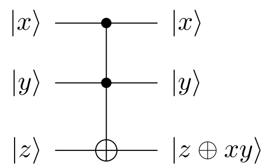
Introduction to Quantum Computing
Programming Exercises

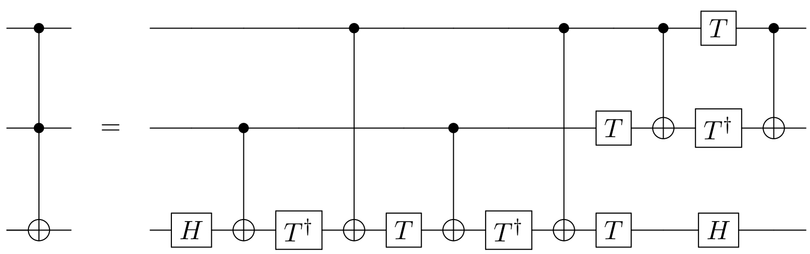
1 Quantum Circuits with Quirk

1. In the lecture, we introduced the Toffoli gate, which is an example of a multi-controlled gate: when both control bits are $|1\rangle$, the target bit is flipped.

   Quirk lets us easily build an n-controlled gate, with n control bits and a single target. Build a Quirk circuit that implements a controlled gate with 4 control qubits, using only Toffoli, CNOT, and single-bit gates.

2. Most quantum computers do not directly implement three-qubit gates like the Toffoli gate. The circuit below[1] shows an implementation of Toffoli using only single- and two-qubit gates.

   Use Quirk to show that this circuit is equivalent to a Toffoli gate. Note that the $T^\dagger$ gate is the inverse of T, which is labelled as T^{-1} in Quirk.

3. Read the brief tutorial on *teleportation* provided on the web site. Implement the circuit in Quirk and see how an arbitrary quantum state can be reproduced at a remote site, using an entangled pair of qubits and two classical bits.

   If you just want to experiment with the circuit, there's a demo versions already provided by Quirk.

2 Quantum Circuits with Qiskit

4. Write a Qiskit function for a generalized n-qubit full adder. The function takes a quantum circuit, integer n, two n-qubit input registers, and an $(n+1)$-qubit output register.

   You can create additional (ancilla) qubits in the function and add them to the circuit using the **add_register** method of `QuantumCircuit`. Be sure to uncompute any ancilla bits.

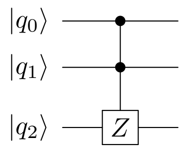
[1] Source: `https://en.wikipedia.org/wiki/Toffoli_gate`

Use the function to generate a 4-bit adder circuit and print a drawing of the circuit. Prepare the one of the inputs as a fixed value, and the other as a complete superposition. Measure the output qubits and plot the result.

5. Use Qiskit to create a circuit that performs a phase shift on a 3-qubit value when all of the qubits are equal to 1.

   First implement with a four-qubit circuit, as described in the lecture. Then implement the following circuit and convince yourself that it does the same thing without using an extra qubit.

   Use the statevector simulator. It will be easier to look at the output if you use `.round(2)` on the result vector.



6. Read the brief tutorial on *teleportation* provided on the web site. Implement the circuit in Qiskit and see how an arbitrary quantum state can be reproduced at a remote site, using an entangled pair of qubits and two classical bits.

7. From the Qiskit textbook: Show that the Hadamard gate can be implemented using R_x and R_z gates (rotations in the X and Z axis, respectively).

   `https://qiskit.org/textbook/ch-ex/ex2.html`

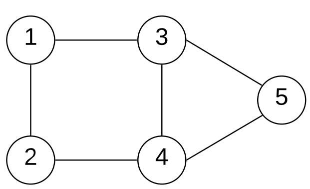
3 Optimization with D-Wave

8. Write a D-Wave program to compute XOR, such that $z = x \oplus y$.

   This is a little more challenging than it sounds. First, try to solve with three qubits, and prove to yourself that it can't be done with three qubits! Then add a fourth qubit as an ancilla — the state of this qubit does not matter in the answer (it can be either 0 or 1), but it is used to help drive the other three qubits into the proper states.

   Demonstrate your solution using the Quantum Apprentice spreadsheet, or the D-Wave ExactSolver.

9. Solve the MaxCut problem for the undirected graph in the figure below.



   Given a graph $G(V, E)$ with vertices V and edges E, MaxCut seeks to partition V into two sets, such that the number of edges between the two sets is as large as possible. (We "cut" these edges to partition the graph.)

   Let x_i be a binary variable. If vertex i is in set 0, then $x_i = 0$, otherwise $x_i = 1$ indicates that the vertex is in set 1. A cut edge goes from a vertex in one set to a vertex in the other. Therefore, if (i, j) is cut, the value $x_i + x_j - 2x_ix_j$ is 1.

   Therefore, we wish to find a set of vertex assignments, x_i, that maximizes the sum of $x_i + x_j - 2x_ix_j$ over all edges.

Derive a QUBO for this problem. (We can maximize a function by minimizing its negative.) Use the D-Wave simulator (ExactSolver) and hardware (DWaveSampler) to find a solution.

10. Implement the three-bit adder circuit described in the lecture. Experiment with various minor embedding strategies and compare the quality of results.

11. Write a program to find all combinations of 3-bit values x and y such that $x + y = 6$. You might try to write the QUBO directly, but also see the QUBO tutorial by Glover and Kochenberger (on the web site) to learn how to use penalty functions to incorporate constraints.