CS222 Algorithm Design and Analysis Homework 5

Zhou Litao 518030910407 F1803016

November 10, 2020, Fall Semester

Exercise 1 Figure 1 shows a flow network on which an s-t flow has been computed. The capacity of each edge appears as a label next to the edge, and the numbers in boxes give the amount of flow sent on each edge. Edges without boxed numbers have no flow being sent on them.

1. What is the value of this flow? Is this a maximum (s, t) flow in this graph? If not, calculate the value of the maximum flow.
2. Find a minimum s-t cut in the flow network. Calculate its capacity.

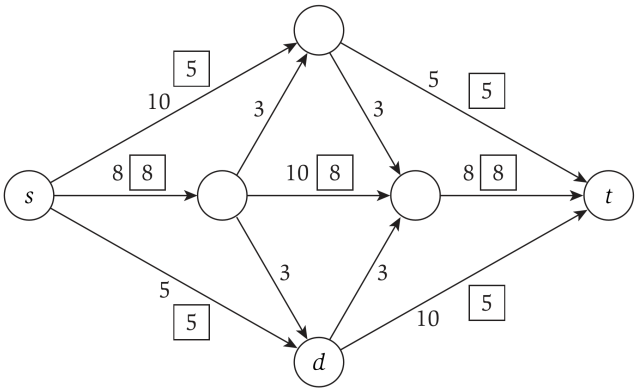

Figure 1: A flow network.

Solution.

1. The value of this flow is $5+8+5=18$. It is not a maximum flow. The maxmimum flow is 21, depicted in 2.
2. The minimum s-t cut is composed of $\{s, a\}$ and the other nodes. Its capacity is $5+8+5+3=21$

□

Exercise 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 collections $S \subset V$ are designated as *safe nodes*. 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 head 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 safe nodes, without overly congesting any edge in G.

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

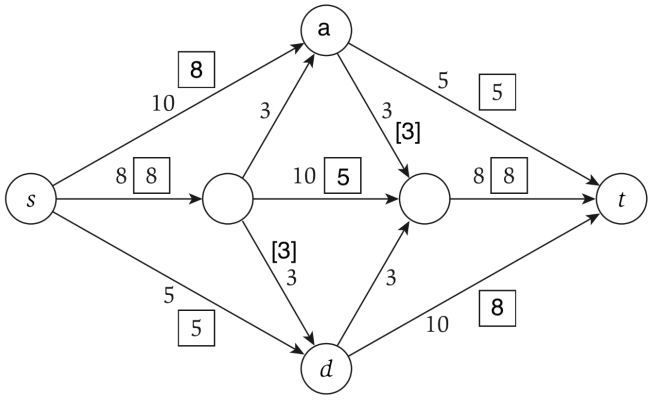

Figure 2: My flow network.

2. 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).

Solution.

1. Based on the graph given in the problem, let all edges have a unit capacity. Then, add a starting node s and add an edge with unit capacity from s to every node in X. Also, add a end ending node t and connect all the nodes in S to it with unit capacity. With the flow graph constructed, use Ford-Fulkerson algorithm to find the maximum flow, which runs in polynomial time given all the capacities are 1. If the maximum flow is $|X|$, such a set of evacuation routes can be found. Otherwise, there exists no such non-intersecting evacuation routes.

2. For every node that are neither populated nodes nor safe nodes, expand it into two nodes, with a unit capacity between them. Then we can make sure that the node in the original graph will be assigned a unique group to pass by.

   An example is presented below in 3

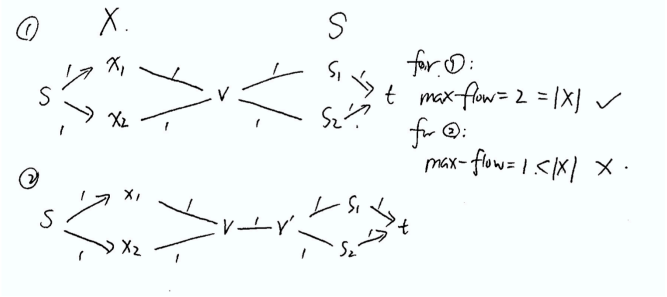

Figure 3: An example for Problem 2.

□

Exercise 3 Suppose there are n people living in a cooperative apartment. Over the next n nights, each of you is supposed to cook dinner for the co-op exactly once, so that someone cooks on each of the nights.

Of course, everyone has scheduling conflicts with some of the nights, so deciding who should cook on which night becomes a tricky task. For concreteness, let's label the people $\{p_1, ..., p_n\}$ and the nights $\{d_1, ..., d_n\}$. For person p_i, there's a set of nights $S_i \subset \{d_1, ..., d_n\}$ when they are able to cook.

A feasible dinner schedule is an assignment of each person in the co-op to a different night, so that each person cooks on exactly one night, and there is someone cooking on each night. If p_i cooks on night d_j, then $d_j \in S_i$.

1. Describe a bipartite graph G so that G has a perfect matching if and only if there is a feasible dinner schedule for the co-op.

2. There is a schedule constructed by your friend. Unfortunately, when you look at the schedule she created, you notice a big problem. $n-2$ of the people at the co-op are assigned to different nights on which they are available: no problem there. But for the other two people, p_i and p_j, and the other two days, d_k and d_l, you discover that she has accidentally assigned both p_i and p_j to cook on night d_k, and assigned no one to cook on night d_l.

   You want to fix this mistake without having to recompute everything from scratch. Show that it's possible, using the "almost correct" schedule, to decide in only $O(n^2)$ time whether there exists a feasible dinner schedule for the co-op.

Solution.

1. We can construct a G with $V = P \cup D$ and $E = \{(p_i, d_j) | d_j \in S_i\}$. The graph is divided into P and D, with $|P| = |D| = n$. There is a feasible dinner schedule if and only if there is a perfect matching in G.

2. We can solve the problem by using a max-flow model. First, construct the graph described in 1, then remove all the correctly assigned (p_x, d_x) edges, and add a reverse edge (d_x, p_x) in the graph. Then let (p_i, d_k) also be reversed. To correct the mistake, we just need to start from p_j and check whether there exists a simple path to d_l. If the path exists, then there exists a feasible dinner schedule by reversing the edges along the path. Then all the backward edges can form a perfect matching.

   The construction of the graph takes $O(n^2)$ time and the finding of simple path takes $O(|E|) < O(n^2)$ time. Therefore the overall algorithm runs in $O(n^2)$ time.

□

Exercise 4 Recall that in the standard network flow problem we required that the for each vertex v (excluding the source s and sink t) the sum of the flow into that vertex is equal to the sum of flow out of that vertex. Suppose that we replace this conservation constraint with a taxation constraint. In particular, suppose each vertex represents a country and that each country $v \notin \{s, t\}$ has an associated tax-rate $0 < t_v < 1$ meaning that country v will keep t_v fraction of the goods flowing through node v. Given a flow network $G = (V, E)$ with maximum capacities $c(e)$ on each edge $e \in V$ and tax rates t_v for each node $v \notin s, t$ our goal is to find the maximum amount of goods that can be transported from s to t under these taxation constraints. Write down your algorithm to solve this problem. You should explain why your algorithm is correct.

Solution.

The problem can be formulated as a linear programming problem and be solved through simplex algorithm.

Let all the nodes excluding s and t be given index $i = 1, \ldots, n-1$, and s be given index 0, t be given index n. Let c_{ij} denote the capacity from node i to j and f_{ij} denote the actual flow from i to j used in the final solution. For every flow, we have that

$$0 \leq f_{ij} \leq c_{ij}, \forall i, j \tag{1}$$

For every node, we have that

$$t_k \sum_{i=0}^{n} f_{ik} = \sum_{j=0}^{n} f_{kj}, \forall k = 1, \ldots, n-1 \tag{2}$$

The maximum amount of goods that can be transported can be defined as $\sum_{i=0}^{n-1} f_{in}$. i.e. the flow into t. We have formulated a linear programming model as follows.

$$\begin{aligned} \max_{f_{ij}, f'_{ij}} & \sum_{i=0}^{n-1} f_{in} \\ \text{s.t. } & f_{ij} \geq 0, \forall i, j = 1 \ldots n \\ & f'_{ij} \geq 0, \forall i, j = 1 \ldots n \\ & f'_{ij} = c_{ij} - f_{ij}, \forall i, j = 1 \ldots n \\ \sum_{j=0}^{n} f_{kj} - t_k & \sum_{i=0}^{n} f_{ik} = 0, \forall k = 1 \ldots n-1 \end{aligned} \tag{3}$$

The problem has already been transformed into a slack problem P in Problem 3, with basis variables f'_{ij}. A trivial initial solution is to assign all f_{ij} to be 0. We can use the simplex algorithm to solve this problem P. The algorithm is described as follows. Every time we run the algorithm, the solution will be improved, until the solution converges.

For sake of simplicity in expression, we denote c to be the coefficient vector for the objective function and A to be the coefficient matrix for equality constraints.

Algorithm 2: Simplex Algorithm

Input : Linear program (P) and feasible basis B
Output: An optimal solution for (P) or a certificate proving that (P) is unbounded

1 Rewrite (P) so that it is in canonical form for the basis B
2 Let $\vec{x}$ be the basic feasible solution for B
3 **if** $\vec{c_N} \leq \vec{0}$ **then**
4 **{**
5 **stop**
6 ($\vec{x}$ is optimal)
7 **}**
8 Select $k \in N$ such that $c_k > 0$
9 **if** $A_k \leq \vec{0}$ **then**
10 **{**
11 **stop**
12 ((P) is unbounded)
13 **}**
14 Let r be any index i where the following minimum is attained:

$$t = \min \left\{ \frac{b_i}{A_{i,k}} : A_{i,k} > 0 \right\} \tag{4}$$

15 Let ι be the r^{th} basis element
16 Set $B := B \cup \{k\} \backslash \{\iota\}$
17 Go to step 1

□