

Merge Sort for Linked Lists

Merge sort is often preferred for sorting a linked list. The slow random-access performance of a linked list makes some other algorithms (such as quicksort) perform poorly, and others (such as heapsort) completely impossible.

Let head be the first node of the linked list to be sorted and headRef be the pointer to head. Note that we need a reference to head in MergeSort() as the below implementation changes next links to sort the linked lists (not data at the nodes), so head node has to be changed if the data at original head is not the smallest value in linked list.

```
MergeSort(headRef)
```

- 1) If head is NULL or there is only one element in the Linked List then return.
- 2) Else divide the linked list into two halves.
FrontBackSplit(head, &a, &b); /* a and b are two halves */
- 3) Sort the two halves a and b.
MergeSort(a);
MergeSort(b);
- 4) Merge the sorted a and b (using SortedMerge() discussed [here](#)) and update the head pointer using headRef.
*headRef = SortedMerge(a, b);

```
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct node
{
    int data;
```

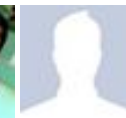
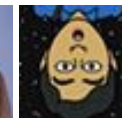
Google™ Custom Search



GeeksforGeeks



53,528 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

```

    struct node* next;
};

/* function prototypes */
struct node* SortedMerge(struct node* a, struct node* b);
void FrontBackSplit(struct node* source,
    struct node** frontRef, struct node** backRef);

/* sorts the linked list by changing next pointers (not data) */
void MergeSort(struct node** headRef)
{
    struct node* head = *headRef;
    struct node* a;
    struct node* b;

    /* Base case -- length 0 or 1 */
    if ((head == NULL) || (head->next == NULL))
    {
        return;
    }

    /* Split head into 'a' and 'b' sublists */
    FrontBackSplit(head, &a, &b);

    /* Recursively sort the sublists */
    MergeSort(&a);
    MergeSort(&b);

    /* answer = merge the two sorted lists together */
    *headRef = SortedMerge(a, b);
}

/* See http://geeksforgeeks.org/?p=3622 for details of this
function */
struct node* SortedMerge(struct node* a, struct node* b)
{
    struct node* result = NULL;

    /* Base cases */
    if (a == NULL)
        return(b);
    else if (b==NULL)
        return(a);

    /* Pick either a or b, and recur */
    if (a->data <= b->data)
    {

```



Popular Posts

All permutations of a given string

Memory Layout of C Programs

Understanding "extern" keyword in C

Median of two sorted arrays

Tree traversal without recursion and without stack!

Structure Member Alignment, Padding and Data Packing

Intersection point of two Linked Lists

Lowest Common Ancestor in a BST.

Check if a binary tree is BST or not

Sorted Linked List to Balanced BST

```

    result = a;
    result->next = SortedMerge(a->next, b);
}
else
{
    result = b;
    result->next = SortedMerge(a, b->next);
}
return(result);
}

/* UTILITY FUNCTIONS */
/* Split the nodes of the given list into front and back halves,
   and return the two lists using the reference parameters.
   If the length is odd, the extra node should go in the front list.
   Uses the fast/slow pointer strategy. */
void FrontBackSplit(struct node* source,
                   struct node** frontRef, struct node** backRef)
{
    struct node* fast;
    struct node* slow;
    if (source==NULL || source->next==NULL)
    {
        /* length < 2 cases */
        *frontRef = source;
        *backRef = NULL;
    }
    else
    {
        slow = source;
        fast = source->next;

        /* Advance 'fast' two nodes, and advance 'slow' one node */
        while (fast != NULL)
        {
            fast = fast->next;
            if (fast != NULL)
            {
                slow = slow->next;
                fast = fast->next;
            }
        }

        /* 'slow' is before the midpoint in the list, so split it in two
           at that point. */
        *frontRef = source;
        *backRef = slow->next;
    }
}

```

Shouldn't you
expect a
cloud with:

SYSTEM MONITORING

Plus the experts
to help run it?

TRY MANAGED CLOUD ▶



```

        slow->next = NULL;
    }
}

/* Function to print nodes in a given linked list */
void printList(struct node *node)
{
    while (node!=NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Function to insert a node at the beginging of the linked list */
void push(struct node** head_ref, int new_data)
{
    /* allocate node */
    struct node* new_node =
        (struct node*) malloc(sizeof(struct node));

    /* put in the data */
    new_node->data = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Drier program to test above functions*/
int main()
{
    /* Start with the empty list */
    struct node* res = NULL;
    struct node* a = NULL;

    /* Let us create a unsorted linked lists to test the functions
    Created lists shall be a: 2->3->20->5->10->15 */
    push(&a, 15);
    push(&a, 10);
    push(&a, 5);
    push(&a, 20);
    push(&a, 3);
    push(&a, 2);

```

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 50 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool

tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this

sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

AdChoices

► [Linked List](#)

► [C++ Code](#)

► [Java Source Code](#)

AdChoices

► [Linked Data](#)

```

/* Sort the above created Linked List */
MergeSort(&a);

printf("\n Sorted Linked List is: \n");
printList(a);

getchar();
return 0;
}

```

Time Complexity: $O(n \log n)$

Sources:

http://en.wikipedia.org/wiki/Merge_sort

<http://cslibrary.stanford.edu/105/LinkedListProblems.pdf>

Please write comments if you find the above code/algorithm incorrect, or find better ways to solve the same problem.

[▶ Java Array](#)

AdChoices ▶

[▶ Null Pointer](#)

[▶ Merge Data](#)

[▶ C++ Example](#)

The advertisement features the Sauce Labs logo at the top. Below it, the text 'Supercharged Selenium Testing' is written in a large, bold, teal font. Underneath, it says 'Spin up browsers in seconds'. A prominent red button with the text 'Test for free' is centered. At the bottom left, there is a speedometer graphic with a needle pointing towards 100. To the right of the speedometer, a screenshot of a Selenium test results table is visible, showing columns for Name, Environment, Page, Build, and Results.

Related Topics:

- Given a linked list, reverse alternate nodes and append at the end
- Pairwise swap elements of a given linked list by changing links

- Self Organizing List | Set 1 (Introduction)
- Merge a linked list into another linked list at alternate positions
- QuickSort on Singly Linked List
- Delete N nodes after M nodes of a linked list
- Design a stack with operations on middle element
- Swap Kth node from beginning with Kth node from end in a Linked List



18



Tweet

1



1

Writing code in comment? Please use ideone.com and share the link here.

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team