**19. Remove Nth Node From End of List**

<https://leetcode.com/problems/remove-nth-node-from-end-of-list/>

1. **Listen**

**Problem Description:**

Given the **head** of a **linked list**, **remove the nth node from the end of the list** and **return its head**.

**Input**:

**head** of a singly **linked list**

**n** is the **nth**nodefrom the right side of the list to remove

**Goal**:

Removethe **nth**node **from the** **end** of the list

**Return**:

head of list with **nth**node **from the** **end** removed

1. **Example**

**Constraints:**

* The number of nodes in the list is **size**.
  + 1 <= size <= 30
* The value of the **nth**node is n.
  + 1 <= n <= size
* 0 <= Node.val <= 100

**Test Cases:**

* linked list is even length
* linked list is odd length

**Edge Cases:**

* **nth**node is beginning node
* **nth**node is end node
* linked list is empty
* linked list has single node

**Questions & Assumptions:**

* What should be returned with an empty list?

**Example 1**

Diagram

Description automatically generated

**Input:** head = [1,2,3,4,5], n = 2

**Output:** [1,2,3,5]

1. **Brute Force**

**Solution 1:**

Traverse fast pointer to nth node from the left side of the list.

Now traverse slow pointer starting from head and fast pointer one node at a time.

When fast pointer reaches null, the slow pointer will be at the n+1 node from the right side of the list.

When we move the fast pointer forward n nodes, we maintain this gap by having the slow and fast pointer moving the same speed until they reach the end of the list.

We can accomplish this in O(n) time and O(1) space.

1. **Optimize**

A picture containing table

Description automatically generated

1. **Implement**

public ListNode removeNthFromEnd(ListNode head, int n) {

ListNode fast = head;

ListNode slow = head;

for(int i = 0; i < n; i++)

{

fast = fast.next;

}

while(fast != null)

{

slow = slow.next;

fast = fast.next;

}

slow.next = slow.next.next;

return head;

}

1. **Test**

This code does not work when

the size of the list is <= 2

or

if we try to remove the first node

We can fix this by using a dummy head node to avoid these edge cases.

public ListNode removeNthFromEnd(ListNode head, int n) {

ListNode dummy = new ListNode();

dummy.next = head;

ListNode fast = dummy, slow = dummy;

for(int i = 0; i < n+1; i++) fast = fast.next;

while(fast != null)

{

slow = slow.next;

fast = fast.next;

}

slow.next = slow.next.next;

return dummy.next;

}