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

Algorithm Theory
Exercise Sheet 10

Due: Friday, 10th of January, 2025, 10:00 am

Exercise 1: Sinkless Orientation *(7 Points)*

(a) Let $B = (U \cup V, E)$ be a bipartite graph and assume that for every $A \subseteq U$, we have $|N(A)| \geq |A|$. Show that this implies that there exists a matching of size $|U|$ in B (i.e., a matching that matches every node in U). *(3 Points)*

(b) Let $G = (V, E)$ be a graph with minimum degree at least two (i.e., each node has at least two incident edges). Use the prior statement to show that there is a way to orient the edges of G such that each node of G has at least one out-going edge (this is known as a sinkless orientation).

Hint: Sinkless orientation can be seen as matching nodes and edges. *(3 Points)*

(c) Let us now assume that the graph G has minimum degree 4. Show that there exists an orientation in which each node has at least one in-coming and at least one out-going edge.
Hint: Use part (b). *(1 Point)*

Exercise 2: Maximality Helps *(5 Points)*

Let $B = (U \cup V, E)$ be a bipartite graph. For finding a maximum matching on bipartite graphs, we would like to speed up the Ford Fulkerson approach. Recall that in the max flow reduction approach, we try to improve our matching by iteratively searching for an augementing path, so what if instead we can find a good bunch of disjoint augementing paths? Indeed in here we consider our bipartite graph B as input (and not its flow network representation) and the following algorithm: We start with an empty matching M, then repeat the following 2 steps until no more augmenting paths with respect to the matching M in B can be found:

1. Search for a ***maximal*** *set of* ***vertex-disjoint shortest*** *augmenting paths* $\{P_1, P_2, \ldots, P_k\}$

2. Update M by flipping every matched edge in $P_1, P_2, ..., P_k$ to unmatched and vice versa.

Finally, return M.

Remark: Maximal above means that no more such paths can be added.

(a) Show that after $O(\sqrt{n})$ repetitions of the 2 steps above, M is a maximum matching of B. *(3 Points)*

(b) Argue that the running time of the algorithm can be $O(m\sqrt{n})$, where as usual m is the number of edges and n is the number of nodes. *(2 Points)*

Exercise 3: 2-Claws *(4 Points)*

We are given a bipartite graph $B = (U \cup V, E)$ on two disjoint node sets U and V; each edge connects a node in U to a node in V. In the following, we define a *2-claw* to be a set of three distinct nodes $\{u, x, y\}$ such that $u \in U$, $x, y \in V$ and there is an edge from u to both nodes x and y.
We consider the following maximization problem on the graph B: Find a largest possible set of vertex-disjoint set of 2-claws. In other words, we want to find a largest possible subset of edges such that every node in U is incident to either 0 or 2 of the edges and each node in V is incident to either 0 or 1 of the edges (i.e., each node is either part of one 2-claw or it is not part of any 2-claw at all). We also assume that in the given bipartite graph B, each node in U has at most 3 neighbors in V.

Give a polynomial-time algorithm which computes a maximum set of vertex-disjoint 2-claws.

Hint: Try to reduce the problem to the maximum matching problem in general graphs.

Exercise 4: Special Offer for the Holidays *(4 Points)*

To increase its sales a small kiosk has a special offer: If a customer buys two articles whose prices add up to a value which ends with $11, 33, 55, 77$ or 99 cents, he will receive a voucher, worth the corresponding cent value.

Devise an algorithm which computes an optimal strategy for buying a given collection of goods (here only the price of a good matters).