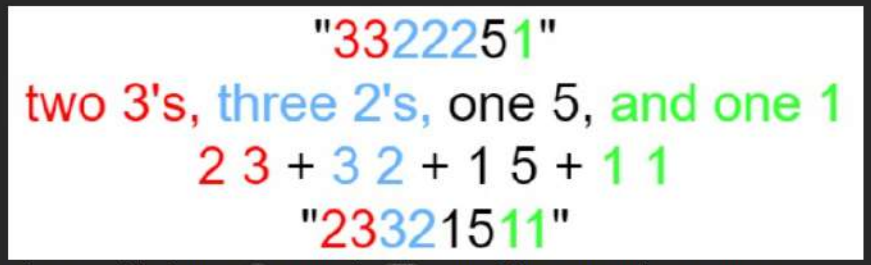
Companies

The **count-and-say** sequence is a sequence of digit strings defined by the recursive formula:

- `countAndSay(1) = "1"`
- `countAndSay(n)` is the way you would "say" the digit string from `countAndSay(n-1)`, which is then converted into a different digit string.

To determine how you "say" a digit string, split it into the **minimal** number of substrings such that each substring contains exactly **one** unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.

For example, the saying and conversion for digit string `"3322251"`:



Given a positive integer `n`, return *the* n^{th} *term of the* ***count-and-say*** *sequence.*

Example 1:

Input: n = 1

Java | Auto

```java
class Solution {
    public String countAndSay(int n) {
        if(n==1)
        {
            return "1";
        }
        if(n==2)
        {
            return "11";
        }
        StringBuilder nm=new StringBuilder();
        nm.append("11");
        for(int i=3;i<=n;i++)
        {
            StringBuilder kk=new StringBuilder();
            task(nm.toString() kk);
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

n =
1

Output

"1"

Expected

Console | Run | Submit