区块链实验Ex5

一.实验设计

1.合约中的函数

（1）lookup(address debtor, address creditor) public view returns

(uint32 ret):

该函数返回债务人欠债权人的金额.

（2）add\_IOU(address creditor, uint32 amount, ...):

为调用者添加一个欠条,如果你已经欠钱，金额会增加。金额必须为正数。

2.客户端中的功能

（1）getUsers():

返回一个地址列表，这个地址列表包含:“曾经发送或收到欠条的每个人”或“目前欠或被欠钱的每个人”。

（2）getTotalOwed(user):

返回指定用户所欠的总金额。

（3）getLastActive(user):

返回该用户上次记录活动的UNIX时间戳(自1970年1月1日以来的秒数)(发送欠条或被列为欠条上的“债权人”)。如果找不到活动，则返回null。.

（4）add\_IOU(creditor, amount):

向合约提交一个欠条以及相应的债权人和所欠数量，不返回任何值，参见下面的循环解析说明。

(5) getNeighbors(node):

返回node该欠债人所欠债的债权人，根据提示为方便调用doBFS函数及其他调用

辅助函数：

（1）getAllFunctionCalls(addressOfContract, functionName)：

在块历史记录中搜索对“addressOfContract”合约上函数名为“functionName”的所有调用

返回一个对象数组，每个调用一个对象数组，包含发送方（“from”）和参数（“args”）

（2）doBFS(start, end, getNeighbors)：

找到从头到尾的路径（如果不存在，则返回 null）

二.实验实现

1.合约中的函数

用一个结构体表示债务代表所欠的一些金额。

struct Debt {

uint32 amount;

}

然后跟踪债务。从债务人映射到映射从债权人到债务。例如，debts[Alice][Bob] = 10 意味着爱丽丝欠鲍勃10块钱。

mapping(address=>mapping(address=>Debt)) internal all\_debts;

查找债务人欠债权人的债务总额.

function lookup(address debtor, address creditor) public view returns (uint32 ret) {

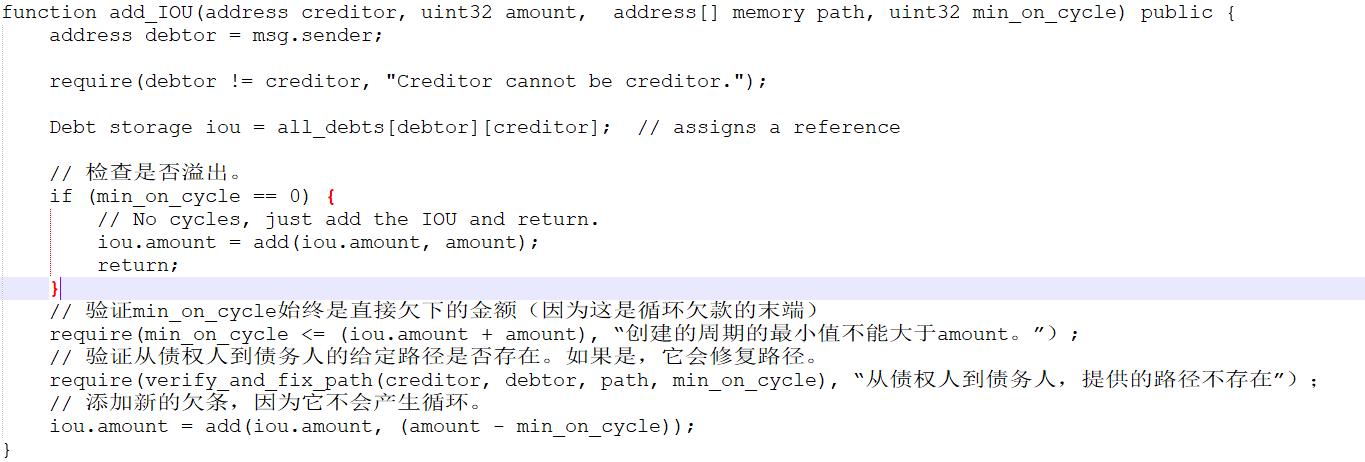
ret = all\_debts[debtor][creditor].amount;

}

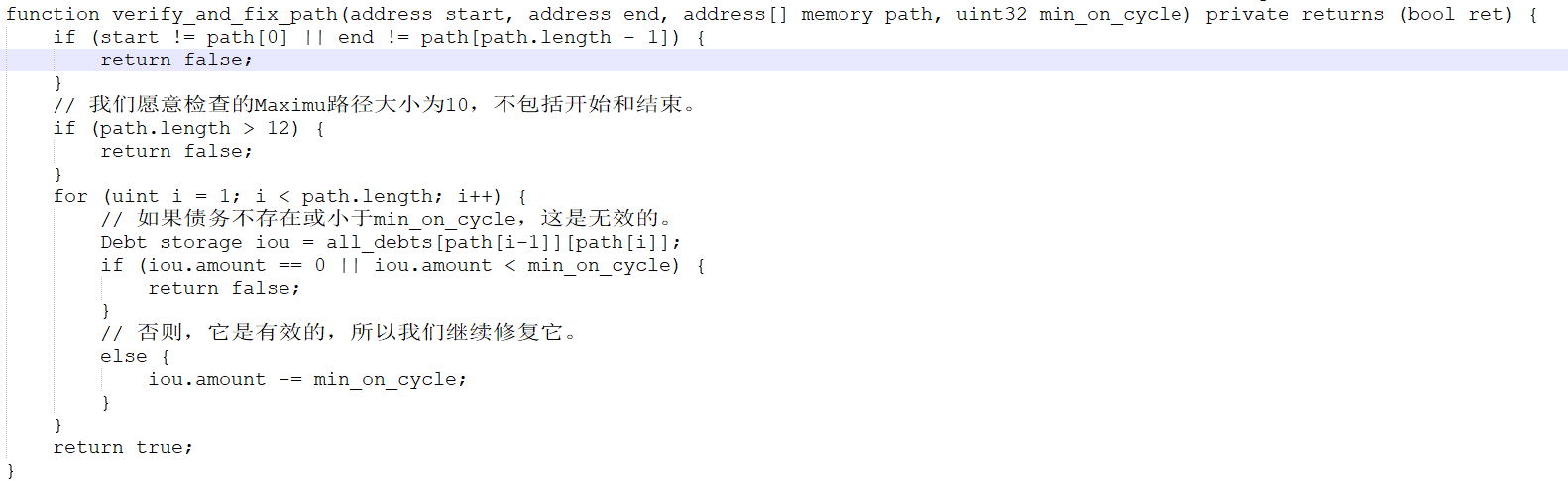
然后一个函数满足如下两个要求：

1. 若给出了路径，它确实存在，并将债权人和债务人联系起来。

2. min\_on\_cycle必须是循环上的最小值。



最后验证路径是否有效并修复它。路径可以是部分固定的，所以调用者负责撤消部分固定的路径。最简单的方法是使用require（），因为这将回滚事务。



2.客户端中的功能

function getNeighbors(node) {

    var neighbors=[]

    for(var user of getUsers()){

        if(BlockchainSplitwise.lookup(node,user).toNumber()>0){

            neighbors.push(user);

        }

    }

    return neighbors;

}

查询所有曾经发送或收到欠条的每个人”或“目前欠或被欠钱的每个人，如果存在node为欠债人的债务，储存该债权人

function getUsers() {

    var users=new Set();

    for(var add\_IOU\_call of getAllFunctionCalls(contractAddress, 'add\_IOU')){

        users.add(add\_IOU\_call.from);

        users.add(add\_IOU\_call.args[0]);

    }

    return [...users];

}

Set 是唯一值的集合，用其存储每个地址唯一的地址集合

在块历史记录中搜索对“contractAddress”合约上函数名为“add\_IOU”的所有调用，对这些调用，添加调用的返回进set，从而返回一个地址列表，这个地址列表包含:“曾经发送或收到欠条的每个人”或“目前欠或被欠钱的每个人”。

function getTotalOwed(user) {

    var owed = 0;

    for(var creditor of getNeighbors(user)){

        owed+=BlockchainSplitwise.lookup(user,creditor).toNumber();

    }

    return owed;

}

查询user所欠的每笔债务的amount再相加返回

function getLastActive(user) {

    for(var add\_IOU\_call of getAllFunctionCalls(contractAddress, 'add\_IOU')){

        if(user==add\_IOU\_call.from || user==add\_IOU\_call.args[0]){

            return add\_IOU\_call.timestamp;

        }

    }

    return 0;

}

add\_IOU调用记录从新到旧搜索，如果user为信息发送人或args[0]即“address creditor”，则说明找到user为发送欠条或被列为欠条上的债权人的记录，则返回记录活动的UNIX时间戳，否则返回NULL

function add\_IOU(creditor, amount) {

    var debtor=web3.eth.defaultAccount;

var path=doBFS(creditor,debtor,getNeighbors);

    var offset=amount;

if(path!=null){

//如果不为空，找到从creditor到debtor的路径，现在要新建从debtor到creditor的路径，出现环路

        for(var i=1;i<path.length;i++){

            var amount=BlockchainSplitwise.lookup(path[i-1],path[i]).toNumber();

            offset=Math.min(offset,amount);//找到环路上的最小值，offset为这个值，即需要路径减去这个值

        }

    }else{

        path=[]

        offset=0;//没有环路

    }

    console.log(creditor,amount,path,offset)

BlockchainSplitwise.add\_IOU(creditor, amount, path, offset);

//将债务信息记录上