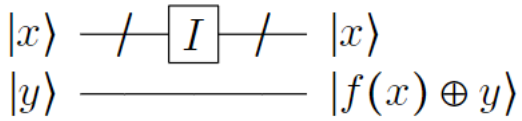
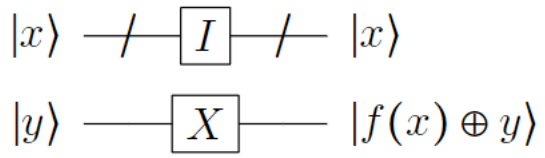
通信專題　　Lab_3　　B08901019 吳冠緯

1. **(20 points) The Deutsch-Jozsa Algorithm**
   (a) (10 points)



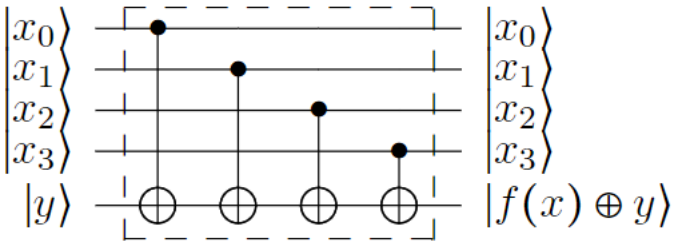
x0x1x2x3	f(x)⊕y	x0x1x2x3	f(x)⊕y
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	0	1110	0
0111	0	1111	0

Constant function



x0x1x2x3	f(x)⊕y	x0x1x2x3	f(x)⊕y
0000	1	1000	1
0001	1	1001	1
0010	1	1010	1
0011	1	1011	1
0100	1	1100	1
0101	1	1101	1
0110	1	1110	1
0111	1	1111	1

Constant function

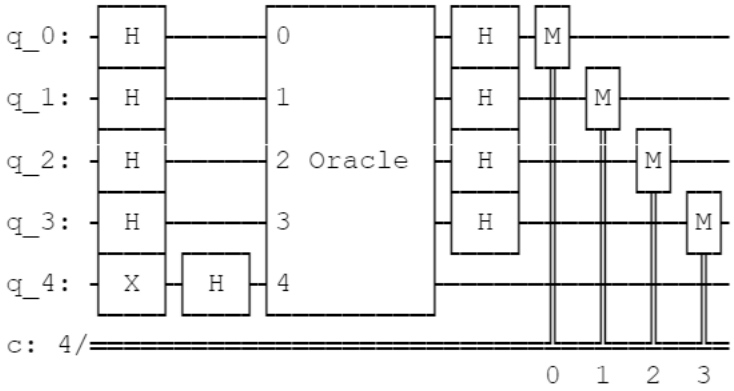


x0x1x2x3	f(x)⊕y	x0x1x2x3	f(x)⊕y
0000	0	1000	1
0001	1	1001	0
0010	1	1010	0
0011	0	1011	1
0100	1	1100	0
0101	0	1101	1
0110	0	1110	1
0111	1	1111	0

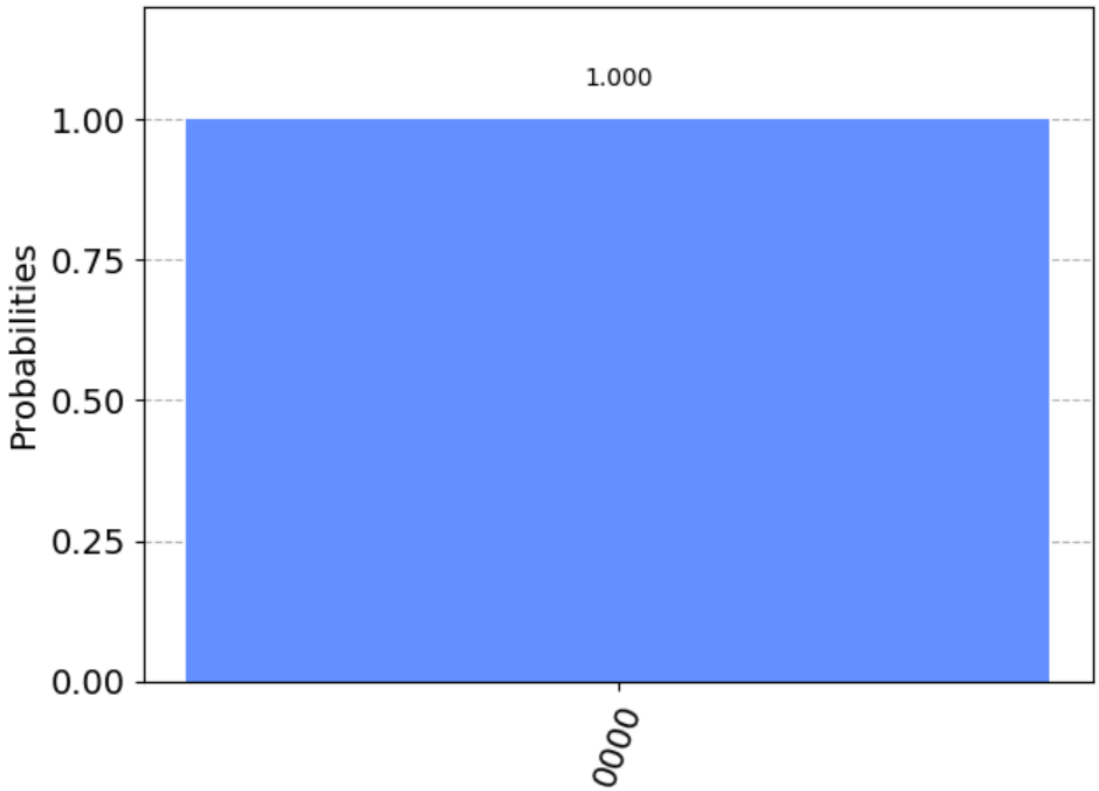
Balanced function

(b)(15 points)

(use '1a_1' for 'dj_oracle')

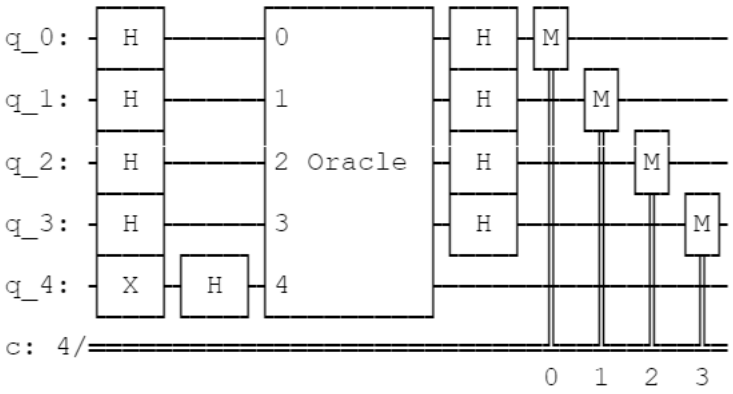


Oracle content: I-gate on q_0 ~ q_3

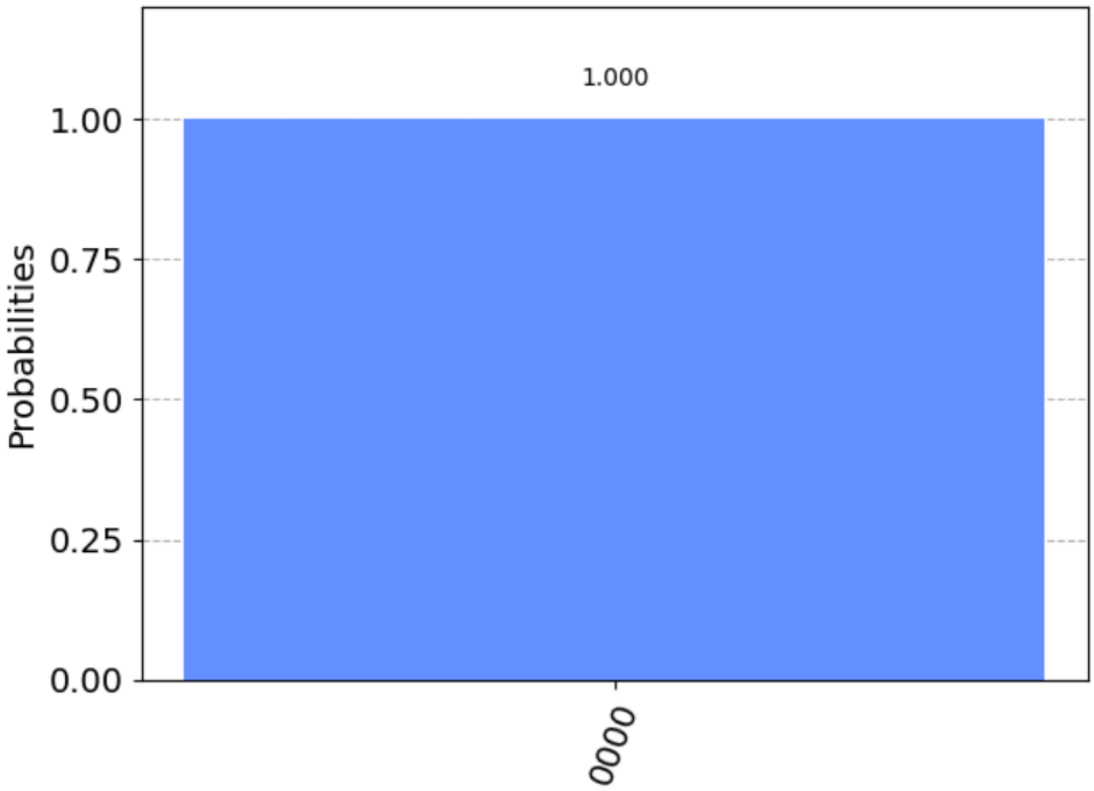


Constant function all 0

(use ‘1a_2’ for ‘dj_oracle’)

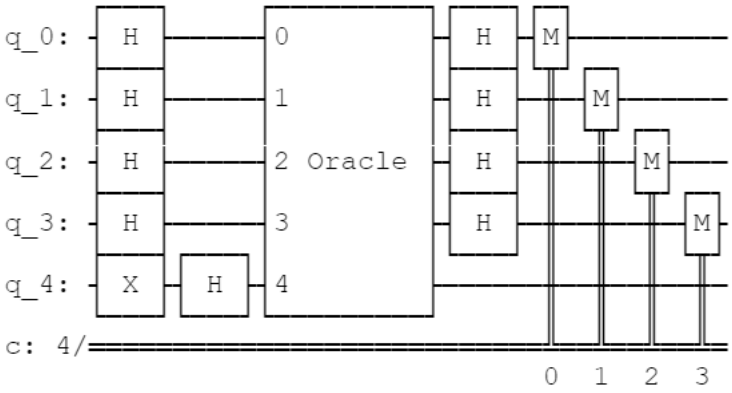


Oracle content: I-gate on q_0 ~ q_3；X-gate on q_4

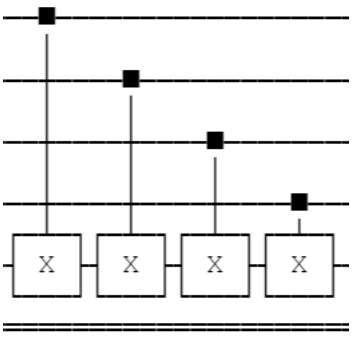


Constant function all 0

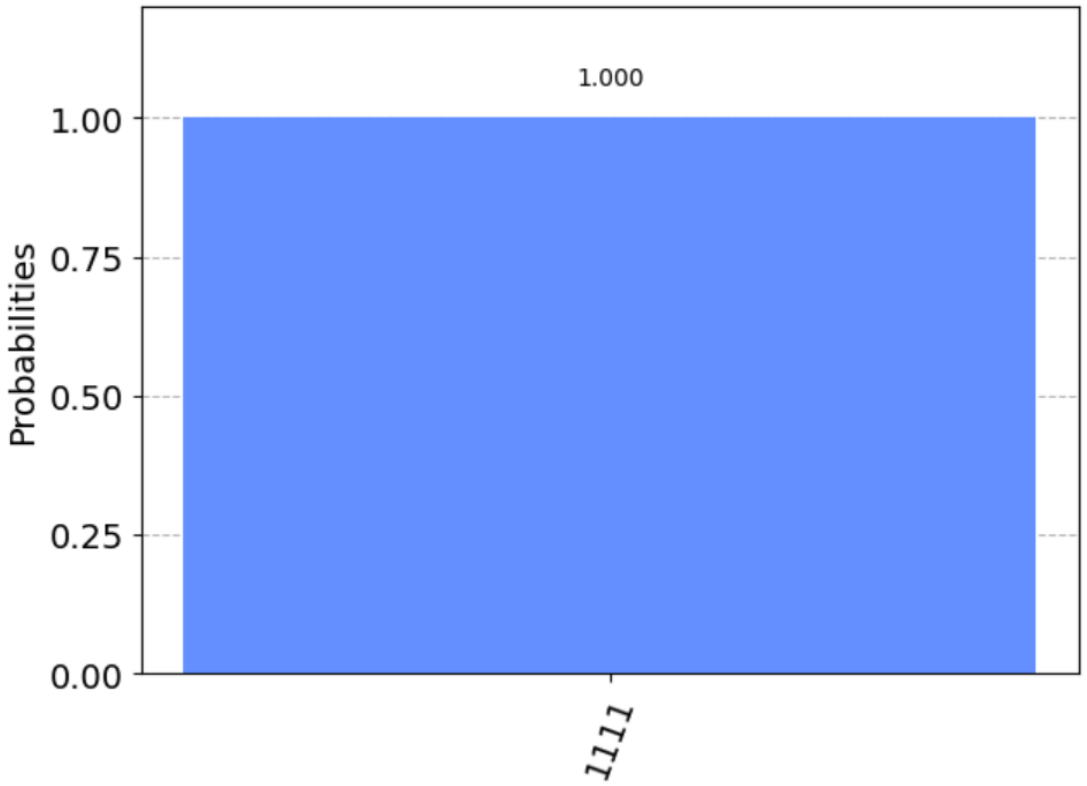
(use ‘1a_3’ for ‘dj_oracle’)



Oracle content:



```
{'1111': 1024}
```

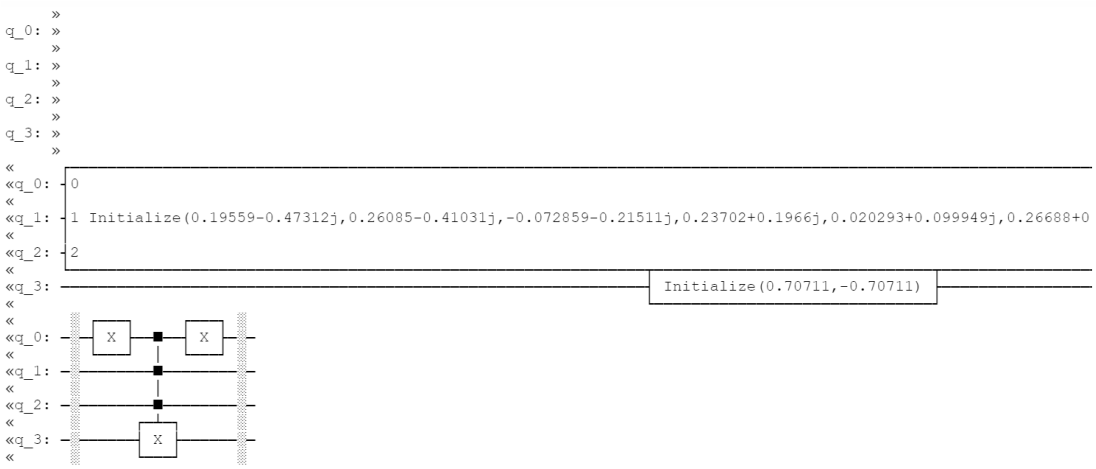


Balanced function not all 0

2. (40 points) Grover's Search

(a)

```
Statevector([ 0.13830436-0.334547j  ,  0.18445016-0.29013411j,
             -0.05151938-0.15210748j,  0.16759779+0.13901827j,
              0.01434899+0.07067449j,  0.18871571+0.15666858j,
              0.32065336+0.02128682j,  0.01866285-0.09254763j,
             -0.13830436+0.334547j  , -0.18445016+0.29013411j,
              0.05151938+0.15210748j, -0.16759779-0.13901827j,
             -0.01434899-0.07067449j, -0.18871571-0.15666858j,
             -0.32065336-0.02128682j, -0.01866285+0.09254763j],
            dims=(2, 2, 2, 2))
```

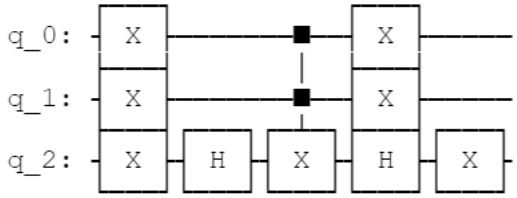



```
Statevector([ 0.13830436-0.334547j   ,  0.18445016-0.29013411j,
             -0.05151938-0.15210748j,  0.16759779+0.13901827j,
              0.01434899+0.07067449j,  0.18871571+0.15666858j,
             -0.32065336-0.02128682j,  0.01866285-0.09254763j,
             -0.13830436+0.334547j   , -0.18445016+0.29013411j,
              0.05151938+0.15210748j, -0.16759779-0.13901827j,
             -0.01434899-0.07067449j, -0.18871571-0.15666858j,
              0.32065336+0.02128682j, -0.01866285+0.09254763j],
            dims=(2, 2, 2, 2))
```

0110

1110

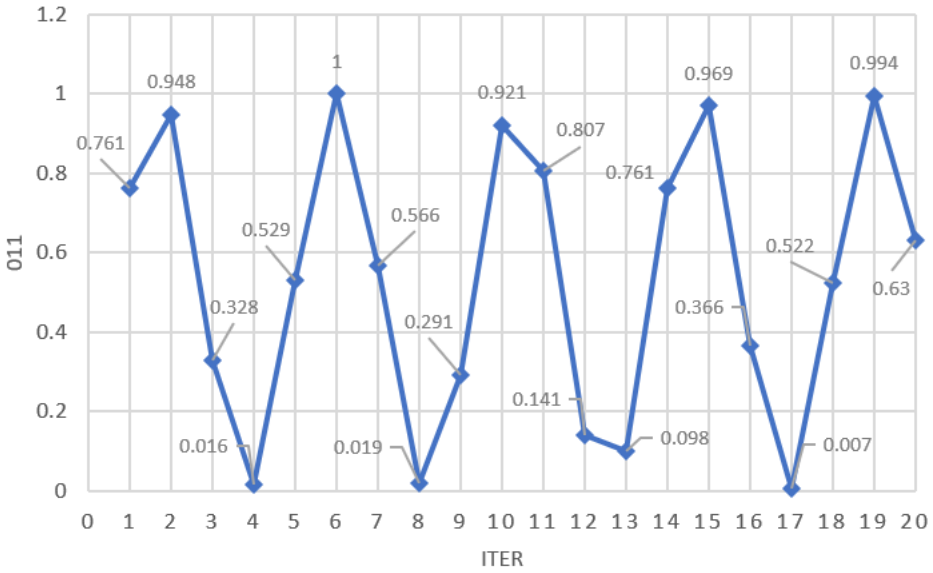
(b)



For 3-qubit version

In the front and the end, remember H-gate

(c)

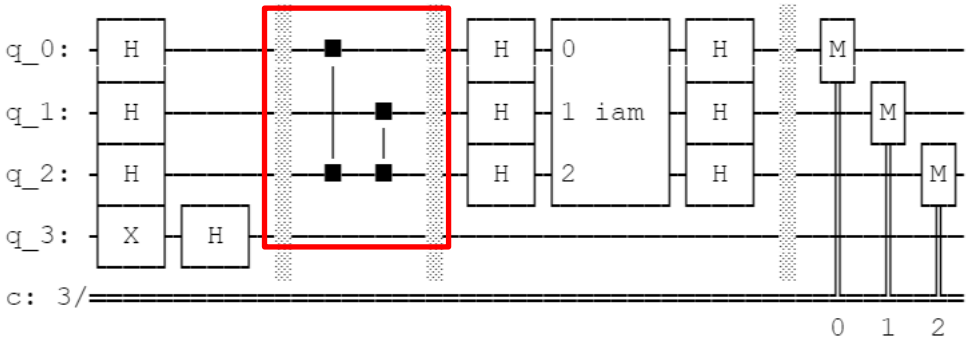


8^(1/2) = 2….

After 2 iter, get the acc of 0.948, not always right

(d)

```
Statevector([ 0.45673475-0.23247405j, -0.17471694-0.03892088j,
              0.1417575 +0.24842453j,  0.37406777+0.03859122j,
              0.33060483-0.34510371j,  0.20371487-0.11982149j,
             -0.2925818 +0.30906573j, -0.12864081-0.01245685j],
            dims=(2, 2, 2))
```



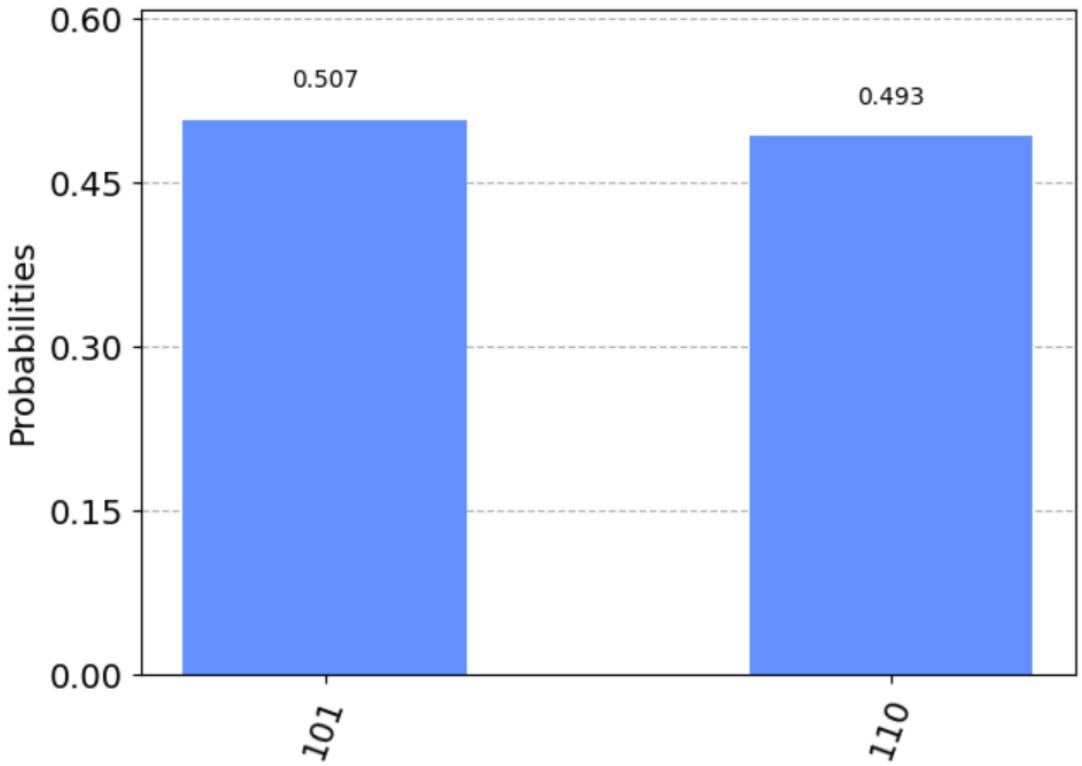

```
Statevector([ 0.45673475-0.23247405j, -0.17471694-0.03892088j,
              0.1417575 +0.24842453j,  0.37406777+0.03859122j,
              0.33060483-0.34510371j, -0.20371487+0.11982149j,
             -0.2925818 -0.30906573j, -0.12864081-0.01245685j],
            dims=(2, 2, 2))
```

110

101

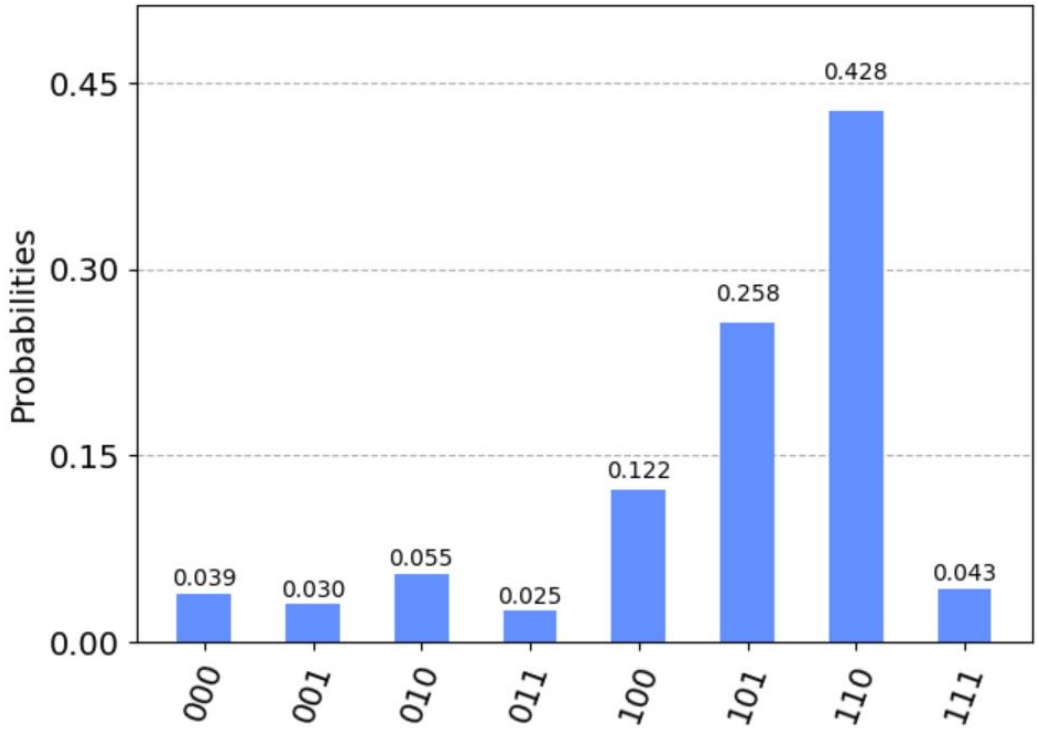
(e)

```
{'110': 505, '101': 519}
```



One query is enough

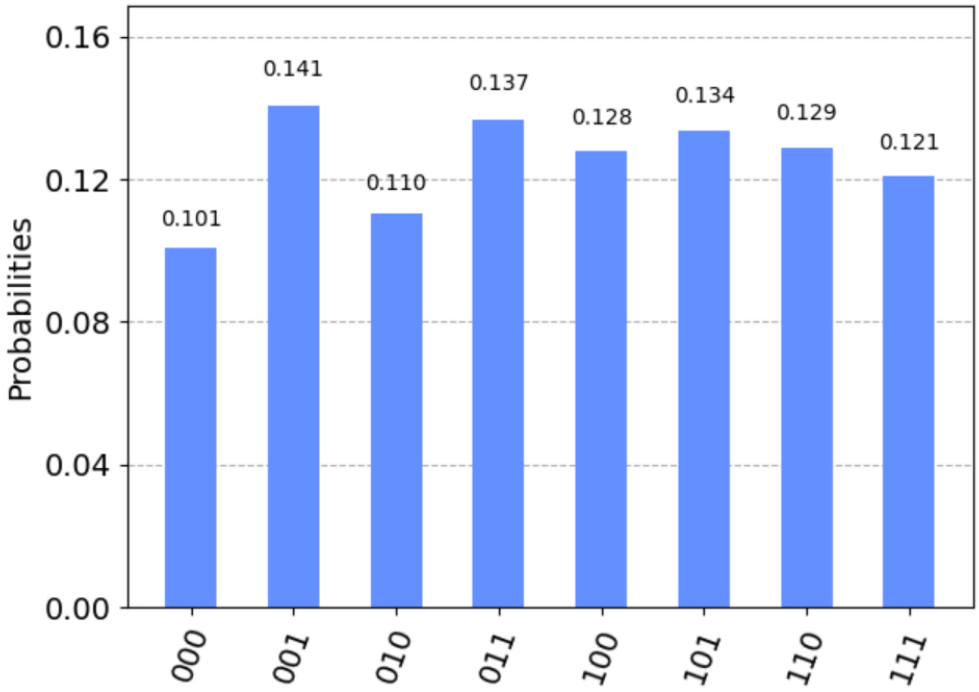
```
Job Status: job has successfully run
{'000': 40, '001': 31, '010': 56, '011': 26, '100': 125, '101': 264, '110': 438, '111': 44}
```



Drop to 0.686 using real device

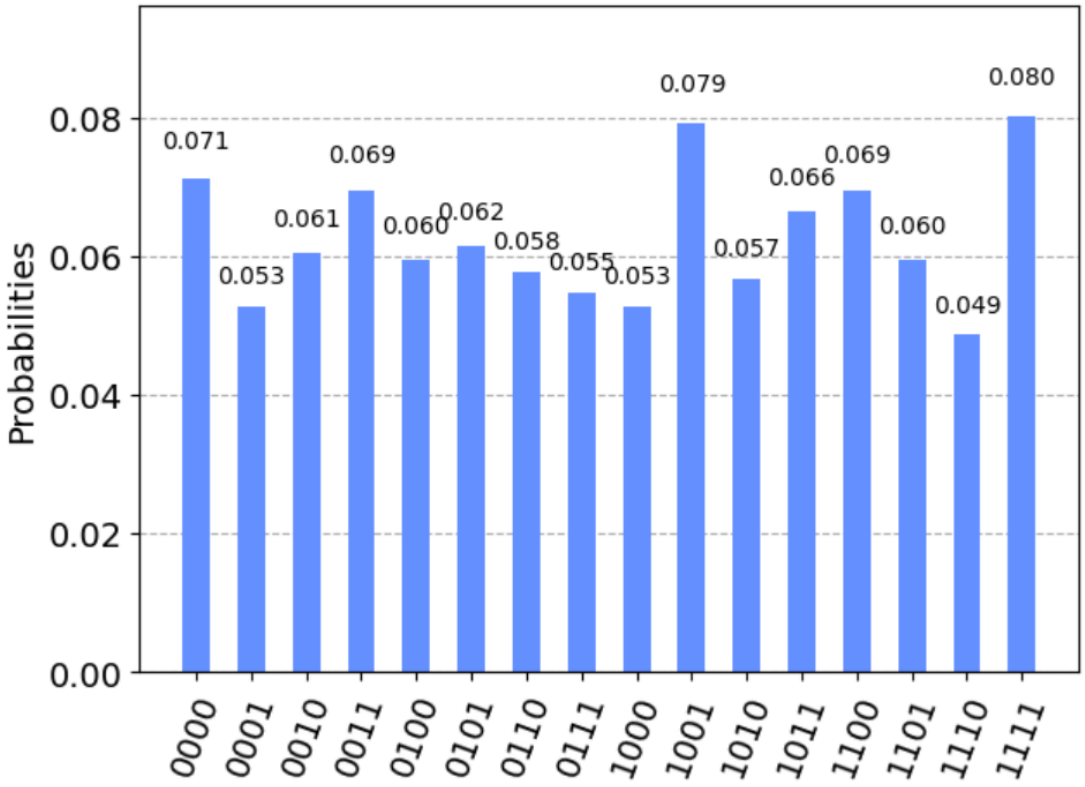
(f)

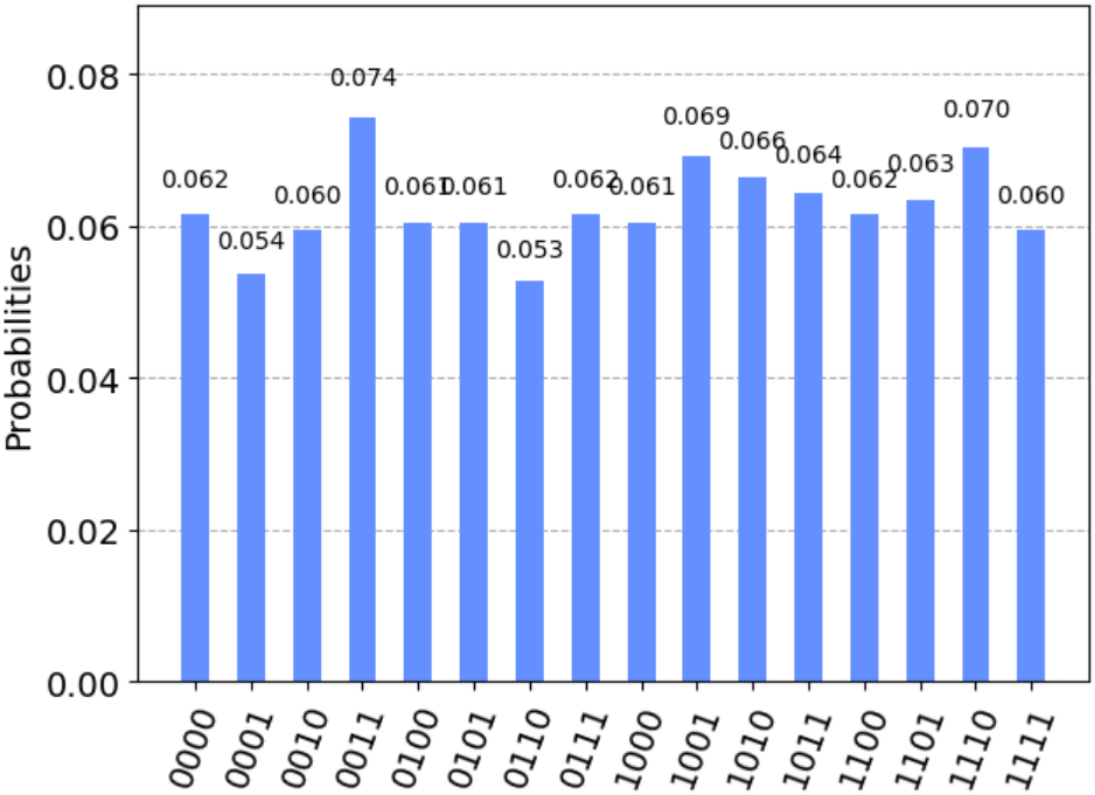
{'011': 140, '100': 131, '101': 137, '010': 113, '111': 124, '110': 132, '000': 103, '001': 144}

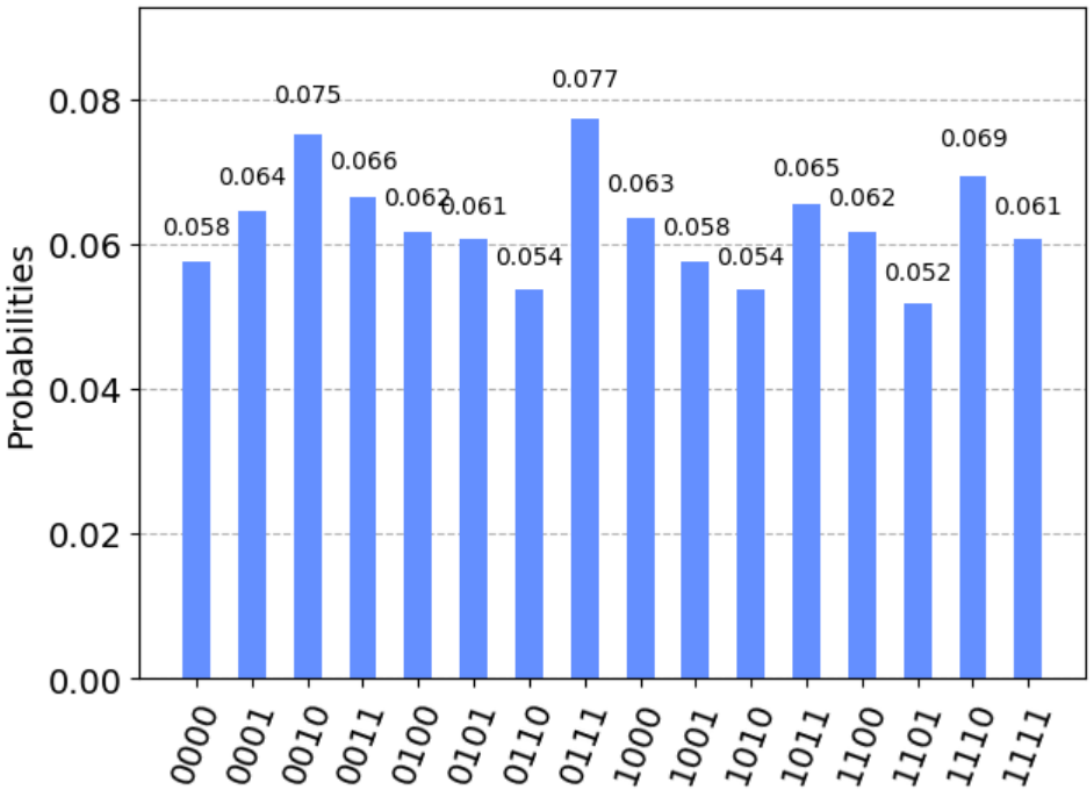


Every iteration shows the same results, can't tell the different from two groups' amplitude.

3. (15 points) Quantum Fourier Transform

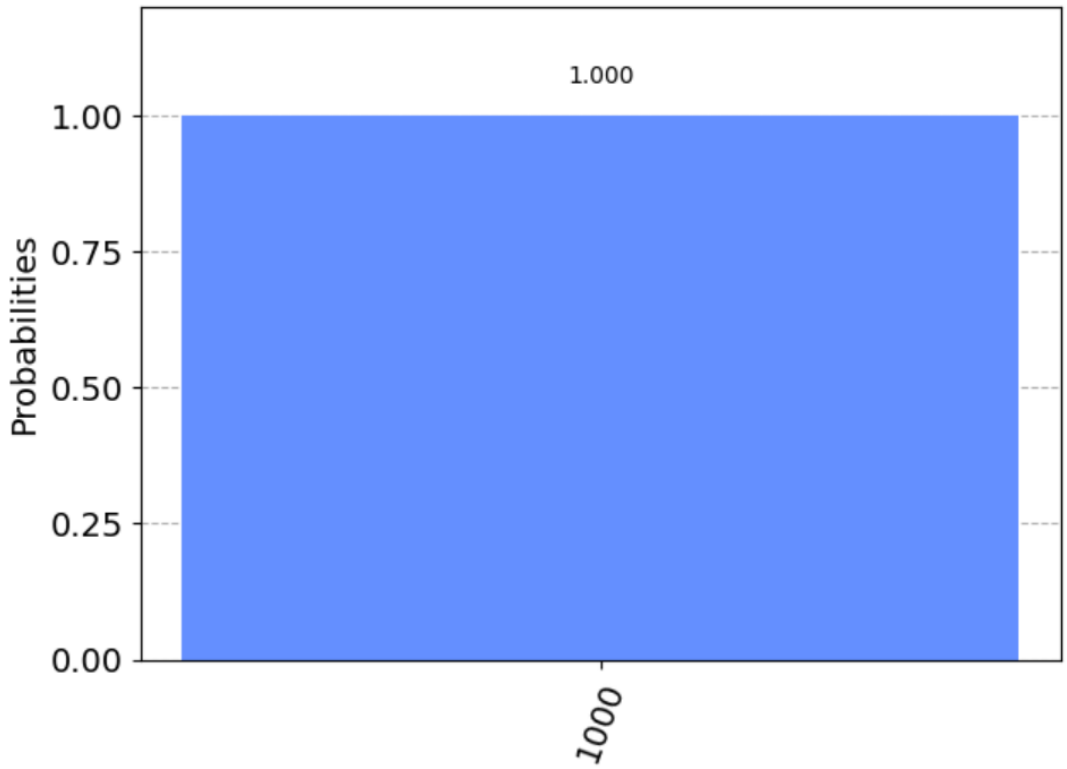

Probabilities
0.00
0.02
0.04
0.06
0.08
0.071
0.053
0.061
0.069
0.060
0.062
0.058
0.055
0.053
0.079
0.057
0.066
0.069
0.060
0.049
0.080
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

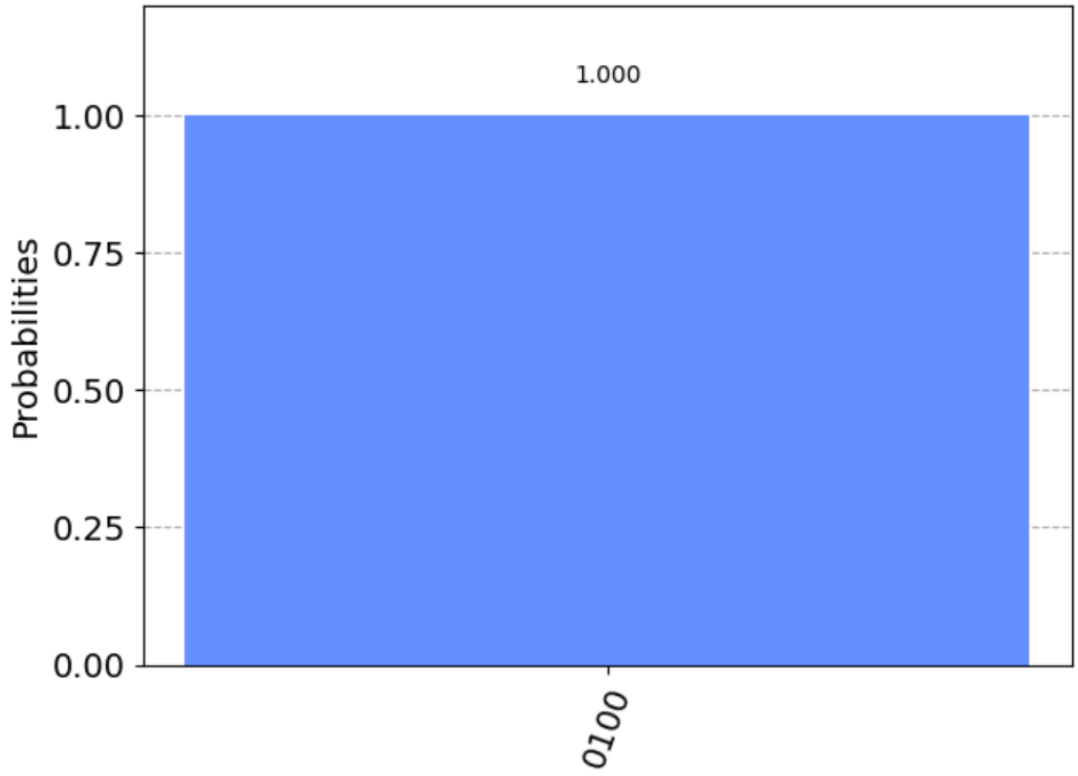

Probabilities
0.00
0.02
0.04
0.06
0.08
0.062
0.054
0.060
0.074
0.061
0.061
0.053
0.062
0.061
0.069
0.066
0.064
0.062
0.063
0.070
0.060
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

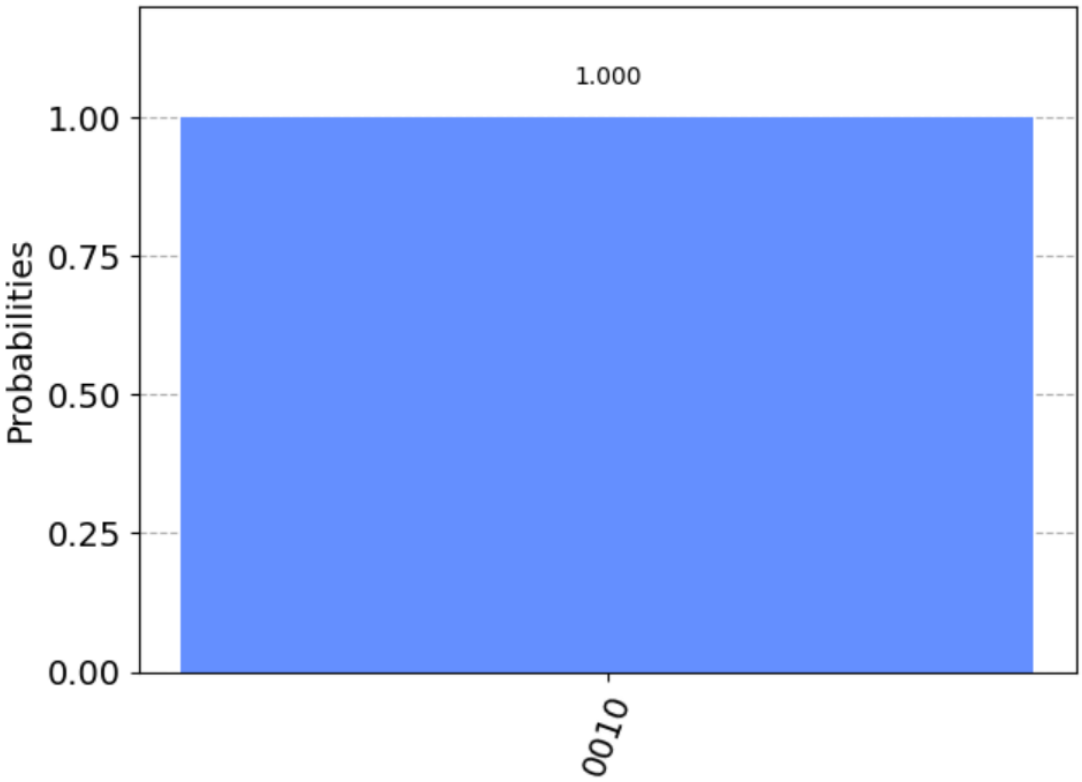


Three graphs look the same

After QFT







Here, numbers using different rotations around the Z-axis are stored

1000 = 8 and the four qubits rotate to 2pi/16 * 8 = pi 、2pi/8 * 8 =

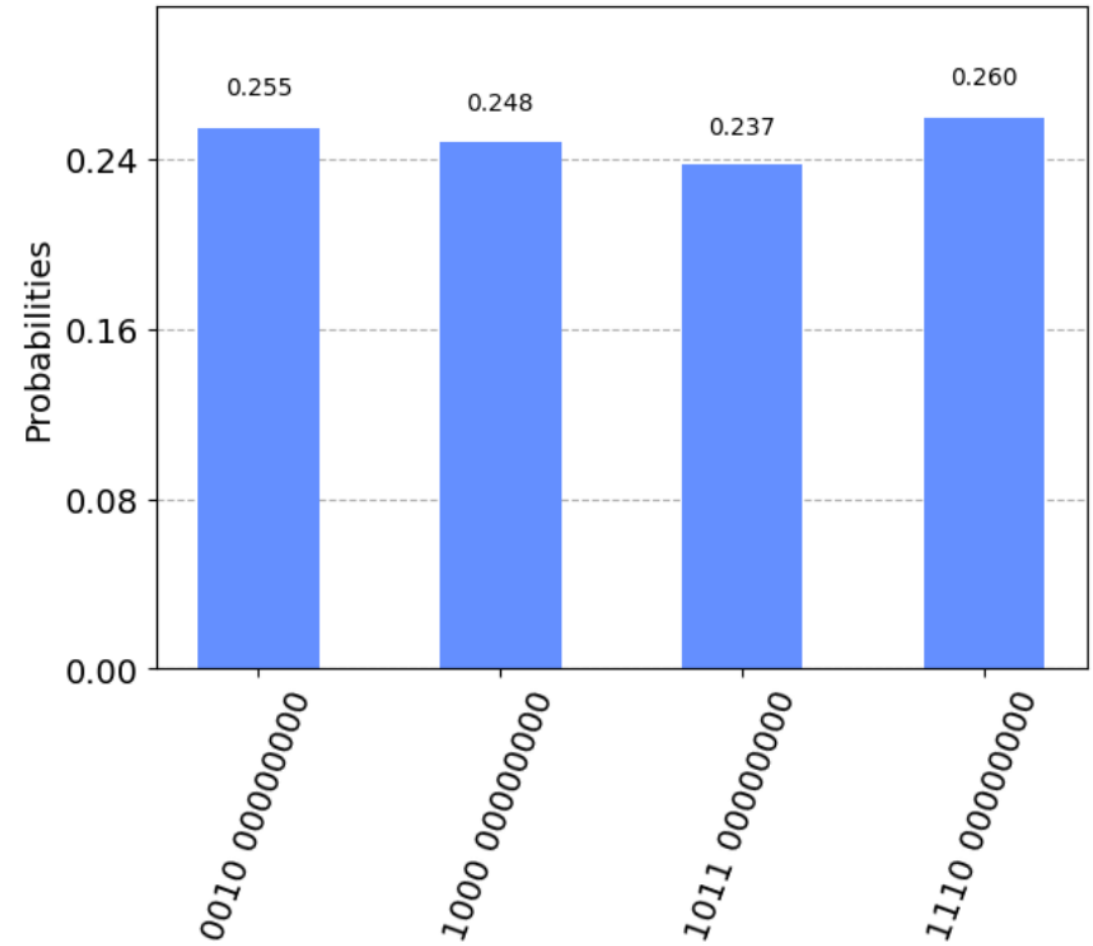
2pi = 0、2pi/4 * 8 = 4pi = 0、2pi/2 * 8 = 8pi = 0 → {Rz(pi), x, x, x}

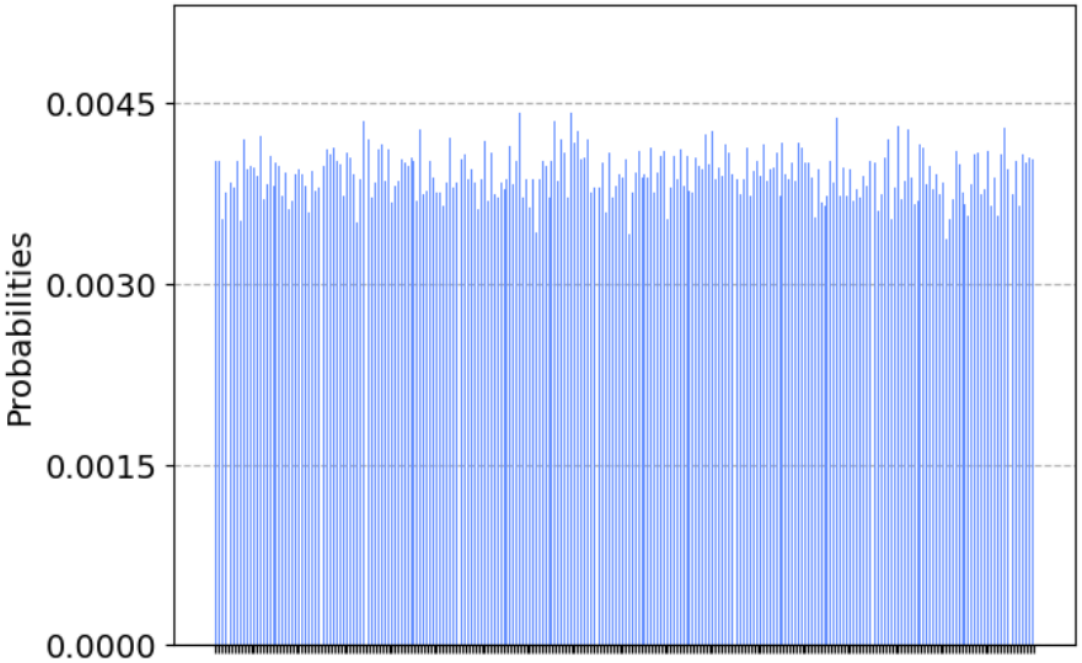
0100 = 4 and the four qubits rotate to 2pi/16 * 4 = pi/2、2pi/8 * 4 = pi、2pi/4 * 4 = 2pi = 0、2pi/2 * 4 = 4pi = 0 → {Rz(pi/2), Ry(pi), x, x}
(Doing Ry(pi) is same as doing Rz(pi) on '+')

0010 = 2 and the four qubits rotate to 2pi/16 * 2 = pi/4、2pi/8 * 2 = pi/2、2pi/4 * 2 = pi、2pi/2 * 2 = 2pi = 0 → {Rz(pi/4), Rz(pi/2), Rz(pi), x}

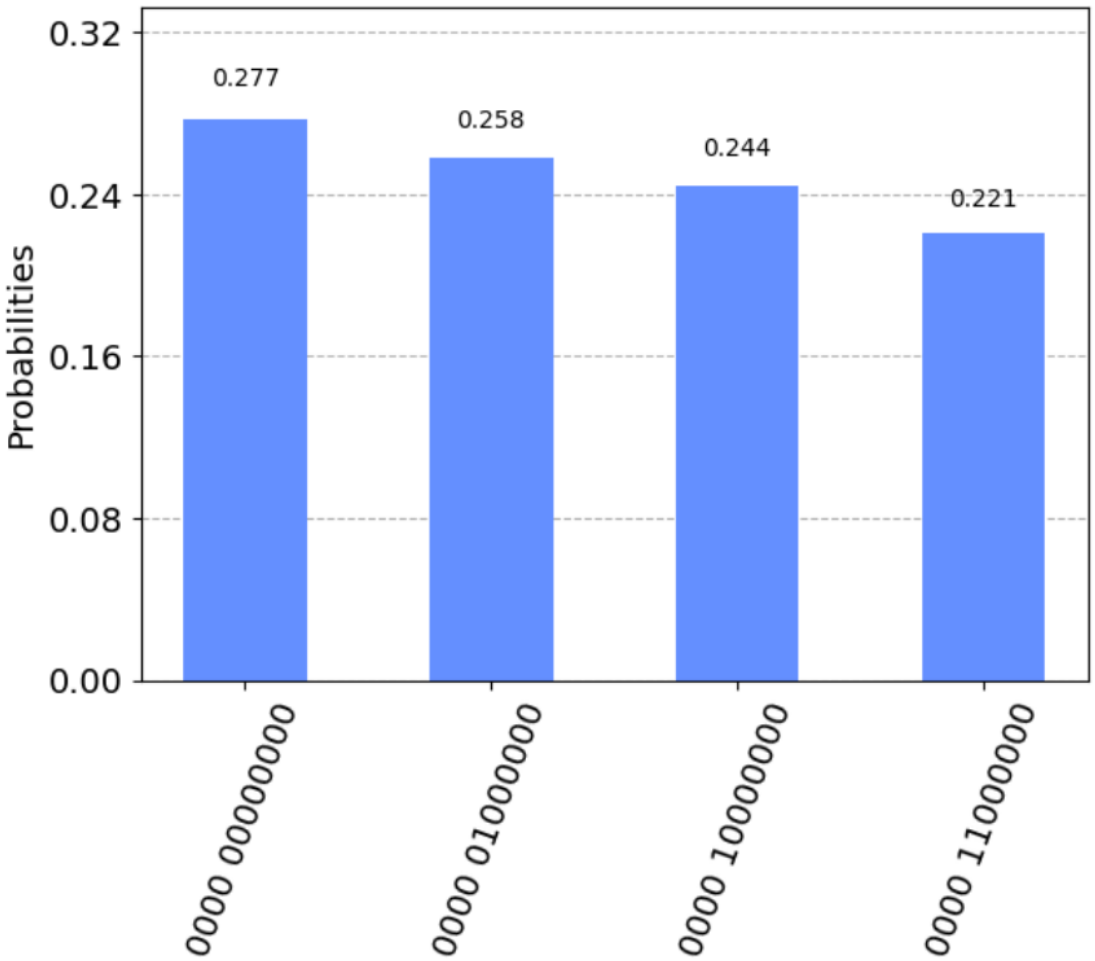
4. **(25 points) Period-Finding Algorithm**
   7, 4, 13, 1, 7, 4, 13, 1, 7, … (7mod15, 49mod15, 343mod15, …)

   Qr2, probability for 1, 4, 7, 13 is the same because it's periodic





Periodic with r, random selection and become uniform in the probability.



0, 64, 128, 192 (64 * {0, 1, 2, 3})

a=7 r=4

	Phase	Fraction	Guess for r
0	0.00	0/1	1
1	0.75	3/4	4
2	0.25	1/4	4
3	0.50	1/2	2

a=2 r=4 (2, 4, 8, 1)

	Phase	Fraction	Guess for r
0	0.00	0/1	1
1	0.50	1/2	2
2	0.25	1/4	4
3	0.75	3/4	4

a=8 r=4 (8, 4, 2, 12)

	Phase	Fraction	Guess for r
0	0.00	0/1	1
1	0.75	3/4	4
2	0.50	1/2	2
3	0.25	1/4	4

a=11 r=2 (11, 1)

	Phase	Fraction	Guess for r
0	0.0	0/1	1
1	0.5	1/2	2

a=13 r=4 (13, 4, 7, 1)

	Phase	Fraction	Guess for r
0	0.00	0/1	1
1	0.75	3/4	4
2	0.25	1/4	4
3	0.50	1/2	2

All the above results have a 0.5 probability