Write a function to reverse a linked list

Iterative Method

Iterate trough the linked list. In loop, change next to prev, prev to current and current to next.

Implementation of Iterative Method

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

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

/* Function to reverse the linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}

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

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

     push(&head, 20);
     push(&head, 4);
     push(&head, 15);
     push(&head, 85);

     printList(head);
     reverse(&head);
     printf("\n Reversed Linked list \n");
     printList(head);
     getchar();
}
```

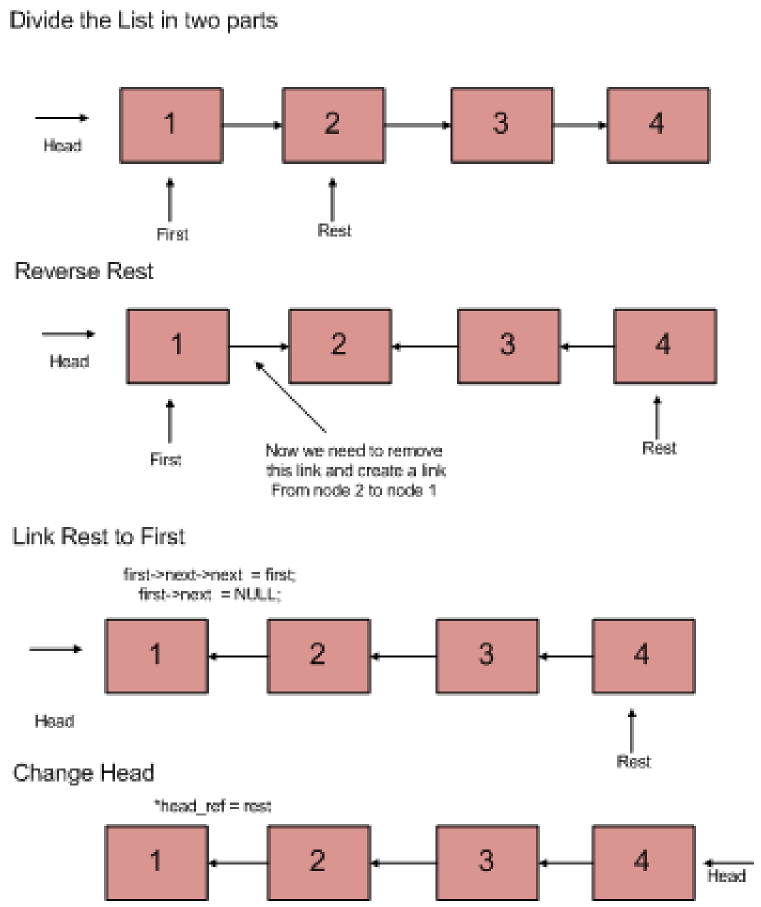
Run on IDE

Time Complexity: O(n)

Space Complexity: O(1)

Recursive Method:

```
1) Divide the list in two parts - first node and rest of the linked list.
2) Call reverse for the rest of the linked list.
3) Link rest to first.
4) Fix head pointer
```



```
void recursiveReverse(struct node** head_ref)
{
    struct node* first;
    struct node* rest;

    /* empty list */
    if (*head_ref == NULL)
       return;

    /* suppose first = {1, 2, 3}, rest = {2, 3} */
    first = *head_ref;
    rest  = first->next;

    /* List has only one node */
    if (rest == NULL)
       return;

    /* reverse the rest list and put the first element at the end */
    recursiveReverse(&rest);
    first->next->next  = first;

    /* tricky step -- see the diagram */
    first->next  = NULL;

    /* fix the head pointer */
    *head_ref = rest;
}
```

Run on IDE

Time Complexity: O(n)

Space Complexity: O(1)

A Simpler and Tail Recursive Method

Below is C++ implementation of this method.

```
// A simple and tail recursive C++ program to reverse
// a linked list
#include<bits/stdc++.h>
using namespace std;

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

void reverseUtil(node *curr, node *prev, node **head);

// This function mainly calls reverseUtil()
// with prev as NULL
void reverse(node **head)
{
    if (!head)
        return;
    reverseUtil(*head, NULL, head);
}

// A simple and tail recursive function to reverse
// a linked list.  prev is passed as NULL initially.
void reverseUtil(node *curr, node *prev, node **head)
{
    /* If last node mark it head*/
    if (!curr->next)
    {
        *head = curr;

        /* Update next to prev node */
        curr->next = prev;
        return;
    }

    /* Save curr->next node for recursive call */
    node *next = curr->next;

    /* and update next ..*/
    curr->next = prev;

    reverseUtil(next, curr, head);
}

// A utility function to create a new node
node *newNode(int key)
{
    node *temp = new node;
    temp->data = key;
    temp->next = NULL;
    return temp;
}

// A utility function to print a linked list
void printlist(node *head)
{
    while(head != NULL)
    {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

// Driver program to test above functions
int main()
{
    node *head1 = newNode(1);
    head1->next = newNode(2);
    head1->next->next = newNode(2);
    head1->next->next->next = newNode(4);
    head1->next->next->next->next = newNode(5);
    head1->next->next->next->next->next = newNode(6);
    head1->next->next->next->next->next->next = newNode(7);
    head1->next->next->next->next->next->next->next = newNode(8);
    cout << "Given linked list\n";
    printlist(head1);
    reverse(&head1);
    cout << "\nReversed linked list\n";
    printlist(head1);
    return 0;
}
```

Run on IDE

Output:

```
Given linked list
1 2 2 4 5 6 7 8

Reversed linked list
8 7 6 5 4 2 2 1
```

Thanks to Gaurav Ahirwar for suggesting this solution.

References:

http://cslibrary.stanford.edu/105/LinkedListProblems.pdf

333 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)

2.7 Average Difficulty : **2.7/5.0**
Based on **10** vote(s)

☐ Add to TODO List
☐ Mark as DONE

Like Share 50 people like this. Be the first of your friends.

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