Exam problems for the course 'Network Programming' (DM817) Part A

Department of Mathematics and Computer Science
University of Southern Denmark

The problems are published on Tuesday October 6, 2020. The solutions must be returned by Tuesday November 3, 2020 at 16.00
You must hand in via SDU assignment in blackboard. You must specify your name and the first 6 digits of your cpr number on the first page.

It is important that you explain how you obtain your answers and argue why they are correct. If you are asked to describe an algorithm, then you must supply enough details so that a reader who does not already know the algorithm can understand it (but you should not give pseudo code). You should also give the complexity of the algorithm when relevant. Note also that illustrating an algorithm means that one has to follow the steps of the algorithm meticulously (slavisk). When you are asked to give the best complexity you can find for a problem, you must say which (flow) algorithm you use to get that complexity and give a reference to its complexity (in the book) or prove it directly. All algorithms asked for should be polynomial in the size of the input. Unless otherwise stated the numbers n and m always denote the number of vertices and arcs respectively of the network in question. Note that we use '$\diamond$' to denote the end of a (sub)question.

There are 100 points to earn in total for this part A of the problems. The second part B which will be posed in the beginning of December. The final grade for the course will be based on an overall impression of your performance on both sets of problems.

You may refer to results from Ahuja and Bang-Jensen and Gutin as well as results from exercises that have been posed on the weekly notes, but not to material found elsewhere (this includes exercises that we have not done in the course). It is strictly forbidden to work in groups and any exchange of ideas and results before the deadline for handing in will be considered as exam fraud.

PROBLEM 1 (15 point)

Let $\mathcal{N} = (V, A, \ell \equiv 0, u)$ be the network in Figure 1. In the figure two flows x, x' are indicated.

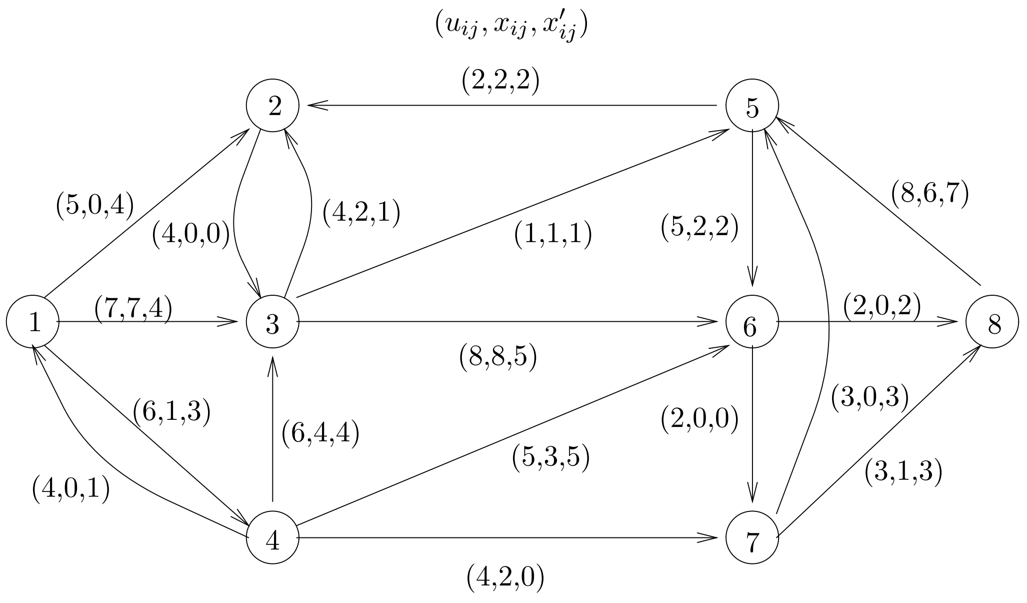

Figure 1: A network $\mathcal{N}$ and two flows x, x' in $\mathcal{N}$. The data on each arc is $(u_{ij}, x_{ij}, x'_{ij})$.

Question a:

Give the values of the balance vectors b_x and $b_{x'}$ for x respectively x' and explain (by describing, in words, an algorithm that works for any such pair of flows) how one can find a flow $\bar{x}$ in $\mathcal{N}(x)$ (the residual network wrt x) such that $x' = x \oplus \bar{x}$. You must give the definition of '$\oplus$'. You should also illustrate your algorithm on the flows x, x'.

Question b:

Let $\mathcal{N}$ be a network with n vertices and m arcs. Explain briefly how to decompose a given flow y in $\mathcal{N}$ into at most $n + m$ path and cycle flows. Illustrate the algorithm on the flow x in Figure 1. You should NOT draw a new figure for each step, but instead give a few steps and then show a list of paths and cycle flows (including their values) which form a decomposition of x.

Problem 2 (20 point)

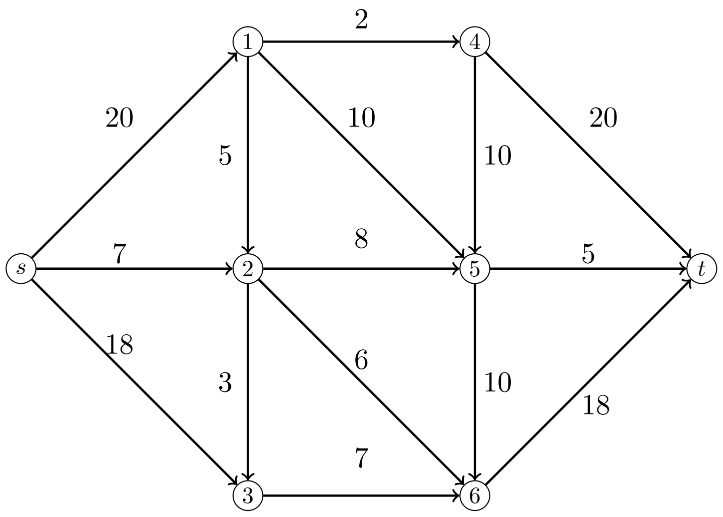

Figure 2: The network $\mathcal{N} = (V \cup \{s,t\}, A, \ell \equiv 0, u)$

Question a:

Illustrate the shortest augmenting path method for finding a maximum flow on the network $\mathcal{N}$ in Figure 2. To ease the verification, you should choose the paths in lexicographic order according to the vertices that appear on the paths (e.g., $s \to 1 \to 5 \to t$ is before $s \to 2 \to 5 \to t$)

Question b:

Give a short but sufficient description of the preflow push algorithm. Remember to say how you initialize the flow and other functions involved in the algorithm. You are not supposed to argue why the algorithm is correct.

Question c:

Illustrate the algorithm by running it on the network in Figure 2. In order to make your solution easy to check you should follow the following rules:

- Always select the smallest numbered vertex when selecting a vertex i with $b_x(i) < 0$ and continue doing operations on this vertex until it becomes balanced.
- If there are several possible arcs ij for which the push operation may be performed, you should choose j such that its number is as small as possible.

You may list a group of operations concerning the same vertex in one line, e.g. push 3 units from 2 to 4, push 4 units from 2 to 5, lift 2 to height 5, push 3 units from 2 to 7 etc. and you only have to show the updated flow after such a group of operations.

Problem 3 (15 point)

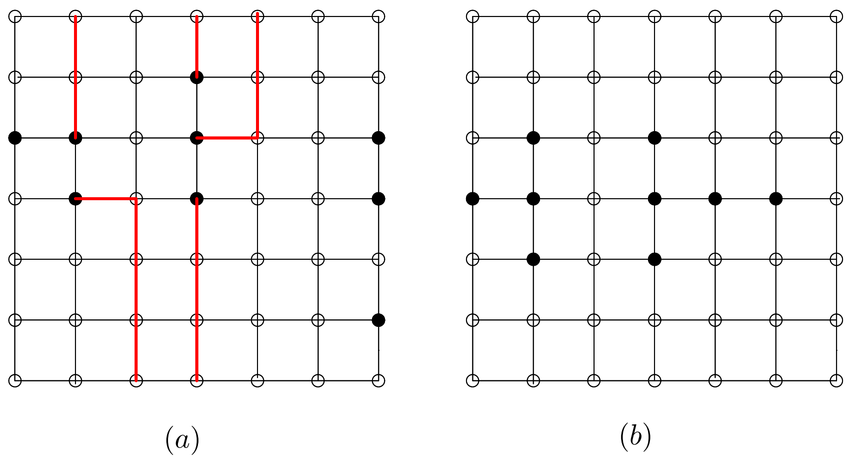

Figure 3: Two 7×7 grids with special vertex sets Y marked in black. In (a) there exist a set of escape routes, while no such set of routes exist in (b).

An $q \times q$ **grid** is an undirected graph, $G_{q,q}$, with q^2 vertices denoted $v_{1,1}, \ldots, v_{1,q}, v_{2,1}, \ldots, v_{q,1}, \ldots, v_{q,q}$ and the edges formed by the q paths "horizontal" paths $v_{1,1}v_{1,2} \ldots v_{1,q}, \ldots v_{q,1}v_{q,2} \ldots, v_{q,q}$ and the q "vertical" paths $v_{1,1}v_{2,1} \ldots v_{q,1}, \ldots, v_{1,q}v_{2,q} \ldots, v_{q,q}$. See Figure 3. We call the subset $V' = \{v_{i,j} : \min\{i,j\} = 1 \text{ or } \max\{i,j\} = n\}$ the **edge** of $G_{q,q}$.

Suppose that we are given $r \leq q^2$ distinct vertices $v_{i_1,j_1}, \ldots, v_{i_r,j_r}$ of $G_{q,q}$. The goal is to decide whether there exist r vertex disjoint paths $P_1, \ldots, P_r$ so that P_k starts in v_{i_k,j_k} and ends in a vertex belonging to the edge of the grid. This problem, which we call the **escape problem**, is related to VLSI design and several other important problems.

Question a:

Show how to model the escape problem as a flow problem. You must argue that your model is correct and explain how one can obtain a solution (when one exists) from a solution to your flow problem. You should also show how to find a certificate for the non-existence of the desired paths when they do not exist. Give the best complexity you can obtain for the problem.

Question b:

Now we consider the following generalization of the escape problem: Find a collection of paths $P_1, \ldots, P_r$ from Y to V' so that the maximum number of paths intersecting in the same vertex is minimized. So if there is a solution to the original escape problem, then this number will be zero. Explain how to solve this generalization of the escape problem in polynomial time.

Question c:

Suppose none of the vertices $Y = \{v_{i_1,j_1}, \ldots, v_{i_r,j_r}\}$ are on the edge of $G_{q,q}$ (so $Y \cap V' = \emptyset$) and we wish to find a minimum cardinality set X of vertices of $G_{q,q}$ so that $X \cap Y = \emptyset$ and

$G_{q,q} - X$ has no path from Y to V' (so every path from Y to the edge of $G_{q,q}$ intersects X). Note that we are not allowed to delete any vertex of Y and the set X always exists, since we could remove the $4(q-1)$ vertices on the edge of $G_{q,q}$. Explain how to solve this problem in polynomial time using flows. Hint: use a modified version of vertex-splitting and suitable arc capacities.

Problem 4 (20 point)

For each of the following claims you should argue why the claim is true. You may do this by a short argument, e.g. using results from exercises that we have done or results from Ahuja and Bang-Jensen+Gutin but you should supply enough details so that the validity of your claim is easy to check.

1. One can find a minimum path cover in any acyclic digraph by solving a constant number of maximum flow problems.
2. The buildup algorithm runs in polynomial time on unit capacity networks.
3. One can check the existence of a feasible flow in any unit capacity network $\mathcal{N} = (V, A, \ell \equiv 0, u \equiv 1, b)$ in time $O(n^{\frac{2}{3}}m)$.
4. A digraph $D = (V, A)$ has a cycle factor (a spanning collection of vertex-disjoint cycles) if and only if there is no subset $X \subseteq V$ such that X is independent (no arcs inside X) and we can kill all paths from X to itself by deleting less than $|X|$ vertices.
5. Let x be a feasible flow in $\mathcal{N} = (V, A, \ell \equiv 0, u, b, c)$. Suppose that $\mathcal{N}(x)$ has precisely one negative cycle W and that the cost of W is -10. Assume also that the minimum residual capacity of an arc of W is 5, so $\delta(W) = 5$. Then the flow $x' = x \oplus \delta(W)$ that we obtain by adding a cycle flow of value $\delta(W)$ along W in $\mathcal{N}(x)$ to x is a minimum cost flow in $\mathcal{N}$ and its cost is $cx' = cx - 50$.

Problem 5 (15 point)

In the present situation under Covid-19 the health authorities need all the help they can get to maximize the capacity of the Corona test centers. Hence you have decided to use your DM817 skills to help and you have gotten the task of devising an algorithm that can be used for assigning the bookings to a test center to the different test teams in the center. We assume that our test center has the capacity to run K parallel lines where people are tested (in separate tents but in the same geographical location.) and that you are given a large set $\mathcal{S}$ of bookings for the next day and your task is to schedule as many of these to the K test tents as possible. Each booking is for a specific time, e.g 16.04 and we assume that each test tent can handle two different bookings if they are at least 5 minutes apart.

Question a:

Explain how the problem of assigning bookings to test tents can be modeled as a path-covering problem in an acyclic digraph D. Explain how one can see from this model whether it is possible to handle all the bookings and describe a flow based method for finding a good assignment or a certificate that no solution exists. $\diamond$

Suppose your algorithm determines that it is not possible to handle all the bookings, due to too little capacity, either because there are to few tents or because it takes too long before a new person can be tested in the same tent (recall that there must be 5 minutes between consecutive tests in the same tent).

Question b:

First explain how to determine the minimum number K' of test tents (lines) that are needed to process all the bookings (test all the persons who have a booking).

Question c:

Explain how you can augment your model so that you can determine the maximum number of bookings can be handled with the present capacity. Hint: use minimum cost flows. What is the complexity of your algorithm?

Question d:

Explain how you can use the buildup algorithm for minimum cost flows to solve the problem of determining the maximum number of bookings you can handle if you open up i new tents for each $i = 1, 2, \ldots, K' - K$. Express the complexity of your algorithm in terms of $|\mathcal{S}|$, the number of pairs of bookings that may be handled in the same tent and K'

Problem 6 (15 point)

The health authorities (especially the minister of health) are impressed by your skills in helping with the test center above, so they get the idea of asking you to optimize the test center by devising an algorithm for constructing many disjoint test teams. Here we assume that each test tent is run by two persons, one of type A and the other of type B, where the type designates what the person is supposed to do during a test. No person is both of type A and type B and we assume that we are given a set V_A of n persons of type A, a set V_B of n persons of type B as well as a list of possible pairs ab where $a \in V_A, b \in V_B$ who are qualified to work together in a tent (how this is determined is not our concern). A collection $a_1b_1, \ldots, a_pb_p$ of pairs is **good** if $a_i \neq a_j$ and $b_i \neq b_j$ when $i \neq j$ (no person chosen is in more than one pair).

Question a:

Explain how to model the problem of selecting a good collection of maximum size as a bipartite matching problem and describe a flow based algorithm for finding such a maximum collection in polynomial time. You should argue why your algorithm is correct and give the running time of your algorithm in terms of n and m, where m is the number of possible pairs. You should also explain how one can certify that the collection returned by the algorithms is indeed of maximum size. $\diamond$

Suppose that the maximum size, p, of a good collection is too small (the authorities want to be able to run more than p tents in parallel) so you are given a set V'_A of k new persons of type A and a set V'_B of k new persons of type B and you must try to use these together with the persons you already have to find a good collection which is larger than p. Of course, if there are possible pairs among the new persons, you can do this trivially, but even if there are no such pairs, it may be possible to find a good collection which is larger then p.

Question b:

Suppose that k is much smaller than p. Explain how one can find a new good collection of maximum size (much) faster than running the algorithm from Question a again on the sets $V_A \cup V'_A, V_B \cup V'_B$ and the possible pairs among these. Hint: show that you can find a good collection of maximum size in time $O(k(n' + m'))$, where $n' = n + k$ and $m' \leq m + 2kn + k^2$.

Question c:

Assume that before you were told that the number p of good pairs that you found was not sufficient, you have already communicated the solution to the persons involved in the pairs. In order to minimize confusion you should thus try to find a new maximum good collection which shares as many pairs as possible with the collection of size p that you have already found. Explain how this can be formulated as a minimum cost flow problem.