Computer Science

1. Intro to Python and OOP
2. Algorithms
3. Data Structures
4. CS Unit 1 Build
5. Hash Tables and Blockchain
   - Arrays
   - Hash Tables
   - Blockchain I - Structure
   - Blockchain II - Distribution
6. Graphs
7. Computer Architecture
8. CS Unit 2 Build

Blockchain II - Distribution

Distribution is a key principle of blockchain. No one person or entity owns the chain or is responsible for maintaining it. Instead, a network of servers running the protocol for a given type of blockchain share this responsibility.

At the end of this module, you should be able to:

- implement a simple system for transactions in a blockchain
- derive a user's account balance by examining data in a simplified blockchain ledger

Pro Tip

Celebrate wrong answers, because that means someone's about to learn!

Prepare

Review each preclass resource before class.

Learn

Learn to implement a simple system for transactions in a blockchain

Overview

Intro

Let's start by reviewing a bit about blockchain. Blockchains usually:

1. Record changes in the data to a data store (database).
2. Replicate the data store across several systems in real time.
3. Use “peer-to-peer” network architecture rather than client-server.
4. Use cryptographic methods to prove ownership and authenticity.
5. Use hashes for references

Blockchain is not a singular technology; it is a collection of technologies. Blockchains are not new inventions; they are a collection of existing technologies put together in such a way as to create new capabilities.

Bitcoin

We commonly think of Bitcoin as a digital currency, virtual currency, or cryptocurrency. These are accurate descriptions; however, it might be easier if we think of Bitcoin as an electronic asset. The word “currency” has lots of baggage that can confuse us when we are trying to understand Bitcoin. Comparing Bitcoin to conventional currencies causes one to ask, “What backs it?” (nothing) and, “Who sets the interest rate?” (no one). Both these questions don’t help you understand Bitcoin.

What Are Bitcoins

> Bitcoins are digital assets (‘coins’) whose ownership is recorded on an electronic ledger that is updated (almost) simultaneously on about 10,000 independently operated computers around the world that connect and gossip with each other. “The Basics of Bitcoins and Blockchains” by Antony Lewis

Bitcoin’s blockchain is the electronic ledger described in the quote above. Transactions on the blockchain record the transfer of ownership of bitcoins. Software running on a participant’s computer implements a protocol to validate transactions. Each machine that runs the software forms a ‘node’ on the network. Each network node validates all pending transactions and updates its own copy of the ledger with validated blocks of confirmed transactions. Some nodes, called miners, work to bundle together valid transactions into blocks and then distribute those blocks to other nodes across the network.

Who Can Use Bitcoins

Anyone can buy, own, and send Bitcoin. Bitcoins’ blockchain records every transaction and makes it public. There is a common misconception that Bitcoin encrypts its blockchain. This is not so. Bitcoin designed its blockchain so that everyone can see all the details of all the transactions.

Follow Along

Basic Transactions

We need to implement a simple system for transactions in blockchain. This is one of the primary reasons for blockchain's existence, so we should at least be able to implement a simplified version in code.

Given we have a `Blockchain` class, we need to add a method to `Blockchain` that will add a transaction. For our purposes, transactions will be simple. They need to have a:

1. sender
2. recipient
3. amount

Our method will be simple, it:

1. accepts these three arguments
2. appends them as a new dictionary in the list of transactions
3. returns the index of the block that this transaction will be saved in (which is the index of the block currently being mined)

Credit for Mining

There are a few ways we could test our `new_transaction` method. One of the simplest is to call `new_transaction` from the `mine` method.

Miners expect to get rewarded for the electricity they consumed to find a proof for the next block. We should modify our `mine` function to add this.

We want to add a call to `new_transaction` if the proof is confirmed to be valid, but *before* the block is created because we want to make sure that this transaction is included in the block. We will identify the sender as `"0"` to indicate that we're creating new coins that didn't exist before. The recipient is the `id` of the miner and the amount is whatever we choose as an appropriate reward for this work. Many blockchain systems adjust this over time on a pre-planned schedule, but we can just give a reward of 1 coin for now.

Here is what the call to `new_transaction` would look like:

Copy

```
blockchain.new_transaction(
    sender="0",
    recipient=node_identifier,
    amount=1,
)
```

Transactions Endpoint

In addition to transactions for miner rewards, We also need to able to add transactions for when coins are being sent from one user to another. To do this, we can add an endpoint to collect the data from a POST and feed it to the method we've written.

It's important to note that this system is completely insecure. While these transactions will be permanently written into the blockchain ledger, we don't yet have a system to prevent anyone from creating a transaction that gives them all the money they want - or takes it from someone else.

We also don't have anything to prevent spending money you don't have, or double-spending it.

In real blockchains, these problems are prevented by a fairly complicated set of systems. To create a transaction, you need to spend the entirety of a previous, unspent transaction by proving you have access to that transaction. Often this is done with a system of public-private key encryption.

For our demonstration, we just need to be able to add a few transactions so that we can see them in the ledger and work with them.

Challenge

1. Give a few reasons why this simplified transactions system has security issues.

Learn to derive a user's account balance by examining data in a simplified blockchain ledger

Overview

Wallets

You may think, like cash, Bitcoin wallets store actual bitcoins. But, if you think deeper, if that were true, you could double your number of bitcoins just by copying your wallet. This wouldn't be a workable solution for digital currency.

That still leaves us with the question, where are bitcoins stored? Bitcoin's blockchain records the ownership of bitcoins. Bitcoin replicates its blockchain on over 10,000 computers around the world and contains every Bitcoin transaction ever. So, you can inspect that database and check how many bitcoins are associated with a specific address. As an example, Bitcoin's blockchain would show that the address `1Jco97X5FbCkev7lsVDpRtpNNi4zX6Wy4r` received 2.5 BTC. It would also show that those 2.5 BTC had not been sent elsewhere. To reiterate, Bitcoin's blockchain is not a store of account balances, it stores transactions. So, to get the actual balance of an account, you must inspect all the inbound and outbound transactions through that account and derive the balance from that information.

So, what does a bitcoin wallet store? Bitcoin wallets store private keys and the underlying software allows a wallet user to see their account balances and to make payments. Cloning a wallet would merely clone your private keys and would not change Bitcoin's blockchain.

Software Wallets

Bitcoin wallets are apps that can at least achieve the ability to:

- create new Bitcoin addresses and store the corresponding private keys
- display your addresses to someone who wants to send you a payment
- display how many bitcoins are in your addresses
- make Bitcoin payments

We will now dive deeper into each of these functionalities:

Address Creation

Creating a new Bitcoin address is done offline. A Bitcoin address is a pair of public and private keys. Here is an example of how you could generate an address:

1. Pick a random number between 1 and (2 ^ 256 - 1). This is your private key.
2. Mathematically manipulate that number a bit to generate a public key
3. Hash your public key twice to create your Bitcoin address.
4. Save the private key and its corresponding address.

Unlike creating an account with your bank or Twitter, where you ask a third party to create an account for you, you can assign yourself a Bitcoin address without asking or checking with anyone to see if it is already taken. How does that work, you might wonder? What if someone else already chose your private key? This is unlikely. 2 ^ 256 is a huge number with 78 digits and you can pick any number up to that. Theoretically, a person could generate billions of accounts per second and check them for coins to steal. The problem is that the number of valid accounts is so large that they would do this forever before finding a single account that has been used before. In practice, there are some flaws in random number generation when picking your private key that can be exploited to reduce the search space for the potential thief.

Address Display

To receive bitcoins, you need to tell the sender your address. Just like if you want someone to transfer money to your bank account, the sender would need to know your bank's routing number and your account number. Below, is an example Bitcoin address:

"1LfSBaeySpe6UBw49oH9VLSGonPvujmhFXV"

As you can see, it's just a text string of characters. There are many ways you could share this address, but a popular way is showing it as a QR code. QR codes are just text, encoding visually so QR code scanners can quickly read it and convert it back into text.

You can also just copy and paste the address itself.

Account Balance

In order for a wallet to work, it needs access to a current version of the blockchain so it knows about all the transactions related to the addresses it is tracking. The wallet can either store the entire blockchain and keep it current (called a full node wallet). Or, it can connect to a node somewhere on the network and let that node do the heavy lifting (called a lightweight wallet).

A full node wallet would have to contain over 100 gigabytes of data and would require a constant connection to other Bitcoin nodes. This is a serious constraint, especially on mobile phones. Most wallet software is lightweight and connects to a server which hosts the blockchain. The software asks the server for all the transactions related to a specific address.

Making Payments in Bitcoin

A good wallet also needs to make payments. They make payments using a bundle of data called a transaction. The transaction references the coins that will be spent (which are the unspent outputs of previous transactions), and also references which accounts it will send the coins to (these are the new outputs). Next, the transaction digitally signed using the private keys of the addresses that hold the coins being transacted. Once the transaction is signed, it is sent to other network nodes, and eventually a miner will add the transaction to a block.

Follow Along

Account Balances

As was explained above, an account balance is not a value that is stored in a blockchain. Instead, it is derived by adding up all of the transactions that give coins to a user, and subtracting all the coins given away in other transactions.

Consider the following example:

```
{
    chain: [
        {
            index: 1,
            previous_hash: 1,
            proof: 100,
            timestamp: 1571852367.484206,
            transactions: [ ]
        },
        {
            index: 2,
            previous_hash:
"ddf1adddad9af96695fda647492897f058aa702f806d7eaa21dfd46ecab0fcd1",
            proof: 24368051,
            timestamp: 1571852436.924649,
            transactions: [
                {
                    amount: 1,
                    recipient: "Brian",
                    sender: "0"
                }
            ]
        },
        {
            index: 3,
            previous_hash:
"2fa2bcc7b423d5d74d621835b098842cf9dd34591bfc0e68800a41cf20b2ec90",
            proof: 8132268,
            timestamp: 1571852467.247827,
            transactions: [
                {
                    amount: 1,
                    recipient: "Brian",
                    sender: "0"
                }
            ]
        },
        {
            index: 4,
            previous_hash:
"3f4b18d04371d8ce3129643ccf5ad46bb9943a87adf8c5addded0d3612128f59",
            proof: 1301845,
            timestamp: 1571852472.199991,
            transactions: [
                {
                    amount: 1,
                    recipient: "Brian",
                    sender: "0"
                }
            ]
        },
        {
            index: 5,
            previous_hash:
"9177932144818f9c3072d11849251bdd31096621162f0e081016fa59e25010d2",
            proof: 13176802,
            timestamp: 1571852599.8256152,
            transactions: [
                {
                    amount: "3",
                    recipient: "Beej",
                    sender: "Brian"
                },
                {
                    amount: 1,
                    recipient: "Brian",
                    sender: "0"
                }
            ]
        },
        {
            index: 6,
            previous_hash: 
"1e110e46bd7a6a86cd39c3adae667439a40e31b20db3b314bed5b1fa56c746ea",
            proof: 41571496,
            timestamp: 1571852940.9420102,
            transactions: [
                {
                    amount: ".5",
                    recipient: "Brady",
                    sender: "Beej"
                },
                {
                    amount: ".5",
                    recipient: "Elissa",
                    sender: "Beej"
                },
                {
                    amount: ".5",
                    recipient: "Tom",
                    sender: "Beej"
                },
                {
                    amount: 1,
                    recipient: "Brian",
                    sender: "0"
                }
            ]
        }
    ],
    length: 6
}
```

The first block is the genesis block. It does not have any transactions. We could have used this to give ourselves a few billion LambdaCoins, but that's probably not a good idea. It would seem unfair and might make people less interested in mining on our blockchain.

The next blocks have only one transaction each - the one created by the function we just wrote when a miner successfully submits a new proof and is rewarded for it.

After a few of these, we see our first person to person transaction. Brian has transferred three coins to Beej. In the block after, Beej has shared his newfound wealth with Elissa, Brady, and Tom. Note that in both of these blocks, we also see a transaction where Brian has been rewarded for continuing to mine blocks.

To find anyone's balance, we have to tally all of these transactions up? So how many coins to Brian, Beej, Elissa, Brady, and Tom each have?

Challenge

Use the example transactions above in order to tally how many coins Brian, Beej, Elissa, Brady, and Tom each have.

Review

Class Recordings

You can use class recordings to help you master the material.

Blockchain II - Distribution for CSPT4 with Tim Roy

Blockchain II - Distribution for CSPT4 with Tim Roy

All previous recordings

Demonstrate Mastery

To demonstrate mastery of this module, you need to complete and pass a code review on each of the following:

- Objective challenge:
    1. Give a few reasons why this simplified transactions system has security issues.
- Objective challenge:

    Use the example transactions above in order to tally how many coins Brian, Beej, Elissa, Brady, and Tom each have.