**Merge Sort on Doubly Linked List**

Given Pointer/Reference to the head of a doubly linked list of n nodes, the task is to Sort the given doubly linked list using Merge Sort in both non-decreasing and non-increasing order.

Example 1:

Input:

n = 8

value[] = {7,3,5,2,6,4,1,8}

Output:

1 2 3 4 5 6 7 8

8 7 6 5 4 3 2 1

Explanation: After sorting the given

linked list in both ways, resultant

matrix will be as given in the first

two line of output, where first line

is the output for non-decreasing

order and next line is for non-

increasing order.

Example 2:

Input:

n = 5

value[] = {9,15,0,-1,0}

Output:

-1 0 0 9 15

15 9 0 0 -1

Explanation: After sorting the given

linked list in both ways, the

resultant list will be -1 0 0 9 15

in non-decreasing order and

15 9 0 0 -1 in non-increasing order.

Your Task:

The task is to complete the function sortDoubly() which takes reference to the head of the doubly linked and Sort the given doubly linked list using Merge Sort in both non-decreasing and non-increasing. The printing is done automatically by the driver code.

Expected Time Complexity: O(nlogn)

Expected Space Complexity: O(logn)

Constraints:

1 <= n <= 104

-105 <= values[i] <= 105

class Solution {

public:

struct Node\* merge(struct Node \*left, struct Node \*right){

struct Node \*ans = NULL;

if(left->data < right->data){

ans = left;

left = left->next;

}else{

ans = right;

right = right->next;

}

struct Node \*tail = ans;

while(left != NULL && right != NULL){

if(left->data < right->data){

struct Node \*x = left;

tail->next = x;

x->prev = tail;

tail = tail->next;

left = left->next;

}else{

struct Node \*x = right;

tail->next = x;

x->prev = tail;

tail = tail->next;

right = right->next;

}

}

while(left != NULL){

struct Node \*x = left;

tail->next = x;

x->prev = tail;

tail = tail->next;

left = left->next;

}

while(right != NULL){

struct Node \*x = right;

tail->next = x;

x->prev = tail;

tail = tail->next;

right = right->next;

}

tail->next = NULL;

return ans;

}

struct Node\* mergesort(struct Node\* head, int n){

if(n <= 1)

return head;

int mid = (n-1)/2;

int curr = 0;

struct Node \*temp1 = head;

while(curr < mid){

temp1 = temp1->next;

curr++;

}

struct Node \*temp2 = temp1->next;

temp1->next = NULL;

temp2->prev = NULL;

struct Node \*left = mergesort(head, mid+1);

struct Node \*right = mergesort(temp2, n-mid-1);

return merge(left, right);

}

// Function to sort the given doubly linked list using Merge Sort.

struct Node \*sortDoubly(struct Node \*head) {

int n = 0;

struct Node \*temp = head;

while(temp != NULL){

n++;

temp = temp->next;

}

return mergesort(head, n);

}

};

Link : <https://www.geeksforgeeks.org/problems/merge-sort-on-doubly-linked-list/1>