CMPUT 675 - Winter 2019
Assignment #2 - Due Feb 26 (Tuesday) by 12:30pm

All **Exercises** have the same weight, regardless of their difficulty. You are allowed to skip one exercise freely with no penalty. If you answer all questions, I will drop the one with the lowest mark when computing your mark for this assignment.

It is highly recommended that you typeset your solutions in LaTeX. I still want hard copies of your solution, so submit a printout if you do typeset it.

Pages: 3

Exercise 1: Easy Warmup

Show that if SAT $\leq_p \overline{\text{SAT}}$ then **PH** = **NP**.

Exercise 2: Efficient Zero-Error Algorithms for NP

Show that if any **NP**-complete language lies in co-**RP** then **NP** = **ZPP**.

Exercise 3: Verifiers for NL

In the lectures, we defined verifiers for **NL** as those where the head of the second tape (the *proof tape*) cannot move left.

Consider the slightly weaker restriction where the verifier is simply restricted to read-only access to the proof tape but the head may move back and forth. Specifically, say that **NL**$'$ is all languages L such that there is a verifier M expecting inputs x, y on two separate *input tapes* where:

- The two input tapes containing x and y (respectively) are *read-only*.
- M uses $O(\log |x|)$ space on the remaining work tapes.
- M only reads up to the first $p(|x|)$ bits of y for some polynomial p.
- If $x \in L$ then there is some $y \in \{0,1\}^{p(|x|)}$ such that $M(x, y) = \texttt{accept}$.
- If $x \notin L$ then for any $y \in \{0,1\}^{p(|x|)}$ we have $M(x, y) = \texttt{reject}$.

Show that **NL**$'$ = **NP**.

Exercise 4: $\sum_k$SAT

Show for every integer $k \geq 1$ that $\sum_k$SAT is complete (with respect to Karp reductions) for $\sum_k^p$.

Hint, essentially from the book
One can avoid an involved "Cook-Levin-style reduction" by simply using the fact that TMSAT $\leq_p$ SAT as a black box.

Exercise 5: Intersections of Different Languages in NP and co-NP

Let **DP** be all languages L such that there are two languages $L_1 \in$ **NP** and $L_2 \in$ co-**NP** and $L = L_1 \cap L_2$. **Note**: this is not necessarily (and likely not) the same as **NP** $\cap$ co-**NP**.

Recall ExactIndSet $= \{(G, k) :$ the largest independent set in G has size exactly $k\}$.

- Show ExactIndSet $\in$ **DP**.
- Show $L \leq_p$ ExactIndSet for every $L \in$ **DP**.

Hint: Recall we already proved that IndSet is **NP**-complete in Lecture 4. This is also found in Theorem 2.15 from the book.

Exercise 6: Even Smaller Circuits

In the lectures, we briefly discussed how any $f : \{0,1\}^n \to \{0,1\}$ has a circuit of size $O(n \cdot 2^n)$: for example the circuit could be simply the OR of a bunch of AND-clauses that are indicators for each x with $f(x) = 1$. In this exercise, you will get asymptotically better results.

Easier - For Partial Credit: Show every $f : \{0,1\}^n \to \{0,1\}$ has a circuit of size at most $c \cdot 2^n$ for some absolute constant c (i.e. that does not depend on n).

Harder - For Full Credit: For all $n \geq 1$, show every $f : \{0,1\}^n \to \{0,1\}$ has a circuit of size at most $c' \cdot 2^n/n$ for some absolute constant c'.

Hint from the book (for the easier part)
Consider writing

$$f(x_1, \ldots, x_n) = (x_n \wedge f'(x_1, \ldots, x_{n-1})) \vee (\overline{x_n} \wedge f''(x_1, \ldots, x_{n-1}))$$

for some appropriate functions f', f'' over $\{0,1\}^{n-1}$ and use induction on n.

Hint for the harder part
Start with the same idea, but include "memoization" of circuits depending on only a few inputs to avoid recalculating the same circuit multiple times. Here "a few" is a slow growing function of n.

Exercise 7: Circuit Lower Bounds in PH

For every $k \geq 1$, show **PH** contains a language L that requires circuits of size $\Omega(n^k)$. That is, $L \notin$ **SIZE**$(f(n))$ for any $f(n) \in o(n^k)$.

Hint from the book
Keep in mind the proof of *existence* of functions with high complexity, you might want to look at the proof of the Nonuniform Hierarchy Theorem (Theorem 6.22 in the book) to see an example of how to project this hardness down to smaller size bounds than $O(2^n/n)$. Try to show that you can compute the *lexicographically smallest* such function using a constant number of quantifiers for some notion of lexicographically smallest.

It is possible to find languages L in $\sum_2^p$ that require circuits of size $\Omega(n^k)$ for any fixed k, but you can use any (fixed) number of quantifications you require to solve the problem.

Exercise 8: Exploring Randomized Reductions

Say that a probabilistic Turing Machine M computes a *randomized reduction* from a language L' to a language L if, when given $x \in \{0,1\}^*$, M randomly computes some output[1] $M(x) \in \{0,1\}^*$ in polynomial time such that:

- $x \in L \Rightarrow \mathbf{Pr}[M(x) \in L'] \geq 2/3$
- $x \notin L \Rightarrow \mathbf{Pr}[M(x) \in L'] \leq 1/3$

Generally, we say $L \leq_r L'$ if there is a polynomial-time probabilistic TM M that computes a randomized reduction from L to L'.

- Show that if SAT $\leq_r L$ for some $L \in \mathbf{P}$ then $\mathbf{NP} \subseteq \mathbf{BPP}$.
- The relation $\leq_r$ is not, strictly speaking, transitive. But "practically speaking" it is: show that if there are languages L, L' such that SAT $\leq_r L$, $L \leq_r L'$, and $L' \in \mathbf{P}$ then $\mathbf{NP} \subseteq \mathbf{BPP}$.
- Now consider all languages randomly-reducible **to** SAT: define $\mathbf{BP} \cdot \mathbf{NP} = \{L : L \leq_r \text{SAT}\}$. The motivation for this class is to consider what problems could be solved by an efficient randomized algorithm if we could solve SAT efficiently. Clearly $\mathbf{NP} \subseteq \mathbf{BP} \cdot \mathbf{NP}$, but does it capture more languages?

  Maybe not, show that if $\overline{\text{SAT}} \in \mathbf{BP} \cdot \mathbf{NP}$ then $\mathbf{PH} = \sum_3^p$.

Hint for the last part, from the book
Follow the ideas of the proof of the Karp-Lipton Theorem showing if $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}}$ then $\mathbf{PH} = \sum_2^p$.

[1] Temporarily override our notation of $M(x) \in \{\text{Accept}, \text{Reject}\}$ for this exercise. Here, $M(x)$ is the contents of some tape (designated the output tape) when M halts given input x.