203. Remove Linked List Elements

Sept. 27, 2019 | 16.5K views

Average Rating: 4.94 (18 votes)

Remove all elements from a linked list of integers that have value ***val***.

Example:

```
Input:  1->2->6->3->4->5->6, val = 6
Output: 1->2->3->4->5
```

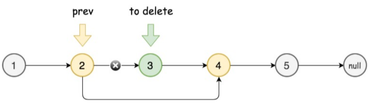
Solution

Approach 1: Sentinel Node

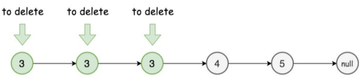
Intuition

The problem seems to be very easy if one has to delete a node in the middle:

- Pick the node-predecessor `prev` of the node to delete.
- Set its next pointer to point to the node next to the one to delete.



The things are more complicated when the node or nodes to delete are in the head of linked list.

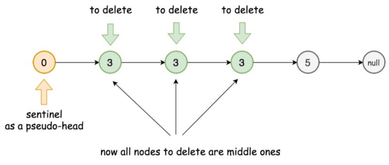


> How to deal with that? To reduce the problem to the deletion of middle nodes with the help of sentinel node.

Sentinel nodes are widely used in trees and linked lists as pseudo-heads, pseudo-tails, markers of level end, etc. They are purely functional, and usually does not hold any data. Their main purpose is to standardize the situation, for example, make linked list to be never empty and never headless and hence simplify insert and delete.

Here are two standard examples:

- LRU Cache, sentinel nodes are used as pseudo-head and pseudo-tail.
- Tree Level Order Traversal, sentinel nodes are used to mark level end.



Here sentinel node will be used as pseudo-head.

Algorithm

- Initiate sentinel node as `ListNode(0)` and set it to be the new head: `sentinel.next = head`.
- Initiate two pointers to track the current node and its predecessor: `curr` and `prev`.
- While `curr` is not a null pointer:
  - Compare the value of the current node with the value to delete.
    - In the values are equal, delete the current node: `prev.next = curr.next`.
    - Otherwise, set predecessor to be equal to the current node.
  - Move to the next node: `curr = curr.next`.
- Return `sentinel.next`.

Implementation

C++ | Java | Python — Copy

```
class Solution {
  public:
  ListNode* removeElements(ListNode* head, int val) {
    ListNode* sentinel = new ListNode(0);
    sentinel->next = head;

    ListNode *prev = sentinel, *curr = head, *toDelete = nullptr;
    while (curr != nullptr) {
      if (curr->val == val) {
        prev->next = curr->next;
        toDelete = curr;
      } else prev = curr;

      curr = curr->next;

      if (toDelete != nullptr) {
        delete toDelete;
        toDelete = nullptr;
      }
    }

    ListNode *ret = sentinel->next;
    delete sentinel;
    return ret;
  }
};
```

Complexity Analysis

- Time complexity: $\mathcal{O}(N)$, it's one pass solution.
- Space complexity: $\mathcal{O}(1)$, it's a constant space solution.

Rate this article:

Comments: 7

Sort By

Type comment here... (Markdown is supported)

Preview Post

Mrmagician ★325 March 29, 2020 4:48 PM

My recursive and iterative solutions

```
def recursive(self, head, val):
        def rec(node, val):
            if node:
```

Read More

4 Share Reply

SHOW 1 REPLY

sajeeree ★2 February 11, 2020 1:50 AM

What is the problem with this approach

```
class Solution {
public:
ListNode* removeElements(ListNode* head, int val) {
```

Read More

1 Share Reply

SHOW 1 REPLY

VickyFengYu ★54 January 29, 2020 6:59 PM Report

can use fewer variables:

```
public ListNode removeElements(ListNode head, int val) {
    ListNode dummy = new ListNode(0);
```

Read More

Vote Up

1 Share Reply

SHOW 1 REPLY

icebear1215 ★10 November 12, 2019 1:59 AM

I feel the removed nodes' next still point to some nodes in the list...
For example, [1,2,6,3,4,5,6] , after removing all 6, the first 6 still points to 3

1 Share Reply

SHOW 4 REPLIES

asadm ★0 February 4, 2020 1:09 PM

Why can't you return "head" at the end instead of sentinel? Doesn't that point to the same thing as sentinal.next?

0 Share Reply

SHOW 1 REPLY

sriharik ★151 May 14, 2020 9:32 AM Report

I ended up coming up with 3 solutions.
Brute Force - Just make a list and skip the val elements, then rebuild the list. This is a good starting point for brute force.

```
public ListNode removeElements(ListNode head, int val) {
```

Read More

0 Share Reply

Swallow_Liu ★6 May 3, 2020 8:53 AM

in the solution code, why we have to return sentinel? it seems sentinel has never been changed since it was assigned in the first two lines (i understand the while loop part, for me, it seems like only prev and curr are. changing inside the while loop)

0 Share Reply

SHOW 1 REPLY