Quantum Computing Exercises # 7

Ronald de Wolf

Mar 15, 2011

(to be handed in before or at the start of the lecture on Mar 29)

1. Let P be the projector on a d-dimensional subspace $V \subseteq \mathbb{R}^n$ that is spanned by orthonormal vectors $v_1, \ldots, v_d$. This means that $Pv = v$ for all $v \in V$, and $Pw = 0$ for all w that are orthogonal to V.

   (a) Show that P can be written in Dirac notation as $P = \sum_{i=1}^{d} |v_i\rangle\langle v_i|$.

   (b) Show that $R = 2P - I$ is a reflection through the subspace corresponding to P, i.e., $Rv = v$ for all v in the subspace and $Rw = -w$ for all w that are orthogonal to the subspace.

2. Let A, B, and C be $n \times n$ matrices with real entries. We'd like to decide whether or not $AB = C$. Of course, you could multiply A and B and compare the result with C, but matrix multiplication is expensive (the current best algorithm takes time roughly $O(n^{2.38})$).

   (a) Give a classical randomized algorithm that verifies whether $AB = C$ (with success probability at least 2/3) using $O(n^2)$ steps, using the fact that matrix-vector multiplication can be done in $O(n^2)$ steps. *Hint: Choose a uniformly random vector $v \in \{0,1\}^n$, calculate ABv and Cv, and check whether these two vectors are the same.*

   (b) Show that if we have query-access to the entries of the matrices (i.e., oracles that map $i, j, 0 \mapsto i, j, A_{i,j}$ and similarly for B and C), then any classical algorithm with small error probability needs at least n^2 queries to detect a difference between AB and C. *Hint: Consider the case $A = I$.*

   (c) Give a quantum random walk algorithm that verifies whether $AB = C$ (with success probability at least 2/3) using $O(n^{5/3})$ queries to matrix-entries. *Hint: Modify the algorithm for collision-finding: use a random walk on the Johnson graph $J(n, r)$, where each vertex corresponds to a set $R \subseteq [n]$, and that vertex is marked if there are $i, j \in R$ such that $(AB)_{i,j} \neq C_{i,j}$.*

3. A 3-SAT instance ϕ over n Boolean variables $x_1, \ldots, x_n$ is a formula which is the AND of a number of clauses, each of which is an OR of 3 variables or their negations. For example, $\phi(x_1, \ldots, x_4) = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee x_3 \vee \overline{x_4})$ is a 3-SAT formula with 2 clauses. A satisfying assignment is a setting of the n variables such that $\phi(x_1, \ldots, x_n) = 1$ (i.e, TRUE). In general it's NP-hard to find a satisfying assignment to such a formula. Brute force would try out all 2^n possible truth-assignments, but something better can be done by a classical random walk. Consider the following simple algorithm, which is a random walk on the set of all $N = 2^n$ truth assignments:

Start with a uniformly random $x \in \{0,1\}^n$.
Repeat the following at most $3n$ times: if $\phi(x) = 1$ then STOP, else find the leftmost clause that is false, randomly choose one of its 3 variables and flip its value.

One can show that this algorithm has probability at least $(3/4)^n$ of finding a satisfying assignment (if ϕ is satisfiable). You may assume this without proof.

(a) Use the above to give a classical algorithm that finds a satisfying assigment with high probability in time $(4/3)^n \cdot p(n)$, where $p(n)$ is some polynomial factor (no need to use the C, U, S-framework of the lecture notes here; the answer is much simpler).

(b) Give a quantum algorithm that finds one (with high probability) in time $\sqrt{(4/3)^n} \cdot p(n)$. *Hint: view the $3n$-step random walk algorithm as a deterministic algorithm with an additional input $r \in \{0,1\}^n \times \{1,2,3\}^{3n}$, where the first n bits determine x, and the last $3n$ entries determine which variable of the leftmost false clauses will be flipped in the $3n$ steps of the random walk. Use Grover search on the space of all such r (no need to write out complete circuits here).*