Instructions: (i) Submit a written assignment, or email to Khoa by the deadline. (ii) Submit one writeup per
group – other resources including the Web are not allowed for consultation. Write your solutions neatly. (iii) For all problems, we assume that we have fixed the alphabet Σ = {0, 1}, and hence that all languages we are referring to are subsets of {0, 1}∗.

1. We discussed the following ancient GCD algorithm in class, to find the GCD of two positive integers a and b:

\[
\text{gcd}(a, b) \{
\text{if } (a > b) \text{ then return}(\text{gcd}(b, a)); \\
\text{else if } (a \text{ divides } b) \text{ then return}(a) ; \\
\text{else } \\
\quad x = (b \text{ mod } a); /* Comment: the remainder got when dividing } b \text{ by } a */ \\
\quad y = a ; \\
\text{return}(\text{gcd}(x, y));
\}
\]

(i) Show that there is a constant \(c < 1\) with the following property: whenever we enter the final “else” block, the integers \(x\) and \(y\) that we compute there satisfy \(x + y \leq c(a + b)\).

(ii) Use part (i) to show that our algorithm terminates in polynomial time.

2. Suppose languages \(L_1, L_2\) lie in NP. Then, is \(L_1 \cup L_2 \in NP\)? Is \(L_1 \cap L_2 \in NP\)? Justify your answers.

3. Suppose that for all \(L_1, L_2 \in NP\), we also have \(L_1 - L_2 \in NP\). Which two complexity classes coincide in this case?

4. As mentioned above, assume that our alphabet Σ is, say, \(\{0, 1\}\); let \(\leq_p\) be the symbol that denotes polynomial-time reductions, as defined in class.

(a) Identify all languages \(L\) for which \(\emptyset \leq_p L\) (as usual, \(\emptyset\) refers to the empty set).

(b) Identify all languages \(L\) for which \(\Sigma^* \leq_p L\) (as usual, \(\Sigma^*\) is the language of all finite-length strings composed from the alphabet Σ).

(c) Prove that the “\(\leq_p\)” relation is not symmetric: display languages \(L_1, L_2\) with \(L_1 \leq_p L_2\), but \(L_2 \not\leq_p L_1\).

5. Let \(L_0 \subseteq \Sigma^*\) be the following special type of language: for some infinite sequence of positive integers \(i_1 < i_2 < i_3 < \cdots\), \(L_0 = \{1^{i_1}, 1^{i_2}, 1^{i_3}, \ldots\}\). (Recall that the string \(1^i\) is the string of \(i\) ones.) Prove that if \(SAT \leq_p L_0\), then there is a polynomial-time algorithm for SAT. The following hints could be useful.

Let \(f\) be the reduction from SAT to \(L_0\), and let \(g(x)\) be the positive integer such that \(f(x) = 1^{g(x)}\). Prove that there is some constant \(c\) such that for any given instance \(\phi\) of SAT, with any length \(N\) and any number \(n \leq N\) of Boolean variables, \(g(\phi) \leq N^c\). Let \(x_1, x_2, \ldots, x_n\) be the variables in \(\phi\). Let \(\phi_0\) be the simplification of \(\phi\) obtained by setting \(x_1 := 0\), and \(\phi_1\) be the simplification of \(\phi\) obtained by setting \(x_1 := 1\). Imagine a binary tree \(T\) with root \(\phi\), whose children are \(\phi_0\) and \(\phi_1\). We can similarly create children for \(\phi_0\) and \(\phi_1\): \(\phi_{00}\) is the simplification of \(\phi_0\) obtained by setting \(x_2 := 0\), and \(\phi_{01}\) is the simplification of \(\phi_0\) obtained by setting \(x_2 := 1\). We can analogously create the children \(\phi_{10}\) and \(\phi_{11}\) of \(\phi_1\). Continuing this way, we imagine the full binary tree \(T\) of depth \(n\). Do a suitable search of \(T\). In particular, how can you relate the satisfiability of a node \(u \in T\) with the satisfiabilities of \(u\’s \) two children?