CSE 190 / Math 152 – Introduction to Quantum Computing (Fall 2026)
Homework 4
Due Thursday, November 5, 11:59pm

Instructions: There are two parts to this homework:

- Concept check (Problem 1): Complete individually on Gradescope.
- Written Homework (Problems 2–3): You may work individually or in a team of up to 3 people. If working in a group, make only one submission per group on Gradescope: one person uploads the submission and then adds the other group members to the submission by selecting their names in the "Add Group Members" dialog box. All problems will be graded for completeness. Completeness means that it looks like there was a good-faith effort to solve the problem. A LaTeX template for the homework is provided on Canvas.

Problems:

1. **Concept check**

   Complete the assignment "Homework 4 - Concept Check" on Gradescope.

   Must be completed individually even if working in a team for the rest of the assignment.

2. **Constructing states with partial measurement**

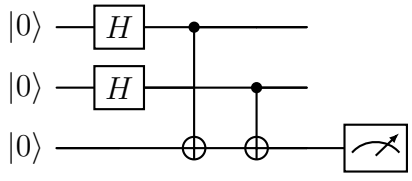
   In the last homework assignment, you built circuits for various states by giving an explicit sequence of unitary gates to construct them. Sometimes, however, it will be helpful to use partial measurements as another tool. In this problem, we will explore the use of partial measurements to construct a special class of states. First, for a bit string $x \in \{0,1\}^n$, we define the *Hamming weight* $|x|$ to be the number of 1's in x. For integers k, n such that $0 \leq k \leq n$, we define the state $|D_k^n\rangle$ to be the uniform superposition of all n-qubit classical basis states that have Hamming weight k:

$$|D_k^n\rangle = \frac{1}{\sqrt{\binom{n}{k}}} \sum_{\substack{x \in \{0,1\}^n \\ \text{s.t. } |x|=k}} |x\rangle$$

   For example, we have the states

$$|D_1^2\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \qquad \text{and} \qquad |D_2^3\rangle = \frac{|011\rangle + |101\rangle + |110\rangle}{\sqrt{3}}$$

   which are superpositions of states with Hamming weight 1 and 2, respectively. While there is a straightforward circuit of unitary gates to construct $|D_1^2\rangle$, let's still look at how we might use partial measurements to construct it as a toy example:



Suppose the measurement on the third qubit has outcome $b \in \{0, 1\}$.

(a) Show that the first two qubits are in the state $|D_1^2\rangle$ if and only if $b = 1$.

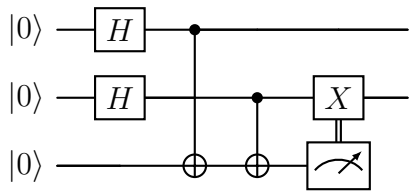
What you just showed above is that state construction based on partial measurement can have some probability of failure. This can be an acceptable trade-off for many quantum applications—if you fail to create the state you wanted to, just start over and try again.

(b) Using the same idea for the construction of the $|D_1^2\rangle$ state, construct the $|D_2^3\rangle$ state with partial measurement. Your circuit should have 5 total qubits—the top three for the state and the bottom two for the measurement. Furthermore, you can only use Hadamard, CNOT, and Toffoli gates.

(c) With what probability does your construction succeed? That is, with what probability are the first three qubits in the state $|D_2^3\rangle$ after measurement?

Recall that you also constructed $|D_2^3\rangle$ in the previous homework using a sequence of unitary gates. Notice in this case that we were able to construct the same state with a simpler set of gates. This comes at the cost of having some probability of failure.

Sometimes, we can boost our probability of success by post-processing the quantum state based on the measurement result. Consider the circuit:



Here, the vertical double wire indicates that the application of the X gate depends on the outcome of the measurement. Once again, let $b \in \{0, 1\}$ be the outcome of the measurement on the third qubit.

(d) Suppose we apply the X gate only when $b = 0$. Show that the above circuit always constructs the state $|D_1^2\rangle$. In other words, by applying an extra gate depending on the outcome of the partial measurement, we can boost our probability of success all the way to 1.

Challenge question (not graded): For all values of n and k, describe a quantum circuit that constructs the state $|D_k^n\rangle$. What is the probability of success of your construction? If you generalize the approach introduced in this problem, your circuit will probably succeed with very small probability, which is okay. However, there exist constructions of this state that succeed with high probability.

3. **Quantum circuits are just as good as classical circuits**

In this class, our goal has always been to show there's some problem that a quantum computer can solve efficiently that looks out of reach for classical computers. But, is it

possible that the reverse is also true? That is, is there a problem that a classical computer could solve efficiently that cannot be solved efficiently with a quantum computer? In this problem, we will show that the answer is "no".

First, let's explore why the answer might be "yes". Namely, a classical circuit has access to gates that do not need to be unitary. A classical gate $g\colon \{0,1\}^c \to \{0,1\}$ is just a Boolean function which maps c bits to 1 bit. For example, the classical AND gate is the 2-input gate such that the output bit is 1 if and only if both of the input bits are also 1:

$$\text{AND}(0,0) = 0; \quad \text{AND}(0,1) = 0; \quad \text{AND}(1,0) = 0; \quad \text{AND}(1,1) = 1$$

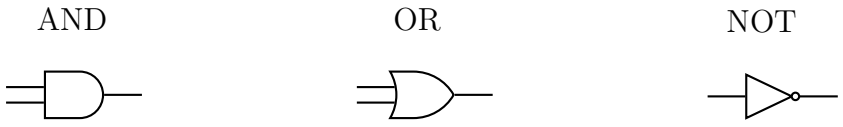
The OR gate is 1 when either of the input bits is 1:

$$\text{OR}(0,0) = 0; \quad \text{OR}(0,1) = 1; \quad \text{OR}(1,0) = 1; \quad \text{OR}(1,1) = 1$$

Another important gate is the NOT gate, which simply flips the value of the input bit:

$$\text{NOT}(0) = 1; \quad \text{NOT}(1) = 0$$

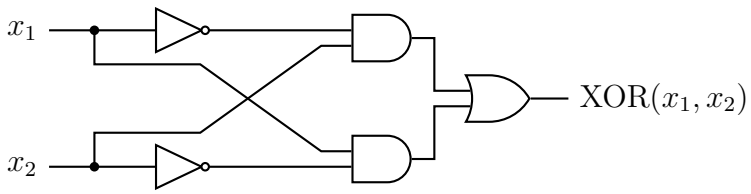
We can build classical circuits for other Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ out of classical gates by creating a network of gates, where the output of a gate can be fed as input to another. Much like with quantum circuits, it can be easiest to describe a classical circuit through some sort of visualization. Pictorially, we can draw the AND, OR, and NOT gates as follows:



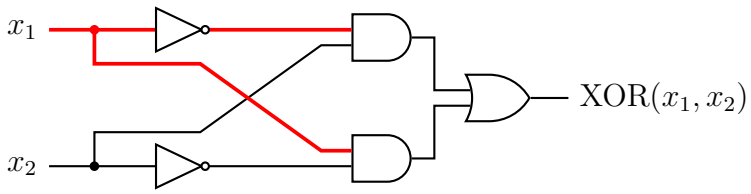
Each gate has the inputs coming in from the left and the outputs leaving from the right. To see an example of a classical circuit, let's consider the XOR function, which sums the input bits mod 2:

$$\text{XOR}(0,0) = 0; \quad \text{XOR}(0,1) = 1; \quad \text{XOR}(1,0) = 1; \quad \text{XOR}(1,1) = 0$$

A classical circuit for XOR is shown below:



Notice that one key feature of the circuit above (and classical circuits in general) is that wires can be "split" or "copied" so that the value of one wire in the circuit can be fed into multiple gates. In the circuit below, we highlight in red one of these copy operations. The input x_1 is used as input to both the NOT gate on the top wire as well as the first input to the bottom AND gate.



Although it's outside the scope of this course, it will turn out that classical circuits completely capture the power of classical computation in the same way that quantum circuits completely capture the power of quantum computation. One aspect of this characterization is the following important theorem.

Theorem 1. *Every Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$ *can be computed by a classical circuit consisting entirely of* NOT *gates and 2-input* AND *gates.*

For those of you who haven't seen this result before, there is an optional question at the end of this assignment that will walk you through the proof. For now, let us simply take this theorem as a fact. Using Theorem 1 as a starting point, we can prove that any function a classical circuit can implement, a quantum circuit can also implement via the standard oracle:

Theorem 2. *For every Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$*, there is a quantum circuit which implements the standard oracle* B_f*:*

$$B_f(|x\rangle \otimes |b\rangle) = |x\rangle \otimes |f(x) \oplus b\rangle$$

for all $x \in \{0,1\}^n$ *and* $b \in \{0,1\}$*. Moreover, this construction is* efficient*—if there is a classical circuit which computes* f *with few gates, then there is also a quantum circuit which computes* B_f *with few gates.*

This problem will walk you through the steps of the proof.

(a) Let's first tackle the issue that classical circuits can "split" wires, whereas quantum circuits cannot. Here, we will use the fact that we are only dealing with classical bit strings, so we can copy the classical bit value into another ancilla register if we want to use it another time. Show there is a quantum circuit which implements the following "copy" operation:

$$\text{COPY}\,|x\rangle\,|0\rangle = |x\rangle\,|x\rangle$$

for all $x \in \{0,1\}$. Which basic quantum gate implements this operation?

(b) Show there is a quantum circuit for the NOT operation:

$$\text{NOT}\,|x\rangle = |x \oplus 1\rangle$$

for all $x \in \{0,1\}$. Which basic quantum gate implements this operation?

(c) Show there is a quantum circuit for the "Quantum AND" operation:

$$\mathrm{QAND}(|x\rangle \otimes |y\rangle \otimes |0\rangle) = |x\rangle \otimes |y\rangle \otimes |\mathrm{AND}(x, y)\rangle$$

for all $x, y \in \{0, 1\}$. Which basic quantum gate implements this operation?

(d) Suppose there is a classical circuit with 2-input AND gates and NOT gates which computes some Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. Combining all of the above, show there is a quantum circuit J_f which implements the standard oracle with an extra junk register:

$$J_f(|x\rangle \otimes |b\rangle \otimes |0 \cdots 0\rangle) = |x\rangle \otimes |f(x) \oplus b\rangle \otimes |\mathrm{junk}(x)\rangle$$

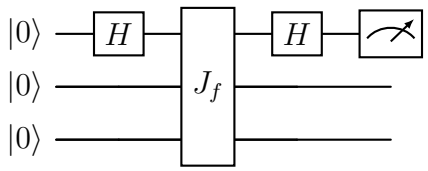
for all $x \in \{0,1\}^n$ and $b \in \{0,1\}$. Here, $\mathrm{junk}(x)$ is some classical bit string which depends on x.

It may at first seem like the J_f unitary is just as good as the B_f unitary. Unfortunately, as we will now see, this junk can be quite annoying if you want to implement f in superposition:

$$J_f\left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle\, |b\rangle\, |0 \cdots 0\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle\, |f(x) \oplus b\rangle\, |\mathrm{junk}(x)\rangle$$

The junk can act like an unwanted measurement of x. To get some intuition for how the junk can destroy the careful superposition of our state, we will look at a small example.

(e) Let $f\colon \{0,1\} \to \{0,1\}$ be the constant-0 function (i.e., $f(0) = f(1) = 0$). Consider the following circuit:



Show that the probability distribution of the partial measurement is different in the following two cases:

- $\mathrm{junk}(0) = 0$ and $\mathrm{junk}(1) = 1$; and
- $\mathrm{junk}(0) = 0$ and $\mathrm{junk}(1) = 0$.

To get around this issue we must "uncompute" the junk state, mapping it back to the all-zeros state.

(f) Notice that if we apply J_f and then $J_f^\dagger$, we will have reset the junk register, but we will have *also* reset the function register, which we don't want to do. Modify this idea slightly to obtain a complete description of a circuit that computes B_f using J_f, $J_f^\dagger$, COPY, NOT, and QAND gates. *Hint: You are allowed to introduce new ancilla registers.*

(g) Suppose there is a classical circuit with k_A 2-input AND gates and k_N NOT gates which computes some Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. Write down, in terms of k_A and k_N, the number of gates needed to implement B_f in a quantum circuit. You are only allowed to use the COPY, NOT, and QAND gates. Conclude by noticing that if f can be implemented efficiently by a classical circuit, then B_f can be implemented with just a constant factor overhead by a quantum circuit.

4. **Optional problem: AND, OR, and NOT are universal for classical computation**

This question is optional and will not be graded

In this problem, we will walk through the steps of proving Theorem 1.

(a) Show there is a classical circuit consisting only of 2-input AND gates that can compute the n-input AND gate. Formally, the function $\mathrm{AND}_n\colon \{0,1\}^n \to \{0,1\}$ we are trying to compute has the behavior

$$\mathrm{AND}_n(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } x_1 = x_2 = \cdots = x_n = 1 \\ 0 & \text{otherwise} \end{cases}$$

for all inputs $x_1, x_2, \ldots, x_n \in \{0,1\}$. You can describe your circuit for AND_n in words, or you can draw a sketch of what the circuit would look like.

(b) Notice that we can now use AND_n gates in our circuit constructions since we've shown how to construct them from 2-input AND gates in part (a). Show that using one AND_n gate and some NOT gates, we can construct the OR_n function:

$$\mathrm{OR}_n(x_1, x_2, \ldots, x_n) = \begin{cases} 0 & \text{if } x_1 = x_2 = \cdots = x_n = 0 \\ 1 & \text{otherwise} \end{cases}$$

for all inputs $x_1, x_2, \ldots, x_n \in \{0,1\}$.

(c) Suppose the function $f\colon \{0,1\}^n \to \{0,1\}$ we are trying to compute only has a single input $y = (y_1, \ldots, y_n)$ for which $f(y) = 1$. Show there is a classical circuit consisting of AND_n gates and NOT gates that computes f.

(d) Combining all of the above, show that there is a classical circuit consisting of AND and NOT gates for *every* Boolean function $f\colon \{0,1\}^n \to \{0,1\}$.