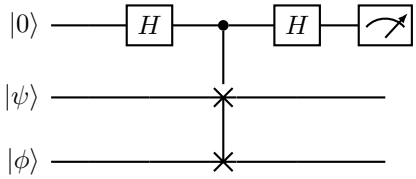
Quantum computing: algorithms and complexity

Assignment 2

Due: 22.04.2023, 23:59

Problem 1 (The quantum SWAP test)

Suppose we have two n-qubit quantum states $|\psi\rangle$ and $|\phi\rangle$ and would like to test whether they are close to each other or far apart. By this we mean that we would like to know whether $|\langle\psi|\phi\rangle|^2$ is close to 1 or close to 0. One way to do this is with the *quantum SWAP test* given by the following circuit:



The operation in the middle is a controlled-SWAP, or $CSWAP$, operation. It swaps the bottom two registers whenever the control qubit is in the $|1\rangle$ state. Formally:

$$|0\rangle|\psi\rangle|\phi\rangle \xrightarrow{CSWAP} |0\rangle|\psi\rangle|\phi\rangle \qquad\qquad |1\rangle|\psi\rangle|\phi\rangle \xrightarrow{CSWAP} |1\rangle|\phi\rangle|\psi\rangle$$

A. For the case when $|\psi\rangle$ and $|\phi\rangle$ are single-qubit states, the $CSWAP$ operation is a 3-qubit operation. Give a circuit that performs $CSWAP$ using only $CCNOT$ (or Toffoli) gates. **Hint:** Recall from Assignment 1 how $SWAP$ can be performed in terms of $CNOT$ gates.

B. Compute the probability of the top qubit being measured as $|0\rangle$ in the quantum SWAP test circuit given above. You should express this probability in terms of $|\langle\psi|\phi\rangle|^2$. What is the probability when $|\psi\rangle = |\phi\rangle$? What about when the states are orthogonal?

C. Write what the post-measurement state will be for the bottom two registers, when the top qubit is measured as $|0\rangle$ and as $|1\rangle$, respectively.

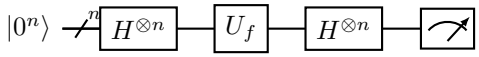
Problem 2 (Deutsch-Jozsa)

In the lectures, we've seen a number of problems where quantum computers offer an advantage in terms of query complexity over classical computers. Here we'll consider another such problem, though, as you'll show, it only provides an advantage against *deterministic* classical algorithms.

Suppose we are given access to a function $f : \{0,1\}^n \to \{0,1\}$, which is promised to be either *constant* or *balanced*. Constant means that the function is either $f(x) = 0$, for all $x \in \{0,1\}^n$, or $f(x) = 1$, for all $x \in \{0,1\}^n$. Balanced means that the function is 0 on half of the inputs and 1 on the other half. In other words, there are sets $S_0, S_1 \subset \{0,1\}^n$ such that $|S_0| = |S_1| = 2^{n-1}$, $S_0 \cap S_1 = \emptyset$ and for any $x \in S_0$, $f(x) = 0$ and for any $x \in S_1$, $f(x) = 1$.

A. Argue that any deterministic classical algorithm requires $2^{n-1} + 1$ queries to f in order to determine whether it is constant or balanced. In other words, $D(\mathrm{DJ}) = 2^{n-1} + 1$.

B. Consider the following quantum circuit:



Here, U_f is the *phase oracle* for f, so that $U_f |x\rangle = (-1)^{f(x)} |x\rangle$. What is the probability that the output of the circuit is measured as $|0^n\rangle$ when the function is constant? What about when the function is balanced? Justify your answers. This shows that $Q(\mathrm{DJ}) = 1$.

C. Give an efficient *randomized* classical algorithm that correctly decides whether the function is constant or balanced, with probability at least 99%. This shows that $R(\mathrm{DJ}) = O(1)$.

Problem 3 (Recursive Fourier Sampling)

In the lectures, we talked about how the Bernstein-Vazirani (BV) problem gives a separation in query complexity between quantum and classical algorithms with $Q(\mathrm{BV}) = 1$, $R(\mathrm{BV}) = n$. In this problem, we will consider the generalization of BV, known as Recursive Fourier Sampling (RFS).

First, we will consider a *decision version* of BV. Recall that in BV you are given oracle access to a function $f(x) = x \cdot s$, for some secret $s \in \{0,1\}^n$, and the goal is to find s. In the decision version you are asked to find a *hardcore bit* of s. A hardcore bit (or hardcore predicate) is some function, denoted h, such that $s \mapsto h(s)$ is as hard to compute as s itself. An example is the following:

$$h(s) = \begin{cases} 0, & \text{if } |s| \bmod 3 = 0 \\ 1, & \text{otherwise} \end{cases}$$

where $|s|$ denotes the Hamming weight of s. The decision version of BV is: given oracle access to f, output $h(s)$. Clearly, this can be solved by first finding s and then applying h on it. We refer to this as RFS_1, the 1-level version of RFS, which is just BV.

Next, let's consider a 2-level version, RFS_2. Suppose we're given access to $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, such that:

$$f(x, y) = s_x \cdot y,$$

where the strings $s_x \in \{0,1\}^n$ are chosen so that $h(s_x) = s \cdot x$, for some $s \in \{0,1\}^n$. The task is again to output $h(s)$ (which requires finding s itself).

Why is this a 2-level version of BV (or RFS_1)? If we fix x, $f(x, y) = s_x \cdot y$ is a BV function with secret s_x. Suppose we solve this instance of BV and recover s_x. If we then compute $h(s_x)$ we obtain the bit $s \cdot x$, for the x fixed in the beginning. Taking x to be successively $00..01, 00..10, ..., 01..00, 10..00$, we would then be able to recover the bits of s. But note that for each value of x, we have to solve an instance of BV. As $R(\text{BV}) = n$, we find that $R(\text{RFS}_2) = n^2$.

A. Write a quantum circuit that solves RFS_2 using 2 queries to the phase oracle $U_f |x\rangle |y\rangle = (-1)^{s_x \cdot y} |x\rangle |y\rangle$. Note that you can also assume that you have a unitary[1] for computing h, i.e. $U_h |s\rangle |z\rangle = |s\rangle |z \oplus h(s)\rangle$.

B. We will now consider RFS_k to be the k-level version of BV. In this case, we are given oracle access to $f : \{0,1\}^{n \times k} \to \{0,1\}$,

$$f(x_1, x_2, ... x_k) = s_{x_1, ... x_{k-1}} \cdot x_k,$$

where $s_{x_1, ... x_{k-1}} \in \{0,1\}^n$ are chosen such that

$$h(s_{x_1, ... x_{k-1}}) = s_{x_1, ... x_{k-2}} \cdot x_{k-1}.$$

Similarly, $s_{x_1, ... x_{k-2}} \in \{0,1\}^n$ are chosen such that

$$h(s_{x_1, ... x_{k-2}}) = s_{x_1, ... x_{k-3}} \cdot x_{k-2},$$

and so on, until $s_{x_1} \in \{0,1\}^n$ are chosen such that

$$h(x_1) = s \cdot x_1.$$

for some secret $s \in \{0,1\}^n$. The task is to output $h(s)$.

Give a classical algorithm for solving this problem, making n^k queries to the oracle. While you're not required to prove this, it turns out that this algorithm is optimal in the classical case (for both deterministic and randomized strategies).

C. Give a quantum algorithm for solving RFS_k, that makes 2^{k-1} queries to the phase oracle:

$$U_f |x_1, x_2, ... x_k\rangle = (-1)^{s_{x_1, ... x_{k-1}} \cdot x_k} |x_1, x_2, ... x_k\rangle .$$

Taking $k = \log n$ will then yield a suprapolynomial separation in query complexity between quantum and classical algorithms, as $R(\text{RFS}_{\log}) = \Omega(n^{\log n})$ while $Q(\text{RFS}_{\log}) = O(n)$.

[1] Also note that this is not an oracle since, unlike f, h is given to us explicitly, so we can implement it as a quantum circuit.

Problem 4 (Forrelation)

So far, the biggest query complexity separation we have seen between quantum and classical was given by Simon's problem (or any general period finding problem) where we found that $Q(\text{Simon}) = O(n)$, whereas $R(\text{Simon}) = 2^{\Omega(n)}$. Can we do better? In particular, can we define a problem which has $O(1)$ quantum query complexity but $2^{\Omega(n)}$ randomized query complexity? The answer is yes and comes in the form of a problem called *Forrelation*.

We are given oracle access to two functions $f, g : \{0,1\}^n \to \{0,1\}$. We are promised that either f is strongly correlated with the $\mathbb{Z}_2^n$ Fourier transform of g, or f has almost no correlation with the Fourier transform of g. The task is to decide which of these two is the case.

To be more precise, the $\mathbb{Z}_2^n$ Fourier transform of g is defined as

$$\hat{g}(x) = \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} (-1)^{g(y)}$$

Note that $\hat{g} : \{0,1\}^n \to \mathbb{R}$. What does it mean to say that f is correlated with $\hat{g}$? Informally, it simply means that $f(x) = 1$ for inputs x for which $\hat{g}(x)$ is large and $f(x) = 0$ whenever $\hat{g}(x)$ is small. More formally, we define the quantity

$$\Phi(f, g) = \frac{1}{2^{3n/2}} \sum_{x,y \in \{0,1\}^n} (-1)^{f(x)} (-1)^{x \cdot y} (-1)^{g(y)}$$

called the *forrelator*. We say that f and g are forrelated if, say, $|\Phi| \geq 0.5$ and not forrelated if $|\Phi| \leq 0.01$ and we are promised that the functions are chosen such that one of these two is the case.[2]

It is known that $R(\text{Forrelation}) = 2^{\Omega(n)}$, so that any classical algorithm attemping to solve this problem would require exponentially many queries to the oracles for f and g. Your task is to show that there is a quantum algorithm which can solve Forrelation with just $O(1)$ queries.

A. To get a feel for the problem, consider the case $n = 2$. Take g to be the constant 0 function, so $g(x) = 0$, for all x. Compute $\hat{g}(x)$ for all x. Find a function f that is maximally forrelated with g and one that is minimally forrelated. In both cases, compute $\Phi(f, g)$. Do the same for the case where g is a non-constant (and also non-balanced) function.

B. Before tackling Forrelation, let's solve a related problem called *Fourier Fishing*. Give a quantum circuit acting on n qubits, that makes exactly one query to the phase oracle for g (and no queries to f) and for which the output of the circuit is measured in the computational basis. It should be that the probability of measuring output $|x\rangle$ is exactly $|\hat{g}(x)|^2$. In other words, the circuit is sampling strings x with probabilities given by the squared Fourier spectrum of g. Incidentally, this can also be shown to be classically hard (requiring exponentially many queries to g). Justify why your circuit achieves the intended output probabilities.

C. We are now ready to solve Forrelation. Give a quantum circuit acting on n qubits, whose output is measured in the computational basis and for which the probability of seeing the outcome $|0^n\rangle$ is $|\Phi(f, g)|^2$.

[2] The constants here are somewhat arbitrary. The point is that in one case the forrelator significantly larger than in the other case.

The circuit should make $O(1)$ queries to the phase oracles for f and g. Justify why your circuit achieves the intended output probability for $|0^n\rangle$.

D. Now suppose you are given access to an oracle that performs the mapping

$$U|b\rangle|x\rangle = [(1-b)(-1)^{f(x)} + b(-1)^{g(x)}]|b\rangle|x\rangle$$

In other words, if $|b\rangle = |0\rangle$, it applies a phase of $(-1)^{f(x)}$ to the state and if $|b\rangle = |1\rangle$ it applies a phase of $(-1)^{g(x)}$ to the state. Give a quantum circuit that makes exactly one query to U and for which the probability of obtaining a zero output (say, even by meausuring a single qubit) is exactly $\frac{1}{2}(1 + \Phi(f, g))$. Justify your answer.

Hint: The quantum SWAP test. Note also that you are allowed to perform controlled versions of unitaries we've seen so far.

Problem 5 (The Quantum Fourier Transform)

We've seen that a crucial component in Shor's algorithm is the quantum Fourier transform (QFT). The QFT acting on n qubits is defined as the unitary:

$$QFT_n|x\rangle = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \omega_n^{xy} |y\rangle \tag{1}$$

where $\omega_n = \exp\left(\frac{2\pi i}{2^n}\right)$ is a root of unity of order 2^n. The notation is slightly different from what we used in the lectures—technically the QFT can be defined on any M-dimensional Hilbert space, however we are only interested in 2^n-dimensional spaces, so everything is indexed with n. Note that we are treating x and y as integers, rather than n-bit strings (the states represent binary encodings of those integers).

We're going to see how the QFT can be implemented from elementary gates. To do whis, we will make use of the unitary:

$$R_n = \begin{bmatrix} 1 & 0 \\ 0 & \omega_n \end{bmatrix}$$

This is simply a rotation around the Z axis (of the Bloch sphere) by an angle of $\frac{2\pi i}{2^n}$. Note that for $n = 3$, this becomes the familiar T gate. From the Solovay-Kitaev theorem, we know that R_n can be approximated to within arbitrary precision using (at most $poly(n)$) H and T gates. We will thereofre assume that the R_n gate is a fundamental gate that we can use, in addition to its controlled version CR_n, which performs the mapping:

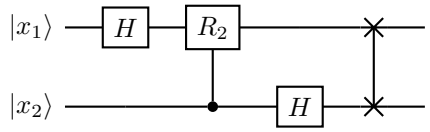
$$CR_n|x\rangle|y\rangle = \omega_n^{xy}|x\rangle|y\rangle$$

where here $x, y \in \{0, 1\}$.

A. We'll first consider the 2-qubit QFT_2:

$$QFT_2 |x\rangle = \frac{1}{2}\sum_{y=0}^{3} \exp\left(\frac{2\pi i\, xy}{4}\right)|y\rangle = \frac{1}{2}(|00\rangle + i^x |01\rangle + (-1)^x |10\rangle + (-i)^x |11\rangle)$$

Show that the following circuit implements QFT_2:



where $|x\rangle = |x_1, x_2\rangle$ (x_1 and x_2 are the bits of x).

B. Notice that the output of the previous circuit is a product state:

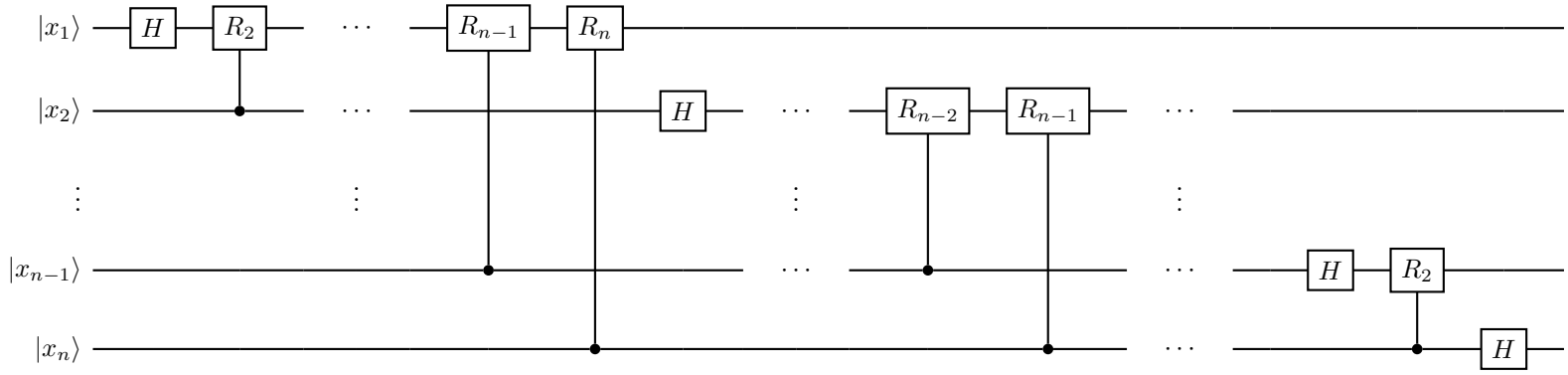
$$QFT_2 |x_1, x_2\rangle = \frac{1}{2}(|0\rangle + (-1)^{x_2} |1\rangle) \otimes (|0\rangle + (-1)^{x_1}\, i^{x_2} |1\rangle)$$

Using the notation $[0.x_1x_2...x_n] = \sum_{k=1}^{n} x_k 2^{-k}$ and the definition of QFT_n from Equation 1, show that:

$$QFT_n |x_1, x_2, ...x_n\rangle = \frac{1}{N}(|0\rangle + e^{2\pi i[0.x_n]} |1\rangle) \otimes (|0\rangle + e^{2\pi i[0.x_{n-1}x_n]} |1\rangle) \otimes ... \otimes (|0\rangle + e^{2\pi i[0.x_1x_2...x_n]} |1\rangle) \quad (2)$$

Hint: In Equation 1, when writing ω_n^{xy}, we are treating x and y as integers. As an integer, x expressed in terms of its bits is simply $\sum_{k=1}^{n} x_k 2^{n-k}$.

C. Using Equation 2, show that QFT_n can be implemented by the following circuit:



Note that the order of the qubits is reversed. So, strictly speaking, for QFT_n, we would also perform $n/2$ $SWAP$ gates at the end of the circuit to reverse the order of the qubits.