Quantum Computation **CMU 15-859BB, Fall 2015**

HOMEWORK 6

Due: Wednesday December 9, 11:59pm; email the pdf to pgarriso@andrew.cmu.edu

---

If you have scribed twice, solve 2 out of 7.
If you have scribed once, solve 4 out of 7.

1. [**PGM on two qubits.**] Suppose you are given the state $|\psi_0\rangle = \cos(\theta)|0\rangle + \sin(\theta)|1\rangle$ with probability $\frac{1}{2}$ and the state $|\psi_1\rangle = \cos(-\theta)|0\rangle + \sin(-\theta)|1\rangle$ with probability $\frac{1}{2}$, for some angle θ. Write down the POVM matrices used by the Pretty Good Measurement in this case and compute the success probability.

2. [**The symmetric subspace.**] Given a permutation $\pi \in \mathrm{S}_n$, where S_n is the set of permutations of the numbers $\{1, \ldots, n\}$, let $P(\pi)$ be the matrix acting on $(\mathbb{C}^d)^{\otimes n}$ for which $P(\pi)|v_1\rangle \otimes \ldots \otimes |v_n\rangle = |v_{\pi(1)}\rangle \otimes \ldots \otimes |v_{\pi(d)}\rangle$ for any $|v_1\rangle, \ldots, |v_d\rangle \in \mathbb{C}^d$. Define

$$\mathrm{Sym}_B := \{|v\rangle \in (\mathbb{C}^d)^{\otimes n} \ : \ P(\pi)|v\rangle = |v\rangle \ \text{ for all } \pi \in \mathrm{S}_n\}.$$

(Recall that $|v\rangle \in \mathrm{Sym}_B$ may not be expressible as $|v_1\rangle \otimes \cdots \otimes |v_d\rangle$ for vectors $|v_1\rangle, \ldots, |v_d\rangle \in \mathbb{C}^d$, for the same reason that "entangled states exist".)

   (a) Check that Sym_B is a subspace of $(\mathbb{C}^d)^{\otimes n}$ and find a simple orthogonal basis for it.

   (b) Compute the dimension of Sym_B.

   (c) Recall from class the definition $\mathrm{Sym} := \mathrm{span}\{|v\rangle^{\otimes n} : |v\rangle \in \mathbb{C}^d\}$. Show that $\mathrm{Sym} \subseteq \mathrm{Sym}_B$.

   (d) In the case of $d = 2$, show that $\mathrm{Sym}_B \subseteq \mathrm{Sym}$. You may use the fact that the Vandermonde matrix is invertible. (In fact, $\mathrm{Sym} = \mathrm{Sym}_B$ for all n, d.)

3. [**Tomography lower bound.**] Consider the tomography problem in which one is given n copies of an unknown mixed state $\rho \in \mathbb{C}^{d \times d}$, and the goal is to output an estimate $\widetilde{\rho}$ such that $\mathrm{d}_{tr}(\rho, \widetilde{\rho}) \leq .01$. Show that this requires $n = \Omega(d^2/\log(d))$ copies of ρ. You may use the fact that for any d, there exist $m = \Omega(2^{d^2})$ mixed states $\{\rho_1, \ldots, \rho_m\}$ such that $\mathrm{d}_{tr}(\rho_i, \rho_j) \geq .2$ for all $i \neq j$.

Hitting the restart button. The following blurb pertains to the remaining problems. Suppose we have some kind of randomized model of computation $\mathcal{M}$. We define the *restarting* version of it, Restarting$\mathcal{M}$, as follows: At any time during the $\mathcal{M}$-computation, the machine is allowed to "hit the restart button" (if you like, think of this as entering a special kind of Turing Machine state). When this occurs, every aspect of the computation is set back to its initial value, and computation begins again. Nothing from the earlier computation is remembered. The machine may do multiple restarts, and each is called a *run*. Assuming that we are interested in the time complexity of $\mathcal{M}$, we define the time complexity of a Restarting$\mathcal{M}$ computation to be the *maximum* of the time of the runs (*not* the total time). Basically, if the $\mathcal{M}$-computation does not like the way a computation is going, it can freely restart (but has to forget everything).

A note: If a Restarting$\mathcal{M}$ has the property that it restarts with probability 1, we call it *invalid* and don't think of it as deciding any language. This is nothing unusual: it's just like what we do for Turing Machines that don't halt on all inputs. In particular, this means the "restarting" concept is not interesting for deterministic computation models.

We will mainly be concerned with "restarting" as applied to BPP, BQP, and PP; for example:[1]

Definition. RestartingPP *is the class of all decision problems* $f : \{0,1\}^* \to \{0,1\}$ *such that there is a (valid) restarting probabilistic algorithm Alg with the property that for all* $x \in \{0,1\}^*$ *we have* $\mathbf{Pr}[\mathrm{Alg}(x) = f(x)] > 1/2$. *(Here the Alg*$(x)$ *refers to the final output of the machine, possibly after several restarts. So you can think of the* $\mathbf{Pr}[\cdot]$ *as the probability conditioned on doing a non-restarting run.)*

4. **(On restarting BPP computations.)**

   (a) Explain why RestartingBPP has "efficient error amplification" just like BPP. In more details: RestartingBPP is defined as the class of decision problems for which there is a restarting probabilistic algorithm that outputs the correct answer (0 or 1) with probability at least θ, where θ is taken to be $3/4$. Explain why the definition of the class wouldn't change if θ were taken to be any quantity satisfying $1/2 + \frac{1}{\mathrm{poly}(n)} \leq \theta \leq 1 - 2^{-\mathrm{poly}(n)}$. You *don't* need to get into all the Chernoff bounds and mathematics that show why this is true for BPP; you can just assume them. Just explain why the same things work in the case of RestartingBPP.

   (b) Give the full details of the following proof sketch showing that SAT (and hence all of NP) is in RestartingBPP. "On input an n-variable Boolean formula F, guess an assignment $\alpha \in \{0,1\}^n$. If it satisfies F, output 1. Otherwise, restart with probability $1 - 2^{-n^2}$ and output 0 with probability 2^{-n^2}." (One thing you should include a sentence on: how much time does it take in our model to do an action with probability 2^{-n^2}?)

   (c) Give a very simple explanation for why UNSAT (and hence all of coNP) is also in RestartingBPP. Finally, show that SAT-UNSAT is in RestartingBPP, where SAT-UNSAT is the following problem: The input is a pair of Boolean formulas (F, G); the goal is to decide whether the following statement is true: "F is satisfiable and G is unsatisfiable".

   (d) (This part of the problem is for lovers of complexity theory only, and counts for 0 points.) Show that RestartingRP = NP, where RP is the one-sided-error version of BPP in which

[1] A technical point: As is tradition, we use somewhat bad notation like "RestartingBPP". In truth, BPP denotes a class of *decision problems* (languages). However, we often conflate it with the associated computational model: $\mathcal{BPP}$, the model of coin-flipping Turing machines that give the correct answer with probability at least 3/4. Technically, we should remember that $\mathcal{BPP}$ is a computational model, and say something like BPP $= L(\mathcal{BPP})$, the class of languages decided by that model. We bring this point up because Restarting is really an augmentation to the *model*, not the class of languages. I.e., we should really define Restarting$\mathcal{BPP}$ to be the *model* of coin-flipping Turing machines with the restart ability, that give the correct answer (once they finally halt) with probability at least 3/4. Then we can consider the class of languages L(Restarting$\mathcal{BPP}$) defined by this computational model. But for brevity, we just write RestartingBPP. If you know some classical complexity theory, you may appreciate the following point: Whereas it is a famous theorem that IP = PSPACE, just by knowing this it is not immediately clear whether or not RestartingIP = RestartingPSPACE. If we define $\mathcal{IP}$ to be the computational model of "probabilistic interactive proofs" and $\mathcal{PSPACE}$ to be the computational model of polynomial-space-bounded algorithms, the famous theorem asserts that $L(\mathcal{IP}) = L(\mathcal{PSPACE})$. But it's not immediately clear whether L(Restarting$\mathcal{IP}$) $= L$(Restarting$\mathcal{PSPACE}$), because the power of restarting may affect the models $\mathcal{IP}$ and $\mathcal{PSPACE}$ in different ways. For example, restarting is useless for the deterministic model $\mathcal{PSPACE}$, but it might be quite powerful for the randomized model $\mathcal{IP}$.

"no-inputs" must be rejected with probability 1. Also, show that RestartingBPP contains $\mathsf{P}_{\|}^{\mathsf{NP}}$, the class of languages decided by a deterministic polynomial-time algorithm that can make nonadaptive queries to a SAT oracle. (Note that SAT-UNSAT is one of the simplest examples of a language in $\mathsf{P}_{\|}^{\mathsf{NP}}$.)

5. **(On restarting PP computations, and BQP.)**

   (a) Show that RestartingPP = PP.

   (b) Show that RestartingBQP $\subseteq$ PP. (Since we proved in class that BQP $\subseteq$ PP, the notation suggests that RestartingBQP $\subseteq$ RestartingPP is an "obvious" consequence; then we get RestartingBQP $\subseteq$ PP by the previous problem. However, this consequence is actually not "obvious", for exactly the reason discussed in the footnote on page 2. Still, you should be able to prove the result by modifying our proof of BQP $\subseteq$ PP. You may freely use the following not-hard-to-prove "principle of deferred restarts": Without loss of generality, a RestartingBQP computation works as follows: a deterministic polynomial-time algorithm outputs a quantum CCNOT-Hadamard circuit with hard-coded inputs; then it measures the first two bits of the output. If the first bit is 1 then the algorithm restarts. Otherwise, its final answer is the second bit.)

6. **(Restarts are a convenient proof-tool, part I.)** Show that QMA $\subseteq$ PP. You may find it helpful to use the concept of "restarting".

7. **(Restarts are a convenient proof-tool, part II.)** The goal of this problem is to show that PP $\subseteq$ RestartingBQP. When combined with Problem 5, this shows that PP = RestartingBQP. When combined with some easy observations about closure properties of RestartingBQP (similar to Problems 4(a),(c)), this shows that PP is "closed under intersection": if $L_1, L_2 \in \mathsf{PP}$ then $L_1 \cap L_2 \in \mathsf{PP}$. This was an open problem in classical complexity theory from 1975 to 1995.

   (a) Suppose $L \in \mathsf{PP}$. Describe some simple "hacks" (like those in the Piazza post) that allow one to assume there is a probabilistic algorithm Alg with the following properties. On input $x \in \{0,1\}^n$, first Alg flips some poly(n) coins. Then it does some deterministic poly(n)-time computation and then either "accepts" or "rejects". Finally:

$$x \in L \Rightarrow \frac{1}{2} \leq \mathbf{Pr}[\text{Alg}(x) \text{ accepts }] < 1;$$
$$x \notin L \Rightarrow 0 < \mathbf{Pr}[\text{Alg}(x) \text{ accepts }] < \frac{1}{2}.$$

   (b) Show that PP $\subseteq$ RestartingBQP is a consequence of the following:

   Claim 1. *There is an efficient "restarting quantum query algorithm" for* Majority_N. *More precisely, suppose one is given a quantum oracle gate* O_f *for a Boolean function* $f : \{0,1\}^n \to \{0,1\}$, *with the promise that* $f \not\equiv 0, 1$. *Then there exists*[2] *a "restarting" quantum circuit using* poly(n) *total gates that correctly decides with probability at least* $3/4$ *whether* $|\{x : f(x) = 1\}|$ *is at least* 2^{n-1} *or less than* 2^{n-1}. *(Here a "restarting quantum circuit" is one as described in Problem 5(b), with the two output bits.)*

[2]Technically, we should also state here that the circuit not just exists but that it can be constructed by a poly(n)-time algorithm, but this will be obvious, and we allow you to ignore this minor point.

The remainder of the problem is devoted to showing the above Claim. However for simplicity, we will work at a slightly higher level, in the sense that we'll describe a kind of "restarting BQP computation" (you'll see what we mean). Also for simplicity, we will assume that we are allowed all one- and two-qubit gates, so in particular we can do controlled-Hadamards, and we can prepare the qubit $\alpha|0\rangle + \beta|1\rangle$ for any reals α, β with $\alpha^2 + \beta^2 = 1$.

(c) Let $N = 2^n$ and $s = |\{x : f(x) = 1\}|$. Explain how a restarting BQP computation can obtain a qubit in the state

$$c\left(\alpha s|0\rangle + \frac{\beta}{\sqrt{2}}(N-2s)|1\rangle\right) \tag{1}$$

for any reals α, β of its choice satisfying $\alpha^2 + \beta^2 = 1$. Here c is whatever normalizing constant is necessary to make (1) a quantum state. (Hint: First, do that thing that you always do in quantum query algorithms; then do a certain measurement and restart if it doesn't come out how you like. This should allow you to get a qubit in a state proportional to $(N-s)|0\rangle + s|1\rangle$. Next, attach another qubit, get a controlled-Hadamard into the picture, measure one of your qubits, and restart if it doesn't come to your liking.)

(d) Argue that if $0 < s < \frac{1}{2}N$ then there exist positive α, β such that (1) is equal to $|+\rangle$, whereas if $\frac{1}{2}N \leq s < 1$ then for all positive α, β the state (1) is "far" from $|+\rangle$. Further, argue that in the former case, if we consider all α, β with $\alpha/\beta = 2^i$ and $i \in \{-m, -m+1, \ldots, -1, 0, 1, \ldots, m-1, m\}$, $m = O(n)$, then for at least one such choice the state (1) is "close" to $|+\rangle$.

(e) Explain how we can use the power of restarting BQP to distinguish the two necessary cases, completing the proof. (Hint: somewhat similar to Problem 4(b).)