超过100道以太坊区块链开发技术岗位的面试题，附参考答案。面试题目涵盖以太坊的基本概念、Geth客户端使用、智能合约基本概念、Solidity开发语言、去中心化 应用DApp、web3.js开发库等方面。

以太坊

问：以太坊的有价通证叫什么?
答：以太（ETH：Ether）
问：Wei和以太有什么区别?
答：Wei是一个面额，像美分到美元或便士到磅。 1 ETH =10^18 Wei
问：以太坊的平均出块时间是多少?
答：大约14秒
问：以太坊的平均块大小是多少?
答：大约2KB，实际值取决于具体情况。
问：以太坊是否支持脚本？ 如果是这样，支持什么类型的脚本?
答：是的。 它支持智能合约
问：你如何得到以太?
答：有几种方法： 1.成为一名矿工
2.用其他货币换取
3.使用以太Faucet，例如 https://faucet.metamask.io
4.接受别人的赠送
问：以太从哪里来的?
答：在2014年预售中首次创建了6000万个。另外，在挖出新块时也会生成以太。
问：什么是节点?
答：一个节点本质上是一台连接到网络的计算机，它负责处理交易。
问：你熟悉多少种以太坊网络?
答：有三种类型的网络 – 实时网络（主），测试网络（如Ropsten和Rinkeby）和私有网络。
问：与以太坊网络交互的方式有哪些?

答：可以使用电子钱包或DApp
问：你可以“隐藏”一个以太坊交易吗?
答：不可以。所有交易对每个人都是可见的。
问：交易记录在哪里?
答：在公共账本上。
问：这些网络的ID是什么?
答：Live（id = 1），Ropsten（id = 3），Rinkeby（id = 4），Private（由开发人员分配）
问：我可以在Rinkeby测试网络中挖一些以太，然后转移到Live网络吗?
答：不可以。不能在不同的以太坊网络之间传递以太。
问：为什么需要私有网络?
答：有很多原因，但主要是为了数据隐私、分布式数据库、权限控制和测试。
问：你如何轻松查看有关交易和区块的详细信息?
答：使用区块链浏览器，如etherscan.io或live.ether.camp
问：私有网络的交易和区块信息怎么查看呢?
答：可以使用开源浏览器客户端，例如https://github.com/etherparty/explorer
问：区块链的共识是什么?
答：遵循特定协议（如以太坊）验证交易（创建块）的过程。
问：区块链中两种常用的共识模型是什么?
答：工作量证明（POW）和权益证明（POS）。
问：简单地解释下工作量证明。
答：它实际上是矿工为了证明自己的工作量并验证交易而对一个计算密集型问题的求解。
问：以简单的方式解释权益证明。
答：区块的创建者是根据节点所持有的财富和股权随机选择的。它不是计算密集型的。
问：以太坊使用什么共识模式?
答：截至2018年初，它使用工作量证明，但今后将切换到权益证明。

问：怎么挖以太币?
答：使用钱包或geth客户端。
问：用什么来对交易进行签名?
答：用户的私钥。
问：丢失私钥后还能恢复以太坊账户吗?
答：可以，可以使用助记词组。

以太坊节点软件（Geth）

问：有哪些方法可以连接到一个以太坊节点?
答：IPC-RPC、JSON-RPC和WS-RPC。
问：那么Geth是什么?
答：Geth是以太坊的客户端。
问：连接到geth客户端的默认方式是什么?
答：默认情况下启用IPC-RPC，其他RPC都被禁用。
问：你知道geth的哪些API?
答：Admin、eth、web3、miner、net、personal、shh、debug和txpool。
问：你可以使用哪些RPC通过网络连接到geth客户端?
答：可以使用JSON-RPC和WS-RPC通过网络连接到geth客户端。 IPC-RPC只能连接到同一台机器上的geth客户端。
问：如果启动geth时使用了-rpc选项，哪些RPC会被启用?
答：JSON-RPC。
问：哪些RPC API是默认启用的?
答：eth、web3和net。
问：如何为JSON RPC启用Admin API?
答：使用-rpcapi选项。
问：选项-datadir有什么作用?
答：它指定了区块链的存储位置。
问：什么是geth的“快速”同步，为什么它更快?
答：快速同步会将事务处理回执与区块一起下载并完整提取最新的状态数据库，而不是重新执行所有发生过的交易。
问：选项--testnet是做什么的?
答：它将客户端连接到Ropsten网络。

问：启动geth客户端会在屏幕上输出大量文字，应该如何减少输出信息?
答：可以将--verbosity设置为较低的数字（默认值为3）
问：如何使用IPC-RPC将一个geth客户端连接到另一个客户端?
答：首先启动一个geth客户端，复制它的管道位置，然后使用同一个datadir启动另一个geth客户端并使用--attach 选项传入管道位置。
问：如何将自定义javascript文件加载到geth控制台中?
答：通使用--preload选项传入js文件的路径。
问：geth客户端的帐户存储在哪里?
答：在keystore目录中。
问：为了进行交易，需要对账户进行什么操作?
答：必须先解锁该账户 - 可以传入账户地址或账户序号来解锁。 也可以使用--password选项传入一个密码文件， 其中包含每个账户的密码。
问：你提到了一些有关账户序号的内容。 什么因素决定账户的序号?
答：添加帐户的先后顺序。
问：是否可以使用geth进行挖矿?
答：可以，使用--mine选项开启。
问：什么是“etherbase”?
答：这是接收挖矿奖励的帐户，它是序号为0的帐户。

智能合约和Solidity

问：什么是智能合约?
答：这是用多种语言编写的计算机代码。 智能合约存在于以太坊网络上，它们根据预定规则执行动作，规则是由 参与者在这些合约中商定的。
问：智能合约可以使用哪些语言编写?
答：Solidity，这是最常用的语言，也可以使用Serpent和LLL。
问：你能举出一个智能合约的用例吗?
答：卖方-买方应用场景：买方在智能合约中存入款项，卖方看

到存款并发送货物，买方收到货物并 放行付款。
问：什么是Metamask?
答：Metamask是一个可以帮助用户在浏览器中与以太坊网络进行交互的工具
问：Metamask使用哪个以太坊节点?
答：它使用infura.io
问：Metamask不支持什么?
答：挖矿和合约部署。
问：执行合约是否免费?
答：不，调用合约方法是一个交易，因此需要支付费用。
问：访问智能合约的状态是否免费?
答：是的，查询状态不是交易。
问：谁执行合同?
答：矿工。
问：为什么调用智能合约的方法需要付费?
答：有些方法不会修改合约的状态，也没有其他逻辑，只是返回一个值，这样的方法是可以免费调用的。 调用那些改变合约状态的方法则需要付费，因为它们需要gas来执行。
问：为什么需要gas?
答：由于矿工在他们的机器上执行合约代码，他们需要gas来覆盖执行合约代码的成本。
问：是不是gas的价格决定了交易什么时候被处理?
答：即是，也不是。 gas价格越高，交易成功的可能性就越大。尽管如此，gas价格并不能保证更快的交易处理。
问：交易中的gas使用量取决于什么?
答：这取决于合约所用的存储量、指令（操作码）的类型和数量。 每个EVM操作码都对应一个固定的gas用量。
问：交易费是如何计算的?
答：gas用量*gas价格（由调用方指定gas价格）
问：如果智能合约的执行成本低于调用方指定的gas用量，用户是否得到退款?
答：是的

问：如果智能合约的执行成本高于指定的gas用量，会发生什么情况?
答：用户不会得到退款，并且一旦所有的gas用完，执行就会停止，合约也不会改变。
问：谁支付智能合约的调用费用?
答：调用合约的用户。
问：节点在什么上面运行智能合约代码?
答：EVM – 以太坊虚拟机。 EVM遵循EVM规范，该规范是以太坊协议的组成部分。 EVM只是节点上的一个进程。
问：为了运行智能合同，EVM需要什么?
答：它需要合约的字节码，是通过编译Solidity等更高级别的语言编写的合约来生成字节码。
问：粗略的说，EVM有哪些组成部分?
答：内存区域、堆栈和执行引擎。
问：什么是Remix?
答：开发，测试和部署合约的在线工具。 适合快速构建和测试轻量级合约，但不适合更复杂的合约。
问：在Remix中，可以连接哪些节点?
答：可以使用Metamask连接到公共节点、也可以链接到使用Geth搭建的本地节点，或者在Javascript VM中模拟的内存节点。
问：什么是DApp，它与App有什么不同？有什么不同?
答：App通常包含一个客户端，这个客户端会与一些中心化的资源（由一个组织拥有）进行通信， 通常客户端通过一个中间层连接到中心化的数据层，如果中心化的数据层中的信息丢失，不能很轻松地恢复。 DApp表示去中心化应用程序。 DApps通过智能合约与区块链网络进行交互。 DApp使用的数据驻留在合约实例中。 中心化数据可能比去中心化数据更容易受到破坏。

DApps和web3

问：DApp的前端是否局限于某些技术或框架?
答：不受限制。可以使用任何技术来开发DApp的前端，比如HTML，CSS，JS，Java，Python...

问：前端用什么库连接后端（智能合同）？
答：Web3.js库。
问：在DApp的前端需要哪些东西才能与指定的智能合约进行交互?
答：合约的ABI和字节码。
问：ABI有什么作用?
答：ABI是合约的公开接口描述对象，被DApps用于调用合约的接口。
问：字节码有什么作用?
答：节点上的EVM只能执行合约的字节码。
问：为什么要使用BigNumber库?
答：因为Javascript不能正确处理大数。
问：为什么需要检查在Web DApp代码的开始部分是否设置了web3提供器（Provider）？
答：因为Metamask会注入一个web3对象，它覆盖其他的web3设置。
问：为什么要使用web3.js版本1.x而不是0.2x.x?
答：主要是因为1.x的异步调用使用Promise而不是回调，Promise目前在javascript世界中 是处理异步调用的首选方案。
问：如何在web3 1.x中列出账户?
答：web3.eth.getAccounts
问：.call和.send有什么区别?
答：.send发送交易并支付费用，而.call查询合约状态。
问：这样发送1个以太对吗：.send({value:1})?
A：不对，这样发送的是1 wei。 交易中总是以wei为单位。
问：那么为了发送1个以太，我必须将这个值乘以10^18?
答：可以使用web3.utils.toWei(1，'ether')。
问：调用.send()时需要指定什么?
答：必须指定from字段，即发送账户地址。 其他一切都是可选的。
问：web3.eth.sendTransaction()的唯一功能是将以太发送到特定的地址，这个说法是否正确?

答：不对，也可以用它调用合约方法。
问：你是否知道以太坊的可扩展性解决方案?
答：2层协议。可能的解决方案是状态通道（state channels）和Plasma。

Solidity

问：Solidity是静态类型的还是动态类型的语言?
答：它是静态类型语言，这意味着类型在编译时是已知的。
问：Solidity中与Java“Class”类似的是什么?
答：合约。
问：什么是合约实例?
答：合约实例是区块链上已部署的合约。
问：请说出Java和Solidity之间的一些区别。
答：Solidity支持多重继承，但不支持重载。
问：你必须在Solidity文件中指定的第一件事是什么?
答：Solidity编译器的版本，比如指定为^ 0.4.8。 这是必要的，因为这样可以防止在使用其他版本的编译器时引入不兼容性错误。
问：合约中包含什么?
答：主要由存储变量、函数和事件组成。
问：合约中有哪些类型的函数?
答：有构造函数、fallback函数、修改合约状态的函数和只读的constant函数。
问：如果我将多个合约定义放入单个Solidity文件中，我会得到什么错误?
答：将多个合约定义放入单个Solidity文件是完全正确的。
问：两个合约之间交互的方式有哪些?
答：一个合约可以调用另一个合约，也可以继承其他合约。
问：当你尝试使用部署一个包含多个合约的文件时会发生什么?
答：编译器只会部署该文件中的最后一个合约，而忽略所有其他合约。
问：如果我有一个大项目，我需要将所有相关的合约保存到一个文件中吗?

答：不需要。可以使用import语句导入其他合约文件，例如 import "./MyOtherContracts.sol";。
问：我只能导入本地合约文件吗?
答：还可以使用HTTP协议导入其他合约文件，例如从Github导入： import "http://github.com/owner/repo/path_to_file";。
问：EVM的内存分成了哪些部分?
答：它分为Storage、Memory和Calldata。
问：请解释一下Storage。
答：可以把它想象成一个数据库。 每个合约管理自己的Storage变量。 它是一个键-值数据库（256位键值）。 就每次执行使用的gas而言，在Storage上读取和写入的成本更高。
问：请解释一下Memory。
答：这是一个临时存储区。 一旦执行结束，数据就会丢失。 可以在Memory上分配像数组和结构这样复杂的数据类型。
问：请解释一下Calldata 。
答：可以把calldata视为一个调用堆栈。 它是临时的、不可修改的，用来存储EVM的执行数据。
问：哪些变量存储在Storage，那些变量存储在Memory?
答：状态变量和局部变量（它们是对状态变量的引用）存储在Storage区域， 函数参数位于Memory区域。
问：看看下面的代码，并解释代码的哪一部分对应于哪个内存区域：

```
contract MyContract {
  // part 1
  uint count;
  uint[] totalPoints;

  function localVars(){
     // part 2
     uint[] localArr;
     // part 3
     uint[] memory memoryArr;
     // part 4
     uint[] pointer = totalPoints;
  }
}
```

答： 第1部分 – Storage
第2部分 – Storage
第3部分 – Memory
第4部分 – Storage
问：这样做对吗：

```
function doSomething(uint[] storage args) internal
returns(uint[] storage data) {…}
```

答：可以，可以强制将函数的参数设置为Storage存储。 在这种情况下，如果没有传递存储引用，编译器 会报错