CSE531: Complexity Theory May 29, 2020

Homework 3

Anup Rao Due: June 5, 2020

Read the fine print[1]. Each problem is worth 10 points:

1. Recall the following function:

$$2\mathbf{COL}(G) = \begin{cases} 1, & \text{if graph } G \text{ has a coloring with two colors} \\ 0, & \text{otherwise,} \end{cases}$$

   where a coloring of G with c colors is an assignment of a number in $[c]$ to each vertex such that no adjacent vertices get the same number.

   Prove that $2\mathbf{COL} \in \mathbf{NL}$. You can use the following fact: A graph G can be colored with two colors if and only if it contains no cycle of odd length.

2. Suppose **TQBF** is also **PSPACE**-complete under log-space reductions—meaning that for every $f \in \mathbf{PSPACE}$, there is a logspace computable function h such that $f(x) = \mathbf{TQBF}(h(x))$. Prove that this implies that $\mathbf{TQBF} \notin \mathbf{NL}$. Hint: Use Savitch's theorem and one of the hierarchy theorems.

3. In class we showed that any monotone circuit computing whether or not a graph has a large matching has depth at least $\Omega(n)$. Use similar ideas to prove that any monotone circuit computing whether or not the graph has a clique of size k on graphs of size n must have depth $\Omega(n)$. Hint: Use the circuit to give a randomized algorithm for the disjointness problem, as we did for the case of matchings. Alice should use her input set to make a graph on $3n + 2$ vertices that has a clique of size $n + 2$. Bob should use his input set to make an $n + 1$-partite graph on $3n + 2$ vertices, which does not have such a clique. Argue that if the monotone circuit has depth d, then this yields a randomized protocol for computing disjointness with communication complexity $O(d)$.

4. An arithmetic circuit is the same as a boolean circuit, except that every gate either computes the product of the two inputs, or the sum of the two inputs. One can also have integer constants that feed into the circuit. The circuit maps inputs in $\mathbb{R}^n$ to a real number $\mathbb{R}$. The circuit can also be thought off as encoding a polynomial.

[1] In solving the problem sets, you are allowed to collaborate with fellow students taking the class, but **each submission can have at most one author**. If you do collaborate in any way, you must acknowledge, for each problem, the people you worked with on that problem. The problems have been carefully chosen for their pedagogical value, and hence might be similar to those given in past offerings of this course at UW, or similar to other courses at other schools. Using any pre-existing solutions from these sources, for from the web, constitutes a violation of the academic integrity you are expected to exemplify, and is strictly prohibited. Most of the problems only require one or two key ideas for their solution. It will help you a lot to spell out these main ideas so that you can get most of the credit for a problem even if you err on the finer details. Please justify all answers. Some other guidelines for writing good solutions are here: `http://www.cs.washington.edu/education/courses/cse421/08wi/guidelines.pdf`.

(a) Suppose you are given two arithmetic circuits such that every gate of each circuit computes a polynomial of degree at most n, and the coefficients of the polynomial is promised to have magnitude at most 2^n. Use the Schwartz-Zippel lemma to give a randomized algorithm in **RP** to decide whether the two polynomials are equal or not. (Be careful when analyzing your agorithm: if x, y are numbers, then $x \times y$ can be significantly larger. You need to make sure that the numbers do not become so big that your algorithm is unable to multiply them!).

(b) Suppose you are given two arithmetic circuits, with no other promises. Give a randomized algorithm in **RP** to decide whether the polynomials are the same or not. To do this:

i. Prove that the degree of the polynomials computed by the circuits is at most 2^s, where s is the size of the larger circuit.

ii. Now, the problem is that we cannot evaluate these circuits on large integers in polynomial time, because the size of the integers might become exponentially large. However, if p is a prime of size at most 2^s, then we can evaluate these circuits modulo p in polynomial time, by carrying out all arithmetic modulo p. Use this fact to give a randomized algorithm to decide whether the polynomials are the same or not in **RP**. It might be useful to use the fact that there are many primes of size n, as we did in the proof of $IP = PSPACE$.

(c) Suppose you could give a polynomial time randomized algorithm to tell whether or not two *boolean* circuits compute the same boolean function. What consequence would this have with regards to the relationship between the classes $\mathbf{P}, \mathbf{BPP}, \mathbf{NP}$?