Alternating split of a given Singly Linked List

Write a function AlternatingSplit() that takes one list and divides up its nodes to make two smaller lists 'a' and 'b'. The sublists should be made from alternating elements in the original list. So if the original list is 0->1->0->1->0->1 then one sublist should be 0->0->0 and the other should be 1->1->1.

Method 1(Simple)
The simplest approach iterates over the source list and pull nodes off the source and alternately put them at the front (or beginning) of 'a' and b'. The only strange part is that the nodes will be in the reverse order that they occurred in the source list. Method 2 inserts the node at the end by keeping track of last node in sublists.

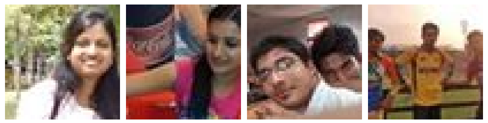
```
/*Program to alternatively split a linked list into two halves */
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>

/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* pull off the front node of the source and put it in dest */
void MoveNode(struct node** destRef, struct node** sourceRef) ;

/* Given the source list, split its nodes into two shorter lists.
  If we number the elements 0, 1, 2, ... then all the even elements
  should go in the first list, and all the odd elements in the second.
  The elements in the new lists may be in any order. */
void AlternatingSplit(struct node* source, struct node** aRef,
```

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** bRef)
{
  /* split the nodes of source to these 'a' and 'b' lists */
  struct node* a = NULL;
  struct node* b = NULL;

  struct node* current = source;
  while (current != NULL)
  {
    MoveNode(&a, &current); /* Move a node to list 'a' */
    if (current != NULL)
    {
       MoveNode(&b, &current); /* Move a node to list 'b' */
    }
  }
  *aRef = a;
  *bRef = b;
}

/* Take the node from the front of the source, and move it to the fron
   It is an error to call this with the source list empty.

   Before calling MoveNode():
   source == {1, 2, 3}
   dest == {1, 2, 3}

   Affter calling MoveNode():
   source == {2, 3}
   dest == {1, 1, 2, 3}
*/
void MoveNode(struct node** destRef, struct node** sourceRef)
{
  /* the front source node  */
  struct node* newNode = *sourceRef;
  assert(newNode != NULL);

  /* Advance the source pointer */
  *sourceRef = newNode->next;

  /* Link the old dest off the new node */
  newNode->next = *destRef;

  /* Move dest to point to the new node */
  *destRef = newNode;
}

/* UTILITY FUNCTIONS */
```

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

```
/* Function to insert a node at the begining 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;
}

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

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

  /* Let us create a sorted linked list to test the functions
   Created linked list will be 0->1->2->3->4->5 */
  push(&head, 5);
  push(&head, 4);
  push(&head, 3);
  push(&head, 2);
  push(&head, 1);
  push(&head, 0);

  printf("\n Original linked List:  ");
  printList(head);
  /* Remove duplicates from linked list */
  AlternatingSplit(head, &a, &b);

  printf("\n Resultant Linked List 'a' ");
  printList(a);

  printf("\n Resultant Linked List 'b' ");
  printList(b);

  getchar();
  return 0;
}
```

Time Complexity: O(n) where n is number of node in the given linked list.

Method 2(Using Dummy Nodes)
Here is an alternative approach which builds the sub-lists in the same order as the source list. The code uses a temporary dummy header nodes for the ‘a’ and ‘b’ lists as they are being built. Each sublist has a “tail” pointer which points to its current last node — that way new nodes can be appended to the end of each list easily. The dummy nodes give the tail pointers something to point to initially. The dummy nodes are efficient in this case because they are temporary and allocated in the stack. Alternately, local “reference pointers” (which always points to the last pointer in the list instead of to the last node) could be used to avoid Dummy nodes.

```
void AlternatingSplit(struct node* source, struct node** aRef,
                                            struct node** bRef)
{
  struct node aDummy;
  struct node* aTail = &aDummy; /* points to the last node in 'a' */
  struct node bDummy;
  struct node* bTail = &bDummy; /* points to the last node in 'b' */
  struct node* current = source;
  aDummy.next = NULL;
  bDummy.next = NULL;
  while (current != NULL)
  {
    MoveNode(&(aTail->next), &current); /* add at 'a' tail */
    aTail = aTail->next; /* advance the 'a' tail */
    if (current != NULL)
    {
      MoveNode(&(bTail->next), &current);
      bTail = bTail->next;
    }
```

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...

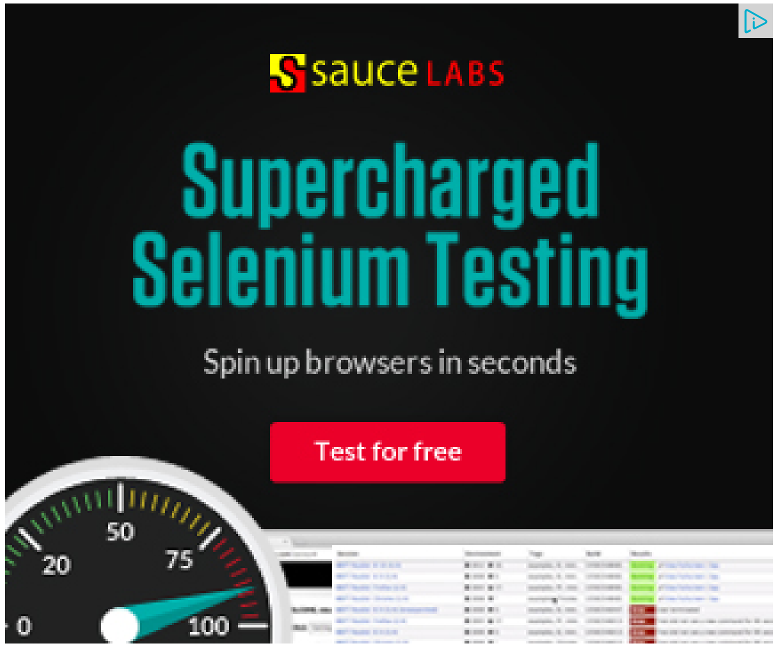
Find subarray with given sum · 2 hours ago

```
    }
    *aRef = aDummy.next;
    *bRef = bDummy.next;
}
```

Time Complexity: O(n) where n is number of node in the given linked list.

Source: 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.



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

2 Tweet 0 0

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

64 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Ateet Kakkar • 12 days ago

//In main(), alt_split(head, &head1, &head2, 0) will be passed where head1 an
Complexity is O(n).

```
void alt_split(struct node *curr, struct node **pfirst1, struct node **pfirst2, int x
if(!curr)
return;
alt_split(curr->next, pfirst1, pfirst2, 1-x);
if(x){
curr->next = *pfirst2;
*pfirst2 = curr;
}
else{
curr->next = *pfirst1;
*pfirst1 = curr;
}
}
```

• Reply • Share ›

Mohan Rajoria • a month ago

An alternate way is the same as deleting alternative nodes. We can add delete

get solution for both the problem 1.Delete alternative nodes.
2. Split alternative nodes into two list.

Reply • Share ›

Ankur Teotia • a month ago

an easier way to do this would be to traverse the list and have a counter intege
increment whenever a node would get traversed.

now if the counter is even then put the node in one list and if it is odd , put it in

the complete code -> http://ideone.com/CEN0ss

here's the code snippet of the modifield alternating split function.

```
struct node* b1 = NULL;

struct node* a1= NULL;

void AlternatingSplit(struct node* source, struct node** aRef,

struct node** bRef)

{

/* split the nodes of source to these 'a' and 'b' lists */
```

see more

Reply • Share ›

Vishal • 2 months ago

My below post will give the list as required but having the reference of the origi
So below program is just to print the expected output...
no new memory allocations...

Reply • Share ›

Vishal • 2 months ago

```
void AlternatingSplit(Node *head, Node **evenlist,Node **oddlist)
{
Node *temp = head;
Node *pevenlist = *evenlist;
Node *poddlist = *oddlist;
int i =0;
while(temp)
{
if(i%2 == 0)
{
if(pevenlist){
pevenlist->next= temp;
pevenlist = pevenlist->next;}
else{

pevenlist = temp;
*evenlist = pevenlist;
}
```

see more

• Reply • Share ›

Himanshu Dagar • 3 months ago

whole source code from dummy variable concept is here(at below link)

http://ideone.com/Xw7Wh1

• Reply • Share ›

bhavesh • 4 months ago

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

```
#include<stdlib.h>

#include<assert.h>

/* Link list node */

struct node

{

int data;

struct node* next;

};

/* pull off the front node of the source and put it in dest */

void MoveNode(struct node** destRef, struct node** sourceRef) ;
```

see more

^ | v • Reply • Share ›

Marsha Donna • 4 months ago

http://ideone.com/JvD1UP
can this be simplified further

^ | v • Reply • Share ›

Vikash876 • 4 months ago

```
node * list_split_alternate(node *head)
{
node *sav, *retnode = head->next;
while(head->next!=NULL)
{
```

sav = head->next;
head->next = head->next->next;
head = sav;
}
sav->next = NULL;
return retnode;
}

• Reply • Share ›

ankit • 8 months ago

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

struct treeNode
{
    int val;
    struct treeNode *next;

};
struct treeNode *root=NULL;

struct treeNode *a=NULL;
struct treeNode *b=NULL;

struct  treeNode*  getNode()
{
    return (struct treeNode*)malloc(sizeof(struct treeNode));
}
```

see more

• Reply • Share ›

CP • 8 months ago

GP • 8 months ago

```
[sourcecode language="C"]

/* simple approach*/

void alternatingSplit(struct node** head_ref,struct node** head1,struct node**

{

if(*head_ref==NULL||(*head_ref)->next==NULL)

return;

struct node* current=*head_ref;

struct node* nNext,*next;

*head1=*head_ref;

*head2=(*head_ref)->next;

while(current!=NULL&&nNext!=NULL)
```

see more

Reply • Share ›

Arshan Qureshi • 9 months ago

```
void AlternatingSplit(struct node* source, struct node** aRef, struct node** bR
{

struct node* a=NULL;.
struct node* b=NULL;.

if(source) {.
*aRef=source;.
*bRef=source->next;.
a=source;.
b=source->next;.
while(b).
{.

a->next=b->next;.
a=b;.
b=b->next;.
}.

}.

}
```

^ | ⌄ • Reply • Share ›

Soumya Sengupta • 10 months ago

```
split_list(struct node*head)
{
split_listUtil(struct node*head)
{
if(head==null)
return;
temp=head->next;
if(temp==null)
return;

struct node *new=(struct node*)malloc(sizeof(stuct node));
new=temp;
head->next=temp-next;
new->next=split_listUtil(head->next);
return new;
}
return head;
```

}

Reply • Share ›

Soumya Sengupta • 10 months ago

```
split_list(struct node*head)
{
split_listUtil(struct node*head)
{
if(head==null)
return;
temp=head->next;
if(temp==null)
return;

struct node *new=(struct node*)malloc(sizeof(stuct node));
new=temp;
head->next=temp-next;
temp->next=split_listUtil(head->next);
return temp;
}
return head;
}
```

Reply • Share ›

Deepak • 10 months ago

Here is third method.. just remove alternate node from original list and add to a
sublists.. original list is not preserve..

```
/* Dividing a list into two sbulists where each list has
```

alternate element of original list and only assumption made
is we don't need original list anymore */

```
#include
#include
typedef struct node
{
int data;
struct node *link;
}list;
void insert(list **,int);
void show(list *);
```

see more

^ | ⌄ • Reply • Share ›

omguptanitdgp • 11 months ago
@geeksforgeeks
here is an easy solution
take two pointer ,one points to first node in the list and other
points to second node initiaaly .now advance both pointer by next->next(mean
linked lists.

please inform me whether i m correct or not

^ | ⌄ • Reply • Share ›

Marsha Donna ➦ omguptanitdgp • 4 months ago
@omguptanitdgp
see my code given above implements the same algorithm

^ | ⌄ • Reply • Share ›

Himanshu • 11 months ago
/* we can use a bool variable which when true add nodes of source list to a ar

#include
#include
#include
#include
using namespace std;

/* Link list node */
struct node
{
int data;
struct node* next;
};

void AlternatingSplit(struct node* source, struct node** aRef,
struct node** bRef)
{
bool var=true;

see more

Reply • Share ›

lizard • 11 months ago
I think the following code is fairly easy to understand and to code also and it dc

```
void alternate(node *head,node **one,node **two)
{
    node *a=head,*b=head->nxt;
    *one=a;
    *two=b;
    while(b!=NULL)
    {
        a->nxt=b->nxt;
        a=b;
        b=a->nxt;
    }
}
```

see more

3 • Reply • Share ›

aishlnch lizard • a month ago

can anyone explain it to me please???

• Reply • Share ›

Vibhu Tiwari • 11 months ago

@GeeksForGeeks i think it can also be done by the following method. It uses
a linked list. Now before freeing the node copy the node's data to another linke
nodes.At last just reverse the new linked list that we have made to get the alte
In the end just display the two lists.Below is the code

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

typedef struct node node_t;
struct node
{
    int data;
    node_t *next;
};
node_t *reverse(node_t *n)
{
```

```
{
    node_t *new_root=NULL;
    while(n)
```

see more

^ | ˅ • Reply • Share ›

Ankur • 11 months ago

```
void split(node *temp,node **one,node **two)
{
if(temp==NULL)
{
*one =NULL;
*two=NULL;
return ;
}
node *ptr1=(node *)malloc(sizeof(node));
node *ptr2=(node *)malloc(sizeof(node));
*one=ptr1;
*two=ptr2;
ptr1->val=temp->val;
if(temp->next)
{
ptr2->val=temp->next->val;
split(temp->next->next,&(ptr1->next),&(ptr2->next));
return ;
}
ptr1->next=NULL;
*two=NULL;
}
```

^ | ˅ • Reply • Share ›

Hanish Bansal • 11 months ago

Here is one simple implementation without using Movenode function or dumm

```
void AlternatingSplit(struct node* source, struct node** aRef,
struct node** bRef)
{
struct node *a, *b=NULL;
a=source;
if(a)
b=a->next;
*aRef = a;
*bRef = b;
while ( a && b )
{
a->next=b->next;
a=a->next;
if(a)
b->next=a->next;
b=b->next;
}
}
```

2 ^ | ˅ • Reply • Share ›

mualloc ➦ Hanish Bansal • 3 months ago

First of all, thank you very much for your neat solution and also, I have
lists. Sorry for the identifier names.

```
typedef struct node
{
int data;
struct node *prev;
struct node *next;
} node;
```

```
#define DATA(p) ((p)->data)
#define NEXT(p) ((p)->next)
#define PREV(p) ((p)->prev)

void split ( node* head, node **first, node **second )
{
node* firstCurrent = head;
node* secondCurrent = NULL;
node* dummyforbprev = NULL;
```

see more

Reply • Share ›

vikasnitt Hanish Bansal • 10 months ago

Excellent approach hanish..:)

1 • Reply • Share ›

lakshay Hanish Bansal • 10 months ago

Exactly!
Great approach hanish :D

1 • Reply • Share ›

Ronny Hanish Bansal • 11 months ago

Even I thought this approach on reading this question.
This is a neat and simple algorithm without using additional notes.
Thanks for providing with code.

1 • Reply • Share ›

hunter • 11 months ago

```
while(p&&p->next)
{
temp=p->next;
```

p->next=temp->next;
temp->next=NULL;
if(second==NULL)
second=q=temp;
else
{
q->next=temp;
q=temp;
}
p=p->next;
}

Reply • Share ›

ultimate_coder • 11 months ago

I think its simpler to understand and easier one.
head,head1 and head2 global variables.

1.Fisrt list head is head1.
2.Second list head is head2.
3.Original list head is head.

```
 void splitlist(void)
{
    unsigned count=1;
    struct node *temp=head;
               //iterative loop
    while(temp)
    {
        if(count & 1)        //check for even or odd
            push(&head1,temp->data);   //odd to first list
        else push(&head2,temp->data);  //even to second list
        temp=temp->next;
        ++count;
    }
}
```

⌃ | ⌄ • Reply • Share ›

Marsha Donna ➦ ultimate_coder • 4 months ago

i think the question is to split 1 list into 2 separate lists..not create 2 list
me if i m wrong

⌃ | ⌄ • Reply • Share ›

beginner ➦ ultimate_coder • 11 months ago

very easy approach compared to al...thank u....

⌃ | ⌄ • Reply • Share ›

abhishek08aug • a year ago

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

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

void insert_node(struct node ** head_ref, int value) {
  struct node * head=*head_ref;
  struct node * new_node=NULL;
  if(head==NULL) {
    new_node=(struct node *)malloc(sizeof(struct node));
    new_node->data=value;
    new_node->next=NULL;
    *head_ref=new_node;
  } else {
    insert_node(&head->next, value);
```

see more

Reply • Share ›

yatharth.sharma • a year ago

```
void alternate()
{

head1=head;
head2=head->next;
 node *t1=head1,*t2=head2;
while(t1!=NULL && t2!=NULL)
 {       if(t1->next==NULL)
            break;
       t1->next=t2->next;
       t1=t1->next;
       if(t2->next==NULL)
       break;
       t2->next=t1->next;
       t2=t2->next;
 }

}
```

Reply • Share ›

Shivam • a year ago

void AlternatingSplit(struct node *source, struct node *a, struct node *b) // whe

{

if(a==NULL).

return;.

while(a->next!=NULL&&b->next!=NULL){

a->next=a->next->next;

a=a->next;

b->next=b->next->next;

b=b->next;

}

if(a->next!=NULL)

a->next=NULL;

}

Reply • Share ›

Nikin Kumar Jain • a year ago

```
 void addNode(node **sr, int data)
{
        node *temp = new node;
        temp->data = data;
        temp->next = NULL;
        if(*sr == NULL)
        {
                *sr = temp;
                return;
        }
        else
        {
                while((*sr)->next != NULL)
                while((*sr)->next != NULL)
                        *sr = (*sr)->next;
                (*sr)->next = temp;
        }
}
```

see more

^ | ⌄ • Reply • Share ›

ramkumarp • a year ago

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

void SplitAlt (Node* head, Node*& hx, Node*& hy){

  if (head == NULL || head -> next == NULL){
        hx = head;
        hy = NULL;
        return;
  }
  hx = head;
  hy = head->next;

  Node *h1 = head;

  Node* h2 = head->next;

  while ( h1 != NULL && h2 != NULL){
```

see more

^ | ⌄ • Reply • Share ›

Arindam Sanyal • a year ago

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

struct node{
int info;
struct node *link;.
};

struct node * addtoempty(struct node *, int);.
struct node * addtoend(struct node *, int);.
void display(struct node *);.
void splitlink(struct node *);.

void main(){
clrscr();
struct node *start=NULL;.
int num, d;
printf("n enter the number of nodes...");.

see more

Reply • Share ›

sush • a year ago

There is no need to do so much. Alternating split can be written in this simple

```
void AlternatingSplit(struct node* h, struct node** a,struct node** b
                                                        {
                                                                if(h==
                                                                {
                                                                }
                                                                *a=h;
                                                                Alter
                                                        }
```

^ | v • Reply • Share ›

PG • a year ago

Any comment on this solution?

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

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

void pushfront(struct node **, int);
void print_list(struct node*);

void split_alternate_keep_original_list(struct node *, struct node **
int main(){
        struct node *head = NULL;
        struct node *head2 = NULL;
```

see more

^ | v • Reply • Share ›

himanshu • a year ago

can be done using recursion like this easily...

```
[sourcecode language="C++"]
void alternate_split(node *start,node *&first,node *&second)
{
if(start==NULL)
{
first=NULL;second=NULL;return;
}
if(start->n==NULL)
{
first=start;second=NULL;return;
}
first=start;
second=start->n;
alternate_split(start->n->n,first->n,second->n);
}
```

Reply • Share ›

Gurusimhe • 2 years ago

What is the problem here in this code?
When I give Input 1-->0-->1-->0 it prints 1-->0 and 0-->1
instead of 1-->1 and 0-->0

```
[sourcecode language="C++"]
void Altsplit( list *head, list **h1, list **h2)
{
if(head==NULL) return;
list *p = head;
*h1 = head; *h2 = head->next;

while(p && p->next)
{
list *t = p->next;
```

p->next = t->next;
p = t;
}
}

Reply • Share ›

Gurusimhe Gurusimhe • 2 years ago

Its working fine.My list was being modified.Please check if it has any ot
Thanks,
Gurusimhe

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

Reply • Share ›

vick • 2 years ago

its a recursive approach...n simple enough..
plz comment if anything found wrong with the code..

```
void alternateSplit(node *q,node **a,node **b)
{
        if(q==NULL)return;

        node *aa,*bb;
        aa=q;
        bb=q->link;

        if(bb!=NULL)
        aa->link=bb->link;

        if(aa->link!=NULL)
        bb->link=aa->link->link;

        alternateSplit(aa->link,a,b);

        (*a)=aa;
        (*b)=bb;

        return;

}
```

^ | ⌄ • Reply • Share ›

Manish Kumar • 2 years ago

```
 void AlternatingSplit(struct node* head, struct node** aRef,
                            struct node** bRef)
{
  /* split the nodes of source to these 'a' and 'b' lists */
  struct node* head1 = head;
  struct node* head2 = head->next;
  struct node *s1 = head1;
  struct node *s2 = head2;
  struct node *p1,*p2;

  while(s1!=NULL && s2!=NULL && s2->next!=NULL && s2->next->next!=NUL
      p1=s2->next;
      p2=s2->next->next;
      s1->next=p1;
      s2->next=p2;
      s1=p1; s2=p2;
  }
```

see more

Reply • Share ›

shen • 2 years ago

Recursive approach....
function will be called in giving start of the linked list and pointer to the new sta

```
 void alternativeSplit(node *s,node **s1, node **s2)
{
    if(s==NULL)
    {
        *s1=NULL;
        *s2=NULL;
        return;
    }
    node *a,*b,*temp;
    *s1=s;
    *s2=s->link;
    if(s->link!=NULL)
    {
        alternativeSplit(s->link->link,&a,&b);
        temp=s->link;
        s->link=a;
        temp->link=b;
    }
}
```

Reply • Share ›

Ankur Garg • 2 years ago

A Recursive solution in O(n)

```
 void AlternatingSplit(struct node* &head, struct node* &A,struct nod
  if(! head ){
    A=NULL;
    B=NULL;
    return;
  }
  if(! head->next ){
    A=head;
    B=NULL;
    return;
  }
  node* current=head;
  A=current;
  B=current->next;
  current=current->next->next;
  AlternatingSplit(current,A->next,B->next);
}
```

^ | ⌄ • Reply • Share ›

Sudha Malpeddi • 3 years ago

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

void copy(struct node *p, struct node **q)
{
        if(p!=NULL)
        {
                *q=(struct node*)malloc(sizeof(struct node));
                (*q)->num=p->num;
                (*q)->next=NULL;
                copy(p->next, &((*q)->next));
                (*q)->arbit=p->arbit;
        }
}
```

^ | ⌄ • Reply • Share ›

kamlesh meghwal • 3 years ago

//p Points To Second Node And q Points To First Node
q->next=p->next;
while(p&&q)
{
if(q->next)
p->next=q->next->next
else
p->next=Null
}
increament p And q i.e p=p->next,q=q->next

^ | ⌄ • Reply • Share ›

Venki • 3 years ago

For Method 1, iterative code without "MoveNode" function. As @fuzz pointed p create confusion.

```
 void AlternatingSplit(struct node* source, struct node** aRef,
struct node** bRef)
{
    struct node** current_list = aRef;
    struct node* current_node = source;
    struct node* next_node = NULL;

    while (current_node != NULL)
    {
        next_node = current_node->next;
        current_node->next = (*current_list);
        (*current_list) = current_node;
        current_node = next_node;

        current_list = (current_list == aRef) ? bRef : aRef;
    }
}
```

Reply • Share ›

Venki ➦ Venki • 3 years ago

Removing if (ternary) condition inside the loop

```
 void AlternatingSplit(struct node* source, struct node** aRef,
struct node** bRef)
{
    unsigned exor;
    struct node** current_list = aRef;
    struct node* current_node = source;
    struct node* next_node = NULL;

    /* Smarty compiler, don't warn me,
       I know what I am doing */
    exor = (unsigned)aRef ^ (unsigned)bRef;

    while (current_node != NULL)
    {
        next_node = current_node-&gt;next;
        current_node-&gt;next = (*current_list);
        (*current_list) = current_node;
```

```
            current_node = next_node;

            current_list = (struct node**)((unsigned)current_list 
        }
}
```

Reply • Share ›

Load more comments

Subscribe   Add Disqus to your site