Practice Problems 9

Problem 1. In standard network flow, every vertex other that s and t must satisfy flow conservation, that is, the total flow into the vertex must equal the total flow out. In this problem, we'll consider a variant where vertices are allowed to "leak" so that some of the incoming flow is lost due to leakage.

A *leaky network* is the same as a standard flow network (that is, a digraph $G = (V, E)$, source s and sink t, and edge capacities $c(u, v)$ for each $(u, v) \in E$), but in addition, each vertex $v \in V$, is associated with a nonnegative *leak capacity*, denoted $h(v)$, which indicates the maximum leakage that can occur at this vertex (see Fig. 1(a)).

A flow f in a leaky network is valid if it satisfies the usual capacity constraint, but (other than s and t) the flow out of a vertex can be smaller than the flow in by up to $h(v)$, that is,

$$f^{\text{in}}(v) - h(v) \;\leq\; f^{\text{out}}(v) \;\leq\; f^{\text{in}}(v)$$

The value of a flow f is the flow out of the source, $f^{\text{out}}(s)$. (see Fig. 1(b)).)

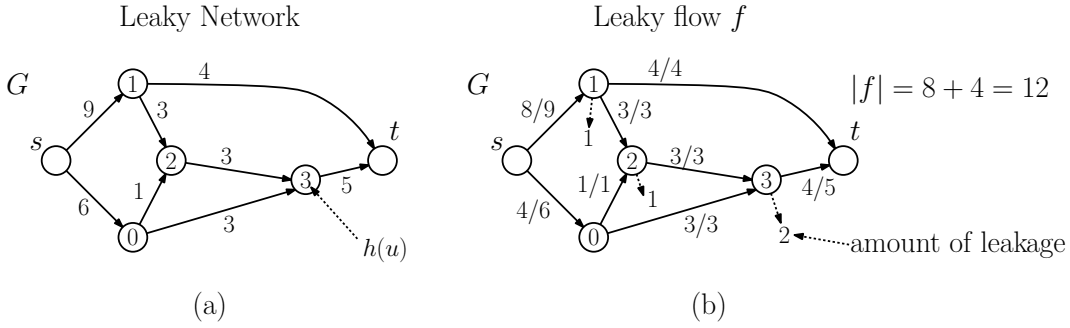

Figure 1: (a) A leaky network (each vertex u contains its leak capacity $h(u)$) and (b) a flow in the network (dashed lines indicate leak amounts).

(a) Explain how to transform any leaky network G into an equivalent (non-leaky) network G' so that, for any valid leaky flow f in G there exists a flow f' of equivalent value in G', and vice versa. (Note: Your transformation is given G, the edge capacities, $c(u, v)$, and the leak capacities, $h(v)$, but not the flow.) Justify the correctness of your construction.

(b) Demonstrate how your transformation works on the leaky network in Fig. 1(a). (Just show the transformed non-leaky network. You do not need to show the flow.)

Problem 2. You are given a directed network $G = (V, E)$ with a root node r and a set of terminal nodes $T = \{t_1, \ldots, t_k\}$. Present a polynomial time algorithm to determine the minimum number of edges to remove so that there is no path from r to any of the terminals (see Fig 2). Prove that your algorithm is correct.

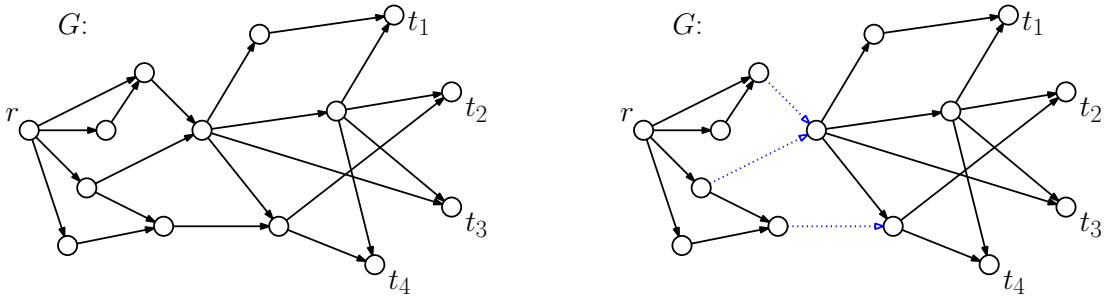

Figure 2: Eliminating edges to separated r from terminals.

Problem 3. The Ford-Fulkerson algorithm operates by finding an augmenting path in the residual network. The effect of pushing flow along this path may result in flows increasing on some edges and decreasing on others. Consider instead an algorithm that *only increases* flows along the edges of some path from s to t. (There may generally be many such paths, and the algorithm is free to chose any of them.) We call this the *nondecreasing flow algorithm*.

Show that the nondecreasing flow algorithm can be *arbitrarily bad*. In particular, given any positive integer $b > 1$, give an example of an s-t network G such that the ratio between the optimum flow in G and the flow generated by the nondecreasing algorithm is at least as large as b. (The structure of the network will depend of course on b. You may give your example for a specific value of b, but it should be easy to see how to generalize it to arbitrary values of b. Remember that you may select the sequence of paths along which augmentations are to be performed.) Briefly explain.