University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, M. Fuchs, G. Schmid

Algorithm Theory
Exercise Sheet 8

Due: Wednesday, 21st of December 2022, 11:59 pm

Exercise 1: How Many Fully Saturated Edges? *(14 Points)*

Let $G = (V, E)$ be an $s - t$ flow network with integer capacity's $c_e > 0$ on each edge $e \in E$. We say that edge $e \in E$ is *fully saturated* if the flow uses its full capacity in every maximum $s - t$ flow in G.

(a) Assuming we are given a maximum flow f^* of G, describe an algorithm with time complexity $\mathcal{O}(|E|)$ that computes a new maximum flow of G when the capacity of an arbitrary edge $\{u, v\} \in E$ decreases by one unit. Note that flow networks are connected and thus $|V| = \mathcal{O}(|E|)$. *(3 Points)*

(b) Prove that any minimum cut of G exclusively contains fully saturated edges. *(2 Points)*

(c) Prove that an edge $e \in E$ is fully saturated *if and only if* decreasing the capacity of e by 1 decreases the maximum value of an $s - t$ flow in G. *(5 Points)*
Hint: Notice that in general even if the maximum flow value is an integer, the flow values per edges don't necessarily need to be integers.

(d) Devise an algorithm that computes the number of fully saturated edges in the flow network G and analyze its running time in dependency of $|f^*|$. *(4 Points)*

Exercise 2: Seating Arrangement *(6 Points)*

A group of students goes out to eat dinner together. To increase social interaction, they would like to sit at tables such that no two students from the same faculty are at the same table. For that purpose assume there are students from x different faculties while $n_1, n_2, ..., n_x$ describe the related number of students from these x faculties. Also assume that there are y tables available while r_j students can take place on the j-th table.
Let us define the *seating arrangement* as the decision problem returning true if one can distribute the students from same faculties to different tables and false otherwise. Formulate this *seating arrangement* problem as a *maximum flow* problem and write down the condition that should hold whenever the original decision problem returns true. Further, give the runtime it takes to solve the corresponding flow problem in terms of x, y, n_i and r_j for all $1 \leq i \leq x$ and $1 \leq j \leq y$.