CSCI 3130: Formal Languages and Automata Theory — Homework 6
The Chinese University of Hong Kong, Fall 2020 — due 11:59pm Thursday December 10

Collaborating on homework is encouraged, but you must write your own solutions in your own words and list your collaborators. Copying someone else's solution will be considered plagiarism and may result in failing the whole course.

Please answer clearly and concisely. Explain your answers. Unexplained answers will get lower scores or even no credits.

(1) (30 points) For each of the following problems, show that it is NP-complete: Namely, (1) it is in NP; and (2) some NP-hard language reduces to it. When showing NP-hardness, you can start from any language that was shown NP-hard in class or tutorial.

(a) $L_1 = \{\langle G, k\rangle \mid \text{Graph } G \text{ contains (at least) two vertex covers, each of size } k\}$

(b) $L_2 = \{\langle S_1, S_2, \ldots, S_m, \ell\rangle \mid \text{Some subset of size } \ell \text{ intersects all the sets } S_1, S_2, \ldots, S_m\}$.
A subset W intersects all the sets $S_1, S_2, \ldots, S_m$ if for every $1 \leqslant j \leqslant m$, some element in S_j also belongs to W.
For example, if $S_1 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, S_2 = \{\mathtt{c}, \mathtt{d}\}, S_3 = \{\mathtt{b}, \mathtt{e}\}$, then $W = \{\mathtt{c}, \mathtt{e}\}$ intersects all the sets S_1, S_2, S_3.
Hint: Reduce from Vertex Cover.

(2) (30 points) For each of the following languages, suppose some polynomial-time algorithm A decides the corresponding *decision* problem. Using A, give a polynomial-time algorithm to *search* for a solution, whenever such a solution exists.

Briefly justify your algorithm, e.g. by giving an invariant. Insufficient explanation will get zero points.

(a) $L_1 = \{\langle G, s, t\rangle \mid \text{Graph } G \text{ has a Hamiltonian path from vertex } s \text{ to vertex } t\}$
Recall that a Hamiltonian path in G visits every vertex of G exactly once.

(b) $L_2 = \{\langle G\rangle \mid \text{Graph } G \text{ has a clique of size exactly } n/2, \text{ where } n = |V(G)|\}$
In other words, we are looking for a clique on exactly half of the vertices.

(3) (20 points) Throughout the semester, we looked at various models of computation and we came up with the following "hierarchy" of classes of languages:

$$\text{regular} \subseteq \text{context-free} \subseteq \text{P} \subseteq \text{NP} \qquad \text{decidable} \subseteq \text{recognizable}$$

We also gave examples showing that the containments are strict (e.g., a context-free language that is not regular), except for the containment $\text{P} \subseteq \text{NP}$, which is not known to be strict.

There is one gap in this picture between NP languages and decidable languages. In this problem you will fill this gap.

(a) Show that 3SAT is decidable, and the decider has running time $2^{O(n)}$. (Unlike a *verifier* for 3SAT which is given a 3CNF formula φ together with a potential satisfying assignment for φ, a *decider* for 3SAT is only given a 3CNF formula but not an assignment for it.)

(b) Argue that for every NP-language L there is a constant c such that L is decidable in time $2^{O(n^c)}$. (Use the Cook–Levin Theorem.)

(c) Let L' be the following language:

$$L' = \{\langle M, w\rangle \mid \text{Turing machine } M \text{ does not accept input } \langle M, w\rangle \text{ within } 2^{2^{|w|}} \text{ steps}\}.$$

It is not hard to see that L' can be decided in time $O(2^{2^n})$.

Show that L' cannot be decided in time $2^{O(n^c)}$ for any constant c, and therefore it is not in NP.

Hint: Assume that L' can be decided by a Turing machine D in time $2^{O(n^c)}$. What happens when D is given input $\langle D, w\rangle$, where w is a sufficiently long string?

(4) (20 points) A *heuristic* is an algorithm that often works well in practice, but may not always produce the correct answer. In this problem, we will consider a heuristic for Dominating Set.

A vertex subset S is a dominating set of a graph G if for every vertex v in G, $v \in S$ or v is adjacent to some vertex in S.

The Dominating Set problem asks whether a graph G has a dominating set of size at most k:

$$\text{DS} = \{\langle G, k\rangle \mid \text{Graph } G \text{ has a dominating set of size at most } k\} .$$

Recall that the *degree* of a vertex is the number of edges incident to it. Consider the following heuristic A for DS:

Algorithm 1 GREEDYDS(G, k)

Require: G is a graph, k is a nonnegative integer

1: Initialize $S = \emptyset$ and $H = G$
2: **while** H contains at least one vertex **do**
3: Let v be the vertex of maximum degree in H
4: Add v to the set S
5: Let $N(v) = \{u \in V(H) \mid u = v \text{ or } u \text{ is adjacent to } v \text{ in } H\}$ be the neighborhood of v
6: Update $H = H \setminus N(v)$ by removing from H all vertices in $N(v)$ (and edges incident to $N(v)$)
7: **end while**
8: **accept** if and only if $|S| \leqslant k$

We assume the vertices in the input graph G are labelled from 1 to n. On line 3, if multiple vertices have the maximum degree, we break ties by picking the one with the smallest label.

(a) Show that A runs in polynomial time.

(b) Show, using a loop invariant, that S is guaranteed to be a dominating of G at the end of the while loop.
(Note: As a result, if A accepts $\langle G, k\rangle$, then $\langle G, k\rangle \in$ DS)

(c) Show that it is possible that A rejects $\langle G, k\rangle$, even though $\langle G, k\rangle \in$ DS.
Give such an instance $\langle G, k\rangle$ where the graph G contains at most 10 vertices. Explain why your instance belongs to DS (by giving a dominating set of size k in G), and why heuristic A rejects $\langle G, k\rangle$.