Summer of Bitcoin Code Challenge

Thank you for applying to the Summer of Bitcoin program.

This task will give you a chance to showcase your abilities and give us a sense of how you approach problems and write code. The challenge is open-ended and allows for multiple approaches.

The problem

Bitcoin miners construct blocks by selecting a set of transactions from their mempool. Each transaction in the mempool:

- includes a *fee* which is collected by the miner if that transaction is included in a block
- has a *weight*, which indicates the size of the transaction
- may have one or more *parent transactions* which are also in the mempool

The miner selects an ordered list of transactions which have a combined weight below the maximum block weight. Transactions with parent transactions in the mempool may be included in the list, but only if all of their parents appear *before them* in the list.

Naturally, the miner would like to include the transactions that maximize the total fee.

Your task is to write a program which reads a file mempool.csv, with the format:

`<txid>,<fee>,<weight>,<parent_txids>`

- `txid` is the transaction identifier
- `fee` is the transaction fee
- `weight` is the transaction weight
- `parent_txids` is a list of the txids of the transaction's **unconfirmed** parent transactions (confirmed parent transactions are not included in this list). It is of the form: `<txid1>;<txid2>;...`

The output from the program should be txids, separated by newlines, which make a valid block, maximizing the fee to the miner. Transactions **MUST** appear in order (no transaction should appear before one of its parents).

We've included a non-working `block_sample.txt` file to demonstrate the expected format.

Input file

Here are two lines of the `mempool.csv` file:

```
2e3da8fbc1eaca8ed9b7c2db9e6545d8ccac3c67deadee95db050e41c1eedfc0,452,1620,
```

This is a transaction with txid `2e3da8...`, fees of 452 satoshis, weight of 1620, and no parent transactions in the mempool.

```
9d317fb308fd5451fd0ec612165638cb9e37bd8aa8918dff99a48fe05224276f,350,1400,
288ea91bb52d8cb28289f4db0d857356622e39e78f33f26bf6df2bbdd3810fad;b5b993bda
3c23bdefe4a1cf75b1f7cbdfe43058f2e4e7e25898f449375bb685c;c1ae3a82e52066b670
e43116e7bfbcb6fa0abe16088f920060fa41e09715db7d
```

This is a transaction with txid `9d317f...`, fees of 350 satoshis, weight of 1400 and three parent transactions in the mempool with txids `288ea9....`, `b5b993...` and `c1ae3a...`

Parsing the input file

Here is some sample Python code to parse the input file. You may use this snippet in your solution if you want:

```
class MempoolTransaction():
    def __init__(self, txid, fee, weight, parents):
        self.txid = txid
        self.fee = int(fee)
        # TODO: add code to parse weight and parents fields

def parse_mempool_csv():
    """Parse the CSV file and return a list of MempoolTransactions."""
    with open('mempool.csv') as f:
        return([MempoolTransaction(*line.strip().split(',')) for line in
f.readlines()])
```

Hints

- The total weight of transactions in a block must not exceed 4,000,000 weight. For this exercise assume that there is no coinbase transaction.
- A transaction may only appear in a block if all of its parents appear *earlier* in the block.

General advice

- Spend no more than two to three days on the exercise. The idea is not that you come up with a perfect solution, but that you think about your approach. First, make a naive solution that constructs a valid block, then iterate to improve it.
- We're most familiar with Python, C++, JavaScript, Java, Rust, Scheme, Lisp, Ruby, and Elixir and would prefer to receive solutions in those languages. If you'd like to use a different language, please check with us first to make sure we'll be able to review it!
- You should be able to explain your reasoning, design decisions, and trade-offs.

What to send us

- the source code for your solution (sending a GitHub repo URL works as well -- you will need to do this if you used JS)
- the output from running the program with `mempool.csv` as `block.txt`.
- You may optionally also include `.git` files to show your commit history.