Problem 1: 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 $w'$ the same as the one according to $w$. Why?

Solution:

- If you run Dijkstra on the graph below the algorithm will assign distance 2 to node c. That's because a is removed from the priority queue before b with "temporary" distance 1 while the correct distance is −1. The edge $(a, c)$ is relaxed only when a is removed from the queue, giving c the "permanent" distance 2. Later in the algorithm (when b comes out of the queue) the edge $(b, a)$ is relaxed giving a the correct distance −1 but at that point is too late to fix the distance of c since the edge $(c, a)$ will not be revisited.

- In the graph below (with negative edges) the shortest path from s to c is the path $s \rightarrow a \rightarrow b \rightarrow c$ of weight 0. After you add 1 to all the edges that path has weight 3, while the path $s \rightarrow c$ has weight 2 and is the new shortest path. Therefore the transformation does not preserve shortest paths.

Problem 2: 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 $c_{i_1}, \ldots, c_{i_k}$.

An 

**Solution:** Consider the complete graph $G$ where the nodes are the currencies $c_i$, there is an edge between each pair of nodes and the weight of that edge is the negative logarithm (in any base) of the currency rate: i.e. if $r_{ij}$ is the exchange rate between currency $c_i$ and $c_j$ we put weight $w_{ij} = -\log r_{ij}$ on edge $(c_i, c_j)$.

An arbitrage opportunity (the type of trade described in the problem where starting with 1 unit of currency $c$ one ends with more than 1 unit of the same currency) corresponds to a negative cycle in the graph. That’s easy to see by noticing that on that cycle the product of the exchange rates should be larger than 1.

Therefore we can run Bellman-Ford to detect if such a cycle exists. Since this is a complete graph it has $m = n^2$ edges, therefore the running time is $O(n^3)$.

**Problem 3:** Suppose that we want to maintain the transitive closure of a directed graph $G = (V, E)$ as we insert edges into $E$. As we saw in class we represent the transitive closure $G^* = (V, E^*)$ of $G$ as a Boolean adjacency matrix. Show how $G^*$ can be updated in time $O(n^2)$ when we add a new edge to $G$ (as usual $n = |V|$).

**Solution:** When you insert edge $(u, v)$ into $E$, then you need to add an edge $(u, w)$ to $G^*$ for every vertex $w$ that is now reachable from $u$. So you need to check $n - 1$ nodes, and for each node $w$ you need to see if a path from $u$ to $w$ exists (which takes $O(n)$ time).