Assignment 4

***Due**: Wednesday, Apr 13th, 2022 @ 5:00PM*

CPSC 447/547 Introduction to Quantum Computing (Spring 2022)

1 Introduction

Welcome to Assignment 4 for CPSC 447/547 (Introduction to Quantum Computing). As usual, collaboration is encouraged; if you discussed with anyone besides the course staff about the assignment, *please list their names* in your submission.

Getting Started.

This assignment has two parts, a *written portion* and a *programming portion*. Typesetting your solutions to the written portion is not mandatory but highly encouraged. See the instructor's note on Ed for details about Latex for quantum computing. Some basic familiarity with Python and object-oriented programming is required to complete the programming portion of this assignment. To start,

- Create a folder for Assignment 4, e.g., `A4/`
- Download the starter files for this assignment to that folder from the course website:
  - `written.tex`
  - `A4.py`
  - `requirement_A4.py` (Do not modify)
- Write your solutions in `written.tex` and `A4.py`
- Debug and test your solution by running '`python3 A4.py`' on command line. This will check for any violation of the requirements and run correctness tests. Feel free to add more tests in `A4.py`. Do not hardcode your solutions for each test cases.
- Three days before the submission deadline (on Monday), we are going to release an online grading tool, where you can submit your programming solution and run a small set of public tests. Our official grading script contains more private tests than the public online script. You can access from within Yale's secure network once it is live:

  `http://172.28.228.90/cpsc447/A4-public-tests.html`

Submission.

Once you have completed and are ready to submit, upload two files to Canvas: `written.pdf` and `A4.py`. Late submissions (for up to two days) will receive a 50% penalty. Your written solution will be graded manually by our course staff; your programming solution will be graded using our grading script. If your file fails the `requirement_A4.py` check, a **0** score will be assigned, in which case you will have 2 days to fix the issue and re-submit, with a 50% penalty.

WRITTEN PORTION
This portion of the assignment has a total of 65 *points.*

2 Grover's Algorithm

Task 2.1 (10 pts)

Consider a function $f : \{0,1\}^4 \to \{0,1\}$. In the standard Grover's algorithm, we assume we know the number of inputs such that $f(x) = 1$. Suppose there are $K = 2$ of those satisfied inputs: $x = 8 = 1000$ and $x = 9 = 1001$.

(a) Given an oracle to the above function, $O_f^{\pm}$, we want to find those inputs x such that $f(x) = 1$. What is the optimal number of iterations in Grover's algorithm? That is, after how many iterations of oracle access is the probability of measuring $|1000\rangle$ or $|1001\rangle$ maximized? (Write the exact number, not in the big-O notation.)

(b) After the optimal number of iterations, what is the probability that $|1001\rangle$ is measured? (Write the exact number, not in big-O.)

Task 2.2 ($\star$ pts)

Following the previous question,

(a) Suppose we apply one more iteration than the optimal number, what is the probability that $|1001\rangle$ is measured? (Write the exact number, not in big-O.)

3 Density Operators

Task 3.1 ($\star$ pts)

Recall from lecture, we define the density operator for an n-qubit quantum state as a Hermitian matrix $\rho \in \mathbb{C}^{2^n \times 2^n}$ where $\rho \succcurlyeq 0$ (positive semi-definite) and $\mathrm{Tr}(\rho) = 1$ (unit trace).

(a) Show that if $\langle v|\rho|v\rangle \geq 0$ holds for all unit vectors $|v\rangle$, then $\rho \succcurlyeq 0$.

Task 3.2 (30 pts)

A density operator can be used to represent an ensemble of quantum states. Suppose the probability of sampling $|\psi_j\rangle$ is p_j, then the matrix for the density operator is

$$\rho = \sum_j p_j |\psi_j\rangle \langle\psi_j|$$

(a) Suppose we have a pure state, e.g., $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ with probability 1. What is its density matrix ρ_a? (Write down the matrix explicitly.)

(b) Suppose we have a mixed state, e.g., $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$. What is its density matrix ρ_b? (Write down the matrix explicitly.)

(c) Suppose we have another mixed state, e.g., $|0\rangle$ with probability $\frac{1}{2}$ and $|+\rangle$ with probability $\frac{1}{2}$. What is its density matrix ρ_c? (Write down the matrix explicitly.)

(d) Recall from lecture, in the Bloch sphere picture, we can locate a density matrix $\rho = \frac{1}{2}(I + r_x\sigma_x + r_y\sigma_y + r_z\sigma_z)$ with Cartesian coordinate (r_x, r_y, r_z). Write down the coordinates for ρ_a, ρ_b, ρ_c from previous questions.

(e) Notice that pure states are on the surface of the Bloch sphere and mixed states are at the interior of the Bloch sphere. We can define "how pure" a quantum state is by how close ρ is to the surface. Define $r = \sqrt{r_x^2 + r_y^2 + r_z^2}$ to be the distance from the origin to ρ. Show that $\mathrm{Tr}(\rho^2) = \frac{1}{2}(1 + r^2)$.

(f) We can use $\mathrm{Tr}(\rho^2)$ to evaluate the "purity" of n-qubit quantum states. Here $\rho \in \mathbb{C}^{2^n \times 2^n}$. What is the lower bound and the upper bound of $\mathrm{Tr}(\rho^2)$? Also write down two example quantum states, $\rho_{lower}, \rho_{upper}$, that saturate the bounds respectively.

4 Entropy

Task 4.1 ($\star$ pts)

In information theory, entropy is a quantity commonly used to measure the level of "uncertainty" in a random variable's outcomes. For example, if we have a discrete random variable x, such that x has outcome $x_1, x_2, \ldots, x_n$ with probability $p_1, p_2, \ldots, p_n$ respectively, then we can define the entropy of x as

$$H(x) = -\sum_{j=1}^{n} p_j \log(p_j).$$

Here the log function has base 2, so the entropy is in the unit of bits [1]. In the special case where a random variable has two outcomes, such as a coin flip where the probability of heads is p and the probability of tails is $1 - p$. We define the binary entropy as

$$H(p) = -p\log(p) - (1-p)\log(1-p).$$

(a) Suppose Alice flips a fair coin, i.e., getting the outcome of heads ($x = H$) or tails ($x = T$) with equal probability. What is $H(x)$?

(b) Consider Alice and Bob play a series of 5 games. First player who wins 3 games wins the series. For example, the series might end with outcome AAA, or BAAA, or BBAAB. Assuming either player has equal probability of winning a game (with no ties). Let x be the random variable for the number of games played in a series. What is $H(x)$?

Task 4.2 (25 pts)

Recall from lecture, our definition of density matrix ρ is analogous to that of probability density $p = \{p_1, p_2, p_3, \ldots, p_n\}$. For example, the positive condition in $\rho \succcurlyeq 0$ is analogous to $p \geq 0$ and the unity condition $\mathrm{Tr}(\rho) = 1$ is analogous to $\sum_j p_j = 1$.

[1] https://en.wikipedia.org/wiki/Entropy_(information_theory)

We can therefore define a similar measure of uncertainty in qubits. This is called the von Neumann entropy:

$$S(\rho) = -\operatorname{Tr}(\rho \log(\rho)),$$

where $\log(\rho)$ is defined by Spectral theorem and applying the log function to the eigenvalues: $\log(\rho) = \sum_j \log(\lambda_j)\,|\psi_j\rangle\langle\psi_j|$.

(a) For a single qubit pure state. e.g., $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ with probability 1. What is its entropy $S(\rho_a)$?

(b) For a mixed state, e.g., $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$. What is its entropy $S(\rho_b)$?

(c) For another mixed state, e.g., $|0\rangle$ with probability $\frac{1}{2}$ and $|+\rangle$ with probability $\frac{1}{2}$. What is its entropy $S(\rho_c)$?

(d) For an arbitrary single-qubit mixed state with density matrix $\rho = \frac{1}{2}(I + r_x\sigma_x + r_y\sigma_y + r_z\sigma_z)$, that is with Cartesian coordinate (r_x, r_y, r_z) in Bloch sphere. What are the eigenvalues of ρ? (Write your answer in terms of r_x, r_y, r_z.)

(e) Following the previous question, what is $S(\rho)$? (Write your answer in terms of r_x, r_y, r_z.)

PROGRAMMING PORTION

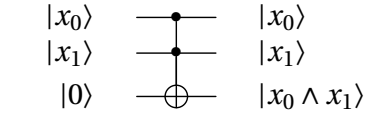
This portion of the assignment has a total of 35 *points.*

5 Oracles

In lectures, we discussed quantum algorithms using quantum oracles. The power of quantum oracles lies in computing Boolean functions in superposition. For example, for a Boolean function $f : \{0,1\}^n \to \{0,1\}^m$, we have the (bit) oracle:

$$\sum_j \alpha_j\,|j\rangle\,|0\rangle \xrightarrow{O_f} \sum_j \alpha_j\,|j\rangle\,|f(j)\rangle\,.$$

In the following tasks, we will explore how to use quantum gates to implement quantum oracles. Let's take a look at an example. Suppose the function is the AND logic gate: $f(x_0, x_1) = x_0 \wedge x_1$. We can use a Toffoli gate to implement AND reversibly:



The implementation can be found in the `AND(qc, in_qubit_1, in_qubit_2, out_qubit)` function. The oracle transforms $|x_0x_10\rangle$ to $|x_0x_1(x_0 \wedge x_1)\rangle$, for all $x_0, x_1 \in \{0,1\}$. We can verify the correctness of this implementation with some test cases. For example, for input $|110\rangle$, we should get $|111\rangle$; for input $\frac{1}{\sqrt{2}}|010\rangle + \frac{1}{\sqrt{2}}|110\rangle$, we should get $\frac{1}{\sqrt{2}}|010\rangle + \frac{1}{\sqrt{2}}|111\rangle$.

Task 5.1 (10 pts)

Implement the logical-OR oracle in the function

```
OR(qc, in_qubit_1, in_qubit_2, out_qubit)
```

using gates in the `QuantumCircuit` class. Here, `in_qubit_1, in_qubit_2, out_qubit` are qubit indices. `OR(qc,0,1,2)` transforms $|x_0x_10\rangle$ to $|x_0x_1(x_0 \vee x_1)\rangle$. You can assume the output qubit is always initialized to $|0\rangle$.

For example, if the quantum state is initialized to $|010\rangle$ in `qc`, then `OR(qc,0,1,2)` should transform the quantum state to $|011\rangle$. If the quantum state is initialized to $\frac{1}{\sqrt{2}}|010\rangle + \frac{1}{\sqrt{2}}|000\rangle$ in `qc`, then `OR(qc,2,1,0)` should transform the quantum state to $\frac{1}{\sqrt{2}}|110\rangle + \frac{1}{\sqrt{2}}|000\rangle$.

Task 5.2 ($\star$ pts)

Implement the n-bit logical-AND oracle in the function

```
nAND(qc, in_qubits, out_qubit, ancilla)
```

using gates in the `QuantumCircuit` class. Here, `in_qubits, out_qubit` are qubit indices. `ancilla` has the same length as `in_qubits`, initialized to $|0^n\rangle$. `nAND(qc,[0,1,2,3],4,ancilla)` transforms $|x_0x_1x_2x_30\rangle$ to $|x_0x_1x_2x_3(x_0 \wedge x_1 \wedge x_2 \wedge x_3)\rangle$. Notice that `ancilla` is omitted for simplicity. You can assume the output qubit is always initialized to $|0\rangle$. You do not have to use the qubits in `ancilla`, but if you do, they must be returned to the $|0^n\rangle$ state.

For example, if the quantum state is initialized to $|0100\rangle$ in `qc`, then `nAND(qc,[0,1,2],3)` should transform the quantum state to $|0100\rangle$. If the quantum state is initialized to $\frac{1}{\sqrt{2}}|1110\rangle + \frac{1}{\sqrt{2}}|1010\rangle$ in `qc`, then `nAND(qc,[0,1,2],3)` should transform the quantum state to $\frac{1}{\sqrt{2}}|1111\rangle + \frac{1}{\sqrt{2}}|1010\rangle$.

Task 5.3 (10 pts)

Implement the n-bit logical-OR oracle in the function

```
nOR(qc, in_qubits, out_qubit, ancilla)
```

using gates in the `QuantumCircuit` class. Here, `in_qubits, out_qubit` are qubit indices. `ancilla` has the same length as `in_qubits`, initialized to $|0^n\rangle$. `nOR(qc,[0,1,2,3],4,ancilla)` transforms $|x_0x_1x_2x_30\rangle$ to $|x_0x_1x_2x_3(x_0 \vee x_1 \vee x_2 \vee x_3)\rangle$. Notice that the `ancilla` is omitted here for simplicity. You can assume the output qubit is always initialized to $|0\rangle$. You do not have to use the qubits in `ancilla`, but if you do, they must be returned to the $|0^n\rangle$ state.

For example, for `nOR(qc,[0,1,2],3,ancilla)`, if the quantum state is initialized to $|0100\rangle$ in `qc`, then it should be transformed to $|0101\rangle$. Again, `ancilla` is omitted here for simplicity. If the quantum state is initialized to $\frac{1}{\sqrt{2}}|1110\rangle + \frac{1}{\sqrt{2}}|1010\rangle$ in `qc`, then it should be transformed to $\frac{1}{\sqrt{2}}|1111\rangle + \frac{1}{\sqrt{2}}|1011\rangle$.

Task 5.4 ($\star$ pts)

Implement the Majority oracle in the function

`MAJ(qc,in_qubit_1,in_qubit_2,in_qubit_3,out_qubit)`

using gates in the `QuantumCircuit` class. Here, `MAJ(qc,0,1,2,3)` transforms $|x_0x_1x_20\rangle$ to $|x_0x_1x_2((x_0 \cdot x_1) \oplus (x_0 \cdot x_2) \oplus (x_1 \cdot x_2))\rangle$. You can assume the output qubit is always initialized to $|0\rangle$.

For example, for `MAJ(qc,0,1,2,3)`, if the quantum state is initialized to $|0100\rangle$ in `qc`, then it should be transformed to $|0100\rangle$. If the quantum state is initialized to $\frac{1}{\sqrt{2}}|1010\rangle + \frac{1}{\sqrt{2}}|1000\rangle$ in `qc`, then it should be transformed to $\frac{1}{\sqrt{2}}|1011\rangle + \frac{1}{\sqrt{2}}|1000\rangle$.

Task 5.5 ($\star$ pts)

Implement the n-bit Majority oracle in the function

`nMAJ(qc,in_qubits,out_qubit, ancilla)`

using gates in the `QuantumCircuit` class. Here, `ancilla` has the same length as `in_qubits`, initialized to $|0^n\rangle$. The `MAJ(qc,[0,1,2,3,4],5,ancilla)` function transforms $|x_0x_1x_2x_3x_40\rangle$ to $|x_0x_1x_2x_3x_4(Maj(x_0,x_1,x_2,x_3,x_4))\rangle$. Notice that `ancilla` is omitted for simplicity. You can assume the output qubit is always initialized to $|0\rangle$. You can also assume n is odd. You do not have to use the qubits in `ancilla`, but if you do, they must be returned to the $|0\rangle$ state.

For example, for `nMAJ(qc,[0,1,2,3,4],5)`, if the quantum state is initialized to $|010010\rangle$ in `qc`, then it should be transformed to $|010010\rangle$. If the quantum state is initialized to $\frac{1}{\sqrt{2}}|101010\rangle + \frac{1}{\sqrt{2}}|100010\rangle$ in `qc`, then it should be transformed to $\frac{1}{\sqrt{2}}|101011\rangle + \frac{1}{\sqrt{2}}|100010\rangle$.

6 Reflections

Some quantum algorithms use special unitary operations, such as the reflections, to perform computation. For example, Grover's algorithm interleaves an (phase) oracle with a reflection operator (also known as the diffusion operator). In this question, we will be implementing these reflection operations using quantum circuit.

Task 6.1 (15 pts)

Implement the reflection through the $|0^n\rangle$ hyperplane in the function

`ReflectZero(qc, qubits, ancilla)`

Here `qubits` and `ancilla` are qubit indices. Here `ancilla` has the same length as `in_qubits`, initialized to $|0^n\rangle$. Recall from lecture, we define the reflection $R_0^{\pm}$ as follows:

$$R_0^{\pm}|x\rangle = \begin{cases} -|x\rangle \text{ if } |x\rangle = |0^n\rangle \\ |x\rangle \text{ otherwise} \end{cases}$$

Notice that the we can also view the reflection as adding a "negative sign" if n-bit OR function on x is false. In other words, it is the phase oracle for the `Not(nOR(x))` function. (Hint: the previous bit oracle implementation for `nOR` is going to be useful in this task.) The unitary matrix form of the reflection is:

$$R_0^{\pm} = I - 2|0^n\rangle\langle 0^n|$$

For example, for `ReflectZero(qc, [0,1,2,3], ancilla)`, if the quantum state is initialized to $|0100\rangle$ in `qc`, then it should be transformed to $|0100\rangle$. Notice that `ancilla` is omitted for simplicity. If the quantum state is initialized to $\frac{1}{\sqrt{2}}|1010\rangle + \frac{1}{\sqrt{2}}|0000\rangle$ in `qc`, then it should be transformed to $\frac{1}{\sqrt{2}}|1010\rangle - \frac{1}{\sqrt{2}}|0000\rangle$.

Task 6.2 ($\star$ pts)

Implement the reflection through the $|+^n\rangle$ hyperplane in the function

```
ReflectUniform(qc, qubits, ancilla)
```

Here `qubits` and `ancilla` are qubit indices. Here `ancilla` has the same length as `in_qubits`, initialized to $|0^n\rangle$. Recall from lecture, we define the reflection $R_+^{\pm}$as follows:

$$R_+^{\pm}|x\rangle = \begin{cases} -|x\rangle \text{ if } |x\rangle = |+^n\rangle \\ |x\rangle \text{ otherwise} \end{cases}$$

This is also known as the "Diffusion operator" in Grover's algorithm. (Hint: the previous implementation of `ReflectZero` is going to be useful in this task.) The unitary matrix form of the reflection is:

$$R_+^{\pm} = I - 2|+^n\rangle\langle+^n|$$

For example, for `ReflectUniform(qc, [0,1], ancilla)`, if the quantum state is initialized to $|01\rangle$ in `qc`, then it should be transformed to $\frac{1}{2}(-|00\rangle + |01\rangle - |10 - |11\rangle\rangle)$. Notice that `ancilla` is omitted for simplicity.