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

Department of Mathematics and Computer Science
University of Southern Denmark

The problems are available as of Wednesday March 21, 2018. The solutions must be returned by Monday April 16, 2018 at 14.00
You should hand in electronically on Blackboard and also place 2 copies of your report in Jørgen Bang-Jensen's mailbox in the mail room next to IMADA's library. Alternatively you may hand them in at the lecture on April 16.

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 do not have to 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 (danish: 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.

When you illustrate an algorithm it is OK to hand in a (readable) scan of hand drawn figures, showing how your algorithm works instead of spending a lot of time in making nice figures in LaTeX.

When the numbers n and m are used in the text, these always denote the number of vertices and arcs respectively of the network/(di)graph in question.

There are 60 points to earn in total for this part A of the problems. The final grade for the course will be based on an overall impression of your performance on this and the second set of problems (to be posed around May 1st).

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 (12 point)

Let $\mathcal{N} = (V, A, \ell \equiv 0, u)$ be the network in Figure 1 with arc capacities as specified on each arc. Note that all lower bounds are zero and there are no costs specified on the arcs.

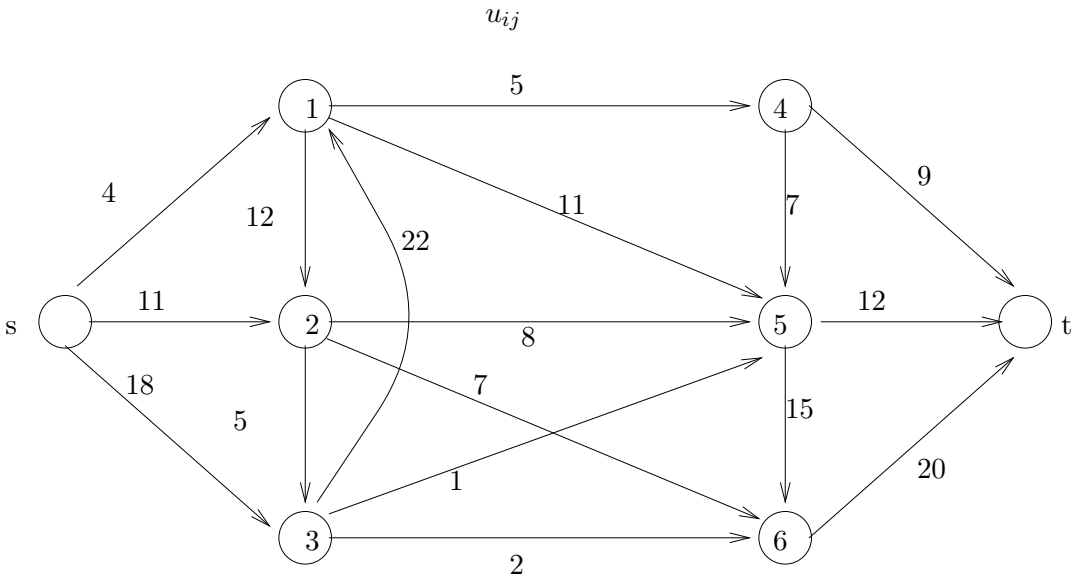

Figure 1: The Network $\mathcal{N}$

Question a:

Give a short description of the capacity scaling algorithm for finding a maximum (s, t)-flow in a network and illustrate the algorithm on the network $\mathcal{N}$. Your description should include a short argument for the complexity of the algorithm and a certificate showing that the final flow is a maximum (s, t)-flow.

Question b:

Give a short description of Dinic's algorithm for finding a maximum (s, t)-flow in a network and illustrate the algorithm on the network $\mathcal{N}$. Your description should include a short argument for the complexity of the algorithm. You may show one blocking show for each iteration as long as you explain how it it was obtained.

PROBLEM 2 (7 point)

Let $\mathcal{N} = (V, A, \ell \equiv 0, u)$ be the network in Figure 2. In the figure two flows x, x' are indicated.

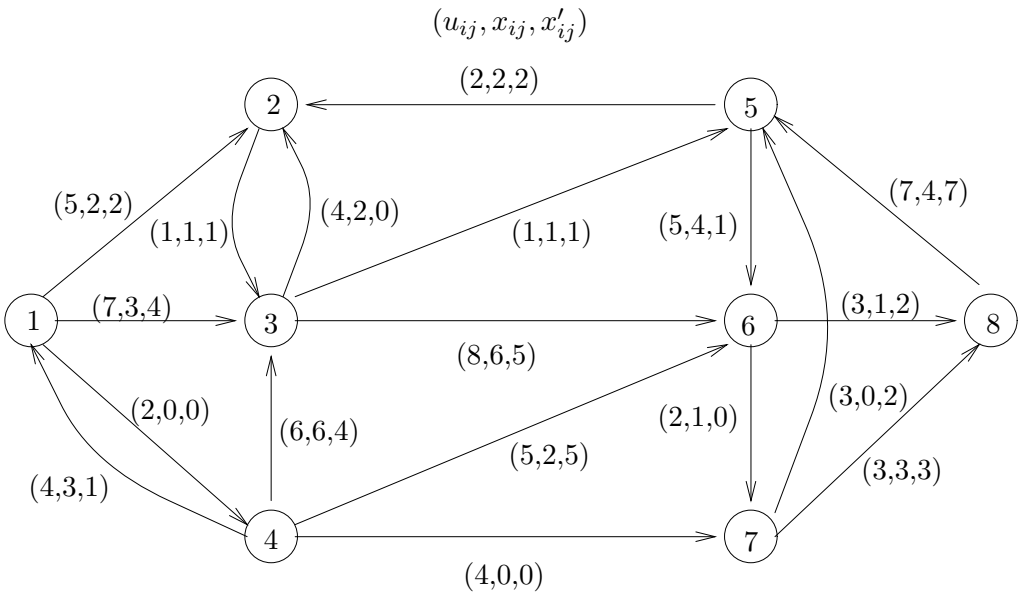

Figure 2: A network $\mathcal{N}$ and two flows x, x' in $\mathcal{N}$

Question a:

Give the values of the balance vectors b_x and $b_{x'}$ for x respectively x' and explain (by describing an algorithm that works for any such pair of flows) how one can find a flow $\bar{x}$ i $\mathcal{N}(x)$ (the residual network wrt x) such that $x' = x \oplus \bar{x}$. You must give the definition of '$\oplus$'. 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 2.

PROBLEM 3 (15 point)

You have been hired by a hotel to handle their room bookings. You were hired because you claim you can solve all their planning problems using your skills from DM817. The hotel has a list of bookings in the form $\mathcal{B} = \{[a_1, d_1], \ldots, [a_n, d_n]\}$, where $[a_i, d_i]$ denotes a booking for a room which starts on day a_i and ends on day $d_i > a_i$. In practice there needs to be some time for cleaning the room etc before the next usage, but we can ignore that here. Two bookings $[a_i, d_i]$ and $[a_j, d_j]$ **overlap** if $a_i < a_j < d_i$ or $a_j < a_i < d_j$. Note that bookings $[a_i, d_i], [a_j, d_j]$ with $a_i < d_i = a_j < d_j$ do not overlap. Overlapping bookings cannot be assigned the same room.

The hotel wants to know the minimum number, K, of rooms needed to satisfy (be able to accept) all the bookings in $\mathcal{B}$.

Question a:

Explain how to solve the problem above as a minimum value (s, t)-flow problem and explain carefully why your model works. You should also give the best possible running time for your algorithm (based on the algorithms you have seen in the course).

The boss is not satisfied with the result (too many rooms seem to be needed) and he wants a certificate (something that he can understand) that shows that the number you return as K in fact is the minimum.

Question b:

Prove that K is equal to the maximum number of bookings from $\mathcal{B}$ that are all pairwise overlapping (every pair of bookings overlap) and describe a flow based method to derive such a subset $\mathcal{B}'$ of K bookings from information that can be obtained when your algorithm has constructed flow which corresponds to a solution to Question a. Hint: design your capacities so that such a set $\mathcal{B}'$ can be found efficiently.

It turns out that K is larger than the number of rooms, R, that are available to cover the bookings, so the boss now wants you to solve the problem of optimizing the number of nights that can be booked, where a booking $[a_i, d_i]$ must either be accepted in full (all days in the same room), or rejected.

Question c:

Show how to solve this problem as a minimum cost flow problem and give the complexity of your algorithm. Remember that your solution should use the fact that it is better to accept a booking of a room for 4 days than to accept one for 3 days.

Question d:

Can your algorithm handle the more general problem of finding the maximum number of bookings that can be made with i rooms for all $1 \leq i \leq R$? If 'yes', explain how and if 'no'

explain how you could modify the algorithm to solve that problem also.

Seeing how skilled you are at solving his problems, the boss now gets an idea: he wants to schedule as many nights as possible from the full set of bookings $\mathcal{B}$ where he now allows bookings to be shortened in both ends and there is a fixed cost C (to the hotel) for shortening a booking by one day. For instance the two bookings $[3, 8], [6, 10]$ overlap, but we can shorten the first to $[3, 6]$ and then they can be in the same room at a cost of $(8 - 6) \cdot C = 2C$. We could also have shortened both as $[3, 7]$ and $[7, 10]$ and again the cost would be $2C$.

Question e:

Explain how to formulate the problem of maximizing the total number of nights that can be booked, when we allow shortenings of booking, by formulating this problem as a minimum cost flow problem.

Problem 4 (16 point)

For each of the following claims you should give a detailed argument for why it is correct.

Claim a:

Let $B = (V_1 \cup V_2, E)$ be a bipartite graph (all edges have the form vv' with $v \in V_1$ and $v' \in V_2$) such that $|V_1| = |V_2| = n$. Suppose the maximum matching in B has size $n - k$ for some integer $0 \leq k \leq n$. Then the following holds: for every set $\emptyset \neq W \subset V_1$ we have $|N(W)| \geq |W| - k$, where $N(W) \subseteq V_2$ is the set of vertices with at least one edge to W. Furthermore, there exists a set $W \subset V_1$ for which we have $|N(W)| = |W| - k$. You should also explain how to find k and a set W as above. Hint: use the maxflow in mincut theorem on a suitable network.

Claim b:

Let $\mathcal{N} = (V, A, \ell \equiv 0, u \equiv 1, b)$ be a unit capacity network and let x be a feasible flow. Then x can be decomposed into at most m path and cycle flows and these are all arc-disjoint. Furthermore the decomposition can be done in time $O(n + m)$.

Claim c:

Let $D = (V, A)$ be a digraph and $k \geq 1$ an integer. There exists a collection of directed cycles $C_1, C_2, \ldots, C_p$, $p \geq 1$ of D such that every arc $ij \in A$ is on at least one and at most k of the cycles if and only if we have $k \cdot d^+(X) \geq d^+(V \setminus X)$ for every non-empty subset X of V. Hint: use Hoffmann's circulation theorem. Recall that $d^+(X)$ is the number of arcs from the set X to $V \setminus X$.

Claim d:

For every fixed constant K, Dinic's algorithm runs in time $O(n^{\frac{2}{3}}m)$ when it is applied to networks of the kind $\mathcal{N} = (V, A, \ell \equiv 0, u, s, t)$ where $\max\{u_{ij} | ij \in A\} \leq K$.

PROBLEM 5 (10 point)

Let $D = (V, A)$ be a digraph and let $X \subseteq V$.

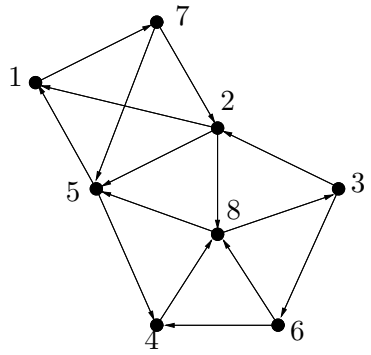

Figure 3:

Question a:

Explain how one can use flows to either find a collection of vertex-disjoint cycles $C_1, \ldots, C_k$, $k \geq 1$ (k is not fixed and may vary), which cover all vertices of X, that is, $X \subseteq V(C_1) \cup \ldots \cup V(C_k)$, or decide that no such set of cycles exists in D for the given X. What is the best complexity you can obtain for your algorithm?

Question b:

Illustrate your algorithm on the digraph in Figure 3 with $X = \{1, 2, 5, 8\}$.

Question c:

Explain how to find disjoint cycles $C_1, \ldots, C_k$, $k \geq 1$ with $X \subseteq V(C_1) \cup \ldots \cup V(C_k)$ and such that the total number of arcs on the cycles is as small as possible. You should also give the best possible complexity you can find for your algorithm.