CSCI 570 - Fall 2015 - HW 9

1. At a dinner party, there are n families $a_1, a_2, ..., a_n$ and m tables $b_1, b_2, ..., b_m$. The i^{th} family a_i has g_i members and the j^{th} table b_j has h_j seats. Everyone is interested in making new friends and the dinner party planner wants to seat people such that no two members of the same family are seated in the same table. Design an algorithm that decides if there exists a seating assignment such that everyone is seated and no two members of the same family are seated at the same table.

   Construct the following network G=(V,E). For every family introduce a vertex and for every table introduce a vertex. Let a_i denote the vertex corresponding to the i^{th} family and let b_j denote the vertex corresponding to the j^{th} table. From every family vertex a_i to every table vertex b_j, add an edge (a_i,b_j) of capacity 1. Add two more vertices s and t. To every family vertex a_i add an edge (s,a_i) of capacity g_i. From every table vertex b_j add an edge (b_j,t) of capacity h_j.

   Claim: There exists a valid seating if and only if the value of max flow from s to t in the above network equals $g_1 + g_2 + ... + g_n$.

   Proof of Claim: Assume there exists a valid seating, that is a seating where every one is seated and no two members in a family are seated at a table. We construct a flow f to the network as follows. If a member of the i^{th} family is seated at the j^{th} table in the seating assignment, then assign a flow of 1 to the edge (a_i,b_j). Else assign a flow of 0 to the edge (a_i,b_j). The edge (s,a_i) is assigned a flow equaling the number of members in the i^{th} family that are seated (which since the seating is valid equals g_i). Likewise the edge (b_j,t) is assigned a flow equaling the number of seats taken in the table b_j (which since the seating is valid is at most h_j). Clearly the assignment is valid since by construction the capacity and conservation constrains are satisfied. Further, the value of the flow equals $g_1 + g_2 + ... + g_n$.

   Conversely, assume that the value of the max s-t flow equals g_1+g_2+...+g_n. Since the capacities are integers, by the correctness of the Ford-Fulkerson algorithm, there exists a maxflow (call f) such that the flow assigned to every edge is an integer. In particular, every edge between the family vertices and table vertices has a flow of either 0 or 1 (since these edges are

of capacity 1). Construct a seating assignment as follows, seat a person of the i^{th} family at the j^{th} table if and only if $f(a_i, b_j)$ is 1. By construction at most one member of a family is seated at a table. Since the value of f equals the capacity of the cut $(s, V - s)$, every edge out of s is saturated. Thus by flow conservation at a_i, for every a_i the number of edges out of a_i with a flow of 1 is g_i. Thus in the seating assignment, every one is seated. Further, since the flow $f(b_j, t)$ out of b_j is at most h_j, at most h_j persons are seated at table b_j. Thus we have a valid seating.

2. The edge connectivity of an undirected graph is the minimum number of edges whose removal disconnects the graph. Describe an algorithm to compute the edge connectivity of an undirected graph with n vertices and m edges in $\mathcal{O}(m^2 n)$ time.
For a cut $(S, \bar{S})$, let $c(S, \bar{S})$ denote the number of edges crossing the cut. By definition, the edge connectivity

$$k = \min_{S \subset V} c(S, \bar{S})$$

Fix a vertex $u \in V$. For every cut $(S, \bar{S})$, there is a vertex $v \in V$ such that u and v are on either side of the cut. Let $C_{u,v}$ denote the value of the min u-v cut. Thus

$$k = \min_{v \in V, v \neq u} C_{u,v}$$

For each $v \neq u$, $C_{u,v}$ can be determined by computing the max flow from u to v. Since G is undirected, we need to implement an undirected variant of the max flow algorithm. Set all edge capacities to 1. During each step of the flow computation, search for an undirected augmenting path using breadth first search and send a flow of 1 through this path.

There are $n - 1$ flow computations and each flow computation takes at most $\mathcal{O}(m^2)$ time.

3. Problem 7 from Chapter 7.
Let s be a source vertex and t a sink vertex. Introduce a vertex for every base station and introduce a vertex for every client.

For every base station vertex b, add an edge (s, b) of capacity L. For every client vertex v, add an edge (v, t) of unit capacity. For every base station vertex b, add a unit capacity edge from b to every client c within its range.

We claim that every client can be connected to a base station subject to load and range conditions if and only if the max flow of the above network is at least n.

If every client can be connected to a base station subject to load and range conditions, then if a client c is served by base station b, assign a

unit flow to the edge (b, c). Assign the the flows to the edges leaving the source (respectively the edges leaving the sink) so that the conservation constraints are satisfied at every base station (respectively client) vertex is satisfied. (note that such an assignment is unique). Further, since every client can be connected to a base station subject to load conditions, the capacity constraints at the edges leaving the source are also satisfied. Hence we are left with a valid flow assignment for the network. Since every client is connected, the flow entering the sink is at least the number of clients n (in fact, it is exactly n).

Conversely, if the max flow of the network is at least n then there exists a integer flow of flow value at least n (since all capacities are integral). We will work with this integral flow. For every edge (b, c) with a flow of 1, assign c to the base station b. Since a client vertex can at most contribute one unit of flow entering t, and the total flow entering t is at least n, every client vertex has flow entering it. This implies that every client is serviced by exactly one station. Since the flow entering a base station vertex b is at most L, a base station is assigned to at most L clients. Thus every client is connected to a base station subject to load and range conditions.

Hence our claim is true and all that is left to do is to compute the max-flow of the network using Ford-Fulkerson and to output YES if and only if the max-flow value is at least n.

4. Problem 9 from Chapter 7.

   This problem is virtually identical to the previous problem. Let s be a source vertex and t a sink vertex. Introduce a vertex for every hospital and introduce a vertex for every injured-person.

   For every hospital vertex h, add an edge (s, h) of capacity $\lceil n/k \rceil$. For every injured-person vertex c, add an edge (c, t) of unit capacity. For every hospital vertex h, add a unit capacity edge from h to every injured-person vertex c within half-hour's driving.

   We claim that every injured person can be admitted to a hospital (within 1/2 hour driving) such that the load on the hospitals are balanced if and only if the max flow of the above network is at least n. The proof is identical to the previous problem.

5. A vertex cover of an undirected graph $G = (V, E)$ is a subset of the vertices $A \subseteq V$ such that for every edge $e \in E$, at least one of its incident vertices is in A (that is every edge has at least one of its ends in A). Design an algorithm that takes a bipartite graph G' and a positive integer k, and decides if G' has a vertex cover of size at most k.

   Partition the vertices of the graph into left vertices and right vertices,

that is $V = L \cup R$, $L \cap R = \emptyset$ and every edge has one end in L and the other in R.

We construct a network $\bar{G}'$ as follows. Add a source s, a sink t and the following edges ; (i) directed edges of capacity 1 from s to every vertex in L , (ii) directed edges of capacity 1 from every vertex in R to t and (iii) make every edge in G' directed from L to R and of infinite capacity.

Let $(S, \bar{S})$ be a cut of $\bar{G}'$ of finite capacity. Without loss of generality assume that $s \in S$. We claim that we can construct a vertex cover of G' of size equal to the capacity of the cut.

Define $A := (L - S) \cup (R \cap S)$.

Edges that are incident on $L - S$ are covered by $L - S$. The only edges left to check are the ones with an end in $L \cap S$. The edges that are incident on $L \cap S$ have their other end in S and are hence covered by $R \cap S$. Thus A is indeed a vertex cover.

Edges between L and R have infinite capacity and hence cannot be in the cut. The only edges crossing the cut are either incident on s or incident on t. The number of edges of the form (s, u) where $u \in \bar{S}$ equals $|L - S|$. The number of edges of the form (v, t) where $v \in S$ is $R \cap S$. Hence the size of A equals the capacity of the cut $(S, \bar{S})$ and our claim is true.

We now claim the converse. That is, given a vertex cover A of G', we can construct a cut of $\bar{G}'$ of capacity equal to the size of A.

Define the cut $S := \{s\} \cup (L - A) \cup (R \cap A)$.

For an edge $(u, v) \in E$, if $u \in S$ then $v \in S$ (since A is a vertex cover). Thus edges in E cannot cross the cut.

The number of edges of the form (s, u) where $u \in \bar{S}$ equals $|L \cap A|$. The number of edges of the form (v, t) where $v \in S$ equals $|R \cap A|$. Hence the capacity of the cut is $|L \cap A| + |R \cap A|$ which is exactly the size of A. Hence, the converse follows.

We have thus proven that the size of the min-vertex cover of G' equals the capacity of the min-cut of $\bar{G}'$. The capacity of the min-cut of $\bar{G}'$ can be computed using the Ford-Fulkerson algorithm (say the capacity is m). If $k < m$, output NO, otherwise output YES.