# Problems: Applications of Max Flow

Sudeshna Kolay
Indian Institute of Technology, Kharagpur

August 26, 2024

1. Using max flow, prove Menger's theorem: If an undirected graph remains connected after removing any set of fewer than $k$ edges, then the size of the minimum cut is at least $k$. A cut is a set of edges whose removal makes the graph disconnected.

2. [KT book] Network flow issues come up in dealing with natural disasters and other crises, since major unexpected events often require the movement and evacuation of large numbers of people in a short amount of time.

   Consider the following scenario. Due to large-scale flooding in a region, paramedics have identified a set of $n$ injured people distributed across the region who need to be rushed to hospitals. There are $k$ hospitals in the region, and each of the $n$ people needs to be brought to a hospital that is within a half-hour's driving time of their current location (so different people will have different options for hospitals, depending on where they are right now).

   At the same time, one doesn't want to overload any one of the hospitals by sending too many patients its way. The paramedics are in touch by cell phone, and they want to collectively work out whether they can choose a hospital for each of the injured people in such a way that the load on the hospitals is balanced: Each hospital receives at most $\frac{n}{k}$ people.

   Give a polynomial-time algorithm that takes the given information about the people's locations and determines whether this is possible.

3. [KT book] Your friends have written a very fast piece of maximum-flow code based on repeatedly finding augmenting paths. However, after you've looked at a bit of output from it, you realize that it's not always finding a flow of maximum value. The bug turns out to be pretty easy to find; your friends hadn't really gotten into the whole backward-edge thing when writing the code, and so their implementation builds a variant of the residual graph that only includes the forward edges. In other words, it searches for s-t paths in a graph $\tilde{G}_f$ consisting only of edges $e$ for which $f(e) < c(e)$, and it terminates when there is no augmenting path consisting entirely of such edges. We'll call this the Forward-Edge-Only Algorithm. (Note that we do not try to prescribe how this algorithm chooses its forward-edge paths; it may choose them in any fashion it wants, provided that it terminates only when there are no forward-edge paths.)

   It's hard to convince your friends they need to reimplement the code. In addition to its blazing speed, they claim, in fact, that it never returns a flow whose value is less than a fixed fraction of optimal. Do you believe this? The crux of their claim can be made precise in the following statement.

   *There is an absolute constant b ¿ 1 (independent of the particular input flow network), so that on every instance of the Maximum-Flow Problem, the Forward-Edge-Only Algorithm is guaranteed to find a flow of value at least 1/b times the maximum-flow value (regardless of how it chooses its forward-edge paths).*

   Decide whether you think this statement is true or false, and give a proof of either the statement or its negation.

4. [KT book] 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 $X \subseteq V$ are designated as populated nodes, and a certain other collection $S \subseteq V$ 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).

5. [KT book] You've been called in to help some network administrators diagnose the extent of a failure in their network. The network is designed to carry traffic from a designated source node s to a designated target node t, so we will model the network as a directed graph G = (V, E), in which the capacity of each edge is 1 and in which each node lies on at least one path from s to t.

   Now, when everything is running smoothly in the network, the maximum s-t flow in G has value k. However, the current situation (and the reason you're here) is that an attacker has destroyed some of the edges in the network, so that there is now no path from s to t using the remaining (surviving) edges. For reasons that we won't go into here, they believe the attacker has destroyed only k edges, the minimum number needed to separate s from t (i.e., the size of a minimum s-t cut); and we'll assume they're correct in believing this.

   The network administrators are running a monitoring tool on node s, which has the following behavior. If you issue the command ping(v), for a given node v, it will tell you whether there is currently a path from s to v. (So ping(t) reports that no path currently exists; on the other hand, ping(s) always reports a path from s to itself.) Since it's not practical to go out and inspect every edge of the network, they'd like to determine the extent of the failure using this monitoring tool, through judicious use of the ping command. So here's the problem you face: Give an algorithm that issues a sequence of ping commands to various nodes in the network and then reports the full set of nodes that are not currently reachable from s. You could do this by pinging every node in the network, of course, but you'd like to do it using many fewer pings (given the assumption that only k edges have been deleted). In issuing this sequence, your algorithm is allowed to decide which node to ping next based on the outcome of earlier ping operations. Give an algorithm that accomplishes this task using only O(k log n) pings.

6. [KT book] Let M be an $n \times n$ matrix with each entry equal to either 0 or 1. Let $m_{ij}$ denote the entry in row i and column j. A diagonal entry is one of the form $m_{ij}$ for some i.

   Swapping rows i and j of the matrix M denotes the following action: we swap the values $m_{ik}$ and $m_{jk}$ for k = 1, 2, . . . , n. Swapping two columns is defined analogously.

   We say that M is rearrangeable if it is possible to swap some of the pairs of rows and some of the pairs of columns (in any sequence) so that, after all the swapping, all the diagonal entries of M are equal to 1.

   (a) Give an example of a matrix M that is not rearrangeable, but for which at least one entry in each row and each column is equal to 1.

   (b) Give a polynomial-time algorithm that determines whether a matrix M with 0-1 entries is rearrangeable.