

# ✂️ ■ Remove Nth Node From End

One-pass deletion using a gap of N between two pointers

● Linked List

■ MEDIUM

## ■ Intuition First

To find the Nth from the end in one pass: put two pointers N apart. When the front pointer reaches the end, the back pointer is exactly at the node to delete. It's like two soldiers marching — one starts N steps ahead. When the leader exits, the follower is at the target.

```
List: 1 → 2 → 3 → 4 → 5, N=2
Remove 2nd from end = node 4

Use dummy head to handle edge cases
dummy → 1 → 2 → 3 → 4 → 5
    fast                               (advance fast N+1=3 steps)
    fast moves: dummy→1→2→3
    dummy → 1 → 2 → 3 → 4 → 5
    slow                fast

Move both until fast==null:
    slow → 3, fast → null

slow.next = slow.next.next (skip node 4)
Result: 1 → 2 → 3 → 5 ✓
```

## ■ Java Code

```
1  class Solution {
2  public ListNode removeNthFromEnd(ListNode head, int n) {
3      ListNode dummy = new ListNode(0);
4      dummy.next = head;
5      ListNode slow = dummy, fast = dummy;
6
7      // Move fast N+1 steps ahead
8      for (int i = 0; i <= n; i++) {
9          fast = fast.next;
10     }
11     // Move both until fast is null
12     while (fast != null) {
13         slow = slow.next;
14         fast = fast.next;
15     }
16     // Delete the nth node from end
17     slow.next = slow.next.next;
```

```
18     return dummy.next;
19 }
20 }
```

## ■ Complexity

| Type  | Value  | Meaning                   |
|-------|--------|---------------------------|
| Time  | $O(n)$ | One pass through the list |
| Space | $O(1)$ | Only pointer variables    |

### ■ Why N+1 not N?

We want slow to stop at the node BEFORE the one to delete (so we can do `slow.next=slow.next.next`). Moving fast N+1 steps (not N) achieves this gap correctly.

## ■ Quick Revision

|   |                                                            |
|---|------------------------------------------------------------|
| ✓ | Dummy head handles removing the first node edge case       |
| ✓ | Move fast N+1 steps ahead of slow                          |
| ✓ | Move both together until <code>fast==null</code>           |
| ✓ | <code>slow.next = slow.next.next</code> removes the target |
| ✓ | Time $O(n)$ , Space $O(1)$                                 |