25. Reverse Nodes in k-Group

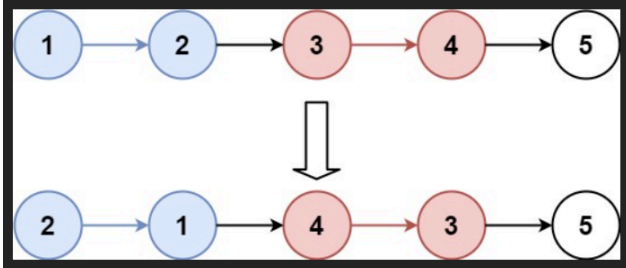
Problem Statement

1. Given the `head` of a linked list, reverse the nodes of the list `k` at a time, and return *the modified list*.

   `k` is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of `k` then left-out nodes, in the end, should remain as it is.

   You may not alter the values in the list's nodes, only nodes themselves may be changed.

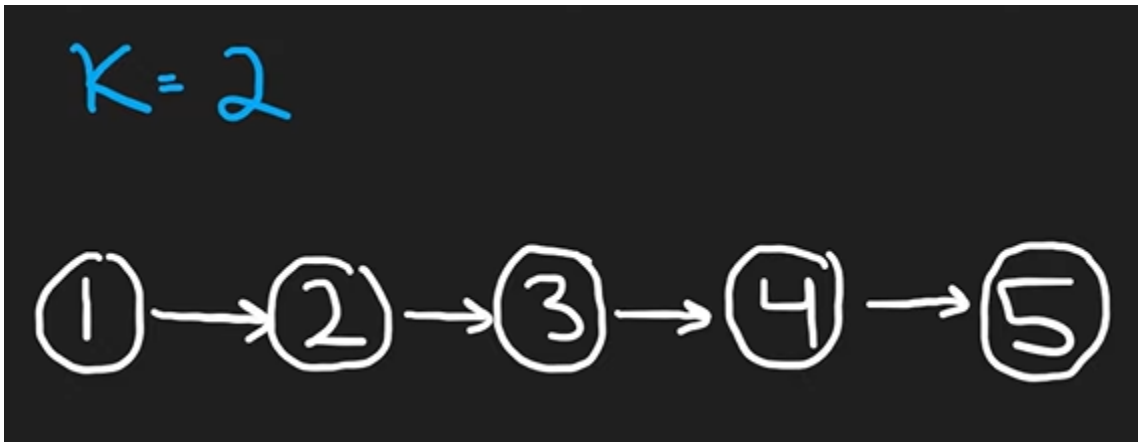
   Example 1:

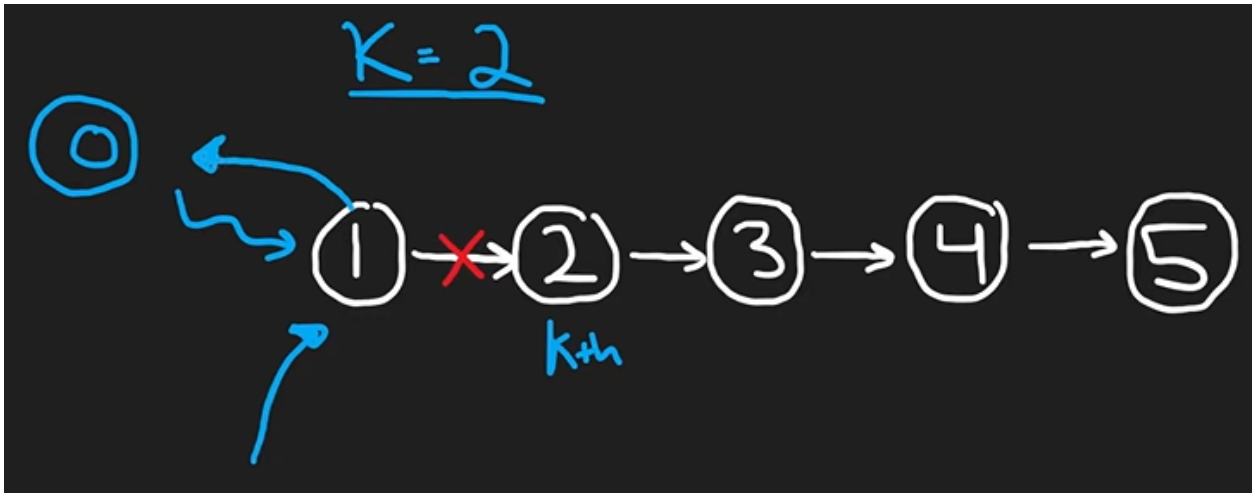

   ```
   Input: head = [1,2,3,4,5], k = 2
   Output: [2,1,4,3,5]
   ```

Drawing a Solution

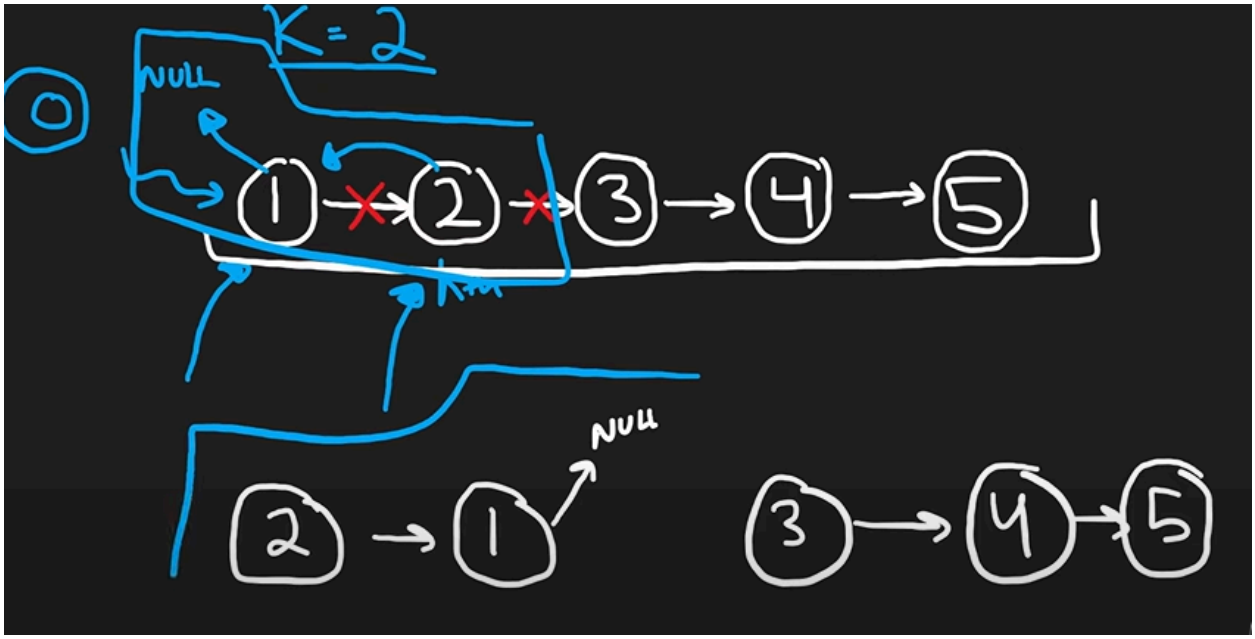
1. Lets consider this list



2. So here we'll take one dummy node(0) ..which helps us identifying the head

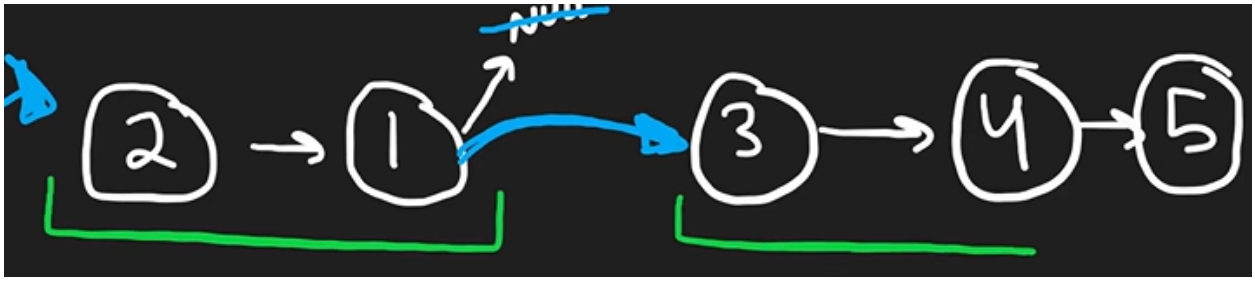


3. So we'll move till k nodes and reverse the nodes



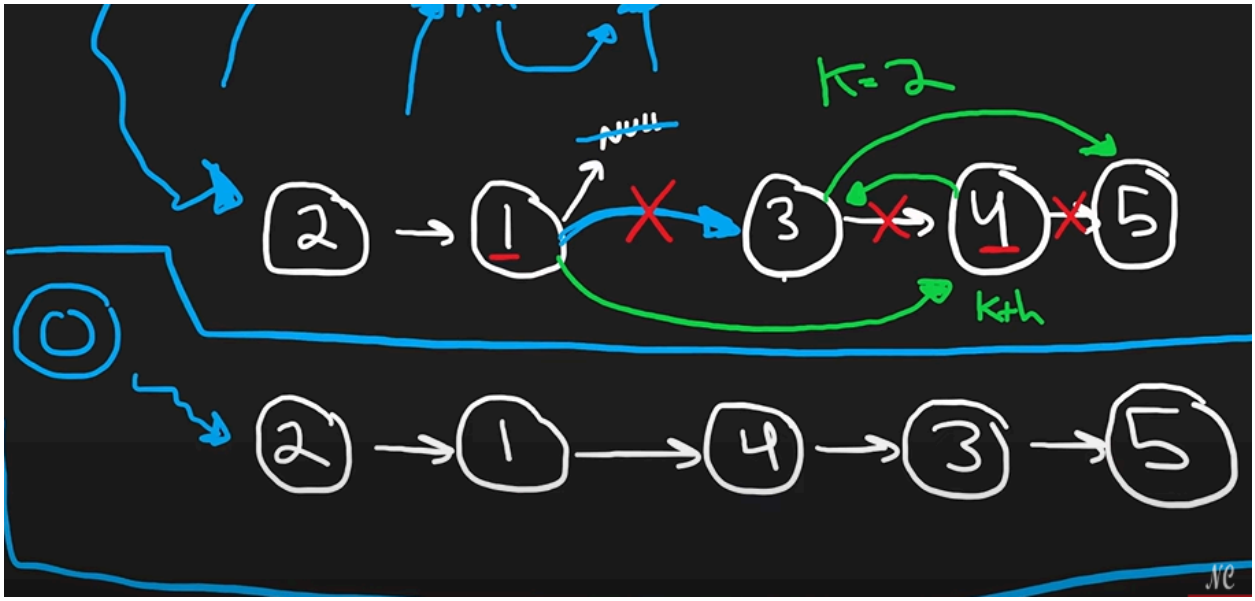
Now node 1 must point to K.next(k is 2 and k.next which is node 3)

4. Here we are done with the first group reversal



5. Now again we move our pointer k steps and perform reversal on node 3 and node 4

6. Now after performing the reversal our nodes will look like this



Python code using Recursion:

1.
```python
class Solution:
    def reverseKGroup(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:
        if not head: return
        start = end = head
        # find the k-th node (the end node for the current group)
        for _ in range(k):
            if not end: return head # not enough items (< k) => remain the order
            end = end.next
        # reverse the current group with k nodes
        newHead = self.reverse(start, end)
        # after reverse start is the end for the group, link it with the next reversed group
        start.next = self.reverseKGroup(end, k)

        return newHead

    # reverse diapason [start:end), end not inclusive
    def reverse(self, start, end):
        prev = None
        while start != end:
            start.next, start, prev = prev, start.next, start
        return prev # return head node of the reversed group
```
2. Here we push our k to k steps and perform the reverse option from start to k..
3. Now start will come to end..and its start.next will be reverse of next k elements..we have done this in recursive function