

Problem 19: Remove Nth Node From End of List

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given the

head

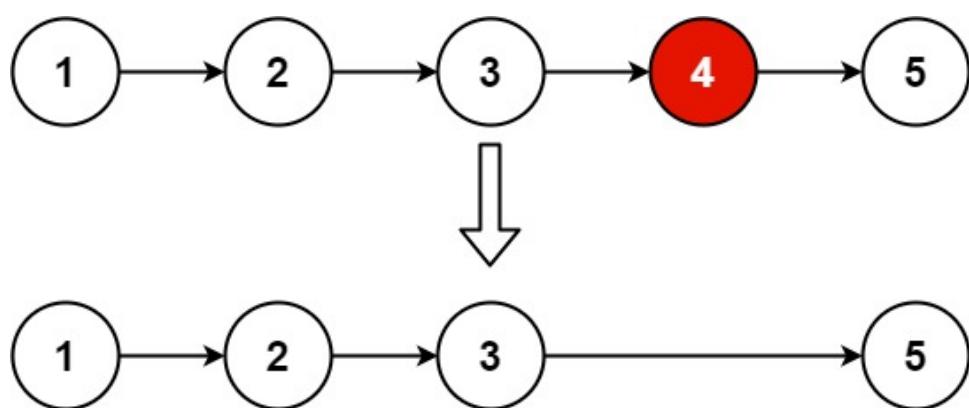
of a linked list, remove the

n

th

node from the end of the list and return its head.

Example 1:



Input:

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

Output:

[1,2,3,5]

Example 2:

Input:

head = [1], n = 1

Output:

[]

Example 3:

Input:

head = [1,2], n = 1

Output:

[1]

Constraints:

The number of nodes in the list is

sz

.

$1 \leq sz \leq 30$

$0 \leq Node.val \leq 100$

$1 \leq n \leq sz$

Follow up:

Could you do this in one pass?

Code Snippets

C++:

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {

public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {

    }
};
```

Java:

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {

    public ListNode removeNthFromEnd(ListNode head, int n) {

    }
}
```

```
}
```

Python3:

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, val=0, next=None):
#     self.val = val
#     self.next = next
class Solution:
    def removeNthFromEnd(self, head: Optional[ListNode], n: int) ->
        Optional[ListNode]:
```

Python:

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution(object):
    def removeNthFromEnd(self, head, n):
        """
        :type head: Optional[ListNode]
        :type n: int
        :rtype: Optional[ListNode]
        """
```

JavaScript:

```
/**
 * Definition for singly-linked list.
 * function ListNode(val, next) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.next = (next===undefined ? null : next)
 * }
 */
/**
 * @param {ListNode} head
 * @param {number} n
 * @return {ListNode}
 */
```

```
var removeNthFromEnd = function(head, n) {  
};
```

TypeScript:

```
/**  
 * Definition for singly-linked list.  
 * class ListNode {  
 * val: number  
 * next: ListNode | null  
 * constructor(val?: number, next?: ListNode | null) {  
 * this.val = (val==undefined ? 0 : val)  
 * this.next = (next==undefined ? null : next)  
 * }  
 * }  
 */  
  
function removeNthFromEnd(head: ListNode | null, n: number): ListNode | null  
{  
  
};
```

C#:

```
/**  
 * Definition for singly-linked list.  
 * public class ListNode {  
 * public int val;  
 * public ListNode next;  
 * public ListNode(int val=0, ListNode next=null) {  
 * this.val = val;  
 * this.next = next;  
 * }  
 * }  
 */  
  
public class Solution {  
public ListNode RemoveNthFromEnd(ListNode head, int n) {  
  
}  
}
```

C:

```
/**  
 * Definition for singly-linked list.  
 * struct ListNode {  
 *     int val;  
 *     struct ListNode *next;  
 * };  
 */  
struct ListNode* removeNthFromEnd(struct ListNode* head, int n) {  
  
}
```

Go:

```
/**  
 * Definition for singly-linked list.  
 * type ListNode struct {  
 *     Val int  
 *     Next *ListNode  
 * }  
 */  
func removeNthFromEnd(head *ListNode, n int) *ListNode {  
  
}
```

Kotlin:

```
/**  
 * Example:  
 * var li = ListNode(5)  
 * var v = li.`val`  
 * Definition for singly-linked list.  
 * class ListNode(var `val`: Int) {  
 *     var next: ListNode? = null  
 * }  
 */  
class Solution {  
    fun removeNthFromEnd(head: ListNode?, n: Int): ListNode? {  
  
    }  
}
```

Swift:

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     public var val: Int
 *     public var next: ListNode?
 *     public init() { self.val = 0; self.next = nil; }
 *     public init(_ val: Int) { self.val = val; self.next = nil; }
 *     public init(_ val: Int, _ next: ListNode?) { self.val = val; self.next =
 * next; }
 * }
 */
class Solution {
    func removeNthFromEnd(_ head: ListNode?, _ n: Int) -> ListNode? {
}
}
```

Rust:

```
// Definition for singly-linked list.
// #[derive(PartialEq, Eq, Clone, Debug)]
// pub struct ListNode {
//     pub val: i32,
//     pub next: Option<Box<ListNode>>
// }
//
// impl ListNode {
//     #[inline]
//     fn new(val: i32) -> Self {
//         ListNode {
//             next: None,
//             val
//         }
//     }
// }
//
impl Solution {
    pub fn remove_nth_from_end(head: Option<Box<ListNode>>, n: i32) ->
        Option<Box<ListNode>> {
}
}
```

Ruby:

```
# Definition for singly-linked list.

# class ListNode
# attr_accessor :val, :next
# def initialize(val = 0, _next = nil)
#   @val = val
#   @next = _next
# end
# end

# @param {ListNode} head
# @param {Integer} n
# @return {ListNode}
def remove_nth_from_end(head, n)

end
```

PHP:

```
/***
 * Definition for a singly-linked list.
 * class ListNode {
 *     public $val = 0;
 *     public $next = null;
 *     function __construct($val = 0, $next = null) {
 *         $this->val = $val;
 *         $this->next = $next;
 *     }
 * }
 */
class Solution {

/***
 * @param ListNode $head
 * @param Integer $n
 * @return ListNode
 */
function removeNthFromEnd($head, $n) {

}
```

Dart:

```
/**  
 * Definition for singly-linked list.  
 * class ListNode {  
 * int val;  
 * ListNode? next;  
 * ListNode([this.val = 0, this.next]);  
 * }  
 *  
 class Solution {  
 ListNode? removeNthFromEnd(ListNode? head, int n) {  
  
}  
}  
}
```

Scala:

```
/**  
 * Definition for singly-linked list.  
 * class ListNode(_x: Int = 0, _next: ListNode = null) {  
 * var next: ListNode = _next  
 * var x: Int = _x  
 * }  
 *  
 object Solution {  
 def removeNthFromEnd(head: ListNode, n: Int): ListNode = {  
  
}  
}
```

Elixir:

```
# Definition for singly-linked list.  
#  
# defmodule ListNode do  
# @type t :: %__MODULE__{  
# val: integer,  
# next: ListNode.t() | nil  
# }  
# defstruct val: 0, next: nil  
# end
```

```

defmodule Solution do
  @spec remove_nth_from_end(head :: ListNode.t | nil, n :: integer) :: 
  ListNode.t | nil
  def remove_nth_from_end(head, n) do
    end
  end
end

```

Erlang:

```

%% Definition for singly-linked list.

%% -record(list_node, {val = 0 :: integer(),
%% next = null :: 'null' | #list_node{}}).

-spec remove_nth_from_end(Head :: #list_node{} | null, N :: integer()) ->
#list_node{} | null.
remove_nth_from_end(Head, N) ->
.
.
```

Racket:

```

; Definition for singly-linked list:
#| 

; val : integer?
; next : (or/c list-node? #f)
(struct list-node
  (val next) #:mutable #:transparent)

; constructor
(define (make-list-node [val 0])
  (list-node val #f))

|#
(define/contract (remove-nth-from-end head n)
  (-> (or/c list-node? #f) exact-integer? (or/c list-node? #f)))
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Remove Nth Node From End of List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {

    }
};
```

Java Solution:

```
/**
 * Problem: Remove Nth Node From End of List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */
```

```

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 *
 class Solution {
    public ListNode removeNthFromEnd(ListNode head, int n) {
        }

    }
}

```

Python3 Solution:

```

"""
Problem: Remove Nth Node From End of List
Difficulty: Medium
Tags: array, linked_list

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:

    def removeNthFromEnd(self, head: Optional[ListNode], n: int) ->
        Optional[ListNode]:
        # TODO: Implement optimized solution
        pass

```

Python Solution:

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution(object):
    def removeNthFromEnd(self, head, n):
        """
:type head: Optional[ListNode]
:type n: int
:rtype: Optional[ListNode]
"""

```

JavaScript Solution:

```
/**
 * Problem: Remove Nth Node From End of List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * function ListNode(val, next) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.next = (next===undefined ? null : next)
 * }
 */
/**
 * @param {ListNode} head
 * @param {number} n
 * @return {ListNode}
 */
var removeNthFromEnd = function(head, n) {

};
```

TypeScript Solution:

```
/**  
 * Problem: Remove Nth Node From End of List  
 * Difficulty: Medium  
 * Tags: array, linked_list  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */  
  
/**  
 * Definition for singly-linked list.  
 * class ListNode {  
 * val: number  
 * next: ListNode | null  
 * constructor(val?: number, next?: ListNode | null) {  
 * this.val = (val===undefined ? 0 : val)  
 * this.next = (next===undefined ? null : next)  
 * }  
 * }  
 */  
  
function removeNthFromEnd(head: ListNode | null, n: number): ListNode | null  
{  
};
```

C# Solution:

```
/*  
 * Problem: Remove Nth Node From End of List  
 * Difficulty: Medium  
 * Tags: array, linked_list  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */  
  
/**
```

```

* Definition for singly-linked list.
* public class ListNode {
*     public int val;
*     public ListNode next;
*     public ListNode(int val=0, ListNode next=null) {
*         this.val = val;
*         this.next = next;
*     }
* }
*/
public class Solution {
    public ListNode RemoveNthFromEnd(ListNode head, int n) {
        }

    }
}

```

C Solution:

```

/*
* Problem: Remove Nth Node From End of List
* Difficulty: Medium
* Tags: array, linked_list
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
/***
* Definition for singly-linked list.
* struct ListNode {
*     int val;
*     struct ListNode *next;
* };
*/
struct ListNode* removeNthFromEnd(struct ListNode* head, int n) {

}

```

Go Solution:

```

// Problem: Remove Nth Node From End of List
// Difficulty: Medium
// Tags: array, linked_list
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func removeNthFromEnd(head *ListNode, n int) *ListNode {
}

```

Kotlin Solution:

```

/**
 * Example:
 * var li = ListNode(5)
 * var v = li.`val`
 *
 * Definition for singly-linked list.
 * class ListNode(var `val`: Int) {
 *     var next: ListNode? = null
 * }
 */
class Solution {
    fun removeNthFromEnd(head: ListNode?, n: Int): ListNode? {
        }
    }
}

```

Swift Solution:

```

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     public var val: Int
 *     public var next: ListNode?
 * }
 */

```

```

* public var next: ListNode?
* public init() { self.val = 0; self.next = nil; }
* public init(_ val: Int) { self.val = val; self.next = nil; }
* public init(_ val: Int, _ next: ListNode?) { self.val = val; self.next =
next; }
* }
*/
class Solution {
func removeNthFromEnd(_ head: ListNode?, _ n: Int) -> ListNode? {
}
}

```

Rust Solution:

```

// Problem: Remove Nth Node From End of List
// Difficulty: Medium
// Tags: array, linked_list
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

// Definition for singly-linked list.
// #[derive(PartialEq, Eq, Clone, Debug)]
pub struct ListNode {
    pub val: i32,
    pub next: Option<Box<ListNode>>
}

// impl ListNode {
// #[inline]
// fn new(val: i32) -> Self {
//     ListNode {
//         next: None,
//         val
//     }
// }
// }

impl Solution {
    pub fn remove_nth_from_end(head: Option<Box<ListNode>>, n: i32) ->

```

```
Option<Box<ListNode>> {  
}  
}  
}
```

Ruby Solution:

```
# Definition for singly-linked list.  
# class ListNode  
# attr_accessor :val, :next  
# def initialize(val = 0, _next = nil)  
#   @val = val  
#   @next = _next  
# end  
# end  
# @param {ListNode} head  
# @param {Integer} n  
# @return {ListNode}  
def remove_nth_from_end(head, n)  
  
end
```

PHP Solution:

```
/**  
 * Definition for a singly-linked list.  
 * class ListNode {  
 *   public $val = 0;  
 *   public $next = null;  
 *   function __construct($val = 0, $next = null) {  
 *     $this->val = $val;  
 *     $this->next = $next;  
 *   }  
 * }  
 */  
class Solution {  
  
/**  
 * @param ListNode $head  
 * @param Integer $n  
 * @return ListNode  
 */
```

```
function removeNthFromEnd($head, $n) {  
}  
}  
}
```

Dart Solution:

```
/**  
 * Definition for singly-linked list.  
 * class ListNode {  
 * int val;  
 * ListNode? next;  
 * ListNode([this.val = 0, this.next]);  
 * }  
 */  
class Solution {  
ListNode? removeNthFromEnd(ListNode? head, int n) {  
  
}  
}
```

Scala Solution:

```
/**  
 * Definition for singly-linked list.  
 * class ListNode(_x: Int = 0, _next: ListNode = null) {  
 * var next: ListNode = _next  
 * var x: Int = _x  
 * }  
 */  
object Solution {  
def removeNthFromEnd(head: ListNode, n: Int): ListNode = {  
  
}  
}
```

Elixir Solution:

```
# Definition for singly-linked list.  
#  
# defmodule ListNode do
```

```

# @type t :: %__MODULE__{
#   val: integer,
#   next: ListNode.t() | nil
# }
# defstruct val: 0, next: nil
# end

defmodule Solution do
@spec remove_nth_from_end(head :: ListNode.t() | nil, n :: integer) :: 
ListNode.t() | nil
def remove_nth_from_end(head, n) do
end
end

```

Erlang Solution:

```

%% Definition for singly-linked list.

%%
%% -record(list_node, {val = 0 :: integer(),
%% next = null :: 'null' | #list_node{}}).

-spec remove_nth_from_end(Head :: #list_node{} | null, N :: integer()) ->
#list_node{} | null.
remove_nth_from_end(Head, N) ->
.
.
```

Racket Solution:

```

; Definition for singly-linked list:
#|
;

; val : integer?
; next : (or/c list-node? #f)
(struct list-node
  (val next) #:mutable #:transparent)

; constructor
(define (make-list-node [val 0])
  (list-node val #f))

```

```
| #  
  
(define/contract (remove-nth-from-end head n)  
  (-> (or/c list-node? #f) exact-integer? (or/c list-node? #f)))  
)
```