CMPSC 463 Project 1

Description:

This project is a stock price analysis tool created in python that is designed to parse stock market data from a CSV file and apply several algorithms to analyze trends, anomalies, and calculate profits and losses across critical time frames. The program uses algorithms such as merge sort for ordering the series by time, Kadane's algorithm for calculating maximum profit and loss, and divide-and-conquer algorithms for anomaly detection through closest and farthest pairs.

Structure of the Code:

CSV Parsing and Data Preprocessing

The `read_stock_data` function reads the CSV file, extracts relevant fields (e.g., Date, Close, Volume, Open, High, Low), and converts them into a list of dictionaries for analysis.

Sorting

The **`merge_sort`** function sorts the parsed stock data by a key, which in most cases will be the date for time series analysis. Merge sort implements a divide-and-conquer approach for sorting which is suitable for large data sets such as financial data, splitting the problems into smaller subproblems then merging them back together in a sorted array.

Profit/Loss Calculation

The `max_profit_period` and `max_loss_period` functions use Kadane's Algorithm to calculate the timeframes with which buying from one point and selling at a point in the future will result in the most profit or loss. By using these algorithms the program can give insight into when the optimal times were to buy and sell to optimize gains and minimize losses.

Anomaly Detection

The `closest_pair` and `farthest_pair` functions analyze the stock data to detect anomalies, such as when stock prices across two consecutive days have the smallest or largest differences. This can help to identify unusual market data or events where volatility is high, low, or due to annual anomalies such as new years.

Running Code:

Step 1: Install Python 3.x

Step 2: Prepare CSV File

A CSV of Tesla stock data is already included in the src folder. However, if you choose to use your own stock data it must have the following columns:

- Date
- Close/Last

- Volume
- Open
- High
- Low

Any additional columns will not be processed, and any columns that are missing may result in errors when calculating insights such as max profits, losses, closest pairs, and farthest pairs. Also Ensure that the Date column is formatted as %m/%d/%Y (e.g., 08/17/2023) and that currency values such as Close, Open, High, and Low begin with a dollar sign ($).

To run the code execute the following command in the project's "src" directory:

```
python analysis.py
```

Verification of code functionality:

Consider the following dataset in stock_data.csv:

```
Date,Close/Last,Volume,Open,High,Low

01/01/2023,$100,10000,$95,$105,$90

01/02/2023,$120,15000,$100,$125,$95

01/03/2023,$110,12000,$115,$130,$105

01/04/2023,$140,16000,$110,$145,$100

01/05/2023,$130,14000,$140,$150,$125
```

CSV Parsing and Processing

`sorted_data = read_stock_data('stock_data.csv')` parses the data into a list of dictionaries, where each element in the list represents a row, and where each element in the row is a dictionary where the key is the column header and the value is the value in the cell. It also strips dollar signs from the currency values. This information is stored in the `sorted_data` variable.

```
sorted_data[0] =

{'Date': datetime.datetime(2024, 9, 13, 0, 0), 'Close': 100,
'Volume': 10000, 'Open': 95, 'High': 105, 'Low': 90}
```

Merge Sort

`merge_sort(sorted_data, key='Date')` sorts the data by date. Comparisons will be made through the divide and conquer merge sort algorithm, comparing `datetime.datetime` values with one another until the whole list is sorted. In this case, no sorting is needed since the dataset is already in order by time.

Max Profit/Loss

`max_profit_period(sorted_data)` and `max_loss_period(sorted_data)` will respectively find the maximum and minimum subarrays within the dataset and return the buy and sell dates with which one would make the most money or lose the most money so long as the buy date is before the sell date.

```
Max Profit Period: $100 at 01/01/2023 to $140 at 01/04/2023

Max Loss Period: $140 at 01/04/2023 to $130 at 01/05/2023
```

Anomaly Detection

`closest_pair(sorted_data)` and `farthest_pair(sorted_data)` will respectively find the two consecutive days where there was the least and most disparity in closing prices. This will identify anomalies or critical timeframes where prices change the least or the most, and may identify points of low and high volatility.

```
Closest Pair Anomaly: 01/04/2023 - 01/05/2023, Difference: $10.00

Farthest Pair Anomaly: 01/03/2023 - 01/04/2023, Difference: $30.00
```

Discussion of Findings:

This program provides a rudimentary way to see trends in past stock data such as when to buy and sell to maximize profit, when you should avoid buying to avoid depreciation of assets, or when there are large fluctuations in the market due to high volatility or some other event. It could definitely be improved however, as using algorithms intended for maximum and minimum subarrays may find the greatest points to buy and sell for profit, but does not take into account the amount of time that may take. Buying and selling multiple times throughout the month may prove more profitable than holding for many years. I do believe the anomaly detection works well as is and may prove useful for avoiding patterns throughout the year where there is high, low, or unpredictable volatility.