Lab 2 Analyzing Quantum Algorithms

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`, `question4.py`, `question5.py`, and `question6.py` to Gradescope Lab-2.

I. SUPERDENSE CODING

For this question, implement the components of superdense coding in `question1.py` to transmit n bits of classical binary information.

Superdense Coding is implemented by the following steps

1. Preparation of qubits by Charlie in `qubit_preparation`.
2. Distribution of the prepared qubits from Charlie to Alice and Bob. (No implementation needed)
3. Encoding of first $\lceil n/2 \rceil$ qubits according to the classical information by Alice in `qubit_encoding`.
4. Transmission of qubits from Alice to Bob. (No implementation needed)
5. Restoration of the classical information by Bob in `qubit_decoding`.

Your implementation should work for binary classical information of length 1 to 16.

II. DEUTSCH AND DEUTSCH-JOZSA ALGORITHM

In this question you need to implement the quantum oracles U_f of Deutsch and Deutsch-Jozsa circuits based on the definition of the given functions.

A. Deutsch

Deutch algorithm tests if a 1-bit function $f : \{0, 1\} \to \{0, 1\}$ is constant or balanced. Implement the following four functions (f_1, f_2, f_3, f_4) as oracles of Deutsch circuits in `question2.py`.

- $f_1(0) = 0 \wedge f_1(1) = 0$, in `deutsch_f1_oracle`.
- $f_2(0) = 1 \wedge f_2(1) = 0$, in `deutsch_f2_oracle`.
- $f_3(0) = 0 \wedge f_3(1) = 1$, in `deutsch_f3_oracle`.
- $f_4(0) = 1 \wedge f_4(1) = 1$, in `deutsch_f4_oracle`.

B. Deutsch-Jozsa

A four-bit function $g(x)$ where x is a 4-bit unsigned integer can be defined as the following:

$$g(x) = 0, \text{ if } x \text{ is even}, \tag{1}$$

$$g(x) = 1, \text{ if } x \text{ is odd}. \tag{2}$$

Implement $g(x)$ as an oracle for Deutsch-Jozsa algorithm in the `deutsch_jozsa_oracle` function in `question2.py`.

III. BERNSTEIN VAZIRANI ALGORITHM

In this question, you will need to implement the Bernstein Vazirani (BV) algorithm with an oracle function U_f of secrete key `0101010`. Your implementation should not contain any measurments.

A. Ideal BV Circuit

Implement your BV circuit in the `bv_ideal` function.

B. Noisy BV Circuit

On a real quantum computer with physical qubits, we get hardware errors. One type of error is the systematic control error, which can cause the state vector to have an under or over-rotation on the Bloch sphere. We will use U gates to emulate control error in H. For example, if all the H gates in BV are replaced by $U(\frac{\pi}{2}, \frac{\pi}{16}, \pi)$, we can emulate a type of control error and study its impact on the fidelity of BV. Implement your circuit in the `bv_noisy` function with all H gates substituted by $U(\frac{\pi}{2}, \frac{\pi}{16}, \pi)$ gates.

IV. SIMON'S ALGORITHM

Simon's algorithm deals with the period finding problem which enables quantum advantage against classical algorithms towards solving large number factorization problems. A particular oracle of Simon's algorithm is shown in Figure 1:

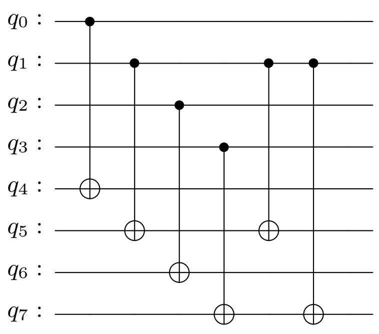

FIG. 1. The oracle for Simon's circuit.

A. Circuit Implementation

Implement Simon's algorithm based on the given oracle in `simon` function in `question4.py` based on the given oracle. Your implementation should not contain any measurments.

B. Secret String

Return the calculated secret string `s` in `calculate_secret` in `question4.py`. (*Hint: The answer can be hard coded.*)

V. GROVER'S ALGORITHM

Grover's algorithm can be used to amplify the probability of a target state by a rotation oracle and an inversion oracle. In this question, you need to complete the Grover's circuit using the given phase rotation and inversion oracles (see Part **??** Figure **??** and Figure **??**), find the target state, and also find the probability of the target state after running Grover's circuit.

A. Rotation and Inversion

Implement Grover's algorithm circuit in `grover` function in `question5.py` using the given phase rotation and inversion oracles. Your implementation should not contain any measurments.

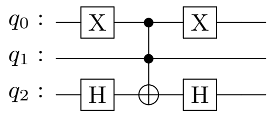

FIG. 2. The rotation operator of Grover's circuit.

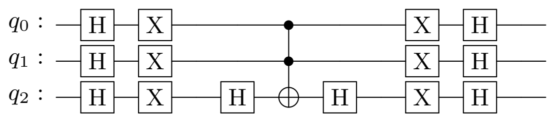

FIG. 3. The inversion operator of Grover's circuit.

B. Target State

Implement the `target_state` function in `question5.py` to return the target state found by the Grover's circuit.

C. Probability

Implement the `post_processing` function in `question5.py` to return the probability of the target state of the given Grover's algorithm circuit of n rounds of Grover application where $n \in \mathbb{Z} \wedge n \in [1, 10]$.

VI. DESIGN QUANTUM CIRCUITS

In this question, you need to design quantum circuits to generate the given quantum states in Dirac's notations. For each part, return a QuantumCircuit object that corresponds to the given state $|\psi\rangle$. The Dirac's notations should be understood in Qiskit little-endian ordering. Your implementation need to satisfy the following two constraints:

- You may not use the `initialize` instruction.
- You can only use the following quantum gates: `h`, `x`, `y`, `z`, `s`, `t`, `u`, `cu`, `rx`, `ry`, `rz`, `cx`, `crx`, `cry`, `crz`, `ccx`.

A. $|\psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$

B. $|\psi\rangle = \frac{1}{2}|010\rangle + \frac{\sqrt{3}}{2}|101\rangle$

C. $|\psi\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$

D. $|\psi\rangle = \frac{1}{2}|0001\rangle + \frac{i}{2}|0010\rangle - \frac{1}{2}|0100\rangle - \frac{i}{2}|1000\rangle$

E. $|\psi\rangle = \frac{1}{\sqrt{3}}|001\rangle + \frac{1}{\sqrt{3}}|010\rangle + \frac{1}{\sqrt{3}}|100\rangle$