DIGITAL ASSIGNMENT
CRYPTOCURRENCY MINING
(DR. POORNIMA NEDUNCHEZHIAN)

CSE1006(BLOCKCHAIN AND CRYPTOCURRENCY TECHNOLOGIES)
A2+TA2

AUGUST 19, 2022

ANIRUDH VADERA
20BCE2940

QUESTION:

Try to implement cryptocurrency mining with any programming language and explain with screenshots. (Simple code is enough)

SOLUTION:

THEORY:

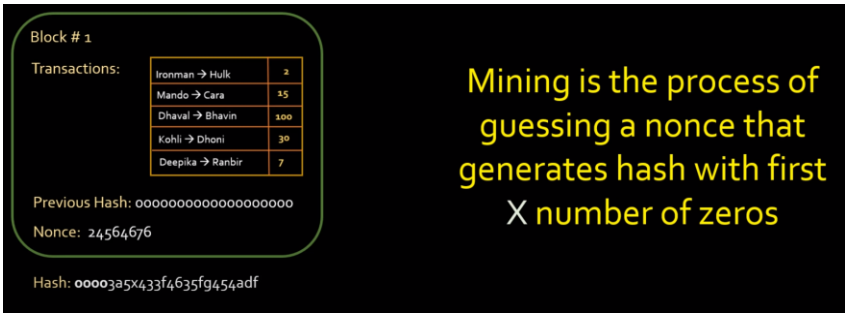
Cryptocurrency Used for demonstration: Bitcoin

A blockchain is a time-stamped decentralized series of fixed records that contains data of any size is controlled by a large network of computers that are scattered around the globe and not owned by a single organization. Every block is secured and connected with each other using hashing technology which protects it from being tampered by an unauthorized person.

Mining is the process by which bitcoins are gradually released to become a part of the circulation. Mining generally refers to solving a computationally tough mathematical puzzle. Bitcoin Mining is the process of adding verified transactions to the chain and the reward gets halved every 210,000 blocks that are mined

Bitcoin follows a protocol that the first 'n' number of digits of the has must be zero. Currently the value of 'n' stands at 20. Our goal is to find the Nonce value so that the hash of the block produces the required number of zeros in the beginning according to the protocol.

The length of the hash is 64 and each digit is hexadecimal, which makes it equal to 4 bits and hence the whole number is actually 256 bits and that is why it is also known as SHA256.



There is drastic change in the time taken by the same code when the difficulty is increased from 4 to 5 and it only keeps on increasing exponentially. Thus that is the main reason why mining one bitcoin takes so much energy and computational power. If that was not enough, you have to be the first one to find the hash or you will not be rewarded. So you are also competing with all the other miners out there and this whole system works on the 'PoW' or 'Proof of Work concept'.

CODE:

```
from hashlib import sha256
import datetime
import time

class Block:
    blockNo = 0
    data = None
    nonce = 0
    next = None
    transactions = None
    previousHash = None
    timeStamp = datetime.datetime.now()

    def __init__(self,data,transactions):
        self.data = data
        self.transactions = transactions

    def SHA256(self,text):
        return sha256(text.encode("ascii")).hexdigest()

    def giveHash(self):
        text = str(self.nonce) + str(self.data) + str(self.previousHash) +
str(self.timeStamp) + str(self.blockNo) + str(self.transactions)
        return self.SHA256(text)

    def printBlock(self):
         print("Block Hash: " + str(self.giveHash()) + "\n" + "BlockNo: " +
str(self.blockNo) + "\n" + "Block Data: " + str(self.data) + "\n" + "Hashes: "
+ str(self.nonce) + "\n" + "Block Transactions: " + str(self.transactions) +
"\n" + "Block Previous Hash: " + str(self.previousHash) + "\n" + "Block Time
Stamp: " + str(self.timeStamp) + "\n" + "Next Block: " + str(self.next) + "\n-
-------------")

class Blockchain:
    difficulty = None
    maxNonce = None

    def __init__(self,difficulty,maxNonce):
        self.difficulty = difficulty
        self.maxNonce = maxNonce

    # Origin of BlockChain

    block = Block("Genesis(First Block(Origin))",'''Satoshi Nakamoto-
>Anirudh:50''')
    dummy = head = block
```

```
    def add(self, block):

        block.previousHash = self.block.giveHash()
        block.blockNo = self.block.blockNo + 1

        self.block.next = block
        self.block = self.block.next

    def mine(self,block):
        prefix_str = '0'*(self.difficulty)
        start = time.time()
        print("\nStarted mining for next Block to be Added")
        for n in range(self.maxNonce):
            blockHash = block.giveHash()
            if blockHash.startswith(prefix_str):
                self.add(block)
                block.printBlock()
                print(f"Yay! Successfully mined bitcoins with nonce
value:{block.nonce}")
                total_time = str((time.time() - start))
                print(f"End mining. Mining took: {total_time} seconds")
                return 1
            else:
                block.nonce += 1

        raise BaseException(f"Couldn't find correct hash after trying
{MAX_NONCE} times")


if __name__=='__main__':
    difficulty = int(input("Set difficulty (Number of prefix zeros) : ")) #
changing this to higher number and you will see it will take more time for
mining as difficulty increases
    MAX_NONCE = 2**32
    totalBlocks = int(input("Enter the Number of Blocks : "))

    blockchain = Blockchain(difficulty,MAX_NONCE)

    print("\nMining New Blocks: ")

    for n in range(totalBlocks):
        transactions = input("Enter Transactions for Block "+str(n+1) + " :
\n")
        output = blockchain.mine(Block("Block " + str(n+1),transactions))

    print("\nThe BlockChain Currently is: ")
```

```
    while blockchain.head != None:
        blockchain.head.printBlock()
        blockchain.head = blockchain.head.next
```

EXPLANATION:

We first import the required libraries

Then we try to stimulate a Block Chain with the following structure:

Structure of Block Chain:

Block1->Block2->Block3->None

Structure of a Single Block:

```
5   class Block:
6       blockNo = 0
7       data = None
8       nonce = 0
9       next = None
10      transactions = None
11      previousHash = None
12      timeStamp = datetime.datetime.now()
```

Block = {

blockNo = Integer

data = String

nonce = Integer

next = Memory Location

transactions = String

previousHash = HexString

timeStamp = Date

}

The fingerprinting is done by using hash and to be particular we will use the SHA256 hashing algorithm. Every block will contain its own hash and also the hash of the previous function so that it cannot get tampered with.

This fingerprinting will be used to chain the blocks together. Every block will be attached to the previous block having its hash and to the next block by giving its hash.

The giveHash() function along with SHA256() gives the value of hash of the current block by encrypting all the information of the block into string(Ascii) and then converting that string using sha256() into the required hash.

```
    def SHA256(self,text):
        return sha256(text.encode("ascii")).hexdigest()

    def giveHash(self):
        text = str(self.nonce) + str(self.data) + str(self.previousHash) + str(self.timeStamp) + str(self.blockNo) + str(self.transactions)
        return self.SHA256(text)
```

Then we have a class of BlockChain which contains major information about our current BlockChain

Our BlockChain has a block Pointer which stores the information about the last mined block

```
class Blockchain:
    difficulty = None
    maxNonce = None

    def __init__(self,difficulty,maxNonce):
        self.difficulty = difficulty
        self.maxNonce = maxNonce

    # Origin of BlockChain

    block = Block("Genesis(First Block(Origin))",'''Satoshi Nakamoto->Anirudh:50''')
    dummy = head = block
```

We also initialized our BlockChain with creating first ever block named Genesis having transaction as stated above.

Our Mining Function Looks Like this:

We change the nonce value of the block until we get a hash with required number of prefix zeros that we require. If we are not able to mine it we raise an exception

```
def mine(self,block):
    prefix_str = '0'*(self.difficulty)
    start = time.time()
    print("\nStarted mining for next Block to be Added")
    for n in range(self.maxNonce):
        blockHash = block.giveHash()
        if blockHash.startswith(prefix_str):
            self.add(block)
            block.printBlock()
            print(f"Yay! Successfully mined bitcoins with nonce value:{block.nonce}")
            total_time = str((time.time() - start))
            print(f"End mining. Mining took: {total_time} seconds")
            return 1
        else:
            block.nonce += 1

    raise BaseException(f"Couldn't find correct hash after trying {MAX_NONCE} times")
```

After Mining the Block Successfully, we add the mined block to the blockchain

```
def add(self, block):

    block.previousHash = self.block.giveHash()
    block.blockNo = self.block.blockNo + 1

    self.block.next = block
    self.block = self.block.next
```

OUTPUT:

Taking Inputs:

```
In [2]: runfile('C:/Users/Anirudh/OneDrive/Desktop/python/blockChainMining.py',

Set difficulty (Number of prefix zeros) : 4

Enter the Number of Blocks : 3
```

Mining The First Block:

```
Mining New Blocks:

Enter Transactions for Block 1 :
Anirudh->Om:10; Om->Animesh:5

Started mining for next Block to be Added
Block Hash: 85dd28e516470a47d33ee6900259ca16c7f66436aa46f099b4b1b662d4eae806
BlockNo: 1
Block Data: Block 1
Hashes: 3409
Block Transactions: Anirudh->Om:10; Om->Animesh:5
Block Previous Hash: 9357739e7c87800447da58aeca1ebdae2689918c6962cc83500e38ee24259f94
Block Time Stamp: 2022-08-19 16:26:52.346316
Next Block: None
--------------
Yay! Successfully mined bitcoins with nonce value:3409
End mining. Mining took: 0.018006086349487305 seconds
```

Mining The Second Block:

```
Enter Transactions for Block 2 :
Animesh->Anirudh:3; Anirudh->Harshil:10

Started mining for next Block to be Added
Block Hash: 9a1fc23945694bf329aeadb45c2f158ac080b05d0461bbabd194cf55b88e5c2c
BlockNo: 2
Block Data: Block 2
Hashes: 31075
Block Transactions: Animesh->Anirudh:3; Anirudh->Harshil:10
Block Previous Hash: 85dd28e516470a47d33ee6900259ca16c7f66436aa46f099b4b1b662d4eae806
Block Time Stamp: 2022-08-19 16:26:52.346316
Next Block: None
--------------
Yay! Successfully mined bitcoins with nonce value:31075
End mining. Mining took: 0.1470470428466797 seconds
```

Mining The Third Block:

```
Enter Transactions for Block 3 :
Harshil->Anirudh:10

Started mining for next Block to be Added
Block Hash: be4f609f2d038db3f49989956353644b5e2208e870b8359a9c9631f3044047c1
BlockNo: 3
Block Data: Block 3
Hashes: 81477
Block Transactions: Harshil->Anirudh:10
Block Previous Hash: 9a1fc23945694bf329aeadb45c2f158ac080b05d0461bbabd194cf55b88e5c2c
Block Time Stamp: 2022-08-19 16:26:52.346316
Next Block: None
-------------
Yay! Successfully mined bitcoins with nonce value:81477
End mining. Mining took: 0.38710618019104004 seconds
```

The BlockChain Currently is:

```
The BlockChain Currently is:
Block Hash: 9357739e7c87800447da58aeca1ebdae2689918c6962cc83500e38ee24259f94
BlockNo: 0
Block Data: Genesis(First Block(Origin))
Hashes: 0
Block Transactions: Satoshi Nakamoto->Anirudh:50
Block Previous Hash: None
Block Time Stamp: 2022-08-19 16:26:52.346316
Next Block: <__main__.Block object at 0x00000245455C0790>
-------------
Block Hash: 85dd28e516470a47d33ee6900259ca16c7f66436aa46f099b4b1b662d4eae806
BlockNo: 1
Block Data: Block 1
Hashes: 3409
Block Transactions: Anirudh->Om:10; Om->Animesh:5
Block Previous Hash: 9357739e7c87800447da58aeca1ebdae2689918c6962cc83500e38ee24259f94
Block Time Stamp: 2022-08-19 16:26:52.346316
Next Block: <__main__.Block object at 0x00000245455C0A00>
-------------
Block Hash: 9a1fc23945694bf329aeadb45c2f158ac080b05d0461bbabd194cf55b88e5c2c
BlockNo: 2
Block Data: Block 2
Hashes: 31075
Block Transactions: Animesh->Anirudh:3; Anirudh->Harshil:10
Block Previous Hash: 85dd28e516470a47d33ee6900259ca16c7f66436aa46f099b4b1b662d4eae806
Block Time Stamp: 2022-08-19 16:26:52.346316
Next Block: <__main__.Block object at 0x00000245455C08B0>
-------------
Block Hash: be4f609f2d038db3f49989956353644b5e2208e870b8359a9c9631f3044047c1
BlockNo: 3
Block Data: Block 3
Hashes: 81477
Block Transactions: Harshil->Anirudh:10
Block Previous Hash: 9a1fc23945694bf329aeadb45c2f158ac080b05d0461bbabd194cf55b88e5c2c
Block Time Stamp: 2022-08-19 16:26:52.346316
Next Block: None
-------------
```