GeeksforGeeks
A computer science portal for geeks

Practice   IDE   Q&A   GeeksQuiz

Find a triplet from three linked lists with sum equal to a given number

Given three linked lists, say a, b and c, find one node from each list such that the sum of the values of the nodes is equal to a given number.

For example, if the three linked lists are 12->6->29, 23->5->8 and 90->20->59, and the given number is 101, the output should be tripel "6 5 90".

In the following solutions, size of all three linked lists is assumed same for simplicity of analysis. The following solutions work for linked lists of different sizes also.

A simple method to solve this problem is to run three nested loops. The outermost loop picks an element from list a, the middle loop picks an element from b and the innermost loop picks from c. The innermost loop also checks whether the sum of values of current nodes of a, b and c is equal to given number. The time complexity of this method will be O(n^3).

Sorting can be used to reduce the time complexity to O(n*n). Following are the detailed steps.
1) Sort list b in ascending order, and list c in descending order.
2) After the b and c are sorted, one by one pick an element from list a and find the pair by traversing both b and c. See isSumSorted() in the following code. The idea is similar to Quadratic algorithm of 3 sum problem.

Following code implements step 2 only. The solution can be easily modified for unsorted lists by adding the merge sort code discussed here.

C/C++

```
// C/C++ program to find a triplet from three linked lists with
// sum equal to a given number
#include<stdio.h>
#include<stdlib.h>

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

```c
/* A utility function to insert a node at the beginning of a
   linked 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;
}

/* A function to chech if there are three elements in a, b
   and c whose sum is equal to givenNumber.  The function
   assumes that the list b is sorted in ascending order
   and c is sorted in descending order. */
bool isSumSorted(struct node *headA, struct node *headB,
                 struct node *headC, int givenNumber)
{
    struct node *a = headA;

    // Traverse through all nodes of a
    while (a != NULL)
    {
        struct node *b = headB;
        struct node *c = headC;

        // For every node of list a, prick two nodes
        // from lists b abd c
        while (b != NULL && c != NULL)
        {
            // If this a triplet with given sum, print
            // it and return true
            int sum = a->data + b->data + c->data;
            if (sum == givenNumber)
            {
                printf ("Triplet Found: %d %d %d ", a->data,
                                        b->data, c->data);
                return true;
            }

            // If sum of this triplet is smaller, look for
            // greater values in b
            else if (sum < givenNumber)
                b = b->next;
            else // If sum is greater, look for smaller values in c
                c = c->next;
        }
        a = a->next;  // Move ahead in list a
    }

    printf ("No such triplet");
    return false;
}


/* Drier program to test above function*/
int main()
{
```

```
        /* Start with the empty list */
        struct node* headA = NULL;
        struct node* headB = NULL;
        struct node* headC = NULL;

        /*create a linked list 'a' 10->15->5->20 */
        push (&headA, 20);
        push (&headA, 4);
        push (&headA, 15);
        push (&headA, 10);

        /*create a sorted linked list 'b' 2->4->9->10 */
        push (&headB, 10);
        push (&headB, 9);
        push (&headB, 4);
        push (&headB, 2);

        /*create another sorted linked list 'c' 8->4->2->1 */
        push (&headC, 1);
        push (&headC, 2);
        push (&headC, 4);
        push (&headC, 8);

        int givenNumber = 25;

        isSumSorted (headA, headB, headC, givenNumber);

        return 0;
}
```

Run on IDE

Java

```
// Java program to find a triplet from three linked lists with
// sum equal to a given number
class LinkedList
{
    Node head;  // head of list

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

    /* A function to chech if there are three elements in a, b
       and c whose sum is equal to givenNumber.  The function
       assumes that the list b is sorted in ascending order and
       c is sorted in descending order. */
    boolean isSumSorted(LinkedList la, LinkedList lb, LinkedList lc,
                        int givenNumber)
    {
        Node a = la.head;

        // Traverse all nodes of la
        while (a != null)
        {
            Node b = lb.head;
            Node c = lc.head;
```

```java
            // for every node in la pick 2 nodes from lb and lc
            while (b != null && c!=null)
            {
                int sum = a.data + b.data + c.data;
                if (sum == givenNumber)
                {
                    System.out.println("Triplet found " + a.data +
                                    " " + b.data + " " + c.data);
                    return true;
                }

                // If sum is smaller then look for greater value of b
                else if (sum < givenNumber)
                    b = b.next;

                else
                    c = c.next;
            }
            a = a.next;
        }
        System.out.println("No Triplet found");
        return false;
    }


    /*  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(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;
    }

     /* Drier program to test above functions */
    public static void main(String args[])
    {
        LinkedList llist1 = new LinkedList();
        LinkedList llist2 = new LinkedList();
        LinkedList llist3 = new LinkedList();

        /* Create Linked List llist1 100->15->5->20 */
        llist1.push(20);
        llist1.push(5);
        llist1.push(15);
        llist1.push(100);

        /*create a sorted linked list 'b' 2->4->9->10 */
        llist2.push(10);
        llist2.push(9);
        llist2.push(4);
        llist2.push(2);

        /*create another sorted linked list 'c' 8->4->2->1 */
        llist3.push(1);
        llist3.push(2);
        llist3.push(4);
```

```
        llist3.push(8);

        int givenNumber = 25;
        llist1.isSumSorted(llist1,llist2,llist3,givenNumber);
    }
} /* This code is contributed by Rajat Mishra */
```

Run on IDE

Output:

```
Triplet Found: 15 2 8
```

Time complexity: The linked lists b and c can be sorted in O(nLogn) time using Merge Sort (See this). The step 2 takes O(n*n) time. So the overall time complexity is O(nlogn) + O(nlogn) + O(n*n) = O(n*n).

In this approach, the linked lists b and c are sorted first, so their original order will be lost. If we want to retain the original order of b and c, we can create copy of b and c.

This article is compiled by **Abhinav Priyadarshi** and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

42 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 **2** vote(s)   Add to TODO List

                                                      Mark as DONE

Like   Share   5 people 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!