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

Theoretical Computer Science - Bridging Course
Winter Term 2019/2020
Exercise Sheet 8

for getting feedback submit electronically by 12:15, Monday, December 16 2019

Exercise 1: The Class $\mathcal{P}$ *(2+3+2+3 Points)*

$\mathcal{P}$ 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$\}$

(d) 17-DOMINATINGSET := $\{\langle G\rangle \mid G$ has a *dominating set* of size at most 17$\}$

Remark: A *dominating set* for a graph $G = (V, E)$ is a set $D \subseteq V$ such that for every vertex $v \in V$, v is either in D or adjacent to a node in D.
Remark: A *clique* in a graph $G = (V, E)$ is a set $Q \subseteq V$ such that for all $u, v \in Q : \{u, v\} \in E$.

Sample Solution

(a) We have already seen in exercise sheet 5 that the problem can be solved with a Turing machine with $\mathcal{O}(n^2)$ head movements. The same idea/algorithm shows that the problem is in $\mathcal{P}$.

(b) Assume that the length of A is n. We can simply go through all combinations of tuples of A with two for-loops with indexes i and j where i ranges from 1 to n and j ranges from $i+1$ to n. Then we can simply test whether $A[i] + A[j] == c$ and accept if this ever evaluates to true, otherwise we reject. This has a runtime of $\mathcal{O}(n^2)$ because the test can be done in $\mathcal{O}(1)$ and there are $\mathcal{O}(n^2)$ tuples.

(c) Let $G = (V, E)$ and $|V| = n$. Then we know $|E| = \mathcal{O}(n^2)$. Upon input G, we can enumerate all possible triples (v_1, v_2, v_3) such that $v_1 \neq v_2 \neq v_3 \neq v_1$. There exist at most $\binom{n}{3} = \mathcal{O}(n^3)$ such triples. For each such triple (v_1, v_2, v_3), we examine whether $(v_1, v_2) \in E$, $(v_1, v_3) \in E$, and $(v_2, v_3) \in E$. Since $|E| = O(n^2)$, this examination can be done in $O(n^2)$ time. If during the examination process, we find one triple that satisfies the requirement, we found a clique of size 3 accept G. Otherwise, when we finish examining all possible triple, we reject G since it does not contain a clique of size 3. The runtime of the above procedure is $\mathcal{O}(n^5)$, thus 3-CLIQUE $\in \mathcal{P}$.

(d) A dominating set of size 17 exists if and only if all nodes are covered by a subset of the nodes $D \subseteq V$ with $|D| = 17$. There are less than n^{17} sets with $|D| = 17$. We iterate through all of the sets (e.g., by using 17 nested for loops iterating over the identifiers of the node set if we number the nodes from 1 to n.). Then for each set we test whether it dominates the whole graph by testing whether every node not in the set has a neighbor in the set. If this is the case for some set we accept. We reject if it's not the case for all sets.

This test can be done in time $\mathcal{O}(n)$ for each other node by going through the respective part of the adjacency matrix. There are at most $|V \setminus D| \leq n$ nodes outside of D for a D with $|D| = 17$. So to check whether a specific D is a dominating set we only need $\mathcal{O}(n^2)$ time. In total the time complexity is $\mathcal{O}(n^{17} \cdot n^2) = \mathcal{O}(n^{19})$. Therefore 17-DominatingSet $\in \mathcal{P}$.

Exercise 2: The Class $\mathcal{NP}$ *(3 Points)*

Consider the following problem, called SUBSET-SUM. Given a collection S of integers $x_1, \ldots, x_k$ and a target t, it is required to determine whether S contains a sub-collection that adds up to t. Then, the problem can be given by

$$\text{SUBSET-SUM} = \left\{ \langle S, t \rangle | S = \{x_1, \ldots, x_k\}, \text{and for some } \{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\} \text{ we have } \sum_i y_i = t \right\}$$

Show that SUBSET-SUM is in $\mathcal{NP}$.

Sample Solution

Consider the following verifier for SUBSET-SUM on input $\langle \langle S, t \rangle, C \rangle$. The verifier first tests whether C is a collection of numbers that add up to t. Then, it tests whether all the numbers in C are also in S. If both pass, it accepts, and otherwise it rejects.
Regarding the first test, it scans C once, and regarding the second test, it scans S at most $|C|$ times. Therefore, the total running time is polynomial in the input size.

Exercise 3: The Class $\mathcal{NPC}$ *(7 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.

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

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.

Show HittingSet $:= \{\langle \mathcal{U}, S, k \rangle \,|\, \text{universe } \mathcal{U} \text{ has subset of size} \leq k \text{ that } \textit{hits} \text{ all sets in } S \subseteq 2^{\mathcal{U}}\} \in \mathcal{NPC}$.[1]

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).*

[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 $\mathcal{U}$.

*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 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(!).*

Sample Solution

We first show that hitting set belongs in $\mathcal{NP}$, by engineering a deterministic polynomial time verifier for it. Then we will prove that it is an $\mathcal{NP}$-hard problem, by reducing a known $\mathcal{NP}$-hard problem, vertex cover (as mentioned in the hint), to hitting set in polynomial time.

Guess and Check: Given a finite set $\mathcal{U}$, a collection S of subsets of $\mathcal{U}$, a positive integer k and a finite set H as a certificate, the following deterministic polynomial time verifier for hitting set verifies in polynomial time that $(\mathcal{U}, S)$ has a hitting set of size at most k. Let λ be the sum of the sizes of all the subsets S_i in S and δ the size of $\mathcal{U}$. Note that we can check if A is a subset of B with the following brute-force algorithm: $\forall a \in A$ check if $\exists b \in B : a = b$ which needs $\mathcal{O}(|A| \cdot |B|)$ comparisons. We can check if H is a subset of $\mathcal{U}$ that has at most k elements with $\mathcal{O}(k \cdot \delta)$ comparisons and if it contains at least one element from each subset S_i in the collection S, with $\mathcal{O}(\lambda \cdot k)$ comparisons. We accept iff both checks are true. These two checks are obviously equivalent to the problem's definition, so hitting set has a polynomial time verifier. Therefore it belongs in $\mathcal{NP}$.

Polynomial Reduction of VertexCover **to** HittingSet**:** We will create a polynomial time reduction from vertex cover to hitting set, proving that since vertex cover is $\mathcal{NP}$-hard, hitting set must also be $\mathcal{NP}$-hard.
The reduction takes as input an undirected graph $G = (V, E)$, where V is a set of nodes and E a set of edges defined over those nodes, as well as a positive integer k and outputs the set V, the collection $E = \{e_1, e_2, \ldots, e_n\}$ of subsets of V and the positive integer k. We claim the following equivalence holds:

"G has a vertex cover of size at most k" $\Leftrightarrow$ "(V, E) has a hitting set of size at most k"

Here is the proof:

"G has a vertex cover of size at most k" $\Leftrightarrow$
$\exists V' \subseteq V : |V'| \leq k$ and $\forall$ edge $e_i = \{u_i, v_i\} \in E, u_i \in V'$ or $v_i \in V' \Leftrightarrow$
$\exists V' \subseteq V : |V'| \leq k$ and $\forall$ subset e_i in collection E $\exists c \in e_i : c \in V' \Leftrightarrow$
"(V, E) has a hitting set of size at most k"

This reduction takes time linear to the size of the input (all it does is copy the input to the output), therefore polynomial. Also, as we showed, it is correct. Therefore, hitting set is at least as hard as vertex cover and since vertex cover is $\mathcal{NP}$-hard, so is hitting set.
One might notice that this reduction was rather straightforward. This makes sense, since vertex cover is a special version of hitting set, where each subset S_i in the collection S has exactly two elements of $\mathcal{U}$. Obviously, no problem can be harder than its generalization and since vertex cover is $\mathcal{NP}$-hard, hitting set (as a generalization of vertex cover) must also be $\mathcal{NP}$-hard.