Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
P. Schneider, Y. Maus January 10, 2017

Theoretical Computer Science - Bridging Course
Winter Term 2016
Exercise Sheet 9

Hand in (electronically or hard copy) before your weekly meeting but not later than 23:59, Wednesday, January 11, 2016

Exercise 1: The class $\mathcal{NP}$ (2+3+2+0 points)

$\mathcal{NP}$ is the set of languages which can be decided by a non-deterministic Turing machine whose run-time (minimum number of steps required to reach the accepting state) can be bounded by $p(n)$, where p is a polynomial and n the size of the respective input (problem instance).
A non-deterministic Turing machine N is typically used for a non-deterministic procedure called 'guess and check'. First N 'guesses' a solution for given problem instance s, which leads to the correct answer of the decision problem ($s \in L$ or $s \notin L$)[1]. Then the solution guessed by N is verified by a deterministic Turing machine D which accepts if and only if the solution is correct (D sifts out wrong guesses).
Show that the following problems are in $\mathcal{NP}$ by describing the solution that the non-deterministic machine is expected to guess, and giving a deterministic algorithm that verifies it in polynomial time. Use the $\mathcal{O}$ notation to bound the run time. Since it is easy (i.e. possible in polynomial time) to decide whether inputs are well-formed instances, your algorithm only needs to consider well-formed inputs.

(a) $\textsc{Clique} = \{\langle G, k\rangle \mid G$ is a Graph with a complete subgraph with k nodes$\}$

(b) $\text{ISO} = \{\langle G, H\rangle \mid G$ and H are isomorphic graphs$\}$
Remark: Two graphs G, H are isomorphic, if a ***bijective*** *mapping $f : V(G) \to V(H)$ exists such that $u, v \in V(G)$ are adjacent in G, if and only if $f(u), f(v) \in V(H)$ are adjacent in H.*

(c) $\text{3-SAT} = \{\langle\phi\rangle \mid$ bool. formula ϕ in 3-CNF has assignment of variables s.t. ϕ evaluates to TRUE.$\}$
Remark: ϕ is in 3-CNF if it is of the form $C_1 \wedge \ldots \wedge C_m$, where $C_i = L_{i,1} \vee L_{i,2} \vee L_{i,3}$ are clauses of ***at most*** *three* ***literals*** *$L_{i,j} = x_k$ or $L_{i,j} = \overline{x}_k$ of negated or non-negated* ***variables*** *$x_1, \ldots, x_n$ of ϕ.*

(d) Show that 2-SAT $\in \mathcal{P}$ (voluntary).
Hint: Clauses with two literals can be transformed into the form $A \to B$.

Exercise 2: The class $\mathcal{NPC}$ (3+4 points)

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.

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

[1]A Turing machine that 'guesses' solutions can be constructed as as follows: N writes a random input sequence and halts. Since a non-deterministic Turing machine 'explores all possibilities' it will give the correct solution in polynomial time, if it exists.

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 HalfClique $:= \{\langle G \rangle \mid$ Graph G with n nodes has Clique of size at least $\lceil n/2 \rceil\} \in \mathcal{NPC}$. Use that Clique $\in \mathcal{NPC}$.
Hint: Describe an algorithm (with poly. run-time!) that transforms G and k into a graph G' by adding nodes and connecting them with edges in a suitable manner, s.t. a Clique of size k in G becomes a Clique of size $\lceil n/2 \rceil$ in G' and vice versa(!). Be mindful of the cases k odd or even.

(b) Show DominatingSet $:= \{\langle G, k \rangle \mid$ Graph G has a dominating set of size at most $k\} \in \mathcal{NPC}$. Use that VertexCover $:= \{\langle G, k \rangle \mid$ Graph G has a vertex cover of size at most $k\} \in \mathcal{NPC}$.
*Remark: A **dominating set** is a subset of nodes of G such that every node not in the subset is adjacent to some node in the subset. A **vertex cover** is a subset of nodes of G such that every edge of G is adjacent to a node in the subset.*
Hint: Transform a Graph G into a Graph G' such that a vertex cover of G will result in a dominating set G' and vice versa(!). Note that a dominating set is not necessarily a vertex cover ($G = (\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}\})$) has the dominating set $\{v_1, v_4\}$ which is not a vertex cover). Also a vertex cover is not necessarily a dominating set (consider isolated notes).

Exercise 3: Complexity Classes: Big Picture (1+2+3+0+0+0 points)

(a) Why is $\mathcal{P} \subseteq \mathcal{NP}$?

(b) Show that $\mathcal{P} \cap \mathcal{NPC} = \emptyset$ if $\mathcal{P} \neq \mathcal{NP}$.
Hint: Assume that there exists a $L \in \mathcal{P} \cap \mathcal{NPC}$ and derive a contradiction to $\mathcal{P} \neq \mathcal{NP}$.

(c) Give a Venn Diagram showing the sets $\mathcal{P}, \mathcal{NP}, \mathcal{NPC}$ for both cases $\mathcal{P} \neq \mathcal{NP}$ and $\mathcal{P} = \mathcal{NP}$.
Remark: Use the results of (a) and (b) even if you did not succeed in proving those.

(d) Show that the Halting Problem H is $\mathcal{NP}$-hard. You can use that

$$\text{SAT} = \{\langle \phi \rangle \mid \text{bool. formula } \phi \text{ has assignment of variables s.t. } \phi \text{ evaluates to TRUE.}\}$$

is $\mathcal{NP}$-hard. (voluntary)
Hint: For any boolean formula ϕ give an algorithm $\mathcal{A}$ that stops if and only if ϕ is satisfiable.

(e) Argue why $H \notin \mathcal{NP}$. (voluntary) *Hint: You can use results from previous exercise sheets.*

(f) Add the class of $\mathcal{NP}$-hard problems to the Venn Diagrams from exercise (c). (voluntary)

We wish you happy holidays and a good start into the new year 2017.