# Homework 8

**有两个单向链表（链表长度分别为m，n），这两个单向链表有可能在某个元素合并，如下图所示的这样，也可能不合并。现在给定两个链表的头指针，在不修改链表的情况下，如何快速地判断这两个链表是否合并？如果合并，找到合并的元素，也就是图中的 x 元素。**

```
package com.ll;

class LinkedList {

    static Node head1, head2;

    static class Node {
        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }

    public int getNode() {
        Node n = _getIntersectionNode(head1, head2);
        if (n == null) {
            return -1;
        }
        return n.data;
    }

    private Node _getIntersectionNode(Node node1, Node node2) {
        if (node1 == null || node2 == null) return null;

        Node a = node1;
        Node b = node2;

        while (a != b) {
            a = a == null ? node2 : a.next;
            b = b == null ? node1 : b.next;
        }

        return a;
    }
```

```
}

public class Main {
    public static void main(String[] args) {
        // there are interaction between LinkedList
        LinkedList hasInteractionList = new LinkedList();

        // creating the first linked list
        LinkedList.head1 = new LinkedList.Node(3);
        LinkedList.head1.next = new LinkedList.Node(6);
        LinkedList.head1.next.next = new LinkedList.Node(9);
        LinkedList.head1.next.next.next = new LinkedList.Node(15);
        LinkedList.head1.next.next.next.next = new LinkedList.Node(30);

        // creating the second linked list
        LinkedList.head2 = new LinkedList.Node(10);
        LinkedList.head2.next = LinkedList.head1.next.next.next;
        LinkedList.head2.next.next = LinkedList.head1.next.next.next.next;

        System.out.println("The node of intersection is " +
hasInteractionList.getNode());


        // No interaction between linked list
        LinkedList noInteractionList = new LinkedList();

        LinkedList.head1 = new LinkedList.Node(3);
        LinkedList.head1.next = new LinkedList.Node(6);
        LinkedList.head1.next.next = new LinkedList.Node(9);

        LinkedList.head2 = new LinkedList.Node(10);
        LinkedList.head2.next = new LinkedList.Node(14);

        System.out.println("The node of interaction is " +
noInteractionList.getNode());
    }
}
```

输出:

```
The node of intersection is 15
The node of interaction is -1
```

时间复杂度:

O(m+n)

空间复杂度:

O(1)

请画出DataNode服务器节点宕机的时候，HDFS的处理过程时序图。