CMPS 102 — Winter Quarter 2017 – Homework 3

Christopher Hsiao - chhsiao@ucsc.edu - 1398305

Solution to Problem 1 - Doing Dishes

Consider a set $S$of$n$ computer scientist roommates who would like to determine "who will do the dishes after dinner" for the next m nights. Not all roommates have dinner jat home every night, so let S_i denote the subset of roommates who will have dinner at home on the i^{th} night and let $w_i \leq |S_i|$ be the number of people needed to do the dishes on the i^{th} night (some nights multiple people are needed).

For each roommate $j \in S$ and night $1 \leq i \leq m$, let $Q_i(j) = 0$ if j does not eat at home on the i^{th} night, otherwise let $Q_i(j) = w_i/|S_i|$, i.e., if j eats at home on the i^{th} night. Now, for each roommate j, let

$$R_j = \sum_{i=1}^{m} Q_i(j). \tag{1}$$

Clearly, we can't hope that everyone does dishes exactly R_j nights since, in general, R_j won't even be an integer.

Nevertheless, the computer scientists claim:

There exists a dish-washing plan under which
each roommate $j \in S$ does the dishes at most $\lceil R_j \rceil$ times.

Prove the claim. That is, be a computer scientist.

Hint: Formulate as a flow problem between roommates and nights and use the Integrality Theorem.

We will form our graph as shown above. The source node s is connected to all n people, such that there exists an edge between s and all $\{p_1, p_2, p_3, ...p_n\}$

Solution to Problem 2 - Fussy Eaters

You are planning a dinner party for n friends who, naturally, have all sorts of dietary restrictions and allergies. Your plan is to cook some appetizer dishes and some main dishes and portion them so that you end up with n appetizer *portions* and n main *portions*. You email everyone, and for each person $i \in \{1, ..., n\}$ you get back their list A_i of OK-to-eat appetizer dishes and their list M_i of OK-to-eat main dishes.

Design an efficient algorithm that takes as input the number of portions of each dish and the lists A_i, M_i, designs a max flow instance, has it solved, and uses the (value of the) maximum flow to decide whether or not it is possible to give each friend an appetizer and a main portion that is OK for them to eat.

Appetizers

We build the following flow network. There is a node $p_i \in \{p_1, ..., p_n\}$ for each person i, and a node $a_j \in \{a_1, ..., a_n\}$ for each appetizer j. There is an edge (p_i, a_j) of capacity 1 if appetizer $a_j \in A_i$, where A_i represents the appetizers person i is okay with eating. We then connect a super-source s to every person node $p_i \in \{p_1, ..., p_n\}$ with an edge (s, p_i) of capacity 1. Then, we connect every appetizer node $a_j \in \{a_1, ..., a_n\}$ to a super-sink t with an edge (t, a_j) with capacity 1.

Solution to Problem 3 - Verbal Assault

You are an English teacher assigning presentations to students. Each of your n students will receive a topic and 31 fancy words that they must use in discussing the topic. You've announced the set of topics T and the set of fancy words F, and have received from each student $i \in \{1, ..., n\}$ a set $T_i \subseteq T$ of topics the student is interested in, and a set $F_i \subseteq F$ of fancy words that the student would like to use.

You want to see if it is possible to assign to each student a topic and *exactly* 31 words such that:

- Each topic is assigned to at most 2 students.
- Each fancy word w_i is assigned to at most t_i students.

Describe a method that takes as input the sets T_i and F_i and the number $t_1, ..., t_k$ and returns either "No", or an assignment of topics and words that meets the requirements.

Solution to Problem 4 - Mi-k Drop

Let $G = (V, E)$ be a directed graph with a source $s \in V$, a sink $t \in V$, and where every edge $e \in E$ has capacity exactly 1. Let k be he value of the maximum flow in G.

Question: Given an arbitrary integer $1 \leq q \leq k$, can you always remove q edges from G so that in the resulting graph G' the value of the maximum flow is $k - q$?

Yes.

Proof. By the MaxFlow-MinCut theorem, the maximum flow in a network is bounded by $|min\ cut|$, which means that if k is the maximum flow in G, then k is also $|min\ cut|$. Since $|min\ cut|$ acts as an upper bound on our flow, it stands that removing edges in G is a perfectly valid operation as long as the number of edges removed is bounded by $1 \leq q \leq k$. Since every edge in G has capacity 1, removing any edge reduces the maximum flow by 1. Repeat this process q times, and you have a max flow reduced by q, aka $k - q$. □