Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn

Theoretical Computer Science - Bridging Course
Winter 2020/21
Exercise Sheet 8

for getting feedback submit electronically by 12:15, Monday, January 11th, 2021

Exercise 1: The Class $\mathcal{P}$

$\mathcal{P}$ is the set of languages which can be decided by an algorithm whose runtime can be bounded by $p(n)$, where p is a polynomial and n the size of the respective input (problem instance). Show that the following languages ($\cong$ problems) are in the class $\mathcal{P}$. Since it is typically easy (i.e. feasible in polynomial time) to decide whether an input is well-formed, your algorithm only needs to consider well-formed inputs. Use the $\mathcal{O}$-notation to bound the run-time of your algorithm.

(a) PALINDROME:= $\{w \in \{0,1\}^* \mid w \text{ is a Palindrome}\}$

(b) LIST:=$\{\langle A, c\rangle \mid A$ is a finite list of numbers which contains two numbers x,y such that $x + y = c\}$.

(c) 3-CLIQUE := $\{\langle G\rangle \mid G$ has a *clique* of size at least 3$\}$

Remark: A *clique* in a graph $G = (V, E)$ is a set $Q \subseteq V$ such that for all $u, v \in Q : \{u, v\} \in E$.

Exercise 2: The Class $\mathcal{NPC}$

Let L_1, L_2 be languages (problems) over alphabets Σ_1, Σ_2. Then $L_1 \leq_p L_2$ (L_1 is polynomially reducible to L_2), iff a function $f : \Sigma_1^* \to \Sigma_2^*$ exists, that can be calculated in polynomial time and

$$\forall s \in \Sigma_1^* : s \in L_1 \iff f(s) \in L_2.$$

Language L is called $\mathcal{NP}$-hard, if *all* languages $L' \in \mathcal{NP}$ are polynomially reducible to L, i.e.

$$L \text{ is } \mathcal{NP}\text{-hard} \iff \forall L' \in \mathcal{NP} : L' \leq_p L.$$

The reduction relation '$\leq_p$' is transitive ($L_1 \leq_p L_2$ and $L_2 \leq_p L_3 \Rightarrow L_1 \leq_p L_3$). Therefore, in order to show that L is $\mathcal{NP}$-hard, it suffices to reduce a known $\mathcal{NP}$-hard problem $\tilde{L}$ to L, i.e. $\tilde{L} \leq_p L$. Finally a language is called $\mathcal{NP}$-complete ($\Leftrightarrow$: $L \in \mathcal{NPC}$), if

1. $L \in \mathcal{NP}$ and
2. L is $\mathcal{NP}$-hard.

(a) Show CLIQUE$:=\{\langle G,k\rangle \mid G$ has a clique of size at least k $\}\in\mathcal{NPC}$.

(b) Show HITTINGSET $:= \{\langle \mathcal{U}, S, k\rangle \mid$ universe $\mathcal{U}$ has subset of size at most k that *hits* all sets in $S \subseteq 2^{\mathcal{U}}\}\in\mathcal{NPC}$.[1]

For both parts, use that VERTEXCOVER $:= \{\langle G,k\rangle \mid$ Graph G has a *vertex cover* of size at most $k\} \in \mathcal{NPC}$.

Remark: A ***hitting set*** *$H \subseteq \mathcal{U}$ for a given universe $\mathcal{U}$ and a set $S = \{S_1, S_2, \ldots, S_m\}$ of subsets $S_i \subseteq \mathcal{U}$, fulfills the property $H \cap S_i \neq \emptyset$ for $1 \leq i \leq m$ (H 'hits' at least one element of every S_i).*
A ***vertex cover*** *is a subset $V' \subseteq V$ of nodes of $G = (V,E)$ such that every edge of G is adjacent to a node in the subset.*

Hint: For the poly. transformation ($\leq_p$) you have to describe an algorithm (with poly. run-time!) that transforms an instance $\langle G,k\rangle$ of VERTEXCOVER *into (a) an instance $\langle G',k'\rangle$ of* CLIQUE *s.t. a vertex cover of size $\leq k$ in G becomes a clique of G' of size $\geq k'$ vice versa(!); and (b) an instance $\langle \mathcal{U}, S, k\rangle$ of* HITTINGSET*, s.t. a vertex cover of size $\leq k$ in G becomes a hitting set of $\mathcal{U}$ of size $\leq k$ for S and vice versa(!).*

[1]The power set $2^{\mathcal{U}}$ of some ground set $\mathcal{U}$ is the set of *all subsets* of $\mathcal{U}$. So $S \subseteq 2^{\mathcal{U}}$ is a collection of subsets of $\mathcal{U}$.