

Tutorial Sheet 11

- This tutorial sheet contains problems in network flow. When presenting your solutions, please draw/describe the flow network clearly. The correctness and running time arguments, if required, can be brief (1-2 sentences).
- The tutorial quiz problem for different groups will be selected from the following sets:
 - Group 1 (Oct 19, Mon): Problems 1-2
 - Group 2 (Oct 19, Mon): Problems 1-2 (rescheduled tutorial)
 - Group 3 (Oct 22, Thurs): Problems 1-5
 - Group 4 (Oct 23, Fri): Problems 1-7

1. In class, we discussed the maximum flow problem in directed graphs. In the *undirected* version of the problem, the input is an undirected graph $G = (V, E)$, a source vertex $s \in V$, a sink vertex $t \in V$, and an integer capacity $u_e \geq 0$ for each edge $e \in E$. Flows are defined exactly as before, and remain *directed*. Formally, a flow consists of two nonnegative numbers $f_{u,v}$ and $f_{v,u}$ for each undirected edge $\{u, v\} \in E$, indicating the amount of traffic traversing the edge in each direction. Conservation constraints (flow in = flow out) are defined as before. Capacity constraints now state that, for every undirected edge $e = \{u, v\} \in E$, the total amount of flow $f_{u,v} + f_{v,u}$ on the edge is at most the edge's capacity u_e . The value of a flow is the net amount going out of the source, i.e.,

$$\sum_{v: \{s,v\} \in E} f_{s,v} - \sum_{v: \{s,v\} \in E} f_{v,s}.$$

Prove that the maximum flow problem in undirected graphs reduces to the maximum flow problem in directed graphs. That is, given an instance of the undirected problem, show how to (i) produce an instance of the directed problem such that (ii) given a maximum flow to this directed instance, you can recover a maximum flow of the original undirected instance. Your implementations of steps (i) and (ii) should run in linear time. Include a brief proof of correctness.

2. Suppose we generalize the maximum flow problem so that there are *multiple* source vertices $s_1, \dots, s_k \in V$ and sink vertices $t_1, \dots, t_\ell \in V$. (As usual, the rest of the input is a directed graph with integer edge capacities.) You should assume that no vertex is both a source and sink, that source vertices have no incoming edges, and that sink vertices have no outgoing

edges. A flow is defined as before: a nonnegative number f_e for each edge $e \in E$ such that capacity constraints are obeyed on every edge and such that conservation constraints hold at all vertices that are neither a source nor a sink. The value of a flow is the total amount of outgoing flow at the sources, i.e.,

$$\sum_{i=1}^k \sum_{e \in \delta^+(s_i)} f_e$$

where $\delta^+(s_i)$ denotes the outgoing edges from the vertex s_i .

Prove that the maximum flow problem in graphs with multiple sources and sinks *reduces* to the single-source single-sink version of the problem. That is, given an instance of the multi-source multi-sink version of the problem, show how to (i) produce a single-source single-sink instance such that (ii) given a maximum flow to this single-source single-sink instance, you can recover a maximum flow of the original multi-source multi-sink instance. Your implementations of steps (i) and (ii) should run in linear time. Include a brief proof of correctness.

3. For every positive integer U , show that there is an instance of the maximum flow problem with edge capacities in $\{1, 2, \dots, U\}$ and a choice of augmenting paths so that the Ford-Fulkerson algorithm runs for at least U iterations before terminating. The number of vertices and edges in your networks should be bounded above by a constant independent of U . (This shows that the algorithm is only “pseudopolynomial.”)
4. Show that any $s - t$ flow can be written as a sum of flows along at most m $s - t$ paths and cycles, where m is the number of edges in the graph.
5. Suppose in a directed graph G , there are k edge-disjoint paths from s to t and from t to u . Prove that there are at least k edge-disjoint paths from s to u .
6. In the $s - t$ directed *edge-disjoint* paths problem, the input is a directed graph $G = (V, E)$, a source vertex s , and a sink vertex t . The goal is to output a maximum-cardinality set of edge-disjoint $s - t$ paths $P_1, \dots, P_k$. That is, P_i and P_j should share no edges for each $i \neq j$, and k should be as large as possible.

Prove that this problem reduces to the maximum flow problem. That is, given an instance of the disjoint paths problem, show how to (i) produce an instance of the maximum flow problem such that (ii) given a maximum flow to this instance, you can compute an optimal solution to the disjoint paths instance. Your implementations of steps (i) and (ii) should run in linear and polynomial time, respectively. (Can you achieve linear time also for (ii)?) Include a brief proof of correctness.

7. How will your answer to Problem 6 change if you are required to compute *vertex-disjoint*

paths instead of edge-disjoint paths?

8. (★) Suppose we are given a $m \times n$ matrix A of nonnegative real numbers. We want to *round* A to an integer matrix, by replacing each entry x in A with either $\lceil x \rceil$ or $\lfloor x \rfloor$, without changing the sum of entries in any row or column of A . (Assume that all row and column sums of A are integral.) For example:

$$\begin{bmatrix} 1.2 & 3.4 & 2.4 \\ 3.9 & 4 & 2.1 \\ 7.9 & 1.6 & 0.5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 4 & 2 \\ 4 & 4 & 2 \\ 8 & 1 & 1 \end{bmatrix}.$$

- a) Prove that such a rounding is guaranteed to exist.
 - b) Describe and analyze an efficient algorithm that rounds A in this fashion.
9. (★) Consider the following modification to the Ford-Fulkerson algorithm: Among all possible $s - t$ paths in the residual graph, pick the one such that the maximum amount of flow can be sent along this path. How will you find such a path? Show that you will need at most $m \cdot \log(nU)$ iterations. What is the overall running time?

Hints:

- a) Show how to modify Dijkstra's shortest-path algorithm, without affecting its asymptotic running time, so that it computes an $s - t$ path with the maximum-possible minimum residual edge capacity.
- b) Suppose the current flow f has value F and the maximum flow value in G is F^* . Prove that there is an augmenting path in G_f such that every edge has residual capacity at least $(F^* - F)/m$, where $m = |E|$. You may find it helpful to refer to the paths-and-cycles decomposition of a flow (Problem 4).
- c) You might find the inequality $(1 - x) \leq e^{-x}$ for $x \in [0, 1]$ useful.