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

Department of Mathematics and Computer Science
University of Southern Denmark

The problems are handed out Monday April 30, 2018. The solutions must be returned by Tuesday May 22, 2018 at 10.00
You should hand in both via SDU assignment in blackboard (use your name and birth date) and place 2 copies of your report in Jørgen Bang-Jensen's mailbox (dueslag) in our post office (across from David Kyed's office).

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 (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 of the network or digraph in question.

There are 50 points to earn in total for this part B of the problems. The final grade for the course will be based on an overall impression of your performance on both sets of problems. Note that a total of 100 points out of the 110 points in the two problem sets will be considered as a full solution.

You may refer to results from Ahuja and Bang-Jensen and Gutin as well as results from the course page and 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 (10 point)

In this problem graphs and digraphs may have parallel edges and arcs.

A digraph $D = (V, A)$ is **eulerian** if $d^-(x) = d^+(x)$ for every vertex $x \in V$[1]. An **eulerian orientation** of an undirected graph $G = (V, E)$ is an eulerian digraph D that is obtained from G by assigning an orientation to each edge of G. It is a well known fact that a connected undirected graph G has an eulerian orientation if and only if the degree of every vertex in G is even.

Suppose now that we are given a digraph $D = (V, A)$ which is not eulerian but satisfies that $d^-(x) + d^+(x)$ is even for all $x \in V$. Then the result above implies that it is possible to reorient a subset X of the arcs of D so that we obtain a new digraph $D' = (V, A')$ which is eulerian.

Question a:

Explain how to formulate the problem of finding a such set X of arcs in D as a feasible flow problem. Hint: use the flow variable to indicate whether an arc is reversed or not and express the resulting in-degree of a vertex v in terms of the in-degree in D and the values of x on arcs incident with v.

Question b:

Explain how we can use the flow model above to develop a polynomial algorithm for finding the minimum number of arcs we need to reverse in D in order to obtain an eulerian digraph D'.

Question c:

Suppose now that we would like to minimize the maximum number of arcs that are reversed at any vertex, that is, we wish to determine the minimum k such that D can be made eulerian by reversing at most k arcs at any vertex $v \in V(D)$. Show how to formulate this problem as a convex cost flow problem. Hint: what is the absolute minimum number of arcs we must reverse at a given vertex and how do we avoid reversing more extra arcs than necessary at v (and the other vertices of D)?

[1]NB: This does not mean that D must be regular, as some vertex x may have in- and out-degree p and another vertex y in- and out-degree q with $q \neq p$.

PROBLEM 2 (10 point)

This problem deals with arc-connectivity. Let $D = (V, A)$ be a digraph. First we introduce some notation:

- Denote by $\lambda(D) = \min\{d^+(X)|\text{X is a proper subset of } V\}$. Thus $\lambda(D)$ is the arc-connectivity of D.
- Denote by $r^k(D)$ the minimum number of arcs we need to reverse (replacing $i \to j$ by $j \to i$) so that the resulting digraph D_1 has $\lambda(D_1) \geq k$. If there is no such set of arcs we let $r^k(D) = \infty$.
- Denote by $a^k(D)$ the minimum number of new arcs (parallel arcs allowed) we need to add to D so that the resulting digraph D_2 has $\lambda(D_2) \geq k$.

Question a:

Describe briefly how to find the number $a^k(D)$ in polynomial time based on Frank's algorithm. You do not have to argue about the correctness, but you should explain briefly how to use a maximum flow algorithm as a subroutine in some of the steps of Frank's algorithm. Illustrate the algorithm on the digraph in Figure 1. In doing so you should observe the following:

- Give a subpartition that shows that the number of arcs you add is optimal.
- You should follow the following rule when running the splitting phase: if you add an arc from a vertex i to a vertex j, then the label of j should be as small as possible among the possible arcs you may add from i.

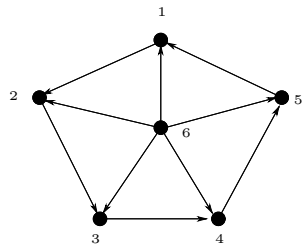

Figure 1: A digraph which is not 2-arc-strong.

Question b:

When is the number $r^k(D) < \infty$? Hint: this can be expressed as a property of $UMG(D)$. Here the underlying multigraph $UMG(D)$ of a digraph D is obtained by forgetting the orientation of all arcs and keeping parallel edges that arise this way (e.g. from a directed 2-cycle).

Question c:

Explain how to calculate $r^k(D)$ in polynomial time using submodular flows. You should explain how you define the flow and how to interpret it.

PROBLEM 3 (10 point)

This problem is about closures in digraphs.

Question a:

Give a brief (but sufficient) description of the flow based algorithm for finding a maximum weight closure in a digraph with weights on the vertices. You should also explain how it can be modified to find the maximum weight closure which is neither the empty set nor the whole set.

Question b:

Explain how to use the flow based algorithm above to find, for a given input digraph $D = (V, A)$ which is not strongly connected, a maximum cardinality set $X \subset V$ with $X \neq \emptyset, V$ and $d^+(X) = 0$. You must also give the running time of your algorithm.

Question c:

Prove that if $X, Y \subset V$ are vertex sets in a non strong digraph $D = (V, A)$ such that $X \cup Y \neq V$ and $d^+(X) = d^+(Y) = 0$, then we also have $d^+(X \cup Y) = 0$.

Question d:

Let $D = (V, A)$ be a digraph which is not strong and define for each vertex $v \in V$ the set X_v^+ as the set of vertices that can be reached from v by directed paths. Similarly, let X_v^- be the set of vertices that can reach v by directed paths.

(a) Show that if X is a closure and $v \in X$, then $X_v^+ \subseteq X$.

(b) Show that if X is a closure and $v \notin X$ then $X \cap X_v^- = \emptyset$.

Question e:

Use the observations above to design a fast algorithm for solving the problem in Question b and compare the running time of your algorithm to the flow based algorithm that you gave in your answer to Question b.

PROBLEM 4 (10 point)

Question a:

Give a brief but sufficient description of the algorithm for finding a minimum cost out-branching from a given vertex s in a digraph $D = (V, A)$ with a non-negative weight function $c : A \to \mathcal{R}$ on its arcs.

Question b:

Illustrate the algorithm on the digraph in Figure 2 by finding a min cost out-branching from the vertex s. Remember to specify how the cost function is modified.

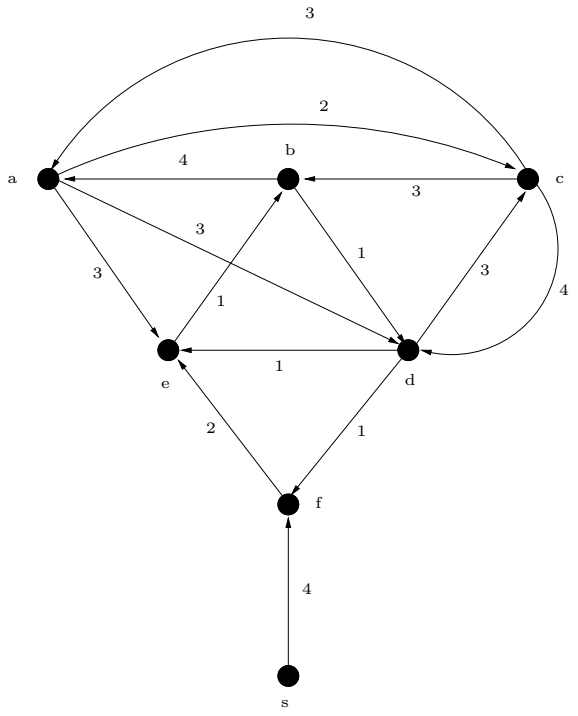

Figure 2: An arc-weighted digraph D.

Question c:

Suppose now that we have specified a subset $A' \subset A$ of the arcs. First explain how to decide whether D has an out-branching B_s^+ from s which contains all the arcs of A' and then explain how to modify the min cost branching algorithm so that it can find a minimum cost out-branching from s which contains all arcs in A' (in the case when there is such a branching).

Question d:

Suppose now that $D = (V, A)$ is strongly connected with a weight function c on its arcs. Explain how to find a minimum cost out-branching when the root is not fixed in the same running

time as the min cost branching algorithm (using just one application of this algorithm).

Question e:

Can you modify your algorithm above so that it can also find a minimum cost out-branching without a root specified, such that the branching contains a prescribed set of arcs A' or report that no such branching exists?

PROBLEM 5 (10 point)

Let x be a fixed feasible flow in $\mathcal{N} = (V, A, \ell \equiv 0, u, b)$.

Question a:

Formulate the problem of finding another feasible flow y which minimizes the number of arcs $ij \in A$ for which $x_{ij} \cdot y_{ij} > 0$ as a network design problem. Hint: introduce a new variable z_{ij} that you can use to control the number of arcs where $x_{ij} \cdot y_{ij} > 0$.

Question b:

Perform Lagrangian relaxation on the inequality which involves the variables z_{ij} and explain how to solve the resulting optimization problem.

Question c:

Explain how to solve the problem of finding a feasible flow y such that $\max\{y_{ij} | x_{ij} > 0\}$ is minimized. Hint: use convex cost flows.