Delete alternate nodes of a Linked List

Given a Singly Linked List, starting from the second node delete all alternate nodes of it. For example, if the given linked list is 1->2->3->4->5 then your function should convert it to 1->3->5, and if the given linked list is 1->2->3->4 then convert it to 1->3.

Method 1 (Iterative)

Keep track of previous of the node to be deleted. First change the next link of previous node and then free the memory allocated for the node.

C/C++

```
// C program to remove alternate nodes of a linked list
#include<stdio.h>
#include<stdlib.h>

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

/* deletes alternate nodes of a list starting with head */
void deleteAlt(struct node *head)
{
    if (head == NULL)
        return;

    /* Initialize prev and node to be deleted */
    struct node *prev = head;
    struct node *node = head->next;

    while (prev != NULL && node != NULL)
    {
        /* Change next link of previous node */
        prev->next = node->next;

        /* Free memory */
        free(node);

        /* Update prev and node */
        prev = prev->next;
        if (prev != NULL)
```

```
            node = prev->next;
        }
    }

    /* UTILITY FUNCTIONS TO TEST fun1() and fun2() */
    /* Given a reference (pointer to pointer) to the head
      of a list and an int, push a new node on the front
      of the 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;

        /* Using push() to construct below list
          1->2->3->4->5  */
        push(&head, 5);
        push(&head, 4);
        push(&head, 3);
        push(&head, 2);
        push(&head, 1);

        printf("\nList before calling deleteAlt() \n");
        printList(head);

        deleteAlt(head);

        printf("\nList after calling deleteAlt() \n");
        printList(head);

        return 0;
    }
```

Run on IDE

Java

```
// Java program to delete alternate nodes of a linked list
class LinkedList
{
    Node head;  // head of list

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

    void deleteAlt()
    {
       if (head == null)
          return;

       Node prev = head;
       Node now = head.next;

       while (prev != null && now != null)
       {
           /* Change next link of previus node */
           prev.next = now.next;

           /* Free node */
           now = null;

           /*Update prev and now */
           prev = prev.next;
           if (prev != null)
              now = prev.next;
       }
    }


    /* Utility functions */

    /* Inserts a new Node at front of the list. */
    public 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;
    }

    /* Function to print linked list */
    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 llist = new LinkedList();

        /* Constructed Linked List is 1->2->3->4->5->null */
        llist.push(5);
        llist.push(4);
        llist.push(3);
        llist.push(2);
        llist.push(1);

        System.out.println("Linked List before calling deleteAlt() ");
        llist.printList();

        llist.deleteAlt();

        System.out.println("Linked List after calling deleteAlt() ");
        llist.printList();
    }
}
/* This code is contributed by Rajat Mishra */
```

Run on IDE

Output:

```
List before calling deleteAlt()
1 2 3 4 5
List after calling deleteAlt()
1 3 5
```

Time Complexity: O(n) where n is the number of nodes in the given Linked List.

Method 2 (Recursive)

Recursive code uses the same approach as method 1. The recursive coed is simple and short, but causes O(n) recursive function calls for a linked list of size n.

```
/* deletes alternate nodes of a list starting with head */
void deleteAlt(struct node *head)
{
    if (head == NULL)
        return;

    struct node *node = head->next;

    if (node == NULL)
        return;

    /* Change the next link of head */
    head->next = node->next;

    /* free memory allocated for node */
    free(node);

    /* Recursively call for the new next of head */
    deleteAlt(head->next);
}
```

Run on IDE

Time Complexity: O(n)

Please write comments if you find the above code/algorithm incorrect, or find better ways to solve the same problem.

70 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)

2.3 Average Difficulty : **2.3/5.0**
Based on **3** vote(s)

Add to TODO List

Mark as DONE

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