8/5/2022

Projet BLOCKCHAIN

Manuel d'utilisation de la mini blockchain

MEMBRES DU GROUPE:

ELBEBLAWY RAMI
NGUYEN TO DUNG
TRAN THIBAULT
WANE YEYA

Table des matières

Objectif

Notre but est d'implémenter une mini blockchain en Python.

Réalisation

Nous avons tout d'abord créé un réseau P2P. Ainsi chaque nouveau miner est connecté aux autres miners présents dans le réseau. De plus, tout message envoyé par un wallet à un miner est transmis à tous les autres miners.
Ensuite, nous avons regroupé les transactions envoyées par les wallets dans un bloc. Pour ce faire, nous avons utilisé l'algorithme de Proof-Of-Work avec une difficulté fixée à 4.
Pour synchroniser la blockchain de tous les miners, chaque miner a la possibilité de lancer une commande /RESOLVE qui leur permettra de se mettre à jour.
Pour finir, nous avons implémenté un client léger pour bitcoin en utilisant la structure de l'arbre de Merkel. Ainsi, chaque wallet peut vérifier si une transaction est présente dans la blockchain s après que la taille de la blockchain ait augmenté de 6.

Description des commandes

Nous disposons de deux personnages: un miner et un wallet. Chaque personnage peut effectuer diverses commandes.

I. Miner

Pour lancer un miner, il suffit de lancer la commande :

```
>>> python miner.py [liste de numéros de port]
```

Exemple : python miner.py 8000 ou python miner.py 8001 8000

Un miner peut effectuer 3 opérations sur son interface :

1. Voir l'état actuel de la blockchain avec la commande : /CHAIN
2. Voir l'état du bloc courant : /BLOCK
3. Miner avec la commande: /MINE
4. Mettre à jour son blockchain avec la commande : /RESOLVE

II. Wallet

Pour lancer un wallet, il suffit de lancer la commande :

```
>>> python wallet.py [nom_wallet] [numero_port_miner]
```

Exemple : python wallet.py Alice 8000

Un wallet peut effectuer les opérations suivantes :

1. Lancer une transaction avec la commande : /TRANS [sender] [recipient] [value]

   (Exemple : /TRANS Alice Bob 2)

2. Afficher la liste des transactions effectuées: /LIST

3. Vérifier qu'une transaction est bien dans la blockchain: /MP [transaction_id]
   La commande retourne TRUE ou FALSE.

Manuel d’utilisation

Nous allons maintenant tester le fonctionnement de chaque exercice:

Exercice 1: Maillage du réseau

1. Un miner A se connecte au réseau sur le port 8000.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8000
Miner 8000 is running!
```

2. Un miner B se connecte au réseau sur le port 8001 et indique sa présence au miner A.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8001 8000
Connected to 8000
List of miners: ['8000']
Miner 8001 is running!
```

3. Un miner C se connecte sur le port 8002 et indique sa présence au miner B.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8002 8001
Connected to 8001
List of miners: ['8001']
Miner 8002 is running!
Connected to 8000
8002 is already connected with 8001
List of miners: ['8001', '8000']
List of miners: ['8001', '8000']
```

On constate que le miner B va indiquer au miner A, la présence du miner C. Et inversement, le miner A est au courant de la présence du miner C (voir image ci-dessous).

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8000
Miner 8000 is running!
Miner 8001 is connected
List of miners: ['8001']
Miner 8002 is connected
List of miners: ['8001', '8002']
```

4. Le miner C est déconnecté du réseau. Les miners A et B en sont alertés.

```
(base) admin@a-144106 ~/Blockchain/final $
python miner.py 8000
Miner 8000 is running!
Miner 8001 is connected
List of miners: ['8001']
Miner 8002 is connected
List of miners: ['8001', '8002']
Error to handle miner connection: Miner 800
2 is closed
Miner 8002 is removed
List of miners: ['8001']
```

```
(base) admin@a-144106 ~/Blockchain/final $
 python miner.py 8001 8000
Connected to 8000
List of miners: ['8000']
Miner 8001 is running!
Miner 8002 is connected
List of miners: ['8000', '8002']
Error to handle miner connection: Miner 80
02 is closed
Miner 8002 is removed
List of miners: ['8000']
```

Exercice 2: Gestion des wallets

5. Le wallet Alice se connecte au réseau sur le nœud du miner A (port 8000). Il est informé de la présence du miner B (port 8001).

```
(base) admin@a-144106 ~/Blockchain/final $ python wallet.py Alice 8000
Connected to miner 8000
List of miners: ['8001']
```

Inversement, le miner A est informé de la présence du wallet Alice sur le réseau.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8000
Miner 8000 is running!
Miner 8001 is connected
List of miners: ['8001']
Miner 8002 is connected
List of miners: ['8001', '8002']
Error to handle miner connection: Miner 8002 is closed
Miner 8002 is removed
List of miners: ['8001']
Welcome Alice to MINER-8000
```

6. Le wallet Alice envoie un message au miner A (8000).

```
(base) admin@a-144106 ~/Blockchain/final $ python wallet.py Alice 8000
Connected to miner 8000
List of miners: ['8001']
Hello miner A !
```

7. Tous les miners du réseau à savoir le miner A et le miner B reçoivent le message de Alice.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8000
Miner 8000 is running!
Miner 8001 is connected
List of miners: ['8001']
Miner 8002 is connected
List of miners: ['8001', '8002']
Error to handle miner connection: Miner 8002 is closed
Miner 8002 is removed
List of miners: ['8001']
Welcome Alice to MINER-8000
Alice : Hello miner A !
```

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8001 8000
Connected to 8000
List of miners: ['8000']
Miner 8001 is running!
Miner 8002 is connected
List of miners: ['8000', '8002']
Error to handle miner connection: Miner 8002 is closed
Miner 8002 is removed
List of miners: ['8000']
Alice : Hello miner A !
```

8. Le miner A auquel Alice est connecté, se déconnecte du réseau. Alors le wallet Alice est connecté à un autre miner (le miner B dans notre cas).

```
(base) admin@a-144106 ~/Blockchain/final $ python wallet.py Alice 8000
Connected to miner 8000
List of miners: ['8001']
Hello miner A !
Miner 8000 closed
Connecting to new miner...
Connected to miner 8001
```

Et le miner B est informé de la connexion du wallet Alice.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8001 8000
Connected to 8000
List of miners: ['8000']
Miner 8001 is running!
Miner 8002 is connected
List of miners: ['8000', '8002']
Error to handle miner connection: Miner 8002 is closed
Miner 8002 is removed
List of miners: ['8000']
Alice : Hello miner A !
Error to handle miner connection: Miner 8000 is closed
Miner 8000 is removed
List of miners: []
Welcome Alice to MINER-8001
```

Exercice 3: Proof-Of-Work

On repart de zéro. On crée un autre réseau avec deux miners A (port 8000) et B (port 8001) et un wallet Alice connecté au miner A.

1. Le wallet Alice effectue la transaction suivante avec la commande /TRANS Alice Bob 2: (sender: Alice, recipient: Bob, value: 2). Les deux miners reçoivent la transaction et l'ajoute à leur bloc actuel.

```
(base) admin@a-144106 ~/Blockchain/final $
python miner.py 8000
Miner 8000 is running!
Miner 8001 is connected
List of miners: ['8001']
Welcome Alice to MINER-8000
Alice : /TRANS Alice Bob 2
The id of the next transaction is  1
The current length of the blockchain is  1
Transaction will be added to the Block  2
New transaction id added:  {'1': 1}
```

```
(base) admin@a-144106 ~/Blockchain/final $
python miner.py 8001 8000
Connected to 8000
List of miners: ['8000']
Miner 8001 is running!
Alice : /TRANS Alice Bob 2 1 1
Transaction will be added to the Block  2
New transaction id added:  {'1': '1'}
```

De plus, l'index de la transaction est renvoyé au wallet Alice pour faciliter la demande de la preuve de Merkle par la suite.

```
(base) admin@a-144106 ~/Blockchain/final $ python wallet.py Alice 8000
Connected to miner 8000
List of miners: ['8001']
/TRANS Alice Bob 2
Transaction id is  1
```

2. Un nouveau wallet Bob rejoint le réseau à travers le miner B (port 8001) et effectue une transaction: (sender: Bob, recipient: Alice, value: 4)

```
(base) admin@a-144106 ~/Blockchain/final $ python wallet.py Bob 8001
Connected to miner 8001
List of miners: ['8000']
/TRANS Bob Alice 4
Transaction id is  2
```

De même, tous les miners ajoutent la nouvelle transaction à leur bloc actuel.

```
(base) admin@a-144106 ~/Blockchain/final $ python m
iner.py 8000
Miner 8000 is running!
Miner 8001 is connected
List of miners: ['8001']
Welcome Alice to MINER-8000
Alice : /TRANS Alice Bob 2
The id of the next transaction is  1
The current length of the blockchain is  1
Transaction will be added to the Block  2
New transaction id added:  {'1': 1}
Bob : /TRANS Bob Alice 4 2 1
Transaction will be added to the Block  2
New transaction id added:  {'1': 1, '2': '1'}
```

```
(base) admin@a-144106 ~/Blockchain/final $ python
miner.py 8001 8000
Connected to 8000
List of miners: ['8000']
Miner 8001 is running!
Alice : /TRANS Alice Bob 2 1 1
Transaction will be added to the Block  2
New transaction id added:  {'1': '1'}
Welcome Bob to MINER-8001
Bob : /TRANS Bob Alice 4
The id of the next transaction is  2
The current length of the blockchain is  1
Transaction will be added to the Block  2
New transaction id added:  {'1': '1', '2': 1}
```

3. Le miner A décide de miner et d'ajouter son bloc actuel à la blockchain grâce à la commande /MINE. Tout d'abord, on va synchroniser notre blockchain avec les blockchain des autres miners. Ensuite, le miner va passer par l'algorithme du proof of work (Nous utilisons une difficulté de 4). Puis, le miner aura une transaction bonus de 1 en guise de récompense. Enfin, le bloc actuel du miner sera ajouté à la blockchain. Nous pouvons bien distinguer dans la liste des transactions, les deux transactions effectuées par Alice et Bob et la transaction bonus du miner A qui porte le nom 8000.

```
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  3
Transaction will be added to the Block  2
{'message': 'Forged new block.', 'index': 2, 'transactions': [{'id': 1,
'sender': 'Alice', 'recipient': 'Bob', 'value': '2'}, {'id': '2', 'sende
r': 'Bob', 'recipient': 'Alice', 'value': '4'}, {'id': 3, 'sender': 0, '
recipient': '8000', 'value': 1}, {'id': 3, 'sender': 0, 'recipient': '80
00', 'value': 1}], 'proof': 155, 'previous_hash': 'dd1d95119df39b5f9fa8e
150eec5ce08871591c6392de542315dc7688954b4bd'}
```

4. Le miner A consulte son bloc actuel avec la commande /BLOCK. Comme attendu, le bloc est vide car le miner A vient tout juste d'ajouter son bloc à la blockchain.

```
/BLOCK
[]
```

5. Le miner A décide de consulter l'état de la blockchain avec la commande /CHAIN. On retrouve bien les deux blocs (le bloc initial et le bloc du miner A), ainsi que le nonce trouvé qui vaut 65 et le hash du bloc précédent.

```
/CHAIN
[{'index': 1, 'timestamp': 1652039218.067582, 'tran
sactions': [], 'nonce': 0, 'previous_hash': '00'},
{'index': 2, 'timestamp': 1652039302.3848631, 'tran
sactions': [{'id': 1, 'sender': 'Alice', 'recipient
': 'Bob', 'value': '2'}, {'id': '2', 'sender': 'Bob
', 'recipient': 'Alice', 'value': '4'}, {'id': 3, '
sender': 0, 'recipient': '8000', 'value': 1}, {'id'
: 3, 'sender': 0, 'recipient': '8000', 'value': 1}]
, 'nonce': 65, 'previous_hash': 'ac7afad4ce4f521f63
e9e9e05345daaabdcd9514d8fdae0dc7234b03014352be'}]
```

6. Le miner B (port 8001) décide de consulter son bloc actuel. Il possède toujours les deux transactions lancées par Alice et Bob car il n'a pas encore synchronisé sa blockchain.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8001 8000
Connected to 8000
List of miners: ['8000']
Miner 8001 is running!
Alice : /TRANS Alice Bob 2 1 1
Transaction will be added to the Block  2
New transaction id added:  {'1': '1'}
Welcome Bob to MINER-8001
Bob : /TRANS Bob Alice 4
The id of the next transaction is  2
The current length of the blockchain is  1
Transaction will be added to the Block  2
New transaction id added:  {'1': '1', '2': 1}
/BLOCK
[{'id': '1', 'sender': 'Alice', 'recipient': 'Bob', 'value': '2'}, {'id': 2, 'sender'
: 'Bob', 'recipient': 'Alice', 'value': '4'}]
```

7. Le miner B synchronise sa blockchain avec la commande /RESOLVE. Il obtient un message qui lui indique que sa blockchain a été remplacée.

```
/RESOLVE
Our chain was replaced
```

8. Le miner B consulte à nouveau l'état de son bloc actuel qui est vide désormais.

```
/RESOLVE
Our chain was replaced
/BLOCK
[]
```

9. Le miner B consulte l'état de sa blockchain qui est identique à celui du miner A.

```
/RESOLVE
Our chain was replaced
/BLOCK
[]
/CHAIN
[{'index': 1, 'timestamp': 1652039218.067582, 'tran
sactions': [], 'nonce': 0, 'previous_hash': '00'},
{'index': 2, 'timestamp': 1652039302.3848631, 'tran
sactions': [{'id': 1, 'sender': 'Alice', 'recipient
': 'Bob', 'value': '2'}, {'id': '2', 'sender': 'Bob
', 'recipient': 'Alice', 'value': '4'}, {'id': 3, '
sender': 0, 'recipient': '8000', 'value': 1}, {'id'
: 3, 'sender': 0, 'recipient': '8000', 'value': 1}]
, 'nonce': 65, 'previous_hash': 'ac7afad4ce4f521f63
e9e9e05345daaabdcd9514d8fdae0dc7234b03014352be'}]
```

Exercice 4: Merkle Proof

On repart de zéro. On crée un autre réseau avec deux miners A (port 8000) et un wallet Alice connecté au miner A.

1. Le wallet Alice lance une transaction: (sender: Alice, recipient: Bob, value: 2) avec la commande /TRANS Alice Bob 2. Elle enregistre l'index de la transaction qu'elle vient d'effectuer à savoir l'index 1 comme indiqué sur l'image ci-dessous.

```
(base) admin@a-144106 ~/Blockchain/final $ python wallet.py Alice 8000
Connected to miner 8000
/TRANS Alice Bob 2
Transaction id is  1
```

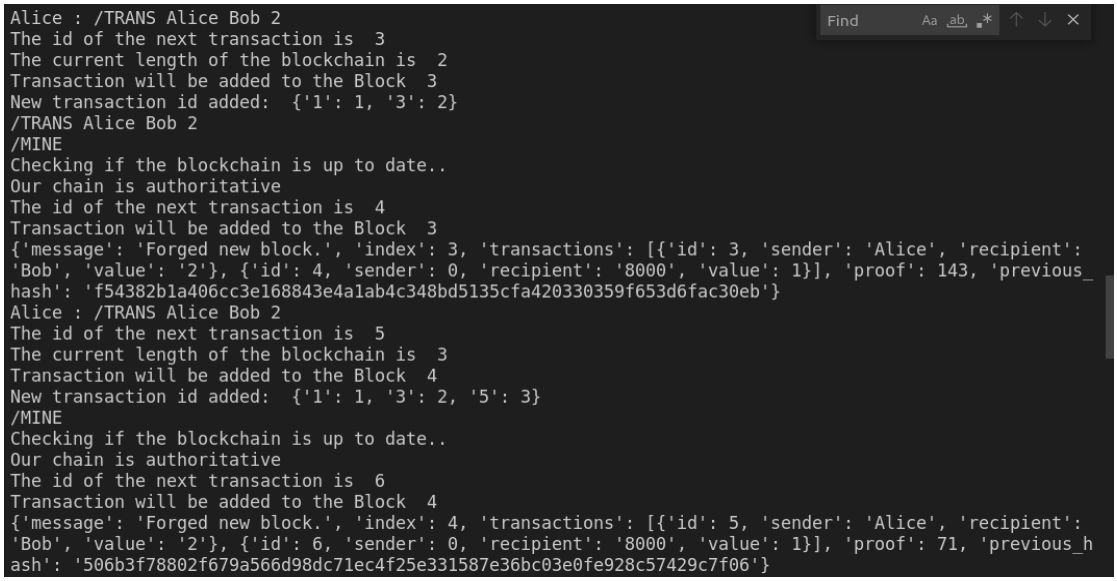
2. Le miner A décide de miner.

```
(base) admin@a-144106 ~/Blockchain/final $ python miner.py 8000
Miner 8000 is running!
Welcome Alice to MINER-8000
Alice : /TRANS Alice Bob 2
The id of the next transaction is  1
The current length of the blockchain is  1
Transaction will be added to the Block  2
New transaction id added:  {'1': 1}
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  2
Transaction will be added to the Block  2
{'message': 'Forged new block.', 'index': 2, 'transactions': [{'id': 1, 'sender': 'Alice', 'recipient':
'Bob', 'value': '2'}, {'id': 2, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 581, 'previous_
hash': '5d3dff9e023206e2d7672f714dc50fb65481b0a9d02f9cb49083b0126950f1e7'}
```

3. Alice souhaite vérifier à partir de la preuve de Merkel, de l'entête du bloc dans lequel se trouve sa transaction 1 et l'index de la transaction 1, si sa transaction de bien dans la blockchain. Pour ce faire à 6 reprises elle va lancer des transactions et à chaque reprise le miner va miner (pour qu'il ait un total de 6 blocs enregistrés dans la blockchain depuis qu'elle a effectué la transaction 1.)

```
/TRANS Alice Bob 2
Transaction id is  3
/TRANS Alice Bob 2
Transaction id is  5
/TRANS Alice Bob 2
Transaction id is  7
/TRANS Alice Bob 2
Transaction id is  9
/TRANS Alice Bob 2
Transaction id is  11
/TRANS Alice Bob 2
Transaction id is  13
/TRANS Alice Bob 2
Transaction id is  15
```

Les images ci-dessous sont les affichages lorsque le miner A va miner à chaque reprise.



```
Alice : /TRANS Alice Bob 2
The id of the next transaction is  3
The current length of the blockchain is  2
Transaction will be added to the Block  3
New transaction id added:  {'1': 1, '3': 2}
/TRANS Alice Bob 2
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  4
Transaction will be added to the Block  3
{'message': 'Forged new block.', 'index': 3, 'transactions': [{'id': 3, 'sender': 'Alice', 'recipient':
'Bob', 'value': '2'}, {'id': 4, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 143, 'previous_
hash': 'f54382b1a406cc3e168843e4a1ab4c348bd5135cfa420330359f653d6fac30eb'}
Alice : /TRANS Alice Bob 2
The id of the next transaction is  5
The current length of the blockchain is  3
Transaction will be added to the Block  4
New transaction id added:  {'1': 1, '3': 2, '5': 3}
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  6
Transaction will be added to the Block  4
{'message': 'Forged new block.', 'index': 4, 'transactions': [{'id': 5, 'sender': 'Alice', 'recipient':
'Bob', 'value': '2'}, {'id': 6, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 71, 'previous_h
ash': '506b3f78802f679a566d98dc71ec4f25e331587e36bc03e0fe928c57429c7f06'}
```

```
Alice : /TRANS Alice Bob 2
The id of the next transaction is  7
The current length of the blockchain is  4
Transaction will be added to the Block  5
New transaction id added:  {'1': 1, '3': 2, '5': 3, '7': 4}
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  8
Transaction will be added to the Block  5
{'message': 'Forged new block.', 'index': 5, 'transactions': [{'id': 7, 'sender': 'Alice', 'recipient':
'Bob', 'value': '2'}, {'id': 8, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 1818, 'previous
_hash': 'b1c3483a945ccabef1dffdf3872d08c9f646fefeec88607718c88233e2f19207'}
Alice : /TRANS Alice Bob 2
The id of the next transaction is  9
The current length of the blockchain is  5
Transaction will be added to the Block  6
New transaction id added:  {'1': 1, '3': 2, '5': 3, '7': 4, '9': 5}
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  10
Transaction will be added to the Block  6
{'message': 'Forged new block.', 'index': 6, 'transactions': [{'id': 9, 'sender': 'Alice', 'recipient':
'Bob', 'value': '2'}, {'id': 10, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 840, 'previous
_hash': 'f875c0699507ed8687856dbf70193cdc50cb8ea9dabcb1700292a31e8e64af65'}
```

.

```
{'message': 'Forged new block.', 'index': 7, 'transactions': [{'id': 11, 'sender': 'Alice', 'recipient':
 'Bob', 'value': '2'}, {'id': 12, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 111, 'previou
s_hash': 'c5a4d1dae0f8fc94f5313c90cda54dceaa343f349d2266fa681f8a000a876a6f'}
Alice : /TRANS Alice Bob 2
The id of the next transaction is  13
The current length of the blockchain is  7
Transaction will be added to the Block  8
New transaction id added:  {'1': 1, '3': 2, '5': 3, '7': 4, '9': 5, '11': 6, '13': 7}
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  14
Transaction will be added to the Block  8
{'message': 'Forged new block.', 'index': 8, 'transactions': [{'id': 13, 'sender': 'Alice', 'recipient':
 'Bob', 'value': '2'}, {'id': 14, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 722, 'previou
s_hash': 'e05aa2e00fcabeddc15dbf95ff72465ea6781a254cdc51a28403291c5bc0a3cf'}
Alice : /TRANS Alice Bob 2
The id of the next transaction is  15
The current length of the blockchain is  8
Transaction will be added to the Block  9
New transaction id added:  {'1': 1, '3': 2, '5': 3, '7': 4, '9': 5, '11': 6, '13': 7, '15': 8}
/MINE
Checking if the blockchain is up to date..
Our chain is authoritative
The id of the next transaction is  16
Transaction will be added to the Block  9
{'message': 'Forged new block.', 'index': 9, 'transactions': [{'id': 15, 'sender': 'Alice', 'recipient':
 'Bob', 'value': '2'}, {'id': 16, 'sender': 0, 'recipient': '8000', 'value': 1}], 'proof': 390, 'previou
s_hash': 'df1f023b54d2826ea19cd502a5c2c02a08b427aebdf3b5936c3c44851980885e'}
```

4. Alice demande la preuve de merkle pour la transaction 1 avec la commande: /MP 1.

```
/MP 1
Verifying if  /TRANS Alice Bob 2  is in the blockchain...
5b8a5fefe0d3866e872087cfd2cafdb92ae5370766b81886c9311fab997ffe15af0876596500d28f01aad72af332698c05c1c711
701f05ea3f2aad671f81199a
True
```

5. Le miner A après avoir reçu cette commande, va parcourir la blockchain et retrouver le bloc dans lequel se trouve la transaction d'index 1. Une fois avoir trouvé ce bloc, il va récupérer l'arbre de merkle associé à ce bloc pour calculer la merkle proof de la transaction. Ensuite, la merkle proof ainsi que l'entete de l'arbre de merkle va etre envoyé au wallet Alice. Pour finir, le wallet Alice dispose d'une méthode (check_merkle_proof dans la classe merkle_t.py) qui lui permet de vérifier si la transaction d'index 1 est bien présente dans le bloc renvoyé par le miner A. Dans notre cas, nous pouvons voir sur l'image ci-dessus que la réponse est True. Ainsi, la transaction d'index 1 se trouve bien dans la blockchain.

```
Alice : /MP 1
Our chain is authoritative
```

6. Alice souhaite avoir la preuve de merkle de la dernière transaction effectuée au miner A (la transaction d'index 15 avec la commande /MP 15.

```
/MP 15
The block is not valid yet. Try again later...
```

Elle ne pourra pas car le bloc n'a pas encore été validé. Il faudra attendre que 6 blocs aient été validés avant de pouvoir effectuer la vérification.