一、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。请用（伪）代码描述算法，并给出时间复杂度和空间复杂度

```
package main

type DataNode struct {
    data interface{}
    nextData *DataNode
}

func (d *DataNode) next() *DataNode {
    if d.nextData != nil {
        return d.next()
    }
    return nil
}

func (d *DataNode) get() interface{} {
    return d.data
}

func (d *DataNode) isHaveSame(p *DataNode, vd Vector,vp Vector) interface{} {
    if p == nil {
        return nil
    }
    if d == nil {
        return nil
    }
    if d.data == p.data {
        return d.data
    }
    if vd.contains(d.data) {
        return d.data
    }
    if vp.contains(p.data) {
        return p.data
    }
    vd.add(p.data)
    vp.add(d.data)

    dd = d.nextData
    return dd.isHaveSame(p.next(), vd, vp)
}
```

时间复杂度为：O(n)，空间复杂度为O(m+n)

请画出DataNode服务器节点宕机的时候，HDFS的处理过程时序图