Assignment 1

***Due**: Wednesday, Sep 20th, 2023 @ 5:00PM*

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

1 Introduction

Welcome to the first assignment for CPSC 447/547 (Introduction to Quantum Computing). Before getting started, please take a moment to read the course syllabus and familiarize yourself with the homework policies. Importantly, if you discussed with anyone besides the course staff about the assignment, *please list their names* in your submission.

Getting Started.

This assignment covers the basics of quantum computation, focusing on the first two weeks of classes. It has two parts, a *written portion* (52%) and a *programming portion* (48%). 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. To start,

- Create a folder for Assignment 1, e.g., `A1/`
- Download the starter files for this assignment to that folder from the course website:
  - `written.tex`
  - `A1.py`
  - `requirement_A1.py` (Do not modify)
- Write your solutions in `A1.py` (for programming tasks) and in `written.tex` or on paper (for written tasks).
- Debug and test your solution locally by running '`python3 A1.py`' on command line. This will check for any violation of the requirements and run correctness tests. Feel free to add more tests in `A1.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 `A1.py`. Gradescope will immediately show the results from running the requirement test and public test cases. If your file fails the `requirement_A1.py` check, 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 52 *points.*

2 Complex numbers

"Life is complex – it has both the real and imaginary parts." – *Unknown.* In class, we saw that a complex number α has a form: $\alpha = a_1 + a_2 i$, where $a_1, a_2 \in \mathbb{R}$ and $i = \sqrt{-1}$. In the following tasks, we will learn to work with complex numbers.

Task 2.1 (6 pts)

Given a *non-zero* complex number $\alpha = a_1 + a_2 i$, compute the following expressions. Please write your answers by specifying the real and imaginary parts.

(a) $\frac{1}{\alpha}$

(b) $e^{-i\alpha}$

(c) $\frac{1}{2i}(\alpha - \alpha^*)$

Task 2.2 ($\star$ pts)

Given a *non-zero* complex number $\alpha = a_1 + a_2 i$, compute the following expressions. Please write your answers by specifying the real and imaginary parts.

(a) $|\alpha|^2$

(b) α^2

Task 2.3 (6 pts)

Simplify the following expression. Please write your answers in the form of $x + yi$.

(a) $\left(\sin\frac{\pi}{4} + i\cos\frac{\pi}{4}\right)^3$

(b) Find all α such that $\alpha^3 = 1$.

3 State vectors

"Quantum phenomena do not occur in a Hilbert space. They occur in a laboratory." – *Asher Peres.* As we see in lectures, a quantum state for n qubits can be described as a vector in a 2^n dimensional Hilbert space. Mathematically, we use Dirac's bra-ket notation:

$$|\psi\rangle = \sum_{j\in\{0,1\}^n} \alpha_j |j\rangle$$

where $\alpha_j \in \mathbb{C}$ satisfy the normalization condition: $\sum_j |\alpha_j|^2 = 1$.

Task 3.1 (9 pts)

For the following complex vectors, are they valid quantum states? If a vector $|x\rangle$ is a valid quantum state, what is its outer product, $|x\rangle\langle x|$? If not a valid quantum state, what is its inner product, $\langle x|x\rangle$?

(a) $|x\rangle = \frac{1+i}{\sqrt{2}}|0\rangle + \frac{1-i}{\sqrt{2}}|1\rangle$

(b) $|x\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{-i}{\sqrt{2}}|1\rangle$

(c) $|x\rangle = e^{-i\pi/8}|0\rangle + e^{i\pi/8}|1\rangle$

Task 3.2 ($\star$ pts)

A single-qubit quantum state is a vector in the 2-dim Hilbert space. As we discussed in class, a basis for this vector space is a set of vectors such that they span the entire vector space and are linearly independent. An example of such basis is $\{|0\rangle, |1\rangle\}$ (which we call the computational basis). Another example is $\{|+i\rangle, |-i\rangle\}$, where $|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ and $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$. This means that any single-qubit quantum state can be written as a linear combination of the basis vectors. For each of the following quantum states:

- first compute the inner product $\langle\psi| + i\rangle$, and
- then write $\big|\psi\big\rangle$ in the $\{|+i\rangle, |-i\rangle\}$ basis, i.e., find $\alpha, \beta \in \mathbb{C}$ such that $\big|\psi\big\rangle = \alpha|+i\rangle + \beta|-i\rangle$.

(a) $\big|\psi\big\rangle = |0\rangle$

(b) $\big|\psi\big\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

4 Spectral Theorem

Eigenvalues and eigenvectors are everywhere; they play a vital role in our understanding of quantum computing. In class, we have seen that any normal operator A has a spectral decomposition:

$$A = \sum_j \lambda_j \, |j\rangle\langle j|$$

where $\big|j\big\rangle$ are orthonormal eigenvectors of A and λ_j are the corresponding eigenvalues.

Task 4.1 (16 pts)

(a) "Decomposition" – Find the orthonormal eigenvectors and their eigenvalues for the following three matrices, namely Pauli matrices, and write down their spectral decompositions.

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

(b) "Change of basis" – We normally write quantum states in their computational basis. For example, if $\big|\psi\big\rangle = \alpha|0\rangle + \beta|1\rangle$ we can write it down as a column vector $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$. But the choice of basis is arbitrary, we can also choose to write the state vector in another basis, say $\{|+i\rangle, |-i\rangle\}$.

This means that if $|\psi\rangle = \alpha'|+i\rangle + \beta'|-i\rangle$, we write $\begin{bmatrix}\alpha'\\ \beta'\end{bmatrix}_{\pm i}$. Here, we use the subscript "$\pm i$" to avoid ambiguity. Find the operator U such that it serves as a "change of basis" action from the basis $\{|+i\rangle, |-i\rangle\}$ to the basis $\{|0\rangle, |1\rangle\}$, i.e.,

$$U\begin{bmatrix}\alpha'\\ \beta'\end{bmatrix}_{\pm i} \to \begin{bmatrix}\alpha\\ \beta\end{bmatrix}$$

Hint: think eigenvectors of σ_y.

Task 4.2 ($\star$ pts)

(a) "Diagonalization" – Find an operator U such that it diagonalizes σ_y, i.e., $U\sigma_y U^{-1} = D$ where D is a diagonal matrix where the diagonal entries are the eigenvalues of σ_y. Assume the (real) diagonal entries are sorted in a decreasing order from top-left to bottom right.

(b) "Powers of operator" – as we saw in lectures, we can use the Spectral theorem to raise an operator A to some power. Compute the fourth root(s) of the Pauli Y matrix, $M = \sigma_y^{1/4}$. I.e., find all possible M, such that $M^4 = \sigma_y$.

5 Distinguishing quantum states

In class, we learned that we can distinguish two quantum states via physical experiments, such as applying quantum gates and measurements.

Carol is an experimental physicist. One afternoon, she prepares two qubits (i.e., photon A and photon B) in certain state $|\psi_{AB}\rangle$ and gives qubit A to Alice and qubit B to Bob. Out of curiosity, Alice and Bob want to figure out if the qubits they receive are the same or different. Since they only received one copy of the qubits each, they cannot simply measure multiple copies of their qubits and reconstruct the distribution of the measurement outcomes. So they turn to you for help.

Task 5.1 (15 pts)

In each of the following scenario, Carol has told you the state of the qubits; your job is to tell Alice and Bob what quantum gates they can perform, so that a computational basis measurement will distinguish the two states *with certainty*. If this is not possible, explain why. (Note: Alice and Bob need to perform the *exact same* set of gates on their own qubit, if any. No interaction between the two qubits is allowed.)

(a) $|\psi_{AB}\rangle = |+\rangle \otimes |-\rangle$

(b) $|\psi_{AB}\rangle = \left(\frac{|0\rangle + e^{-i\pi/4}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + e^{3i\pi/4}|1\rangle}{\sqrt{2}}\right)$

(c) $|\psi_{AB}\rangle = \left(\frac{|0\rangle + e^{-i\pi/4}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{e^{i\pi/8}|0\rangle + e^{-i\pi/8}|1\rangle}{\sqrt{2}}\right)$

Task 5.2 ($\star$ pts)

When qubits are entangled, your job again is to tell Alice and Bob what quantum gates they can perform, so that a computational basis measurement outcome will be different for them. If this is not possible, explain why.

(a) $\left|\psi_{AB}\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

(b) $\left|\psi_{AB}\right\rangle = \frac{1}{\sqrt{2}}(|+\rangle \otimes |+\rangle + |-\rangle \otimes |-\rangle)$

PROGRAMMING PORTION

This portion of the assignment has a total of 48 *points.*

6 Circuit Simulator

Please read the instructions for how to get started with the programming tasks in Section 1. The goal of these tasks is to understand the basic data structures and operations used in quantum circuits. In the end, we will have a working in-house quantum circuit simulator.

Before attempting the following tasks, you may want to read the starter code in `A1.py` carefully. We have provided a simple implementation of the `Qubit` class, the `Gate` class and the `QuantumCircuit` class.

Task 6.1 (24 pts)

Implement the instruction set for the `QuantumCircuit` class. The implementation of the `h` gate and the `measure` operation have been given. For example, in `A1.py`, you can find the definition of the Hadamard gate:

```
def h(self, qubit):
    # Define Gate by its name, kind (number of qubit), and matrix
    HGate = Gate('h',1,1/np.sqrt(2)*np.array([[1,1],[1,-1]],dtype=complex))
    self._append(HGate, [qubit], [])
    return
```

Notice that the `_append` function takes three arguments: a `Gate` object, qubit indices for the gate, and classical bit indices. For the purpose of this assignment, you can assume the last argument is always `[]`. For example, we can create a quantum circuit with a Hadamard gate on qubit index 0 by running the following lines:

```
qc = QuantumCircuit(1, 1)
qc.h(0)
```

Task 6.2 (24 pts)

Implement the `tensorizeGate` function. It takes as input a quantum gate (excluding measurement) defined on one, two or three qubits, which is a subset of the `QuantumRegister` in

the `QuantumCircuit`. The function returns the full unitary matrix (i.e., acting on all qubit in `QuantumRegister`) for the input gate. Note that we assume the following ordering of qubits: $|q_0 q_1 q_2 \dots q_n\rangle$. For example, if the `QuantumRegister` has 3 qubits and the quantum gate `cx` acts on qubits `q2` (as control) and `q0` (as target), the function `tensorizeGate` will return a $2^3 \times 2^3$ matrix (as an `np.array`), with identity on the remaining qubit.

```
qc = QuantumCircuit(3, 3)
qc.cx(2, 0)
(op, q_arr, _) = qc.circuit[qc.pc]
tensorized = qc.tensorizeGate(op, q_arr)
```

The above lines of code will result in the following matrix:

$$\texttt{tensorized} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Task 6.3 ($\star$ pts)

Implement the `simulate` function. This function simulates the quantum circuit from scratch, returning a final quantum state vector (as an `np.array` of `complex` type) if there are no measurements in the circuit, *or* a sampled classical state vector (as an `np.array` of `bool` type) otherwise. Please use the provided `sampleBit` function for random sampling. You may use/modify the `evolveOneStep` function as a helper function for `simulate`. You may assume that all measurements are the last few elements in `circuit` if any. For this assignment, you may also assume that (computational basis) measurement (if present) will be performed to all qubits.