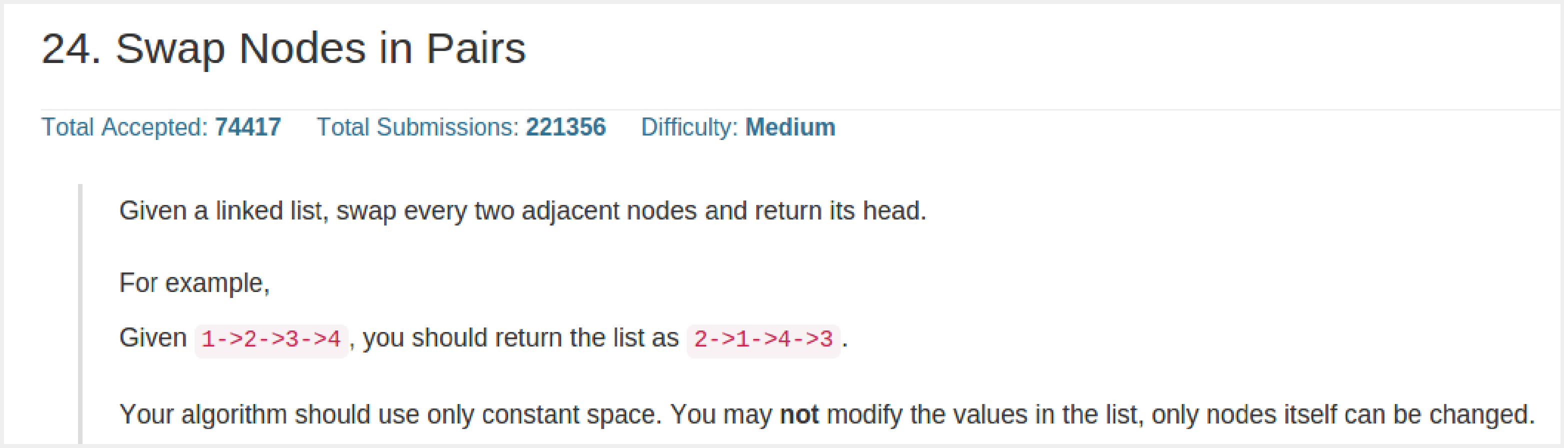
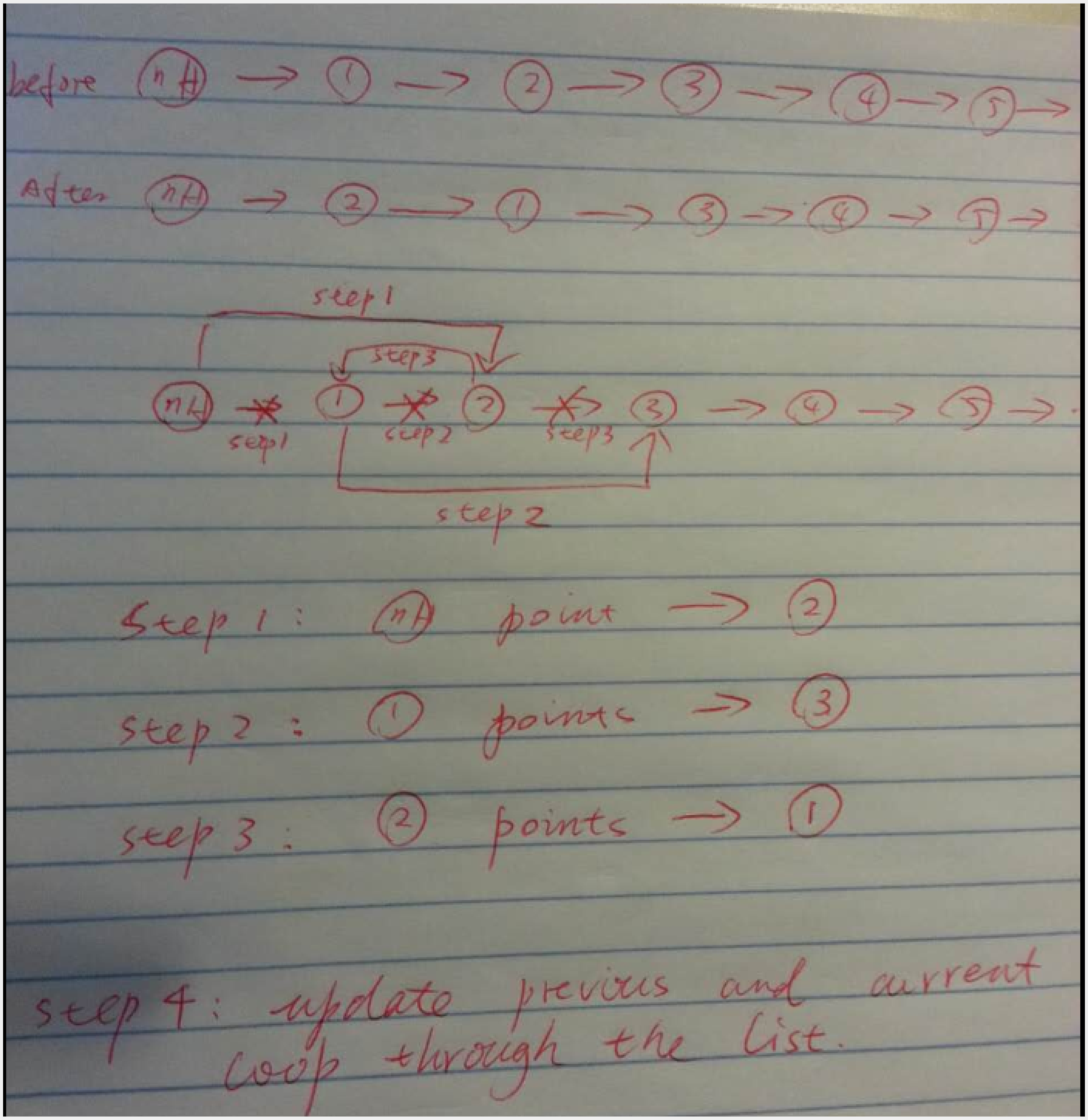
problem:



24. Swap Nodes in Pairs

Total Accepted: **74417** Total Submissions: **221356** Difficulty: **Medium**

> Given a linked list, swap every two adjacent nodes and return its head.
>
> For example,
>
> Given `1->2->3->4`, you should return the list as `2->1->4->3`.
>
> Your algorithm should use only constant space. You may **not** modify the values in the list, only nodes itself can be changed.



before nH → 1 → 2 → 3 → 4 → 5 →
After nH → 2 → 1 → 3 → 4 → 5 →
step 1
step3
nH 1 2 3 → 4 → 5 →
step1
step2
step3
step 2

Step 1: (nH) point → (2)

step 2: (1) points → (3)

step 3: (2) points → (1)

step 4: update previous and current
loop through the list.

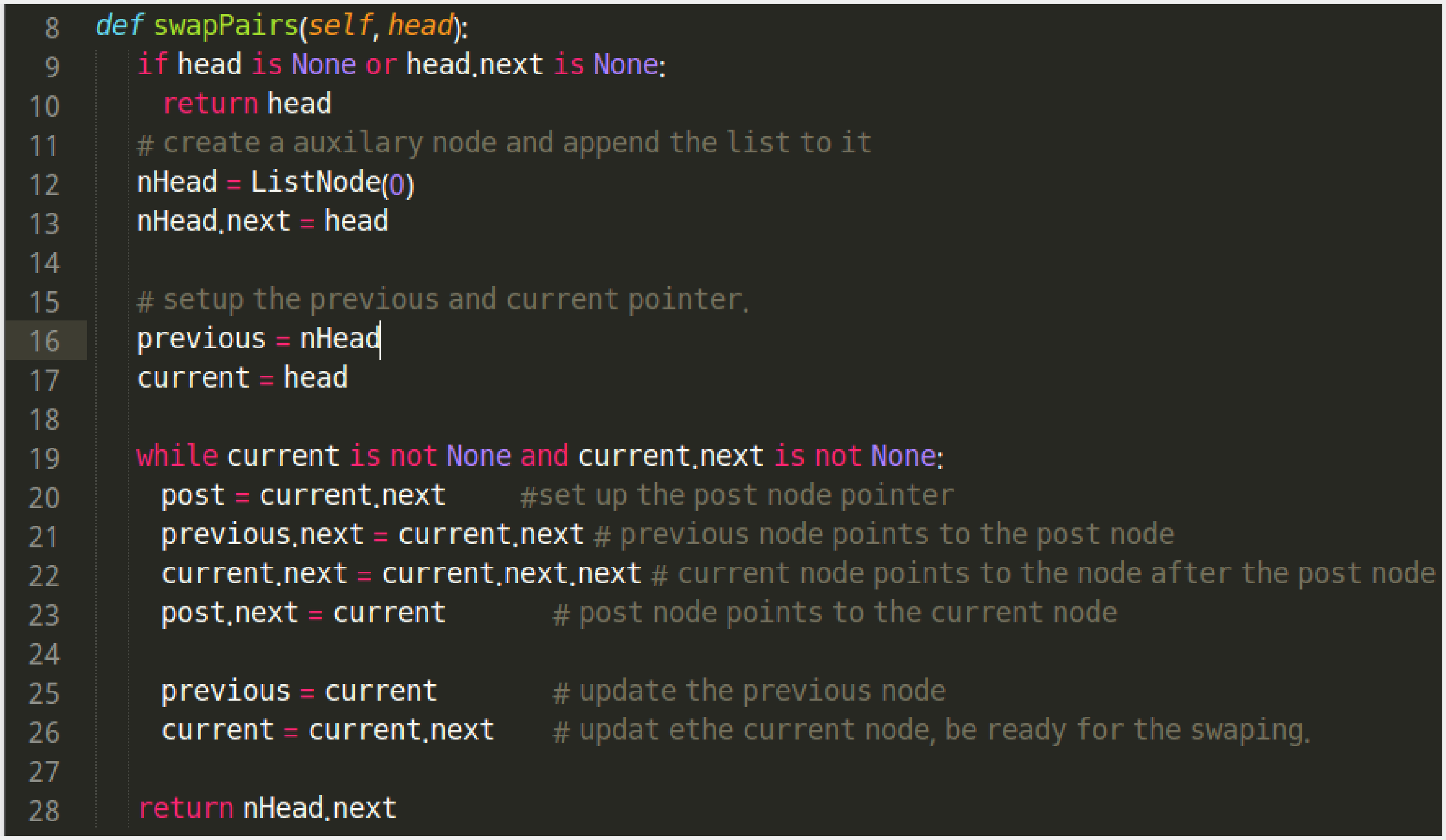
Analysis:

It’s convenient to add a node at the top for help. so we do not need to take care of a case that a node does not have a previous node.

How to swap a pair of nodes inside a list:
As shown in the figure, suppose we have previous, current, and the post node, and we need to swap the current and the post node. What we will do is:
Step1: let previous node points to the post node,
Step2: current node points to the node after the post node.
step3: the post node points to the current node

After these three steps we swap the current and the post node, next we have update the current node and the previous node. Then loop through the list and do the same three steps.

Code:



```python
def swapPairs(self, head):
  if head is None or head.next is None:
    return head
  # create a auxilary node and append the list to it
  nHead = ListNode(0)
  nHead.next = head

  # setup the previous and current pointer.
  previous = nHead
  current = head

  while current is not None and current.next is not None:
    post = current.next       #set up the post node pointer
    previous.next = current.next # previous node points to the post node
    current.next = current.next.next # current node points to the node after the post node
    post.next = current         # post node points to the current node

    previous = current          # update the previous node
    current = current.next      # updat ethe current node, be ready for the swaping.

  return nHead.next
```