CS261: Exercise Set #1

For the week of January 4–8, 2016

Instructions:

(1) *Do not turn anything in.*

(2) The course staff is happy to discuss the solutions of these exercises with you in office hours or on Piazza.

(3) While these exercises are certainly not trivial, you should be able to complete them on your own (perhaps after consulting with the course staff or a friend for hints).

Exercise 1

Suppose we generalize the maximum flow problem so that there are *multiple* source vertices $s_1, \ldots, s_k \in V$ and sink vertices $t_1, \ldots, t_\ell \in V$. (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 $e \in E$ 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: $\sum_{i=1}^{k} \sum_{e \in \delta^+(s_i)} f_e$.

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.

[Hint: consider adding additional vertices and/or edges.]

Exercise 2

In lecture we've focused on 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 a 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_{uv} and f_{vu} 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 edge $e = (u, v) \in E$, the total amount of flow $f_{uv} + f_{vu}$ on the edge is at most the edge's capacity u_e. The value of a flow is the net amount $\sum_{(s,v) \in E} f_{sv} - \sum_{(v,s) \in E} f_{vs}$ going out of the source.

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.

[Hint: consider bidirecting each edge.]

Exercise 3

For every positive integer U, show that there is an instance of the maximum flow problem with edge capacities in $\{1, 2, \ldots, U\}$ 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 constant, independent of U. (This shows that the algorithm is only "pseudopolynomial.")

[Hint: use a network similar to the examples discussed in lecture.]

Exercise 4

Consider the special case of the maximum flow problem in which every edge has capacity 1. (This is called the *unit-capacity* case.) Explain why a suitable implementation of the Ford-Fulkerson algorithm runs in $O(mn)$ time in this special case. (As always, m denotes the number of edges and n the number of vertices.)

Exercise 5

Consider a directed graph $G = (V, E)$ with source s and sink t for which each edge e has a positive integral capacity u_e. For a flow f in G, define the "layered graph" L_f as in Lecture #2, by computing the residual graph G_f and running breadth-first search (BFS) in G_f starting from s, aborting once the sink t is reached, and retaining only the forward edges. (Recall that a forward edge in BFS goes from layer i to layer $(i + 1)$, for some i.)

Recall from Lecture #2 that a *blocking flow* in a network is a flow that saturates at least one edge on each s-t path. Prove that for every flow f and every blocking flow g in L_f, the shortest-path distance between s and t in the new residual graph G_{f+g} is strictly larger than that in G_f.