CSE 421 Intro to Algorithms Autumn 2017

Homework 7

Due Friday December 1, 4:00 pm

Problem 1:

In a standard s-t Maximum-Flow Problem, we assume edges have capacities, and there is no limit on how much flow is allowed to pass through a node. In this problem, we consider the variant of the Maximum-Flow and Minimum-Cut problems with node capacities.

Let $G = (V, E)$ be a directed graph, with source $s \in V$, sink $t \in V$, and nonnegative node capacities $\{c_v \geq 0\}$ for each $v \in V$. Given a flow f in this graph, the flow though a node v is defined as $f^{\text{in}}(v)$. We say that a flow is feasible if it satisfies the usual flow-conservation constraints and the node-capacity constraints: $f^{\text{in}}(v) \leq c_v$ for all nodes.

Give a polynomial-time algorithm to find an s-t maximum flow in such a node-capacitated network. Define an s-t cut for node-capacitated networks, and show that the analogue of the Max-Flow Min-Cut Theorem holds true.

Problem 2:

We define the *Escape Problem* as follows. We are given a directed graph $G = (V, E)$ (picture a network of roads). A certain collection of nodes $X \subset V$ are designated as *populated nodes*, and a certain other collection $S \subset V$ are designated as *safe nodes*. (Assume that X and S are disjoint.) In case of an emergency, we want evacuation routes from the populated nodes to the safe nodes. A set of evacuation routes is defined as a set of paths in G so that (i) each node in X is the tail of one path, (ii) the last node on each path lies in S, and (iii) the paths do not share any edges. Such a set of paths gives a way for the occupants of the populated nodes to "escape" to S, without overly congesting any edge in G.

(a) Given G, X, and S, show how to decide in polynomial time whether such a set of evacuation routes exists.

(b) Suppose we have exactly the same problem as in (a), but we want to enforce an even stronger version of the "no congestion" condition (iii). Thus we change (iii) to say "the paths do not share any *nodes*."

With this new condition, show how to decide in polynomial time whether such a set of evacuation routes exists.

Also, provide an example with the same G, X, and S, in which the answer is yes to the question in (a) but no to the question in (b).

Problem 3:

You've periodically helped the medical consulting firm Doctors Without Weekends on various hospital scheduling issues, and they've just come to you with a new problem. For each of the next n days, the hospital has determined the number of doctors they want on hand; thus, on day i, they have a requirement that *exactly* p_i doctors be present.

There are k doctors, and each is asked to provide a list of days on which he or she is willing to work. Thus doctor j provides a set L_j of days on which he or she is willing to work.

The system produced by the consulting firm should take these lists and try to return to each doctor j a list L'_j with the following properties.

(A) L'_j is a subset of L_j, so that doctor j only works on days he or she finds acceptable.

(B) If we consider the whole set of lists $L'_1, \ldots, L'_k$, it causes exactly p_i doctors to be present on day i, for $i = 1, 2, \ldots, n$.

(a) Describe a polynomial-time algorithm that implements this system. Specifically, give a polynomial-time algorithm that takes the numbers $p_1, p_2, \ldots, p_n$, and the lists $L_1, \ldots, L_k$, and does one of the following two things.
- Return lists $L'_1, L'_2, \ldots, L'_k$ satisfying properties (A) and (B); or
- Report (correctly) that there is no set of lists $L'_1, L'_2, \ldots, L'_k$ that satisfies both properties (A) and (B).

(b) The hospital finds that the doctors tend to submit lists that are much too restrictive, and so it often happens that the system reports (correctly, but unfortunately) that no acceptable set of lists $L'_1, L'_2, \ldots, L'_k$ exists.

Thus the hospital relaxes the requirements as follows. They add a new parameter $c > 0$, and the system now should try to return to each doctor j a list L'_j with the following properties.

(A*) L'_j contains at most c days that do not appear on the list L_j.

(B) *(Same as before)* If we consider the whole set of lists $L'_1, \ldots, L'_k$, it causes exactly p_i doctors to be present on day i, for $i = 1, 2, \ldots, n$.

Describe a polynomial-time algorithm that implements this revised system. It should take the numbers $p_1, p_2, \ldots, p_n$, the lists $L_1, \ldots, L_k$, and the parameter $c > 0$, and do one of the following two things.
- Return lists $L'_1, L'_2, \ldots, L'_k$ satisfying properties (A*) and (B); or
- Report (correctly) that there is no set of lists $L'_1, L'_2, \ldots, L'_k$ that satisfies both properties (A*) and (B).

Problem 4 (Extra Credit):

We saw how repeatedly augmenting along shortest paths (given by a BFS each time) produced a better runtime analysis for general network flow. In this problem you will do something similar in the special case of finding a maximum matching in a bipartite graph $G = (V, E)$ where the two sides of V are X and Y, $n = |V|$, $m = |E|$. The idea here will be to augment along many shortest paths at the same time. In order for this to work, the paths better not share any edges. Your algorithm will add the slightly stronger condition that the paths do not share any vertices and will include as many shortest paths as possible at each step.

More precisely, at each round the new algorithm will simultaneously augment along all paths in some **maximal set** P_{short} of vertex-disjoint shortest augmenting paths in the flow graph associated with the bipartite matching problem. (In other words, if $d_f(s,t)$ is the length of the shortest augmenting path in the residual graph G_f then every path in P_{short} has length $d_f(s,t)$, no two paths in P_{short} share any vertices other than s or t, and any other st-path of length $d_f(s,t)$ shares at least one vertex other than s and t with a path in P_{short}.)

a. Show how to find a set P_{short} and do all the augmentations on its paths to get a flow f' in time $O(m + n)$.
b. Prove that $d_{f'}(s,t) \geq d_f(s,t) + 2$.
c. Given two matchings M and M' on graph G we can define the **symmetric difference**, $M \Delta M'$, of M and M' to be the graph consisting of edges that occur in exactly one of the two matchings. $M \Delta M'$consists of a collection of vertex-disjoint paths and cycles (why?). Show that if M' is a maximum matching and flow f corresponds to a matching M of G then
   i. $M \Delta M'$contains exactly $|M'| - |M|$ vertex-disjoint odd-length paths.
   ii. Any path of odd length k in $M \Delta M'$corresponds to an augmenting path of length $k + 2$ in G_f.
   iii. Use these two properties to prove that once all augmenting paths in G_f are of length at least $k + 2$ then at most n/k additional augmentations will produce a maximum flow (and hence maximum matching).
d. Use the above analysis with a suitable value of k to show that this algorithm computes a maximum matching in $O(n^{\frac{1}{2}}\, m)$ time