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

Algorithm Theory
Exercise Sheet 10

Due: Wednesday, 18th of January, 2023, 11:59 pm

Exercise 1: A Matching Reduction *(12 Points)*

Let $G = (V, E)$ be an undirected graph, a *vertex cover* $C \subseteq V$ is a set of nodes that includes at least one endpoint of every edge of the graph. A minimum vertex cover is a vertex cover of minimum size.

(a) Show that the size of a maximum matching is always at most that of the minimum vertex cover. *(2 Points)*

Let $G = (V, E)$ be a bipartite graph where A, B are the bipartition of the vertex set V and the goal now is to efficiently construct a minimum vertex cover of G by showing that the size of the maximum matching and minimum vertex cover are in fact the same. For that purpose, suppose that we are given a maximum matching M^* of G. A node $v \in V$ is said to be *unmatched* if all its incident edges are unmatched edges in M^* and let U be the set of unmatched nodes in A (possibly empty). Let L be the set of vertices that are either in U or are connected to U by alternating paths (paths that alternate between edges that are in the matching and edges that are not in the matching). Define $C := (A \setminus L) \cup (B \cap L)$.

(b) Prove that C is a vertex cover. *(3 Points)*

(c) Show that $|C| = |M^*|$. *(4 Points)*

(d) Give a polynomial time algorithm that computes a minimum vertex cover in bipartite graphs, argue correctness, and analyze its running time. *(3 Points)*

Exercise 2: Unique ID's Assignment *(8 Points)*

Imagine that we have n processes such that every process is connected to all other processes with communication channels i.e. the network topology is a complete graph. Now if all processes are started with the same input, simultaneously, and all of them run the same deterministic algorithm, how could they ever end up in different states? The answer is that they can't!
However, with some additional assumptions such as assigning a *unique* ID for every process they might be able to. And one way of achieving such assignement is via randomly generating the ID's e.g. by running the following randomized algorithm: each process independently and uniformly at random picks an ID from the set $\{1, 2, ..., n^2\}$ and sticks to it as its ID.

(a) What is the probability that all processes succeed in choosing a unique ID at the end of the algorithm? *(3 Points)*

(b) Modify the algorithm in *two* different ways such that *with high probability*, all processes succeed in choosing a unique ID at the end of the algorithm. *(5 Points)*
Remark: for both approaches, we want an algorithm with a fixed running time (which can be at most $O(\log n)$) that outputs a correct solution with high probablity.