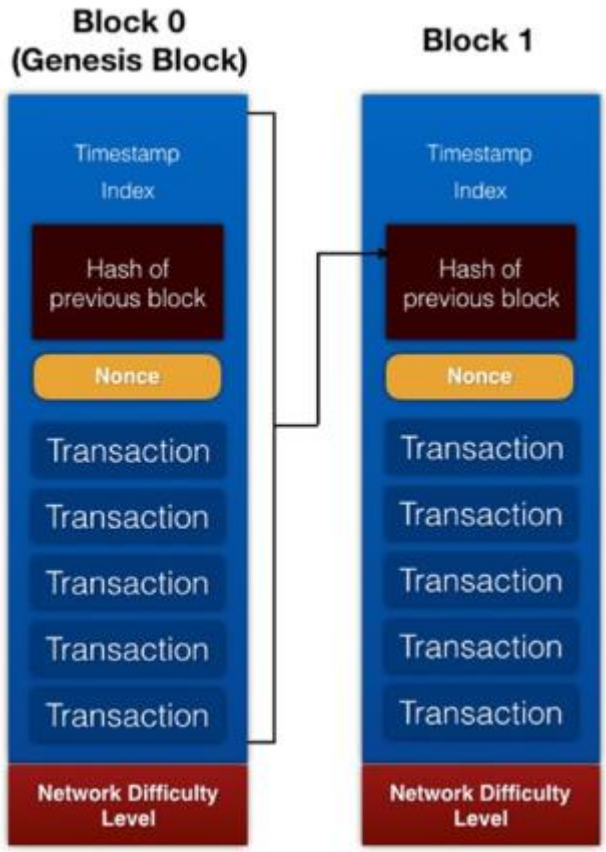
LAB 1: CÀI ĐẶT BLOCKCHAIN VỚI PYTHON

Sau bài thực hành này, sinh viên có thể

- Cài đặt được môi trường Flask
- Thu thập được Nonce
- Thêm block vào blockchain
- Triển khai blockchain class thành REST API
- Thêm giao dịch

1. Blockchain khái niệm



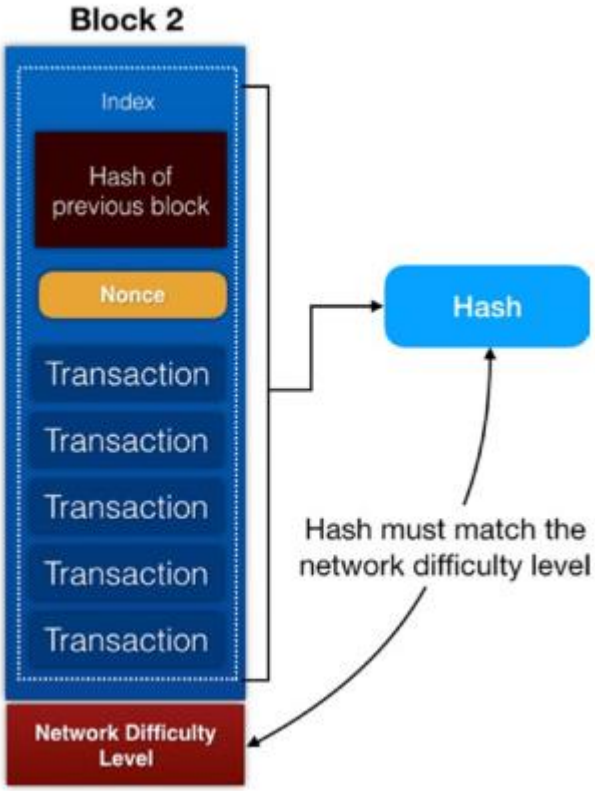
Trong đó:

- Timestamp: là thời gian block được thêm vào blockchain
- Index: là chỉ số xác định số thứ tự block
- Hash of the previous block: là kết quả băm (hash) của block trước. Ví dụ trong hình trên, một hash là việc băm timestamp, index, hash of previous block, nonce và các transaction.

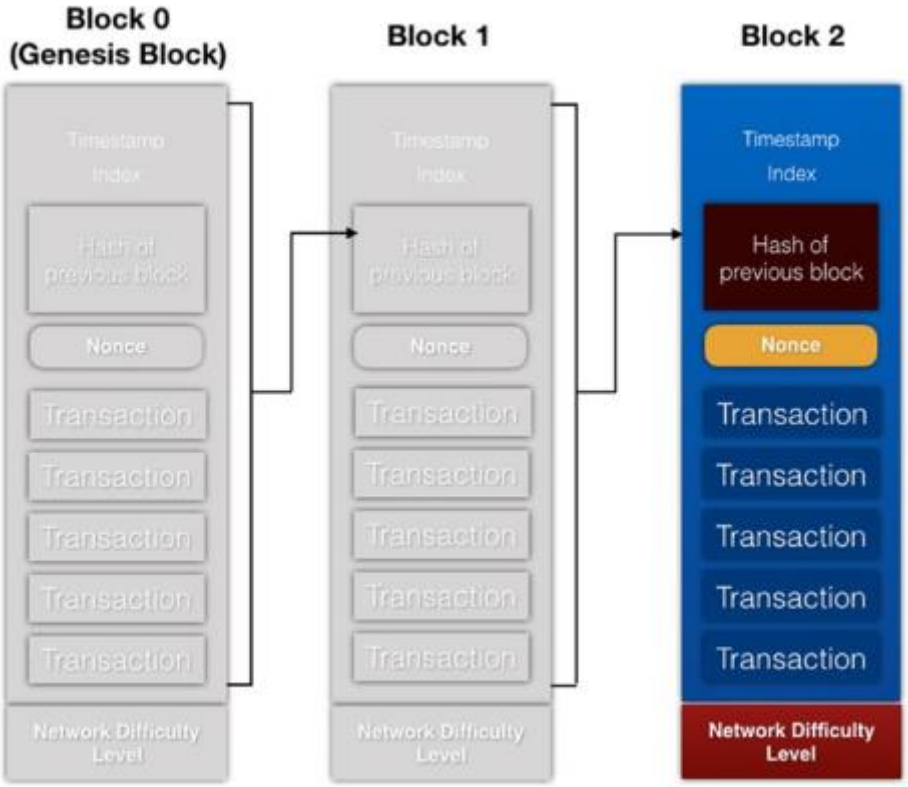
- Nonce: là số dùng gán cho once, once là cơ chế thay đổi nội dung block để đáp ứng Network Difficulty Level. Network Difficulty Level bài tập này là 0000. Vì vậy, kết quả của hash một khối phải bắt đầu bằng 0000
- Transaction: mỗi block lưu trữ một số khác nhau của giao dịch.

2. Thu thập Nonce

Nonce là sự kết hợp index của block, hash của block trước nó, tất cả các giao dịch và kiểm tra kết quả hash có khớp với network difficulty level.



Khi nonce được tìm thấy, nó sẽ được thêm vào block cuối cùng trong blockchain với thời gian được them



3. Cài đặt Flask

Mục tiêu bài thực hành này là triển khai blockchain như là một REST API. Chúng ta cần cài đặt một framework Flask bằng lệnh sau

```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>pip install flask
```

Tiếp tục cài đặt thư viện requests

```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>pip install requests
```

4. Viết mã lệnh tạo blockchain

Tạo tập tin có tên blckchain.py và nhập nội dung sau

4.1. Nạp thư viện

```
1 import sys
2 import hashlib
3 import json
4 from time import time
5 from uuid import uuid4
6
7 from flask import Flask, jsonify, request
8 import requests
9 from urllib.parse import urlparse
```

4.2. Khai báo lớp Blockchain

```
11 class Blockchain(object):
12     difficulty_target = '0000'
13
14     def hash_block(self, block):
15         block_encoded = json.dumps(block, sort_keys=True).encode()
16         return hashlib.sha256(block_encoded).hexdigest()
17
18     def __init__(self):
19         #store all the blocks in the entire blockchain
20         self.chain = []
21
22         #temporarily stores the transactions for the current block
23         self.current_transactions = []
24
25         #create the genesis block with a specific fixed hash
26         # of previous block genesis block starts with index 0
27         genesis_hash = self.hash_block("genesis_block")
28         self.append_block(
29             hash_of_previous_block = genesis_hash,
30             nonce = self.proof_of_work(0, genesis_hash, [])
31             )
```

- Phương thức hash_block() dùng để mã hóa một khối vào một mảng kiểu bytes (dòng 15) và sau đó hash mảng bằng hàm sha256
- Hàm __init__() dùng để lưu toàn bộ blockchain thành list. Mỗi block có genesis block với hash_of_previous_block.

4.3. Hàm tìm Nonce

```
def proof_of_work(self, index, hash_of_previous_block, transactions):
    nonce = 0

    #try hashing the nonce together with the hash of previous
    # block until it is valid
    while self.valid_proof(index, hash_of_previous_block, transactions, nonce) is False:
        nonce += 1
    return nonce
```

Phương thức proof_of_work() dùng để tìm nonce. Bắt đầu với nonce = 0. Sau đó, kiểm tra xem nonce với nội dung block hash có phù hợp với network difficulty hay không. Nếu không, nonce tăng lên 1 cho đến khi tìm được nonce chính xác.

4.4. Hàm hash nội dung một block

```
def valid_proof(self, index, hash_of_previous_block, transactions, nonce):
    # create a string containing the hash of the previous
    # block and the block content, including the nonce
    content = f'{index}{hash_of_previous_block}{transactions}{nonce}'.encode()

    #hash using sha256
    content_hash = hashlib.sha256(content).hexdigest()

    #check if the hash meets the difficulty target
    return content_hash[:len(self.difficulty_target)] == self.difficulty_target
```

Phương thức **valid_proof()** sẽ hash nội dung của một block và kiểm tra hash này có đáp ứng difficulty target không

4.5. Hàm thêm 1 block vào blockchain

```
def append_block(self, nonce, hash_of_previous_block):
    block = {
        'index': len(self.chain),
        'timestamp':time(),
        'transactions': self.current_transactions,
        'nonce': nonce,
        'hash_of_previous_block': hash_of_previous_block
        }
    #reset the current list of transactions
    self.current_transactions = []

    #add the new block to the blockchain
    self.chain.append(block)
    return block
```

Hàm append_block() dùng để thêm 1 block vào blockchain khi nonce được tìm thấy. Ngoài ra, thời gian thêm block cũng được thêm vào.

4.6. Thêm Transactions

```
def add_transaction(self, sender, recipient, amount):
    self.current_transactions.append({
        'amount': amount,
        'recipient': recipient,
        'sender': sender
        })
    return self.last_block['index'] + 1
```

Phương thức add_transaction() dùng để thêm 1 transaction mới vào sau index của transaction cuối cùng trong danh sách transaction của 1 block. Transaction mới thêm sẽ trở thành transaction hiện hành.

4.7. Hàm lấy index của block cuối cùng trong danh sách transaction

```
@property
def last_block(self):
    #return the last block in the blockchain
    return self.chain[-1]
```

5. Xuất Blochain class thành một REST API

Tiếp theo, chúng ta xuất Blockchain thành một REST API đến Flask.

```
app = Flask(__name__)

#generate a globally unique address for this node
node_identifier = str(uuid4()).replace('-','')

#instantiate the blockchain
blockchain = Blockchain()
```

6. Tạo hàm để user có thể lấy blockchain hiện hành

```
#return the entire blockchain
@app.route('/blockchain', methods=['GET'])
def full_chain():
    response = {
        'chain' : blockchain.chain,
        'length':len(blockchain.chain)
    }
    return jsonify(response), 200
```

7. Tạo hàm cho phép user thêm 1 block vào blockchain

```
@app.route('/mine', methods=['GET'])
def mine_block():
    blockchain.add_transaction(
        sender='0',
        recipient=node_identifier,
        amount=1,
    )

    # obtain the hash of last block in the blockchain
    last_block_hash = blockchain.hash_block(blockchain.last_block)
    # using PoW, get the nonce for the new block to be added
    # to the blockchain
    index = len(blockchain.chain)
    nonce = blockchain.proof_of_work(index, last_block_hash,
                                     blockchain.current_transactions)
    # add the new block to the blockchain using the last block
    # hash and the current nonce
    block = blockchain.append_block(nonce, last_block_hash)
    response = {
        'message': "New Block Mined",
        'index': block['index'],
        'hash_of_previous_block': block['hash_of_previous_block'],
        'nonce': block['nonce'],
        'transactions': block['transactions'],
    }
    return jsonify(response), 200
```

8. Thêm 1 giao dịch vào block hiện hành

```
@app.route('/transactions/new', methods=['POST'])
def new_transaction():
    # get the value passed in from the client
    values = request.get_json()
    # check that the required fields are in the POST'ed data
    required_fields = ['sender', 'recipient', 'amount']
    if not all(k in values for k in required_fields):
        return ('Missing fields', 400)
    # create a new transaction
    index = blockchain.add_transaction(
        values['sender'],
        values['recipient'],
        values['amount']
    )
    response = {
        'message': f'Transaction will be added to Block {index}'
        }
    return (jsonify(response), 201)
```

9. Kiểm tra Blockchain đã tạo

Chọn Run → Run Module (F5) để chạy Flask server

```
 * Serving Flask app 'blockchain'
 * Debug mode: off
[31m[1mWARNING: This is a development server. Do not use it in a production de
ployment. Use a production WSGI server instead.[0m
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.82:5000
[33mPress CTRL+C to quit[0m
```

- **Mở một cửa sổ Terminal, và nhập lệnh để xem nội dung blockchain đang chạy trên 1 node**

```
C:\Users\Admin>curl http://localhost:5000/blockchain
```

Index = 0 nghĩa là block gốc (ban đầu)

- **Sinh một khối mới và thêm vào blockchain**

```
C:\Users\Admin>curl http://localhost:5000/mine
```

{"hash_of_previous_block":"bb1c0911e1b84f995719caed6a72f5c075bcae08db92142ad67c7
d554da15339",

"index":1,

"message":"New Block Mined",

"nonce":102076,

"transactions":[{"amount":1,

"recipient":"77038b97f08c46cda8ed8fa3038b0104",

"sender":"0"}

]}

- **Kiểm tra khối mới đã vào blockchain, ta gõ lại**

```
C:\Users\Admin>curl http://localhost:5000/blockchain
```

{"chain":[{"hash_of_previous_block":"181cfa3e85f3c2a7aa9fb74f992d0d061d3e4a6d7461792413aab3f97bd3da95","index":0,"nonce":61093,"timestamp":1660478437.2638743,"transactions":[]},

{**"hash_of_previous_block"**:"bb1c0911e1b84f995719caed6a72f5c075bcae08db92142ad67c7d554da15339",**"index"**:1,**"nonce"**:102076,**"timestamp"**:1660478448.7730875,**"transactions"**:[{**"amount"**:1,**"recipient"**:"77038b97f08c46cda8ed8fa3038b0104",**"sender"**:"0"}]}],
"length":2}

- **Chuẩn bị 1 giao dịch thủ công để thêm vào block tiếp theo**

```
C:\Users\Admin>curl -X POST -H "Content-Type: application/json" -d "{\"sender\": \"04d0988bfa799f7d7ef9ab3de97ef481\",
\"recipient\": \"cd0f75d2367ad456607647edde665d6f\",\"amount\": 5}" "http://localhost:5000/transactions/new"
```

Lệnh trên sẽ gởi nội dung transaction ***sẽ (chuẩn bị)*** thêm vào block tiếp theo (là block 2, index = 2). Kết quả hiện thị

```
C:\Users\Admin>curl -X POST -H "Content-Type: application/json" -d "{\"sender\": \"04d0988bfa799f7d7ef9ab3de97ef481\",
\"recipient\": \"cd0f75d2367ad456607647edde665d6f\",\"amount\": 5}" "http://localhost:5000/transactions/new"
{"message":"Transaction will be added to Block 2"}
```

Thêm nội dung transaction thủ công vào block tiếp theo, ta dùng hàm mine

```
C:\Users\Admin>curl http://localhost:5000/mine
```

Kết quả

{"hash_of_previous_block":"ba037dfd934bf1bfff18cb0be59d489e95aa61317708ceac2919d352e369c8be",

"index":2,

"message":"New Block Mined",

"nonce":89593,

"transactions":[{**"amount":5,"recipient":"cd0f75d2367ad456607647edde665d6f","sender":"04d0988bfa799f7d7ef9ab3de97ef481"**},{"amount":1,"recipient":"77038b97f08c46cda8ed8fa3038b0104","sender":"0"}]}

BÀI TẬP

1. Viết hàm REST API cho biết vị trí block hiện hành trong blockchain ?
2. Viết hàm REST API cho biết vị trí block bất kỳ trong blockchain ?
3. Viết hàm REST API xóa một block bất kỳ (trừ block ban đầu) trong blockchain?
4. Viết hàm REST API cập nhật giá trị amount của một block bất kỳ?
5. Viết hàm REST API để thêm danh sách sinh viên sau:

Charlie	20.5	75.00
Ela	19.6	87.00
Francis	30.8	88.00
Gopal	18.7	65.00
Inba Tamilan	16.3	66.00