第八周作业

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，也可能不合并，如下图所示的这样。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。请用代码（或伪代码）描述算法，并给出时间复杂度。

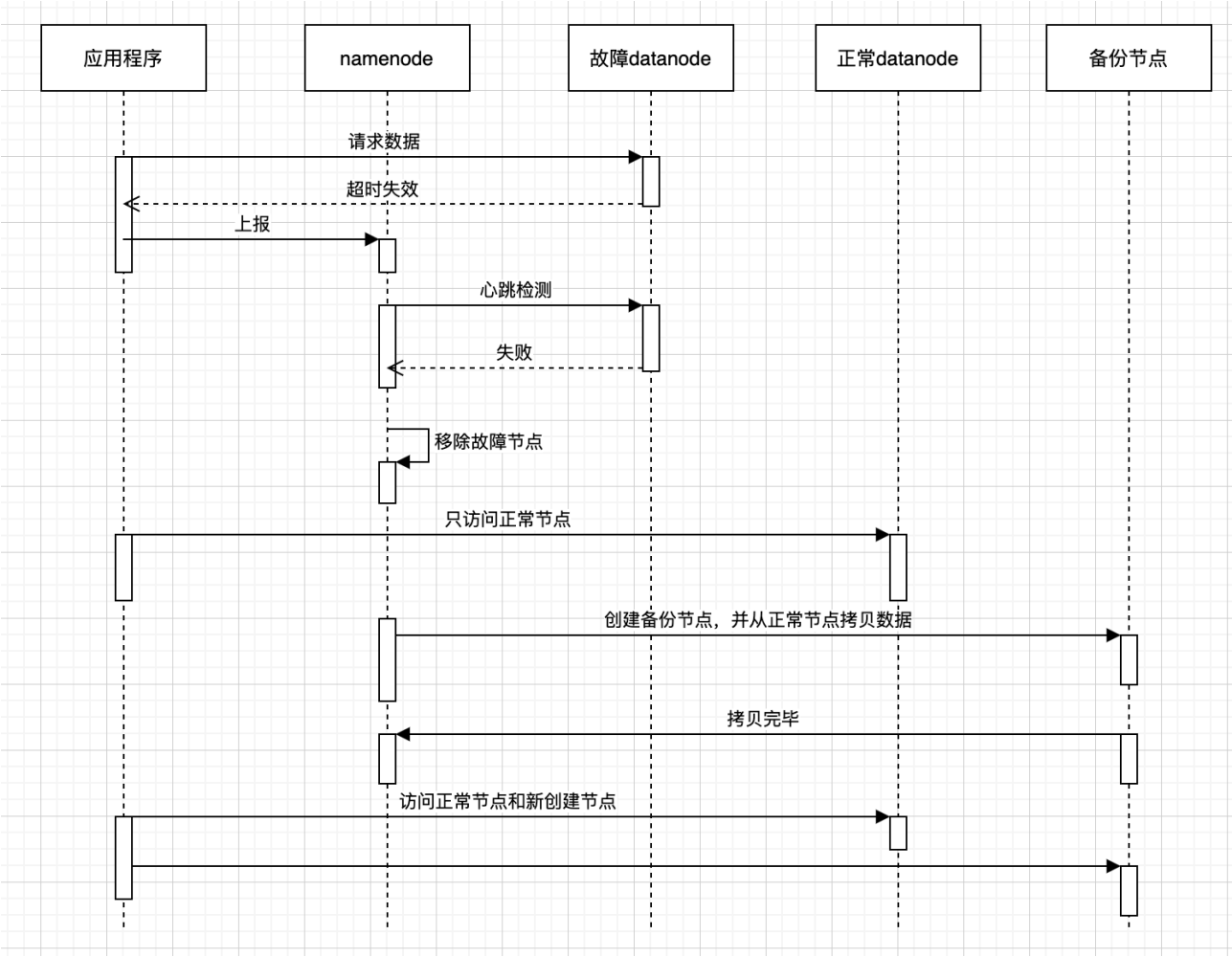
代码实现如下：

```
public class FindMutualNodeForLinkedLists {
    public void findMutualList(LinkedList<Object> a, LinkedList<Object> b) {
        Set<Object> set = new HashSet<>();
        Iterator<Object> a1 = a.iterator();
        Iterator<Object> b1 = b.iterator();
        while (a1.hasNext()) {
            set.add(a1.next());
        }
        boolean hasMutual = false;
        Object mutualElement = null;
        while (b1.hasNext()) {
            Object next = b1.next();
            if (set.contains(next)) {
                hasMutual = true;
                mutualElement = next;
                break;
            }
        }
        System.out.println("是否有共同节点:" + hasMutual);
        if (hasMutual) {
            System.out.println("共同节点是: " + mutualElement);
        }
    }
}
```

代码分析：如上代码片段所示，先把第一个列表遍历一次并放入一个set，然后再遍历第二个列表看有没有相同的元素，如果有，就表示成功找到，并取出该元素即可。

复杂度分析：算法需要遍历第一个list一次，第二个list平均0.5n次，因此时间复杂度是O(m+0.5n)，即O(m+n)，由于需要一个set来存储第一个list的节点，因此空间复杂度是O(m)

二、请画出 DataNode 服务器节点宕机的时候，HDFS 的处理过程时序图。



如上图所示，应用程序访问节点失败后者namenode心跳失败发现故障的datanode节点后，namenode会把这个故障节点摘除，这样应用程序只访问没问题的数据节点。这时候namenode会发现，这个故障节点所存储的那些文件分片备份数不够了。它会找一个备份机器，创建一个新的datanode节点，再把数据从正常节点拷贝到这个新的datanode节点，等数据同步完毕以后，这个新的节点也可以开始对外提供服务