**Merge k sorted list:**

# Definition for singly-linked list.

# class ListNode:

# def \_\_init\_\_(self, val=0, next=None):

# self.val = val

# self.next = next

class Solution:

def mergeKLists(self, lists: List[Optional[ListNode]]) -> Optional[ListNode]:

if not lists:

return None

def mergeTwoLists(l1, l2):

dummy = ListNode(0)

current = dummy

while l1 and l2:

if l1.val < l2.val:

current.next = l1

l1 = l1.next

else:

current.next = l2

l2 = l2.next

current = current.next

current.next = l1 or l2

return dummy.next

# Divide and conquer

while len(lists) > 1:

merged\_lists = []

for i in range(0, len(lists), 2):

l1 = lists[i]

l2 = lists[i + 1] if i + 1 < len(lists) else None

merged\_lists.append(mergeTwoLists(l1, l2))

lists = merged\_lists

return lists[0]

**swap nodes in pairs:**

# Definition for singly-linked list.

# class ListNode:

# def \_\_init\_\_(self, val=0, next=None):

# self.val = val

# self.next = next

class Solution:

def swapPairs(self, head: Optional[ListNode]) -> Optional[ListNode]:

dummy = ListNode(0)

dummy.next = head

prev = dummy

while prev.next and prev.next.next:

first = prev.next

second = first.next

# Swap

prev.next, first.next, second.next = second, second.next, first

# Move prev two nodes forward

prev = first

return dummy. Next

**reverse nodes in k-groups:**

# Definition for singly-linked list.

# class ListNode:

# def \_\_init\_\_(self, val=0, next=None):

# self.val = val

# self.next = next

class Solution:

def reverseKGroup(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:

dummy = ListNode(0)

dummy.next = head

group\_prev = dummy

while True:

# Find the kth node

kth = group\_prev

count = 0

while count < k and kth.next:

kth = kth.next

count += 1

if count < k:

break # Less than k nodes left, stop

group\_next = kth.next

# Reverse group

prev, curr = group\_next, group\_prev.next

for \_ in range(k):

tmp = curr.next

curr.next = prev

prev = curr

curr = tmp

tmp = group\_prev.next

group\_prev.next = prev

group\_prev = tmp

return dummy.next

**remove duplicates from sorted array:**

class Solution:

def removeDuplicates(self, nums: List[int]) -> int:

if not nums:

return 0

# Pointer for the position of the next unique element

i = 0

for j in range(1, len(nums)):

if nums[j] != nums[i]:

i += 1

nums[i] = nums[j]

# Return the number of unique elements

return i + 1