# Swap Two Nodes in Linked List

Given a linked list and two values v1 and v2. Swap the two nodes in the linked list with values v1 and v2. It's guaranteed there is no duplicate values in the linked list. If v1 or v2 does not exist in the given linked list, do nothing.

Given `1->2->3->4->null` and v1 = `2`, v2 = `4`.
Return `1->4->3->2->null`.

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

class Solution:
    # @param {ListNode} head, a ListNode
    # @oaram {int} v1 an integer
    # @param {int} v2 an integer
    # @return {ListNode} a new head of singly-linked list
    def swapNodes(self, head, v1, v2):
        # Write your code here
        if head is None:
            return None

        dummy = ListNode(0)
        dummy.next = head

        prev1, prev2 = self.findNodes(dummy, v1, v2)

        if prev1 is None or prev2 is None:
            return head

        if prev1 == prev2:
            return head

        if prev1.next == prev2:
            self.swapAdjacent(prev1)
        elif prev2.next == prev1:
            self.swapAdjacent(prev2)
        else:
            self.swapRemote(prev1,prev2)
```

```
    return dummy.next

#find pre node position of targets
def findNodes(self, dummy, v1, v2):
    prev1, prev2 = None, None
    node = dummy
    while node.next is not None:
        if node.next.val == v1:
            prev1 = node
        if node.next.val == v2:
            prev2 = node
        node = node.next
    return prev1, prev2

#swap two adjacent nodes
def swapAdjacent(self, prev):
    node1 = prev.next
    node2 = node1.next
    post = node2.next

    prev.next = node2
    node2.next = node1
    node1.next = post

#swap two remote nodes
def swapRemote(self, prev1, prev2):
    node1 = prev1.next
    post1 = node1.next

    node2 = prev2.next
    post2 = node2.next

    prev1.next = node2
    node2.next = post1

    prev2.next = node1
    node1.next = post2
```