Introduction to Quantum Information Science
Homework 8

Due Wednesday, November 6 at 11:59 PM

1. Generalized Deustch-Jozsa Problem Recall that in the Deustch-Jozsa problem we are given oracle access to an unknown function $f\colon \{0,1\} \to \{0,1\}$ and wish to determine if it is constant or balanced. In the generalized Deustch-Jozsa problem we are given oracle access to a function $f\colon \{0,1\}^n \to \{0,1\}$, a function that maps n-bit strings to a single bit. We're promised that the function is either constant or balanced, where balanced means that it has an equal number of 0 and 1 outputs (2^{n-1} of each).

a) [6 Points] Design a quantum algorithm that can determine whether f is constant or balanced using only a single query to f and prove that your algorithm works.

b) [Extra credit, 3 Points] Show that any classical deterministic algorithm for this problem requires $\Omega(2^n)$ queries. Therefore, quantum computers give an exponential speedup over deterministic classical computers for this problem.

c) [3 Points] Explain why, nevertheless, this speedup is not very impressive, in the sense that it is not an exponential quantum speedup over all classical computing for this problem. Give a full explanation and analysis of the any runtime or likelihood of success involved in your explanation.

2. The Birthday Paradox Your favorite local radio station is running a new give-away contest that works as follows: Each day at 5pm the phone lines open up to the public to call in and leave their name and birth-date which is then added to a running list. The contest runs until a matching birth-date (month and date) between any two contestants on the list is found. At that point the contest closes and everyone on the list up to that point is a winner. Assume that every birthday is equally likely and that no contestants are born during a leap year (so that we can ignore Feb. 29th).

As usual, you must show or explain your work for each part. You are free to use numerical software of your choice to help solve the problem

a) [2 Points] What is the minimum number of people who need to call in before the probability of a match being found is at least 50%?

b) [1 Point] How about the minimum number of people needed before there is a 99% chance of the contest coming to a close?

c) [4 Points] Imagine after running the contest for a few days the station decides they aren't being generous enough and so change the rules as follows: Instead of the contest ending as soon as there's a match found between any two contestants on the list it now ends when a match is found between specifically the first caller of the day and any other contestant.

Under these new rules what is the minimum number of people needed before there is a 50% chance of the contest closing? How about for a 99% chance?

3. Two Secrets [5 Points] Suppose the function $f : \{0,1\}^n \to \{0,1\}^n$ is such that there are *two* n-bit secret strings s and t, where that $s \neq t$ and neither are the all zeros string, so that $f(x) = f(y)$ if and only if $x \oplus y$ is either 0, s, t, or $s \oplus t$.
Give a quantum algorithm which can find an $a \in \{0,1\}^n$ such that $a \cdot s = a \cdot t = 0$, and $a \neq 0$. Explain/prove that it works.

4. Bernstein-Vazirani In the Bernstein-Vazirani problem, recall that we're given oracle access to a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. We're promised that there exists a "secret string" $s \in \{0,1\}^n$ such that $f(x) = s \cdot x \bmod 2$ for all x. The problem is to recover s. The Bernstein-Vazirani algorithm solves this problem with just a single quantum query to f. Consider a variant of this problem where we are promised that $f(x) = s \cdot x \bmod 2$ for <u>at most</u> a $(1 - \epsilon)$ fraction of the inputs x, and that $f(x) = (s \cdot x + 1) \bmod 2$ for the remaining ϵ fraction of inputs.

a) [4 Points] Calculate a upper bound on the probability that a single run of the Bernstein-Vazirani algorithm nevertheless succeeds in recovering s. Show your work.

b) [3 Points] Explain what happens when $\epsilon = 1/2$. Is there a reason why, in some sense, no algorithm can possibly succeed at recovering s in that case?