Hash function:

當使用方法去分類資料去存在不同位置的陣列時，如果出現兩個相同的結果要存在同陣列元素中，必須將某值做一定運算轉換再存或者改變Hash的方法

一種解法是直接用Link

Mediantime:挖礦花的時間

### 學生身分註冊:可以獲得私有的空間的權限:

### github education ->[GitHub Education: Engaged students are the result of using real-world](https://education.github.com/) ->Github Student Developer Pack ->傳上學生証圖

### 

### (下周一考試)

### GitHub操作:

### 首頁->左下New repository ->命名專案 -> public ->Initialize打勾 ->創建

### Clone or download -> Open in Desktop ->安裝

### ->sign into GitHub.com ->Continue ->Finish

### 辨識用的:

### [42425342+a104222014@users.noreply.github.com](mailto:42425342+a104222014@users.noreply.github.com)

### 中間是把電腦專案放上去

### 右邊是把網路上的專案抓下來 ->選專案->Clone開始載專案

### 下載路徑C:\Users\w401\Documents\GitHub\專案名

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-3-51.JPG

### 在那個路徑新增檔案後 回到gitHub軟體後發現檔案出現了

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-4-15.JPG

### 左邊代表更動過的檔案內容(未上傳)

### 點左邊的+號 輸入Summary命名(更動的內容)

### 按Commit to master確認後，

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-6-10.JPG

### 然後按下上面的push origin就能上傳

### 回到gitHub網頁會發現那專案已經加了檔案了

### 網頁 專案 Settings Collaborators 新增共同使用者

### 那個人就可以變更這個專案內容

### 新建一般的專案時

### 

### 將已開啟的別人的專案設定這個

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-43-8.JPG

### 將內容設定成一樣

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-43-15.JPG

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-43-44.JPG

### 做些小修改後Commit和Push上去就會傳送到那個自己新建的專案

### (卡住時選Repository clean pull and push)

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-43-57.JPG

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 10-44-11.JPG

### 瀏覽器->更多工具->開發人員工具 ->Console ->JS語言

### 

### JS語言編程:

### <https://www.typescriptlang.org/docs/handbook/generics.html>

### 安裝npm和NodeJS:

### <https://nodejs.org/en/>

### CMD -> 輸入node 有>代表安裝成功

### Cril+C 按兩次跳出

### mkdir test

### cd test

### npm init (創建一個nodejs專案名稱)

### package name預設為資料夾名稱

### git repository:輸入git那個網址

### <https://github.com/a104222014/nodejs01.git>

### 建完後從下面網址開始作流程:

### https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

> npm install -g typescript

### 

tsc greeter.ts

### ts語言轉成js

### ts有做到資料格式檢查

### React + Redux + TypeScript

### Docker運作:

### 安裝docker tool

### 執行 Docker Quickstart Terminal

### $docker run –it gwrx/bitcoin-test /bin/bash

### /#

### /# ls

### /# cd /root

### /# ls

### /# cd bitcoin/

### /# ls

### 

### /# bitcoind –regtest –daemon

### (開啟bitcoin Server)

### /# ls

### /# nodejs one

### (產生一個one.js檔案並執行)

### 

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 14-33-36.JPG

### 產生私鑰方法

### ~/ bitcoin-cli –regtest getnewaddress

### 1.取得新的位置

### ~/ bitcoin-cli –regtest dumpprivkey 地址

### 2.產生私鑰

### 

### <https://github.com/gwrxuk/bitcoin-core-test/blob/master/one.js>

### //引入bitcoin模組

### const bitcoin = require("bitcoinjs-lib");

### //選用regtest網路

### const regtest = bitcoin.networks.testnet

### //https://github.com/bitcoinjs/bip65

### const bip65 = require('bip65')

### //輸入私鑰，教學用，請勿隨意公開私鑰

### var privateKey = "cNcvQphXWjAJ365Y8Tuhti5a6fCh9ftpR3GScZs5vdidBwQqpxT4";

### //產生公鑰跟私鑰

### const keyPair = bitcoin.ECPair.fromWIF(privateKey,regtest);

### //產生付款位址

### const { address } = bitcoin.payments.p2pkh({ pubkey: keyPair.publicKey });

### console.log(bitcoin.payments.p2pkh({ pubkey: keyPair.publicKey }));

### const txb = new bitcoin.TransactionBuilder(regtest);

### //用listunspent 取出最後一筆資料的txid

### txb.addInput('c3acac43c6f0e83538fb5c82ed01379349f5dccf43a8af7bb5d8f0c60fb1fcfc', 0);

### //用getnewaddress取得新的位址

### txb.addOutput('2MxfNS8sB1GcCo3GBLmX2PUw277H9zvrn2L', 12)

### //交易簽名

### txb.sign(0, keyPair); //第一個位置的是上一筆交易中的第一個支出，第二個欄位是我們的公鑰與私鑰

### //取得交易序號

### const transaction\_01 = txb.build().toHex();

### console.log(transaction\_01);

### # bitcoin-cli –regtest generate 111

### # bitcoin-cli –regtest getnewaddress

### # bitcoin-cli –regtest sendtoaddress 地址 10

### # bitcoin-cli –regtest listunspent 0 1

### (取得交易紀錄)

### 交易紀錄的的碼

### 下載gitHub專案:

### # cd ..

### # mkdir test

### # cd test

### # git clone (github專案位址)

### 問題筆記:

### (要先開啟Server)

### $ npm install --save bitcoinjs-lib

### $ rm –rf 資料夾名

### 編輯 github 上的testClass/test01.js

### //引入bitcoin模組

### const bitcoin = require("bitcoinjs-lib");

### //選用regtest網路

### const regtest = bitcoin.networks.testnet

### //https://github.com/bitcoinjs/bip65

### const bip65 = require('bip65')

### //輸入私鑰，教學用，請勿隨意公開私鑰

### var privateKey = "cNcvQphXWjAJ365Y8Tuhti5a6fCh9ftpR3GScZs5vdidBwQqpxT4"; //改成自己交易時用的地址產生的私鑰

### //產生公鑰跟私鑰

### const keyPair = bitcoin.ECPair.fromWIF(privateKey,regtest);

### //產生付款位址

### const { address } = bitcoin.payments.p2pkh({ pubkey: keyPair.publicKey });

### console.log(bitcoin.payments.p2pkh({ pubkey: keyPair.publicKey }));

### const txb = new bitcoin.TransactionBuilder(regtest);

### //用listunspent 取出最後一筆資料的txid

### txb.addInput('d8c711964efb92730437e70764d32d8bddc8e9d852c0bf9304f12d0202eab3f4', 1); //改成自己交易時餘額的txID

### //用getnewaddress取得新的位址

### txb.addOutput('2MxfNS8sB1GcCo3GBLmX2PUw277H9zvrn2L', 12)

### //交易簽名

### txb.sign(0, keyPair); //第一個位置的是上一筆交易中的第一個支出，第二個欄位是我們的公鑰與私鑰

### //取得交易序號

### const transaction\_01 = txb.build().toHex();

### console.log(transaction\_01);

### $ git clone <https://github.com/a104222014/testClass.git>

### # cd testClass

### # nodejs test01.js

### 

### <https://github.com/bitcoinjs/bitcoinjs-lib>

### 

### 隨機產生address 測試的指令

### 

### 公鑰轉成16進位碼

### 三把鑰匙中有2把鑰匙能解開就能解開

### <https://github.com/bitcoinjs/bitcoinjs-lib/blob/master/test/integration/addresses.js#L104>

### 多重簽名: <https://github.com/bitcoinjs/bitcoinjs-lib/blob/master/test/integration/transactions.js#L105>

### p2ms的方法

### txb.sign (keyPair[0] [2] 第一把跟第三把鑰匙)->只要兩人簽名就能拿走錢

* [Create (and broadcast via 3PBP) a Transaction with a 2-of-4 P2SH(multisig) input](https://github.com/bitcoinjs/bitcoinjs-lib/blob/master/test/integration/transactions.js#L105)

### 取得交易內容

### 彼特幣:交易紀錄每個人分清楚的概念(隱私性、交易強)

### 以泰紡:交易紀錄單位是以家庭(一群)為單位 以合約為主要的機制(契約為主)

### 測試的程式碼(可忽略)

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-11-35.JPG

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-11-48.JPG

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-12-48.JPG

### 第四行可改成

### const regtest = bitcoin.networks.testnet

### 以太坊ethereum:

### 白皮書:目標 介紹與說明類的說明-去中心化的平台

### 黃皮書:技術導向規範

### 以太坊的虛擬機器來運作

### 智慧合約: Remix Solidity程式語言

### 主要目的是做程式

### 可以自己產生自己的貨幣

### (直接把總額算出來去計算)

### Ghost protocol greedy:

### GHOST(Greedy Heaviest Observed Subtree) protocol

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-20-51.JPG

### Main : 連接長度最長(彼特幣計算的認為最可信的)

### Ghost:節點數連接的分支最多(以太坊以認為這個為可信的)

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-25-23.JPG

### 每種運算所要求的手續費不同

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-26-6.JPG

### 使用EVM(一種虛擬機器)執行程式，可以對符號操作

### 可以做到forloop 彼特幣只能做stack所以不能forloop

### 最小貨幣單位:wei

### 以ECDSA產生簽名

### 以上一筆交易紀錄算位置

### 為了應付彼特幣的挖礦機，所以這裡為了應付那個加了很多演算法

### **ASIC-resistance**

### 以太訪的交易比比特幣快很多

### Address:每次的地址一樣不用每次交易都用不同address

### Aes-128加密

### Gas Price 交易手續費

### Gas Limit 交易上限

### Eth.

### 如果單次交易手續費不夠交易會失敗，但手續費會吃掉不會回收

### 如果儲存空間被清空，則多的錢會退還給你

### 交易上限:GAS Price \* GAS limit / 10的9次方 以太幣

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-40-40.JPG

### 以太坊網站: <https://www.ethereum.org/>

### 下載以太坊錢包然後解壓縮

### 執行Ethereum Wallet.exe

### 下載Plugin

### Google瀏覽器 ->應用程式->線上商店-> MetaMask

### 點下狐狸頭

### Try it

### Cont….

### 輸入密碼

### 複製左邊的0xae66d6E6d2633e7d610d56be1E0756481F0a147E

### <http://faucet.ropsten.be:3001/>

### 輸入碼(貼上) ->拿到一塊錢(測試錢)

### ERC20智慧條約建立貨幣

### uint8/16/24 建立一個不算正負號8/16/24位元的整數變數

### event 觸發哪些交易行為

### mapping (address => mapping(address => uint256))

### 某地址給某人多少錢

### C:\Certiport\Console\Exams\Microsoft Office Specialist\文件\智慧教學錄影廣播系統\Snapshots\2018-8-16 16-59-20.JPG

### 下周下午考比特幣如何取得貨幣、

### 如何交易取得位置等等。