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 altermative 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.

Solution for Q1: Suppose $e = (u, v) \in E$, with flow $f(e)$ and the capacity $c(e)$, and $f(e^R)$ and $c(e^R)$ are the flow and capacity on the reverse edge $e^R = (v, u)$ (without loss of generality, we can assume only one of $f(e)$ or $f(e^R)$ is positive, and the other is zero). Then $\vec{p}_f(e) = (c(e^R) - f(e^R) + f(e), c(e) - f(e) + f(e^R))$. That is, the residual capacity on the edge from u to v is $c(e) - f(e) + f(e^R)$, which is equal to the residual capacity $c(e) - f(e)$ for the forward directed edge in the original graph plus $f(e^R)$, which is the amount of flow that can be undone that is currently flowing from v to u along this edge e^R.

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'.

Solution for Q2:

2a Note f is an integer-valued feasible flow on the updated graph G' (since we only changed G by increasing one edge capacity). Let G'_f be the corresponding residual graph. Use breadth first search (BFS) to see if there exists an augmenting path in G'_f. Note that, since there is no such path in the residual graph G_f for the original problem (because f is maximal for that graph), there will be an augmenting path P **only if** it contains the edge $e_0 = (u, v)$ in the forward direction.

If there is no such path P then f is a maximal flow for G' (by the Lemma in the proof of the Max-Flow Min-Cut Theorem).

Otherwise, e_0 is on the path P. The residual capacity of e_0 is $c'_f(e_0) = c'(e_0) - f(e_0) = 1$. Note, since we used BFS, the path is a simple path and no edge appears twice or more on P. Therefore the bottleneck for P is one (it cannot be less than one since we have integer-valued flows and capacities). We can then push one more unit of flow on this augmenting path (note that, to know we are able to do this, we are using the fact that P is a simple path). Finally, this flow f' must be maximal because: a) we know the minimum cut capacity for the graph G is $C = v(f)$; b) $v(f') = v(f) + 1$; and c) the minimum cut capacity for the graph G' can be no bigger that $C + 1$ (because we have only added $+1$ to a single edge capacity). But with the above augmentation we have $v(f') = C + 1$, so this must be both the min cut capacity on G' and the maximum possible flow value.

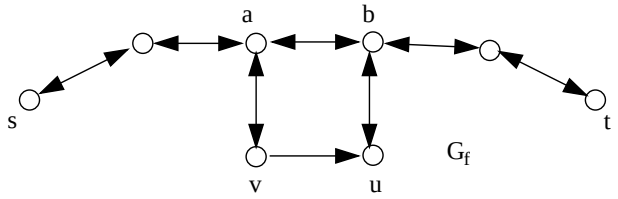
The runtime for this algorithm is dominated by the BFS step, which runs in $O(|E|)$ time. The flow agumentation runs in $O(|V|)$ time.

Alternative approach (significantly harder):

Let $e_0 = (u, v)$. Define $m(G_f)$ to be the modified residual graph which has any edge from u to v or vice versa removed (using $m(G_f)$ is convenient to constrain the paths that the subsequent breadth first search (BFS) finds). Use BFS to see if there exists a path P from s to u in $m(G_f)$. And also use BFS to see if there exists a path Q from v to t in $m(G_f)$. In both cases, if such a path exists, BFS will produce a simple path.

If either sub-path P or Q does not exist, then there is no augmenting path through the edge e_0 in G'_f. Note that there also cannot be any s-t path in G_f or G'_f that **does not** cross the edge e_0, since $c'(e_0)$ is the only edge capacity to change, and f is a maximal flow in G. These two cases of s-t paths (i.e., including or not including the edge e_0) are exhaustive, and we can therefore conclude that the non-existence of P or Q implies there are no augmenting paths in G'_f. In this case the flow f must be maximal.

Otherwise, both sub-paths exist in G_f (and G'_f), say $P = (s..u)$ and $Q = (v..t)$. Since they were generated by BFS they are the shortest such paths (in terms of number of edges). Moreover, by construction, neither contains an edge from u to v nor from v to u (thanks to the use of $m(G_f)$). We then consider the concatenated path $R = P|(u, v)|Q$ which is a path from s to t in the residual graph G'_f. At this point you should be asking yourself what can go wrong?



In order to push one unit of flow along R we will need to argue that no edge appears more than once (in the same direction) on R (i.e., that R is simple). By construction, the only way for such an edge, say (a, b), to appear more than once on R, is for it to appear once on P and once on Q. An example is shown above, where the arrows indicate edges with strictly positive residual capacity in G_f. In the modified graph $m(G_f)$ there is a unique simple path from s to u and a unique simple path from v to t, but both contain the edge (a, b). (In such a case, the path R constructed above is not simple. If the residual capacities of edge (a, b) is 1 then we cannot push one more unit of flow along the whole path R without exceeding the capacity of the duplicated edge (a, b).)

In a proof following this particular strategy you should notice this issue. (If you took this route in your proof, did you notice it?)

Fortunately, we can prove having a duplicated edge on R is impossible by using contradiction. Suppose the edge (a, b) appears once in P and once in Q. Then there is a subpath P' from s to a in P and a sub-path Q' from b to t in Q, neither of which contains e_0, or the edge (a, b). Then $R' = P'|(a, b)|Q'$ is a s-t path in the original residual graph G_f. Even though R' itself might not be simple, it follows that t is reachable from s in in the residual graph G_f. But this contradicts f being a maximal flow. Therefore, the R constructed above must be a simple path in G'_f.

We next show that the bottleneck rate on R in G'_f is one. Note the minimum residual capacity on all edges in P and Q must be at least 1, by the definition of the residual graph. In addition, in G' we have $c'(e_0) = c(e_0) + 1 = f(e_0) + 1$. Therefore, this edge in the residual graph G'_f has residual capacity $c'_f(e_0) = c'(e_0) - f(e_0) = 1$. Therefore the bottleneck rate on R is one.

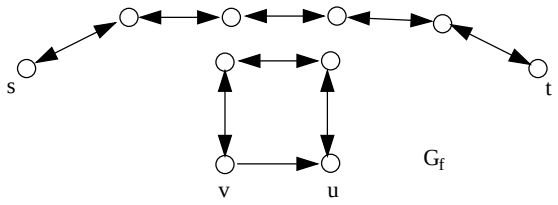
We can then set f' equal f augmented by a flow of $+1$ along this path R (here we have used the fact that R is simple).

The proof that f' is a maximal flow for G' is the same as given above.

The runtime for this is the same order as BFS, i.e., $O(|E|)$.

2b Let $e_0 = (u, v)$ and let $m(G_f)$ be the modified residual graph as in part (a). We seek a simple path P from u to s in $m(G_f)$. In addition, we seek a simple path Q in $m(G_f)$ that goes from t to v.

Do either of these paths exist? Not necessarily, see the example below where we have drawn a possible residual graph.



We are not going to address these special cases here. Instead we will simply assume that the simple paths P and Q described above both exist.

The idea now is to try to reduce the flow by one by effectively pushing one unit of flow back from t to s along the path $R = Q|(v, u)|P$ in the residual graph G_f. Using a similar argument to that in part (a), we can do this if the R is simple. Moreover, when R is simple, the resulting flow f' is feasible for G', and the resulting flow f' has value equal to $v(f) - 1$. Note that I'm **not** claiming this flow f' is optimal.

First, let's consider what happens when the path R is not simple. Similar to part (a), the only way the path R may not be simple is for at least one edge (b, a), say, to appear in both Q and P. (This situation is sketched in the figure in part (a).) If such duplicate edges exists, then it can be shown that there is a simple cycle from v to u and back to v on the path R. By pushing a value of 1 along this simple cycle in the

residual graph G_f we find a feasible flow f'' of G'. But note this feasible flow still has value $v(f'') = v(f)$. Finally, in order to get a feasible flow of value $v(f) - 1$, as requested in question (b), we would also need to subtract one from f on a simple t to s path in the residual graph G_f. Such a path can be found by doing BFS from t on the residual graph for flow f'' (assuming $v(f'') > 0$). We omit the details.

The runtime is the same order as BFS, $O(|E|)$.

2c If the edge e_0 is in any minimal s-t cut (A, B) of G, then the capacity of this cut in G' is one less than in G. Otherwise the minimum cut capacity of G' is the same as for G.

In the first case, since $v(f)$ was a maximal flow, the min capacity cut of G has capacity $v(f)$, and so the min capacity cut of G' must be $v(f) - 1$ and the flow must be maximal.

Otherwise, the min cut capacity of G' is still $v(f)$.

2d Do BFS on the residual graph for G' formed for the flow f' constructed in part (b). If there is an augmenting path, then the bottleneck rate must be one (integer valued flow and the cut capacity is no smaller than $v(f')+1$. Define f^* to be the result of pushing one more unit of flow along this augmenting path. Othewise, if there doesn't exist such an augmenting path, define f^* to be f'. In this case f^* must be maximal (by lemma in the the proof of the max-flow min cut theorem).

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.

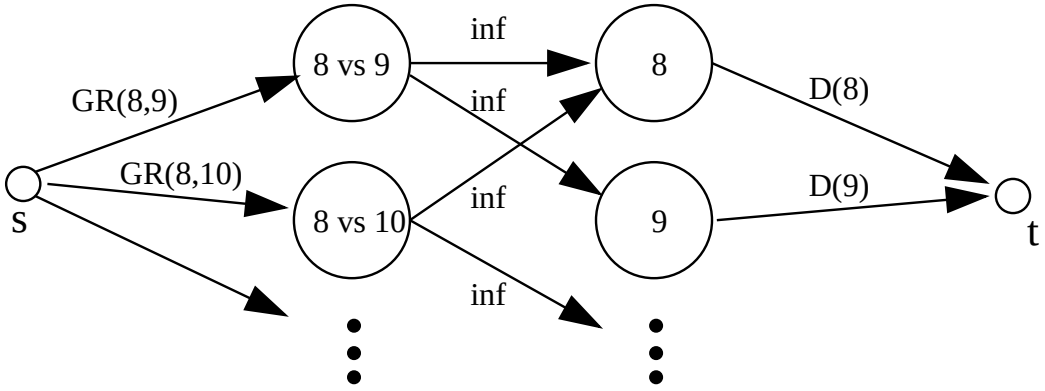
   Your first thought is whether it is possible to use standard integer-valued max-flow to solve this problem (i.e., the first type of problem we considered in the lecture notes on Network Flow). Is it possible to answer this problem using an interger-valued network flow? If so, how? If not, why not?

   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?

MLEY Solution. WLOG assume the top seven teams in the Eastern conference have $k = 1, 2, \ldots, 7$ and the bottom eight teams have $k = 8, 9, \ldots, 15$ (with the Leafs at $k = 16$).

Consider the graph depicted in the figure below. For each pair of bottom eight teams that have yet to play a game, say teams $i, j \in \{8, 9, \ldots 15\}$ with $i \neq j$, connect the source vertex to a vertex representing "i versus j", and let the capacity of this edge be $GR(i, j)$, that is, the number of games remaining between these teams. Connect these "i versus j" vertices to vertices representing the two teams i and j, using edges with infinite capacities (or any integer-valued capacity greater than or equal to $GR(i, j)$ will do). Connect each of these team vertices to the sink vertex t with an edge (k, t) of capacity $D(k)$. We describe these capacities $D(k)$ further below.



Consider an integer valued feasible flow in the above graph. Each unit of flow passing from s through the vertex j vs k then through the vertex for team k and finally to the sink represents one future game played between teams j and k, with team k winning, collecting 2 points (corresponding to one unit of flow), and with team j collecting 0 points. We describe why we do not consider team j gaining one point in lossing this game (e.g., in overtime) further below.

The capacity $D(k)$ on the edge from the team k vertex to the sink should be half the maximum number of additional points team k can obtain and still end up tied or behind the Leafs. Specifically, we define

$$D'(k) = [\text{Max\_Points(Leafs)} - \text{Current\_Points}(k)]/2 \quad (1)$$

$$D(k) = \text{floor}(D'(k)) \quad (2)$$

In equation (1), the term Max_Points(Leafs) denotes the maximum number of points the Leafs can obtain. This is the sum of their current points, plus 2 points for winning each of their remaining games. The second term, Current_Points(k) for $k = 8, \ldots, 15$, is team k's current point total. Here we are assuming team k loses all its remaining games against Western Conference teams, against the Leafs, and also all its games against the the top seven teams in the Eastern Conference (and, moreover, all these losses are within regulation time, so there are no single point games). These are the best possible outcomes for the Leafs in these games (i.e., we are asking if the Leafs are eliminated even in the best possible case for them).

Moreover, we seek a way to assign wins and losses in the remaining games between teams j and k with $j, k \in \{8, \ldots, 15\}$ so that the Leafs might end up tied for eighth spot (or better). In these remaining games, the best case for the Leafs is that there are no ties, that is, a team either wins and gets two

points, or loses and gets zero points. The total of the 2 points for all these wins can be no larger than [Max_Points(Leafs) − Current_Points(k)], otherwise the k^{th} team ends up with more points than the Leafs.

The division by 2 in $D'(k)$ above corresponds to the scaling between having a win denoted by one unit of flow on a s-t path in the above graph, and the fact that a team gets two points for a win.

We are interested in integer-valued flows, where each unit of flow represents individual games, but the division by two means $D'(k)$ might be not be an integer. This would be an issue if we attempted to use $D'(k)$ as an edge capacity, and also tried to apply the Ford-Fulkerson algorithm considered in class. But note that, given we are considerin integer-valued flows, we can set the capacity of the k-t edge as the largest integer less than or equal to $D'(k)$ (see equation (2)), since this is necessary for the Leafs to remain (at least) tied for eighth.

If $D(k) < 0$ for any of these k's, then the Leafs have been eliminated already. Otherwise, if $D(k) = 0$ for any k we can drop the k-t edge from the above graph (indicating that team k cannot win another game).

The claim is then that, given that teams 1 through 7 actually do make the playoffs, it is still possible for the Leafs to be tied for eighth spot if and only if there exists an integer-valued flow for which all the edges leaving s saturated. That is, the value of the max flow must equal to the sum of the capacities of all the edges leaving s.

The reason this must be the case is that such a flow corresponds to a way that all the remaining games can turn out and still have the Leafs at least tied for eighth. The only alternative is that such an assignment of wins and losses in the remaining games is impossible and, in this case, the max (integer-valued) flow will not saturate all the edges leaving s. That is, all the remaining games cannot be played without at least one team k, for $k \in \{8, 9, \ldots, 15\}$, beating the Leafs in overall points.