Tutorial 12

Exercise 1: DNF-SAT is in P

A Boolean formula φ is in disjunctive normal form (DNF) if it is of the form $\varphi = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} \ell_{i,j}$ such that each $\ell_{i,j}$ is a literal, i.e., either a variable x or a negated variable $\neg x$.
For example, $(x \wedge \neg y \wedge z) \vee (\neg x) \vee (\neg y \wedge \neg z)$ is in DNF.

Show that satisfiability of formulas in DNF is in P, i.e.,

$$\text{DNF-SAT} = \{\varphi \mid \varphi \text{ is in DNF and has a satisfying assignment}\} \in \text{P}.$$

Note that the example formula above is satisfiable and hence in DNF-SAT.

Solution:
Let $\varphi = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} \ell_{i,j}$ be a fomula in DNF. Then, φ is satisfiable if and only if one of the disjuncts $\bigwedge_{j=1}^{m_i} \ell_{i,j}$ is satisfiable.

We say that a variable x is contradictory in $\bigwedge_{j=1}^{m_i} \ell_{i,j}$ if $\ell_{i,j} = x$ for some j and $\ell_{i,j'} = \neg x$ for some j'. A formula of the form $\psi = \bigwedge_{j=1}^{m_i} \ell_{i,j}$ is satisfiable if and only if there is no contradictory variable x in ψ. For example, $x \wedge \neg y \wedge z$ is satisfiable while $x \wedge \neg y \wedge z \wedge \neg x$ is not.

Thus, the following algorithm shows that DNF-SAT is in P. Given an input w:

1. If w does not encode a formula in DNF, reject (otherwise, let $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} \ell_{i,j}$ be the formula encoded by w).
2. $i = 1$.
3. If there is no contradictory variable in $\bigwedge_{j=1}^{m_i} \ell_{i,j}$, then accept.
4. Increment i.
5. If $i \leq n$, go to step 3.
6. Reject.

Exercise 2: 3-coloring is in NP

Show that 3-coloring is in NP.

Note that the input is a graph, which we assume is given as an adjacency matrix.

Solution:
We describe a halting NTM with polynomial time complexity:
On input w:

1. If w is not of the form $\mathbb{B}^{n^2}$ for some n, then reject (if the machine does not reject, then w is interpreted as the adjacency matrix of a graph with n vertices).
2. Guess $c_1 \cdots c_n \in \{R, G, B\}^n$.
3. For each edge (i, j) encoded by w check whether $c_i = c_j$. If yes, reject.
4. Accept.

Each step is obviously polynomial in w.

Exercise 3: Vertex cover

Let $G = (V, E)$ be an undirected graph. A *k-vertex cover* is a subset $C \subseteq V$ of $k = |C|$ vertices such that $v \in C$ or $v' \in C$ for all $(v, v') \in E$.

The decision problem is: given an undirected graph and a value k, does G have a k-vertex cover? Formally:

$$\mathrm{VC} = \{(\ulcorner G \urcorner, k) \mid G \text{ is an undirected graph with a } k\text{-vertex cover}\}$$

1. Show that VC $\in$ NP.
2. Show that SAT $\leq_p$ VC (i.e., describe a polynomial-time reduction from SAT to VC).

   You may assume that the formula is in 3-CNF (i.e., in conjunctive normal form with 3 literals per disjunction). (This is no restriction because one can bring any formula to this form – in a certain sense.)

Note that the input is a graph, which we assume is given as an adjacency matrix, and a number $k \in \mathbb{N}$ given in binary.

Solution:

1. Consider the following Turing machine. On input $\ulcorner G \urcorner \in \mathbb{B}^*$ and $k \in \mathbb{N}$:
   (a) Check whether $|\ulcorner G \urcorner| = n^2$ for some $n \in \mathbb{N}$. If not, reject.
   (b) If $k > n$, reject.
   (c) Guess a bit vector $c_1 \cdots c_n$ of length n with exactly k 1's.
   (d) For each edge (i, j) check whether $c_i = 1$ or $c_j = 1$.
   (e) If yes, accept. Otherwise reject.

   The Turing machine is a halting NTM with polynomial time complexity and accepts VC.
2. See the slides of the next lecture.

Exercise 4: Bonus (Implementation)

Implement the algorithm for Clique from Slide 16. Choose any way to make Step 3 deterministic (e.g., using backtracking together with random guessing or systematic enumeration of all bit vectors). Run the algorithm on some random graphs and measure the running time.