Computational Complexity (INFR11102) - 2018 Fall
Supplementary Exercises

These questions are for your own practise. Feel free to contact me if you want to know the answers to some questions.

Heng Guo, 8th November 2018
hguo@inf.ed.ac.uk

1. Show that $\texttt{PSpace} \neq \texttt{DTime}[2^n]$.

2. A *feedback arc set* is a subset $A \subset E$ in a directed graph $D = (V, E)$, if the removal of A makes the graph acyclic. Let the Feedback Arc Set (FAS) problem be the following.

   Name: FAS
   Input: A directed graph $D = (V, E)$, and an integer k.
   Output: Does D has a feedback arc set of size at most k?

   Show that FAS is NP-complete.

   Hint: Reduce from vertex cover.

   Remark: Find the minimum feedback vertex set is NP-hard even in undirected graphs. On the other hand, in undirected graphs, find a minimum feedback arc set is the same as finding a spanning tree, which can be solved in polynomial-time!

3. An *integer program* is a set of linear inequalities over a set of variables $x_i \in \{0, 1\}$ for $1 \leq i \leq n$, with rational coefficients. It is feasible if there exists at least one solution.

   Name: IntProg
   Input: An integer program P.
   Output: Is P feasible?

   Show that IntProg is NP-hard.

   Hint: Reduce from 3-Sat.

   Remark: On the other hand, without the integral requirement, the program becomes linear program, and checking its feasibility is in P!

4. In Turing's original paper in 1937, he defined oracle Turing Machines. Let O be a language. We equip a TM M with an oracle O, denoted M^O, if M has a special tape, and during the execution of M, it can write x on the special tape, and ask the oracle to get whether $x \in O$ in one step. Oracle TMs relate to *Turing reductions*. Then we say A is *Turing reducible* to B, denoted $A \leq_{\texttt{t}} B$, if there exists a TM M with oracle B that computes A. Again, the intuition of this notation is that A is easier than B. The oracle should be thought of as some subroutine that we can invoke.

   Show that

(a) $\Sigma_2^p = \mathtt{NP}^{\mathrm{SAT}}$;

(b) if $\mathtt{NP} \neq \mathtt{coNP}$, then $\mathtt{NP}$ is *not* closed under Turing reductions.

Here, we say a class $\mathtt{C}$ is closed under Turing reductions, if $L \in \mathtt{C}$ and $L' \leq_{\mathtt{t}} L$ imply $L' \in \mathtt{C}$.

Remark: Part (b) is one of the reasons that we do not use Turing reductions to define NP-completeness.

5. (a) Show that if a function $f : \{0,1\}^n \to \{0,1\}$ has a support of size k, then it can be decided by a circuit of size $O(nk)$.

   (The support of f is the set of inputs that map to 1, namely $\mathrm{Supp}(f) = \{\mathbf{x} \mid f(\mathbf{x}) = 1\}$.)

   (b) Show that, for sufficiently large n, there is a language that can be decided by circuits of size n^3 but not n^2.

   Hint: use a counting argument.

6. (a) Deciding whether a given undirected graph has a Hamiltonian cycle (a simple cycle containing all vertices) is NP-complete. Denote this problem by HamCycle, and by HamPath the problem of deciding whether a given graph has a Hamiltonian path (a simple path containing all vertices).

   Show that HamPath is NP-complete. For the NP-hardness part, you should reduce from HamCycle.

   (b) Counting Hamiltonian cycles in *directed* graphs (denoted #DHamCycle) is #P-complete.

   Show that counting cycles of all lengths in *directed* graphs is #P-complete. Call this problem #DCycle. For the #P-hardness, you should show a reduction #DHamCycle $\leq_t$ #DCycle.

   Hint: use interpolation.

7. Recall the isolation lemma. Use it to show that $\mathtt{BPP} \subseteq \Sigma_2^p$.