Punk Floor Price Algorithm Update

Aim

The purpose of this document is to suggest a more efficient algorithm for the computation of the Cryptopunks floor price.

Rationale

The old method took about 2.5 minutes to compute using an AWS t2.xlarge instance (4 vCPUs, 16GiB memory, "Moderate" network performance ~300Mb/s)

The new method should be able to run the same computation in less than a second. This is based on local testing on a Macbook Air M1 (2020) with 50Mb/s network connection.

Old approach

1. Get all offers - Use green threads to get offers [1] on all punks using the API [2]
2. Filter invalid offers - Remove offers that are not on sale [3] or are on private sale [4]
3. Sorting - Sort the list of offers in ascending order of minValue
4. Median - Compute the median of the first 4 (cheapest) punks
5. Return the computed median

New approach

1. Initial state - Use green threads to get and store only valid offers on all punks using API in the database
2. Update state - Watch events [4] emitted by the smart contract and update offer [5] on that punk in the database
3. Get floor price - When current floor price is demanded, we will follow the old approach but instead of getting all offers using the API, we will use the database.

Edge cases - It is possible that we miss some events. Therefore it is important to periodically update state using the the old approach.

Glossary

1. Offer structure

```
struct Offer {
  bool isForSale;
  uint punkIndex;
```

```
  address seller;
  uint minValue;          // in ether
  address onlySellTo;     // specify to sell only to a specific person
}
```

1. API - API refers to referencing public data on the Blockchain using a web3 wrapper implemented in a certain programming language. Python uses web3py for example.
2. Not on sale - This implies that bool isForSale is currently false
3. Private sale - This implies that address onlySellTo ≠ address(0)
4. Events watched

```
event PunkTransfer(address indexed from, address indexed to, uint256 punkIndex);
event PunkOffered(uint indexed punkIndex, uint minValue, address indexed toAddress);
event PunkBidEntered(uint indexed punkIndex, uint value, address indexed fromAddress);
event PunkBidWithdrawn(uint indexed punkIndex, uint value, address indexed fromAddress);
event PunkBought(uint indexed punkIndex, uint value, address indexed fromAddress, address indexed toAddress);
event PunkNoLongerForSale(uint indexed punkIndex);
```

5. Update offer algorithm [function definitions are subject to change]

```
def handle_event(event: dict):
  punk_index = get_punk_index(event)
  latest_offer = get_offer(punk_index)
  if is_private_sale(latest_offer) or not_on_sale(latest_offer):
    if in_database(punk_index):
      db.delete(punk_index)
  else:
    if in_database(punk_index):
      db.update(punk_index, latest_offer)
    else:
      db.insert(punk_index, latest_offer)
```