Assignment I

Introduction to Quantum Computing

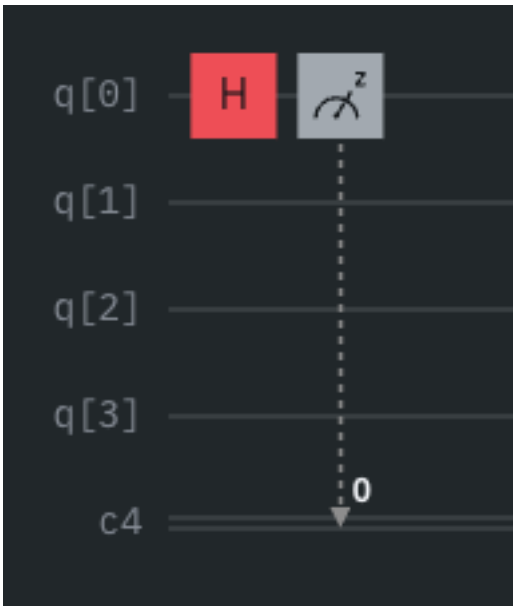
Pupipat Singkhorn
Student ID: 6532142421

1. Quantum Computing Account

Please go to IBM Quantum Experience and register an account, https://quantumcomputing.ibm.com/.

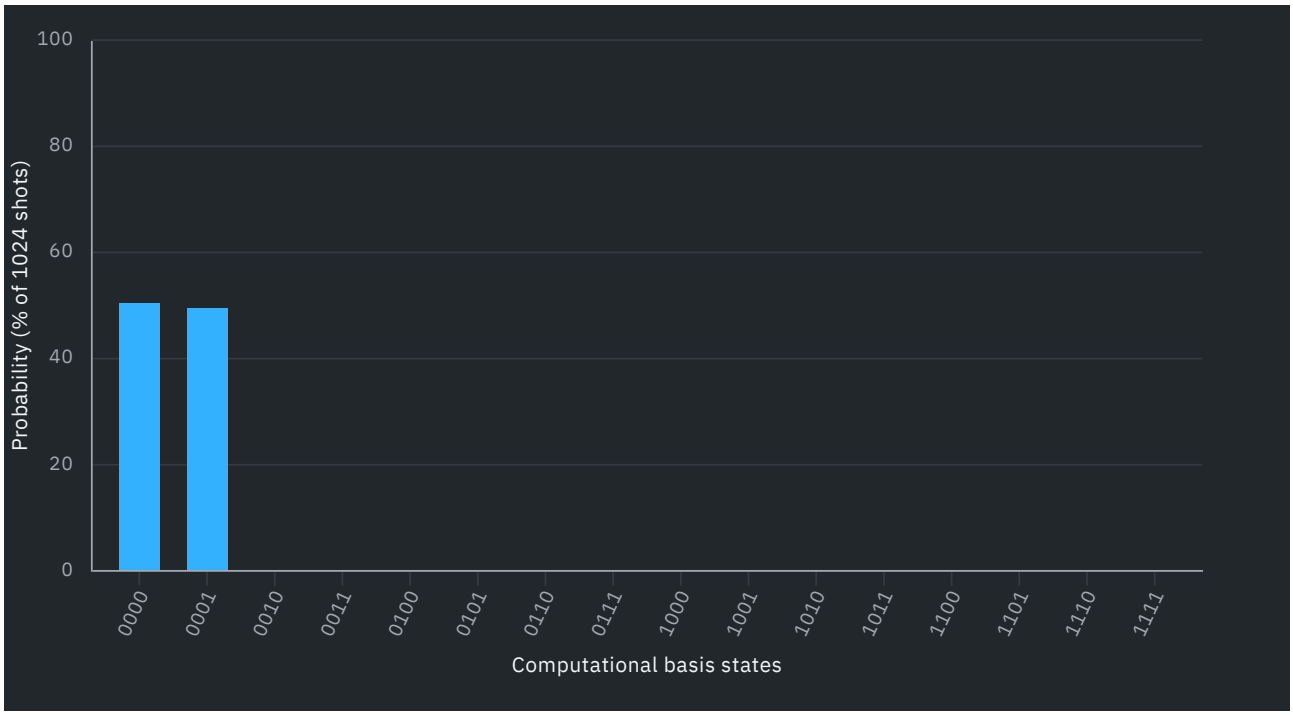
2. My First Quantum Circuit

Try to create a circuit. Click on IBM Quantum Composer and drag the gates to form the following circuit. The first icon is a quantum gate called Hadamard gate and the second icon is a measurement. The input is from the left (q0). Do not worry if you do not understand what it is at this moment. But good job! You have created your first quantum circuit!



3. My First Quantum Computing

Submit the job. Click on the result when it is finished. You see some like this. You may submit to a simulator or real quantum hardware. We started with a state in $|0\rangle$(a basis state). After the Hadamard gate, it becomes a superposition of $|0\rangle$ and $|1\rangle$, i.e. ~ $|0\rangle + |1\rangle$ (we will learn later). Then we performed a measurement on the output and the superposition state collapses to either $|0\rangle$ or $|1\rangle$. The histogram shows that about 50% of the chance we obtain $|0\rangle$ and another 50% of the chance we obtain $|1\rangle$. Cool! We have done our first quantum computation. This job was submitted to a real quantum computer, called "ibmq_casablanca." It is a 5-qubit quantum computer. However, this computer is gone. But definitely, you will find other more powerful ones.



4. Google Colab and Python

We need to do a lot of matrix operations. We will choose Google Colab, in which we can run Python easily. Since IBM Quantum Experience also uses Python, it will save your efforts. Please open a Google Colab account here https://colab. research.google.com/.

5. Vector Inner Product Using Python

We use the NumPy library to perform vector and matrix operations. To import the library, type import numpy as np Then let's implement $\vec{v}$ 1 . $\vec{v}$ 2 = (√3 1)(1 1) = √3 × 1 + 1 × 1 = √3 + 1 ≈ 2.732 v1=np.array([[np.sqrt(3)],[1]]) v2=np.array([[1],[1]]) inner_result=np.vdot(v1,v2) print(inner_result) Type Shift–Enter to let it run. Do you see the same result? Note that we use np.vdot instead of np.dot because we need to use complex vector dot products in quantum computing. Try also print (v1.shape), which will give you (2, 1). It means we constructed a 2 × 1 matrix, which is a column vector with 2 elements.

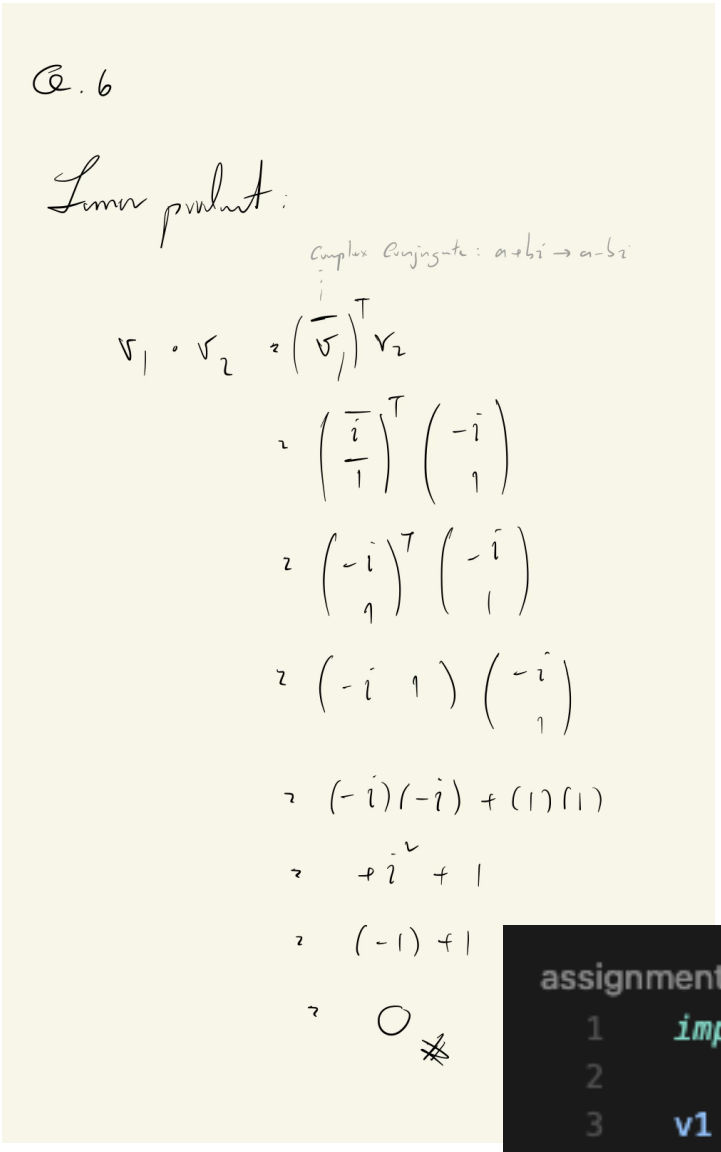
```
import numpy as np

v1 = np.array([[np.sqrt(3)], [1]])
print("v1.shape:", v1.shape)
v2 = np.array([[1], [1]])
print("v2.shape:", v2.shape)
inner_result = np.vdot(v1, v2)
print("inner_result:", inner_result)
```

```
(quantum-computing) (base) pupipatsingkhorn@Pupipats-
MacBook-Air quantum-computing %
/Users/pupipatsingkhorn/Developer/repositories/quantum
computing/.venv/bin/python
/Users/pupipatsingkhorn/Developer/repositories/quantum
computing/assignments-1/q5.py
v1.shape: (2, 1)
v2.shape: (2, 1)
inner_result: 2.732050807568877
```

6. Inner Product with Complex Coefficient 1

Now, let's try $\vec{v}\ 1 = (\ i\ 1\ )$ and $\vec{v}\ 2 = (\ -i\ 1\ )$. Firstly, try to use hand calculation. Then use Python to check if it is correct. Note that in the NumPy library, you need to put i as j. For example, 5 + i should be entered as 5 + 1j (putting the "1" is necessary). Think about that before reading the code below: v1=np.array([[1j],[1]]) v2=np.array([[-1j],[1]]) inner_result=np.vdot(v1,v2) print(inner_result)



¶

```
assignments-1 > q6.py > ...
1   import numpy as np
2
3   v1 = np.array([[1j], [1]])
4   v2 = np.array([[-1j], [1]])
5   inner_result = np.vdot(v1, v2)
6   print(inner_result)
7
```

```
Problems  Output  Debug Console  Termi
(quantum-computing) (base) pupipatsingkh
/pupipatsingkhorn/Developer/repositories
0j
(quantum-computing) (base) pupipatsingkh
```

7. Inner Product with Complex Coefficient 2

Now, let's try $\vec{v}\ 1 = (\ 1+2i\ 1\ )$ and $\vec{v}\ 2 = (\ 1-i\ 1\ )$.

Q.7

Inner product: $v_1 = \begin{pmatrix} 1+2i \\ 1 \end{pmatrix}$, $v_2 = \begin{pmatrix} 1-i \\ 1 \end{pmatrix}$

$$v_1 \cdot v_2 = (\bar{v}_1)^T v_2 = \begin{pmatrix} 1-2i & 1 \end{pmatrix} \begin{pmatrix} 1-i \\ 1 \end{pmatrix}$$

$$= (1-2i)(1-i) + (1)(1)$$

$$= 1 - i - 2i + 2i^2 + 1$$

$$= 2 - 3i - 2$$

$$= -3i \quad \#$$

```
q7.py U
assignments-1 > q7.py > ...
1   import numpy as np
2
3   v1 = np.array([[1 + 2j], [1]])
4   v2 = np.array([[1 - 1j], [1]])
5   inner_result = np.vdot(v1, v2)
6   print(inner_result)
7

Problems  Output  Debug Console  Termina
(quantum-computing) (base) pupipatsingkho
/pupipatsingkhorn/Developer/repositories/
-3j
(quantum-computing) (base) pupipatsingkho
```