TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

TU Kaiserslautern
AG Algorithmen und Komplexität

Issue Date: 17.05.2013
Version: 2013-05-24 15:26

Exercise Sheet 5 for Combinatorial Algorithms, SS 13

Hand In: *Until Monday,* 03.06.2013, *12:00,*
box in the group's hallway or email to `wild@cs.uni...`.

Problem 6

2 + 2 points

Let $G = (V, E)$ be a simple graph with integral edge capacities $c : E \to \mathbb{N}$. (We assume, no lower capacities are given, resp. they are all zero $l(e) = 0$.)

Proof or **disprove**:

a) If all capacities are *even* numbers, i. e. $N = 2\mathbb{N} = \{2n \mid n \in \mathbb{N}_0\}$, then there is a *maximal* (s, t)-flow f^* with even flow values only, i. e. $\forall e \in E : f^*(e) \in N$.

b) If all capacities are *odd* numbers, i. e. $N = 2\mathbb{N} + 1 = \{2n + 1 \mid n \in \mathbb{N}_0\}$, then there is a *maximal* (s, t)-flow f^* with odd flow values only, i. e. $\forall e \in E : f^*(e) \in N$.

Problem 7

3 points

In applications, we often are not interested in a *maximal* flow, but rather would like to answer the following decision problem: Does a given network with its capacities allow to transmit a certain amount of flow from certain sources to certain sinks? For example, consider a waste water system where at some nodes a certain amount of water per time is added and this water has to be moved to a different node.

Formally, we model this as *feasible flow problem*: Given a simple graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}_{\geq 0}$ and *excess* $b : V \to \mathbb{R}$ at the nodes, does there exist a *feasible* flow $f : E \to \mathbb{R}$ with

$$\forall v \in V \qquad b(v) + \sum_{\substack{e \in E \\ e=(u,v)}} f(e) = \sum_{\substack{e \in E \\ e=(v,u)}} f(e) \quad \text{and} \tag{1}$$

$$\forall e \in E \qquad 0 \leq f(e) \leq c(e) \quad ? \tag{2}$$

We call a node v with positive excess $b(v) > 0$ a *source* and one with negative excess $b(v) < 0$ — i. e. a node with *demand* — a *sink*.

Show that the *feasible flow problem* can be reduced to the *max flow problem*, i. e. describe an algorithm that solves the *feasible flow problem* by calling a max flow algorithm as subroutine.

Problem 8

4 points

We consider the following *scheduling problem*: Given m identical machines $M_1, \ldots, M_m$, which can operate in parallel, and n *jobs* to be finished in time.

Each job j has a *processing time* $p_j \in \mathbb{N}$, which is the number of time slots a machine needs to finish j. Moreover, j cannot be started before it becomes known after its *release date* $r_j \in \mathbb{N}$ and it must be completed upon its *deadline* $d_j \in \mathbb{N}$. We always assume $d_j \geq r_j + p_j$. Time is given in discrete intervals (cf. the unix epoche).

Each machine can process at most one job at a time and a single job cannot be worked on in parallel. However, we allow costless preemption, i. e. we may interrupt execution of jobs at any time and continue processing later, possibly on another machine, without any additional cost.

The problem consists in deciding whether there is a schedule of jobs onto machines, such that all jobs are finished before their deadlines — and in computing such a schedule if it exists.

Design an algorithm for solving this problem in time polynomial in the length of the input. Partial credit is given to less efficient solutions.

Hint: Model the problem as max flow instance.