1. Reading Ethereum Blocks

function to get data about blocks on the Ethereum blockchain

In order to use these functions, you will need to connect to an Ethereum node with an open RPC interface. We recommend using one of the free providers you signed up for in Module 1.

On Ethereum, as in almost all blockchains, each block is produced by single block producer, and that block producer has full authority to pack transactions into a block in any way they choose.

Historically, many block builders used a simple, greedy procedure in order to decide which transactions would eventually make it into the block.

Before EIP-1559, the common Ethereum clients like geth used the "greedy" algorithm to include transactions in decreasing order of gasPrice. Now, it's still a possible setting in block builders. The Flashbots builder has three different block building strategies, "mev-geth", "greedy" and "greedy buckets".

Just because it's the default setting, does no mean that all miners follow this rule, however. We can't peek into a miner's brain to figure out what rule they're using, but we can look at a block and see if the transactions were packed greedily, i.e., are the transactions ordered in decreasing order of gasPrice.

In this assignment you're going to use the Python web3 library to grab blocks from the Ethereum blockchain, and see if the transaction in those blocks are ordered in decreasing order of fee. If they're not, that will indicate that the miner ordered the transactions according to some other mechanism (e.g. out of band payments).

assignment

Your assignment is to write a function called is_ordered_block(block_num) that takes a block number and returns a boolean value indicating whether *all* the transactions in the block were ordered in decreasing order of fee.

We have provided a template [is_ordered.py].

You should finish the function `is_ordered_block(block_num)`

Your solution will have to call get_block()to get all the transactions in the block, and then get_transaction() for each transaction in the block.

EIP1559

EIP-1559 which, was introduced as part of the London Hard Fork, changed the fee mechanism significantly. Before EIP-1559, a "greedy" algorithm would order transactions by the "gasPrice" field.

After EIP-1559, however, there is now a new type of transaction (called a type 2 transaction), which includes the fields maxPriorityFeePerGas and maxFeePerGas.

It's possible for a transaction to set both the gasPrice and maxPriorityFeePerGas. In this case, this is a type 2 transaction, and you should use maxPriorityFeePerGas and ignore gasPrice.

The **priority** fee paid on type 2 transactions is:

```
min( maxPriorityFeePerGas, maxFeePerGas-baseFeePerGas )
```

The **total** fee paid on type 2 transactions is