1. Suppose that you have just finished computing the array dist[1..V, 1..V] of shortest-path distances between all pairs of vertices in an edge-weighted directed graph G. Unfortunately, you discover that you incorrectly entered the weight of a single edge $u \rightarrow v$, so all that precious CPU time was wasted. Or was it? Maybe your distances are correct after all!

In each of the following problems, let $w(u \rightarrow v)$ denote the weight that you used in your distance computation, and let $w'(u \rightarrow v)$ denote the correct weight of $u \rightarrow v$.

- (a) Suppose $w(u \rightarrow v) > w'(u \rightarrow v)$; that is, the weight you used for $u \rightarrow v$ was *larger* than its true weight. Describe an algorithm that repairs the distance array in $O(V^2)$ *time* under this assumption. [Hint: For every pair of vertices x and y, either $u \rightarrow v$ is on the shortest path from x to y or it isn't.]
- (b) Maybe even that was too much work. Describe an algorithm that determines whether your original distance array is actually correct in O(1) *time*, again assuming that $w(u\rightarrow v)>w'(u\rightarrow v)$. [Hint: Either $u\rightarrow v$ is the shortest path from u to v or it isn't.]
- (c) **To think about later:** Describe an algorithm that determines in O(VE) *time* whether your distance array is actually correct, even if $w(u \rightarrow v) < w'(u \rightarrow v)$.
- (d) **To think about later:** Argue that when $w(u \rightarrow v) < w'(u \rightarrow v)$, repairing the distance array *requires* recomputing shortest paths from scratch, at least in the worst case.
- 2. You—yes, you—can cause a major economic collapse with the power of graph algorithms!¹ The arbitrage business is a money-making scheme that takes advantage of differences in currency exchange. In particular, suppose that 1 US dollar buys 120 Japanese yen; 1 yen buys 0.01 euros; and 1 euro buys 1.2 US dollars. Then, a trader starting with \$1 can convert their money from dollars to yen, then from yen to euros, and finally from euros back to dollars, ending with \$1.44! The cycle of currencies \$→ ¥→ €→\$ is called an arbitrage cycle. Of course, finding and exploiting arbitrage cycles before the prices are corrected requires extremely fast algorithms.

Suppose n different currencies are traded in your currency market. You are given the matrix R[1..n] of exchange rates between every pair of currencies; for each i and j, one unit of currency i can be traded for R[i,j] units of currency j. (Do *not* assume that $R[i,j] \cdot R[j,i] = 1$.)

- (a) Describe an algorithm that returns an array V[1..n], where V[i] is the maximum amount of currency i that you can obtain by trading, starting with one unit of currency 1, assuming there are no arbitrage cycles.
- (b) Describe an algorithm to determine whether the given matrix of currency exchange rates creates an arbitrage cycle.
- *(c) **To think about later:** Modify your algorithm from part (b) to actually return an arbitrage cycle, if such a cycle exists.

1

¹No, you can't.