Tutorial 13

Exercise 1: Properties of NP-complete problems

1. Prove the theorem on Slide 21 of Lecture 13: Let L be NP-complete. If $L \in P$, then P $=$ NP.
2. In the picture on Slide 21 of Lecture 13 on the right-hand side, we have not argued why the set of NP-complete problems is (almost) equal to P/NP. Prove that statement, formalized as follows:

   If P $=$ NP, every nontrivial problem (i.e., $L \subseteq \Sigma^*$ with $L \neq \emptyset$ and $L \neq \Sigma^*$) in NP is NP-complete.

Solution:

1. Recall the theorem (from Lecture 12, Slide 22) stating that $A \leq_p B$ and $B \in$ P implies $A \in$ P.

   By assumption, we have that L is NP-complete, so we have by definition $A \leq_p L$ for every $A \in$ NP. Also, we have that L is in P.

   Hence, by plugging in L for B in the theorem, we obtain $A \in$ P for every $A \in$ NP, i.e., NP $\subseteq$ P.

   The other inclusion, i.e., P $\subseteq$ NP, holds by definition (see Lecture 12, Slide 14). Hence, we have shown P $=$ NP as required.

2. Let $L \in$ NP be an arbitrary nontrivial problem in NP. We need to prove that L is NP-complete, i.e., that L is NP-hard and in NP. The latter already holds by the choice of L, so we only need to prove the former. We do that by proving $L' \leq_p L$ for all $L' \in$ NP.

   Fix any $L' \in$ NP. By our assumption P $=$ NP, there is a polynomial-time halting DTM for L'; let us call it M'. Also, as L is nontrivial, there are words $w_L^+ \in L$ and $w_L^- \notin L$.

   Now, consider the following function f:

   $$f(w) = \begin{cases} w_L^+ & \text{if } w \in L', \\ w_L^- & \text{if } w \notin L'. \end{cases}$$

   The following Turing machine computes f in polynomial time:

   On input w:

   (a) Simulate M' on w.
   (b) If M' accepts, empty the tape and write w_L^+.
   (c) If M' rejects, empty the tape and write w_L^-.
   (d) Terminate.

   Each step takes a polynomial number of steps, so the overall running time is polynomial. Further, the Turing machine obviously computes f.

   Finally, we show that $w \in L' \Leftrightarrow f(w) \in L$:

   - If $w \in L' = L(M')$, then M' accepts w, which implies $f(w) = w_L^+ \in L$.
   - If $w \notin L' = L(M')$, then M' rejects w, which implies $f(w) = w_L^- \notin L$.

   Altogether, we have $L' \leq_p L$, as required.

   Hence, L is NP-complete.

Exercise 2: 3SAT is NP-hard

Give a polynomial-time reduction from CNF-SAT to 3SAT.

Solution:
We present a reduction from CNF formulas φ into 3CNF formulas $f(\varphi)$ that preserves satisfiability, i.e., φ is satisfiable if and only if $f(\varphi)$ is satisfiable.

Let $\varphi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} \ell_{i,j}$ be an arbitrary formula in CNF. We show how to turn each clause $\bigvee_{j=1}^{m_i} \ell_{i,j}$ into a 3CNF formula ψ_i. Then, the desired 3CNF formula is $f(\varphi) = \bigwedge_{i=1}^{n} \psi_i$, which is indeed in 3CNF.

Consider some clause $\bigvee_{j=1}^{m_i} \ell_{i,j}$. We distinguish several cases for m_i:

- If $m_i = 3$, i.e., the disjunction has already the right length, then we set $\psi_i = \bigvee_{j=1}^{m_i} \ell_{i,j}$.
- If $m_i = 1$, then we define $\psi_i = \ell_{i,1} \vee \ell_{i,1} \vee \ell_{i,1}$ (i.e., we repeat the literal three times). This is logically redundant but does not influence satisfiability.
- If $m_i = 2$, then we define $\psi_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,2}$ (i.e., we repeat the second literal). Again, this is logically redundant but does not influence satisfiability.
- If $m_i > 3$, then we define $\psi_i = (\ell_{i,1} \vee \ell_{i,2} \vee p_1) \wedge (\neg p_1 \vee \ell_{i,3} \vee p_2) \cdots (\neg p_{m_i-3} \vee \ell_{i,m_i-1} \vee \ell_{i,m_i})$ where the p_j are new variables not occurring in φ. This formula is satifiable if and only if $\bigvee_{j=1}^{m_i} \ell_{i,j}$ is satisfiable (make sure you understand why this is true).

Each of these cases can be implemented in polynomial time.

Exercise 3: Spot the error

Every now and then, someone claims to have proven P $=$ NP or P $\neq$ NP. An incomplete list of "proofs" can be found here: `https://www.win.tue.nl/~wscor/woeginger/P-versus-NP.htm`

Last week, a very famous professor published the following proof that P $\neq$ NP:

Proof: Consider the following halting DTM for CLIQUE:

On input $(\ulcorner G \urcorner, k)$:

1. Generate all possible subsets of vertices from G.
2. If one of these subsets is a k-clique, then accept.
3. Otherwise, reject."

Because there are 2^n different subsets of nodes to examine, the algorithm clearly does not run in polynomial time. Therefore, we have proven that CLIQUE has exponential time complexity and this means CLIQUE $\notin$ P. Because we know that CLIQUE $\in$ NP, we conclude that P $\neq$ NP.

Describe the error in the above proof.

Solution:
It is true that the suggested *algorithm* for CLIQUE does not run in polynomial time, but from this fact we cannot conclude that the *language* CLIQUE does not belong to the class P. There can still be other (faster) algorithms for CLIQUE that run in polynomial time; we simply cannot exclude this possibility by presenting one particular algorithm with an exponential running time. To conclude that CLIQUE $\notin$ P, we would have to show *no* halting DTM for CLIQUE has a polynomial time complexity.

Exercise 4: Challenge

Consider the decision problem

Given an undirected graph G, does G have a clique of size 4?

1. Express this problem as a language 4CLIQUE.
2. Prove 4CLIQUE $\in$ NP.
3. Prove 4CLIQUE $\in$ P.
4. Why is CLIQUE not known to be in P if 4CLIQUE is? Justify your answer.

Solution:

1. The language is

$$\text{4CLIQUE} = \{\ulcorner G \urcorner \mid G \text{ is an undirected graph with a clique of size } 4\}$$

2. We show that 4CLIQUE $\in$ NP by constructing a polynomial-time halting NTM:

   On input $\ulcorner G \urcorner$:
   (a) Guess C, a set of 4 nodes from G.
   (b) Check that each pair of nodes from C is connected by an edge.
   (c) If this is the case, then accept, else reject.

   The running time of this halting NTM is obviously polynomial.

3. In a graph with n nodes, there are

$$\binom{n}{4} = \frac{n!}{(n-4)!4!} \leq n(n-1)(n-2)(n-3) = \mathcal{O}(n^4)$$

   sets of nodes with exactly 4 elements. We can therefore create the following polynomial-time algorithm for 4CLIQUE:

   On input $\ulcorner G \urcorner$:
   1. For every set C of 4 nodes from G:
      - Check that every pair of nodes from C is connected by an edge.
      - If this is the case, then accept.
   2. If no set of 4 nodes is a clique, then reject.

   Given a graph with n vertices, the algorithm will perform at most $O(n^4)$ traversals of its main loop. Every traversal will require at most n^2 steps, since each edge must be examined at most once. The algorithm therefore has polynomial time complexity.

4. We get no information about CLIQUE from our knowledge that 4CLIQUE $\in$ P, since CLIQUE is a different problem, i.e., CLIQUE has two parameters, while 4CLIQUE only has one. Furthermore, for 4CLIQUE, there is a polynomial number of candidates to check (for a fixed polynomial), while for CLIQUE there is no such fixed polynomial that bounds the number of candidates.