Algorithm Design and Analysis S2 2023
Software Assignment
Solving Currency Exchange Problems

Due Date 16 October 2023.

Background: A currency exchange graph is a graph whose n nodes represent different currencies, and whose directed edges represent exchange rates r_{uv} between those currencies.

In financial markets, arbitrage opportunities arise when one can buy a set of currencies and sell them in a sequence to end up with more of the starting currency, without any net investment. In a currency exchange graph, this translates to finding a cycle where the product of the exchange rates is greater than 1. For example if we have three currencies A, B and C, with rates $r_{AB} = 0.651, r_{BC} = 0.952, r_{CA} = 1.711$, then the product of the rates $r_{AB} \times r_{BC} \times r_{CA} = 0.651 \times 0.952 \times 1.711 \approx 1.060$, which means 6% profit if trading through the cycle A-B-C-A.

A currency exchange graph with **negative logarithm** is a currency exchange graph with each edge associated with the negative logarithm of the exchange rates as the weights, e.g., $w(A, B) = -log(r_{AB})$, $w(B, C) = -log(r_{BC})$, $w(C, A) = -log(r_{CA})$. We then have $w(A, B)+w(B, C)+w(C, A) = -log(r_{AB} \times r_{BC} \times r_{CA}) < 0$, as $r_{AB} \times r_{BC} \times r_{CA} > 1$. This means if the sum of the weights is negative (i.e., there is a cycle of negative weights), there is an arbitrage opportunity.

A related problem is to find the best conversion rate from one currency to another. We typically assume that there is no arbitrage in this case. In a currency exchange graph with negative logarithm, this translates to finding the shortest distance from one node (currency) to another.

Note that a currency exchange graph can be represented as $n \times n$ adjacency matrix with currency exchange rates. You don't have to consider the case where there is no direct exchange rate between two currencies.

Tasks: To use shortest path algorithms in dynamic programming for solving two Currency Exchange Problems:

- Task 1: Finding the best conversion rate from one currency to another

The input: $n \times n$ adjacency matrix for currency exchange rates, currency X and Y

The output: give the best conversion rate from currency X to Y, and the corresponding sequence of exchanges to achieve the best rate.

- Task 2: Detecting arbitrage opportunities in a currency exchange system

  The input: $n \times n$ adjacency matrix for currency exchange rates

  The output: whether there is arbitrage in the system, and if so, give the currency sequence $v_0, v_1, v_2, ..., v_{k-1}$ for which $r_{v_0 v_1} \times r_{v_1 v_2} \times ... \times r_{v_{k-1} v_0} > 1$.

Instructions: Some extra details on how the tasks can be achieved and what are expected.

- Currency Exchange Graph representation and Test Cases generation.
  - Create at least two currency tables such as Table 1 and convert them into Currency Exchange Graphs with negative logarithm as test cases.

From/To	A	B	C
A	1	0.651	0.581
B	1.531	1	0.952
C	1.711	1.049	1

Table 1: Sample exchange rates

  - Gather real-world currency exchange rates (can include digital currencies). You can use APIs like exchangeratesapi.io or other financial data services. Convert these exchange rates to a graph representation.

- For Task 1: First check if there is an arbitrage, by extending the Bellman-Ford algorithm so that it can be used on the currency exchange graph with negative logarithm to detect if there is a negative cycle.

  Hint, to get the sequence associated with best rate, one needs to maintain the predecessor for each node when computing the shortest path associated with the recurrence:

  $d_{k+1}(u) = min\{d_k(u), min\{d_k(v) + w(v,u) \mid (v,u) \in E\}\}$

- For Task 2: Add functionality to not only detect the presence of an arbitrage cycle but also to retrieve and print the cycle. This will similarly need the predecessor info.

- Any programming languages can be used.

- This allows individual or group work up to 3 students.
- Submission Guidelines:

  The submission shall include the following in a zip file with student ID(s)

  - A code file(s).
  - A report detailing (1) the program design to achieve the task 1 and 2; and (2) findings, including any identified arbitrage opportunities, potential gains, and sample inputs used for testing, alongside expected and observed outputs (screenshots).

  The report should include student name(s), ID(s) and signature(s). If it is by a group, just nominate one student to do the submission via Canvas.
- Marking Guides:
  - Pass with distinction (>= 80%): All functionalities achieved, code well-commented, the report is clear and detailed.
  - Pass with merit (>= 65% and < 80%): Majority of the required functionalities are implemented with few minor issues. Adequate commenting in the code but might lack in some areas. The report provides a basic explanation but might lack clarity in some sections.
  - Pass (>= 50% and < 65%): Basic functionalities are present, but some might be flawed or missing. Minimal commenting in the code, making certain parts hard to understand. The report provides a basic explanation but might lack clarity in some sections.
  - Not pass (< 50%): Significant portions of the required functionalities are missing or flawed. Code is poorly commented, making it difficult to understand. The report is unclear, too brief, or missing major sections.