Trenton Fleming

Exercise 2 report

I approached the problem by using an hashmap as my data structure to represent the ledger. Each block holds a transaction. Each transaction holds an array of accounts and utxos. When adding a transaction, the appropriate previous transactions are accessed and sum all of their outputs are used to verify that they match the output of transaction being added.

As each UTXO is summed from previous transactions, it is also marked as spent. This was initially a problem because If the current transaction turns out to be invalid and transactions were already marked as spent, then those mutations to the ledger were never undone. This was the first time I had a project where I had to keep a record of transformations and undo them as necessary.

Unfortunately, all testing has been done manually. To test transaction parsing, I made lexically mal formed transactions. After testing transaction parsing, I tested the semantics of each new transaction. I’ve written sample ledgers to emulate double spending and incorrect totals.

I’ve gained a better understanding of how bitcoin works. Reading about the process is a bit abstract. However, emulating bitcoin makes it clear how UTXOs are used to verify new transactions. I also learned about making reversible actions.