Problems: Flow Basics, Ford-Fulkerson Method, Edmond-Kard Algorithm

Palash Dey
Indian Institute of Technology, Kharagpur

1. Explain how we can assume without loss of generality that the input directed graph for the network flow problem does not contain any self loop or anti-parallel edges. For two vertices u and v, if the graph has an edge from u to v and another edge from v to u, then these pair of edges are called anti-parallel edges.
2. Run the fattest path first (augment along the path where the maximum amount of flow can be sent from s to t) implementation of the Ford-Fulkerson method and the Edmond-Karp algorithm on large random graphs and evaluate their relative performance by varying the number of vertices, edges, and average capacities of the edges. Use your favorite programming language.
3. Modify your above code to compute a minimum capacity $s-t$ cut in a flow network.
4. [CLRS book] Suppose that, in addition to edge capacities, a flow network has vertex capacities. That is each vertex v has a limit $\ell(v)$ on how much flow can pass though v. Show how to transform a flow network $\mathcal{G}(\mathcal{V},\mathcal{E},w:\mathcal{V}\longrightarrow\mathbb{R}_{>0})$ with vertex capacities into an equivalent flow network $\mathcal{G}'(\mathcal{V}',\mathcal{E}',w':\mathcal{E}\longrightarrow\mathbb{R}_{>0})$ without vertex capacities, such that a maximum flow in $\mathcal{G}'$ has the same value as a maximum flow in $\mathcal{G}$. How many vertices and edges does $\mathcal{G}'$ have?
5. Suppose that we have multiple source and sink vertices in a flow network. The value of a flow in such a network is defined as the total amount of flow going out of all sources. Show how to transform this problem into an equivalent conventional (taught in the class) flow network such that the maximum flow value remains the same.
6. [CLRS book] Suppose that you wish to find, among all minimum cuts in a flow network $\mathcal{G}$ with integral capacities, one that contains the smallest number of edges. Show how to modify the capacities of $\mathcal{G}$ to create a new flow network $\mathcal{G}'$ in which any minimum cut in $\mathcal{G}'$ is a minimum cut with the smallest number of edges in $\mathcal{G}$.
7. Design an algorithm to check if a flow network has an unique minimum capacity $s-t$ cut and if not, then output two different minimum capacity $s-t$ cuts.
8. We are given a flow network and a maximum flow in that network.
   (a) Suppose we increase the capacity of an edge by 1. Given an $\mathcal{O}(m+n)$ time algorithm to update the maximum flow.
   (b) Suppose we decrease the capacity of an edge by 1. Given an $\mathcal{O}(m+n)$ time algorithm to update the maximum flow.
9. Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be an undirected and unweighted graph. For $\mathcal{X}\subseteq\mathcal{V}$, we define $\delta(\mathcal{X})$ to be the capacity of the cut $(\mathcal{X},\mathcal{V}\setminus\mathcal{X})$. Then show the following for every two subsets $\mathcal{A},\mathcal{B}\subseteq\mathcal{V}$
$$\delta(\mathcal{A})+\delta(\mathcal{B})\geqslant\delta(\mathcal{A}\cup\mathcal{B})+\delta(\mathcal{A}\cap\mathcal{B})$$
10. [Path decomposition of a flow] A flow f in a flow network $\mathcal{G}$ is called *not acyclic* if there exists a directed cycle in $\mathcal{G}$ each of its edges carry positive amount of flow. Otherwise, f is called acyclic.
    (a) Prove that, for every flow f, there exists an acyclic flow f' of value the same as the value of f.
    (b) A *path flow* is a flow which assigns a positive flow only to the edges of an $s-t$ path. Prove that, every acyclic flow f can be written as the sum of at most m path flows.

(c) Is the Ford-Fulkerson algorithm guaranteed to find an acyclic flow?

(d) A *cycle flow* is a flow which assigns a positive flow only to the edges of a directed cycle. Prove that, every flow f can be written as the sum of at most m path flows and cycle flows. Given a flow network and a flow f, design an algorithm to find the above decomposition in $\mathcal{O}(mn)$ time.

11. [Dinic's Algorithm] A *blocking flow* f in a flow network $\mathcal{G}(\mathcal{V}, \mathcal{E}, c : \mathcal{E} \longrightarrow \mathbb{R}_{>0})$ is a flow such that, in every $s-t$ path, there is at least one edge e such that $c_e = f_e$.

(a) Given a network $\mathcal{G}$, and an $s-t$ flow f, construct the BFS tree starting from s. Delete backward and cross edges of the BFS tree from the residual graph $\mathcal{G}_f$. We call the resulting graph the layered graph $\mathcal{L}_f$ with respect to f. Prove that $\mathcal{L}_f$ is acyclic.

(b) Design an $\mathcal{O}(mn)$-time algorithm to compute a blocking flow f in $\mathcal{L}_f$. *Hint: Modify the standard DFS appropriately!*

(c) Consider the following algorithm for computing a maximum flow in a network. Start with the zero flow. Compute a blocking flow in the corresponding layered graph', augment it with the current flow, and iterate until there is no path from s to t in the residual graph. Prove that the run time of this algorithm $\mathcal{O}(mn^2)$. *Hint: How does the distance (number of edges in a shortest path) of any vertex from s changes after every iteration?*