Problem 1: A min-max spanning tree of an undirected weighted graph $G$, is a spanning tree $T$ for $G$ that minimizes the weight of the "heaviest" edge in $T$. In other words: define $\text{max}_e(T) = \max_{e \in T} [w(e)]$ (rather than the sum of the weight of the edges in $T$ as in the notion of minimum spanning tree we saw in class).

- Prove that a minimum spanning tree is also a min-max spanning tree;
- Show that you can find a min-max tree in linear time $O(|V| + |E|)$

Problem 2: We saw in class Dijkstra’s algorithm is asymptotically faster than the Bellman-Ford algorithm to find shortest paths, but it requires all edges to be non-negative.

- Show an example of a graph with some edges with negative weights where Dijkstra’s algorithm fails to find a shortest path;
- Let $G = (V, E)$ be a weighted graph with weight function $w$ and some edges with negative weights. Let $W$ be the minimum weight in the graph with $w < 0$. Then for each edge $e \in E$ change the weight function to $w'(e) = w(e) - W$. Note that now all edges are non-negative according to the weight function $w'$. Now run Dijkstra’s algorithm on this graph. What happens? Is the shortest path according to $w'$ be the same as the one according to $w$. Why?

Problem 3: After graduating from City College you moved to a successful career as a data analyst on Wall Street. You notice that there is a potential fortune to be made in currency trading by analyzing real-time currency exchange rates to find a way to turn $1 into more than $1 by a sequence of trades. For example suppose that $1 can buy 1.3EUR and that 1EUR buys .9 Swiss Francs, and that 1 Swiss Franc buys .86$ then by engaging in those trades you would end up with $1.0062.

Given $n$ currencies $c_1, \ldots, c_n$ and a table of mutual exchange rates, show an algorithm that finds such a sequence of trades, i.e. a sequence of currencies $c_{i_1}, \ldots, c_{i_k}$ such that one unit of currency $c_{i_1}$ is transformed into $k > 1$ units of the same currency, after being traded according to the sequence $c_{i_1}, \ldots, c_{i_k}$.

Analyze the correctness and running time of your algorithm.