1. Maximal Independent Set

(a) Consider a path with three vertices. The center vertex alone is a maximal independent set (one vertex is always independent, and the two remaining vertices are both adjacent so neither can be added). However, the two outer vertices are a larger independent set.

(b) Suppose, for the sake of contradiction, there is an independent set that is maximum but not maximal. Then since it is not maximal there is a vertex which can be added while still being independent. But then if we add that vertex we have found an even larger independent set. This is a contradiction to the original independent set being maximum.

(c) A greedy algorithm suffices:
Add an arbitrary vertex u to the independent set, delete u, every vertex v directly connected to u and all edges between them.

The resulting set is a maximal independent set – when we delete a vertex it is not legal to add to the independent set (and remains so for the rest of the algorithm) so we do get a maximal independent set.

2. Short Answer

(a) See 1b in original solutions.

(b) $\mathcal{O}\left(n^2\right)$, where n is the number of agents on each side.

(c) See 1c in the original solutions.

(d) We do not know – proving it NP-complete would show that $P = NP$ (because we know a polynomial time algorithm for 2-coloring), proving it is not NP-complete would prove $P \neq NP$ (We did not discuss this – it turns out that every polynomial-time solvable algorithm is reducible to every other polynomial-time solvable algorithm [by just ignoring what the library says and solving the desired problem directly], thus if $P = NP$, every problem in NP is NP-complete). Most computer scientists believe it is not NP-complete.

4. How Can You

(a) See 4a in original solutions.

(b) If your graph has no negative weight edges, replace every undirected edge (u, v) with two directed edges (u, v) and (v, u). If your graph has negative weight edges, the

(c) See 4b in original solutions.

(d) See 4c in the original solutions.

8. Capacity Reduction for NetFlow

Here is one possible algorithm, assuming the network has only integer capacities.

Let (u, v) be the edge with reduced capacity. Look at f. If it is still feasible (i.e. the capacity is not violated on (u, v) then we have a maximum flow – reducing a capacity can't make the max flow bigger!) Otherwise we must be violating (at least) the capacity on (u, v).

Run a BFS from v, only going along edges that have at least one unit of flow, Run a BFS from u, only going "backwards" along edges, and again only along edges with one unit of flow. (We are working in the original graph, not the residual graph). Record edges used and predecessors. When the first BFS reaches t, and the second BFS reaches s track predecessors back to u and v. Combining gives a path from s to t with at least one unit of flow. Remove one unit of flow from every edge on this path.

Now build the residual graph (in the usual way), and run a single additional iteration of Ford-Fulkerson. And return the updated flow.

Running Time: note that we run two BFSs (linear time), create a residual graph (linear time), and do one additional BFS to do the last Ford-Fulkerson iteration (linear time). We do a constant number of graph searches, so our overall time is $\mathcal{O}(m+n)$.

Correctness: Before we build the residual, we have a feasible flow – since we iniitially had a valid flow, the only possible violation was on the edge (u, v). We reduced its flow by 1, so it is feasible again.

Any flow that is feasible for the altered graph would have been feasible for the original, so the maximum flow for the new graph cannot be larger than the original. Thus adding one unit of flow is sufficient to guarantee we have the maximum. For integral capacities, we always increase by at least one unit of flow, so the single iteration is sufficient. (And if the altered version already is a maximum flow, the single iteration will confirm that by not finding an s-t path in the residual).

9. Currency Conversion

See 7 in original solutions, and replace "Afghanis" in the problem with "Francs."