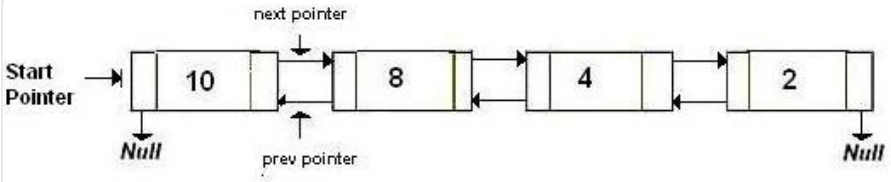
Merge Sort for Doubly Linked List

Given a doubly linked list, write a function to sort the doubly linked list in increasing order using merge sort.

For example, the following doubly linked list should be changed to 2<->4<->8<->10



We strongly recommend to minimize your browser and try this yourself first.

Merge sort for singly linked list is already discussed. The important change here is to modify the previous pointers also when merging two lists.

Below is C implementation of merge sort for doubly linked list.

```
// C program for merge sort on doubly linked list
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next, *prev;
};

struct node *split(struct node *head);

// Function to merge two linked lists
struct node *merge(struct node *first, struct node *second)
{
    // If first linked list is empty
    if (!first)
        return second;

    // If second linked list is empty
    if (!second)
        return first;

    // Pick the smaller value
    if (first->data < second->data)
    {
        first->next = merge(first->next,second);
        first->next->prev = first;
```

```
        first->prev = NULL;
        return first;
    }
    else
    {
        second->next = merge(first,second->next);
        second->next->prev = second;
        second->prev = NULL;
        return second;
    }
}

// Function to do merge sort
struct node *mergeSort(struct node *head)
{
    if (!head || !head->next)
        return head;
    struct node *second = split(head);

    // Recur for left and right halves
    head = mergeSort(head);
    second = mergeSort(second);

    // Merge the two sorted halves
    return merge(head,second);
}

// A utility function to insert a new node at the
// beginning of doubly linked list
void insert(struct node **head, int data)
{
    struct node *temp =
        (struct node *)malloc(sizeof(struct node));
    temp->data = data;
    temp->next = temp->prev = NULL;
    if (!(*head))
        (*head) = temp;
    else
    {
        temp->next = *head;
        (*head)->prev = temp;
        (*head) = temp;
    }
}

// A utility function to print a doubly linked list in
// both forward and backward directions
void print(struct node *head)
{
    struct node *temp = head;
    printf("Forward Traversal using next poitner\n");
    while (head)
    {
        printf("%d ",head->data);
        temp = head;
        head = head->next;
    }
    printf("\nBackword Traversal using prev pointer\n");
    while (temp)
    {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
}
```

```
// Utility function to swap two integers
void swap(int *A, int *B)
{
    int temp = *A;
    *A = *B;
    *B = temp;
}

// Split a doubly linked list (DLL) into 2 DLLs of
// half sizes
struct node *split(struct node *head)
{
    struct node *fast = head,*slow = head;
    while (fast->next && fast->next->next)
    {
        fast = fast->next->next;
        slow = slow->next;
    }
    struct node *temp = slow->next;
    slow->next = NULL;
    return temp;
}

// Driver program
int main(void)
{
    struct node *head = NULL;
    insert(&head,5);
    insert(&head,20);
    insert(&head,4);
    insert(&head,3);
    insert(&head,30);
    insert(&head,10);
    printf("Linked List before sorting\n");
    print(head);
    head = mergeSort(head);
    printf("\n\nLinked List after sorting\n");
    print(head);
    return 0;
}
```

Run on IDE

Output:

```
Linked List before sorting
Forward Traversal using next pointer
10 30 3 4 20 5
Backward Traversal using prev pointer
5 20 4 3 30 10

Linked List after sorting
Forward Traversal using next pointer
3 4 5 10 20 30
Backward Traversal using prev pointer
30 20 10 5 4 3
```

Thanks to Goku for providing above implementation in a comment here.

Time Complexity: Time complexity of the above implementation is same as time complexity of MergeSort for arrays. It takes Θ(nLogn) time.

You may also like to see QuickSort for doubly linked list

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

10 Comments Category: Linked Lists Sorting

Related Posts:

- Merge two sorted linked lists such that merged list is in reverse order
- Compare two strings represented as linked lists
- Rearrange a given linked list in-place.
- Sort a linked list that is sorted alternating ascending and descending orders?
- Select a Random Node from a Singly Linked List
- Point to next higher value node in a linked list with an arbitrary pointer
- Swap nodes in a linked list without swapping data
- Generic Linked List in C

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.