CSC373— Algorithm Design, Analysis, and Complexity — Spring 2016

Solutions for Tutorial Exercise 4: Network Flow

---

1. **Bookkeeping for the Residual Graph.** Suppose $G = (V, E, c)$ is the directed graph for an s-t flow problem. that is, there are two identified vertices $s, t \in V$, and the edge capacities $c(e)$ are strictly positive integers. Suppose f is a feasible integer-valued flow on G.

   The lecture notes describe how the residual graph G_f can be computed from G and f. Specifically, for each unsaturated edge $e = (u, v) \in E$, that is, with $f(e) < c(e)$, that edge e is in G_f with capacity $c_f(e) = c(e) - f(e)$. In addition, if $f(e) > 0$ then we also added an edge $e^R = (v, u)$ to the graph with $c_f(e^R) = f(e)$.

   This process is convenient for introducing max-flow, but it is a bit awkward for implementing it. For example, if both forward and backward edges between u and v are in E, then the residual graph G_f can include these two edges along with their reverses (for a total of four edges between u and v). Also, the existence or absence of an edge depending on whether the residual capacity is larger than zero or exactly zero is also awkward during the implementation. We'd prefer to simply note that a particular edge (u, v) (in that direction) has zero for its residual capacity rather than deleting it from our data structure of edges. If we did this we need to remember, during the search for augmenting paths, that we need to avoid paths which traverse an edge having capacity zero.

   As an alternative representation, suppose the original graph G is such that $e \in E$ if and only if the reverse edge e^R is also in E. The capacities of these two edges can be represented by the pair $\vec{p}(e) = (p_1(e), p_2(e)) = (c(e^R), c(e))$. The pair $\vec{p}(e^R)$ is then just $\vec{p}(e^R) = (p_2(e), p_1(e))$, which is $\vec{p}(e)$ with its two elements swapped. If the only the forward direction edge e is to have positive capacity then $\vec{p}(e) = (0, c(e))$. Similarly, if both the edge e and e^R have positive capacities in G, then both elements of $\vec{p}(e)$ are positive.

   Given a feasble flow f on G, show how to compute the residual capacities $\vec{p}_f(e) = (p_{f,1}(e), p_{f,2}(e))$ using this type of representation. That is, for each edge $e = (u, v) \in E$, the maximum flow in the residual graph G_f from u to v along this edge should $p_{f,2}(e) \geq 0$, and the residual capacity in the reverse direction along e^R is $p_{f,1}(e) \geq 0$. Note, these capacities should allow for exactly the same flow between u and v as in the residual graph described in the lecture notes. And, moreover, no edges need to be added or deleted from the data structure during the Ford-Fulkerson algorithm.

2. **Updates to a Max Flow Problem.** Suppose $G = (V, E, c)$ is a standard network flow problem where all the capacities $c(e)$ are positive integers, and f is an integer-valued maximum flow. Suppose $e_0 \in E$ is a saturated edge (i.e., $f(e_0) = c(e_0)$).

   (a) **Increase the capacity on e_0 by one.** Consider $G' = (V, E, c')$ where $c'(e) = c(e)$ for all $e \in E \backslash \{e_0\}$, and $c'(e_0) = c(e_0) + 1$. Given f (as above) and the residual graph G_f, describe an efficient algorithm for updating the maximum flow f to form a maximum flow f' for G'. What is the run-time for your update algorithm? Explain why the updated flow f' is maximal for G'.

   (b) **Decrease the capacity on e_0 by one.** Consider $G' = (V, E, c')$ where $c'(e) = c(e)$ for all $e \in E \backslash \{e_0\}$, and $c'(e_0) = c(e_0) - 1$. Given f and the residual graph G_f, describe an efficient algorithm for updating f to form a **feasible flow** f' for G' with value $v(f') = v(f) - 1$. Note f is not a feasible flow on G' since it $f(e_0) = c'(e_0) + 1$. What is the run-time for your update algorithm?

   (c) **Value of the Max-flow in case (b).** For G' as in case (b), what are the possible values of the maximal flow? Explain.

   (d) **Max-flow in case (b).** Describe an efficient algorithm for updating the feasible flow f' (constructed in part (b)) to form a **maximal flow** f^* for G'.

3. **Have the Maple Leafs been elimated from the playoffs yet?** We wish to build an **MLEY** app which identifies, as soon as possible, whether the Maple Leafs have been eliminated from the NHL playoffs. The app will have access to the current number of points for the 16 hockey teams in the Eastern Conference (including the Leafs), and the list of all games yet to be played this season between NHL teams. Specifically, you are given both $GR(j, k)$, the number of games remaining between teams j and k, and also $P(k)$, the

total number of points team k has to date (for any $1 \leq j, k \leq N$ with $j \neq k$). Suppose the Leafs are denoted by team $k = 16$ (which was their position at 3pm, Feb. 10, 2016), the Eastern Conference teams are for $k = 1, \ldots 16$ and the Western Conference teams have $k > 16$. Both arrays P and GR will be updated regularly as the season progresses.

One complication in working out the points any team can get is that, in the NHL, the losing team can get either 0 points or 1 point (the latter if the game is tied after the regulation time and that team then loses an overtime period or shoot-out), while the winning team always gets 2 points.

As a key member of the MLEY app team you are given the following task. Suppose you are given a subset of exactly seven top Eastern Conference teams that are assumed to make the playoffs, and your job is then to determine whether or not there is a way for the Leafs to beat or tie the other 8 (aka "bottom") Eastern Conference teams. That is, at the end of the season, can the total number of points for the Leafs be greater than or equal to the total points for each of the bottom eight teams? Specifically, you need to decide if there exist a suitable assignment of points for wins, losses, and overtime losses for all the remaining games in such a way that the Leafs end up with at least as many points at the end of the season as each of 8 bottom teams. Given all seven "top" teams that you were given do in fact make the play-offs, then there exists such an assignment for all the remaining games if and only if the Leafs have not been mathematically eliminated yet.

Your first thought is whether it is possible to use standard integer-valued max-flow to solve this problem (i.e., the first type of problem we considered in the lecture notes on Network Flow). Is it possible to answer this problem using an interger-valued network flow? If so, how? If not, why not?

Hint: Note that you are looking for the best case for the Leafs and the worst case for the (other) bottom 8 teams. The question can be reduced to whether or not you can assign either wins (2 points) and losses (0 points), only, on each of the remaining games between the bottom 8 teams in the Western Conference in such a way that the total number of points for these teams will not exceed the maximum number of points the Leafs can get. If you can and the assumption about the top seven teams is correct, then the Leafs have not been eliminated yet.

Hint: To pose this as an integer-valued flow problem, try using one unit of flow per game, with a win adding a single unit of flow from s through a vertex which represents the team that wins that game. To do this you will find you need to divide the current number of points $P(k)$ by 2 (since a win counts as two points in the NHL, not just one as you have here). This division may lead to capacities on edges which are fractional, that is, integer multiples of 1/2. Is there then an equivalent problem with only integer valued capacities?