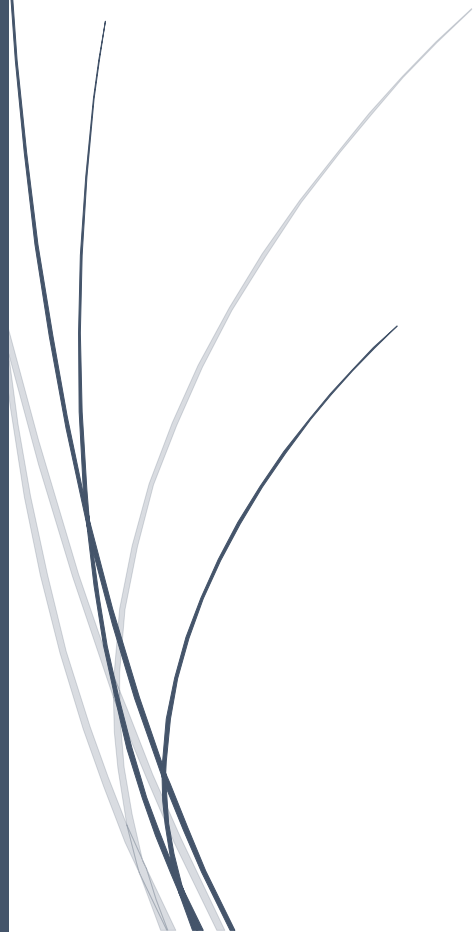
10/10/2020

DSA Assignment 1

Project Report

Student Name: Timo Reichelt

Student ID: 19663858



Information for Use

Introduction

The main program cryptoGraph.py analyses data from the cryptocurrency trading platform www.binance.com. The program can be run in interactive mode, or in report mode. The report mode will print a list of all available currencies on wwww.binance.com, as well as the exchange rate of all possible trading options for each currency. After that it will print lists of the top ten trading pairs with the highest volume; the highest price; the highest price change in percent; the lowest spread; and the highest spread between ask price and bid price. Ultimately, it will display a pie chart that shows the three trading pairs with the highest volume. In the interactive mode, users can load the from www.binance.com provided trade and asset data, or a serialised file. Users are able to create a graph by utilising the trade and asset data, or by loading a previously serialised graph. Then they are able to search for a particular currency (main menu option 2), or trading pair (main menu option 3). The particular currency or trading pair will be printed together with additional information. Users are able to search for all available trading paths between two particular currencies (main menu option 4). Users have the option to display trading paths only, or to display trading paths including the exchange rate and the cheapest possible trading path. In the interactive mode, users are able to delete currencies, or to delete half of all available trading pairs which have a high spread between ask price and bid price (main menu option 5). As the ask price is the lowest price a seller would trade a particular currency for another currency and the bid price is the highest price a buyer would pay for the particular currency, the removal of trading pairs with a high spread eliminates expensive trading paths and reduces the time to identify the cheapest trading path. Moreover, users are able to display a list of all available currencies including their exchange rate for possible trading options (main menu option 6), or the pie chart of the top 3 currencies with the highest volume and the various top 10 lists such as in the report mode (main menu option 7). Moreover, in the interactive mode, users are able to save their modified graph in form of a serialised file (main menu option 8).

Installation

Requirements:

- Asset data
  Link: https://www.binance.com/api/v3/exchangeInfo

- Trade data
  Link: https://www.binance.com/api/v3/ticker/24hr

Dependencies:

- Python 3.7 or later

Required to run the program
Link: https://www.python.org/downloads/

- Json
  Required to read asset and trade data
  Link: http://json.org/

- Pandas
  Required to plot data
  Link: https://pandas.pydata.org/getting_started.html

- Matplotlib.pyplot
  Required to plot data
  Link: https://matplotlib.org/3.1.1/users/installing.html

- Pickle
  Required to save and load serialised files
  Link: https://docs.python.org/3.0/library/pickle.html

- Sys
  Required to increase max recursion limit in order to avoid errors with serialising
  Link: https://docs.python.org/3/library/sys.html

- Classes
  Contains required classes and functions for the main program cryptoGraph.py

Description of files:

- README.txt – Readme file for assignment 1

- cryptoGraph.py – Main program

- classes.py – Contains classes and functions for main program cryptoGraph.py

- unitTestGraph.py – Test harness for class: Graph, and related classes

- unitTestLinkedList.py – Test harness for class: LinkedList, and related classes

- assetData.json – Asset data from https://www.binance.com/api/v3/exchangeInfo

- tradeData.json – Trade data from https://www.binance.com/api/v3/ticker/24hr

- testTradeData.json – Minimised trade data for unitTestGraph.py

Terminology / Abbreviations

Base Asset = The first listed currency in a trading pair.

Quote Asset = The second listed currency in a trading pair.

Ask Price = Lowest price the seller of a base asset would accept in exchange for quote asset.

Ask Qty = The total amount of base assets that are for sale.

Bid Price = Highest price the potential buyer of a base asset would pay in exchange for the quote asset.

Bid Qty = The total amount of quote asset offered in exchange for the base asset.

Highest Price = The highest price the base currency was sold for in the last 24h.

Lowest Price = The lowest price the base currency was bought for in the last 24h.

Volume = The total number of trades.

Open Price = The price of the base currency in quote currency 24h ago.

Last Price = The last price the base currency has been sold for in exchange for the quote currency.

Price Change = The amount and percentage of the total price change between the open price and last price.

Spread = The spread between the ask price and the bid price in percent.

Walkthrough

cryptoGraph.py

The program cryptoGraph.py can be started by typing “python3 cryptoGraph.py” in the terminal window of the directory “../DSA Assignment”.

The following message will appear:

“Usage:

Interactive Mode:	python3 cryptoGraph.py -i
Report Mode:	python3 cryptoGraph.py -r <asset_filename> <trade_filename>”

When the program is started in report mode by typing “python3 cryptoGraph.py -r” in the terminal window, a list of all available currencies on wwww.binance.com, as well as the exchange rate of all possible trading options for each currency will be printed. After that, lists of the top ten trading pairs with the highest volume; the highest price; the highest price change in percent; the lowest spread; and the highest spread between ask price and bid price will be printed. Ultimately, a pie chart that shows the three trading pairs with the highest volume will be displayed.

When the program is started in interactive mode by typing “python3 cryptoGraph.py -i <asset_filename> <trade_filename>” (e.g. python3 cryptoGraph.py -i assetData.json tradeData.json) in the terminal window, the following menu will appear (Start Menu):

> Enter selection:
>
> (1) = Load Data
> (0) = Exit
>
> Your selection…
>
> Typing “0” and pressing enter in the start menu will terminate the program.
> Typing “1” and pressing enter in the start menu will show the following menu (Load Data Menu):
>
> Enter selection:
>
> (1) = Load Asset Data
> (2) = Load Trade Data
> (3) = Load Serialised Data
> (4) = Load Data to Graph
> (0) = Exit
>
> Your selection…

Typing “1” and pressing enter in the load data menu will load the file assetData.json and store the required data in a linked list. Time complexity ≈ O(N)

Typing “2” and pressing enter in the load data menu will load the file tradeData.json and store the required data in a linked list. Time complexity ≈ O(N)

Typing “3” and pressing enter in the load data menu will ask you for a file name of the serialised file you would like to load. If the file has been loaded successfully, a graph has been created and you are able to use the other functions of the program.
Time complexity ≈ O(N)

Typing “4” and pressing enter in the load data menu will load the previously loaded asset and trade data to a graph. If the graph has been created successfully, you are able to use the other functions of the program. Time complexity ≈ O(N)

Typing “0” and pressing enter in the load data menu will show the following menu (Main Menu):

Enter selection:

(1) = Load Data
(2) = Find and Display Asset
(3) = Find and Display Trade Details
(4) = Find and Display Trade Paths
(5) = Set Asset Filter
(6) = Asset Overview
(7) = Trade Overview
(8) = Save Data (serialised)
(0) = Exit

Your selection…

Typing “1” and pressing enter in the main menu will show the previously described Load Data Menu.

Typing “2” and pressing enter in the main menu will ask you for an asset name. If the asset name exists, a list of all possible trading options including the exchange price for the particular asset will be printed.

Typing “3” and pressing enter in the main menu will ask you for a trade pair name. If the trade pair name exists, a list containing the base asset; quote asset; ask price; ask quantity; bid price; bid quantity; highest price (last 24h); lowest price (last 24h); volume (last 24h); open price (last 24h); last price; price change (last 24h); and the spread in percent between the ask price and the bid price.

Typing “4” and pressing enter will first ask you to enter the asset name of the start asset, and then to enter the asset name of the desired asset. If both assets exist, the following menu will be shown (Find Path Menu):

Enter selection:

(1) = Find possible paths (fast)
(2) = Find possible paths including price (slow)

Enter selection…

Typing “1” and pressing enter in the find path menu will display the asset names of all possible trading paths from the start asset to the desired asset.

Typing “2” and pressing enter in the find path menu will calculate the exchange rate and display the asset names of all possible trading paths from the start asset to the desired asset. After that, the cheapest possible trading path will be displayed.

Typing “5” and pressing enter in the main menu will show the following menu (Set Asset Filter Menu):

Enter selection:

(1) = Remove Asset
(2) = Remove Trade Pairs with High Spread
(0) = Main Menu

Your selection…

Typing “1” and pressing enter in the set asset filter menu will ask you for an asset name that you would like to delete. If the asset name exists, the asset and all its connections will be removed from the graph.

Typing “2” and pressing enter in the set asset filter menu will remove half of all available trading pairs that have a high spread between ask price and bid price. This will eliminate expensive trading paths and reduces the time to identify the cheapest trading path.

Typing “0” and pressing enter in the set asset filter menu will show the main menu.

Typing “6” and pressing enter in the main menu will print a list of all available currencies, as well as the exchange rate of all possible trading options for each currency.

Typing “7” and pressing enter in the main menu will show the following menu (trade overview menu):

Enter selection:

(1) = Top 10 Volume
(2) = Top 10 Price
(3) = Top 10 Price Change
(4) = Top 10 Lowest Spread
(5) = Top 10 Highest Spread
(6) = Top 3 Highest Volume Pie Chart
(0) = Exit

Enter selection…

Typing “1” and pressing enter in the trade overview menu will display the ten trading pairs with the highest volume including additional information for each trading pair.

Typing “2” and pressing enter in the trade overview menu will display the ten trading pairs with the highest price including additional information for each trading pair.

Typing “3” and pressing enter in the trade overview menu will display the ten trading pairs with the highest price change including additional information for each trading pair.

Typing “4” and pressing enter in the trade overview menu will display the ten trading pairs with the lowest spread including additional information for each trading pair.

Typing “5” and pressing enter in the trade overview menu will display the ten trading pairs with the highest spread including additional information for each trading pair.

Typing “6” and pressing enter in the trade overview menu will display a pie chart that shows the 3 trading pairs with the highest volume.

Typing “0” and pressing enter in the trade overview menu will show the main menu.

Typing “8” and pressing enter in the main menu will ask you to enter a name for the serialised file you would like to save. The name should not contain any special characters and not be longer than 20 characters.

Typing “0” and pressing enter in the main menu will terminate the program.

Future Work

In future, data from asset_info.csv could have been added to the matching graph vertices and included for various functions such as for find and display asset; set asset filter, or asset overview. The various sorting functions in the linked list class should be compacted into one sorting function that takes two arguments: 1 = attribute to sort; 2= increasing or decreasing. Error handling regarding low spread graph function should be improved so that the need of additional functions (e.g. findPath1 and findPath2) after low spread could be avoided. Additional functions could have been added to classes.py in order to compress code in the main program such as for example function displayTradePair() or removeVertex() to the class Graph. Certain functions such as the findPath() function in the class Graph could have been separated in multiple functions: in this example one function to find the path, and the other function to calculate the exchange rate. This would improve the code readability. The time complexity of various functions could have been improved by replacing the O(N) linked list functions with similar functions such as a binary search tree with O(log N), or hash tables with O(1). Moreover, private functions and wrapper functions should be implemented to

establish a clean and consistent code layout, and to avoid access to certain functions from outside the classes or modules.
In further analysis, profitable loops could be identified through modifying the findPath algorithm. Moreover, the ask quantity and bid quantity could be utilised to estimate the future performance of trading pairs. A high ask quantity in combination with a low bid quantity might indicate a price decrease for the base asset and a price increase for the quote asset, and vice versa.

Traceability Matrix

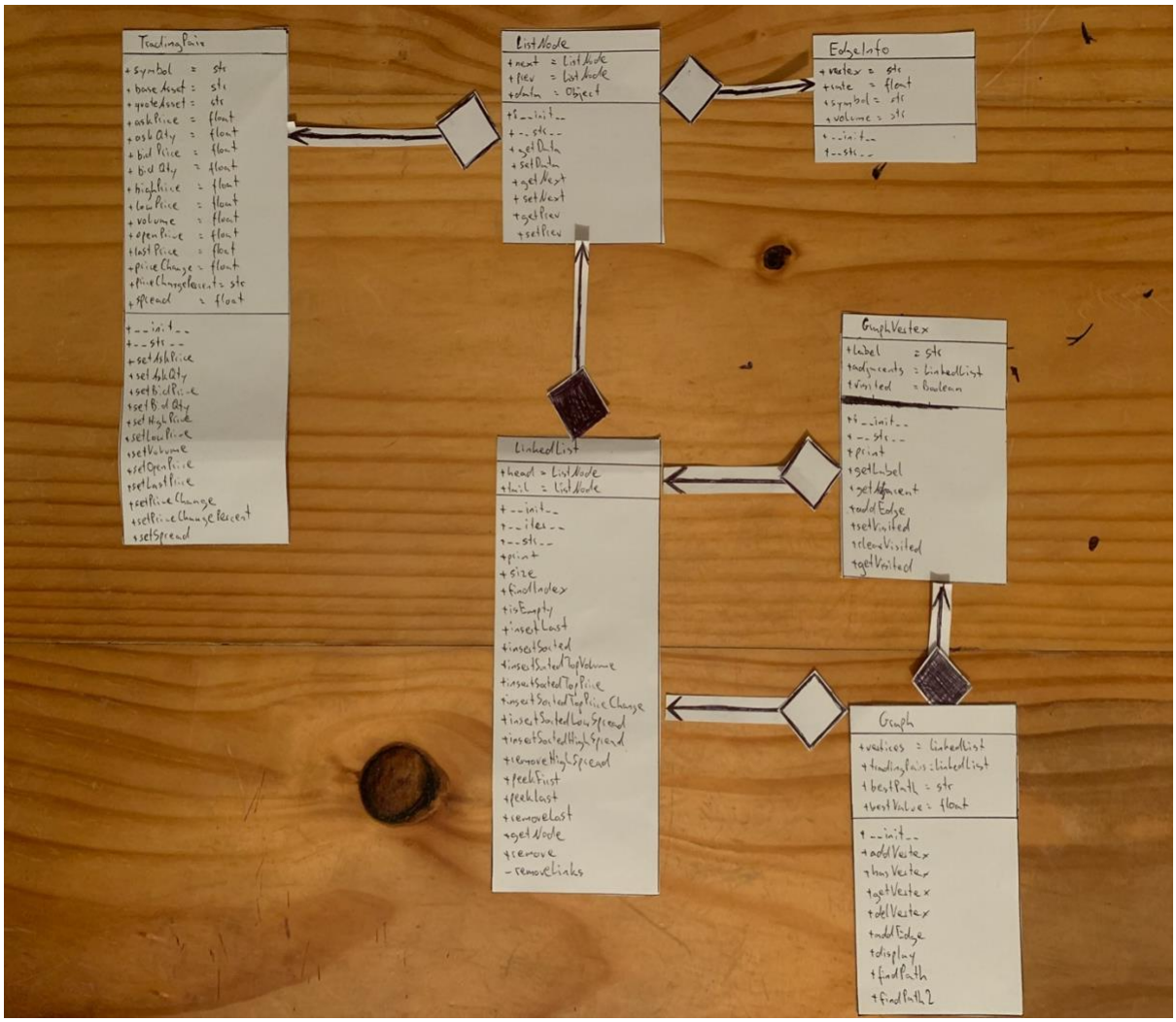
		Requirements	Design / Code	Test
1	Driver/Menu & Modes	1.1 System displays usage if called without arguments.	cryptoGraph.py: line 33	UI Test: Live demonstration
		1.2 System displays report mode with "-r <asset_data> <trade_data>" arguments.	cryptoGraph.py: line 808	UI Test: Live demonstration
		1.3 In report mode, system displays asset overview and trade overview.	cryptoGraph.py: line 886	UI Test: Live demonstration
		1.4 System displays interactive menu with "-i" argument.	cryptoGraph.py: line 37	UI Test: Live demonstration
		1.5 In interactive mode, user enters command and system responds.	cryptoGraph.py	UI Test: Live demonstration
		1.6 System displays usage if called with wrong arguments.	cryptoGraph.py: line 967	UI Test: Live demonstration
		1.7 System displays instructions if user selects unavailable function.	cryptoGraph.py: e.g. line 249	UI Test: Live demonstration. Print feedback
		1.8 System returns appropriate error messages from wrong user input.	cryptoGraph.py: e.g. line 76	UI Test: Live demonstration
2	File I/O	2.1 In report mode, if asset data and trade data loaded successfully, correct graph is produced.	cryptoGraph.py: line 868	UI Test: Live demonstration. Print feedback
		2.2 In interactive mode, to read file, system prompts user for filename.	cryptoGraph.py: line 83 and line 129	UI Test: Live demonstration.

		2.3 After serialised data loaded successfully, correct graph is produced.	cryptoGraph.py: line 196	UI Test: Live demonstration. Print feedback
		2.4 After trade data and asset data loaded successfully, user is able to produce correct graph.	cryptoGraph.py: line 243	UI Test: Live demonstration. Print feedback
		2.5 After graph produced successfully, user is able to save it in a serialised file.	cryptoGraph.py: line 762	UI Test: Live demonstration. Print feedback
		2.6 In interactive mode, to save file, system prompts user for filename.	cryptoGraph.py: line 772	UI Test: Live demonstration.
		2.7 System returns appropriate error messages for user input and file io.	cryptoGraph.py: line 778	UI Test: Live demonstration. Print feedback
		2.8 When loading serialised file, system recognises if graph is low spread or not.	cryptoGraph.py: line 226	UI Test: Live demonstration. Print feedback
3	Find and display asset	3.1 System prompts user for asset name.	cryptoGraph.py: line 289	UI Test: Live demonstration. Print feedback
		3.2 System returns appropriate error messages.	cryptoGraph.py: e.g. line 301	UI Test: Live demonstration. Print feedback
		3.3 System searches and displays asset with additional information	Class Graph: getVertex() classes line 833	UI Test: Live demonstration. unitTestGraph: Test getVertex
4	Find and display trade details	4.1 System prompts user for trade pair name.	cryptoGraph.py: line 326	UI Test: Live demonstration.
		4.2 System returns appropriate error messages	cryptoGraph.py: e.g. line 337	UI Test: Live demonstration. Print feedback
		4.3 System searches and displays trade pair with additional information.	cryptoGraph.py: line 342	UI Test: Live demonstration.
5	Find and display potential trade paths	5.1 System prompts user for start and destination asset names.	cryptoGraph.py: line 385 and line 410	UI Test: Live demonstration.

		5.2 System returns appropriate error messages for user input.	cryptoGraph.py: e.g. line 406	UI Test: Live demonstration. Print feedback
		5.3 After asset names entered successfully, user can select find path mode (fast, or slow with exchange rate)	cryptoGraph.py: line 436	UI Test: Live demonstration.
		5.4 System displays all potential paths.	Class Graph: findPath() line 938; and findPath2() line 1001	UI Test: Live demonstration. unitTestGraph: Test find path fast
		5.5 System displays all potential paths including correct exchange rate.	Class Graph: findPath() line 957; and findPath2() line 1020	UI Test: Live demonstration. unitTestGraph: Test find path not fast
		5.6 After all possible paths identified in slow mode, system displays cheapest possible path.	cryptoGraph.py: line 460 or line 466	UI Test: Live demonstration. unitTestGraph: Test find path not fast
6	Set asset filter	6.1 System prompts user for asset name to delete.	cryptoGraph.py: line 499	UI Test: Live demonstration.
		6.2 System deletes vertex and affected edges.	cryptoGraph.py: lines 524 - 553	UI Test: Live demonstration. Print feedback. unitTestGraph: Test delVertex
		6.3 System returns appropriate error messages for user input.	cryptoGraph.py: e.g. line 521	UI Test: Live demonstration. Print feedback.
		6.4 System deletes old graph and creates new graph with half of previously existing trading pairs that have a low spread.	cryptoGraph.py: line 556	UI Test: Live demonstration. Print feedback.
7	Asset overview	7.1 System searches and displays all assets with additional information.	Class Graph: display() line 898	UI Test: Live demonstration. unitTestGraph: Test display /

				Test display with edges
8	Trade overview	8.1 System identifies and displays the ten trading pairs with highest volume	cryptoGraph.py: identifies: line 635 display: line 680	UI Test: Live demonstration.
		8.2 System identifies and displays the ten trading pairs with highest price.	cryptoGraph.py: identifies: line 635 display: line 690	UI Test: Live demonstration
		8.3 System identifies and displays the ten trading pairs with the highest price change in last 24h.	cryptoGraph.py: identifies: line 635 display: line 700	UI Test: Live demonstration
		8.3 System identifies and displays the ten trading pairs with the lowest spread.	cryptoGraph.py: identifies: line 635 display: line 711	UI Test: Live demonstration
		8.4 System identifies and displays the ten trading pairs with the highest spread.	cryptoGraph.py: identifies: line 635 display: line 722	UI Test: Live demonstration
		8.5 System creates and displays a pie chart showing the three trading pairs with the highest volume.	cryptoGraph.py: line 733	UI Test: Live demonstration

Class Diagram



Class Description

Graph	Graph objects represent the cryptocurrency trading platform. It stores the various available currencies and trading pairs. A graph object allows to simulate the various connections between the available currencies.

GraphVertex	GraphVertex objects represent the various currencies that are available on the cryptocurrency trading platform. They store the possible connections to other GraphVertex objects within the Graph object.
TradingPair	TradingPair objects represent the various trading pairs that are available on the cryptocurrency trading platform. Its main purpose is to store the trading pair name in combination with additional information in one object that allows for easy access.
EdgeInfo	EdgeInfo objects contain information about the various edges between vertices. Their main purpose is to store the exchange rate for each edge in one direction.
LinkedList	LinkedList objects enable the dynamic storage of other objects in one list that allows for flexibility and easy manipulation. It enables users to iterate through all stored objects and to find objects at a particular index.
ListNode	A ListNode object represents an item that is stored in a linked list. It has references to the nodes that are stored before and after its own position within the linked list. It is able to store values or other objects such as a TradingPair object, or an EdgeInfo object.

Justification

Graph + GraphVertex:
A graph enabled the interconnection between currencies without any limitation. Because some currencies are tradeable with many other currencies in both directions, trading cycles occur. This is why a graph data structure as the only possible solution in this regard.

LinkedList + ListNode:
A linked list allowed for easy and fast implementation as it is less complex than other data structures that could have been utilised instead – such as trees or hash tables. Moreover, linked lists allow for flexibility as they don't have a fixed size and are able to grow and shrink according to the requirements. However, hash tables would have been a better solution for some cases as they allow for quicker access to data with a time complexity of O(1) compared to the average time complexity of O(N) for a linked list.

EdgeInfo:
This class requires extra storage and is not necessarily required as the exchange rate could have been calculated by utilising the information that is stored in the TradingPair objects. However, it has been implemented to improve the timely performance for various functions such as graph.findPath() or graph.display(). This is due to the fact that the exchange rate does not need to be calculated during the execution of those functions.

TradingPair:
This class enabled the storage of related data in one object.