Exercise 5 DAPP

(2022区块链基础及其应用)

在本次作业中，你将使用Solidity和web3.js在以太坊（Ethereum）上实现一个复杂的去中心化应用程序(DApp)。你需要编写一个智能合约和访问它的用户客户端，学习DApp的“全栈”开发。为了节省你的时间，请在开始开发之前阅读整个作业介绍特别是注意部分。

1 区块链版本的 Splitwise

我们本次的任务是创建一个去中心化的应用来跟踪借贷—Splitwise的区块链版本 。如果你之前没有听过这个应用的话, Splitwise是一个简单的应用，可以跟踪一群人中谁欠谁的钱 (也许是在分午餐、杂货或账单之后). 为了说明这个应用程序，考虑以下场景:

Alice、Bob和Carol都是喜欢一起出去吃饭的朋友。Bob上次和Alice出去吃饭时付了午饭钱，所以Alice欠了Bob10美元。同样，Carol和Bob出去吃饭时付了钱，所以Bob欠了Carol10美元.

现在, 假设Carol资金紧张，向Alice借了10美元.注意，此时他们不需要在未来的某个时刻偿还他们的借款，而是他们都同意每个人都互不相欠。换句话说，只要有一个债务循环，我们就可以把它从我们的簿记中移除，使一切变得更简单，并减少现金需要转手的次数。我们要建立一个去中心化的方法，去追踪谁欠谁什么，这样就不需要依赖任何一个可信任的第三方。它将是高效的：存储这些数据不会花费过多的gas；使用此应用程序不会在“区块链上”转移任何价值；唯一涉及的以太币（ether）将用于gas。

因为这个应用是在区块链上，当Carol为她和Bob的午饭付钱时，她可以要求Bob提交借条(IOU)(他可以使用我们的DApp来做)，并且Carol可以验证他确实有提交，公链上的存储将作为谁欠谁的钱的单一可信来源。稍后，当上述循环得到解决时，Carol将看到Bob不再欠她的钱。

作为整个应用的一部分，我们还将构建一个用户界面，为用户计算有用的信息，并允许非程序员也使用我们的DApp。

2 开始

1. 安装必备软件:你需要从https://nodejs.org/en/下载并安装Node.js，选择LTS版本。
2. 运行npm install -g Ganache-CLI来安装Ganache CLI，我们将用它在我们的本地机器上模拟一个真实的以太坊节点。然后，运行ganache-cli来运行节点，使用Ctrl-C来停止节点
3. 从课程网站上下载入门代码。
4. 在网页浏览器中打开https://remix.ethereum.org。在“运行”选项卡中，将环境

设置为“Web3提供程序”，在提示时单击“确定”，然后将“Web3提供程序端点”设置为http://localhost:8545——这是默认设置。这是你将开发智能合约的地方(你将使用Solidity语言进行编写)。

5. 在你的web浏览器中打开index.html文件(你应该会看到一个标题为“区块链Splitwise”的页面)。同时打开浏览器的JavaScript控制台也很有帮助，这样你就可以看到错误消息(在本文档的末尾有一个如何做到这一点的链接)。如果到目前为止一切正常，你应该不会在控制台看到错误信息(可能会看到一个警告)。.

6. 在你最喜欢的IDE或文本编辑器(Sublime text, Atom，或Visual Studio code)中打开starter code目录。你将修改script.js来构建客户端，但查看其他文件可能会有所帮助。有地方标记了需要修改的函数-请不要修改任何其他代码。可以随意添加辅助函数。

7. 仔细阅读入门代码、web3.js的API和Solidity文档。在编写代码之前，仔细考虑系统的整体设计。哪些数据应该存储在链上?哪些计算将由合同完成，哪些计算将在客户端完成?
8. 为下面列出的要求实施代码。 当你完成合同时，部署它，然后在 script.js 中更新合同哈希和 ABI。 ABI 可以从“编译”选项卡复制到剪贴板，合约哈希可以从“运行”选项卡的“已部署合约”面板复制。 请注意，合约哈希不是创建合约的交易的交易哈希。

Note on OSs: 以上所有步骤应该在基于Unix的系统和 Windows上都能正常工作。我们要求你执行的命令将在标准的Unix终端和Windows命令提示符中工作。

3 需求

该项目有两个主要组件:一个智能合约，用Solidity编写并在区块链上运行，以及一个在web浏览器上本地运行的客户端，它使用web3.js观察区块链，并且可以调用智能合约中的函数。

3.1 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”。你可能会发现这对你的其他功能很有用。

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. send_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。
.

3.2 合约中的函数

1. lookup(address debtor, address creditor) public view returns (uint32 ret):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。你可以使此函数接受任意数量的附加参数。 请参阅下面有关解决循环的注释.

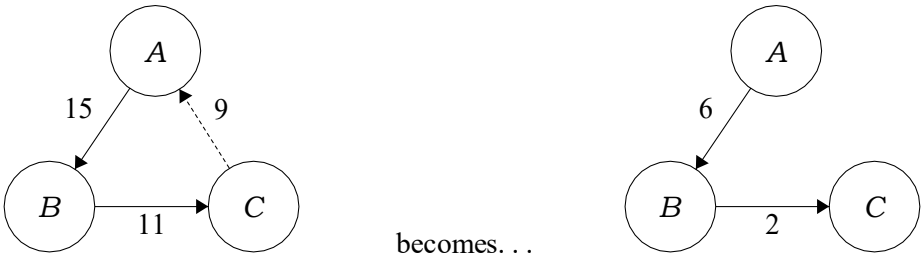
你也可以为客户端或合同编写更多的帮助程序。客户端可以用BlockchainSplitwise.functionname(arguments) 调用合约函数。记住，客户端函数使用JavaScript写，合约函数采用Solidity编写。

4 解决债务循环

把欠条想象成债务图是很有帮助的。也就是说，假设每个用户都是一个节点，从A到B每条权重为X的加权有向边代表“A欠B X美元”的事实.我们将它写成$A \xrightarrow{X} B$. 我们希望我们的应用程序通过从循环中的每个步骤中减去循环中所有权重的最小值来“解决”此图中的任何循环（从而使循环中的至少一个步骤具有权重“0”）.

例如:如果$A \xrightarrow{15} B$并且$B \xrightarrow{11} C$,当C加上$C \xrightarrow{16} A$,实际的余额将会更新为$A \xrightarrow{4} B, B \xrightarrow{0} C, C \xrightarrow{5} A$.相似的,如果$C \xrightarrow{9} A$,那么最后实际的余额将会更新为$A \xrightarrow{6} B, B \xrightarrow{2} C, C \xrightarrow{0} A$.

becomes. . .



要求是，当你在使用客户端（send_iou）添加一个IOU时，如果形成任何潜在的循环，你必须至少“解决”其中一个。 无需担心涉及多个循环的复杂情况，或者在这些情况下优化采用的路径（例如最大流量）。 您可以假设作为两个合约函数（add_iou和lookup）的先决条件--图中没有循环。 最后，您还可以假设找到的任何循环都比较小（例如,小于10）。

我们在代码中为您提供广度优先搜索算法，要使用它，请传入开始和结束节点，以及获取任何给定节点的“邻居”节点的函数。 您也可以不使用此实现。

合约能否安全地执行取决于你。 一旦发布，恶意客户就不可能以某种方式“抹去”他们的债务。

我们现在可以说明如何在这个系统中偿还欠条.假如Alice找Bob借了10美元,现在她想用现金还给Bob,当Alice用现金将10美元还给Bob后,Bob将会在系统中添加一个10美元的欠条(IOU),债权人为Alice,这将会产生一个循环:$A \xrightarrow{10} B, B \xrightarrow{10} A$,通过上面的循环解决方案,最后将会得到$A \xrightarrow{0} B, B \xrightarrow{0} A$.

5 总体需求

你可以用任何你喜欢的方式编写你的合同，只要它具有指定的lookup和add_IOU函数。你的目标是编写一份合同，使这两个合同函数使用的存储和计算量最小化。这将使gas成本最小化。你可以假设交易量足够小，在客户端搜索整个区块链是可行的，但你不应该假设唯一的用户是你钱包里的那些人——换句话说，就是web3.eth。因为账户并不包含系统中所有可能的用户。

6 提交你的代码

提交的内容将根据它是否正确回答查询，以及是否产生合理的汽油费来评分。在提交之前，请确保将您的合约的Solidity代码从Remix复制并粘贴到mycontract.sol中。

7 注意

7.1 系统架构

- 您应该首先决定在区块链上存储什么数据结构。仔细考虑你需要提供给客户的信息。你不需要使用任何特别花哨的数据结构。你的决定可能会使实现变得更加困难，所以你应该可以回去更改你的架构。
- 我们还没有提到在仅仅两个人之间形成循环的情况下该怎么做。我们建议设计你的系统，使这不是一个特殊的情况——当债务人已经“偿还”债权人，债权人只是试图在相反的方向上增加一个借条，触发循环解决，以双方都欠对方0。我们还建议你避免任何“负”债务的概念，因为这可能使事情变得过于复杂。
- 记住，当优化gas成本时，在客户端运行的功能是免费的——它们不会产生任何成本。
- 我们建议在设计完你的系统后，你可以从在Remix中编写和彻底调试合约开始。你可以调用右下角面板中的函数，并使用右上角的“Account”选择器切换账户。要将地址复制到剪贴板，你可以点击选择器旁边的复制图标。一旦你确定合同符合预期，你就应该开始编写客户端。
- 你不需要写大量的代码来完成任务。我们的解决方案是大约40行Solidity代码和大约70行JavaScript代码(不包括ABI)。

7.2 实用的开发和调试

- 由于JavaScript以一种奇怪的方式处理整数，因此你对BlockchainSplitwise.lookup的调用将返回一个JavaScript对象；具体来说，是具有c,e和s属性的BigNumber。 要将其转换为普通整数，请调用bn.toNumber()，其中bn是返回的BigNumber。你可能希望每次在客户端调用BlockchainSplitwise.lookup时执行此操作。
- 要调试客户端代码，可以自由地使用console.log。你应该能在浏览器的JavaScript控制台中看到调用的结果以及它们的起始行号。
- 关于同步XMLHttpRequest的警告可以忽略。关于没有连接到localhost:8545的错误通常是因为你没有运行ganache-cli.
- Solidity有一个非常有用的函数require，可以让你检查前置条件.
- 如果你只想调试你的合约，你可以使用“JavaScript VM”Web3 Provider，然后在失败的交易上按“debug”。确保之后切换回来，以便你可以运行客户端代码。

8 参考

- 你可以在这里了解如何打开浏览器的JavaScript控制台.
- web3.js的API使用.