CSCI570 - Fall 2019 - HW9

Due Nov 6th 11:59pm

October 31, 2019

1 Graded Questions

1.1 Question 1

At a dinner party, there are n families $a_1, \ldots, a_n$ and m tables $b_1, \ldots, b_m$. 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.

Solution 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. For every family vertex a_i add an edge (s, a_i) of capacity g_i. For 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 $\sum_{1 \leq i \leq n} g_i$.

Proof. 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 $g_1 + g_2 + \cdots + g_n$. □

Conversely, assume that the value of the max $s-t$ flow equals $g_1+g_2+\cdots+g_n$. Since the capacities are integers, by the correctness of the Ford- Fulkerson algorithm, there exists a max flow (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, so at most h_j people are seated at table b_j. Thus we have a valid seating.

1.2 Question 2

There is a precious diamond that is on display in a museum at m disjoint time intervals. There are n security guards who can be deployed to protect the precious diamond. Each guard has a list of intervals for which he/she is available to be deployed. Each guard can be deployed to at most A time slots and has to be deployed to at least B time slots. Design an algorithm that decides if there is a deployment of guards to intervals such that each interval has either exactly one or exactly two guards deployed.

Solution We create a circulation network as follows. For the i-th guard, introduce a vertex g_i and for the j-th time interval, introduce a vertex t_j. If the i-th guard is available for the j-th interval, then introduce an edge from g_i to t_j of capacity 1. Add a source s and a sink t. To every guard vertex add an edge from s of capacity A and lower bound B. From every interval vertex add an edge to t of capacity 2 and lower bound 1. Add an edge from t to s of infinite capacity. We claim that there exists a valid deployment if and only if the above network has a valid circulation. The proof of the claim is virtually identical to the proof in section 7.8 of the text for the survey design problem. The algorithm proceeds by determining if the network has a circulation (by reducing it to a flow problem and then applying Ford-Fulkerson) and answers "yes" if and only if there is a circulation. The number of vertices and number of edges in the resulting flow problem are bounded by $O(n)$ and $O(n^2)$ respectively. The running time of our algorithm is dominated by the flow computation which takes $O(n^3)$.

1.3 Question 3

An edge of a flow network G is called critical if decreasing the capacity of this edge results in a decrease in the maximum flow. Is it true that with respect to a maximum flow of G, any edge whose flow is equal to its capacity is a critical edge? Give an efficient algorithm that finds a critical edge in a flow network.

Solution Not true. Consider a flow graph $G = (V, E)$ such that $V = \{a, b, s, t\}$, $E = \{(s, a), (a, b), (a, t), (b, t)\}$ and the capacity of each edge is 1. The maximum flow is 1 and an alternative path from a to b can be found if one path gets blocked.

For a flow network, for a max-flow f, we can find a cut (A, B) such that $value(f) = cap(A, B)$. Assume that A contains source and B contains target. An edge e_c from A to B can be found such that $f(e_c) = c(e_c)$ in f, then e_c is a critical edge.

Proof. Since $cap(A, B) = \sum_{e_i out of A} c(e_i)$, and $e_c \in \{e_i\}$, reducing $c(e_c)$ will then reduce the capacity of cut (A, B) from $cap(A, B) to cap'(A, B)$. Originally, $value(f) = cap(A, B)$. After the reduction, the modified maximumflow f' will satisfy this relation: $value(f') \leq cap'(A, B) < cap(A, B) = value(f)$. Thus e_c is a critical edge. □

The algorithm can be implemented as follows. After finding the maximum flow f, perform DFS from the source node on G_f , and label the reachable nodes as A. Label the rest of the nodes as B. Scan all the edges in G. If an edge goes from A to B, then that edge is an critical edge. DFS runs in $O(|V| + |E|)$, and scanning takes $O(|E|)$. So the final complexity is the same as that of finding the maximum flow f.

2 Practice Problems

2.1 Question 1

Solve Kleinberg & Tardos, Chapter 7, Exercise 28.

2.2 Question 2

Solve Kleinberg & Tardos, Chapter 7, Exercise 34.