Data Structures and Algorithms

Homework 2

Instructions. Complete the functions in the file `hw2.py` following the instructions below and in the file. Submit the file through the Assessment tab on the Hub. Your submission filename should be the same as the current filename, ie `hw2.py`.

The assignment is graded mainly on the correctness of your functions, but writing clean and expressive code may compensate for some shortcomings in correctness. The assignment is worth 15% of your total course grade. Each of the functions below will receive equal weight in grading; each function that works correctly for at least some inputs will earn you partial credit.

You're welcome to discuss solution approaches with your classmates, but please complete the coding individually. The assignment is due on 27/9 at 10am.

Your task: technical analysis for stock trading. Some stock traders vouch for so-called 'technical analysis': using mathematics and statistics to predict stock-price movements from historical data. Others say that this is like driving a car by only looking at the rear-view mirror. You will explore which group is right.

The specific technical analysis idea you'll study is using so-called *moving-average strategies*. A moving average is defined as follows. Consider a daily quoted time series of a stock price, $x(t)$, eg the daily adjusted closing prices for the stock. An n-step (or n-lag) moving average is the average of the n last values of the price. More precisely, the n-step moving average is defined as follows:

$$MA_n(t) = \frac{x(t) + x(t-1) + ... + x(t-(n-1))}{n} \tag{1}$$

The one-step MA is simply the current price: $MA_1(t) = x(t)$, while the two-step MA is the average of the current price and the previous day's price. Essentially, if the moving average span n is short, the MA reacts quickly to any shocks to the stock price; if the span n is long, it reacts slowly. If the moving average spans the stock's entire period of existence, it is just the all-time average price.

Figure 1 below shows an example of a stock index's development, along with two different moving averages. Notice how the two-step moving average follows the index more closely than the five-step one, which reacts to price changes more slowly and less strongly.

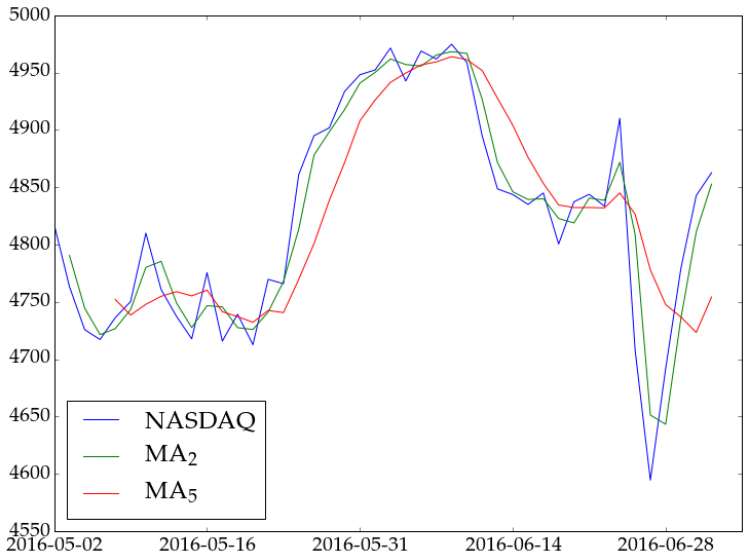

Figure 1: The Nasdaq index in May and June 2016 and its two-step and five-step moving averages.

The theory put forward by proponents for technical analysis has it that moving averages give an indication of the 'momentum' of a price. In particular, by comparing two moving averages of different lengths, one should be able to infer that the momentum of a the stock is changing, giving a signal when to buy or sell the stock.

This comparison between two moving averages is done by finding so-called *crossover points* between different moving averages. For example, some advocates of technical analysis claim that by looking at the 20-day and 50-day moving averages of a stock price, one can identify longer trends and benefit from them. This works as follows:

- Keep track of two moving averages of different lengths: a shorter one (a lower n, eg $n_L = 2$ in the figure) and a longer one (a higher n, eg $n_H = 5$ in the figure)

- When the shorter MA 'crosses' the longer one (it has been lower but becomes higher), this is a signal to buy the stock as it seems to be gaining momentum. More specifically, if for any time t, $MA_{n_H}(t-1) \geq MA_{n_L}(t-1)$ and $MA_{n_L}(t) > MA_{n_H}(t)$, you should buy the stock. In Figure 1, these points are the ones where the green line has been lower than the red line

but then crosses it to become higher, eg around 29 June. (You can find other such points in time by examining the figure.)

- Conversely, when the longer MA 'crosses' the shorter one in the same way, this is a signal to sell the stock as it seems to be losing momentum. In Figure 1, these are the points in time where the red line has been lower than the green one, but then becomes higher, eg around 10 June.

Based on the figure, it looks like those dates may indeed have been good times to trade. But in general, can you make money based on such strategies? Your task is to evaluate this claim by writing a Python script that calculates two moving averages of a (historical) stock/index price, finds all the crossover points between these moving averages, and finally performs trades based on those crossover points.

Complete the following functions in the Python file:

- `movingAverage`: takes as input a `list` of stock prices (which are floats) ordered in time, and an integer n. Returns the n-day moving average for the prices. More specifically, returns a `list` with values `None` for the first $n-1$ values (where the moving average is not well defined) and the n-day moving average for the rest of the values. Do not use readymade Python libraries for calculating moving averages (but you may use `sum` on a list...).

- `crossOvers`: takes as input two lists (containing floats, eg two moving averages of prices) and returns a list of their crossover points. Each item in the returned list is itself a list containing two integers: `[timeIndex,higherIndex]`, where `timeIndex` is the index at which the crossover happens, and `higherindex` indicates which price becomes higher at `timeIndex`: its value is either 1 for the first list of prices or 2 for the second list.

  - The function should start making these comparisons at the point in time where both moving averages have numerical values instead of `None`-values as described above (before that comparisons are impossible).
  - When you eventually call this function to calculate crossovers with two moving averages as input, the first moving average should be the lower-n moving average. This does not matter for how the function itself works, but makes the output consistent with the assumption of the `makeTrades` function below.

- `makeTrades`: takes as input an initial cash position (a float), a list of prices, and a list of crossovers as described above. Uses these data to make trades at crossover points as described below. Returns a list containing the total value of the portfolio for all points in time (whether you currently hold cash or stock).
  - For any crossover `[timeIndex,higherindex]`, assume that whenever `higherindex == 1`, this is an indication to buy, and 2 is an indication to sell at this `timeIndex`.
  - If there is an indication to buy, AND there is a cash position, the function uses all cash to buy stock at the current price. You may assume that whenever a crossover happens, you're able to make a trade at the current day's price, without waiting for the next day.
  - Conversely, if there is an indication to sell AND a stock position, it converts all stock to cash at the current price.
  - As a result, you will always either hold cash or hold stock, but never both.
  - Assume you can buy fractional amounts of stocks, and there are no trading fees.

  Hints:

  - You'll most likely need to somehow track whether your current position at any point in time is in cash or in stocks (and perhaps how many stock you hold).
  - Using this, you'll probably then want to loop over the price time series, making trades at the appropriate crossovers. This would mean changing your position as appropriate at the current price whenever a trade occurs, and also keeping track of the value of your position to return it...

You can test your implementation with the provided functions. When comparing the values, don't worry about the exact precision of eg moving averages – we will round them when testing your implementation.

- `testMovingAverage()`: test your moving-average function
- `testCrossOvers()`: test your crossover function
- `testMakeTrades()`: test your trading function

- `plotComparison()` and its helper functions: this function uses Python libraries to download stock price data and compare your implementation of a moving-average crossover strategy with a buy-and-hold strategy (where you simply buy at the first timestep and never trade again).
  - This function uses the library `pandas_datareader` to read Yahoo Finance data from the web. If this library is not installed on your machine, you can get it by opening the command prompt on your machine and typing `conda install pandas-datareader`. While you're at it, you can update all Python packages in your installation by typing `conda update --all`.

Extra (not graded): are moving-average strategies good? Explore this question:

1. Pick a stock or index (eg Nasdaq) and a period of time, eg the years 2000 – 2010. This is your *training* data set.
2. Using your training data, go through all possible moving average strategies (n_H, n_L) for the stock, for example up to $n = 100$. Store the ones that perform best in terms of return to your investment.
3. Test the *out-of-sample* performance of these best strategies on the years following your training data set, eg 2011 – 2015. Do the strategies still make money in this time period? How would your analysis change if you had to pay trading fees of say 1%?
4. Repeat this for different stocks and time periods. If you find a strategy that consistently outperforms "buy and hold", I'd like to know — send me an email! (Or keep it to yourself and start trading...)