

Department of Computing Curtin University

DATA STRUCTURES AND ALGORITHMS

Assignment Report [CryptoGraph]

Semester 2, 2020

Name: Seenarain Bhavyattaa
ID: 19871253

Table of Contents

Abstract	2
Background.....	2
Methodology	2
Sample	3
Results.....	4
Conclusion	10

Abstract

This report outlines how to analyse and find a way in developing crypto-currency trading data through the program CryptoGraph. Cryptocurrency is a medium of exchange and it uses a decentralised technology such as blockchain which make use of encryption techniques to verify transfer of funds. That is, it enables users to proceed with a secure payment and even store money. This is remarkably effective in some way as it is not necessary for the user to go the bank. Hence, this a brief introduction of cryptocurrency how it typically works. However, in this program we will make use of a trade data, to provide connections between different currencies.

Background

One of the most popular technology known as Blockchain has not only elevate security of trade transactions but also its reliability. Today, the cryptocurrency Blockchain has potentially improve business processes. It increases the confidence of traders as the transparency of information and security of blockchain transaction are truly effective. When it comes to transparency, users can view every single transaction and decided upon their observation. Furthermore, its security methodologies namely mnemonics help to protect crypto wallet. Hence, this is especially useful as it is exceedingly difficult for hackers to infiltrate.

In this CryptoGraph program, two modes are generated. The first one is report mode whereby it extracts values from both the assetFile.json and TradeInfo.json and hence save it to graph. Moreover, the second mode is the interactive mode where eleven options will be provided by the user to execute the interactive mode. Below those options will be given in more details.

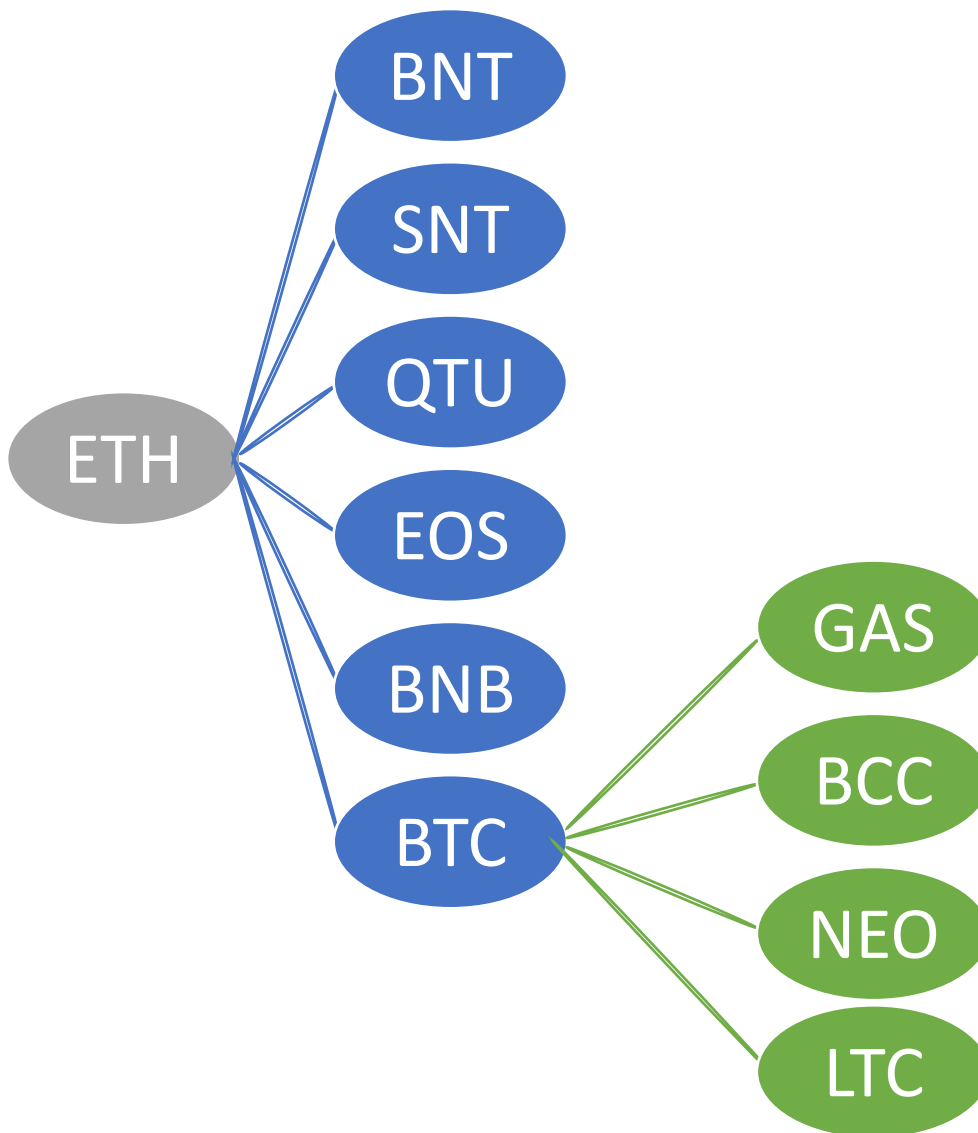
Methodology

When executing the report mode, the extracted values from the json file are save in a graph.

The following diagram is a sample of the graph made up of the base asset and quote asset.

Sample

ETH	BTC
LTC	BTC
BNB	BTC
NEO	BTC
QTUM	ETH
EOS	ETH
BNT	ETH
BNT	ETH
BCC	BTC
GAS	BTC
BNB	ETH



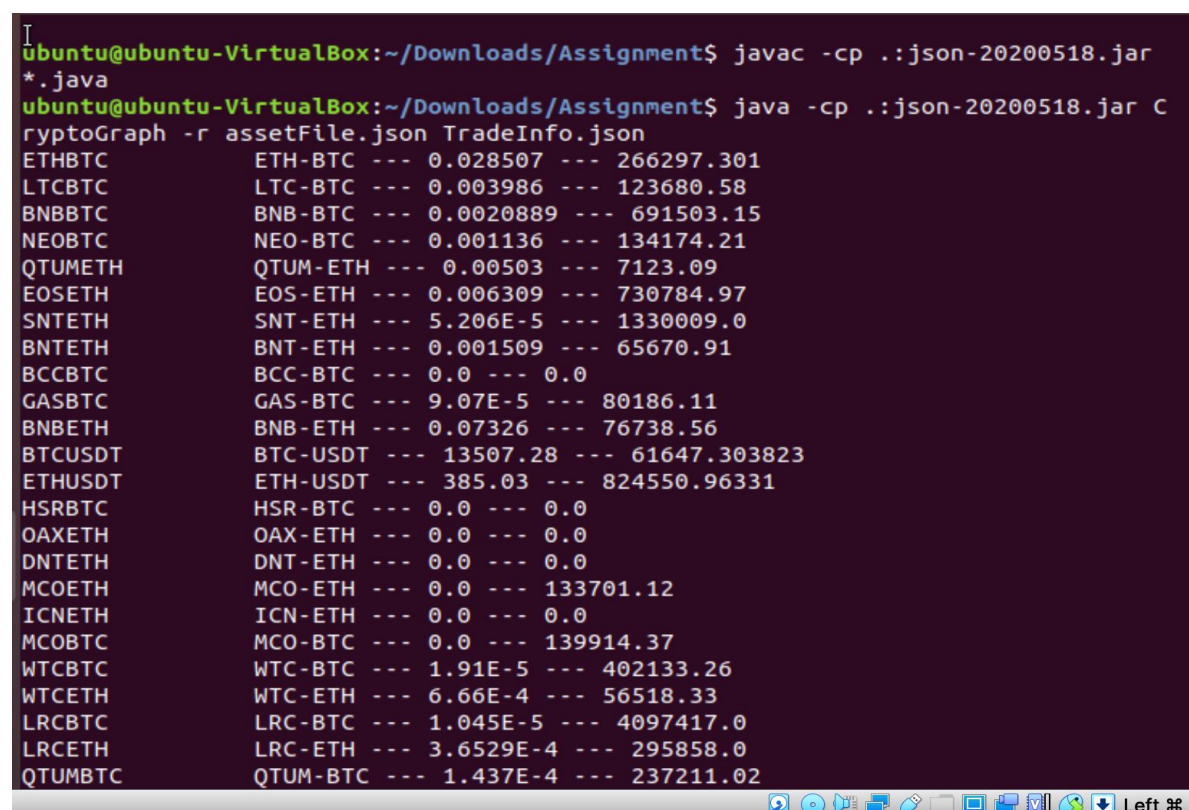
To compile the CryptoGraph program the following command needs to use:

1. For execution : `javac -cp .:json-20200518.jar *.java`
2. For usage information : `java -cp .:json-20200518.jar CryptoGraph`
3. To enter in report mode : `java -cp .:json-20200518.jar CryptoGraph assetFile.json TradeInfo.json`
4. To enter in interactive mode : `java -cp .:json-20200518.jar CryptoGraph -i`

Results

First mode - report mode

Report mode - extract values from the assetFile.json and TradeInfo.json and print them out. The following diagram will show the result of the report mode.



```

ubuntu@ubuntu-VirtualBox:~/Downloads/Assignment$ javac -cp .:json-20200518.jar *.java
ubuntu@ubuntu-VirtualBox:~/Downloads/Assignment$ java -cp .:json-20200518.jar CryptoGraph -r assetFile.json TradeInfo.json
ETHBTC      ETH-BTC --- 0.028507 --- 266297.301
LTCBTC      LTC-BTC --- 0.003986 --- 123680.58
BNBBTC      BNB-BTC --- 0.0020889 --- 691503.15
NEOBTC      NEO-BTC --- 0.001136 --- 134174.21
QTUMETH     QTUM-ETH --- 0.00503 --- 7123.09
EOSETH      EOS-ETH --- 0.006309 --- 730784.97
SNTETH      SNT-ETH --- 5.206E-5 --- 1330009.0
BNTETH      BNT-ETH --- 0.001509 --- 65670.91
BCCBTC      BCC-BTC --- 0.0 --- 0.0
GASBTC      GAS-BTC --- 9.07E-5 --- 80186.11
BNBETH      BNB-ETH --- 0.07326 --- 76738.56
BTCUSDT     BTC-USDT --- 13507.28 --- 61647.303823
ETHUSDT     ETH-USDT --- 385.03 --- 824550.96331
HSRBTC      HSR-BTC --- 0.0 --- 0.0
OAXETH      OAX-ETH --- 0.0 --- 0.0
DNTETH      DNT-ETH --- 0.0 --- 0.0
MCOETH      MCO-ETH --- 0.0 --- 133701.12
ICNETH      ICN-ETH --- 0.0 --- 0.0
MCOBTC      MCO-BTC --- 0.0 --- 139914.37
WTCBTC      WTC-BTC --- 1.91E-5 --- 402133.26
WTCETH      WTC-ETH --- 6.66E-4 --- 56518.33
LRCBTC      LRC-BTC --- 1.045E-5 --- 4097417.0
LRCETH      LRC-ETH --- 3.6529E-4 --- 295858.0
QTUMBTC     QTUM-BTC --- 1.437E-4 --- 237211.02
  
```

Second mode - Interactive mode

Interactive mode – displays all the eleven options, that a user can opt for. Below you will find the option's illustration.

```
ubuntu@ubuntu-VirtualBox:~/Downloads/Assignment$ java -cp .:json-20200518.jar C
ryptoGraph -i
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset overview
10. Trade overview
11. Exit
Enter a choice : █
```

Option 1: Load asset data.

Load asset data - will make the user aware that the asset file has successfully loaded. If not an error message "asset not loaded" will be display.

So, first the user will have to enter the assetFile.json as the filename.

```
ubuntu@ubuntu-VirtualBox:~/Downloads/Assignment$ java -cp .:json-20200518.jar C
ryptoGraph -i
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset overview
10. Trade overview
11. Exit
Enter a choice : 1
Enter file name : assetFile.json
Asset file loaded successfully
```

Option 2: Load trade data.

Load trade data - indicates the user that the TradeInfo.json file has successfully loaded. An error message will be displayed if wrong trade file has been inserted.

Again, here also, the correct TradeInfo.json should be entered as the filename

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset overview
10. Trade overview
11. Exit
Enter a choice : 2
Enter file name : TradeInfo.json
Trade details file loaded successfully
```

Option 3: Display Graph.

In this option a linked list is used to extract values from the trade info file. Some example of extracted values are the symbols, baseAsset and the quoteAsset from the json object.

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset overview
10. Trade overview
11. Exit
Enter a choice : 3
ETHBTC      ETH-BTC --- 0.028507 --- 266297.301
LTCBTC      LTC-BTC --- 0.003986 --- 123680.58
BNBBTC      BNB-BTC --- 0.0020889 --- 691503.15
NEOBTC      NEO-BTC --- 0.001136 --- 134174.21
QTUMETH     QTUM-ETH --- 0.00503 --- 7123.09
EOSETH      EOS-ETH --- 0.006309 --- 730784.97
SNTETH      SNT-ETH --- 5.206E-5 --- 1330009.0
BNTETH      BNT-ETH --- 0.001509 --- 65670.91
BCCBTC      BCC-BTC --- 0.0 --- 0.0
GASBTC      GAS-BTC --- 9.07E-5 --- 80186.11
BNBETH      BNB-ETH --- 0.07326 --- 76738.56
BTCUSDT     BTC-USDT --- 13507.28 --- 61647.303823
ETHUSDT     ETH-USDT --- 385.03 --- 824550.96331
HSRBTC      HSR-BTC --- 0.0 --- 0.0
OAXETH      OAX-ETH --- 0.0 --- 0.0
```

Option 4: Load serialized data.

Load serialized data - reads object from file serializedObjectFile.txt. A message "Object read from file" is printed if "option 4" works accordingly.

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset otherview
10. Trade overview
11. Exit
Enter a choice : 4
Object read from file
```

Option 5: Save serialized data.

Save serialized data - save object from file serializedObjectFile.txt. An error message is displayed if "option 5" does not work accordingly.

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset overview
10. Trade overview
11. Exit
Enter a choice : 5
```


Option 6: Display Asset.

Display Asset - asks the user to insert an asset symbol such as ETHBTC. Then the program will call the function findAsset() where the latter will search for the asset symbols and display true if found.

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset overview
10. Trade overview
11. Exit
Enter a choice : 6
Enter an asset symbol : ETHBTC
Symbol : true
```

Option 7: Display trade details.

Display trade details – the user will be asked to enter a trade detail symbol. BCCBTC is one example of trade symbol. After that the program will call a function findTradeDetails() to search and display the trade details if exist. An example is BCCTCC BCC-BTC ---0.0---0.0.

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset overview
10. Trade overview
11. Exit
Enter a choice : 7
Enter a trade symbol : BCCBTC
BCCBTC          BCC-BTC --- 0.0 --- 0.0
```

Option 8: Display potential trade path

This option has not implemented. But one needs to display direct and indirect path.

Option 9: Asset overview.

Asset overview – a method overviewAsset is called to provide the tradeSymbol, status, baseAsset and quoteAsset when this option is chosen.

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset otherview
10. Trade overview
11. Exit
Enter a choice : 9
Enter an asset to find : ETHBTC
Symbol: ETHBTC -- Status: TRADING
Base asset: ETH -- Quote Asset: BTC
```

Option 10: Trade overview.

Trade overview – it uses overviewAsset method to extract values from the trade detail file. Values such as the bidPrice , volumn and count are provided when choosing for this option.

```
1. Asset Data
2. Trade Data
3. Display Graph
4. Load Serialised data
5. Save Serialized data
6. Display Asset
7. Display trade details
8. Display potential trade paths
9. Asset otherview
10. Trade overview
11. Exit
Enter a choice : 10
Enter a trade detail to find : BCCBTC
Symbol: BCCBTC -- Bid Price: 0.0
Volume: 0.0 -- Count: 0
```

Option 11: Exit.

Exit - Closing the program directly.

Conclusion

“CryptoGraph” program can demonstrate:

1. The report mode, one should insert the appropriate json file that is the assetFile.json and TradeInfo.json to be able to extract values from both asset and trade file. Analysing the report mode accordingly will help to display both the asset and trade details.
2. The asset symbols, if required asset symbols are not used in the program, a Boolean status of false will be indicated that the asset symbol does not exist according to assetFile.json. Else, a boolean status of true will be allocated if the symbol exists.
3. The trade symbols, if correct trade symbols are entered in the program, the trade detail of this specific trade symbol will be shown. Else,

CryptoGraph program make use of json package to extract values from the <https://www.binance/api/v3/exchangeInfo>. These values are to implement a graph representation. Furthermore, most of the options in the interactive mode are implemented but “option 8 “that is display the trade path has not. Hence, the program could not display the direct and indirect path.