Lab 1 Quantum Circuits Basics

CS7400 Online
Georgia Institute of Technology
(Dated: Summer 2025)

For each question, you need to complete the associated `question*.py` file by implementing the required functions for that question. Submit `question1.py`, `question2.py`, `question3.py`, and `question4.py` to Gradescope Lab-1.

I. CREATING QUANTUM CIRCUITS

In this question, you need to construct the quantum circuis and return a QuantumCircuit object for each part according to the figure provided. You may want to refer to qiskit documentation[1] to learn how to build quantum circuits.

A. Creating Superposision

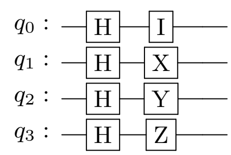

FIG. 1. A quantum circuit that creates superpositions.

B. Three Entangled Qubits Circuit

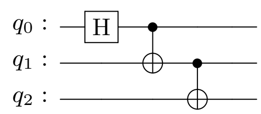

FIG. 2. A quantum circuit that entangles three qubits.

C. Bernstein-Vazirani Algorithm Circuit

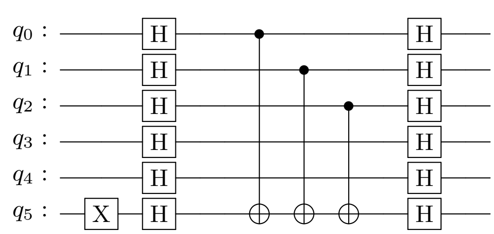

FIG. 3. A quantum circuit of Bernstein-Vazirani Algorithm.

D. Quantum Fourier Transform Circuit

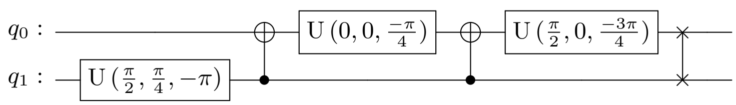

FIG. 4. A quantum circuit of Quantum Fourier Transform.

II. MEASURING QUANTUM CIRCUITS

Qiskit quantum circuits can be run on either real quantum computers, or simulators. For example, the `BasicSimulator`[2]. In this question, you need to follow the instructions and run the circuits you have implemented in Question I, and calculate the probability of measuring the target state using the Counts[3] of the simulation. Your answer p will be considered correct if $|p - p_0| < 0.1p_0$ where p_0 is the theoretical probability. For measuring the qubits you can use`measure_all()` or `measure(qreg,creg)` methods[4][5].

A. Creating Superposision

Calculate the probability of finding $|q_3q_2q_1q_0\rangle$ in state $|0101\rangle$.

B. Three Entangled Qubits Circuit

Calculate the probability of finding $|q_0\rangle$ in state $|1\rangle$.

C. Bernstein-Vazirani Algorithm Circuit

Calculate the probability of finding $|q_2q_1q_0\rangle$ in state $|010\rangle$.

D. Quantum Fourier Transform Circuit

Calculate the probability of finding $|q_1\rangle$ in state $|+\rangle$.

III. BB84 PROTOCOL

In this question, you need to implement the BB84 protocol for quantum information transmission, for the base class `QuantumAgent` for Alice, Bob, and Eve in `question3.py`. The `BB84` protocol is provided. Below is a brief review for the BB84 protocol.

An n-bit classical information is represented by an ordered set (all sets in this question are ordered sets, hence hereafter referred to simply as *set*, and realized by a `numpy.array`) S of n elements where each element $s_i \in S$ satisfies that $s_i = 0 \vee s_i = 1$, for all $i \in [0, n-1]$. Alice randomly generates a set B of n elements where each element $b_i \in B$ satisfies that $b_i = 0 \vee b_i = 1$, for all $i \in [0, n-1]$, and encodes each s_i as a 1-qubit quantum circuit q_i in computational (X) basis if $b_i = 0$, or in Hadamard (Z) basis if $b_i = 1$. The classical information is then encoded as a set $\hat{S}$ of n qubits, each one is a *single qubit quantum circuit*, by Alice. The qubits are then sent to Bob from Alice.

After Bob has received the qubits, Bob randomly creates a set B' from which Bob decides which basis to use measure each qubit, the same way as how Alice creates B, and creates the recovered mesage S' After all qubits are measured, Alice and Bob compare the sets B and B', and together derive an index set $\mathcal{I}$ of the indices of the common elements of B and B', defined as $\mathcal{I} = \{j \,|\, b_j = b'_j,\, \forall j \in [0, n-1],\, b_j \in B,\, b'_j \in B'\}$. If the size of the index set is less than $n/2$: $\mathcal{I} < n/2$ the transmission should be aborted, otherwise Alice and Bob will continue.

For simplicity, Alice and Bob then compare the recovered message $S'_{\mathcal{I}}$ and original message $S_{\mathcal{I}}$ restricted to the index set $\mathcal{I}$, where $S_{\mathcal{I}} = \{s_i | s_i \in S \wedge i \in \mathcal{I}\}$ and $S'_{\mathcal{I}} = \{s_i | s_i \in S' \wedge i \in \mathcal{I}\}$. If more than half of their elements match, the protocol is successful, otherwise, the exisitance of Eve is detected and the protocol has failed.

IV. QUANTUM ADDER

In this question, you need to implement the quantum version of fundamental calculation modules of classical computers. The `encoder` and `decoder` are provided and will be used for Gradescope tests. The `encoder` converts the two classical inputs to a quantum circuit. The `decoder` converts a quantum circuit to a classical output by measuring the circuit exactly once, so you circuit should be deterministic. Your implementation can be tested using the provided `local_test.py` file.

A. Quantum AND

Implement a quantum circuit to perform bitwise AND calculation for two 1-bit binary inputs in `quantum_and` in `question4.py`.

B. Quantum OR

Implement a quantum circuit to perform bitwise OR calculation for two 1-bit binary inputs in `quantum_or` in `question4.py`.

C. Quantum XOR

Implement a quantum circuit to perform bitwise XOR calculation for two 1-bit binary inputs in `quantum_and` in `question4.py`.

D. Quantum Adder

Implement a quantum circuit in `quantum_adder` to add two 4-bit unsigned integers and outputs one 4-bit unsigned integer in `question4.py`.

E. Signed Adder

Complete the `encoder_signed` function and `decoder_signed` function in `question4.py` so that they work with negative input integers.

---

[1] `https://docs.quantum.ibm.com/guides/construct-circuits`.
[2] `https://docs.quantum.ibm.com/api/qiskit/qiskit.providers.basic_provider.BasicSimulator`.
[3] `https://docs.quantum.ibm.com/api/qiskit/qiskit.result.Counts`.
[4] `https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.QuantumCircuit#measure`.
[5] `https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.QuantumCircuit#measure_all`.