Technische Universität München
Department of Computer Science
Chair for Efficient Algorithms
Prof. Dr. Sami Khuri

WS 1999/2000
Solutions 2
May 23, 2000

Efficient Algorithms and Data Structures II

Problem 1 (10 credits)

Let $G = (V, E)$ be a flow network, let f be a flow in G, and let p be an augmenting path in G_f. Define a function $f_p : V \times V \to \mathbb{R}$ by

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p, \\ -c_f(p) & \text{if } (v,u) \text{ is on } p, \\ 0 & \text{otherwise} \end{cases}$$

where $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\}$ and $c_f(u,v) = c(u,v) - f(u,v)$ is the residual capacity of edge (u,v).

Prove that f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$.

Solution.

Trivially $c_f(p) > 0$, since p is an augmenting path, and by definition the minimum residue capacity along p has to be larger than 0. It remains to show that the three basic properties of a flow are fulfilled here.

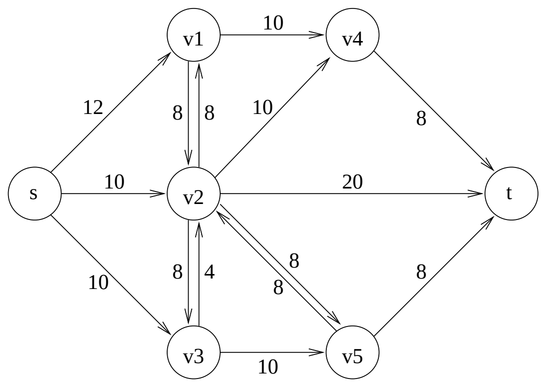
Capacity. We have $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ on } p\}$. Thus, $f_p(u,v) \leq c_f(u,v)\ \forall (u,v) \in E_f$ (either $f_p(u,v) = c_f(u,v)$ for (u,v) on p, $f_p(u,v) = 0$ whenever neither (u,v) nor $v,u)$ on p, or finally $f_p(u,v) < 0$ whenever (v,u) on p).

Skew symmetry. Fulfilled by definition. If neither (u,v) nor (v,u) are on p, we have $f_p(u,v) = f_p(v,u) = 0$. If (u,v) on p, we have by definition $f_p(u,v) = c_f(p)$ and $f_p(v,u) = -f_p(u,v)$. Analogously for (v,u) on p.

Flow conservation. Fulfilled by definition. If neither (u,v) nor (v,u) on p, there is no flow between u and v. Since for all other edges on p (or backwards on p), we have the very same flow, flow conservation is obviously given.

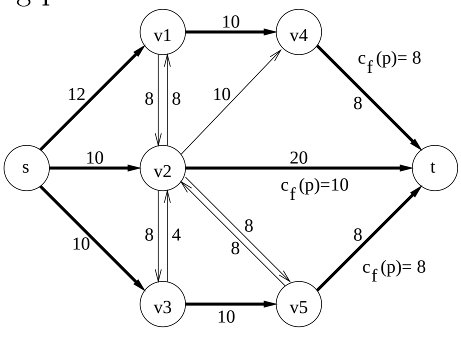
Problem 2 (10 credits)

Use the Ford-Fulkerson algorithm to produce a maximum flow for the following network.

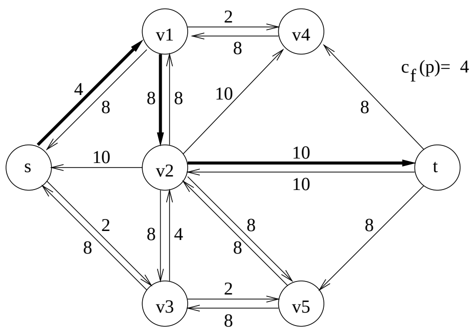


Solution.

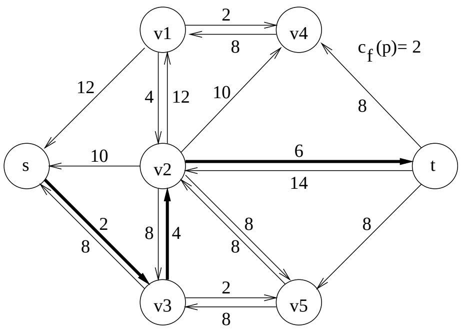
The following figure shows one possible performance of the Ford-Fulkerson algorithm (recall that the augmenting paths are selected arbitrarily).
During intermediate phases we show only the current residue network from which we then finally conclude the flow network and determine the maximum flow. Augmenting paths are depicted by fat arrows. Note that in the first step, we simultaneosly proceed three augmenting paths.



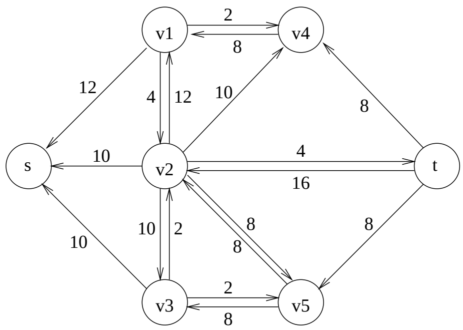
paths s -> v1 -> v4 -> t
s -> v2 -> t
s -> v3 -> v5 -> t



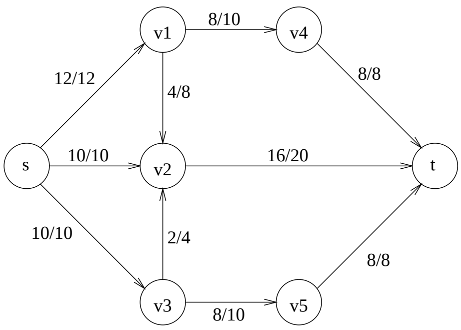
path s -> v1 -> v2 -> t



path s -> v3 -> v2 -> t



Resulting residue network



Resulting flow network
(max.flow is 32)

Problem 3 (10 credits)

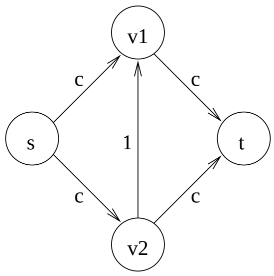
Give an efficient algorithm to find the maximum flow in a given network whose underlying graph is undirected.

Solution.

Let G be the undirected graph in question. From G, we construct a directed graph G' by replacing each edge of G by a pair of anti-parallel directed edges, that is, an undirected edge connecting nodes u and v becomes two directed edges (u, v) and (v, u).
On this graph we now run any (efficient) maximum flow algorithm (like Ford-Fulkerson) on G'. To transport the result back to the undirected case, for any pair of directed edges $(u, v), (v, u)$ we simply assign the difference between the two flows over (u, v) and (v, u) to the undirected edge between u and v. Let $f(u, v)$ and $f(v, u)$ denote those two flows. Then, if $f(u, v) > f(v, u)$, the (undirected) edge (u, v) gets assigned a flow of $f(u, v) - f(v, u)$ with direction $u \rightarrow v$. If $f(u, v) < f(v, u)$, the (undirected) edge (u, v) gets assigned a flow of $f(v, u) - f(u, v)$ with direction $v \rightarrow u$. Otherwise, if $f(u, v) = f(v, u)$, there is no flow over the (undirected) edge (u, v). Clearly, this way we obtain a proper flow in G, as both the capacity and the conservation property are fulfilled (it does not make too much sense to speak of skew symmetry in the undirected case).

Problem 4 (10 credits)

Consider the following network where $c = 10^9$. Use it as an example to illustrate one of the shortcomings of the Ford-Fulkerson algorithm.



Solution.

Since Ford-Fulkerson always chooses an arbitrary augmenting path, it may happen that each of them contains the edge connecting v_1 and v_2. This way, the flow would increase just by one each phase. Since the maximum flow is as large as $c = 10^9$, the algorithm would need $2c = 2 \cdot 10^9$ phases (every other phase all four outer edges would have their flow increased by one, because we would alternatively have $s \rightarrow v_2 \rightarrow v_1 \rightarrow t$ and $s \rightarrow v_1 \rightarrow v_2 \rightarrow t$ as augmenting paths).