

# Kawasaki Quantum Summer Camp 2024

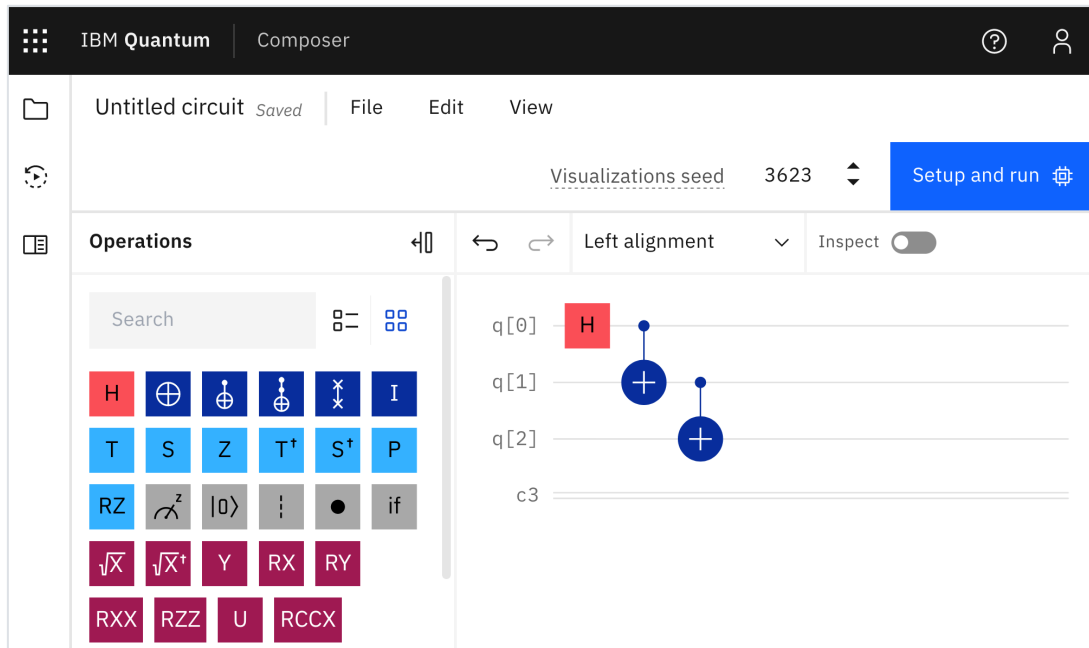
## 量子ゲート基礎 IBM Quantum Composer

Jul 29, 2024

沼田 祈史

Kifumi Numata

IBM Quantum



いつも使っている  
コンピューターのビット

0 または 1

どちらか

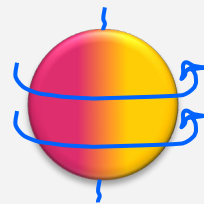


量子コンピューターの  
量子ビット

0 と 1

両方

くるくる回っているコイン (イメージ)



測定すると表か裏にバシッと決まる

いつも使っている  
コンピューターのビット

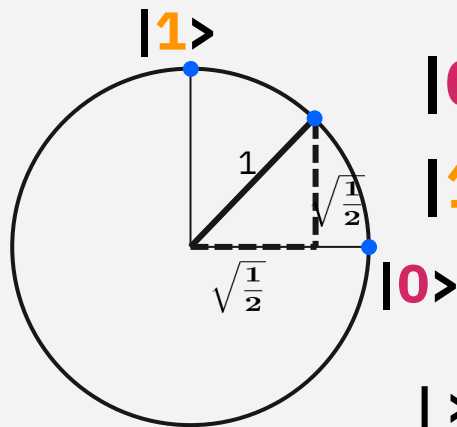
0 または 1

どちらか

量子コンピューターの  
量子ビット

$$\alpha \times |0\rangle + \beta \times |1\rangle$$

0 と 1 の「重ね合わせ」

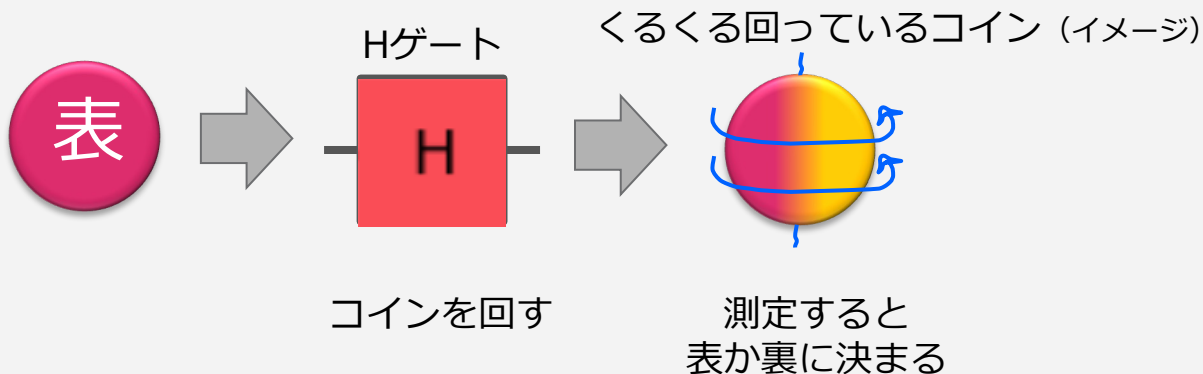
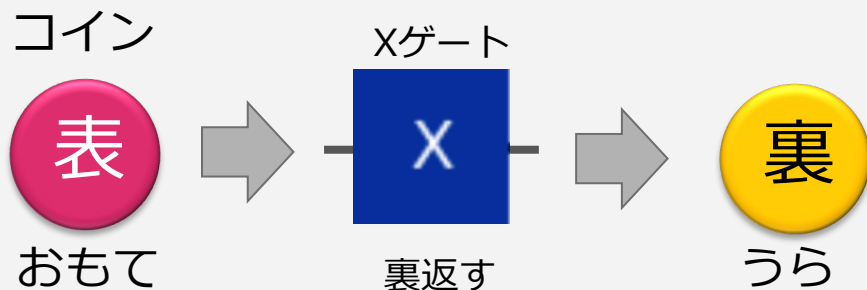


$$|0\rangle = 1 \times |0\rangle + 0 \times |1\rangle$$

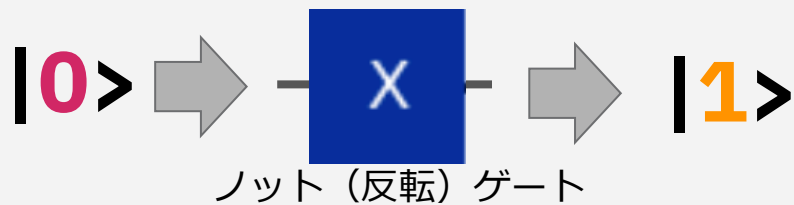
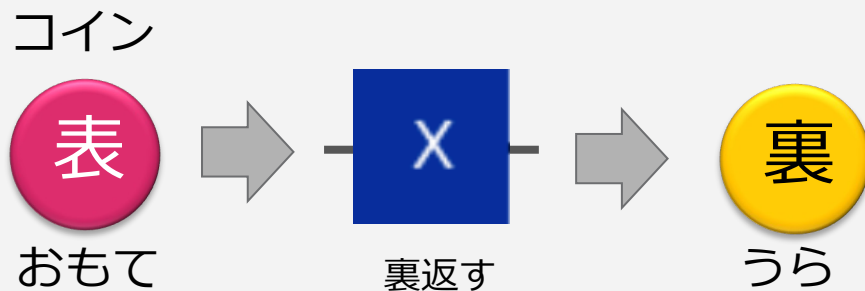
$$|1\rangle = 0 \times |0\rangle + 1 \times |1\rangle$$

$| \rangle$ : 量子ビットを表す記号<sup>6</sup>

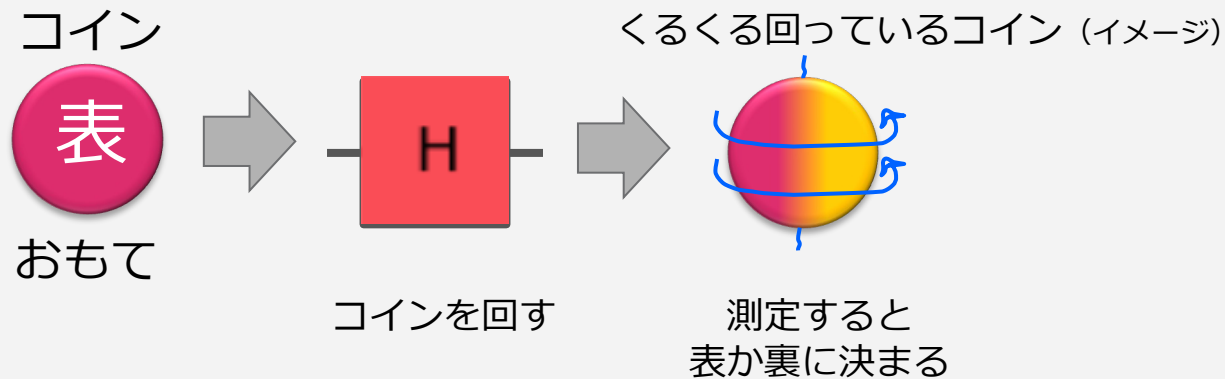
# 量子コンピューターの計算方法



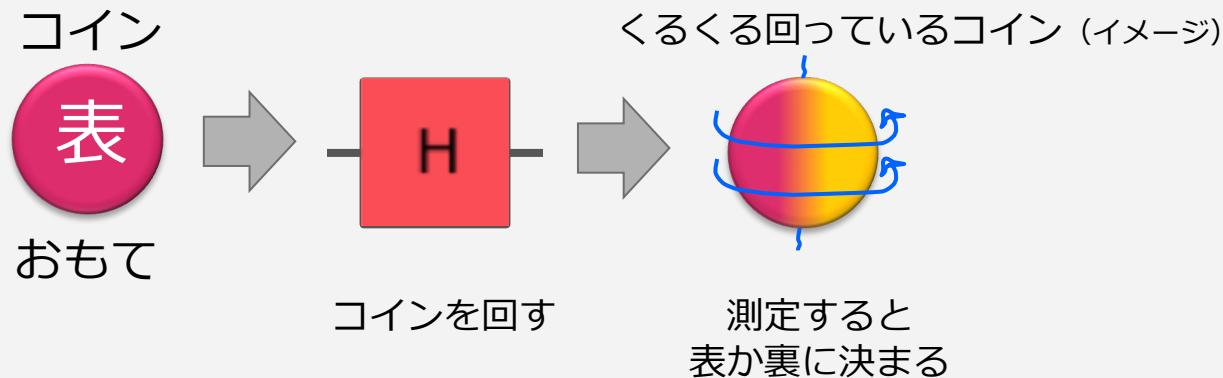
# Xゲート



# Hゲート



# Hゲート

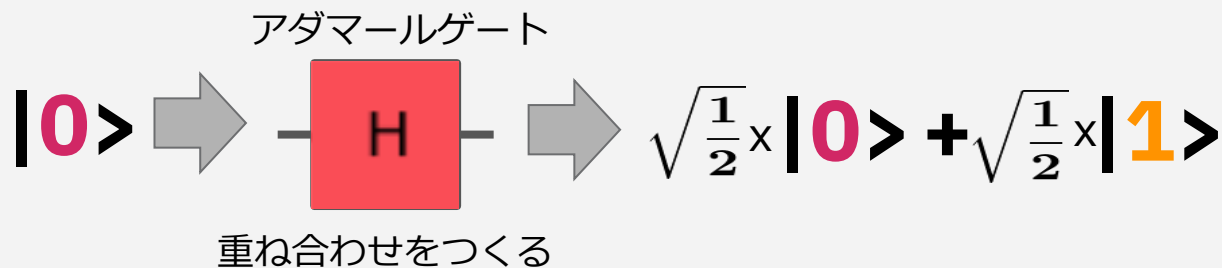
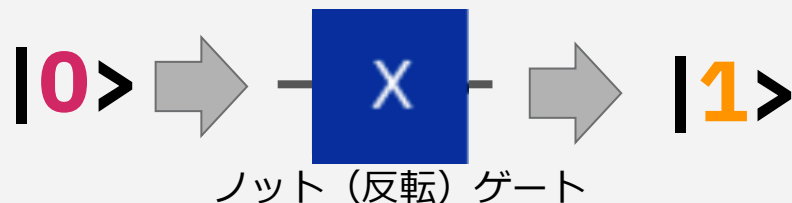


$|0\rangle$  → H →  $\sqrt{\frac{1}{2}} \times |0\rangle + \sqrt{\frac{1}{2}} \times |1\rangle$

重ね合わせをつくる

0 と 1 の重ね合わせ

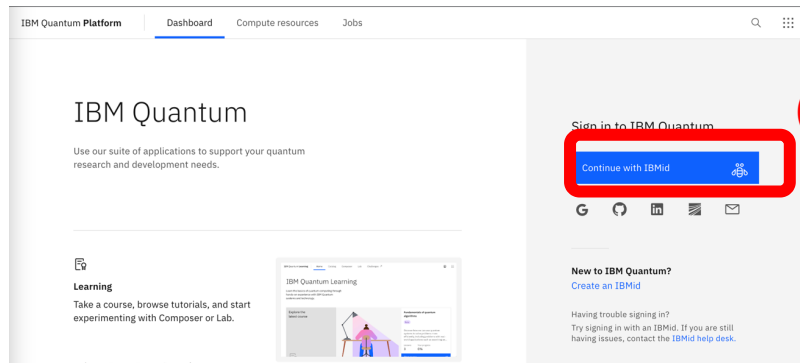
# 量子コンピューターの計算方法



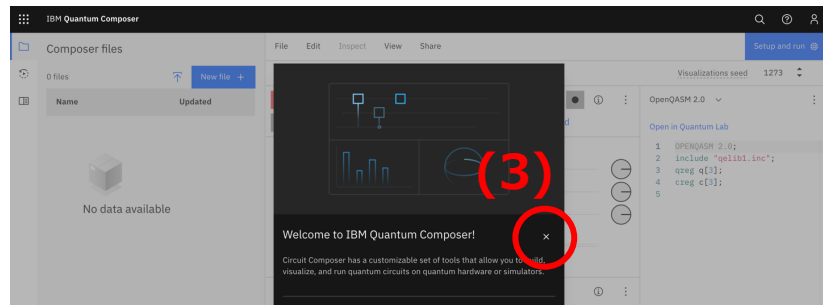


# ハンズオン: IBM Quantum Composer

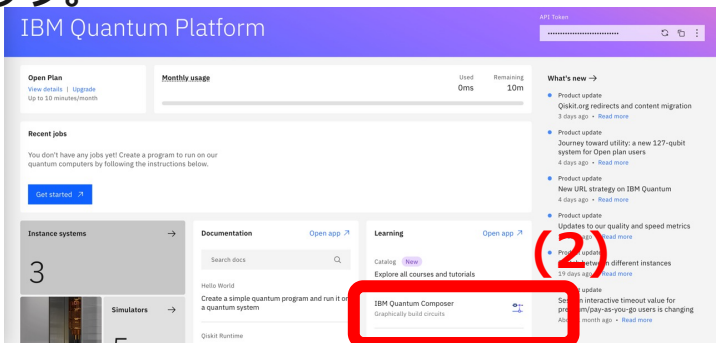
(1) IBM Quantum にログインします。URL:  
<https://quantum.ibm.com/>



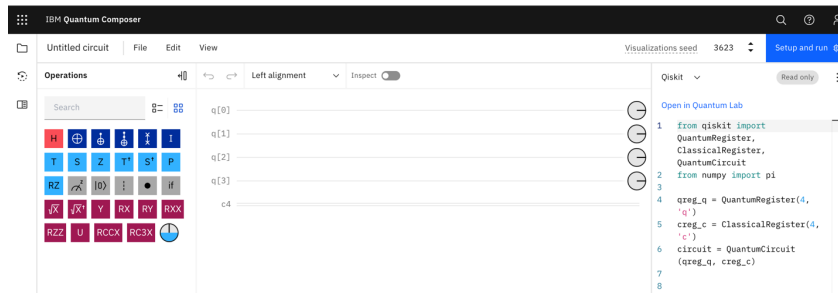
(3) ポップアップウィンドウは「x」をクリックして、閉じます。



(2) 中央下の方の「IBM Quantum Composer」をクリック。

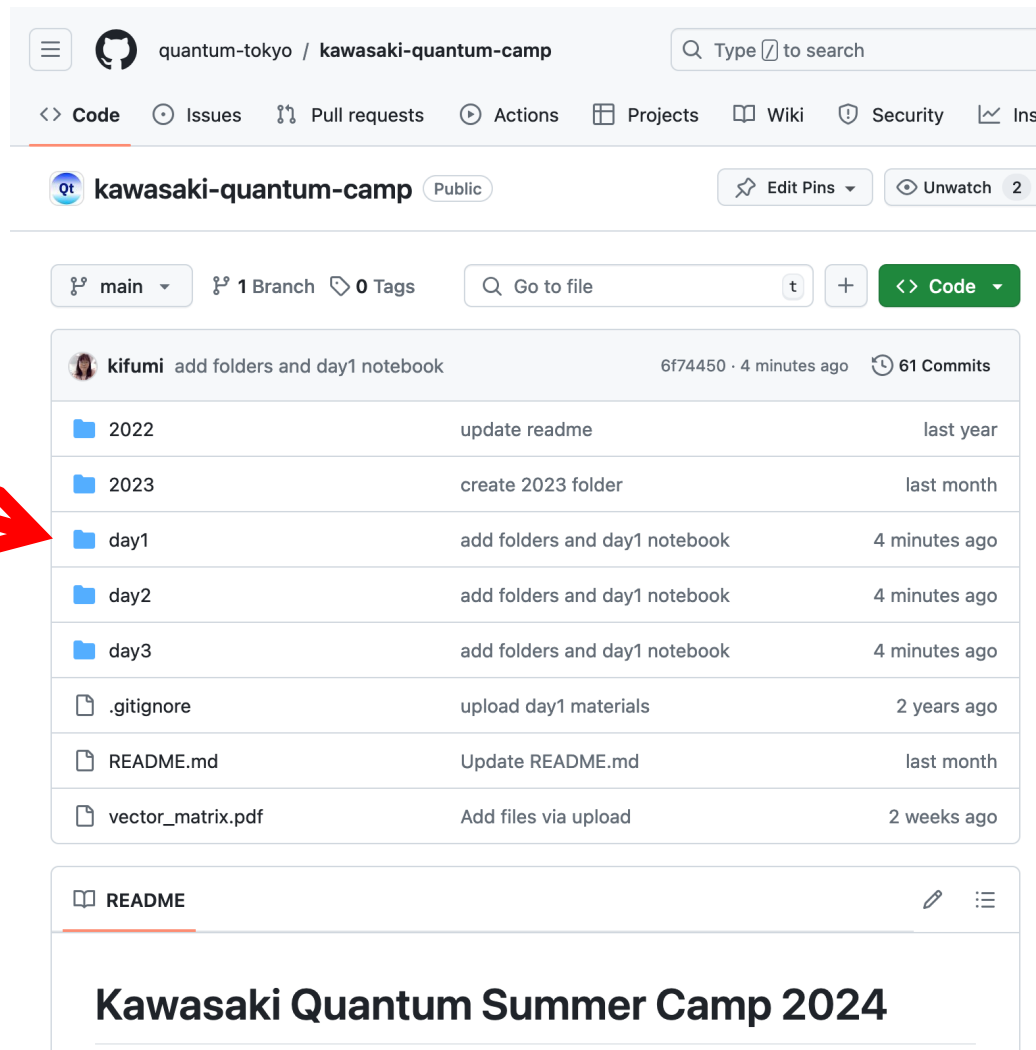


(4) この画面になったら準備完了です。



# ハンズオンの資料

URL: [ibm.biz/kwskgit](https://ibm.biz/kwskgit)



The screenshot shows the GitHub interface for the repository 'kawasaki-quantum-camp'. The repository is public and has 61 commits. The file list shows several folders and files, with the 'day1' folder highlighted by a red arrow.

File/Folder	Commit Message	Time Ago
2022	update readme	last year
2023	create 2023 folder	last month
day1	add folders and day1 notebook	4 minutes ago
day2	add folders and day1 notebook	4 minutes ago
day3	add folders and day1 notebook	4 minutes ago
.gitignore	upload day1 materials	2 years ago
README.md	Update README.md	last month
vector_matrix.pdf	Add files via upload	2 weeks ago

Below the file list, the README section is visible, titled 'Kawasaki Quantum Summer Camp 2024'.

# 1量子ビット回路

The screenshot shows the IBM Quantum Composer interface. At the top, there's a header with the IBM logo and "IBM Quantum Composer". Below it is a menu bar with "File", "Edit", and "View". The main workspace is divided into three sections: "Operations" on the left, a central circuit diagram, and "Qiskit" on the right. The "Operations" section contains a search bar and a grid of quantum gates. The central circuit diagram shows four qubits: q[0], q[1], q[3], and c4. An orange arrow points to a trash icon next to q[1], indicating its deletion. The "Qiskit" section shows a code editor with the following code:

```
1 from qiskit import  
   QuantumCircuit  
2 from numpy import  
3  
4 qreg_q = QuantumRe  
5 creg_c = Classical  
6 circuit = QuantumC
```

マウスでq[1]をクリックするとゴミ箱マークが出てくるので、  
クリックして消します。

q[0]だけにして、1量子ビット回路の準備をします。

# 1量子ビット回路

The screenshot displays the IBM Quantum Composer interface. On the left, the 'Operations' panel contains a grid of quantum gates. A red circle highlights the CNOT gate (represented by a blue square with a white circle and a plus sign). A red arrow points from this gate to the circuit editor. In the center, the circuit editor shows two horizontal lines representing qubits, labeled 'q[0]' and 'c4'. A red arrow points from the CNOT gate to the 'q[0]' line. On the right, the 'Qiskitのコード' (Qiskit code) panel shows the corresponding Qiskit code. A red arrow points from the text 'Qiskitのコード' to the code panel. The code is as follows:

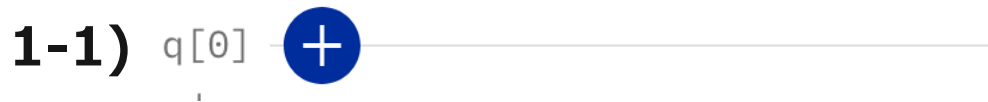
```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(1, 'q')
5 creg_c = ClassicalRegister(4, 'c')
6 circuit = QuantumCircuit(qreg_q, creg_c)
7
8
```

量子ゲートをマウスでドラッグ&ドロップして、量子回路を作ります。

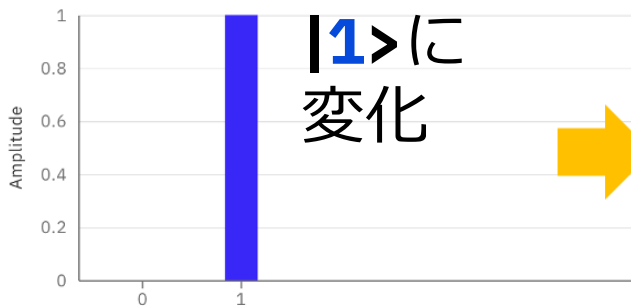
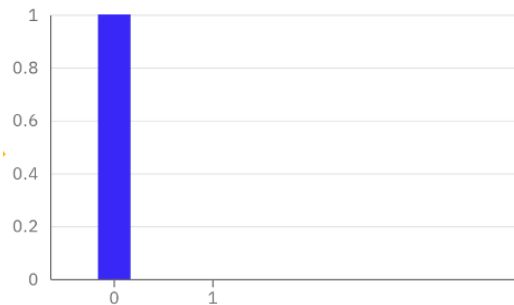
右側には、Qiskitのコードが自動生成されます。

# 1. Xゲート(NOTゲート)

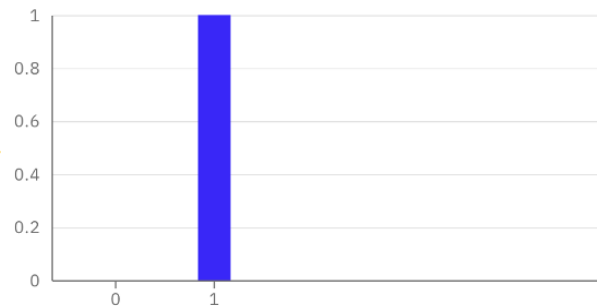
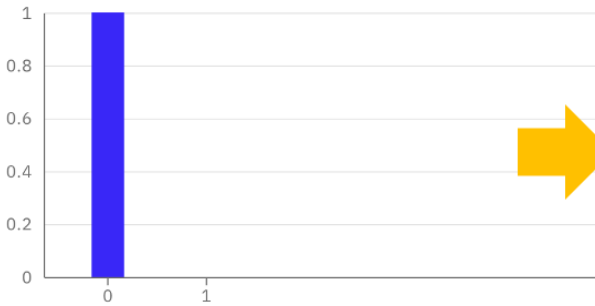
図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。



初期状態は $|0\rangle$



$|1\rangle$ に  
変化



棒グラフ (Statevector 表示) は  
量子ビットの状態

$$\alpha \times |0\rangle + \beta \times |1\rangle$$

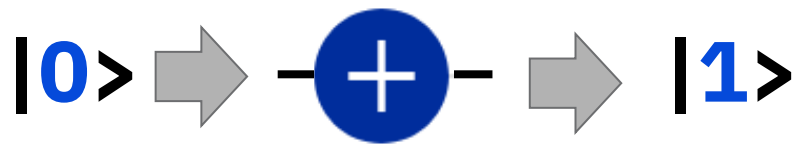
の  $\alpha, \beta$  (確率振幅) です。

$$|0\rangle = 1 \times |0\rangle + 0 \times |1\rangle$$

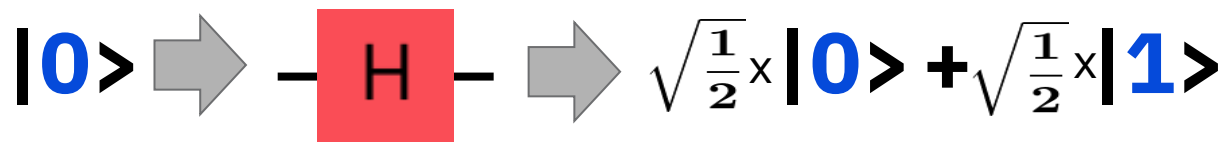
$$|1\rangle = 0 \times |0\rangle + 1 \times |1\rangle$$



# 量子コンピューターの計算方法

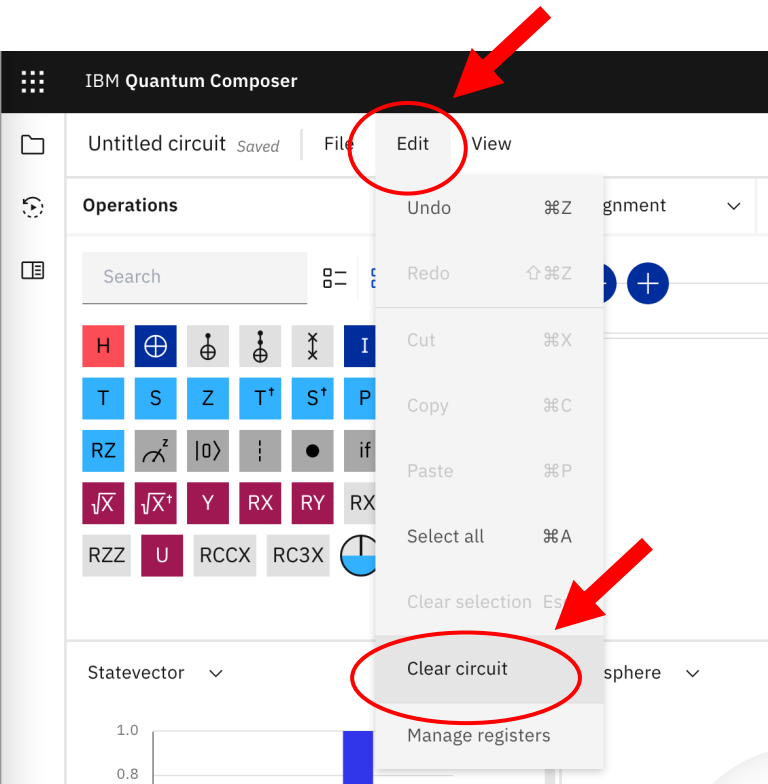


ノット（反転）ゲート

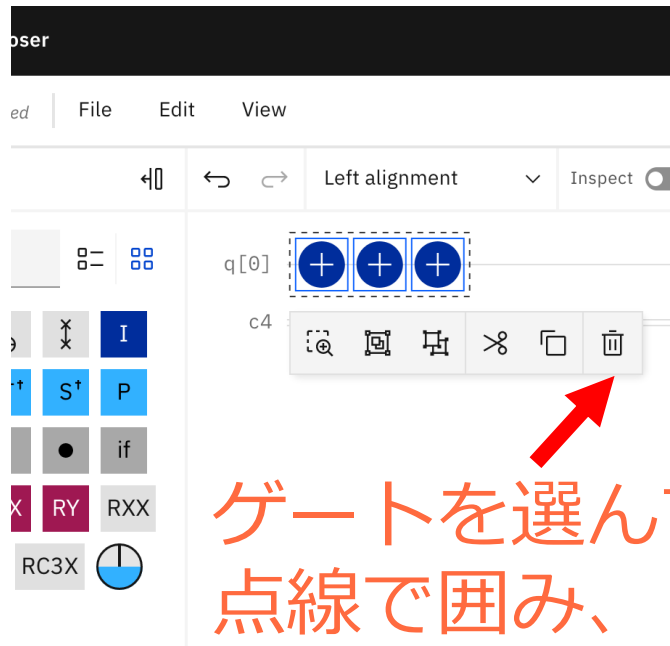


重ね合わせをつくる

# 置いたゲートを取り除く



または



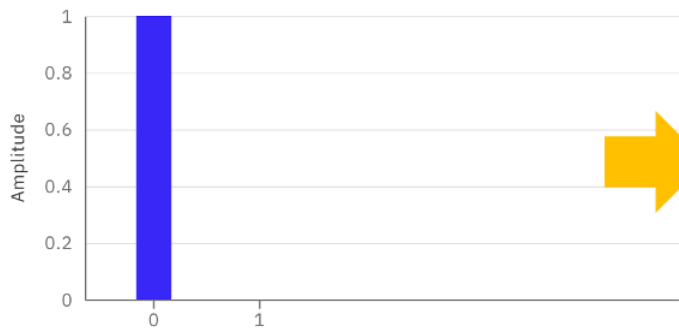
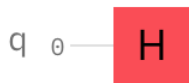
ゲートを選んで  
点線で囲み、  
ゴミ箱マークを  
クリック



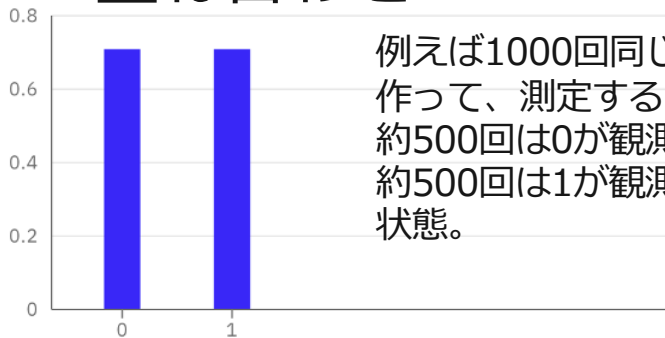
## 2. Hゲート

図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。

2-1)



重ね合わせ



例えば1000回同じ状態を作って、測定すると約500回は0が観測され、約500回は1が観測される状態。

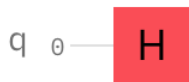
$$|0\rangle = 1 \times |0\rangle + 0 \times |1\rangle$$

$$\begin{aligned} & 0.707 \times |0\rangle + 0.707 \times |1\rangle \\ &= \frac{1}{\sqrt{2}} \times |0\rangle + \frac{1}{\sqrt{2}} \times |1\rangle \end{aligned}$$

## 2. Hゲート

図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。

2-1)



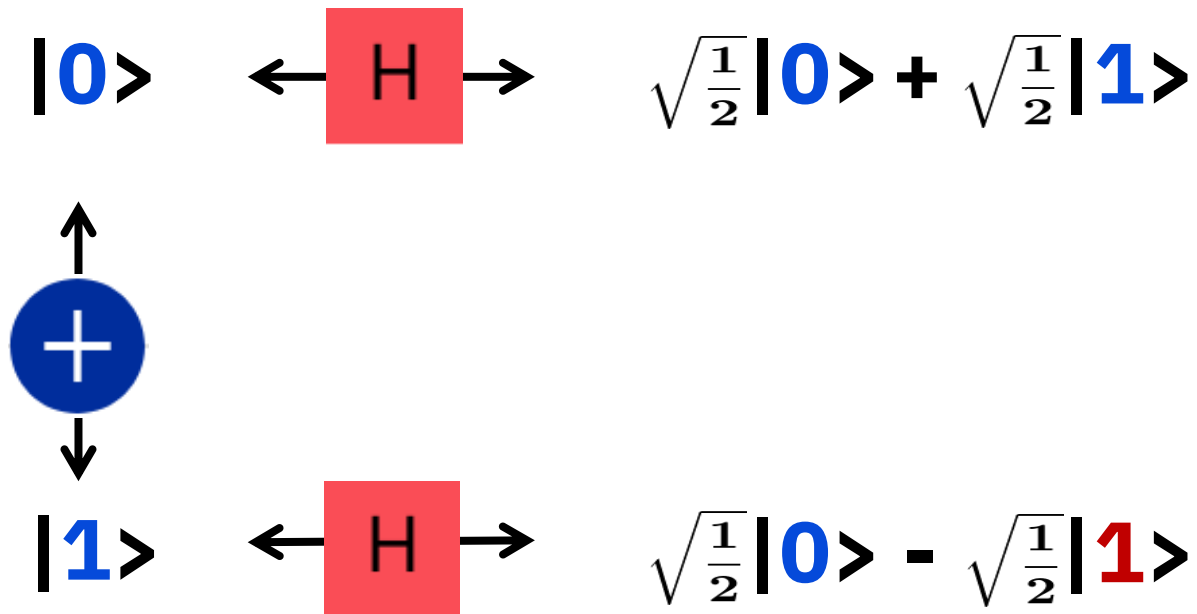
2-2)



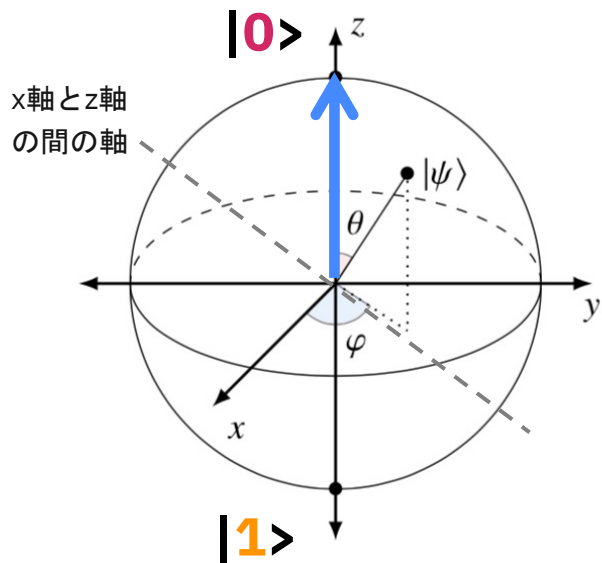
2-3)



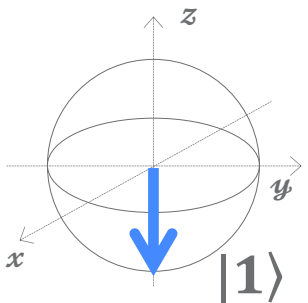
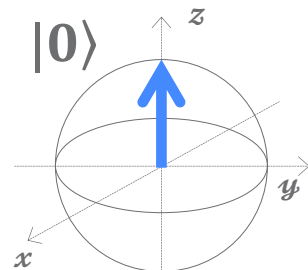
# 量子コンピューターの計算方法



# ブロッホ球

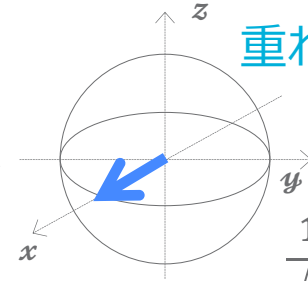
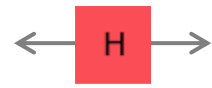


北極 : 0の確率が100%

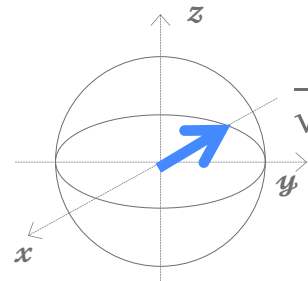
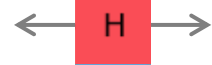


南極 : 1の確率が100%

赤道 : 0と1が50%ずつの  
重ね合わせ状態

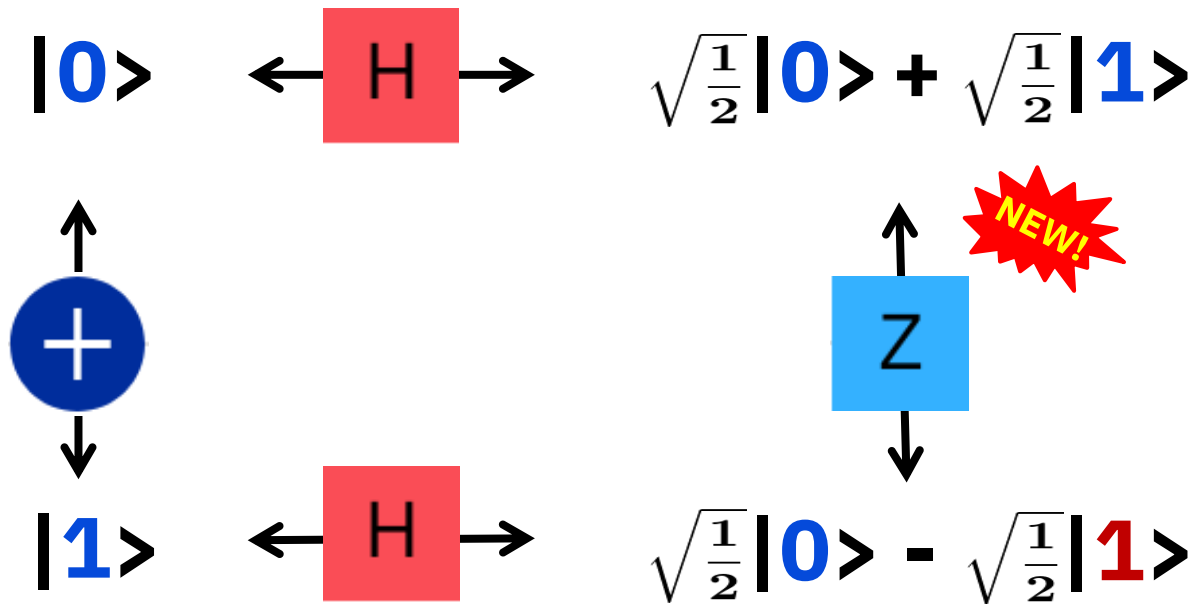


$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$



$$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

# 量子コンピューターの計算方法

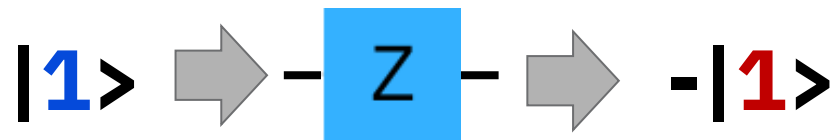
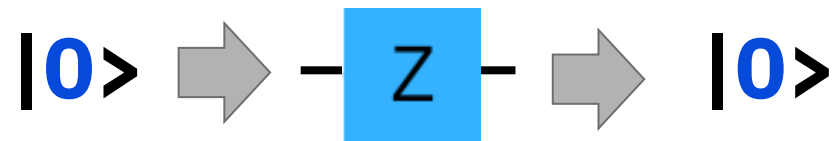


# 3. Zゲート

図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。

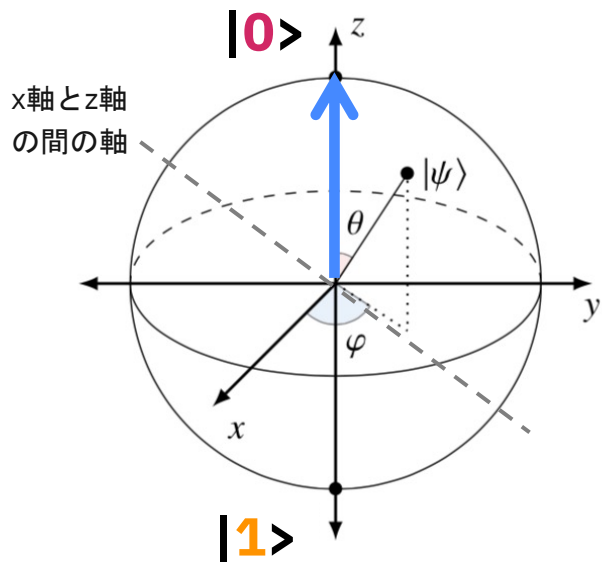


# 量子コンピューターの計算方法

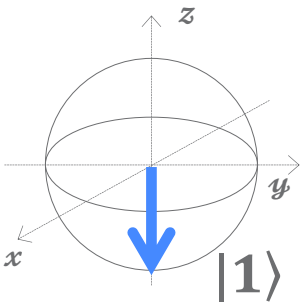
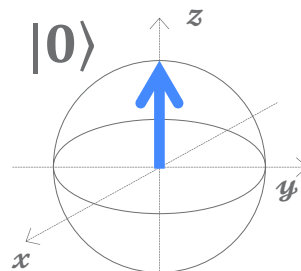


1 の符号を反転する

# ブロッホ球

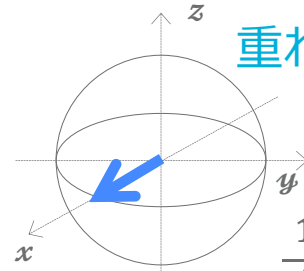
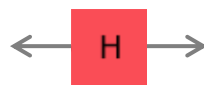


北極 : 0の確率が100%

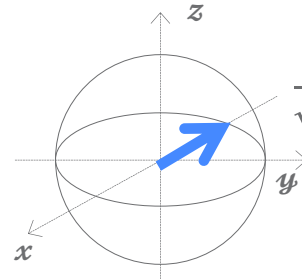
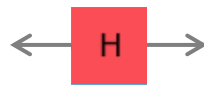


南極 : 1の確率が100%

赤道 : 0と1が50%ずつの  
重ね合わせ状態



$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

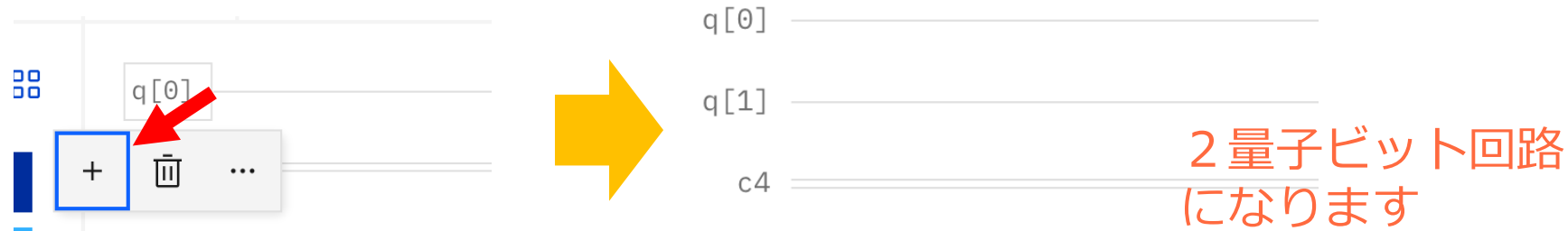


$$\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$



## 4. 量子重ね合わせ

q[0]をクリックして、さらに「+」マークをクリックして、2量子ビットの回路を準備します。



図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。

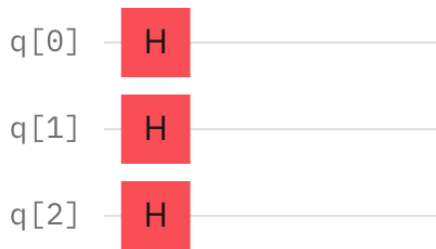
### 4-1)



さらにq[1]をクリックして、さらに「+」マークをクリックして、3量子ビット、4量子ビット、5量子ビットの時の重ね合わせ状態を確認します。



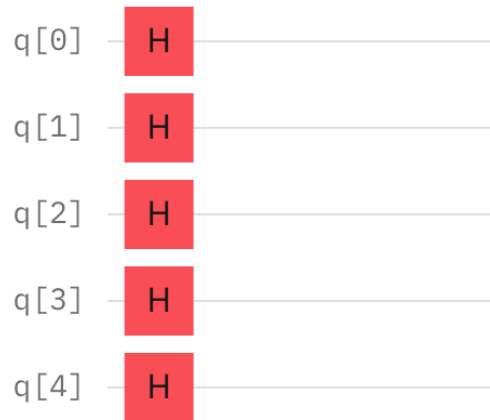
**4-2)**



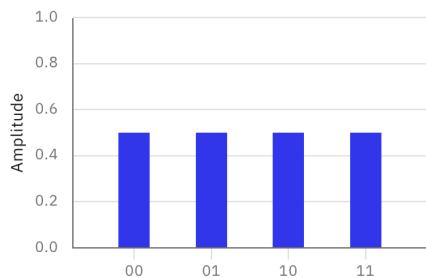
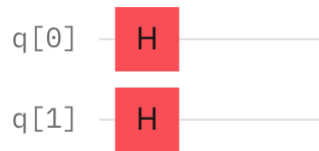
**4-3)**



**4-4)**

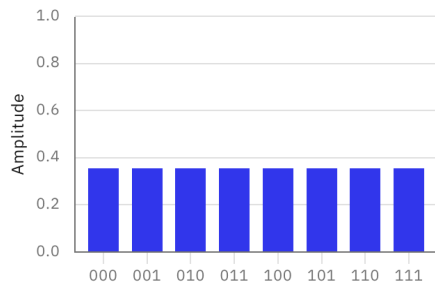
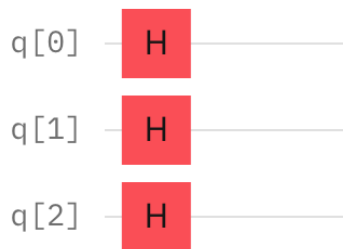


## 4-1)



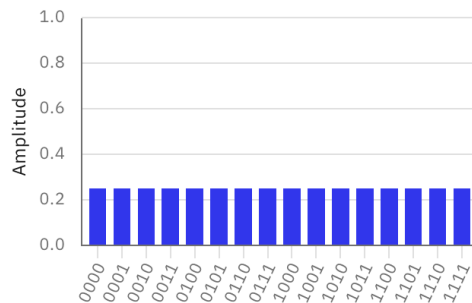
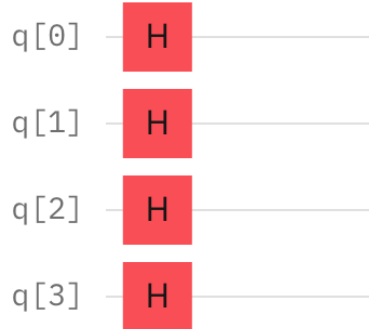
4個

## 4-2)



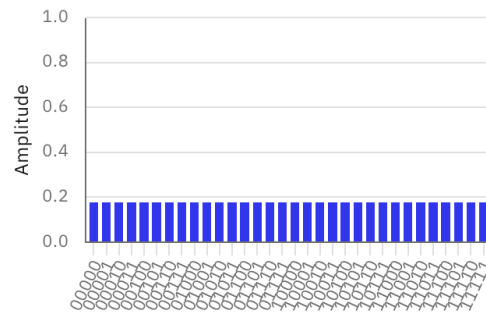
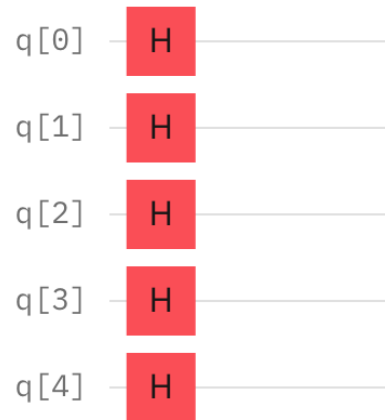
8個

## 4-3)



16個

## 4-4)



32個

量子ビット数( $n$ )が増えるにつれて、量子状態が倍々に( $2^n$ 個に)増えていくことがわかります。

# 2量子ビット・2古典ビットの状態を作る

q[0]をクリックして、さらに「ゴミ箱」マークをクリック、を繰り返して、2量子ビットの回路を準備します。



2量子ビット回路  
にします

次に、c4をクリックして、「-」マークをクリックするを2回繰り返して、2古典ビットにします。



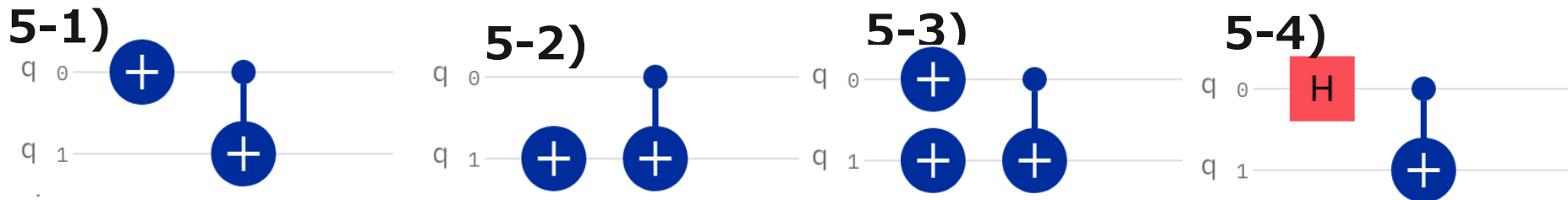
2古典ビットになります

# 5. CNOTゲート(制御Xゲート)

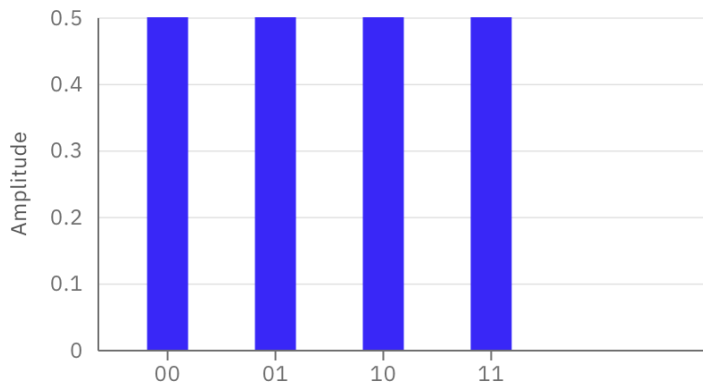
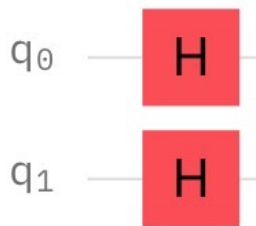
制御ビットが $|1\rangle$ のときのみ、目標ビットを反転（NOT）するゲートです。



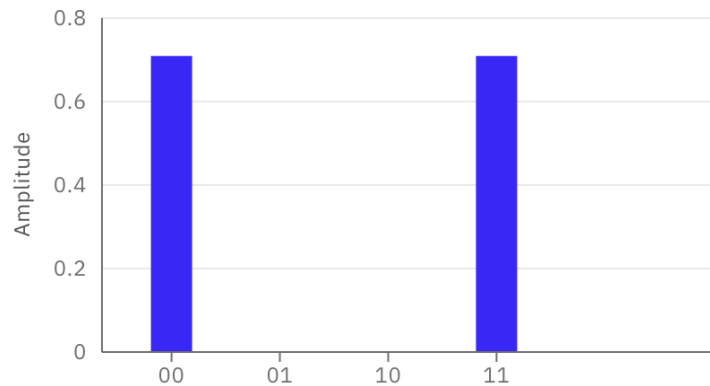
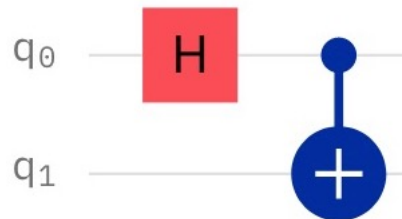
入力		出力	
目標ビット	制御ビット	目標ビット	制御ビット
0	0	0	0
1	0	1	0
0	1	<b>1</b>	<b>1</b>
1	1	<b>0</b>	<b>1</b>



## 量子重ね合わせ

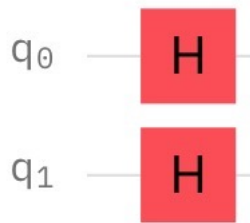


## 量子もつれ (エンタングルメント)



CNOTゲートは、  
エンタングルメントを作ります。

## 量子重ね合わせ



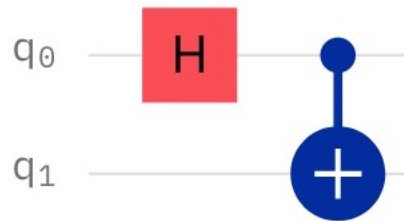
0 0 ... 25%

0 1 ... 25%

1 0 ... 25%

1 1 ... 25%

## 量子もつれ (エンタングルメント)



0 0 ... 50%

0 1 ... 0%

1 0 ... 0%

1 1 ... 50%

1量子ビット目が0だと分かったら  
2量子ビット目も0

## 6.量子コンピューターで実験

## ファイル名を変更 (Entanglementなど)





## Set up and run your circuit



Step 1

Choose a QPU

Search by QPU name ↑↓

☒ **ibm\_sherbrooke** [See details](#)

QPU status ● Online  
Total pending jobs 630  
127 Qubits 2.9% EPLG 5K CLOPS

☐ **ibm\_brisbane** [See details](#)

QPU status ● Online  
Total pending jobs 862  
127 Qubits 4.5% EPLG 5K CLOPS

☐ **ibm\_osaka** [See details](#)

QPU status ● Online  
Total pending jobs 918

(4)

スクロール

Step 2

Choose your settings

Instance

ibm-q/open/main

Shots \*

1024

Job limit: 3

Tags (optional)

Add tags

(5)

デバイスを選択します。

- ・ Total pending jobsが少ない
  - ・ EPLG(エラー)が小さい
- などから選んでみましょう。

Close

Run on ibm\_sherbrooke

(6)

IBM Quantum Composer

Untitled circuit

Saved

File

Edit

View

Visualizations seed

3623

Setup and run

Composer jobs

Search

H

$\oplus$

$\otimes$

$\otimes$

$\otimes$

I

T

S

Z

$T^\dagger$

$S^\dagger$

P

q[0]

H

$\oplus$

$Z$

q[1]

$\oplus$

$Z$

c2

0

1

Qiskit

Open in Quantum Lab

1

from qiskit import QuantumRegister,

2

ClassicalRegister, QuantumCircuit

3

from numpy import pi

4

qreg\_q = QuantumRegister(2, 'q')

Composer jobs

Showing jobs in ibm-q/open/main

Search jobs

from this file

Queued: Jul 28, 2024 5:59 PM

ID: ctk0gpa6g3rg0086x2ng | ibm\_sherbrooke

Untitled circuit

Saved

File

Edit

View

Visualizations seed

3623

Setup and run

Operations

Search

H

$\oplus$

$\otimes$

$\otimes$

$\otimes$

I

T

S

Z

$T^\dagger$

$S^\dagger$

P

RZ

$Z$

$|0\rangle$

$|1\rangle$

$\bullet$

if

q[0]

H

$\oplus$

$Z$

q[1]

$\oplus$

$Z$

c2

0

1

(7) Jobの確認



(8)

IBM Quantum Learning | Home | Catalog | Network | **Composer**

Jobs / ctk0gpa6g3rg0086x2ng

See more details

**待ち時間がある場合**

Q# name: ibm\_sherbrooke

Status timeline

- Created: Jul 28, 2024 5:59 PM
- In queue
  - Estimated wait time: in about 2 hours, project pos: 2, QPU pos: 3
- Running
  - Estimated usage: 40.7s
- Completed

Details

Operations

Search

q[0] q[1] c2

Visualizations seed 3623

Setup and run

Statevector

Q-sphere

待ち時間がかかる場合は、少し経ってから確認します。

[See more details](#)

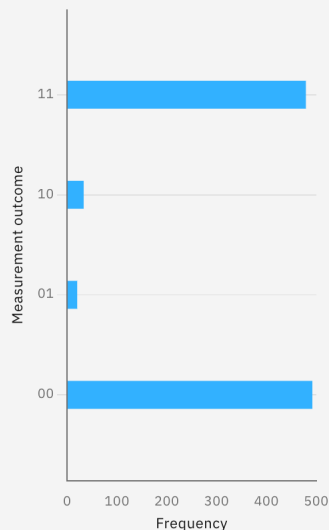
Completed  
Jul 21, 2022 9:00 PM (in 38.2s)

Backend  
ibmq\_belem

Status timeline Completed

Details

Result - histogram

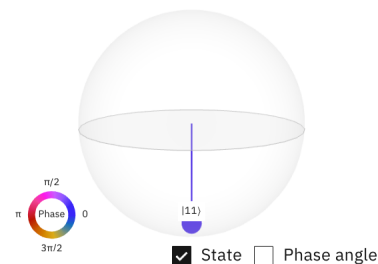
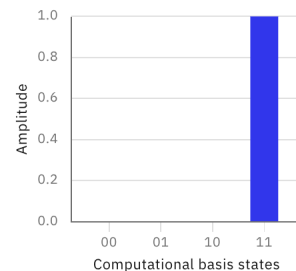


## (9) 実機での結果



00と11以外の結果も含まれています。  
ノイズによるエラーが原因です。

Statevector



Untitled circuit *Saved*

File

Edit

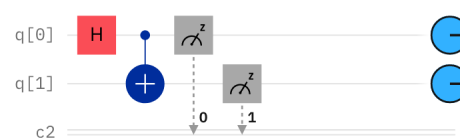
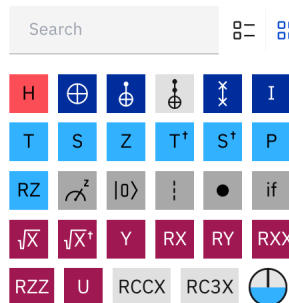
View

Visualizations seed

3623

Setup and run

Operations



Qiskit

Read only

[Open in Quantum Lab](#)

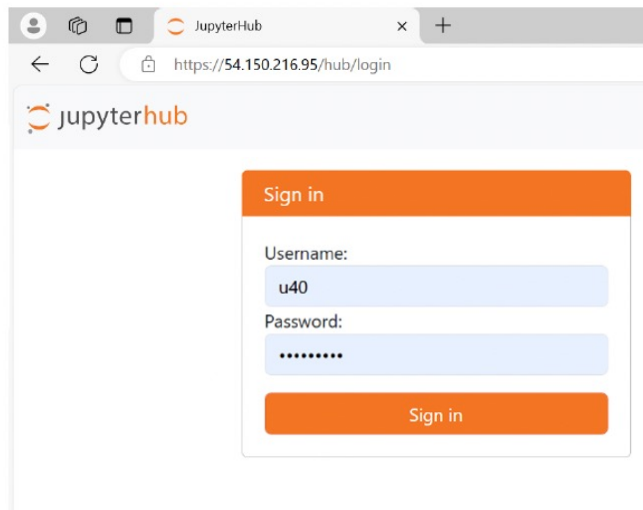
```
1 from qiskit import QuantumRegister,
2   ClassicalRegister, QuantumCircuit
3
4 qreg_q = QuantumRegister(2, 'q')
5 creg_c = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(qreg_q, creg_c)
7
8 circuit.h(qreg_q[0])
9 circuit.cx(qreg_q[0], qreg_q[1])
10 circuit.measure(qreg_q[0], creg_c[0])
11 circuit.measure(qreg_q[1], creg_c[1])
```

# JupyterHubでの実行

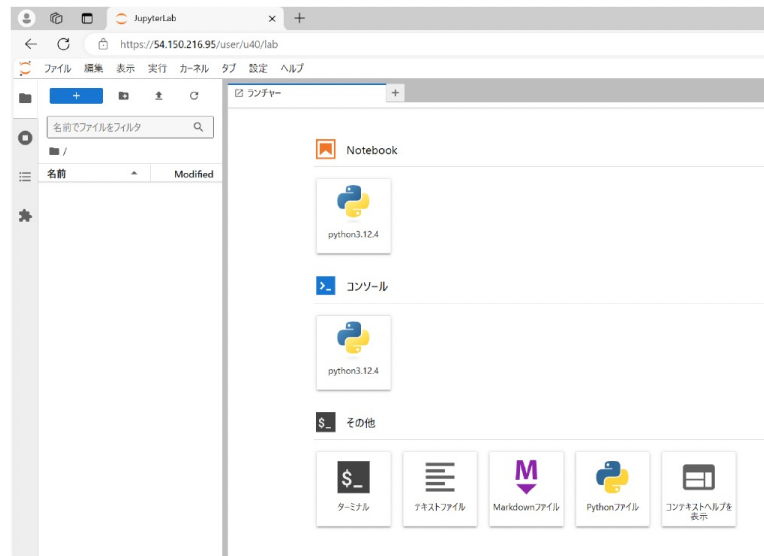
(1) Webブラウザ（ Edge、Safari、Chrome、Firefoxなど）で  
<https://54.150.216.95>にログイン。



(2) ユーザ名とパスワード（メールで配布）を  
入力して、「Sign in」をクリック。



(3) この画面になったら成功です！



# Kawasaki Campが終わった後、Qiskitを実行する場合

(1) Google Colabratory (<https://colab.research.google.com/>) を使う。

毎回、以下のコマンドを最初に実行する必要があります。

```
!pip install qiskit qiskit[visualization] qiskit-ibm-runtime qiskit-aer
```

```
!pip install qiskit-algorithms qiskit-nature scikit-learn
```

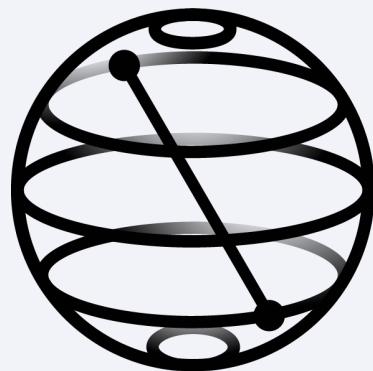
```
!pip install --prefer-binary pycsf
```

参照ブログ : <https://qiita.com/kifumi/private/51a5d2a420e6318f78fb>

(2) qBraid (<https://www.qbraid.com>) を使う。

実機で実行する場合は、以下のコマンドを実行する必要があります。

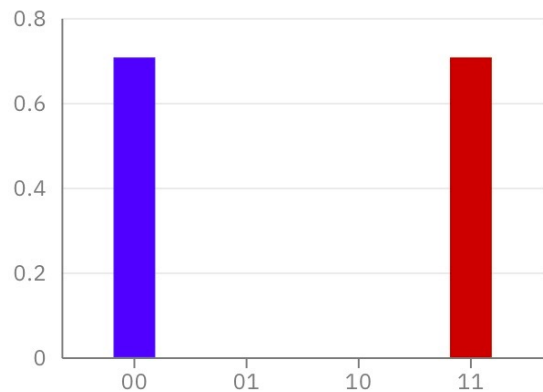
```
%pip uninstall --yes simplejson
```



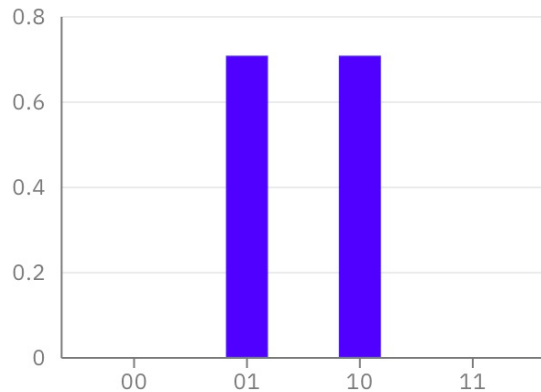
# 演習問題

2量子ビットのエンタングル状態を作ってみましょう。  
答えは一つではないので、どんな作り方でもOKです。

(1)  $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$



(2)  $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$



(3)  $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

