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

WS 1999/2000
Solutions 5
June 13, 2000

Efficient Algorithms and Data Structures II

Problem 1 (30 credits)

Let $G = (V, E)$ be a flow network woth source s and sink t, and an integer capacity $c(u, v)$ on each edge $(u, v) \in E$. Let $C = \max\{c(u, v) : (u, v) \in E\}$.

a. Argue that a minimum cut of G has capacity at most $C|E|$.

b. For a given number K, show that an augmenting path of capacity at least K can be found in $O(|E|)$ time, if such a path exists.

The following modification of FORD-FULKERSON can be used to compute a maximum flow in G.

```
MAX-FLOW-BY-SCALING(G, s, t)
1   C ← max{c(u, v) : (u, v) ∈ E}
2   initialise flow f to 0
3   K ← 2^⌊log C⌋
4   while K ≥ 1 do
5       while there exists an augmenting path p with capacity at least K do
6           augment flow f along p
7       done
8       K ← K/2
9   done
10  return f
```

c. Argue that MAX-FLOW-BY-SCALING returns a maximum flow.

d. Show that the residual capacity of a minimum cut of G is at most $2K|E|$ each time line 4 is executed.

e. Argue that the inner **while** loop of lines 5–7 is executed $O(|E|)$ times for each value of K.

f. Conclude that MAX-FLOW-BY-SCALING can be implemented to run in $O(E^2 \log C)$ time.

Solution.

a. The capacity of a cut is defined to be the sum of the capacities of the edges crossing it. Since the number of such edges is at most $|E|$, and the capacity of each edge is at most C, the capacity of *any* cut of G is at most $C|E|$.

b. The capacity of an augmenting path is the minimum capacity of any edge on the path, so we are looking for an augmenting path whose edges *all* have capacity at least K. Do a BFS search or DFS search as usual to find the path, considering only edges with residual capacity at least K (pretend the lower-capacity edges don't exist). This takes $O(V + E) = O(E)$ time. (Note that $|V| = O(E)$ in a flow network).

c. MAX-FLOW-BY-SCALING uses the Ford-Fulkerson method. It repeatedly augments the flow along an augmenting path until there are no augmenting paths of capacity ≥ 1. Since all the capacities are integers, and the capacity of an augmenting path is positive, this means that there are no augmenting paths whatsoever in the reidual graph. Thus, by the max-flow min-cut theorem, MAX-FLOW-BY-SCALING returns a maximum flow.

d. The first time line 4 is executed, the capacity of any edge in G_f equals its capacity in G, and by part (a) the capacity of a minimum cut of G is at most $C|E|$. Initially, $K = 2^{\lfloor \log C \rfloor}$, hence $2K = 2 \cdot 2^{\lfloor \log C \rfloor} = 2^{\lfloor \log C \rfloor + 1} > 2^{\log C} = C$. So the capacity of a minimum cut of G is initially less than $2K|E|$.

   The other times line 4 is executed, K has just been halved, so the capacity of a cut of G_f is at most $2K|E|$ at line 4 if and only if that capacity was at most $K|E|$ when the **while** loop of lines 5–7 last terminated. So we want to show than when line 8 is reached, the capacity of a minimum cut of G_f is at most $K|E|$.

   Let G_f be the residual network when line 8 is reached.

   there is no augmenting path of capacity $\geq K$ in G_f
   $\Rightarrow$ max flow f' in G_f has value $|f'| < K|E|$
   $\Rightarrow$ min cut in G_f has capacity $< K|E|$

e. By part (d), when line 4 is reached, the capacity of a min cut of G_f is at most $2K|E|$, and the max flow in G_f is at most $2K|E|$. Recall that, given a flow f in G, every flow f' in G_f induces a flow $f + f'$ in G; the reverse claim, that every flow $f + f'$ in GG induces a flow f' in G_f, is proved in a similar manner. Together, these two claims show that the value of the maximum flow in G equals the value of the current flow in G plus the value of the maximum flow in G_f. Therefore the maximum flow in G is at most $2K|E|$ more than the current flow in G. Every time the inner **while** loop finds an augmenting path of capacity at least K, the flow in G increases by $\geq K$. Since the flow cannot increase by more than $2K|E|$, the loop executes at most $2K|E|/K = 2|E|$ times.

f. The time complexity is dominated by the **while** loop of lines 4–9. (The lines outside the loop take $O(E)$ time.) The outer loop executes $O(\log C)$ times, since K is initially $O(C)$ and is halved on each iteration, until $K < 1$. By part (e), the inner loop executes $O(E)$ times for each value of K, and by part (b), each iteration takes $O(E)$ time. So the total time is $O(E^2 \log C)$.

Problem 2 (10 credits)

Which of the following statements are true and which are false? Justify your answers by providing either a proof or a counterexample.

a. If f is a maximum flow, then either $f(e) = 0$ or $f(e) = c(e)$ for every edge $e \in E$ of a network $G = (V, E)$ ($c(e)$ denotes the capacity of edge e).

b. For every network there is a maximum flow f, for which either $f(e) = 0$ or $f(e) = c(e)$ for every edge $e \in E$.

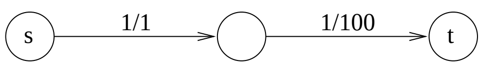
c. If the capacities of all edges of a network are pairwise disjoint, then there is a unique minimum cut.

d. If in a network every directed edge is replaced with an undirected one, then the value of the maximum flow remains the same.

e. If you add (multiply) a positive number λ to the capacity of every edge, then the value of a minimum cut remains the same.

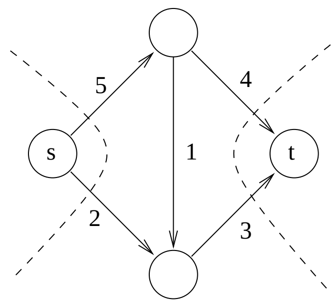
Solution.

a. Wrong. Counterexample:

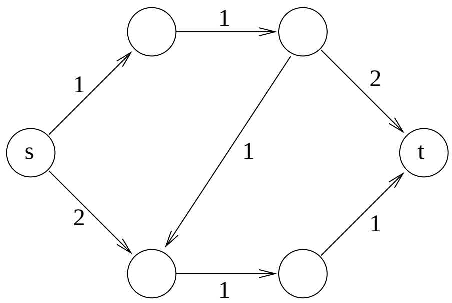


b. Wrong. Counterexample see (a).

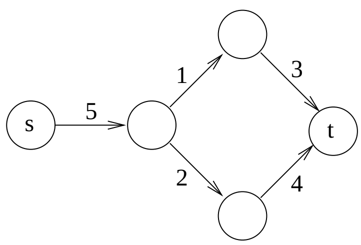
c. Wrong. Counterexample:



d. Wrong. Counterexample:



e. Adding $\lambda > 0$ to each capacity can change the value of a minimum cut. Example (add 20 to each capacity):



Multipling by $\lambda > 0$ clearly does change the value of a minimum cut (unless $\lambda = 1$); actually the value gets multiplied by λ (note that the minimum cuts themselves remain the same, just the value changes).