1. **Quick Fix.**

Your “friend” suggests that the easiest algorithm for finding shortest paths in a directed graph with negative-weighted edges is to make all the weights positive by adding a sufficiently large constant to each weight and then running Dijkstra’s algorithm. Give an example that you can show your friend to prove that his or her method is incorrect.

2. **Reductions.**

Show that the following problems can be reduced to the standard shortest path problems. No proof required.

- Given directed graph $G = (V, E)$ and two disjoint sets of nodes $S, T$. Find the shortest path from some node in $S$ to some node in $T$.
- $G$ is a directed graph and nodes and edges have non-negative lengths. Find $s$-$t$ shortest path where the length of a path is equal to the sum of the lengths of the nodes and edges on the path.
- Given a directed graph $G$ with node lengths (no edge lengths), is there a negative length cycle? Here the length of a cycle is the sum of the lengths of nodes on the cycle.
- $G$ is a DAG and each node has a non-negative length. Given two nodes $s, t$ in $G$, find the $s$-$t$ longest simple path in linear time.

3. **Limited Shortest Paths.**

We are given a directed graph in which the shortest path between any two vertices $u$ and $v$ is guaranteed to have at most $k$ edges. Give an algorithm that finds the shortest path between two vertices $u$ and $v$ in $O(km)$ time. Remember, edges can have negative weights.

4. **Almost Positive.**

We are given a directed graph $G = (V, E)$ with potentially negative edge lengths. Your friend ran Dijkstra’s algorithm and came up with a shortest path tree $T$ for distances from a node $s$. You realize that Dijkstra’s algorithm may not output distances correctly when a graph has negative edge lengths. However, before you run the more expensive Bellman-Ford algorithm, you wish to check whether $T$ is a correct shortest path tree or not. Describe an $O(m + n)$ time algorithm to do this check. Don’t forget to prove that your algorithm is correct!
5. **Average cycle.**

You are given a directed weighted graph $G$ (the weights are positive), and a number $x$. Design an algorithm that decides if $G$ has a cycle with average cost strictly smaller than $x$. The average cost of a cycle is the total weight of its edges divided by the number of edges. How fast is your algorithm?