Assignment 3

***Due**: Wednesday, Nov 1st, 2023 @ 5:00PM*

CPSC 447/547 Introduction to Quantum Computing (Fall 2023)

1 Introduction

Welcome to Assignment 3 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*. The tasks that are marked by "($\star$ pts)" are optional. 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. No Python packages, except for `math` and `numpy`, are allowed. To start,

- Create a folder for Assignment 3, e.g., `A3/`
- Download the starter files for this assignment to that folder from the course website:
  - `written.tex`
  - `A3.py`
  - `requirement_A3.py` (Do not modify)
- Write your solutions in `A3.py` (for programming tasks) and in `written.tex` or on paper (for written tasks).
- Debug and test your solution locally by running '`python3 A3.py`' on command line. This will check for any violation of the requirements and run correctness tests. Feel free to add more tests in `A3.py`. Do not hardcode your solutions for each public test cases.

Submission. Once you have completed and are ready to submit, upload two files to Gradescope (accessed through Canvas): `written.pdf` and `A3.py`. Gradescope will immediately show the results from running the requirement test and public test cases. If your file fails the tests in `requirement_A3.py`, a **0** score will be assigned.

After the deadline, your written solution will be graded manually by our course staff; your programming solution will be graded using our auto-grading script that contains private test cases. Late submissions (for up to two days) will receive a 50% penalty.

WRITTEN PORTION
This portion of the assignment has a total of 65 *points.*
Any tasks marked with (⋆ pts) are optional.

2 Quantum Oracles

Task 2.1 (10 pts)

Recall from lecture, we have defined the phase oracle for a function $f : \{0,1\}^n \to \{0,1\}$ as follows:

$$O_f^{\pm}|x\rangle = (-1)^{f(x)}|x\rangle.$$

Suppose we have an oracle to the following function $f : \{0,1\}^2 \to \{0,1\}$:

$$f(x) = \begin{cases} 1, \text{ if } x = 01 \\ 0, \text{ otherwise} \end{cases}$$

Given an input quantum state $\big|\psi\big\rangle = \frac{1}{\sqrt{3}}(|00\rangle + |01\rangle + |10\rangle)$. What is its state after applying the phase oracle to $\big|\psi\big\rangle$. That is, compute $O_f^{\pm}\big|\psi\big\rangle$.

Task 2.2 (10 pts)

What is the unitary matrix for the $O_f^{\pm}$ from the previous question? (Hint: $O_f^{\pm}$ can be viewed as a reflection operator.)

Task 2.3 (⋆ pts)

Give a circuit implementation of the $O_f^{\pm}$ from the previous questions.

Task 2.4 (⋆ pts)

Suppose now we are given an oracle to the following function $f : \{0,1\}^2 \to \{0,1\}$:

$$f(x) = \begin{cases} 1, \text{ if } x = 01 \text{ or } 00 \\ 0, \text{ otherwise} \end{cases}$$

Answer the previous three questions. That is, compute $O_f^{\pm}\big|\psi\big\rangle$, write down the unitary matrix for $O_f^{\pm}$, and give a quantum circuit for $O_f^{\pm}$.

3 Simon's Algorithm

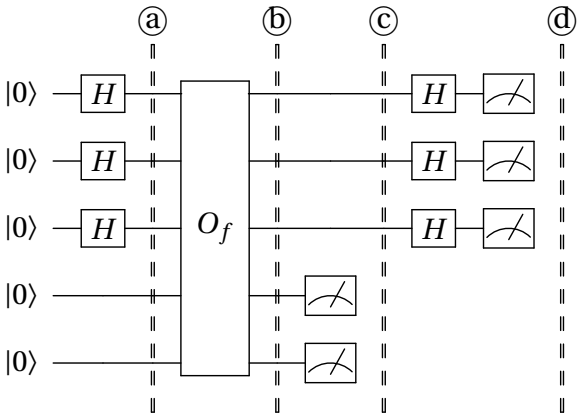
Task 3.1 (25 pts)

In Simon's algorithm, we are given a "periodic", two-to-one function and want to find its period. Let's analyze how this algorithm works for a particular function $f : \{0,1\}^3 \to \{0,1\}^2$, where $f(x + a) = f(x)$ for some unknown period a. Importantly, here addition $(+)$ is a *bit-wise addition modulo 2*. For example, $010 + 011 = 001$. Its truth table looks like this in Table 1.

Input	Output
000	$f(000) = 01$
001	$f(001) = 10$
010	$f(010) = 10$
011	$f(011) = 01$
100	$f(100) = 11$
101	$f(101) = 00$
110	$f(110) = 00$
111	$f(111) = 11$

Table 1: Truth table of function f.

We have colored the output so that you can visually inspect the period. In Simon's algorithm, we use the following quantum circuit to find the period:



We have labeled four important time steps in the quantum circuit. Answer the following questions.

(a) Visually inspecting the whole truth table in Table 1. What is the period a, such that $f(x+a) = f(x)$ for all inputs x?

(b) What is the quantum state at time step ⓐ, i.e., after $H^{\otimes 3}$?

(c) What is the quantum state at time step ⓑ, i.e., after the oracle O_f?

(d) What is the quantum state at time step ⓒ, i.e., after measuring the bottom two qubits, given that the measurement outcome is 01? Please write your answer in the form of $(\alpha_{000}|000\rangle + \alpha_{001}|001\rangle + \alpha_{010}|010\rangle + \alpha_{011}|011\rangle + \alpha_{100}|100\rangle + \alpha_{101}|101\rangle + \alpha_{110}|110\rangle + \alpha_{111}|111\rangle) \otimes |01\rangle$ with the appropriate values for α_i. Here we assume top qubit in the circuit is the most-significant bit in the state, that is $|x\rangle = |x_2 x_1 x_0\rangle$, $x = \sum_s x_s 2^s$, and $|x_2\rangle$ is the top qubit.

(e) What are the possible measurement outcomes at time step ⓓ, assuming the measurement outcome of 01 from (d)? Write down all possible outcomes and their associated probabilities.

Task 3.2 (⋆ pts)

Following the previous question,

1. If the measurement outcome at time step ⓓ is 100, what do we know about the period a? In other words, what are the possible values of a that are consistent with this measurement outcome?

2. If we repeat Simon's algorithm and obtain another measurement outcome 111 at time step ⓓ, what do we know about the period now? In other words, what are the possible values of a that are consistent with both the first measurement outcome 100 and the second measurement outcome 111?

4 Quantum Fourier Transform

Task 4.1 (20 pts)

Quantum Fourier Transform (QFT) is a very useful primitive in quantum algorithms. In this question, we will explore quantum computer's number format in the computational basis and in the Fourier basis, and discover QFT's role in this.

The standard binary representation of a non-negative integer $j \in \{0,1,2,\ldots,2^n-1\}$ is defined by a length-n bitstring: $j_{n-1}j_{n-2}\cdots j_1 j_0$ such that $j = \sum_{s=0}^{n-1} j_s 2^s$,, where $j_s \in \{0,1\}$. Let's use the number j to index the amplitude of an n-qubit quantum state:

$$\big|\psi\big\rangle = \sum_{j=0}^{2^n-1} \alpha_j \big|j_{n-1}j_{n-2}\cdots j_1 j_0\big\rangle \equiv \sum_{j=0}^{2^n-1} \alpha_j \big|j\big\rangle,$$

for complex amplitudes satisfying $\sum_j |\alpha_j|^2 = 1$.

Quantum Fourier Transform (QFT) maps one quantum state to another as follows:

$$\big|\psi\big\rangle \xrightarrow{QFT} \big|\phi\big\rangle = F_{2^n}\big|\psi\big\rangle \equiv \sum_{k=0}^{2^n-1} \beta_k |k_{n-1}k_{n-2}\cdots k_1 k_0\rangle,$$

where $\beta_k = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} w^{jk}\alpha_j$ and $w = e^{i2\pi/2^n}$.

(a) Suppose we only consider two numbers $\{0,1\}$, that is, $n = 1$. What is w? And what is the unitary matrix of F_2?

(b) In the computational basis, we define two quantum states as the basis states, $|0\rangle, |1\rangle$. We can also represent the two numbers in the Fourier basis, $\widetilde{|0\rangle} = F_2|0\rangle, \widetilde{|1\rangle} = F_2|1\rangle$. Write down the Fourier basis states in the computational basis. That is, write down $\widetilde{|0\rangle}, \widetilde{|1\rangle}$ in the form of $\alpha|0\rangle + \beta|1\rangle$ for some values of α, β.

(c) Suppose we consider four numbers $\{0,1,2,3\}$, that is, $n = 2$. What is w? And what is the unitary matrix of F_4?

(d) Since F_4 is unitary, it should be invertible. What is the matrix for the inverse, F_4^{-1}?

Task 4.2 ($\star$ pts)

Continuing from the previous question,

(a) We can define the computational basis, $|0\rangle = |00\rangle, |1\rangle = |01\rangle, |2\rangle = |10\rangle, |3\rangle = |11\rangle$. We also can use Fourier transform to represent the four numbers in the Fourier basis: $\{\widetilde{|0\rangle}, \widetilde{|1\rangle}, \widetilde{|2\rangle}, \widetilde{|3\rangle}\}$. Write down the four quantum states: $\widetilde{|0\rangle} = F_4|0\rangle, \widetilde{|1\rangle} = F_4|1\rangle, \widetilde{|2\rangle} = F_4|2\rangle, \widetilde{|3\rangle} = F_4|3\rangle$.

(b) For the following tasks, let's consider $n = 3$. We can count numbers from 0 to 7 in the computational basis or the Fourier basis. In the computational basis, a number j is represented by the standard binary format: $\big|j\big\rangle = \big|j_2 j_1 j_0\big\rangle$. In the Fourier basis, a number j is represented by

$$\widetilde{\big|j\big\rangle} = F_8\big|j\big\rangle = \frac{1}{\sqrt{8}}\sum_{k=0}^{7} w^{jk}|k\rangle = \frac{1}{\sqrt{8}}\sum_{k_0,k_1,k_2\in\{0,1\}} w^{jk}|k_2 k_1 k_0\rangle.$$

Show that $F_8\big|j\big\rangle$ is *unentangled*, by writing the quantum state as a product state: $\big(\alpha_2|0\rangle + \beta_2|1\rangle\big)\otimes \big(\alpha_1|0\rangle + \beta_1|1\rangle\big) \otimes \big(\alpha_0|0\rangle + \beta_0|1\rangle\big)$. Write your answers in terms of w.

(c) Complete the following table. Write your answers in terms of w.

Integer	0	1	2	3	4	5	6	7
Computational basis ($\big\|j\big\rangle$)	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
Fourier basis ($F_8\big\|j\big\rangle$)								

Table 2: Counting numbers in the computational basis and the Fourier basis.

PROGRAMMING PORTION

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

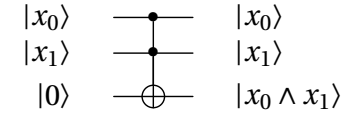
Important: *This portion is optional for CPSC 547 students.*

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:

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

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 $|x_0 x_1 0\rangle$ to $|x_0 x_1 (x_0 \wedge x_1)\rangle$, 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. You can assume the output qubit is always initialized to $|0\rangle$. In particular,

`OR(qc,0,1,2)` transforms $|x_0x_10\rangle$ to $|x_0x_1(x_0 \vee x_1)\rangle$.

Note that we will test values in all qubits, meaning that if you (temporarily) changed the input qubits, they need to be returned to their original value by the end of the function. Some example tests:

- 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 (5 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 $|0^n\rangle$. You can assume the output qubit is always initialized to $|0\rangle$ and $n \geq 2$. For example,

`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 do not have to use the qubits in `ancilla`, but if you do, they must be returned to the $|0^n\rangle$ state. You may use `AND()` inside `nAND()`.

Some example tests:

- 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 (5 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 $|0^n\rangle$. For example,

`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 $|0\rangle$ and $n \geq 2$. You do not have to use the qubits in `ancilla`, but if you do, they must be returned to the $|0^n\rangle$ state. You may use `OR()` inside `nOR()`.

Some example tests 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 (15 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 $|0\rangle$.

Some example tests 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 $\frac{1}{\sqrt{2}}|1011\rangle + \frac{1}{\sqrt{2}}|1000\rangle$.