Add two numbers represented by linked lists | Set 2

Given two numbers represented by two linked lists, write a function that returns sum list. The sum list is linked list representation of addition of two input numbers. It is not allowed to modify the lists. Also, not allowed to use explicit extra space (Hint: Use Recursion).

Example

```
Input:
  First List: 5->6->3  // represents number 563
  Second List: 8->4->2 //  represents number 842
Output
  Resultant list: 1->4->0->5  // represents number 1405
```

We have discussed a solution here which is for linked lists where least significant digit is first node of lists and most significant digit is last node. In this problem, most significant node is first node and least significant digit is last node and we are not allowed to modify the lists. Recursion is used here to calculate sum from right to left.

Following are the steps.
1) Calculate sizes of given two linked lists.
2) If sizes are same, then calculate sum using recursion. Hold all nodes in recursion call stack till the rightmost node, calculate sum of rightmost nodes and forward carry to left side.
3) If size is not same, then follow below steps:
….**a)** Calculate difference of sizes of two linked lists. Let the difference be *diff*
….**b)** Move *diff* nodes ahead in the bigger linked list. Now use step 2 to calculate sum of smaller list and right sub-list (of same size) of larger list. Also, store the carry of this sum.
….**c)** Calculate sum of the carry (calculated in previous step) with the remaining left sub-list of larger list. Nodes of this sum are added at the beginning of sum list obtained previous step.

Following is C implementation of the above approach.

```
// A recursive program to add two linked lists

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

```
// A linked List Node
struct node
{
    int data;
    struct node* next;
};

typedef struct node node;

/* A utility function to insert a node at the beginning of 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;
}

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

// A utility function to swap two pointers
void swapPointer( node** a, node** b )
{
    node* t = *a;
    *a = *b;
    *b = t;
}

/* A utility function to get size of linked list */
int getSize(struct node *node)
{
    int size = 0;
    while (node != NULL)
    {
        node = node->next;
        size++;
    }
    return size;
}

// Adds two linked lists of same size represented by head1 and head2 and returns
// head of the resultant linked list. Carry is propagated while returning from
// the recursion
node* addSameSize(node* head1, node* head2, int* carry)
{
    // Since the function assumes linked lists are of same size,
    // check any of the two head pointers
    if (head1 == NULL)
        return NULL;
    int sum;

    // Allocate memory for sum node of current two nodes
    node* result = (node *)malloc(sizeof(node));

    // Recursively add remaining nodes and get the carry
    result->next = addSameSize(head1->next, head2->next, carry);

    // add digits of current nodes and propagated carry
    sum = head1->data + head2->data + *carry;
    *carry = sum / 10;
    sum = sum % 10;

    // Assigne the sum to current node of resultant list
    result->data = sum;

    return result;
}

// This function is called after the smaller list is added to the bigger
// lists's sublist of same size.  Once the right sublist is added, the carry
// must be added toe left side of larger list to get the final result.
void addCarryToRemaining(node* head1, node* cur, int* carry, node** result)
{
    int sum;

    // If diff. number of nodes are not traversed, add carry
    if (head1 != cur)
    {
        addCarryToRemaining(head1->next, cur, carry, result);

        sum = head1->data + *carry;
        *carry = sum/10;
        sum %= 10;

        // add this node to the front of the result
        push(result, sum);
    }
}

// The main function that adds two linked lists represented by head1 and head2.
// The sum of two lists is stored in a list referred by result
void addList(node* head1, node* head2, node** result)
{
    node *cur;

    // first list is empty
    if (head1 == NULL)
    {
        *result = head2;
        return;
    }

    // second list is empty
    else if (head2 == NULL)
    {
        *result = head1;
        return;
    }

    int size1 = getSize(head1);
    int size2 = getSize(head2) ;

    int carry = 0;
    // Add same size lists
    if (size1 == size2)
        *result = addSameSize(head1, head2, &carry);

    else
    {
        int diff = abs(size1 - size2);

        // First list should always be larger than second list.
        // If not, swap pointers
        if (size1 < size2)
            swapPointer(&head1, &head2);

        // move diff. number of nodes in first list
        for (cur = head1; diff--; cur = cur->next);

        // get addition of same size lists
        *result = addSameSize(cur, head2, &carry);

        // get addition of remaining first list and carry
        addCarryToRemaining(head1, cur, &carry, result);
    }

    // if some carry is still there, add a new node to the fron of
    // the result list. e.g. 999 and 87
    if (carry)
        push(result, carry);
}

// Driver program to test above functions
int main()
{
    node *head1 = NULL, *head2 = NULL, *result = NULL;

    int arr1[] = {9, 9, 9};
    int arr2[] = {1, 8};

    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    // Create first list as 9->9->9
    int i;
    for (i = size1-1; i >= 0; --i)
        push(&head1, arr1[i]);

    // Create second list as 1->8
    for (i = size2-1; i >= 0; --i)
        push(&head2, arr2[i]);

    addList(head1, head2, &result);

    printList(result);

    return 0;
}
```

Run on IDE

Output:

```
1  0  1  7
```

Time Complexity: O(m+n) where m and n are the sizes of given two linked lists.

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

114 Comments Category: Linked Lists

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
- Merge Sort for Doubly 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

(Login to Rate and Mark)

3 Average Difficulty : **3/5.0**
Based on **1** vote(s)

☐ Add to TODO List
☐ Mark as DONE

Like Share 24 people like this.

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

114 Comments **GeeksforGeeks** 1 Login

Recommend 2 Share Sort by Newest

Join the discussion…

Saravana Kumar • 24 days ago

Find the java implementation which with O(N) time complexity here.

```
package addtwolists;

class Node{

int data;

Node next;

public Node(int data){

this.data = data;

this.next = null;

}

public void print(){

System.out.print(data + "\t");
```

see more

• Reply • Share ›

Tulasi Krishna • 2 months ago

Or u can use two stacks to store linked lists and add them up by poping one elemint from each each stack

• Reply • Share ›

Sandeep Ravindra • 2 months ago

addCarryToRemaining(head1, cur, &carry, result);
Shouldn't it be addCarryToRemaining(head1, cur, &carry, &result); ? The '&' before result is missing, in the AddList Function, last but 5th line.

• Reply • Share ›

prashant patel • 4 months ago

Another simplified approach:

```
private int AddList(Node list1, Node list2,int len1, int len2) {
int num = 0;
if (len1 <= 0 && len2 <= 0) {
return 0;
} else if (len1 > len2) {
num = AddList(list1.next, list2,len1 - 1, len2);
num += list1.value;
} else if (len2 > len1) {
num = AddList(list1, list2.next,len1, len2 - 1);
num += list2.value;
} else{
num=AddList(list1.next, list2.next,len1-1, len2-1);
num+=list1.value+list2.value;
}
outList=Add(outList, num%10);
return num / 10;
}
//full code can be found here: http://ideone.com/14kR68
```

• Reply • Share ›

Vikas Jindal • 5 months ago

```
//assume lengthOf(node) returns no. of nodes in linked list. I have not written it.
#include<stdio.h>
#include<stdlib.h>

struct node {
int data;
struct node* next;
};

struct node* addTwoLists(struct node* list1, struct node* list2)
{
if(lengthOf(list1) >= lengthOf(list2))
{
return _addTwoLists(list1, list2);
}
else
{
return _addTwoLists(list2, list1);
```

see more

• Reply • Share ›

Lokesh • 5 months ago

With minor tweak!!

https://ideone.com/gxvD1G

• Reply • Share ›

Talib Hassan • 5 months ago

Hi Guys Please Open My Site And Get All Latest Education's Related News resultadmit.in

• Reply • Share ›

Annu42 • 5 months ago

Result 2015, Educational Information

• Reply • Share ›

Annu42 • 5 months ago

Admit Card , Answer key

• Reply • Share ›

Deepak Sharma • 6 months ago

c++ implementation without recursion..

http://ideone.com/u19fBZ

• Reply • Share ›

Dman • 6 months ago

Implemented by both ways
1) iterative : reverse LL and add them and reverse them again.
2) recursive approach

https://ideone.com/5gCF6J

• Reply • Share ›

Jerry Goyal • 6 months ago

merging above both functions into one..working for all cases and so concise.
here 'd' is the difference between two lists. and first list is always bigger than second.

```
node* full(node* first,node* second,int d,int *carry){
        if(!first&&!second){
                return NULL;
        }

        node* tmp = newNode();

        tmp->next=full(first->next,(d<1)?second->next:second,d-1,carry);

        int sum;
        if(d<1)
```

```
            sum = *carry + first->data + second->data;
        else
            sum = *carry +first->data;
```

see more

• Reply • Share ›

Odin • 8 months ago

Java Solution using Recursion

public class SumOf2LinkedLists {
public static ListNode<integer> total = null;
public static int carry;

public static void main(String[] args) {
ListNode<integer> num1 = new ListNode<integer>(9);
num1.add(num1, 9);
num1.add(num1, 9);
num1.add(num1, 9);
ListNode<integer> num2 = new ListNode<integer>(9);
num2.add(num2, 9);
calculateSum(num1, num2);
ListNode temp = total;
while (temp != null) {
System.out.print(temp.data + " ");
temp = temp.next;

see more

• Reply • Share ›

karansinghkjs • 8 months ago

my function to add two linked lists , if you want the full program just leave in comment i ll post it , if you find a mistake in it please correct me :)

http://ideone.com/83ZkrT

1 • Reply • Share ›

cfh • 8 months ago

```
#include <cstdio>

#include <cstdlib>
```

```
using namespace std;

// A linked List Node

struct Node

{

    int data;

    struct Node* next;

};
```

see more

Reply • Share ›

cfh • 8 months ago

Simple implementation : http://ideone.com/6bGNN3

Reply • Share ›

abhishek jain • 9 months ago

here is more simpler answer

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct node

{

int data;

struct node *next;

};

void push(struct node **head,int ndata)
```

see more

Reply • Share ›

meteor • 9 months ago

what 's the problem with my code ,its not working
#include <iostream>
#include <list>

using namespace std;

int main()

{

list<int> list1;

list1.push_back(1);

list1.push_back(2); // number 123 being represented by list1

list1.push_back(3);

list<int> list2; // number 45 being represented by list2

see more

• Reply • Share ›

Techie Me • 9 months ago

here is another elegant and shorter way of doing this in O(N) time and O(1) auxiliary space. http://techieme.in/adding-numb...

• Reply • Share ›

May Oo • 10 months ago

can you pls guide to the correct response for the following question?

The
below function `alt_sum` was written to return the sum of ALTERNATE
elements of a linked list, starting with the FIRST element.

typedef struct node {

int data;

struct node *next;

} node_t;

int alt_sum(node_t *head) {

int sum = 0;

```
while(head != NULL) {
```

see more

1 ^ | v • Reply • Share ›

Chakra Agarwal • 10 months ago

Another way of doing this will be reverse both the linked list and start to add them. Each node created after adding will be added as head in new linked list.
Eg:
List1: 1->2->3->4
List 2: 1->2
reverse list1: 4->3->2->1
reverse list 2: 2->1
First node 6 is created
then 4->6
and so on
1->2->4->6

^ | v • Reply • Share ›

Techie Me Chakra Agarwal • 9 months ago

true! loved your response... here is an explanation for someone who wants to understand further.

http://techieme.in/adding-numb...

^ | v • Reply • Share ›

Nishith Goswami • a year ago

If in case size of link list is not same in that case..can't we update new nodes with data value zero and follow the same recursive approach.

For Ex:
L1 : 1->2->3->4
L2 : 2->3->4

Changed L2 like : 0->2->3->4

and follow same recursive approach.

1 ^ | v • Reply • Share ›

Vj • a year ago

Why do you need to pass cur in addCarryToRemaining(head1, cur, &carry, result) ?

^ | v • Reply • Share ›

sneha ➦ Vj • a year ago

To stop the recursion. In addCarryToRemaining(), we are starting from head1 and traversing head1->next till head1 gets equal to cur.

• Reply • Share ›

Ashish Jaiswal • a year ago

simple code ...No recursion...iterative...
complexity:O(m+n)
sapce: O(1)

```
#include<stdio.h>
#include<stdlib.h>
struct node;
void push(struct node**,int);
void print(struct node*);
struct node*add(struct node*,struct node*);
struct node*newnode();

typedef struct node
{
int data;
struct node*next;
}Node;

int main()
```

see more

1 • Reply • Share ›

Rahul Assassin ➦ Ashish Jaiswal • 9 months ago

Your code is correct but wouldnt solve the purpose for which we need the linked list for addition. We needed linked list because there can be a scenario where any primitive data type is not able to hold the number . So for those cases , linked list come in handy as we can take number in array and place each digit in linked list and add it. But in your code , you again convert it back to the number and store it in some primitive data type which defeats the purpose for which we originally used the linked list.

1 • Reply • Share ›

Ashish Jaiswal ➦ Ashish Jaiswal • a year ago

But it will have problem when given no is too big....so better take double or long double the variable in which we are storing the no...

• Reply • Share ›

Rachit Nagdev ➦ Ashish Jaiswal • 7 months ago

Rachit Nagdev ➦ Ashish Jaiswal • 7 months ago

for very large numbers which cannot be accommodated in even unsigned long long int data type, are handled using special data structures such as BigInteger Class of Java, this linked list approach is also another data structure for this purpose.

^ | ˅ • Reply • Share ›

mercs ➦ Ashish Jaiswal • 7 months ago

double or long double wont give u that much precision for say thousand or a hundred thousand digits as would be achieved by the above method!!

^ | ˅ • Reply • Share ›

Harit • a year ago

package com.learner.questions.others;

public class LinkedListSum {

public static void main(final String[] args) {

final Node numberOne = buildList(new int[] { 2, 4 });

final Node numberTwo = buildList(new int[] { 1, 8, 9 });

final Node sum = getSum(numberOne, numberTwo);

print(numberOne);

print(numberTwo);

print(sum);

}

private static int getLength(Node node) {

see more

^ | ˅ • Reply • Share ›

Ashish • a year ago

Another approach can be traverse the lists and get the integers. Now add these and make a new Linked List. Efficient with respect to time as well as space.

^ | ˅ • Reply • Share ›

Guest ➦ Ashish • a year ago

What will you do when the integers are really very large that can't be stored in the variable.

4 ^ | ˅ • Reply • Share ›

Ashish Jaiswal ➦ Guest • a year ago

We can take Double or long Double in case no is big...

^ | ˅ • Reply • Share ›

Techie Me ➦ Ashish Jaiswal • 9 months ago

You need to see this to understand the need of this question
http://techieme.in/adding-numb...

^ | ˅ • Reply • Share ›

zxcvmnb • a year ago

If recursion is possible solution, using explicit array(stack) or better still converting lists to ints should also be solution. :)

^ | ˅ • Reply • Share ›

jimmy • a year ago

what if lists are..1>2>3 AND 9>9>9....then ans should be 1>2>2>2 but above code will give 1>2>2..please check it out

^ | ˅ • Reply • Share ›

Nayan • a year ago

Mine is simple implementation

Link of code : http://ideone.com/IB9YbA

^ | ˅ • Reply • Share ›

Kim Jong-il ➦ Nayan • a year ago

Is it? :D :D I doubt.:P

^ | ˅ • Reply • Share ›

Rajat Shrivastava • a year ago

Java implementation of above algorithm

http://ideone.com/7T0LJU

1 ^ | ˅ • Reply • Share ›

kamran siddique • a year ago

success

http://ideone.com/ZHs8kH

^ | ˅ • Reply • Share ›

Ashish Jaiswal ➦ kamran siddique • a year ago

Order is not maintained , you just printing reverse of linked at last while no should

Order is not maintained...you just printing reverse of linked at last while no should be of same order as sum is..
void rprint(struct node*head){
if(head==NULL)
return ;
rprint(head->next);
printf("%d ",head->data);
}
• Reply • Share ›

Guest • a year ago
Reverse both , add them starting from head to tail. then reverse the resulting list???? Is it correct
2 • Reply • Share ›

pol Guest • a year ago
condition is u can't modify the original list.so reversing is not possible.
• Reply • Share ›

sk pol • a year ago
reverse all again.
• Reply • Share ›

Pankaj sk • a year ago
When we say we can't modify the DS, we imply it can be a read-only DS.
• Reply • Share ›

crankyCoder Guest • a year ago
would be correct...yes
• Reply • Share ›

faiz • a year ago
How about traversing the lists and storing them in two stacks. Keep popping from both until one of them becomes empty .. The addition logic is same
• Reply • Share ›

Ashish Jaiswal faiz • a year ago
yes....but extra space for both lists in stack
1 • Reply • Share ›

<HoldOnLife!#> • a year ago
I found it analogous to http://www.geeksforgeeks.org/s... is it?

• Reply • Share ›

Load more comments

@geeksforgeeks, Some rights reserved Contact Us! About Us! Advertise with us!