GeeksforGeeks
A computer science portal for geeks

Practice   IDE   Q&A   GeeksQuiz

Merge a linked list into another linked list at alternate positions

Given two linked lists, insert nodes of second list into first list at alternate positions of first list.

For example, if first list is 5->7->17->13->11 and second is 12->10->2->4->6, the first list should become 5->12->7->10->17->2->13->4->11->6 and second list should become empty. The nodes of second list should only be inserted when there are positions available. For example, if the first list is 1->2->3 and second list is 4->5->6->7->8, then first list should become 1->4->2->5->3->6 and second list to 7->8.

Use of extra space is not allowed (Not allowed to create additional nodes), i.e., insertion must be done in-place. Expected time complexity is O(n) where n is number of nodes in first list.

The idea is to run a loop while there are available positions in first loop and insert nodes of second list by changing pointers. Following are C and Java implementations of this approach.

C/C++

```
// C program to merge a linked list into another at
// alternate positions
#include <stdio.h>
#include <stdlib.h>

// A nexted list node
struct node
{
    int data;
    struct node *next;
};

/* Function to insert a node at the beginning */
void push(struct node ** head_ref, int new_data)
{
    struct node* new_node =
            (struct node*) malloc(sizeof(struct node));
    new_node->data  = new_data;
    new_node->next = (*head_ref);
    (*head_ref)  = new_node;
}

/* Utility function to print a singly linked list */
void printList(struct node *head)
{
```

```c
    struct node *temp = head;
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Main function that inserts nodes of linked list q into p at
// alternate positions. Since head of first list never changes
// and head of second list  may change, we need single pointer
// for first list and double pointer for second list.
void merge(struct node *p, struct node **q)
{
    struct node *p_curr = p, *q_curr = *q;
    struct node *p_next, *q_next;

    // While therre are avialable positions in p
    while (p_curr != NULL && q_curr != NULL)
    {
        // Save next pointers
        p_next = p_curr->next;
        q_next = q_curr->next;

        // Make q_curr as next of p_curr
        q_curr->next = p_next;  // Change next pointer of q_curr
        p_curr->next = q_curr;  // Change next pointer of p_curr

        // Update current pointers for next iteration
        p_curr = p_next;
        q_curr = q_next;
    }

    *q = q_curr; // Update head pointer of second list
}

// Driver program to test above functions
int main()
{
    struct node *p = NULL, *q = NULL;
    push(&p, 3);
    push(&p, 2);
    push(&p, 1);
    printf("First Linked List:\n");
    printList(p);

    push(&q, 8);
    push(&q, 7);
    push(&q, 6);
    push(&q, 5);
    push(&q, 4);
    printf("Second Linked List:\n");
    printList(q);

    merge(p, &q);

    printf("Modified First Linked List:\n");
    printList(p);

    printf("Modified Second Linked List:\n");
    printList(q);

    getchar();
    return 0;
```

}

Run on IDE

Java

```java
// Java program to merge a linked list into another at
// alternate positions
class LinkedList
{
    Node head;  // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }

    /* Inserts a new Node at front of the list. */
    void push(int new_data)
    {
        /* 1 & 2: Allocate the Node &
                  Put in the data*/
        Node new_node = new Node(new_data);

        /* 3. Make next of new Node as head */
        new_node.next = head;

        /* 4. Move the head to point to new Node */
        head = new_node;
    }

    // Main function that inserts nodes of linked list q into p at
    // alternate positions. Since head of first list never changes
    // and head of second list/ may change, we need single pointer
    // for first list and double pointer for second list.
    void merge(LinkedList q)
    {
        Node p_curr = head, q_curr = q.head;
        Node p_next, q_next;

        // While there are available positions in p;
        while (p_curr != null && q_curr != null) {

            // Save next pointers
            p_next = p_curr.next;
            q_next = q_curr.next;

            // make q_curr as next of p_curr
            q_curr.next = p_next; // change next pointer of q_curr
            p_curr.next = q_curr; // change next pointer of p_curr

            // update current pointers for next iteration
            p_curr = p_next;
            q_curr = q_next;
        }
        q.head = q_curr;
    }

    /* Function to print linked list */
```

```java
    void printList()
    {
        Node temp = head;
        while (temp != null)
        {
            System.out.print(temp.data+" ");
            temp = temp.next;
        }
        System.out.println();
    }

    /* Drier program to test above functions */
    public static void main(String args[])
    {
        LinkedList llist1 = new LinkedList();
        LinkedList llist2 = new LinkedList();
        llist1.push(3);
        llist1.push(2);
        llist1.push(1);

        System.out.println("First Linked List:");
        llist1.printList();

        llist2.push(8);
        llist2.push(7);
        llist2.push(6);
        llist2.push(5);
        llist2.push(4);

        System.out.println("Second Linked List:");

        llist1.merge(llist2);

        System.out.println("Modified first linked list:");
        llist1.printList();

        System.out.println("Modified second linked list:");
        llist2.printList();
    }
} /* This code is contributed by Rajat Mishra */
```

Run on IDE

Python

```python
# Python program to merge a linked list into another at
# alternate positions
class LinkedList(object):
    def __init__(self):
    # head of list
        self.head = None

    # Linked list Node
    class Node(object):
        def __init__(self, d):
            self.data = d
            self.next = None

    # Inserts a new Node at front of the list.
    def push(self, new_data):

        # 1 & 2: Allocate the Node &
```

```python
            # Put in the data
            new_node = self.Node(new_data)

            # 3. Make next of new Node as head
            new_node.next = self.head

            # 4. Move the head to point to new Node
            self.head = new_node

    # Main function that inserts nodes of linked list q into p at
    # alternate positions. Since head of first list never changes
    # and head of second list/ may change, we need single pointer
    # for first list and double pointer for second list.
    def merge(self, q):
        p_curr = self.head
        q_curr = q.head

        # While there are available positions in p;
        while p_curr != None and q_curr != None:

            # Save next pointers
            p_next = p_curr.next
            q_next = q_curr.next

            # make q_curr as next of p_curr
            q_curr.next = p_next # change next pointer of q_curr
            p_curr.next = q_curr # change next pointer of p_curr

            # update current pointers for next iteration
            p_curr = p_next
            q_curr = q_next
        q.head = q_curr

    # Function to print linked list
    def printList(self):
        temp = self.head
        while temp != None:
            print str(temp.data),
            temp = temp.next
        print ''

# Driver program to test above functions
llist1 = LinkedList()
llist2 = LinkedList()
llist1.push(3)
llist1.push(2)
llist1.push(1)

print "First Linked List:"
llist1.printList()

llist2.push(8)
llist2.push(7)
llist2.push(6)
llist2.push(5)
llist2.push(4)

print "Second Linked List:"

llist2.printList()
llist1.merge(llist2)

print "Modified first linked list:"
llist1.printList()
```

```
print "Modified second linked list:"
llist2.printList()

# This code is contributed by BHAVYA JAIN
```

Output:

```
First Linked List:
1 2 3
Second Linked List:
4 5 6 7 8
Modified First Linked List:
1 4 2 5 3 6
Modified Second Linked List:
7 8
```

This article is contributed by **Chandra Prakash**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

67 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 **1** vote(s)

☐ Add to TODO List

☐ Mark as DONE

Like   Share   36 people like this.

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