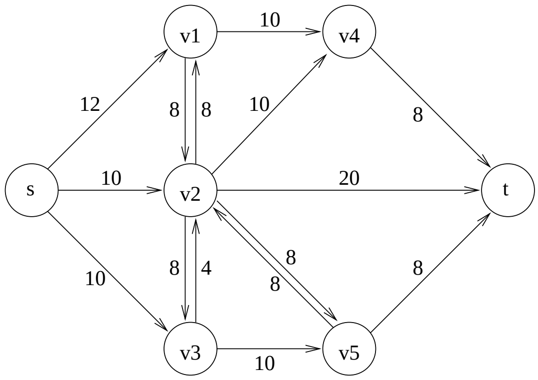
Technische Universität München
Department of Computer Science
Chair for Efficient Algorithms
Prof. Dr. Sami Khuri

WS 1999/2000
Solutions 3
May 30, 2000

Efficient Algorithms and Data Structures II

Problem 1 (8 credits)

Use the preflow-push algorithm to produce a maximum flow for the following network.



Solution.

See Figure 1.

Problem 2 (12 credits)

The *edge connectivity* of an undirected graph is the minimum number k of edges that must be removed to disconnect the graph. For example, the edge connectivity of a tree is 1, and the edge connectivity of a cyclic chain of vertices is 2. Show how the edge connectivity of an undirected graph $G = (V, E)$ can be determined by running a maximum-flow algorithm on at most $|V|$ flow networks, each having $O(V)$ vertices and $O(E)$ edges.

Solution.

Let $u, v \in V$, let G_{uv} denote the directed version of G with u as source and v as sink and all edge capacities equal to one. Let f_{uv} denote a maximum flow in G_{uv} (from u to v), let k denote the edge connectivity of G. Then for any $u \in V$, $k = \min\{|f_{uv}| : v \in V - \{u\}\}$.

First we show that $k \geq \min\{|f_{uv}| : v \in V - \{u\}\}$. Let $m = \min\{|f_{uv}| : v \in V - \{u\}\}$. Suppose we remove only $m - 1$ edges from G. For any vertex v, by the max-flow min-cut theorem, u and v are still connected. (The max-flow from u to v is at least m, hence any cut separating u from v has capacity at least m, which means at least m edges cross any such cut. Thus at least one edge is left crossing the cut when we remove $m - 1$ edges.) Thus evert node is connected to u, which implies that the graph is still connected. So at least m edges must be removed to disconnect the graph, i.e., $k \geq \min\{|f_{uv}| : v \in V - \{u\}\}$.

On the other hand, consider a v with the minimum $|f_{uv}|$. By the max-flow min-cut theorem, there is a cut of capacity $|f_{uv}|$ separating u and v. Since all edge capacities are one, exactly $|f_{uv}|$ edges cross this cut. If these edges are removed, there is no path from u to v and so our graph becomes disconnected. Hence $k \leq \min\{|f_{uv}| : v \in V - \{u\}\}$.

Now we can find k as follows.

Select any vertex $u \in V$
for each vertex $v \in V - \{u\}$ **do** // $|V| - 1$ iterations
    set up the flow network G_{uv} as described above
    find the maximum flow f_{uv} on G_{uv}
return the minimum of the $|V| - 1$ max-flow values

Problem 3 (20 credits)

Let $G = (V, E)$ be a flow network with source s and sink t, and suppose that each edge $e \in E$ is restricted to capacity $c(e) = 1$. Assume also, for convenience, that $|E| = \Omega(V)$.

a. Suppose we run the basic Ford-Fulkerson maximum-flow algorithm on G. What is the worst-case running time? Justify briefly.

   (Hint: the total running time of Ford-Fulkerson is at most $O(E \cdot |f^*|)$ in general, where f^* is the maximum flow found by the algorithm. But here, with the stated restriction, we have a tighter bound.)

b. Suppose a maximum flow for G has been computed, and a new edge with capacity 1 is added to E. Describe how the maximum flow can be efficiently updated. Analyze your algorithm.

   (Hint: it is not only the value of the flow that must be updated, but the flow itself.)

c. Suppose a maximum flow for G has been computed, but an edge is now removed from E. Describe how the maximum flow can be efficiently updated. Analyze your algorithm.

Solution.

a. The total running time of Ford-Fulkerson is at most $O(E \cdot |f^*|)$. But here $|f^*| = O(V)$, because the flow is bounded by the capacity of *any* cut, and the capacity of the cut $(s, V - s)$ is $O(V)$ because there are $O(V)$ edges leaving source s and each edge has capacity one. Hence, the worst-case running time in this case is $O(E \cdot |f^*|) = O(E \cdot O(V)) = O(E \cdot V)$.

b. Just execute one additional iteration of Ford-Fulkerson. The new edge in E adds a new edge to the residual graph, so look for an augmenting path and update the flow if a path is found.

   The running time is $O(V + E)$, whether you find this path by DFS or BFS.

   Only one iteration is needed because adding an edge of capacity one increases the capacity of the min-cut by at most one. Thus the max-flow (which equals the capacity of the min-cut) increases by at most one. Since every edge has capacity one, any augmentation increases the flow by one, so only at most one augmentation is needed.

c. Let the removed edge be (u, v). If (u, v) had no flow, we don't need to do anything. If (u, v) carried flow, the network must be updated. The picture is something like:

$$s \longrightarrow \cdots \longrightarrow u \Rightarrow v \longrightarrow \cdots \longrightarrow t$$

where the edge "$\Rightarrow$" is deleted. There is now one more unit of flow coming into u than is going out (and one more unit of flow going out of v than is coming in). The idea is to try to reroute this unit of flow so that is goes out of u and into v via some other path, if possible. If this is not possible, we must reduce the flow from s to u and from v to t by one unit.

Look for an augmenting path from u to v (not from s to t as we do at every iteration of the basic Ford-Fulkerson method). If there is such a path, augment the flow along that path. If there is no such path, reduce the flow from s to u by augmenting the flow from u to s. That is, find an augmenting path from u to s and augment the flow along that path. (There is definitively such a path because by assumption there is flow from s to u.) Augment the flow from t to v similarly.

The running time is again $O(V + E)$, whether you find two paths, from u to s and from t to v, by DFS or BFS.

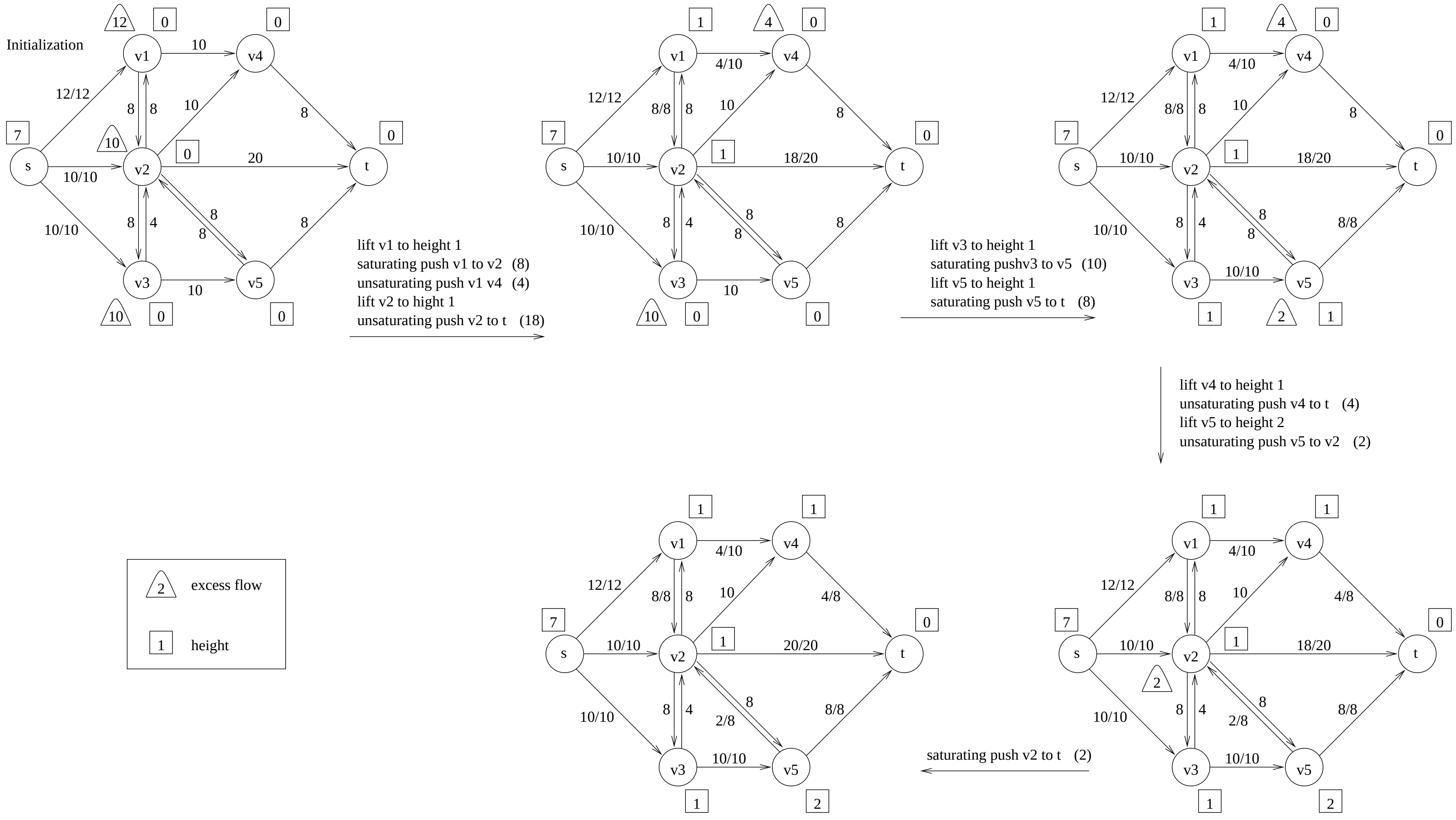


Figure 1: Solution to problem 1