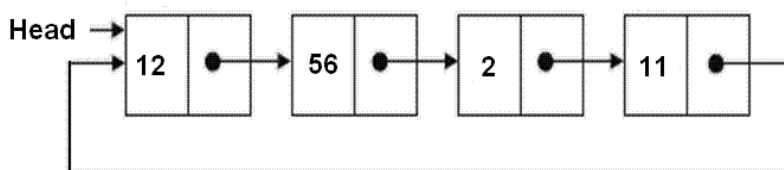
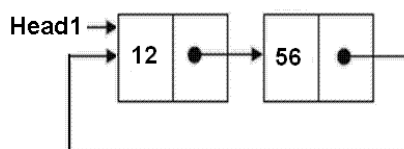


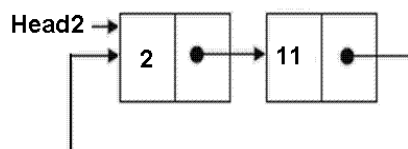
Split a Circular Linked List into two halves

Asked by [Bharani](#)

Original Linked List



Result Linked List 1



Result Linked List 2

Thanks to [Geek4u](#) for suggesting the algorithm.

- 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;

    /* 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 beginning of a Circular
   linked list */
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;
}
```

[Run on IDE](#)

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



A Free Course From Microsoft

Querying with Transact-SQL

edX
edx.org

Enroll now
Self-Paced

36 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)

2

Average Difficulty : **2/5.0**
Based on **3** vote(s)

☐

Add to TODO List

☐

Mark as DONE

Like Share Be the first of your friends to like this.

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

@geeksforgeeks, Some rights reserved

Contact Us!

About Us!

Advertise with us!