CSCI 3104 Fall 2017 Problem Set 9
Austin Griffith (11/18)

---

1. *(20 pts) Given a directed graph G = (V,E) with capacity c(u,v) ¿ 0 for each edge (u,v) E and demand r(v) at each vertex v V, a routing of flow is a function f such (......) that i.e., the total incoming flow minus the total outgoing flow at vertex v is equal to r(v). Notice that the demand r(v) can take positive value, negative value, or zero.*

   (a) *Show how to find a routing or determine that one does not exist by reducing to a maximum flow problem.*

   To show reduced this problem into a maximum flow problem the following steps will be needed.
   Step 1 would be to connect a source vertex to all vertices in G in which incoming flow is less than outgoing flow i.e r(v) < 0

   Step 2 would be to make the flow of these edges connecting the source to r(v) < 0 connection == absolute value(demand of v)
   This ensures that the demand at each vertex v r(v) = 0.

   Step 3 would be to connect a destination vertex to all vertices in G in which incoming flow is greater than outgoing flow i.e r(v) > 0 while taking in to consideration the increased flow from our source vertex.

   Step 4 would be to make the capacity of these edges connecting destination to r(v) > 0 connection == absolute value(demand of v). One additional vertex (source) will have edges with flow r(v) < 0 and capacity == |r(v)| and another vertex (sink) will have edges with flow r(v) > 0 and again capacity == |r(v)|. By setting such capacity's all edges will have capacity equal to absolute value of |r(v)|, and only the sign of the flow will change.

   Step 5: We have now reduced our original problem and graph to a max flow problem and now run Run Ford-Fulkerson algorithm from the new source to new destination.
   The residual graph determines the incoming flow at each vertex in G and sum the values. This can be accomplished by determining the sum of the arrows pointing away from the node. We have now found sum(f(u,v)) to determine the rest of routing function we take the incoming flow from a vertex and subtract the demand r(v) leaving us sum(f(v,u))

b*) Suppose that additionally there is a lower bound l(u, v) > 0 at each edge (u, v), and we are looking for a routing f satisfying f(u, v) l(u, v) for all (u, v) E. Show how to find such a routing or determine that one does not exist by reducing to a maximum flow problem.*

For this problem if we want to make sure our incoming edges are greater than some given lower bound l(u, v) > 0 at each edge (u, v), we need to modify our Ford Fulkerson algorithm to account for it. To do so you will need to add l(u,v) to r(v) and subtract it from r(u). With that we can assume that the residual graph will contain the incoming flow at the vertices's and we can continue on as we did in the first part of the problem.

2. *(20 pts extra credit) Even though in this class we focus on those greedy algorithms that generate optimal solutions, in general a greedy algorithm may not give an optimal solution. So, we are interested in those greedy algorithms that generate a good enough solution, i.e., not too far from the optimal solution. Let us consider one such problem as follows.*

   Given subsets S1,S2, ,Sn of a set S of points and an integer m, a maximum mcover is a collection of m of the subsets that covers the maximum number of points of S. Finding a maximum mcover is a computationally hard problem. Give a greedy algorithm that achieves approximation ratio 11/e; i.e., let y be the maximum number of points that can be covered by m subsets and x be the number of points that are covered by the m subsets generated by your algorithm, then give a greedy algorithm such that x (1 1/e)y.

Greedy Approach: choose a subset with the largest amount of points covered and add more sets until every point is covered. This may not be the most optimal solution hence the greedy choice property is not choosing the most optimal but as in this problem, a "good enough" solution. We then use the fact that for m choices the points left uncovered are <= x(1-1/m)^m and thus x >= 1-(1-1/m)y >= (1-1/e)y .