NAME SARDAR ROSHAN

lab DSA 4

Roll no 12372

Bsse 3rd

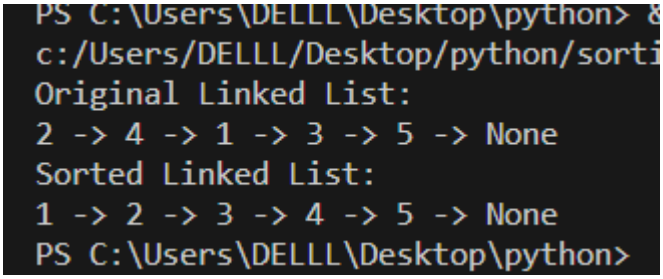
```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr, 0

    # Split the array into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Recursively sort both halves and count inversions
    left_half, inversions_left = merge_sort(left_half)
    right_half, inversions_right = merge_sort(right_half)

    # Merge the sorted halves and count inversions in the merge step
    merged, inversions_merge = merge(left_half, right_half)

    # Total inversions is the sum of inversions in left, right, and merge steps
    total_inversions = inversions_left + inversions_right + inversions_merge

    return merged, total_inversions

def merge(left, right):
    merged = []
    inversions = 0
    i = j = 0
```

```
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            merged.append(left[i])
            i += 1
        else:
            merged.append(right[j])
            j += 1
            # If an inversion is found, add
            inversions += len(left) - i

    merged.extend(left[i:])
    merged.extend(right[j:])

    return merged, inversions

# Example usage
arr = [2, 4, 1, 3, 5]
sorted_arr, inversions = merge_sort(arr)
print("Sorted Array:", sorted_arr)
print("Number of Inversions:", inversions)
```

```
PS C:\Users\DELLL\Desktop\python> & C:/Users/DELLL/
c:/Users/DELLL/Desktop/python/mergedescending.py
Sorted Array (Descending): [5, 4, 3, 2, 1]
PS C:\Users\DELLL\Desktop\python>
```

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def merge_sort(head):
    if not head or not head.next:
        return head

    # Split the linked list into two halves
    mid = find_middle(head)
    left_half, right_half = head, mid.next
    mid.next = None

    # Recursively sort both halves
    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    # Merge the sorted halves
    sorted_list = merge(left_half, right_half)

    return sorted_list

def find_middle(head):
    slow = head
    fast = head
```

```python
def print_linked_list(head):
    while head:
        print(head.val, end=" -> ")
        head = head.next
    print("None")

# Example usage
arr = [2, 4, 1, 3, 5]
head = ListNode(arr[0])
current = head
for num in arr[1:]:
    current.next = ListNode(num)
    current = current.next

print("Original Linked List:")
print_linked_list(head)

sorted_list = merge_sort(head)

print("Sorted Linked List:")
print_linked_list(sorted_list)


    return dummy.next
```


PS C:\Users\DELLL\Desktop\python> &
c:/Users/DELLL/Desktop/python/sorti
Original Linked List:
2 -> 4 -> 1 -> 3 -> 5 -> None
Sorted Linked List:
1 -> 2 -> 3 -> 4 -> 5 -> None
PS C:\Users\DELLL\Desktop\python>

```python
def merge_sort_descending(arr):
    if len(arr) <= 1:
        return arr

    # Split the array into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Recursively sort both halves
    left_half = merge_sort_descending(left_half)
    right_half = merge_sort_descending(right_half)

    # Merge the sorted halves in descending order
    merged = merge_descending(left_half, right_half)

    return merged

def merge_descending(left, right):
    merged = []
    i = 0
    j = 0

    while i < len(left) and j < len(right):
        if left[i] >= right[j]:  # Compare in descending order
            merged.append(left[i])
```

```
PS C:\Users\DELLL\Desktop\python> & C:/Users/DELLL/
c:/Users/DELLL/Desktop/python/mergedescending.py
Sorted Array (Descending): [5, 4, 3, 2, 1]
PS C:\Users\DELLL\Desktop\python>
```

```python
def multiway_merge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Determine the number of sublists to split into (e.g., 3-way merge)
    sublist_size = len(arr) // 3

    # Split the array into sublists
    sublists = [arr[i:i+sublist_size] for i in range(0, len(arr), sublist_size)]

    # Recursively sort each sublist
    sublists = [multiway_merge_sort(sublist) for sublist in sublists]

    # Merge the sorted sublists
    sorted_arr = multiway_merge(sublists)

    return sorted_arr

def multiway_merge(sublists):
    sorted_arr = []

    while any(sublists):
        # Find the minimum element among the sublists
        min_values = [sublist[0] if sublist else float('inf') for sublist in sublist
        min_index = min(range(len(sublists)), key=lambda i: min_values[i])
```

```python
            sublists[min_index].pop(0)

    return sorted_arr

# Example usage
arr = [9, 7, 5, 8, 3, 4, 1, 6, 2]
sorted_arr = multiway_merge_sort(arr)
print("Sorted Array:", sorted_arr)
```

```
PS C:\Users\DELLL\Desktop\python> & C:/Users/DE
c:/Users/DELLL/Desktop/python/mergethreealgorit
Sorted Array: [1, 2, 3, 4, 5, 6, 7, 8, 9]
PS C:\Users\DELLL\Desktop\python>
```