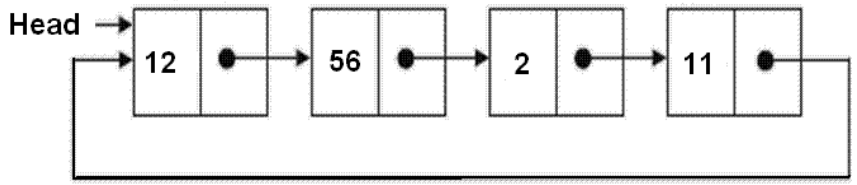
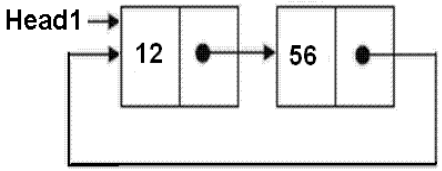
Split a Circular Linked List into two halves

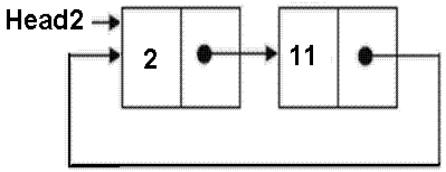
Asked by Bharani



Original Linked List

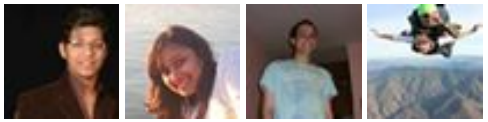


Result Linked List 1



Result Linked List 2

Thanks to Geek4u for suggesting the algorithm.



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

1) Store the mid and last pointers of the circular linked list using tortoise and hare algorithm.
2) Make the second half circular.
3) Make the first half circular.
4) Set head (or start) pointers of the two linked lists.

In the below implementation, if there are odd nodes in the given circular linked list then the first result list has 1 more node than the second result list.

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

/* structure for a node */
struct node
{
  int data;
  struct node *next;
};

/* Function to split a list (starting with head) into two lists.
   head1_ref and head2_ref are references to head nodes of
    the two resultant linked lists */
void splitList(struct node *head, struct node **head1_ref,
                                          struct node **head2_ref)
{
  struct node *slow_ptr = head;
  struct node *fast_ptr = head;

  if(head == NULL)
    return;

  /* If there are odd nodes in the circular list then
     fast_ptr->next becomes head and for even nodes
     fast_ptr->next->next becomes head */
  while(fast_ptr->next != head &&
         fast_ptr->next->next != head)
  {
     fast_ptr = fast_ptr->next->next;
     slow_ptr = slow_ptr->next;
  }

 /* If there are even elements in list then move fast_ptr */
  if(fast_ptr->next->next == head)
    fast_ptr = fast_ptr->next;
```


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

```
  /* Set the head pointer of first half */
  *head1_ref = head;

  /* Set the head pointer of second half */
  if(head->next != head)
    *head2_ref = slow_ptr->next;

  /* Make second half circular */
  fast_ptr->next = slow_ptr->next;

  /* Make first half circular */
  slow_ptr->next = head;
}

/* UTILITY FUNCTIONS */
/* Function to insert a node at the begining of a Circular
   linked lsit */
void push(struct node **head_ref, int data)
{
  struct node *ptr1 = (struct node *)malloc(sizeof(struct node));
  struct node *temp = *head_ref;
  ptr1->data = data;
  ptr1->next = *head_ref;

  /* If linked list is not NULL then set the next of
    last node */
  if(*head_ref != NULL)
  {
    while(temp->next != *head_ref)
      temp = temp->next;
    temp->next = ptr1;
  }
  else
     ptr1->next = ptr1; /*For the first node */

  *head_ref = ptr1;
}

/* Function to print nodes in a given Circular linked list */
void printList(struct node *head)
{
  struct node *temp = head;
  if(head != NULL)
  {
    printf("\n");
    do {
      printf("%d ", temp->data);
```

```
      temp = temp->next;
    } while(temp != head);
  }
}

/* Driver program to test above functions */
int main()
{
  int list_size, i;

  /* Initialize lists as empty */
  struct node *head = NULL;
  struct node *head1 = NULL;
  struct node *head2 = NULL;

  /* Created linked list will be 12->56->2->11 */
  push(&head, 12);
  push(&head, 56);
  push(&head, 2);
  push(&head, 11);

  printf("Original Circular Linked List");
  printList(head);

  /* Split the list */
  splitList(head, &head1, &head2);

  printf("\nFirst Circular Linked List");
  printList(head1);

  printf("\nSecond Circular Linked List");
  printList(head2);

  getchar();
  return 0;
}
```

Time Complexity: O(n)

Please write comments if you find any bug in above code/algorithm, or find other ways to solve the same problem

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 49 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

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

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

16 Comments GeeksforGeeks

Sort by Newest

Join the discussion

Join the discussion…

saurabh • a month ago

I think the condition while(fast_ptr->next != head && fast_ptr->next->next != head) is wrong there should be OR condition.. Let me

• Reply • Share ›

saurabh → saurabh • a month ago

Its correct.. got the logic.. you can delete the post.

• Reply • Share ›

Karshit • 10 months ago

My Code.. hope you find it useful..

```
 #include <iostream>

using namespace std;

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

node *create(int n)
{
    if (n == 0)
        return NULL;

    node *head = new node();
    cin >> (head -> data);
```

see more

• Reply • Share ›

Jitendra.BITS • 10 months ago

What's the use of this line?
Is it to check that the list has just one node or not??

```
if(head->next != head)
```

• Reply • Share ›

suyash → Jitendra.BITS • 5 months ago

when there is only one node , you cann't split list into 2 hence there will set head2...

• Reply • Share ›

neham • a year ago

Here is another way of splitting the circular linked list having O(n) complexity. I split it.

```
stnd* ptr = root;
float count = 1;
int half;

while( ptr->next != root) {
++count;
ptr = ptr->next;
}

ptr = root;
half = ceil(count / 2);
cout <<endl <<count <<"\t" <<half < 1 ) {
--half;
ptr = ptr->next;
```

```
}

*head1 = root;
*head2 = ptr->next;
ptr->next = root;
ptr = *head2;

while( ptr->next != root) {
ptr = ptr->next;
}
ptr->next = *head2;
```

• Reply • Share ›

tuhin@jucse • 2 years ago

tortoise and hare algorithm is amazing\m/

• Reply • Share ›

Nitin Pallindrome ➦ tuhin@jucse • 8 months ago

hmmm...

• Reply • Share ›

checkitout • 3 years ago

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

int nodes=0;
struct node
{
    int data;
    struct node *next;
    struct node *prev;
```

```
};
void insnode(int dt,struct node **hn,struct node **en)
{
     nodes++;
     struct node *newnode=(struct node *)malloc(sizeof(struct node));
     newnode->data=dt;
     newnode->prev=(*en);
```

see more

Reply • Share ›

aman • 3 years ago

Okay, it seems like algo is moving slow pointer by the distance half of other or at the middle of the list while fast_ptr traverses it complete and thus to find poi not getting it right way.

Reply • Share ›

aman • 3 years ago

Can anybody please elaborate here more on tortoise and hare algorithm. Thank you!

Reply • Share ›

anonomus • 3 years ago

while(fast_ptr->next != head &&
fast_ptr->next->next != head)

Should be:
while(**slow_ptr**->next != head &&
fast_ptr->next->next != head)

Reply • Share ›

bunty • 4 years ago

hey dosto....
here is another one iteratively...

```
 split(node* head, node** fast, node** slow)
{
 node *temp = NULL;

 if (head->next==head)
  {/* Head is pointing itself, i.e., empty list*/
   *fast = NULL;
   *slow = head;
   return;
  }

 *fast = head->next;
 *slow = head;

 while(*fast!=head)
```

see more

• Reply • Share ›

Aditya • 4 years ago

Hi there,

I have a question with the 'push' function. Shouldn't last line in the function be \
look something like this..

```
if(*head_ref != NULL)
  {
    while(temp->next != *head_ref)
```

```
    temp = temp->next;
  temp->next = ptr1;
}
else
  { ptr1->next = ptr1; /*For the first node */
    *head_ref = ptr1;
  }
```

• Reply • Share ›

Sandeep → Aditya • 4 years ago

@Aditya: The push() function inserts a node at the beginning of circula
always becomes head. The last line (*head_ref = ptr1) changes head t
which must be done for every inserted node.

• Reply • Share ›

Aditya → Sandeep • 4 years ago

@Sandeep thanks it makes the understanding easier. Btw Kud
learned a lot in past two days about linked lists and pointers in g

• Reply • Share ›