OLD CS 473: Fundamental Algorithms, Spring 2015

Discussion 11

April 9, 2015

11.1. 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?

(Note, that several people might be cooking on the same night.)

(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).

11.2. 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[1]). Specifically, the running time is $O(n^2 m^3 \log n)$. Moreover, one can show that if capacities are integral then there exists an optimum solution where the flow is integral.

Show how each of the following problems can be solved via this algorithm.

[1]But the exact details of how to do this are outside the scope of this class.

(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. What is the running time of your algorithm as a function of n, m.

(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).

11.3. Maximum Weight Set of Non-Overlapping Arcs

Let $A_1, A_2, \ldots, A_n$ be a set of ***arcs*** on a unit circle. Each arc A_i has a weight w_i and is specified as $[s_i, t_i]$ where s_i and t_i are on the circle and the arc is anti-clockwise from s_i to t_i. The goal is to find a maximum weight subset of the arcs that do not overlap. Describe a way to solve this problem by reducing it to the algorithm you have seen previously for finding a maximum weight subset of intervals that do not overlap (that we solved via dynamic programming). Is the reduction a Turing reduction or a Karp reduction?

11.4. Transportation Problem.

Let G be a digraph with n vertices and m edges. In the transportation problem, you are given a set X of x vertices in a graph G, for every vertex $v \in X$ there is a quantity $q_x > 0$ of material available at v. Similarly, there is a set of vertices Y, with associated capacities c_y with each vertex $y \in Y$. Furthermore, every edge of G has an associated distance with it.
The work involved in transporting α units of material on an edge e of length ℓ is $\alpha * \ell$. The problem is to move all the material available in X to the vertices of Y, without violating the capacity constraints of the vertices, while minimizing the overall work involved. (You can assume the edges have no capacity constraints.)
Provide a polynomial time algorithm for this problem. How fast is your algorithm? The minimum such price of moving the commodity from X to Y is known as the ***earth mover distance***.
(Again, assume you can solve the min-cost flow efficiently.)