CS 473: Fundamental Algorithms, Fall 2014

Discussion 10

November 4 / 5, 2014

10.1. $k-$LIGHTEST EDGES.

Consider the following problem. You are given a flow network with unit-capacity edges: It consists of a directed graph $G = (V, E)$, a source $s \in V$, and a sink $t \in V$; and $c_e = 1$ for every $e \in E$. You are also given a parameter k.

The goal s to delete k edges so as to reduce the maximum $s - t$ flow in G by as much as possible. In other words, you should find the set of edges $F \subseteq E$ so that $|F| = k$ and the maximum $s - t$ flow in $G' = (V, E - F)$ is as small as possible subject to this. Give a polynomial time algorithm to solve this problem.

10.2. ROUNDING UP THE TIDE.

Suppose we are given an array $A[1..m][1..n]$ of non-negative real numbers such that each row and column sum is an integer. We want to round A to an integer matrix, replacing each entry x in A with either $\lceil x \rceil$ or $\lfloor x \rfloor$ while maintaining the sum of entries in any row or column of A. For example,

$$\begin{pmatrix} 1.2 & 3.4 & 2.4 \\ 3.9 & 4.0 & 2.1 \\ 7.9 & 1.6 & 0.5 \end{pmatrix} \text{ rounds to } \begin{pmatrix} 1 & 4 & 2 \\ 4 & 4 & 2 \\ 8 & 1 & 1 \end{pmatrix}$$

Describe an algorithm that either outputs a feasible rounding scheme or outputs that there isnt one.

10.3. DINNER SCHEDULING.

Consider a group of n people who are trying to figure out a dinner schedule over the next n nights where each person needs to cook exactly once. Everyone has scheduling conflicts with some of the nights, so deciding who should cook on which night becomes tricky.

Label the people $\{p_1, \ldots, p_n\}$ and the nights $\{d_1, \ldots, d_n\}$. For each person p_i, there's a set of nights $S_i \subset \{d_1, \ldots, d_n\}$ when they are *not* able to cook.

A *feasible dinner schedule* is an assignment of each person to a different night, so that each person cooks on exactly one night, there is someone cooking on each night, and if p_i cooks on night d_j, then $d_j \notin S_i$.

(A) Describe a bipartite graph G so that G has a perfect matching if and only if there is a feasible dinner schedule for the group. What is the running time of your algorithm in this case?

(B) After generating a schedule, they realize there is a problem. There are $n - 2$ of the people that are assigned to different nights on which they are available: no problem there. However, two people p_i and p_j have been assigned to cook on the same day d_l, while no one has been assigned to d_k. Show that it's possible to fix this bad assignment and get a good assignment faster than just computing a solution from scratch. Namely, decide in $O(n^2)$ time, given this bad solution, whether there exists a feasible dinner schedule. How does the running time of your algorithm compares to (A).

10.4. Applications of Min-cost Flow

Consider a flow network with **finite** integer capacities on the edges. You have to send k units of flow (k is an integer) from s to t in this network, and the twist is that there are costs associated with sending one unit of flow on edge. That is, for every edge e of G, there is a cost $w(e)$ associated with it. Formally, given a flow $f(\cdot)$ defined on the edges, the ***cost*** of this flow is $\sum_{e \in E(G)} w(e)f(e)$. The min-cost flow problem is the following: given flow network $G = (V, E)$, $s, t \in V$ and k find a minimum-cost flow from s to t of k units. This problem can be solved efficiently (that is, in polynomial time). Moreover, one can show that if capacities are integral then there exists an optimum solution where the flow is integral. See how each of the following problems can be solved via this algorithm.

(A) Given a directed graph with positive integer costs on the edges and two vertices s and t in the graph, describe how to compute the k edge disjoint paths from s to t, such that the total cost of these paths is minimized.

(B) You are given a bipartite graph G with n vertices and m edges. Every edge has a cost which is a positive integer number. Describe an algorithm that decides if this graph has a perfect matching, and if so, outputs the cheapest such perfect matching.

(C) Banana just released a new version of their iFifi – the first electronic gizmo that not only can surf the web, but it is also dishwasher safe (not to mention that it comes in two colors: black and blacker). Banana has k distribution centers $C_1, \ldots, C_k$ in the US, and you know for each one of them how many iFifi they currently have in stock (i.e., $t_1, \ldots, t_k$). You need to plan the distribution of the iFifis to the Banana stores. You have a list of n stores $S_1, \ldots, S_n$, and for each one of them there is a quota f_i of how many iFifis they need. For every distribution center C_i and a store S_j, you know the distance d_{ij} between them in miles (rounded up so it is an integer).
Sending a single iFifi from a distribution center C_i to a store S_i costs $10^{-4} d_{ij}$ dollars. Describe an algorithm, as efficient as possible, that computes the minimum cost way to send all the required iFifis from the distribution centers to the stores. How fast is your algorithm (for this question you can assume the US diameter is 3000 miles).