CMSC 28100-1 / MATH 28100-1
Introduction to Complexity Theory
Fall 2017 – Homework 8
Solution

November 28, 2017

Exercise 1. Show that if P = NP, then every language in NP is NP-complete, except for $\varnothing$ and Σ^*.

Solution. Let $L \in \mathrm{P}$ be such that $L \neq \varnothing$ and $L \neq \Sigma^*$. Then there exists $x_L \in L$ and $y_L \in \Sigma^* \setminus L$. Since $\mathrm{P} \subseteq \mathrm{NP}$, it follows that $L \in \mathrm{NP}$.

Let now $L' \in \mathrm{NP}$ be an arbitrary language in NP and let us show that $L' \leqslant_p L$.

Since we suppose P = NP, we know that there exists a polynomial-time Turing machine M deciding L'. Consider then the algorithm R that runs M in its input and outputs x_L if M accepts and outputs y_L if M rejects. Clearly R runs in polynomial time. Furthermore, we clearly have $R(x) \in L$ if and only if $x \in L'$, hence R is a polynomial-time reduction of L' to L.

Therefore L is NP-complete. ◁

Exercise 2. Consider the following language:

$$K = \{(M, x, 1^t) : M \text{ is an NTM that accepts } x \text{ within } t \text{ steps}\}.$$

(a) Show that $K \in \mathrm{NTIME}(n)$.

(b) Show directly (not by reduction from another known NP-complete language) that K is NP-complete.

Solution. For item (a), let N be the non-deterministic Turing machine that on input $(M, x, 1^t)$, simulates M on x for t steps, accepting if M accepts (we assume that M is single-taped, so it can be simulated by N).

Clearly N decides K in time $O(t + |x|)$, which is $O(n)$. By linear acceleration, we get $K \in \mathrm{NTIME}(n)$.

For item (b), we have already proved that $K \in \mathrm{NTIME}(n) \subseteq \mathrm{NP}$, so it remains to show that every NP language is polynomially reducible to K.

Fix an NP language L. Then there exists an NTM N deciding L in time n^k for some $k \in \mathbb{N}_+$.

Consider then the (deterministic) algorithm R that on input x outputs $(N, x, 1^{|x|^k})$. Note that one such R can be made in time $O(|x|^k)$, i.e., in polynomial time. Note also that $R(x) \in K$ if and only if x is accepted by N in time $|x|^k$, which is if and only if $x \in L(N) = L$ as N is n^k-time bounded. Hence R reduces L to K.

Therefore K is NP-complete. ◁

Exercise 3. Show that if SAT $\in$ P, then there is a deterministic polynomial-time Turing machine M such that for all formulas ϕ, if ϕ is satisfiable then $M(\phi)$ outputs a satisfying assignment to ϕ, and otherwise M rejects. This is called solving the "search version" of SAT (searching for a witness, rather than merely determining if one exists).

Solution. Suppose SAT $\in$ P, that is, suppose there exists a polynomial-time Turing machine M that decides SAT.

Without loss of generality, we assume that all variables in the input appear in the formula (this is easy to detect and treat in polynomial time). Consider the following algorithm.

Algorithm 3.1: Finding an assignment in SAT.

```
Given a formula φ on the variables x_1, ..., x_n, do the following.
Run M on φ.
if M rejects then Reject.
else
    φ_0 ← φ.
    for i ← 1 to n do
        Let ψ be the formula obtained from φ_{i−1} by assigning x_i to 1 (i.e., true) and
         simplifying.
        Run M on ψ.
        if M accepts then
            a_i ← 1.
            φ_i ← ψ.
        else
            a_i ← 0.
            Let φ_i be the formula obtained from φ_{i−1} by assigning x_i to 0 (i.e., false) and
             simplifying.
return (a_1, ..., a_n).
```

It is easy to see by induction that in the loop of the algorithm the formula ϕ_i is always satisfiable and is obtained from ϕ by assigning some variables to fixed values. From this it follows that if a satisfying assignment exists, the algorithm finds it (and of course the algorithm rejects formulas not in SAT).

Note also that assigning values and simplifying formulas can be done in polynomial time, so the algorithm above runs in polynomial time (as it runs M a total of n times). $\triangleleft$

Exercise 4. A language L is *p-selective* if there is a polynomial-time (deterministic) Turing machine M such that the following hold.

(a) For every $x, y \in \Sigma^*$, we have $M(x, y) \in \{x, y\}$, that is, on input (x, y), the machine M outputs either x or y.

(b) For every $x, y \in \Sigma^*$, if at least one of x and y is in L, then M outputs a string in L (which must be either x or y by (a)).

Show that if SAT is p-selective, then P = NP. Hint: use ideas from Exercise 3.

Solution. Suppose SAT is p-selective and let M be a polynomial-time Turing machine with the properties of p-selection for SAT.

Consider the following algorithm. (We again assume that all variables appear in ϕ.)

Algorithm 4.1: Finding an assignment in SAT.

Given a formula ϕ on the variables $x_1, \ldots, x_n$, do the following.
$\phi_0 \leftarrow \phi$.
for $i \leftarrow 1$ **to** n **do**
    Let $\psi^\top$ and $\psi^\perp$ be the formulas obtained from ϕ_{i-1} by assigning x_i to 1 and 0 respectively and simplifying.
    Run M on $(\psi^\top, \psi^\perp)$.
    if M *returns* $\psi^\top$ **then**
        $a_i \leftarrow 1$.
        $\phi_i \leftarrow \psi^\top$.
    else
        $a_i \leftarrow 0$.
        $\phi_i \leftarrow \psi^\perp$.
if $(a_1, \ldots, a_n)$ *is a satisfying assignment* **then** Accept.
else Reject.

Since the algorithm above runs M a total of n times and all constructions and tests done by the algorithm are polynomial, it follows that the algorithm is polynomial.

Clearly if $\phi \notin$ SAT, then the algorithm above rejects ϕ as no assignment satisfies ϕ.

Suppose then that $\phi \in$ SAT. It is easy to see by induction that the p-selection property of M implies that each ϕ_i is satisfiable. Since ϕ_i is obtained from ϕ by assigning $x_j \leftarrow a_j$ for every $j \in \{1, \ldots, i\}$, it follows that $\phi_n = \phi(a_1, \ldots, a_n)$ and ϕ_n is tautologically true, hence the algorithm accepts ϕ.

Therefore the algorithm decides SAT in polynomial time, which implies SAT $\in$ P.

Since SAT is NP-complete, this implies that P = NP. ◁