This sheet contains exercises about

- Hadamard transform
- Grover
- Deutsch
- Deutsch-Jozsa
- Shor

1. Excercise 1 (1 points)
   Using the matrix representation of the Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

   write down the matrix $H \otimes H$ and find $(H \otimes H)(|0\rangle \otimes |1\rangle)$. Show that this is equivalent to

$$|\phi\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

2. Exercise (1 points)
   The Beam splitter gate has a matrix representation given by

$$B = \frac{1}{\sqrt{2}} \begin{pmatrix} i & 1 \\ 1 & i \end{pmatrix}$$

   Show that B generates superposition states out of the computational basis states $|0\rangle$ and $|1\rangle$. In particular, show that

$$B \otimes B|0\rangle|0\rangle = \left(\frac{i|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{i|0\rangle + |1\rangle}{\sqrt{2}} \right)$$

   Show that two applications of the beam splitter gate on the same state, namely that $B(B|\psi\rangle)$ act analogously to the NOT gate, giving the same probabilities of finding $|0\rangle$ and $|1\rangle$.

3. Exercise (1 points)
   Deutsch-Jozsa's algorithm

   Consider a function with two inputs such that $f(x) = 1$. Explicitly show that the Deutsch-Jozsa algorithm works in this case by generating the vector $|y\rangle = |00\rangle$ as the final output.

4. Exercise (1 points)
   Quantum gates are universal in the sense that quantum gates can be designed that do

   anything a classical gate can do. Design a quantum adder, a gate that takes three inputs $|x\rangle, |y\rangle, |z\rangle$ and that has three output qubits $|x\rangle, |x \oplus y\rangle, |xy\rangle$, where $|x \oplus y\rangle$, is the sum and $|xy\rangle$ is the carry.

5. Exercise (1 points)
   Shor's algorithm

   Consider the eigenvectors

$$|u_t\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i k t}{r}\right) \left| x^k \bmod N \right\rangle$$

   and show that

$$U_x |u_t\rangle = \exp\left(-\frac{2\pi i t}{r}\right) |u_t\rangle$$

   where

$$U_x|y\rangle = \begin{cases} |xy \bmod N\rangle & 0 \le y \le N-1 \\ |y\rangle & N \le y \le 2^L - 1 \end{cases}$$

6. Exercise (1 points)
   Shor's algorithm.

   Show that the probability that the period of $f(r) = x^r \bmod N$ is odd is at most 1/2.

7. Exercise (1 points)
   Grover's algorithm

   Show that the unitary operator corresponding to the phase shift in the Grover iteration is $2|0\rangle\langle 0| - I$.

8. Exercise (1 points)
   Hadamard

   Demonstrate the Hadamard operator on one qubit may be written as

$$H = \frac{1}{\sqrt{2}}[(|0\rangle + |1\rangle)\langle 0| + (|0\rangle - |1\rangle)\langle 1|].$$

9. Exercise (1 points)
   Hadamard

Write out an explicit matrix representation for $H^{\otimes 3}$.

10. Exercise (1 points)
 QFT and H

 Prove if

$$\text{QFT}|0\rangle^{\otimes n} = H^{\otimes n}|0\rangle^{\otimes n}$$

 is true.

1 Deutsh-Jozsa algorithm

The Deutsch-Jozsa algorithm is a generalization of Deutsch's algorithm. Again, this algorithm allows us to determine whether a function $f(x)$ is constant or balanced[1], but this time the function has multiple input values. If $f(x)$ is constant, then the output is the same for all input values x. If the function is balanced, then $f(x) = 0$ for half of the inputs and $f(x) = 1$ for the other half of the inputs, and vice versa. We start with an initial state that includes n qubits in the state $|0\rangle$ and a singlequbit in the state $|1\rangle$. Hadamard gates are applied to all qubits. The circuit is illustrated in Figure 1.

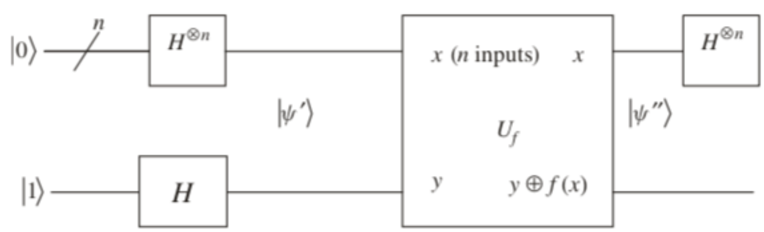

Figure 1: The Deutsch-Jozsa algorithm generizes Deutsch's algorithm to handle a function with n input values and determine whether or not it is constant or balanced.

We start off by calculating

$$\left|\psi'\right\rangle = \left(H^{\otimes n}\right)\left(|0\rangle^{\otimes n}\right) \otimes (H|1\rangle).$$

From

$$H^{\otimes n}\left(|0\rangle^{\otimes n}\right) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

[1]The identity and bit flip functions are called balanced because the outputs are opposite for half the inputs. So a function on a single bit can be constant or balanced. Whether a function on a single bit is constant or balanced is a global property. What we're going to see in the following development is that Deutsch's algorithm will let us put together a state that has all of the output values of the function associated with each input value in a superposition state. Then we will use quantum interference to find out if the given function is constant or balanced.

we see that this is

$$\left|\psi'\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Next we apply $U_f\,|x,y\rangle = |x, y \oplus f(x)\rangle$, to evaluate the function. The first n qubits are the values of x and the last qubit plays the role of y as shown in the figure. The output state of the U_f gate is

$$\left|\psi''\right\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

Applying a Hadamard gate to an n qubit state $|x\rangle$ gives

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_y (-1)^{x\cdot y} |y\rangle$$

So the final output state is

$$|\psi_{\text{out}}\,\rangle = \frac{1}{2^n} \sum_y \sum_x (-1)^{x\cdot y + f(x)} |y\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Now we measure the n inputs. It might not be immediately obvious looking at $|\psi_{\text{out}}\,\rangle$, but there are two possible measurement results on $|y\rangle$ (which is the state of n inputs at this stage) that are of interest. The possible results are as follows: Measurement of the first n input qubits in $|\psi_{\text{out}}\,\rangle$ returns all 0's. In this case $f(x)$ is constant. Otherwise, if at least one of the qubits in $|y\rangle$ is found to be a 1 on measurement, $f(x)$ is balanced.

2 Shor's algorithm

Shor's algorithm was fundamental in demonstrating the power and importance of quantum computation. This is an algorithm that can be used to factor prime numbers - meaning that it can be used to break encryption codes if a practical quantum computer is ever built. Needless to say, this algorithm got the attention of a lot of people.

The first thing we need to know in order to do Shor's algorithm is order finding . Let x and N be positive integers with no common factors such that $x < N$. The order of x is the smallest positive integer r such that

$$x^r = 1 \bmod N$$

(9.50)

Example 1 *what* mod$N$ *means? First of all, x and N can't have any common factors because their greatest common divisor is 1 .*

Suppose that we let $x = 5$ and $N = 44$. To find $x^r = a \bmod N$, we compute x^r and subtract N until we get the last integer greater than 0*. The first two cases are less than $N = 44$, so we don't do anything:* $5^1 = 5$ *and* $5^2 = 25$*. Now since* $5^3 = 125$*, we note that* $(44)(2) = 88$ *and* $125 - 88 = 37$*.*

Hence $5^3 = 37(\text{mod} 44)$. *Next* $5^4 = 625$. *We have* $(14)(44) = 616$, *and so* $5^4 = 9(\text{mod} 44)$. *Finally* $5^5 = 3125$. *It turns out that* $71 \times 44 = 3124$, *which is 1 less than* $5^5 = 3125$. *This is where we stop. Hence*

$$5^5 = 1(\text{mod} 44)$$

The order of 5 is 5 in this case. As you can see, plugging away like this, finding the powers $x^r = 1(\text{mod} N)$ *can be very time-consuming. With large numbers it will swamp the best computers available, the time required is exponential in* $\log N$. *This problem can be solved far more efficiently by using a quantum algorithm based on phase estimation.*