Algorithm Design and Analysis

Assignment Four

Due Tuesday 16 December 2022

A *currency exchange graph* is a graph whose n nodes represent different curriencies, and whose directed edges represent exchange rates r_{uv} between those currencies (for example $r_{€\$} \approx 1.78$ for exchanging Euro to New Zealand dollars).

For convenience the edges can be weighted using $w_{uv} = \log \frac{1}{r_{uv}}$ (for example $w_{€\$} \approx -0.58$), allowing the weight of adjacent edges to be added together to find combined exchange rates, and shorter paths result in higher exchange rates.

The purpose of this assignment is to develop useful graph tools for currency trading.

The assignment should include the following components, each with some suitable test examples:

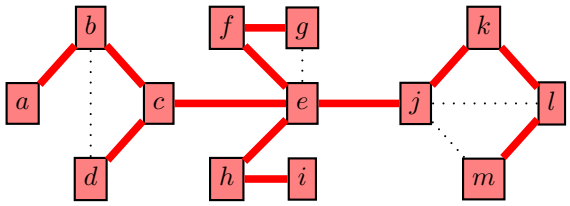
Best Conversion Finder which accepts an $n \times n$ *connectivity* table of exchange rates (where 0 denotes no direct exchange rate between two currencies), and can calculate the best conversion rate between any two specified currencies (both ways), and how that rate can be obtained both ways as sequences of exchanges. **(10 marks)**

Arbitrage Finder which accepts an $n \times n$ table of exchange rates and efficently finds whether there is *arbitrage* in the system, a closed loop of exchanges that results in a profit. If so then all the occurrences of arbitrage, currencies $v_0, v_1, v_2, \ldots, v_{k-1}$ for which $r_{v_0 v_1} \cdot r_{v_1 v_2} \cdot \ldots \cdot r_{v_{k-1} v_0} > 1$ should be found (allowing a currency trader to exploit a price differential to make a profit). **(20 marks)**

Bridge Exchange Finder which accepts an $n \times n$ table of exchange rates and forms an undirected (and unweighted) graph which has edges between currencies if those two currencies can be directly exchanged. It then finds any exchanges (called *bridge* edges in the graph) which if were unavailable (its edge removed from graph) would mean some currencies could no longer be traded with each other via a sequence of exchanges (the graph would be disconnected).

The bridges can be found by first performing a depth first search of the undirected graph, labelling each vertex u with a counter $d(u)$ as the vertex is discovered, and with a value $m(u)$ as the search is finished with the

vertex. The value $m(u)$ is taken to be the smallest value between $d(u)$, the values of $m(v)$ found while visiting each adjacent currently black (child) vertex v, and the value of $d(v)$ for any adjacent currently grey (already discovered) vertex v other than its parent u.



For example, if in the illustrated diagram a depth-first search starting at a might visit the vertices in the following order:

v	a	b	c	d	e	f	g	h	i	j	k	l	m
$d(v)$	0	1	2	3	4	5	6	7	8	9	10	11	12

and when finished with each vertex the following values of $m(v)$ would be calculated:

v	a	b	c	d	e	f	g	h	i	j	k	l	m
$m(v)$	0	1	1	1	4	4	4	7	8	9	9	9	9

An edge between u and v is a bridge if and only if it is included in the depth first tree from u to v and $m(v) > d(u)$. **(15 marks)**

Demonstration of the programs with a 3-5 minute video. **(5 marks)**