CMPUT 675 - Winter 2019
Assignment #3 - Due March 7 (Thursday) by 12:30pm

All **Exercises** have the same weight, regardless of their difficulty. As you have a bit less time to solve this one after the due date of the previous assignment (and since I released this one day later than usual), you are allowed to skip **two** exercises freely with no penalty. If you answer more than what is required, I will drop the ones with the lowest marks 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. As always, if any question is not clear then please feel free to ask me for clarification.

Pages: 3

Exercise 1: Red-Blue Matching Parity

Let $G = (X \dot{\cup} Y; E)$ be a bipartite graph with sides X, Y where $|X| = |Y| = n$. Let $B \subseteq E$ be **blue** edges and let $R = E - B$ be **red** edges. Finally, let k be an integer.

We discussed how to determine if there is a perfect matching M in G with $|M \cap B| = k$ with an efficient randomized algorithm. But no deterministic counterpart is known (yet). Presently, we can at least determine the **parity** of the number of perfect matchings M with $|M \cap B| = k$.

Describe a **deterministic**, polynomial-time algorithm that decides if the **number** of perfect matchings M with $|M \cap B| = k$ is **odd** or **even**.

You may assume without proof that the determinant of an integer matrix can be computed in polynomial time in the total size of the matrix (including the number of bits to write the entries). Though, I would be happy to see you fully address this issue head-on or provide a workaround. No extra marks for this, just brownie points :)

Exercise 2: Details Behind IP $\subseteq$ PSPACE

Recall we said **IP** $\subseteq$ **PSPACE** because we can "compute" the optimum prover P for a given input $x \in \{0,1\}^*$ in polynomial space. Carefully describe this in detail.

Exercise 3: Exploiting Bad Programs For Good!

We have been studying polynomial-time verifiers that do not trust the prover but are willing to accept a valid proof they have the ability to check. Here is something in this vein that is a bit different than what we have done so far, but is still pretty neat.

The **P** vs. **NP** problem has garnered so much attention that there is a rich body of bad proofs claiming to resolve it! Here is an impressive catalog of such "proofs":
`https://www.win.tue.nl/~gwoegi/P-versus-NP.htm`
Some of these claim to solve particular **NP**-complete problems in polynomial time and are so complicated that it is hard to find a counterexample to this claim.

Let us consider SAT. The idea of this exercise is that if ψ is a CNF that is deemed satisfiable by one of these "bad" algorithms, then you can at least exploit it to efficiently construct a satisfying assignment for ψ or discover there is *some* input where M does not work as claimed. That is, you either discover *how* to satisfy ψ or you come up with a proof that M does not solve SAT in polynomial time.

Specifically, suppose someone gives you a deterministic Turing machine M they claim decides SAT in at most $p(|x|)$ steps for each $x \in \{0,1\}^*$ where $p(n)$ is some polynomial that is also given to you by this person.

Describe a deterministic, polynomial-time Turing machine M' such that for every CNF ψ where $M(\psi) = \text{ACCEPT}$ (i.e. is deemed satisfiable by M), when providing ψ to M' either:

- M' will construct an actual satisfying assignment for ψ, or
- M' discovers that the claimed performance of M is incorrect. That is, M' determines correctly that there is some CNF ϕ (perhaps different than ψ) where either M decides ϕ incorrectly or M takes more than $p(|\phi|)$ steps on ϕ. It is not necessary for M' to actually provide the particular ϕ that M performs incorrectly on, it just has to determine that one exists.

Exercise 4: Parallel Repetition

Mostly problem 8.6 from the book, but slightly more general. For 70% credit, you can answer problem 8.6 from the book as it is stated.

Here you will prove that for $L \in \mathbf{AM}[2]$, the acceptance probability of $\geq \frac{2}{3}$ for $x \in L$ and $\leq \frac{1}{3}$ for $x \notin L$ can be improved in a way that is similar to how we improved the probabilities for **BPP** to be exponentially close to 1 or 0, but *without increasing the number of rounds of communication*: the improvement will still use two rounds, one being the public coins of the verifier and one being the answer of the prover.

Consider a verifier V and a prover P in an $\mathbf{AM}[2]$ interaction for deciding L. That is, given some $x \in \{0,1\}^*$ the verifier V samples and sends random coins/bits $a \sim \{0,1\}^{p(|x|)}$ where $p(n)$ is some polynomial. The prover P replies with message $b = P(x,a) \in \{0,1\}^{q(|x|)}$ for some polynomial q. Finally, $V(x,a,b)$ *deterministically* either accepts or rejects the interaction.

Now suppose the protocol is repeated k times **in parallel**. That is, the verifier chooses k random bit strings $a_1, \ldots, a_k \sim \{0,1\}^{p(|x|)}$, each uniformly at random and independent of the others. The prover P then responds with an answer to each random bit string but may consider all a_i when deciding these responses. That is, P responds with $(b_1, \ldots, b_k) = P(x, a_1, \ldots, a_k)$ where each $b_i \in \{0,1\}^{q(|x|)}$. Note that b_i may not depend on x and a_i alone, it may also depend on other $a_j, j \neq i$.

Finally, have V accept the interaction if and only if $V(x, a_i, b_i)$ is accepted for the majority of indices $1 \leq i \leq k$. Show there is some constant c such that:

- If $x \in L$ then there is a prover P such that the k-round parallel repetition of the protocol is accepted by V with probability $\geq 1 - c^{-k}$.
- If $x \notin L$ then for any prover P the k-round parallel repetition of the protocol is accepted by V with probability $\leq c^{-k}$.

Hint: The calculations are standard Chernoff-bound calculations. But one has to also argue why allowing the prover access to all random bit strings does now allow them to coordinate the k answers better than if they were k independent rounds of interaction.

Note: In other fascinating settings we won't get to in the course, parallel repetition like this does not improve the probabilities as well as if we were considering k independent rounds of interaction. Feel free to ask me about it if you have some spare time!

Another Note: One can also get the acceptance probability to be exactly 1 in the $x \in L$ case in an $\mathbf{AM}[2]$ interaction much like we showed $\mathbf{BPP} \subseteq \Sigma_2^p$ by considering random translations of the sets of random strings for Arthur that allow Merlin to make Arthur accept.

Exercise 5: Brief Arthur-Merlin Interactions

This is problem 8.7 from the book.

Show for any constant $k \geq 2$ that $\mathbf{AM}[k] = \mathbf{AM}[2]$.

Hint, mostly from the book but slightly different
Use induction on k. It would be helpful to first prove $\mathbf{MA}[2] \subseteq \mathbf{AM}[2]$ where $\mathbf{MA}[2]$ is analogous to $\mathbf{AM}[2]$ except the prover (Merlin) sends the first message b after seeing the instance of the $x \in L$? question, and the verifier (Arthur) randomly decides whether to accept or reject (x, b).

One can do this so that if $x \in L$ then some prover can still easily cause the verifier to accept the new $\mathbf{AM}[2]$ interaction with the same probability as the $\mathbf{MA}[2]$ interaction. But in the case $x \notin L$ you have to be more careful.

You may assume that for any $L \in \mathbf{MA}[2]$ and any constant c, there is a verifier V such that if $x \in L$ then some prover P causes V to accept the Merlin-Arthur interaction with P with probability at least $1 - 2^{-|x|^c}$ and if $x \notin L$ then for every prover P, V accepts the Merlin-Arthur interaction with P with probability at most $2^{-|x|^c}$. The proof is the same idea as in Exercise 4 (parallel repetition), except it is even slightly easier to see in the $\mathbf{MA}[2]$ setting so you don't have to prove it unless you really want to for more brownie points!

Exercise 6: Connecting Arthur-Merlin to Randomized Reductions

Show $\mathbf{BP} \cdot \mathbf{NP} = \mathbf{AM}[2]$. The definition of $\mathbf{BP} \cdot \mathbf{NP}$ is from the last question in the previous assignment, and also Chapter 7.6 from the book.

Note: In fact, under plausible assumptions about circuit lower bounds we have $\mathbf{AM}[2] = \mathbf{NP}$. But that's a lot more advanced than this exercise. Ultimately, stringing together many of the results we have seen in the lectures and the exercises shows $\mathbf{AM}[2] \subseteq \Sigma_3^p$. To be clear, this is just for your information; you just have to show the basic equality above.