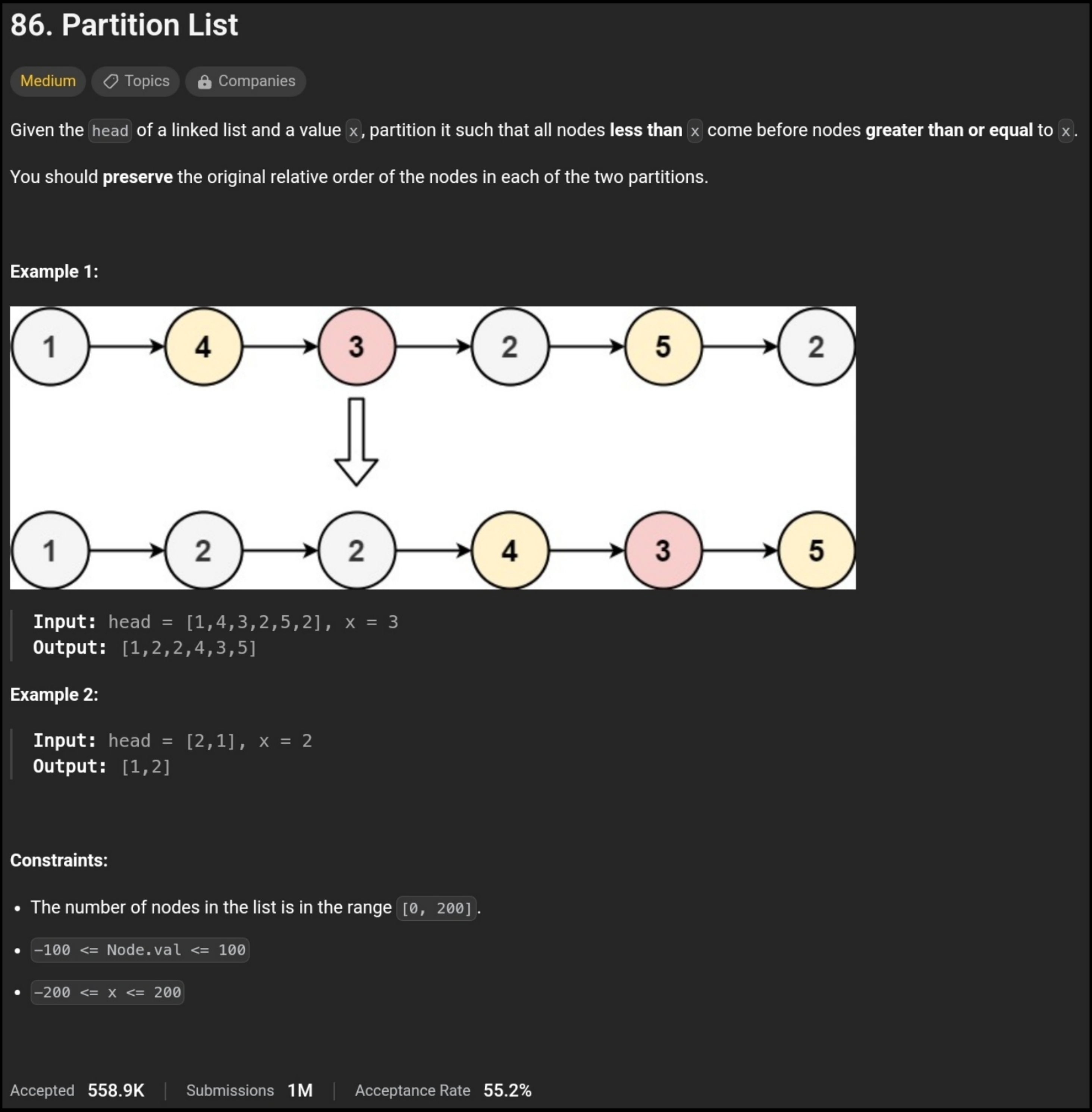


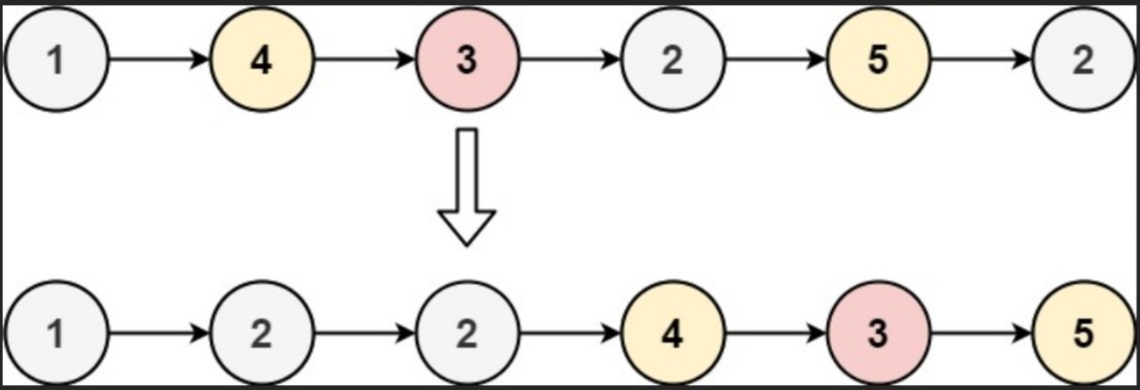
86. Partition List

Medium | Topics | 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]
```

Constraints:

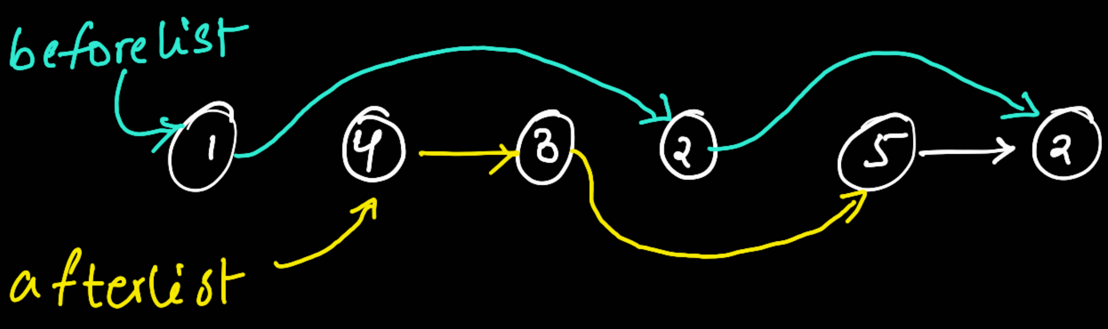
- The number of nodes in the list is in the range `[0, 200]`.
- `-100 <= Node.val <= 100`
- `-200 <= x <= 200`

Accepted 558.9K | Submissions 1M | Acceptance Rate 55.2%

Approach 1: kind of sewing the nodes

→ Create two nodes : beforelist , afterlist
→ Use beforelist to sew all the nodes that are less than x.
→ Use afterlist to sew all the nodes that are greater than x.
→ now link beforelist tail to the afterlist head

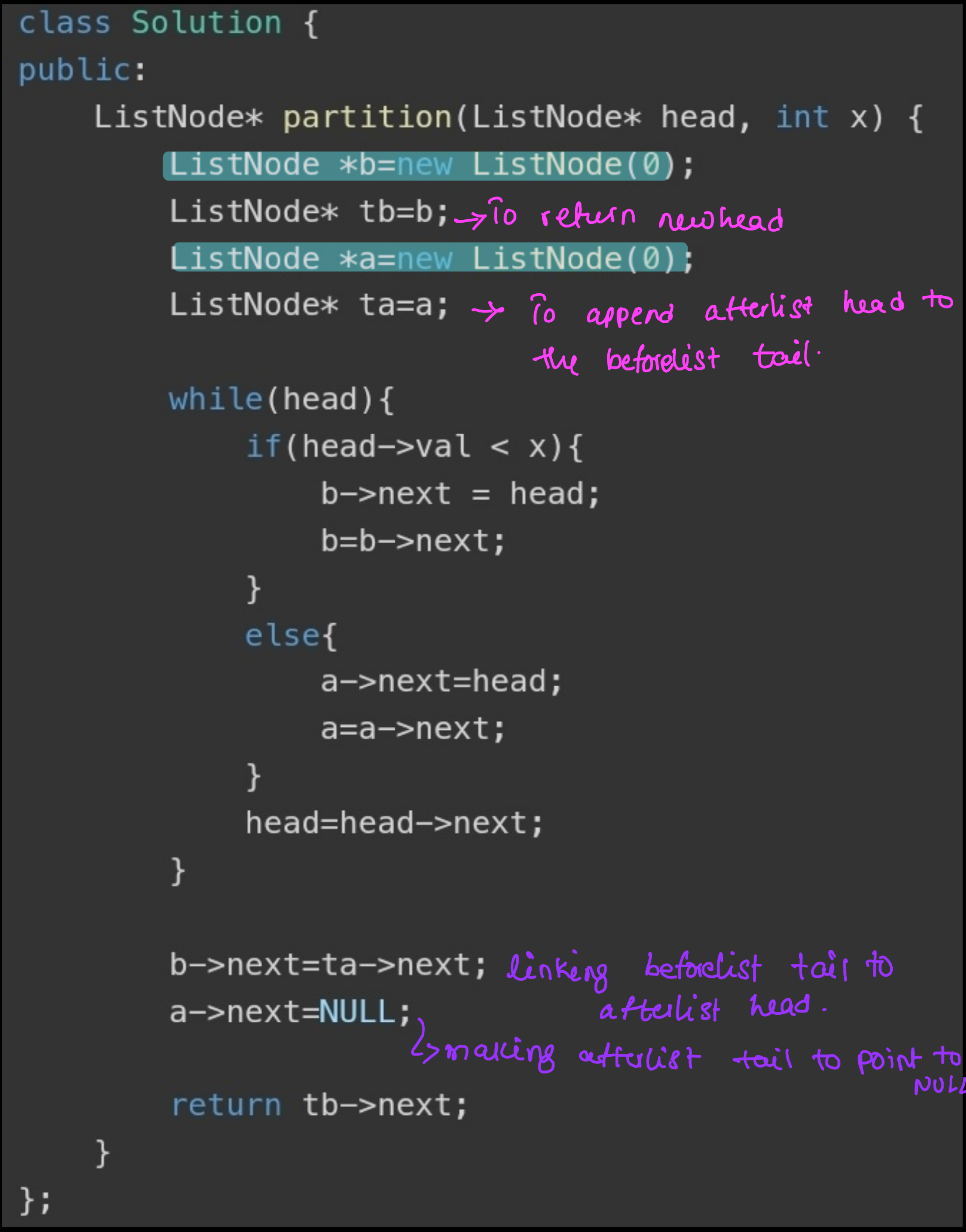
$$\begin{tikzcd} 1 \arrow[r] & 4 \arrow[r] & 3 \arrow[r] & 2 \arrow[r] & 5 \arrow[r] & 2 \end{tikzcd}$$



now make afterlist tail to point to NULL.
and point beforelist tail to afterlist head

$$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] & 2 \arrow[r] & 4 \arrow[r] & 3 \arrow[r] & 5 \end{tikzcd}$$

This is the required state of list



```
class Solution {
public:
    ListNode* partition(ListNode* head, int x) {
        ListNode *b=new ListNode(0);
        ListNode* tb=b; → To return newhead
        ListNode *a=new ListNode(0);
        ListNode* ta=a; → To append afterlist head to
                          the beforelist tail.

        while(head){
            if(head->val < x){
                b->next = head;
                b=b->next;
            }
            else{
                a->next=head;
                a=a->next;
            }
            head=head->next;
        }

        b->next=ta->next; linking beforelist tail to
                                afterlist head.
        a->next=NULL;
                    ↳ making afterlist tail to point to
                                                NULL.

        return tb->next;
    }
};
```

$T(n): O(n)$

$S(n): O(2)$