1

Algorithm to check if a given flow f is a maximum flow in flow network G:

Compute the residual of f. This operates on each edge a constant number of times and takes $O(m)$.

Perform a DFS on the residual starting from s and return true if it reaches t and false otherwise. This takes $O(n+m)$ time.

The whole algorithm will take $O(n+m)$ time.

2

Prove that a flow f' exists such that $|f| = |f'|$ and $f'(e) \leq f(e) - 1$ given e is in a directed cycle in G'.

Since there is a cycle in G' such that e is in the cycle, and G' is defined as containing only edges which have a positive flow, e is a member of a cycle C in the flow f where every edge is positive.

Let F be the smallest flow weight in C. F will be at least 1 because the edge weights are positive integers.

Define f' as $f'(e) = \begin{cases} f(e) - F & e \in C \\ f(e) & e \notin C \end{cases}$

In other words, cancel a flow cycle at C from f.

Since canceling a cycle does not change the value of the (s,t)-flow, and $f'(e) = f(e) - F \leq f(e) - 1$, f' satisfies the requirements.

3

Given a flow f, network graph G, and edge $e \in G$, give an algorithm to find the maximum flow in F with $c_F(e) = c_G(e) - 1$, with all else the same between F and G.

First, note that in the case where $f(e) < c(e)$, the solution is trivial, since there is no need to change the flow.

In the case that $f(e) = c(e)$, we must look for a new flow. Cancel 1 unit of flow off a path from s to t going through e. To do this, label u and v such that e is the edge from u to v. Find the path with the fewest edges from s to u using a breadth-first search. Find the shortest path from v to t similarly. Do these BFS searches such that an edge is only followed when it has positive flow. We also know that these two paths exist because the > 1 unit of flow going through e has to come from and go to somewhere. Subtract one flow unit from each of these edges. This new flow is of value at least $|f| - 1$.

Since a single path augmentation is guaranteed to increase the flow by some positive integer value, if the flow is not already maximum, by augmenting this new path, we are guaranteed to get a flow of value $|f|$, or if none exists, a flow value of $|f| - 1$, which is then guaranteed to be a maximum flow.

The 2 breadth-first searches take $O(n+m)$ time each, and the augmentation procedure takes $O(m)$, so the algorithm runs in $O(n+m)$ time.

Credit for algorithm from https://cs.stackexchange.com/questions/86801/how-to-find-max-flow-in-a-graph-after-decrementing-an-edge-capacity