CSE 291 / Math 277 - Quantum Complexity Theory (Fall 2024)
Homework 4
Due Thursday, October 31, 3:00pm

Note: It is highly recommended (though not required) that you type your answers. It is your responsibility to make any handwriting clear and legible for grading. You may work with 1-2 other collaborators, but you must write the solutions separately and clearly mark the names of each person you worked with.

Problems:

1. **A decision version of Simon's problem**

   Recall the Simon's promise for a function $f\colon \{0,1\}^n \to \{0,1\}^n$: there exists a secret bitstring $s \in \{0,1\}^n$ such that $f(x) = f(y)$ iff $x = y \oplus s$. In class, we defined Simon's problem as a search problem—that is, the goal was to find the secret bitstring s. In that analysis, we were implicitly assuming that $s \neq 0^n$.

   (a) Show that there this is a polynomial-time quantum algorithm that always correctly outputs the secret bitstring (with high probability), including the case where s is the all-zeros string.

   Notice that whether or not $s = 0^n$ induces a dichotomy in the functions f satisfying the Simon's promise: if $s \neq 0^n$, then f is 2-to-1; if $s = 0^n$, then f is 1-to-1. A k-to-1 function f is such that every element in the image of f has exactly k inputs that map to it.

   Let's now define a (promise) decision problem based on this fact: given oracle access to a function f satisfying the Simon's promise, output "YES" if the function is 2-to-1 and output "NO" if the function is 1-to-1.

   Our goal for the next couple of problems will be to see how this language fits into NP, so let's start by recalling the definition of NP in this oracle setting: if a promise language $L = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ is in NP, there exists a poly-time Turing machine M and a polynomial q such that for all oracles $f\colon \{0,1\}^n \to \{0,1\}^n \in \Pi_{\text{yes}} \cup \Pi_{\text{no}}$

   - If $f \in \Pi_{\text{yes}}$, then $M^f(y) = 1$ for some $y \in \{0,1\}^{q(n)}$
   - If $f \in \Pi_{\text{no}}$, then $M^f(y) = 0$ for all $y \in \{0,1\}^{q(n)}$

   Here M^f just means that the poly-time Turing machine has oracle access to f (i.e., can query f on an input and get the function value in unit time).

   (b) Show that the decision Simon's problem is in NP.
   Hint: what is an efficiently veriable certificate showing that f is 2-to-1?

   Let's look at what happens if we were to flip the YES and NO instances of Simon's problem. That is, in the flipped Simon's problem, output "YES" if f is 1-to-1 and "NO" if f is 2-to-1. Using part (a), this version of the problem is no different than the

previous one for a quantum computer—just find s and decide accordingly. However, for a NP machine, this change makes a big difference.

(c) Show that the flipped Simon's problem is not in NP.
Hint: Is there a cerficate for f being 1-to-1?

While you're not being asked to show this, the flipped Simon's problem implies there is an oracle $\mathcal{O}$ relative to which $\mathsf{BQP}^{\mathcal{O}} \not\subseteq \mathsf{NP}^{\mathcal{O}}$. This gives evidence that BQP is not equal to NP.

2. **Parallel Grover search**

Grover's algorithm shows that the unstructed search problem can be solved using $O(\sqrt{2^n})$ quantum queries. In this question we ask if this algorithm can be parallelized. Specifically, let's consider a new generic outline for an arbitrary query algorithm that applies k oracles in parallel:

$$U_T O_f^{\otimes k} U_{T-1} \cdots U_1 O_f^{\otimes k} U_0 \left|0^{nk}\right\rangle$$

That is, the algorithm alternates between applying unitary gates $U_i \in \mathbb{C}^{2^{nk} \times 2^{nk}}$ and k oracle gates in parallel (i.e., $O_f^{\otimes k}$).

(a) Show that there is quantum algorithm with k parallel queries that solves the unstructured search problem in depth $T = O(\sqrt{2^n/k})$.
Hint: Use Grover's algorithm.

Suppose that we wanted to devise an algorithm where the quantum queries were "maximally parallel", that is, $T = 1$ and all oracles are applied in a single layer. The above algorithm suggests that we need to set $k = 2^n$. In other words, we've completely lost the Grover speedup! We'd like to get a parallelization scaling that goes as $1/k$, but instead we have an algorithm that scales as $1/\sqrt{k}$.

(b) Modify the BBBV lower bound to show that such scaling is unavoidable. That is, show that every quantum algorithm making k queries in parallel requires oracle depth $T = \Omega(\sqrt{2^n/k})$.
Hint: How do you define a new "query magnitude" so that it properly captures all the basis states that are queried during a parallel layer of oracle gates?