Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
S. Daum, H. Ghodselahi, O. Saukh

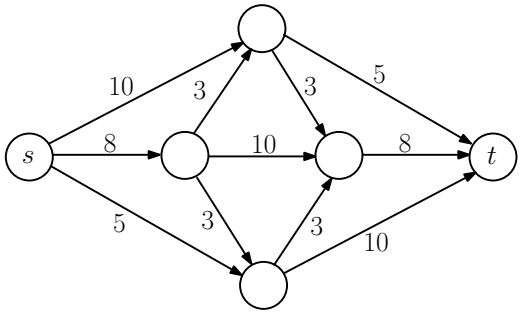
December 19, 2013

Algorithm Theory, Winter Term 2013/14
Problem Set 5

hand in by Thursday, January 9, 2014

Exercise 1: Ford Fulkerson Algorithm

Consider the following flow network:



(a) Solve the maximum flow problem on the above network by using the Ford Fulkerson algorithm. Give all intermediate results.

(b) Give a minimum capacity s-t cut of the given network. Describe how you can get the cut from the maximum flow computed in (a).

Exercise 2: Network Flows

(a) In this problem, we are given a flow network with unit-capacity edges: It consists of a directed graph $G = (V, E)$, a source node $s \in V$, a sink node $t \in V$, and capacity $c_e = 1$ for every $e \in E$. We are also given a positive integer parameter k.

The goal is to delete k edges so as to reduce the maximum s-t flow in G by as much as possible. In other words, you should find a set of edges $F \subseteq E$ so that $|F| = k$ and the maximum s-t-flow in $G' = (V, E \setminus F)$ is as small as possible.

Give a polynomial-time algorithm to solve this problem.

(b) Suppose you are given a directed graph $G = (V, E)$, with a positive integer capacity c_e on each edge e, a source node $s \in V$, and a sink node $t \in V$. You are also given a maximum s-t flow f in G, defined by a flow value f_e on each edge e. The flow f is *acyclic*: There is no directed cycle in G on which all edges carry positive flow. The flow f is also integer-valued.

Now suppose, we pick a specific edge $e^* \in E$ and reduce its capacity by 1 unit. Show how to find a maximum flow in the resulting capacitated graph in time $O(m + n)$, where m is the number of edges in G and n is the number of nodes.

Exercise 3: Forward-Only Paths

A friend of you has written some very fast maximum flow code. Unfortunately it turns out that the program doesn't always compute a correct maximum flow. When inspecting the solution you realize that your friend's program implements a simplified variant of the Ford-Fulkerson algorithm. When computing augmenting paths, the program only considers forward edges of the residual graph and it does not consider backward edges at all. We have seen in the lecture that backward edges are necessary to get a correct algorithm. However your friend claims that his algorithm (let's call it the forward-edge-only algorithm) always computes a solution that is within a constant factor of the optimal one. That is, there is an absolute constant $b > 1$ such that the forward-edge-only algorithm computes a flow of value at least $1/b$ times the value of an optimal flow. Is your friend right? If yes, prove it, otherwise show that the ratio of the maximum flow value and the flow computed by the forward-edge-only algorithm can be arbitrarily large. Assume that the forward-edge-only implementation always takes an arbitrary (possibly worst-case) augmenting path of only forward edges as long as such an augmenting path exists. You can also assume that all edge capacities are positive integers.