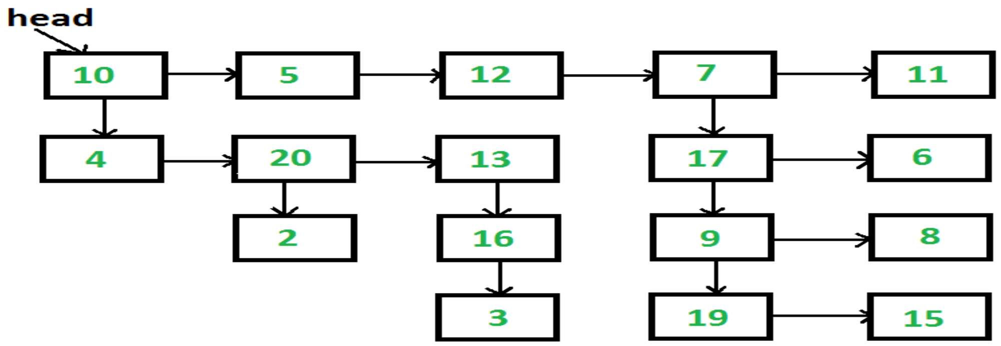
Flatten a multilevel linked list

Given a linked list where in addition to the next pointer, each node has a child pointer, which may or may not point to a separate list. These child lists may have one or more children of their own, and so on, to produce a multilevel data structure, as shown in below figure.You are given the head of the first level of the list. Flatten the list so that all the nodes appear in a single-level linked list. You need to flatten the list in way that all nodes at first level should come first, then nodes of second level, and so on.

Each node is a C struct with the following definition.

```
struct list
{
    int data;
    struct list *next;
    struct list *child;
};
```



The above list should be converted to 10->5->12->7->11->4->20->13->17->6->2->16->9->8->3->19->15

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

The problem clearly say that we need to flatten level by level. The idea of solution is, we start from first level, process all nodes one by one, if a node has a child, then we append the child at the end of list, otherwise we don't do anything. After the first level is processed, all next level nodes will be appended after first level. Same process is followed for the appended nodes.

```
1) Take "cur" pointer, which will point to head of the fist level of the list
2) Take "tail" pointer, which will point to end of the first level of the list
3) Repeat the below procedure while "curr" is not NULL.
    I) if current node has a child then
        a) append this new child list to the "tail"
                tail->next = cur->child
        b) find the last node of new child list and update "tail"
                tmp = cur->child;
                while (tmp->next != NULL)
                        tmp = tmp->next;
                tail = tmp;
    II) move to the next node. i.e. cur = cur->next
```

Following is C implementation of the above algorithm.

```
// Program to flatten list with next and child pointers
#include <stdio.h>
#include <stdlib.h>

// Macro to find number of elements in array
#define SIZE(arr) (sizeof(arr)/sizeof(arr[0]))

// A linked list node has data, next pointer and child pointer
struct node
{
    int data;
    struct node *next;
    struct node *child;
};

// A utility function to create a linked list with n nodes. The data
// of nodes is taken from arr[].  All child pointers are set as NULL
struct node *createList(int *arr, int n)
{
    struct node *head = NULL;
    struct node *p;
```

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

```c
    int i;
    for (i = 0; i < n; ++i) {
        if (head == NULL)
            head = p = (struct node *)malloc(sizeof(*p));
        else {
            p->next = (struct node *)malloc(sizeof(*p));
            p = p->next;
        }
        p->data = arr[i];
        p->next = p->child = NULL;
    }
    return head;
}

// A utility function to print all nodes of a linked list
void printList(struct node *head)
{
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}

// This function creates the input list.  The created list is same
// as shown in the above figure
struct node *createList(void)
{
    int arr1[] = {10, 5, 12, 7, 11};
    int arr2[] = {4, 20, 13};
    int arr3[] = {17, 6};
    int arr4[] = {9, 8};
    int arr5[] = {19, 15};
    int arr6[] = {2};
    int arr7[] = {16};
    int arr8[] = {3};

    /* create 8 linked lists */
    struct node *head1 = createList(arr1, SIZE(arr1));
    struct node *head2 = createList(arr2, SIZE(arr2));
    struct node *head3 = createList(arr3, SIZE(arr3));
    struct node *head4 = createList(arr4, SIZE(arr4));
    struct node *head5 = createList(arr5, SIZE(arr5));
    struct node *head6 = createList(arr6, SIZE(arr6));
    struct node *head7 = createList(arr7, SIZE(arr7));
    struct node *head8 = createList(arr8, SIZE(arr8));
```

705 Subscribe

```
    /* modify child pointers to create the list shown above */
    head1->child = head2;
    head1->next->next->next->child = head3;
    head3->child = head4;
    head4->child = head5;
    head2->next->child = head6;
    head2->next->next->child = head7;
    head7->child = head8;


    /* Return head pointer of first linked list.  Note that all nodes
       reachable from head1 */
    return head1;
}
```

```
/* The main function that flattens a multilevel linked list */
void flattenList(struct node *head)
{
    /*Base case*/
    if (head == NULL)
       return;

    struct node *tmp;

    /* Find tail node of first level linked list */
    struct node *tail = head;
    while (tail->next != NULL)
        tail = tail->next;

    // One by one traverse through all nodes of first level
    // linked list till we reach the tail node
    struct node *cur = head;
    while (cur != tail)
    {
        // If current node has a child
        if (cur->child)
        {
            // then append the child at the end of current list
            tail->next = cur->child;

            // and update the tail to new last node
            tmp = cur->child;
            while (tmp->next)
                tmp = tmp->next;
            tail = tmp;
```

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 44 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

```
        }

        // Change current node
        cur = cur->next;
    }
}
```

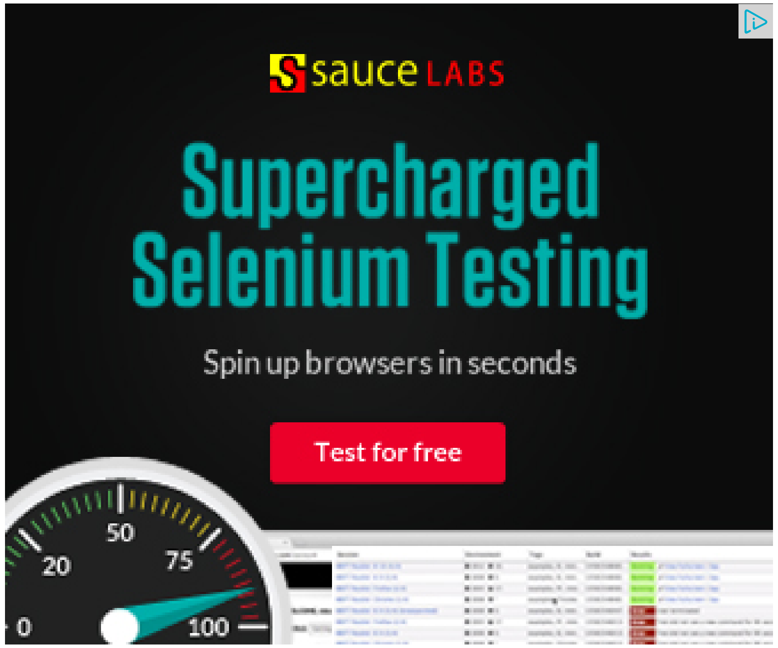
```
// A driver program to test above functions
int main(void)
{
    struct node *head = NULL;
    head = createList();
    flattenList(head);
    printList(head);
    return 0;
}
```

Output:

```
10 5 12 7 11 4 20 13 17 6 2 16 9 8 3 19 15
```

Time Complexity: Since every node is visited at most twice, the time complexity is O(n) where n is the number of nodes in given linked list.

This article is compiled by **Narendra Kangralkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Related Tpoics:

- 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

48 Tweet 3 1

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

36 Comments GeeksforGeeks

Sort by Newest

kaushik Lele • 17 days ago

flattenList has main loop as
while (cur != tail)
So it will not handle to childs of tail. It should have been
while (cur != NULL)

Please update code.

1 • Reply • Share ›

Vishal • 2 months ago

class Queue<node *=""> childList;
/* The main function that flattens a multilevel linked list */
node* flattenList(node *head)
{
if (head == NULL)
return 0;

node *temp = head;
while(temp->next)
{
if(temp->child)
{
childList.Enqueue(temp->child);
}
temp = temp->next;
}
temp->next = childList.Dqueue();
flattenList(temp->next);

return head;

}

Reply • Share ›

xxmajia • 4 months ago

according to "which may or may not point to a separate list"

i think its easier to use a queue structure to solve this puzzle

public static ListNode flattern(ListNode head) {

if (head == null) {

return head;

}

Queue<listnode> queue = new LinkedList<listnode>();

ListNode newHead = new ListNode();

ListNode cur = newHead;

queue.add(head);

while (!queue.isEmpty()) {

see more

Reply • Share ›

Anil • 4 months ago

Queue can be use to solve this problem

Reply • Share ›

Mohan Kishor • 8 months ago

Hey, Your code will not work in the case where all nodes have both the NEXT

node in the list whose NEXT pointer is NULL but with a CHILD pointer. This is t
traverses only considering the NEXT pointer but not the other nodes CHILD pc

2 ^ | ˅ • Reply • Share ›

Ishita • 10 months ago

Your code works incorrectly if the tail has a child. Ex if in your test case 15 we
included in the flattened list. Instead you should change
while(cur!=tail) to while(cur)

```
/* Paste your code here (You may delete these lines if not writing co
```

6 ^ | ˅ • Reply • Share ›

Manisha Barnwal • 11 months ago

/*I think the insert function can be implemented as:
where parent and sibling=-1 if not applicable.
otherwise, it is insert(&start, 2,-1,1);*/.
void insert(list **start, int num, int parent, int sibling).
{

list *p;.

list *temp=(list*)malloc(sizeof(list));.

temp->data=num;.

temp->next=NULL;.

temp->child=NULL;.

if(*start==NULL).

{.

see more

Reply • Share ›

Hanish Bansal • 11 months ago

There is a bug in the program.
In the flattenList function,
while(cur != tail) should be replaced with
while(cur != NULL)

For the input list :
1-2-3-4
|
5
|
6
the answer should be 1-2-3-4-5-6
but the program is giving the output 1-2-3-4.

Also, the child pointers of all nodes should be reset to NULL later or on the fly.

2 • Reply • Share ›

Hanish Bansal Hanish Bansal • 11 months ago

Sorry,in the input list :
5 is the child pointer of 4 (not 1) and 6 is the child pointer of 5

Reply • Share ›

ultimate_coder • 11 months ago

createList function is defined twice here. I don't think it gonna work.
C doesn't support function overloading.

Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

Reply • Share ›

rka143 • a year ago

Also we should make cur->child = NULL; before cur = cur->next; otherwise th
child node.

Reply • Share ›

Hanish • a year ago

Here is a code using queue in O(n) time complexity

cur = head
while(1)
if(cur==NULL)
break;
if cur has a child, enqueue(child).
if(cur->next == NULL)
cur->next = dequeue(q);
cur = cur->next;

Reply • Share ›

CODER_1 ➜ Hanish • a year ago

@Hanish
I think you miss one condition of checking the empty queue,we will sto
elements.
else your nice approach ..!

```
/* Paste your code here (You may delete these lines if not wr
```

Reply • Share ›

Hanish ➜ CODER_1 • a year ago

When queue becomes empty, dequeue(q) will return NULL;
=> current->next = NULL;
current = current->next (= NULL);
if(current == NULL)
break;
since initially queue is empty, we cant break if queue is NULL.

^ | ⌄ • Reply • Share ›

Akash Devkumar • a year ago

I think this can also be implemented using queus(which is used to store the cl
level).

^ | ⌄ • Reply • Share ›

ebcdic666 • a year ago

Complete working code using queues. Enjoy :)

```
 // Program to flatten list with next and child pointers
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
#include<queue>
using namespace std;
// Macro to find number of elements in array
#define SIZE(arr) (sizeof(arr)/sizeof(arr[0]))

// A linked list node has data, next pointer and child pointer
struct node
{
    int data;
    struct node *next;
    struct node *child;
};
```

see more

2 ^ | ˅ • Reply • Share ›

Sreenivas Doosa • a year ago

In the given example what happens if I made 2 connections like

child of 12 is 13
AND
13 next is 17

In this case as per the given algorithm, some part of the list will be added mor

Please correct me if I am wrong..!

^ | ˅ • Reply • Share ›

Michael • a year ago

```
 /* Paste your code here (You may delete these lines if not writing c
Node* flatten(Node* head){
        if(!head)
                return head;

        Node* prev = NULL;
        Node* result = NULL;
        Queue* Q = createQueue();
        Q->enqueue(head);

        while(Q->isEmpty == false)
        {
                Node* temp = Q->dequeue();

                if(prev)
                        prev->next = temp;
                if(!result)
```

see more

^ | ⌄ • Reply • Share ›

Ankit Chaudhary ➦ Michael • a year ago

I think it will not work if last node of a level has child
because
while(temp->next)
will be false if temp is last node of level,
and if it has child, then this child will not be pushed into queue.
Correct me, if I am wrong.
test case: 12
|
13

Only 2 nodes: 13 is child of 12

```
/* Paste your code here (You may delete these lines if not wr
```

^ | ⌄ • Reply • Share ›

Michael ➦ Michael • a year ago

This is a working code:

http://codepad.org/yd52skUK

^ | ⌄ • Reply • Share ›

Shan ➦ Michael • a year ago

This is the simplest method (of using queues) rather than using the po

^ | ⌄ • Reply • Share ›

Nikhil ➦ Michael • a year ago

```
/* Ultimate code */
```

Reply • Share ›

Harsh Gupta • a year ago

in this code you are trying to define "createList" twice. this will not work.

Reply • Share ›

Raj • a year ago

What if it forms a cycle?for instance 7 11 4 20 13 17 7

Reply • Share ›

Shivani Gupta ➦ Raj • a year ago

pls reply admin

Reply • Share ›

aayushkumar • a year ago

i have doubt what if child pointer not point to head of next level??
how can u obtain head of next level in that case??

Reply • Share ›

aayushkumar ➦ aayushkumar • a year ago

@admin plzz reply ASAP

Reply • Share ›

Star_Trek ➦ aayushkumar • a year ago

@ aayushkumar........its simple..
suppose 10 points to 20 as its child node...
then we can specify it by writing...
head1->child=head2->next;

:)

```
/* Paste your code here (You may delete these lines if
```

Reply • Share ›

Shivani Gupta Star_Trek • a year ago

but this you r telling to create the list but how to flatten s

```
/* Paste your code here (You may delete these li
```

Reply • Share ›

spiderman • a year ago

The function for flattening linked list using Queue is as follows:

```
private static NodeNovel flattenLinkedList(NodeNovel head){
                LinkedList<NodeNovel> queue=new LinkedList<NodeNovel>
                queue.addLast(head);
                NodeNovel headAns=null, headAnsCur=null;
                NodeNovel curNode;
                headAns=new NodeNovel(head.val);
                headAnsCur=headAns;
                while(queue.size()>0){
                        curNode=queue.removeFirst();
                        while(curNode!=null){
                                if(curNode.val!=headAns.val){
                                        headAnsCur.next=new NodeNovel
                                        headAnsCur=headAnsCur.next;
                                }
                                queue.addLast(curNode.child);
```

see more

1 ^ | ⌄ • Reply • Share ›

Sandeep Jain • a year ago

Your analysis looks good. We will be adding time complexity to the original pos

^ | ⌄ • Reply • Share ›

Varun Jain • a year ago

What is the time complexity here? I think if we have child node that means we
First to add after tail, 2. time to do a normal traverse for curr = curr.next. Looks
Am I right?

^ | ⌄ • Reply • Share ›

Om Prakash Suthar • a year ago

The one method would be to use queue data structure. The one we use in leve
if(HEAD==NULL)return HEAD;

queue<Node*> q;
q.push(HEAD);
while(1)
{
node * temp;.
temp = q.top();.
q.pop();.
while(temp!=NULL).
{

if(temp->child!= NULL).

q.push(temp->child);.

temp->child = NULL;

see more

Reply • Share ›

GeeksforGeeks • a year ago

We can use a queue. We can enqueue all children to queue one by one. The
child nodes in FIFO order. It doesn&#039t use a an explicit queue though, and

Reply • Share ›

Bharathwaj Soundararajan • a year ago

Wont breadth first search with the child pointers in queue work?

Reply • Share ›

Subscribe   Add Disqus to your site