

## Solutions to assignment 4

**Exercise 1**

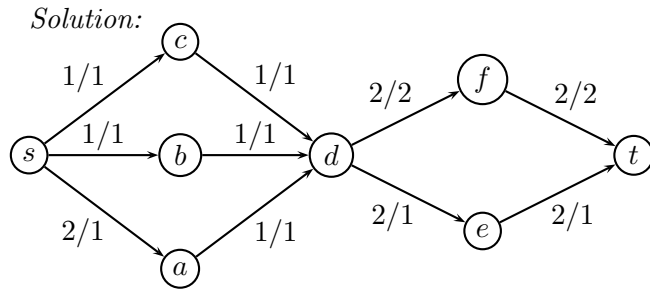
A spammer is located at one node  $q$  in a undirected communication network  $G$  and peaceful email users are located at nodes denoted by the set  $S$ . Let  $c_{uv}$  denote the effort required to install a spam filter for the network edge  $(u, v)$ . The problem is to determine the minimal effort required to isolate the spammer from the email users using the spam filters. Find an efficient polynomial time algorithm to solve this problem.

*Solution:* The communication network can be seen as a capacity-graph. The network participants are the nodes, the connections are the edges and the spam blocking efforts are the edge capacities. We interpret the spam mails as a flow through the network, the start node of which is  $q$ . We create an artificial target node  $t$ , which we connect to all nodes of  $S$ . We calculate the sum  $K$  of all capacities  $c_{uv}$ . We set the capacities of the edges between  $S$  and  $t$  to  $K + 1$ . The goal is to find a minimal  $q$ - $t$ -cut: If we block all edges leaving this cut, we get a total isolation of  $q$  with a minimal effort. In order to find this cut, we first calculate a maximum flow (e.g. by the Highest-Level-Preflow-Push-Algorithm in time  $\mathcal{O}(n^2 \cdot \sqrt{m})$ ). Then we create the residual network (in time  $\mathcal{O}(m)$ ) and look at all nodes reachable from  $q$  (e.g. by breadth-first-search in time  $\mathcal{O}(n + m)$ ). By the Min-Cut-Max-Flow-Theorem, we know that the set of nodes reachable from  $q$  constitutes a minimal cut. The cut does not contain any node of  $S$ : Suppose the cut contains a node  $v \in S$ . Since we set  $cap((v, t)) = K + 1$ , there will always be the edge  $(v, t)$  in the residual network, because no possible flow can use up this capacity. Hence, if  $v$  is reachable from  $q$ , then so is  $t$ . Hence,  $t$  would belong to the cut, which is a contradiction. Now, we block all edges leaving the cut ( $\mathcal{O}(m)$ ) and the spammer is isolated. The total time needed is  $\mathcal{O}(n^2 \cdot \sqrt{m})$ .

**Exercise 2**

We define a *most vital arc* of a network as an arc whose deletion causes the largest decrease in the maximum  $s$ - $t$ -flow value. Let  $f$  be an arbitrary maximum  $s$ - $t$ -flow. Either prove the following claims or show through counterexamples that they are false:

- A most vital arc is an arc  $e$  with the maximum value of  $c(e)$ .
- A most vital arc is an arc  $e$  with the maximum value of  $f(e)$ .
- A most vital arc is an arc  $e$  with the maximum value of  $f(e)$  among arcs belonging to some minimum cut.
- An arc that does not belong to some minimum cut cannot be a most vital arc.
- A network might contain several most vital arcs.



This is a counterexample for (a), (b) and (d) and an example for (e). The specified flow  $f$  is a maximum flow with the value three. The deletion of *any* edge decreases the flow value by one. Hence, each edge is a *most vital arc* ( $\rightsquigarrow$  (e) is true). The edge  $e = (s, b)$  has neither the maximum value of  $c(e)$  nor the maximum value of  $f(e)$ ; still, it is a *most vital arc* ( $\rightsquigarrow$  (a) and (b) are false). The edge  $(s, a)$  does not belong to any minimum cut; still, it is a *most vital arc* ( $\rightsquigarrow$  (d) is false).

(c) is true. The maximum flow value is equal to the value of the flow through any minimum cut. When we remove an edge  $e$  from a minimum cut, the value of the flow through the minimum cut and thus the maximum flow value decrease by  $f(e)$ . Hence, a *most vital arc*  $e$  must have the maximum value of  $f(e)$  among arcs belonging to some minimum cut because, if there was another edge  $e'$  with  $f(e') > f(e)$  in the same minimum cut, the deletion of  $e'$  would cause a larger decrease than the deletion of  $e$  so that  $e$  could not be a *most vital arc*.

### Exercise 3

*Dining Problem.* Several families go out to dinner together. To increase their social interaction, they would like to sit at tables so that no two members of the same family are at the same table. Show how to formulate finding a seating arrangement that meets this objective as a maximum flow problem. Assume that the dinner contingent has  $p$  families and that the  $i$ th family has  $a(i)$  members. Also assume that  $q$  tables are available and that the  $j$ th table has a seating capacity of  $b(j)$ .

*Solution*

The capacitated graph  $G(V, E)$  is defined as:

$$\begin{aligned} V &= \{s, t\} \cup \{u_i \mid 1 \leq i \leq p\} \cup \{v_j \mid 1 \leq j \leq q\} \\ E &= \{(s, u_i) \mid 1 \leq i \leq p\} \cup \{(u_i, v_j) \mid 1 \leq i \leq p, 1 \leq j \leq q\} \cup \{(v_j, t) \mid 1 \leq j \leq q\} \\ \text{cap}(s, u_i) &= a(i) \\ \text{cap}(u_i, v_j) &= 1 \\ \text{cap}(v_j, t) &= b(j) \end{aligned}$$

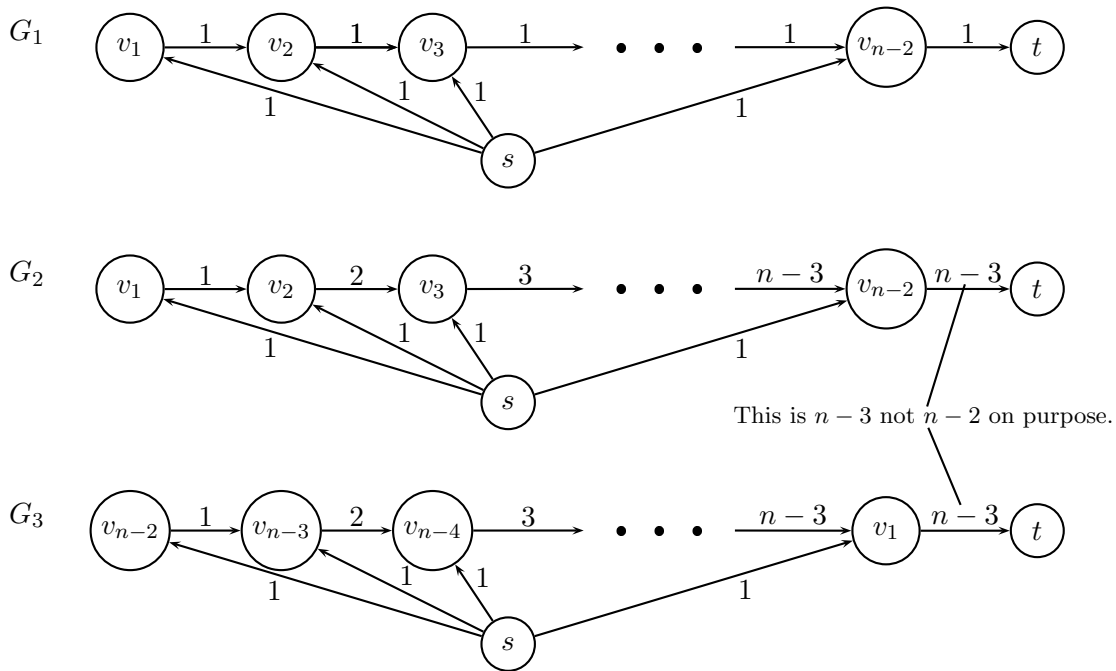
An integral solution to the maxflow problem gives an assignment of families to tables. A flow from  $u_i$  to  $v_j$  is either zero or one, with one meaning that a member of family  $i$  sits in table  $j$ . The maximum flow maximizes the number of people that can be seated under the stated constraints.

### Exercise 4

In this exercise we analyze the running time of different variations of the preflow-push algorithm with the highest-level selection rule. Assume that ties are broken as follows:

- Of active nodes on the highest level the one with the smallest number is selected first.
- Of eligible edges the one leading to the largest numbered node is selected first.  
(Would a different ordering of the eligible edges change your answer in any of the cases?)

Consider the following capacitated graphs:



Give the asymptotic running time (total number of pushes and relabelings) of the following three heuristics for all three graphs. Justify your answer. (Hint: The running time is  $\Theta(n)$  or  $\Theta(n^2)$  in all cases.)

- (a) *Local relabeling*. When relabeling a node  $v$ , the new level is computed as  $d(v) = 1 + \min\{d(w) \mid (v, w) \in G_f\}$ .

*Solution*

$\Theta(n)$  for  $G_1$ ,  $\Theta(n^2)$  for  $G_2$  and  $G_3$ .

- (b) *Global relabeling*. In the beginning and after every  $m$  ( $= 2n - 4$  in this case) operations, the distance labels are recomputed as follows:

$$d(v) = \begin{cases} \mu(v, t) & \text{if there is a path from } v \text{ to } t \text{ in } G_f \\ n + \mu(v, s) & \text{if there is a path from } v \text{ to } s \text{ but not to } t \text{ in } G_f \\ 2n - 1 & \text{otherwise} \end{cases}$$

Here  $\mu(v, w)$  is the smallest number of edges on a path from  $v$  to  $w$ .

*Solution*

$\Theta(n^2)$  for  $G_1$ ,  $\Theta(n)$  for  $G_2$  and  $G_3$ .

- (c) *Gap heuristic*. When a level  $\ell$  becomes empty, all nodes on the levels  $\ell + 1$  to  $n - 1$  are lifted to level  $n$ .

*Solution*

$\Theta(n)$  for  $G_1$  and  $G_2$ ,  $\Theta(n^2)$  for  $G_3$ .

An explanation for  $G_3$ :

We start running the algorithm. In the first step we push one unit of flow through all the outgoing from  $s$  edges. Now the excess of the nodes  $v_1, v_2, \dots, v_{n-2}$  is exactly 1. According to the rules we start choosing the active nodes on the highest level with the smallest index number. The node  $v_1$  will be chosen first. The distance label of all the nodes except  $s$  is zero, so we do not have any eligible edges outgoing from  $v_1$ . Hence, we recalculate the distance label of  $v_1$  increasing it by one. After this procedure we didn't get any gaps but we got an eligible edge  $(v_1, t)$ . We push one unit of flow through this edge. The next node

with positive excess is  $v_2$ . The current distance of  $v_2$  is zero. To get an eligible edge we have to increase the distance label twice. And again we don't get any gaps. We send a unit of flow from  $v_2$  to  $t$ . We continue this procedure  $n - 3$  times (because of the capacity of the edges). In the end we reach the node  $v_{n-2}$  with excess one. We repeat the distance relabelling operation  $n - 2$  times and send this unit of flow to  $v_{n-3}$ . The only outgoing edges from  $v_{n-3}$  are  $(v_{n-3}, v_{n-2})$  with distance  $n - 2$  and  $(v_{n-3}, s)$  with distance  $n$ . Current distance label of  $v_{n-3}$  is  $n - 3$ . We increase it once and get a gap. Hence the labels of the nodes  $v_{n-2}$  and  $v_{n-3}$  become  $n$ . We increase the label of the node  $v_{n-2}$  once again and after this procedure we have two eligible edges  $(v_{n-2}, v_{n-3})$  and  $(v_{n-2}, s)$ . According to our rule that in every step we choose the eligible edge with the largest numbered index, we push the flow from  $v_{n-2}$  to  $v_{n-3}$  and after increasing the distance label of  $v_{n-3}$  by one we have the possibility to send this last unit of flow back to the source. The procedure described above takes  $\Theta(n^2)$  time.