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

for getting feedback submit electronically by 12:15 am, 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 ($\approx$ 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 $O$-notation to bound the run-time of your algorithm.

(a) Palindrome := \{ $w \in \{0, 1\}^* \mid w$ 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 $u, v \in Q : \{u, v\} \in E$.

Sample Solution

(a) We have already seen in exercise sheet 5 that the problem can be solved with a Turing machine with $O(n^2)$ 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 $O(c \cdot n^3)$ because the test can be done in $O(c \cdot n)$ and there are $O(n^3)$ tuples.

(c) Let $G = (V, E)$ and $|V| = n$. Then we know $|E| = O(n^2)$. 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 $\binom{n}{3} = O(n^3)$ 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 $O(n^3)$, thus 3-Clique $\in \mathcal{P}$.
Exercise 2: The Class $\mathcal{NP}$

Let $L_1, L_2$ be languages (problems) over alphabets $\Sigma_1, \Sigma_2$. 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. $L \leq_p \tilde{L}$. Finally a language is called $\mathcal{NP}$-complete ($\iff L \in \mathcal{NP}$), if

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

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

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

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

Remark: A hitting set $H \subseteq U$ for a given universe $U$ and a set $S = \{S_1, S_2, \ldots, S_m\}$ of subsets $S_i \subseteq U$, 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 $\text{VERTEXCOVER}$ into (a) an instance $\langle G', k' \rangle$ of $\text{CLIQUE}$ 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 $\langle U, S, k \rangle$ of $\text{HITTINGSET}$, s.t. a vertex cover of size $\leq k$ in $G$ becomes a hitting set of $U$ of size $\leq k$ for $S$ and vice versa(!).

Sample Solution

We first show that clique/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 clique/hitting set in polynomial time.

(a) Guess and Check: Given a graph $G = (V,E)$, positive integer $k$, and a finite set $C$ as a certificate, we build a deterministic polynomial time verifier (algorithm) for $\text{CLIQUE}$ that verifies in polynomial time that $G$ has a clique of size at least $k$. The Idea for the choice of $C$ is to be a clique of size $k$. Let $|V| = n$. First the verifier checks if $C$ has $k$ nodes from $G$ with $O(k \cdot n)$ comparisons, then it checks whether $(u,v) \in E$ for every $u,v \in C$ in $O(k^2 \cdot n^2)$ comparisons. If both these checks pass, accept; else reject. Thus the verification is done in $O(k^2 \cdot n^2)$ time, which is polynomial in the size of $n$ (since $k \leq n$). Hence, $\text{CLIQUE}$ is in $\mathcal{NP}$.

Polynomial Reduction of $\text{VERTEXCOVER}$ to $\text{CLIQUE}$: We will create a polynomial time reduction from vertex cover to clique, proving that since vertex cover is $\mathcal{NP}$-hard, clique must also be $\mathcal{NP}$-hard. The reduction takes as an 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 complement

1The power set $2^U$ of some ground set $U$ is the set of all subsets of $U$. So $S \subseteq 2^U$ is a collection of subsets of $U$. 
The graph $\overline{G} = (V, \overline{E})$ where $V$ is the same set of $V$ of $G$, the set of all edges that don’t exist in $G$ defined by $\overline{E} = \{(u, v) : u, v \in V, u \neq v, (u, v) \notin E\}$ as well as the positive integer $k' := n - k$.

We observe the following:

"$G$ has a vertex cover of size at most $k$" $\iff$ "$\overline{G}$ has a clique of size at least $n - k$"

The proof is the following. Consider the forward direction: suppose that $G$ has a vertex cover $S \subseteq V$ of size at most $k$ (a yes instance of $\text{VERTEXCOVER}$). Then for all $u, v \in V$, if $(u, v) \in E$, then $u \in S$ or $v \in S$, or both, by definition of the vertex cover that it needs to cover all edges. Now consider $S' := V \setminus S$. Clearly $|S'|$ is at least $n - k$. Also, notice that $S'$ is an independent set of $G$ i.e. there are no edges connecting any two nodes in $S'$, since there cannot exist an edge $\{u, v\}$ in $G$ where $u \in S'$ and $v \in S'$, else we reach a contradiction to that $S$ is a vertex cover. Moreover, if we consider the graph $\overline{G} = (V, \overline{E})$, we deduce that for all $u, v \in S'$, $\{u, v\} \in \overline{E}$. Therefore, $S'$ is a clique in $\overline{G}$ of size at least $n - k$ (a yes instance of $\text{ CLIQUE}$).

Conversely the backward direction: suppose $\overline{G} = (V, \overline{E})$ has a clique $S \subseteq V$ of size $n - k$. All nodes in the clique $S$ are connected to each other by an edge in $\overline{E}$. Hence, $S$ makes up an independent set in $G = (V, E)$. Thus, $V \setminus S$ is a vertex cover in $G$, else there exists an edge $\{u, v\} \in E$ which is not covered by $V \setminus S$ i.e. both $u$ and $v$ are not in $V \setminus S$, thus $u, v \in S$. This is a contradiction since $u, v \in S$, and $\{u, v\} \in E$ and $S$ is an independent set in $G$.

Alternatively: You can suppose that $G$ has no vertex cover $S \subseteq V$ of size at most $k$ (a no instance of $\text{VERTEXCOVER}$) and prove that, $V \setminus S$ is not clique in $\overline{G}$ of size at least $n - k$ (a no instance of $\text{ CLIQUE}$), else you get a contradiction using the same arguments as above.

Hence, we have proven that vertex cover can be reduced to clique. This reduction can be done in polynomial time by generating the complement graph as follows: copy the vertex set $V$ of the input $G$ as is and go through each pair of nodes in $G$ : generate an edge for $\overline{G}$ only if there is no edge between the pair in $G$; as well as outputting the positive integer $n - k$. All these operations can be done in polynomial time. Therefore, $\text{VERTEXCOVER}$ can be reduced in polynomial time to $\text{ CLIQUE}$. Hence, $\text{ CLIQUE}$ is $\mathcal{NP}$-hard.

(b) Guess and Check: Given a finite set $U$, a collection $S$ of subsets of $U$, 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 $(U, S)$ has a hitting set of size at most $k$. Let $\lambda$ be the sum of the sizes of all the subsets $S_i$ in $S$ and $\delta$ the size of $U$. 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 $O(|A| \cdot |B|)$ comparisons. We can check if $H$ is a subset of $U$ that at least $k$ elements with $O(k \cdot \delta)$ comparisons and if it contains at least one element from each subset $S_i$ in the collection $S$, with $O(\lambda \cdot k)$ 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 $\text{VERTEXCOVER}$ to $\text{HITTINGSET}$: 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 $k$" $\iff$ "$(V, E)$ has a hitting set of size at most $k$"

Here is the proof:

"$G$ has a vertex cover of size at most $k$" $\iff$

\[ \exists V' \subseteq V : |V'| \leq k \text{ and } \forall \text{ edge } e_i = \{u_i, v_i\} \in E, u_i \in V' \text{ or } v_i \in V' \iff \exists V' \subseteq V : |V'| \leq k \text{ and } \forall \text{ subset } e_i \text{ in collection } E \exists c \in e_i : c \in V' \iff \]

"$(V, E)$ 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.