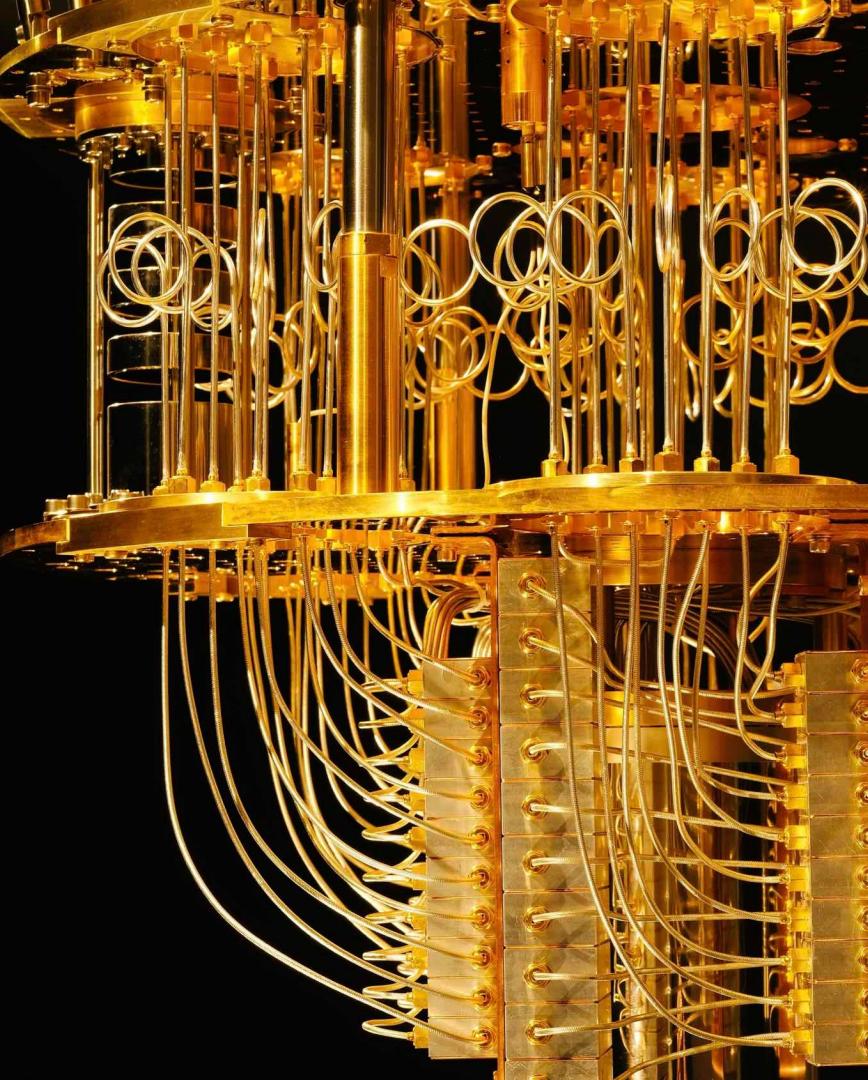


量子コンピューティング ハンズオン

沼田 祢史
Kifumi Numata
IBM Quantum

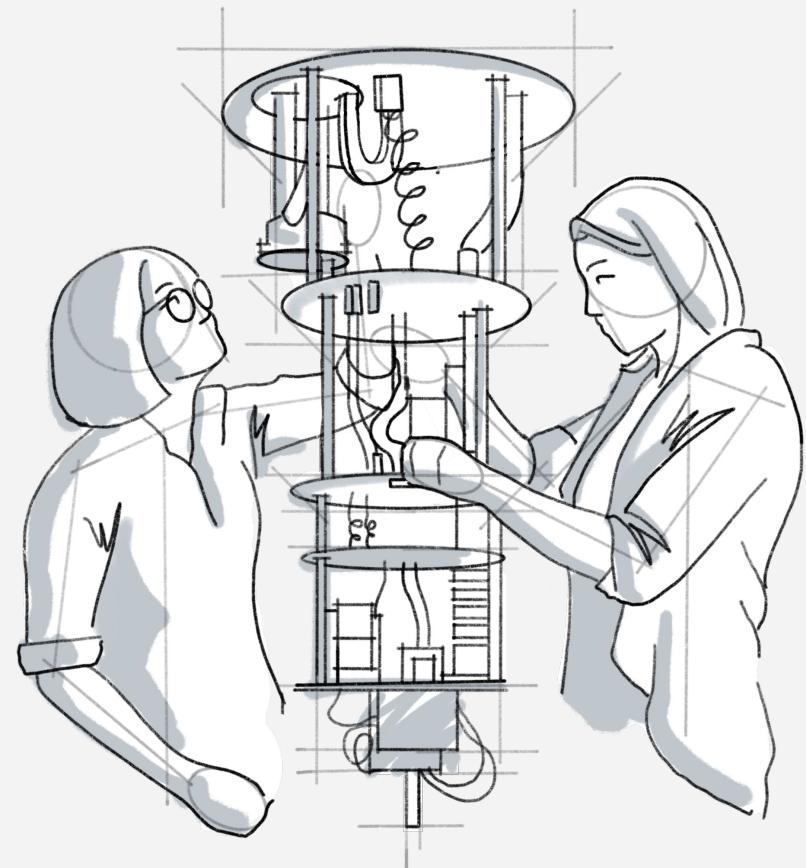
Dec 16, 2022



本日の内容

- ・量子コンピューターとは
- ・量子情報入門
- ・ハンズオン：IBM Quantum Composerでの実験
『IBM Quantum Composerの登録（くわしいバージョン）』
URL : <https://qiita.com/kifumi/items/7ac33ab7939d2dd796d0>

量子情報入門



量子とは・・・

私達の世界の物理法則（ニュートン力学）では表現できない摩訶不思議な世界。

例えば・・・

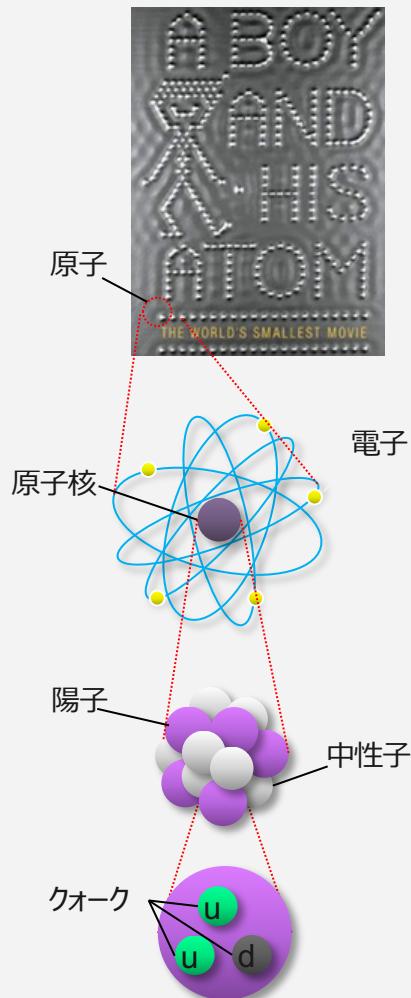
波と粒子の二重性

量子重ね合わせ

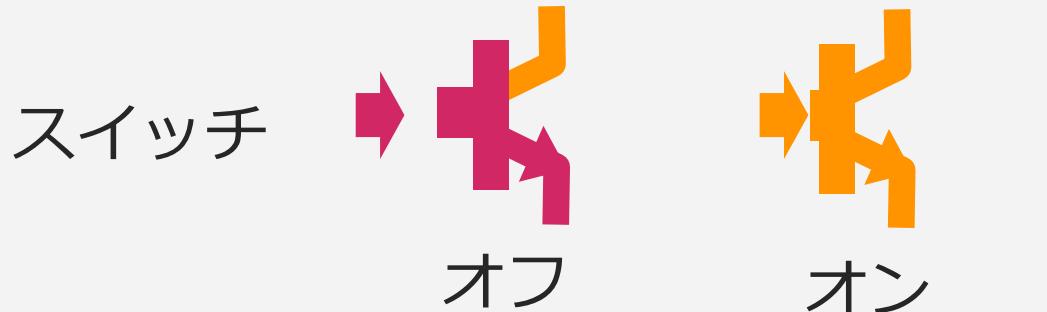
量子もつれ（エンタングルメント）

観測した瞬間に状態が一つに決まる

量子コンピューターは量子現象を使って計算を行う新しい仕組みのコンピューターです。



コンピューターの中は、ビットで計算



ビット

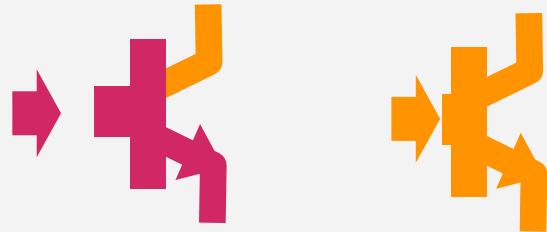
0 1

例)

文字列	ビット
7	111
A	0100 0001



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

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

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

コイン

表

おもて



コイン

裏

うら

量子コンピューターの
量子ビット

0 と **1**

両方

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

コイン

表

おもて



コイン

裏

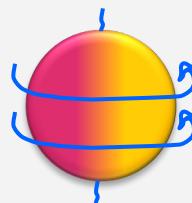
うら

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

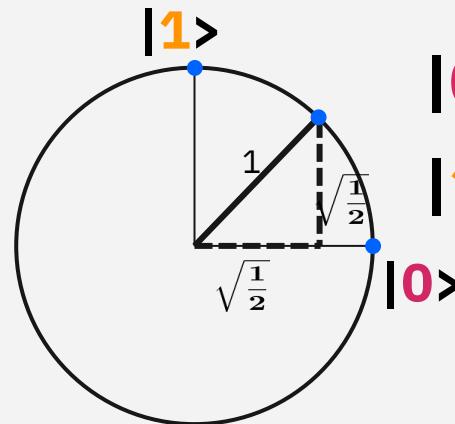
0 または 1

どちらか

量子コンピューターの
量子ビット

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

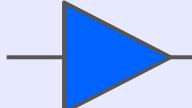
0 と 1 の「重ね合わせ」



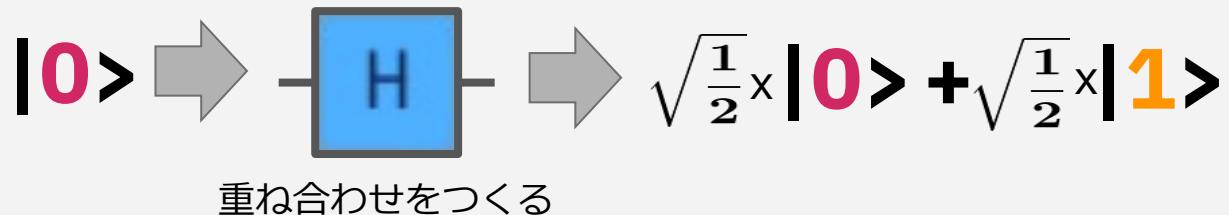
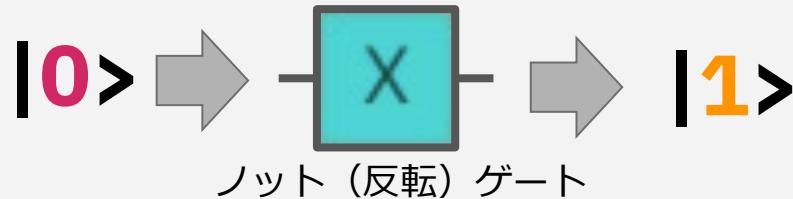
$$|0\rangle = 1 \times |0\rangle + 0 \times |1\rangle$$

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

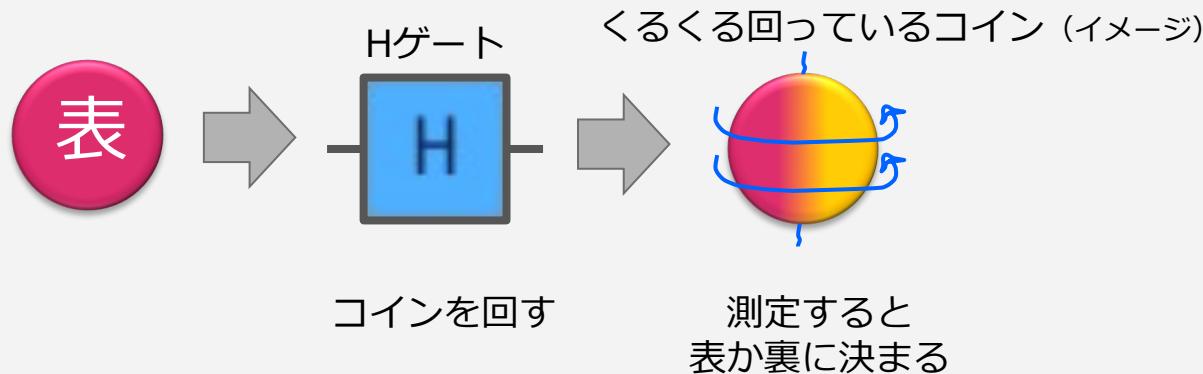
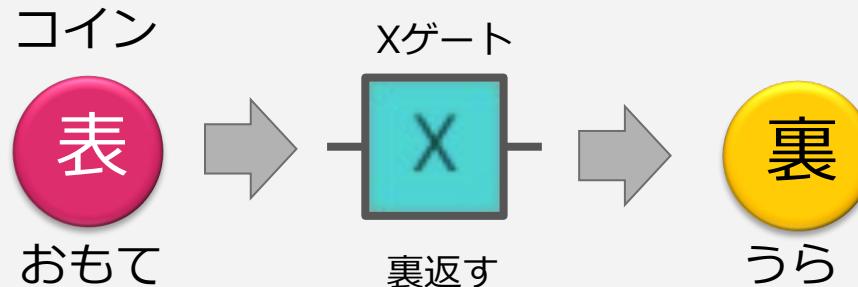
ビットと論理ゲート

	古典	量子
ビット	0 または 1	0と1の重ね合わせ $\alpha \times 0\rangle + \beta \times 1\rangle$
論理ゲート	NOT, OR, AND など  	X, H, CNOT など   

量子コンピューターの計算方法



量子コンピューターの計算方法



ビットと論理ゲート

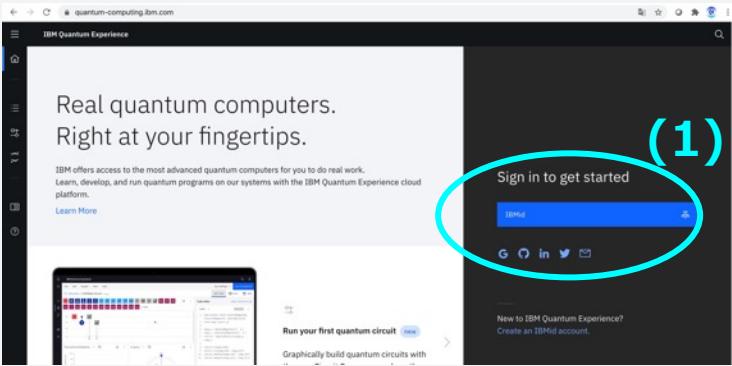
	量子	ベクトル・行列表示
ビット	0と1の重ね合わせ $\alpha \times 0\rangle + \beta \times 1\rangle$	$ 0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, 1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $ \psi\rangle = \alpha 0\rangle + \beta 1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ α, β : 確率振幅 $ \alpha ^2 + \beta ^2 = 1$
論理 ゲート	X, H, CNOT など	$\begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}$ 量子ゲート 入力ビット 出力ビット

例) Xゲートの場合

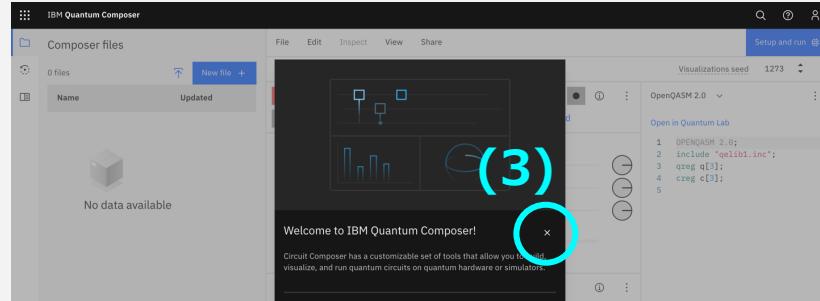
$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 1 + 0 \cdot 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

ハンズオン

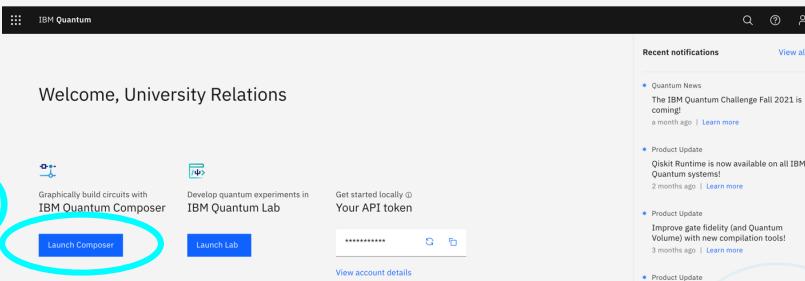
(1) IBM Quantum にログインします。URL:
<https://quantum-computing.ibm.com/>



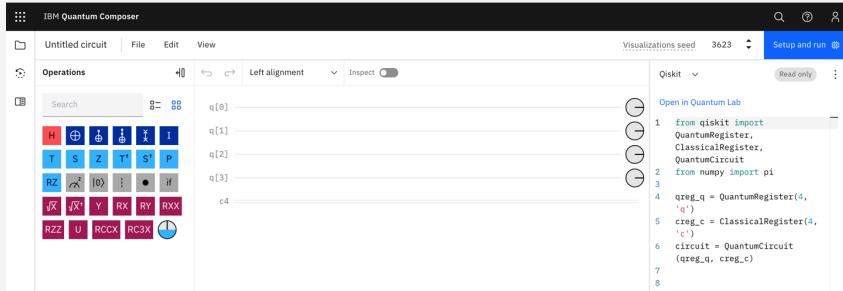
(3) ポップアップウィンドウは「x」をクリックして、閉じます。



(2) 左の青アイコン「Launch Composer」をクリック。



(4) この画面になつたら準備完了です。



ハンズオンの資料

<https://github.com/kifumi/quantum-composer>

The screenshot shows the GitHub repository page for `kifumi/quantum-composer`. The repository is public and contains one commit from `kifumi` titled "Initial commit". The commit was made 20 hours ago and includes a file named "README.md". The repository has 0 stars, 1 watching, and 0 forks. There are no releases or packages published.

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main · 1 branch · 0 tags

Code

About

No description, website, or topics provided.

Readme · 0 stars · 1 watching · 0 forks

Releases

No releases published · Create a new release

Packages

No packages published · Publish your first package

1量子ビット回路

The screenshot shows the IBM Quantum Composer interface. At the top, it says "IBM Quantum Composer". Below that is a toolbar with "Untitled circuit", "File", "Edit", "View", and "Visu". The main area is titled "Operations" and includes a search bar, alignment tools, and an "Inspect" toggle. On the left is a palette of quantum gates: H, CNOT, Toffoli, Swap, Z, T, S, RZ, RY, RY†, V, PY, PY†, P, S†, T†, and If. There are four horizontal lines representing qubits: q[0], q[1], q[2], and q[3]. A vertical line represents a classical bit c4. A red arrow points to the trash bin icon next to q[1].

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

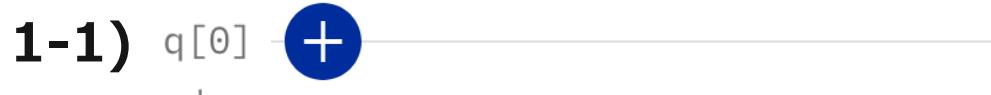
The screenshot shows the IBM Quantum Composer interface. On the left, there's a toolbar labeled "量子ゲート" (Quantum Gates) with various gate icons. A red circle highlights the Hadamard gate (H). A red arrow points from this highlighted gate to the "Operations" section of the main workspace, which contains a search bar and a grid of quantum operations including H, T, RZ, and others. Another red arrow points from the "Operations" section to the workspace itself, where a single-qubit register q[0] and a four-classical-bit register c4 are shown. The workspace has a "Left alignment" button and an "Inspect" toggle. On the right, a large red arrow points from the workspace to a code editor window titled "Qiskit". The code editor displays the following Qiskit code:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi
qreg_q = QuantumRegister(1, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)
```

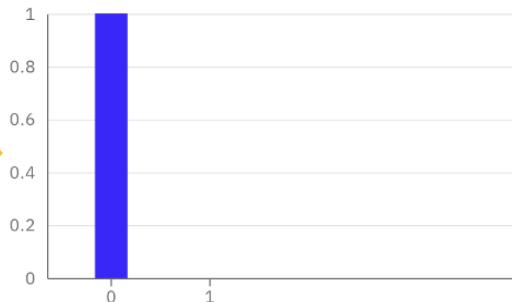
量子ゲートをマウスでドラッグ&ドロップして、
量子回路を作ります。
右側には、Qiskitのコードが自動生成されます。

1. Xゲート(NOTゲート)

図の回路を作つてみてください。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 $|0\rangle$



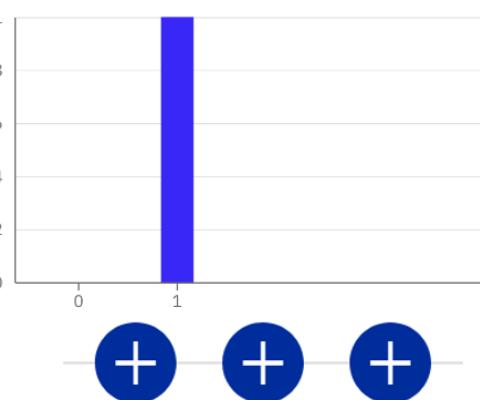
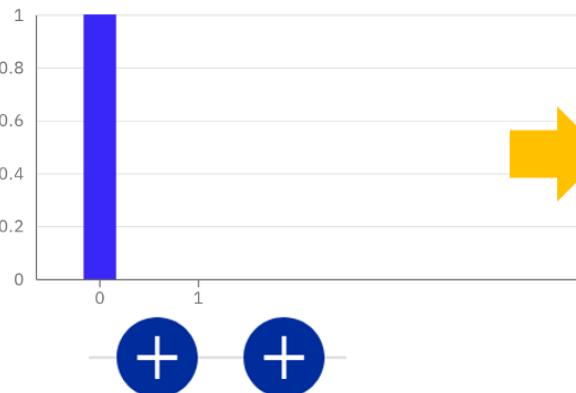
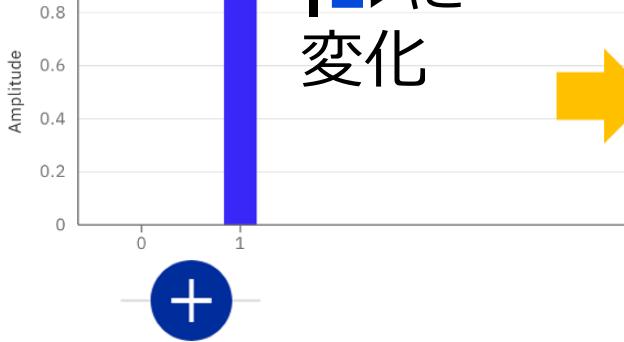
棒グラフ (Statevector 表示) は
量子ビットの状態

$\alpha \times |0\rangle + \beta \times |1\rangle$
の α, β (確率振幅) です。

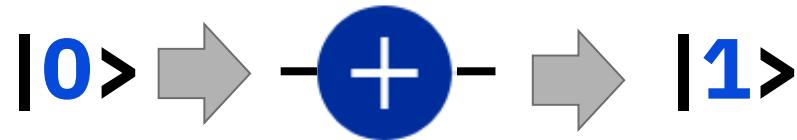
$$|0\rangle = 1 \times |0\rangle + 0 \times |1\rangle$$

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

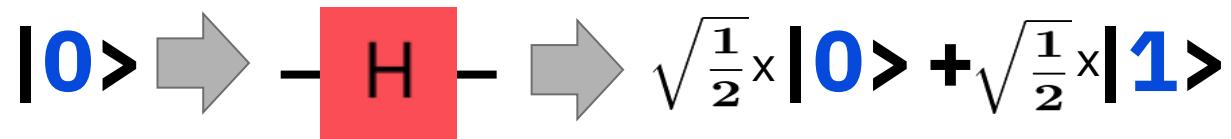
$|1\rangle$ に
変化



量子コンピューターの計算方法

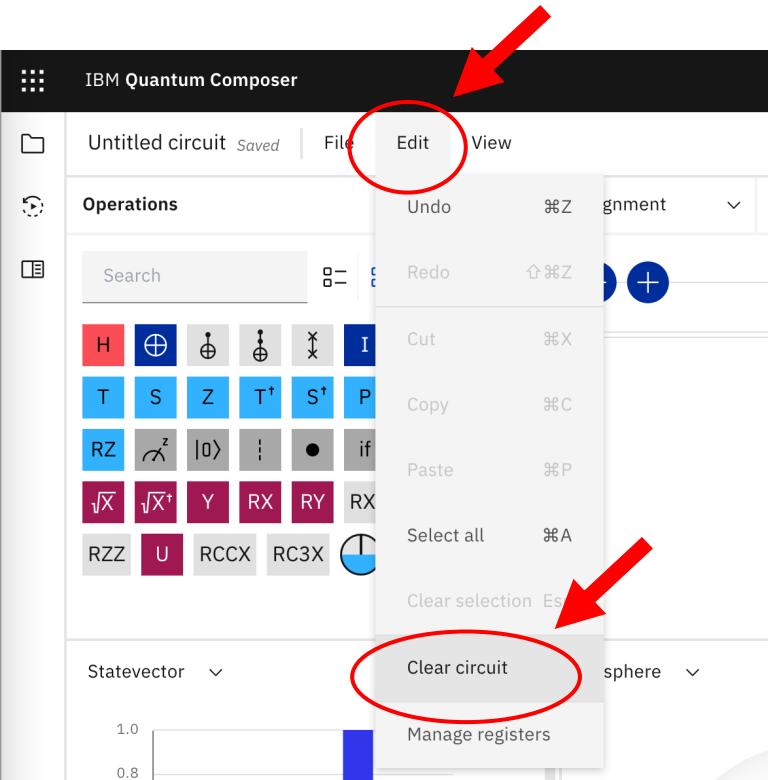


ノット（反転）ゲート

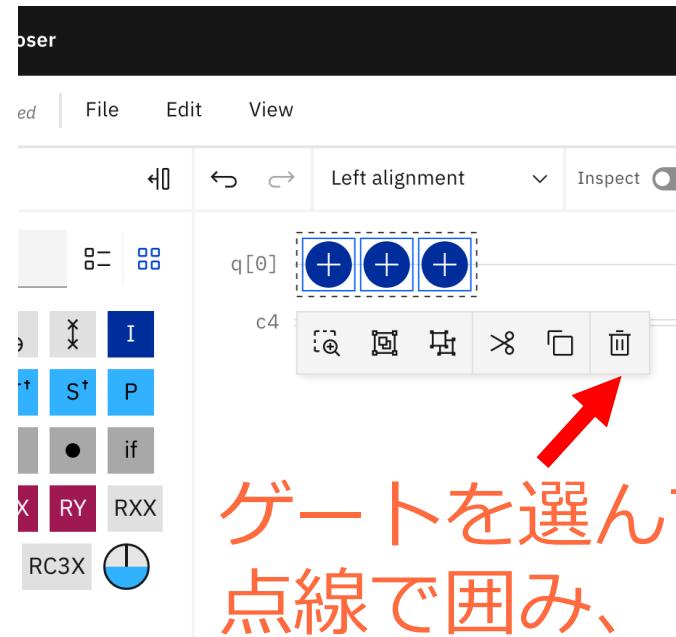


重ね合わせをつくる

置いたゲートを取り除く



または

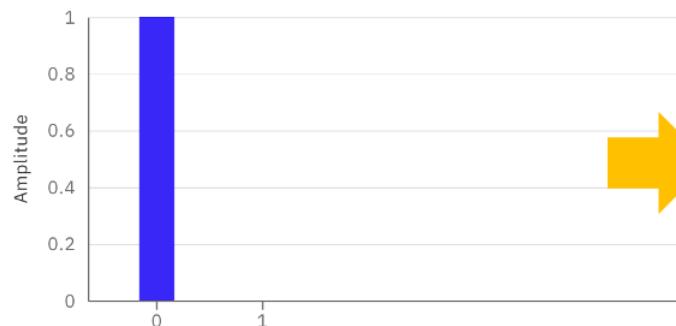
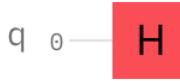


ゲートを選んで
点線で囲み、
ゴミ箱マークを
クリック

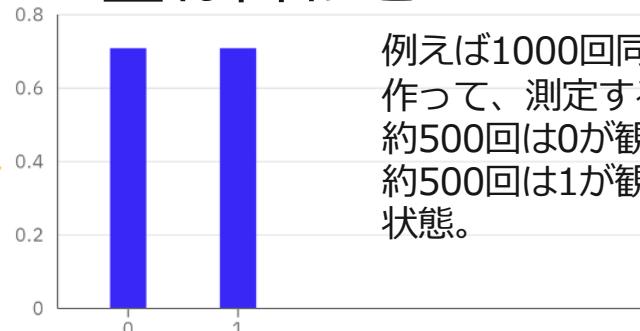
2. Hゲート

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

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



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

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。

3-1)



3-2)



3-3)



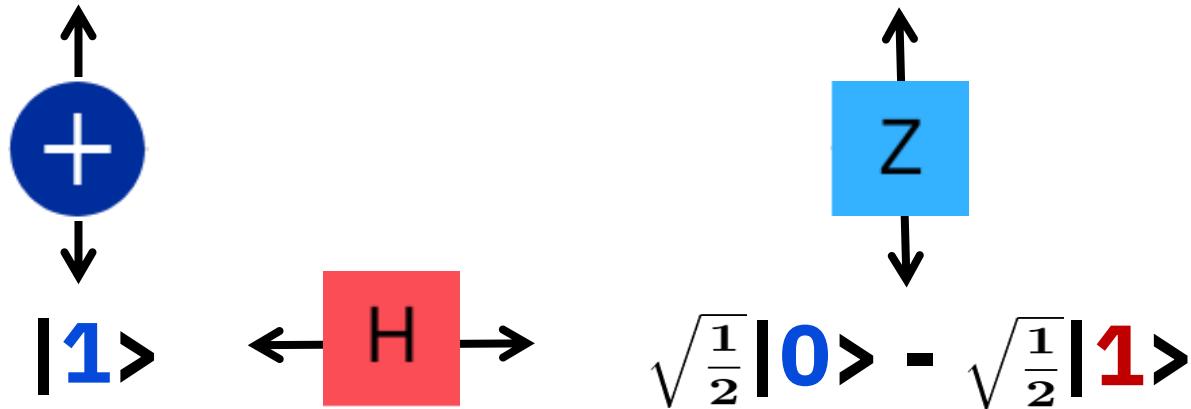
量子コンピューターの計算方法



1 の符号を反転する

量子コンピューターの計算方法

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


$$\begin{array}{c} \text{+} \\ \uparrow \downarrow \\ |1\rangle \end{array} \xleftarrow{H} \sqrt{\frac{1}{2}}|0\rangle - \sqrt{\frac{1}{2}}|1\rangle$$

量子ビットを増やす

$q[0]$ 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。



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。

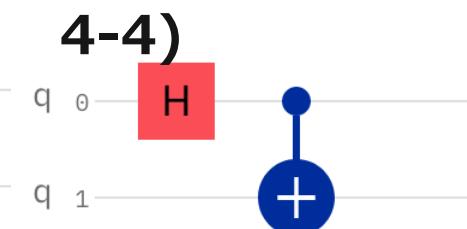
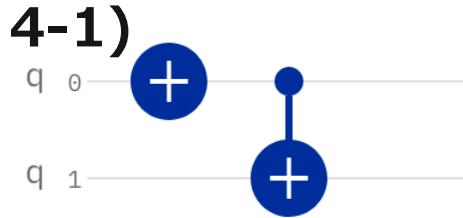


4. CNOTゲート(制御Xゲート)

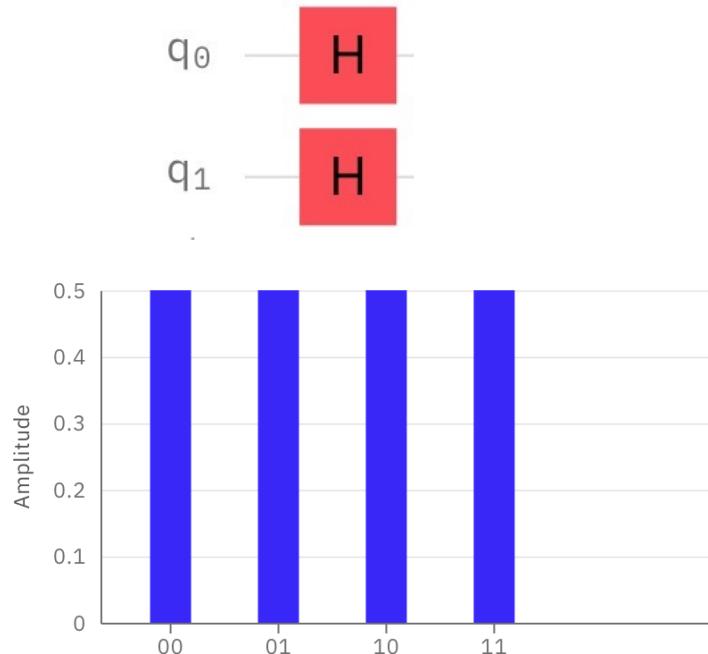
制御ビットが $|1\rangle$ のときのみ、目標ビットを反転（NOT）するゲートです。



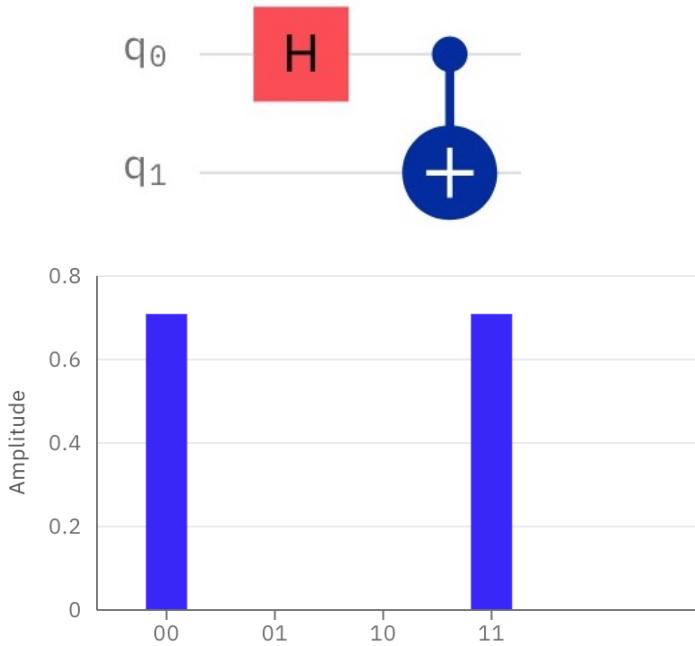
入力		出力	
制御ビット	目標ビット	制御ビット	目標ビット
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0



量子重ね合わせ

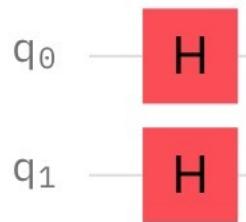


量子もつれ (エンタングルメント)



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



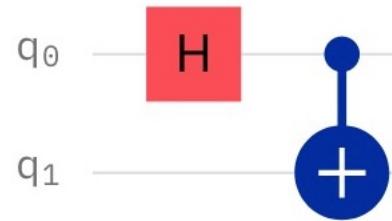
0 0 ... 25%

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

5. 測定

ファイル名を変更
(Entanglementなど)

The screenshot shows the IBM Quantum Compose interface. On the left, the 'Operations' panel displays various quantum gates: H, CNOT, T, S, Z, RZ, RX, RY, RXX, RZZ, U, RCCX, RC3X, and M. A red box highlights the 'Untitled circuit' tab in the top navigation bar. A red circle labeled '(2)' is placed over the 'Saved' button. In the center, a quantum circuit is shown with two qubits (q[0] and q[1]) and one classical register (c[2]). A red box labeled '(1)' encloses a measurement gate (M) on q[1]. A red arrow points from this box to the text '測定ゲートを追加'. On the right, the Qiskit editor shows the generated Python code:

```
1  from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2  from numpy import pi
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])
```

A red circle labeled '(3)' is placed over the 'Setup and run' button in the top right corner, with the text 'クリック' (click) written next to it.

まず量子シミュレーターで実験

Set up and run your circuit

Step 1
Choose a system or simulator

Search by system or simulator name

Total pending jobs 1

63 Qubits

ibmq_qasm_simulator See details

Simulator status • Online

Total pending jobs 1

32 Qubits

simulator_statevector See details

Simulator status • Online

Total pending jobs 1

(4) スクロール

Step 2
Choose your settings

Provider: ibm-q-internal/deployed/default

Shots *: 1024

Job name: e.g. Untitled circuit job

(5) ibmq_qasm_simulator を選択

(6) Run on ibmq_qasm_simulator

Close

IBM Quantum Composer

Untitled circuit Saved File Edit View Visualizations seed 3623 Setup and run

Composer jobs

(7) Jobの確認

Search

Operations

q[0] H c2

q[1] + c2

z

Qiskit

Open in Quantum Lab

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit  
2 from numpy import pi  
3  
4 qreg_q = QuantumRegister(2, 'q')
```



IBM Quantum Composer

Untitled circuit Saved File Edit View Visualizations seed 3623 Setup and run

Composer jobs

View all jobs →

Search jobs

Completed: Jul 21, 2022 8:49 PM ID: 62d93d60aaf3a771... | ibmq_qasm_simulator

(8)

Operations

q[0] H

q[1] +

Qiskit

Open in Quantum Lab

```
1 from qiskit import QuantumRegister, ClassicalRegister,
```

Jobs /
62d93d60aaaf3a771842b9ede

See more details ↗

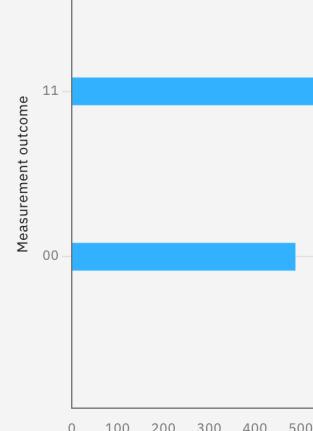
Completed
Jul 21, 2022 8:50 PM (in 8.3s)

Backend
ibmq_qasm_simulator

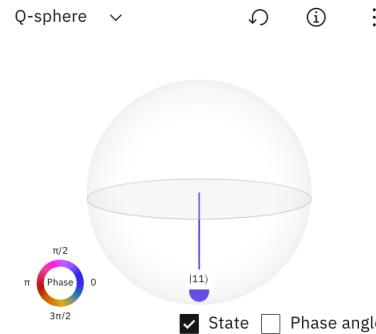
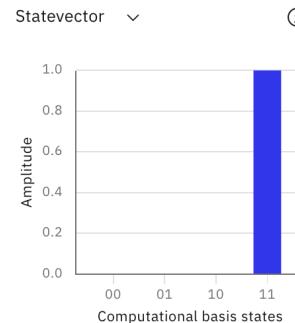
Status timeline Completed

Details

Result - histogram



(9) 測定結果



Untitled circuit Saved

Edit View

Visualizations seed

3623

Setup and run

Read only

(10)

再度実験します

```
1 import numpy as np
2 from qiskit import QuantumRegister, 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])
```

Set up and run your circuit

実機の量子コンピューターで実験

X

Step 1

Choose a system or simulator

Search bar: Search by system or simulator name

ibmq_quito (selected):
System status: Online
Total pending jobs: 7
5 Qubits 16 QV 2.5K CLOPS

ibmq_belem:
System status: Online

(11)

Step 2

Choose your settings

Provider

ibm-q/open/main

Shots *

1024

Job limit: None

Optional

Name your job

(12)

ibm-q/open/main
であることを確認

実デバイスを選択します。

- Total pending jobsが少ない

- QVが大きい

などから選んでみましょう。

Close

Run on ibmq_belem

(13)

IBM Quantum Composer

Jobs / (14) Jobの確認

Untitled circuit Saved File Edit View Visualiz

Completed Jul 21, 2022 8:55 PM (in 1m 8s)

Backend ibmq_qasm_simulator

Status timeline Completed

See more details

Operations

Search

H \oplus \ominus \oplus \ominus \otimes I
T S Z T^\dagger S^\dagger P
RZ r_z $|0\rangle$ i ● if

q[0] H r_z
q[1] + r_z
c2 0 1

Qiskit Open in Quantu

```
1 from qiskit import *
2 ClassicalRegister creg
3 from numpy import *
4 qreg_q = QuantumRegister(2)
5 creg_c = ClassicalRegister(2)
6 circuit =
```



IBM Quantum Composer

View all jobs → Composer jobs

Pending: Jul 21, 2022 9:21 PM ID: 62d944d1e59b9100e90919... | ibmq_belem

Completed: Jul 21, 2022 8:59 PM ID: 62d93fbbaaf3a73b512b9ef4 | ibmq_belem

Operations

Search

H \oplus \ominus \oplus \ominus \otimes I
T S Z T^\dagger S^\dagger P
RZ r_z $|0\rangle$ i ● if

q[0] H r_z
q[1] + r_z
c2 0 1

Qiskit Open in Quantu

```
1 from qiskit import *
2 ClassicalRegister creg
3 from numpy import *
4 qreg_q = QuantumRegister(2)
```



IBM Quantum Composer



Jobs /

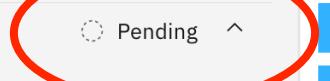
62d944d1e59b9100e90919d8

⋮

待ち時間がある場合

Backend

ibmq_belem

[See more details](#)

Status timeline

- Created: Jul 21, 2022 9:21 PM
- Transpiling: 904ms
- Validating: 892ms
- In queue
- Running
- Completed

Details

Untitled circuit Saved

File

Edit

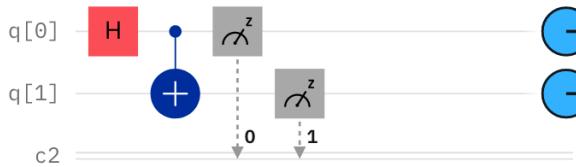
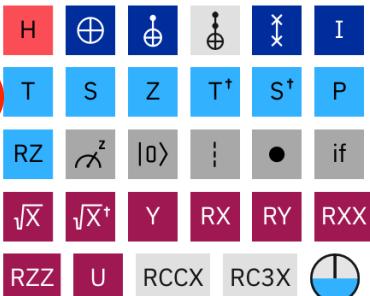
View

Visual

Operations



Search



Qiskit

Open in Quant

```
1 from qis
2 Classics
3 from nu
4 qreg_q :
5 creg_c :
6 circuit
7 circuit
8 circuit
9 circuit
10 circuit
11 circuit
```

[See more details](#)Untitled circuit Saved

File

Edit

View

Visualizations seed

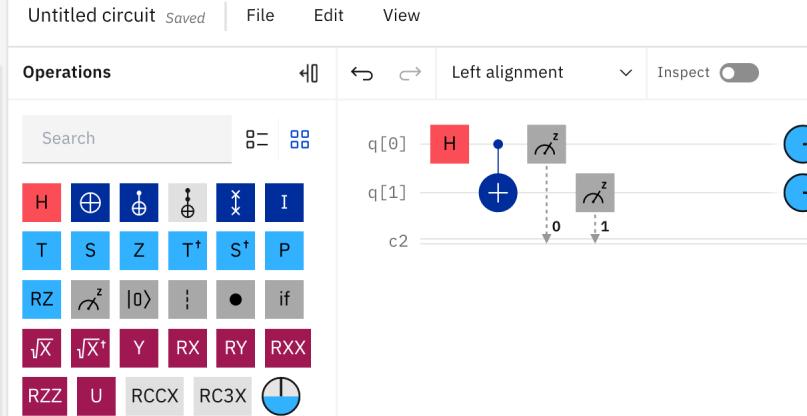
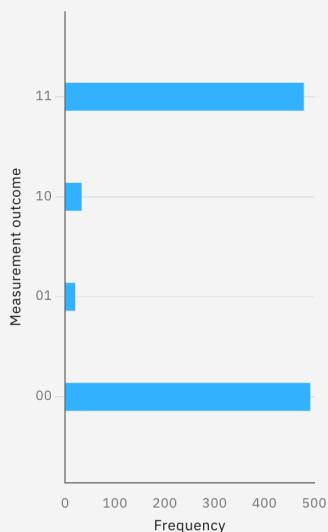
3623

Setup and run

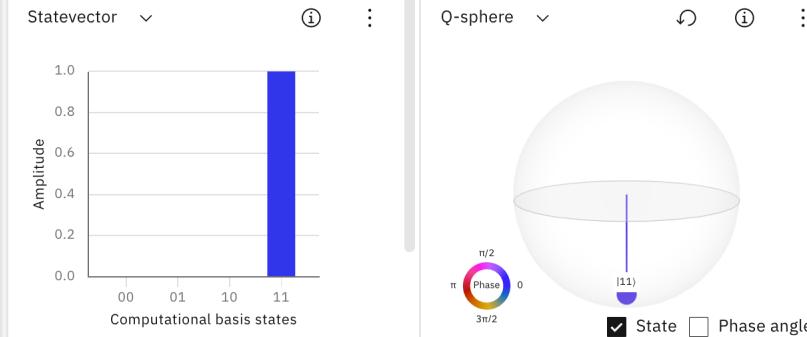
Completed
Jul 21, 2022 9:00 PM (in 38.2s)Backend
ibmq_belemStatus timeline Completed

Details

Result - histogram



(16) 実機での結果



Qiskit Read only

Open in Quantum Lab

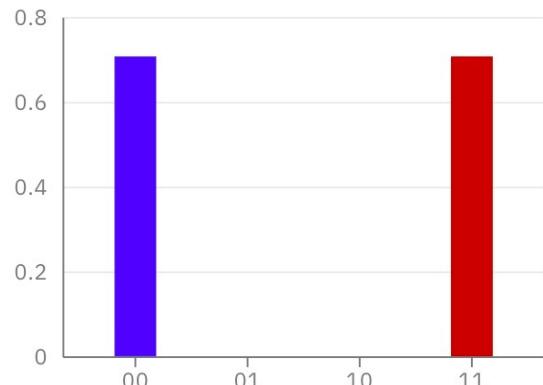
```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(2, 'q')
5 creg_c = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(qreg_q, creg_c)
7
8 circuit.h(qreg_q[0])
9 circuit.cx(qreg_q[0], qreg_q[1])
10 circuit.measure(qreg_q[0],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

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。

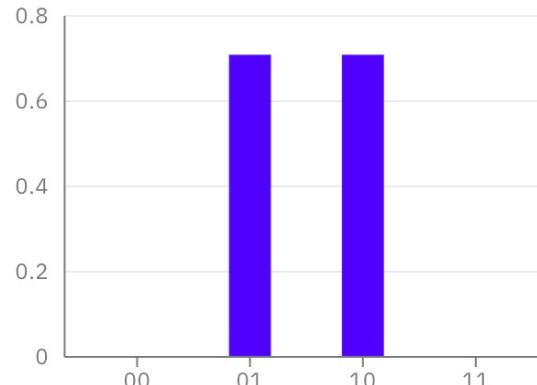
(1)

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



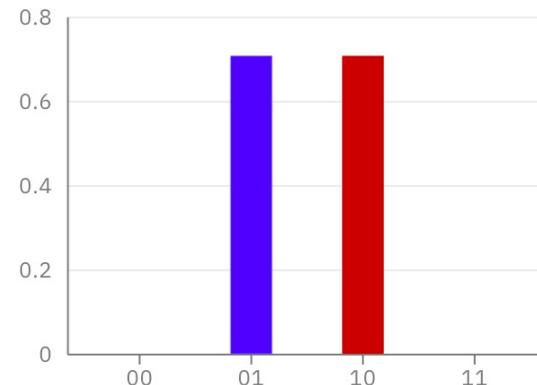
(2)

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

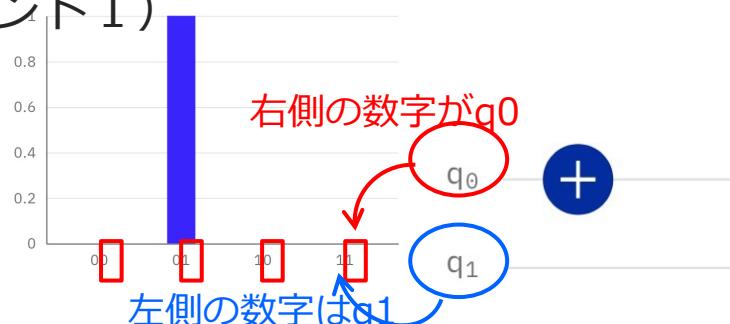


(3)

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

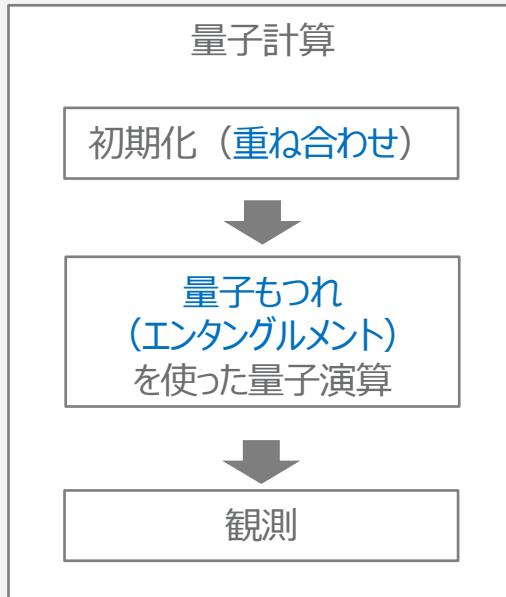


(ヒント1)



(ヒント2)
使うゲートは
X、H、Zにしましょう！

量子アルゴリズム



重ね合わせの状態では全ての出現確率は同じ



量子ビットを干渉させる演算



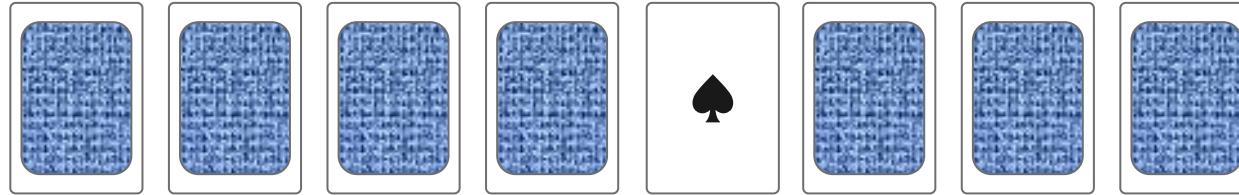
特定のパターンの
確率のみを高める



量子状態の重ね合わせ（並列計算）を
干渉（量子もつれ・エンタングルメント）させて
ほしい解を取り出す。

グローバーのアルゴリズム

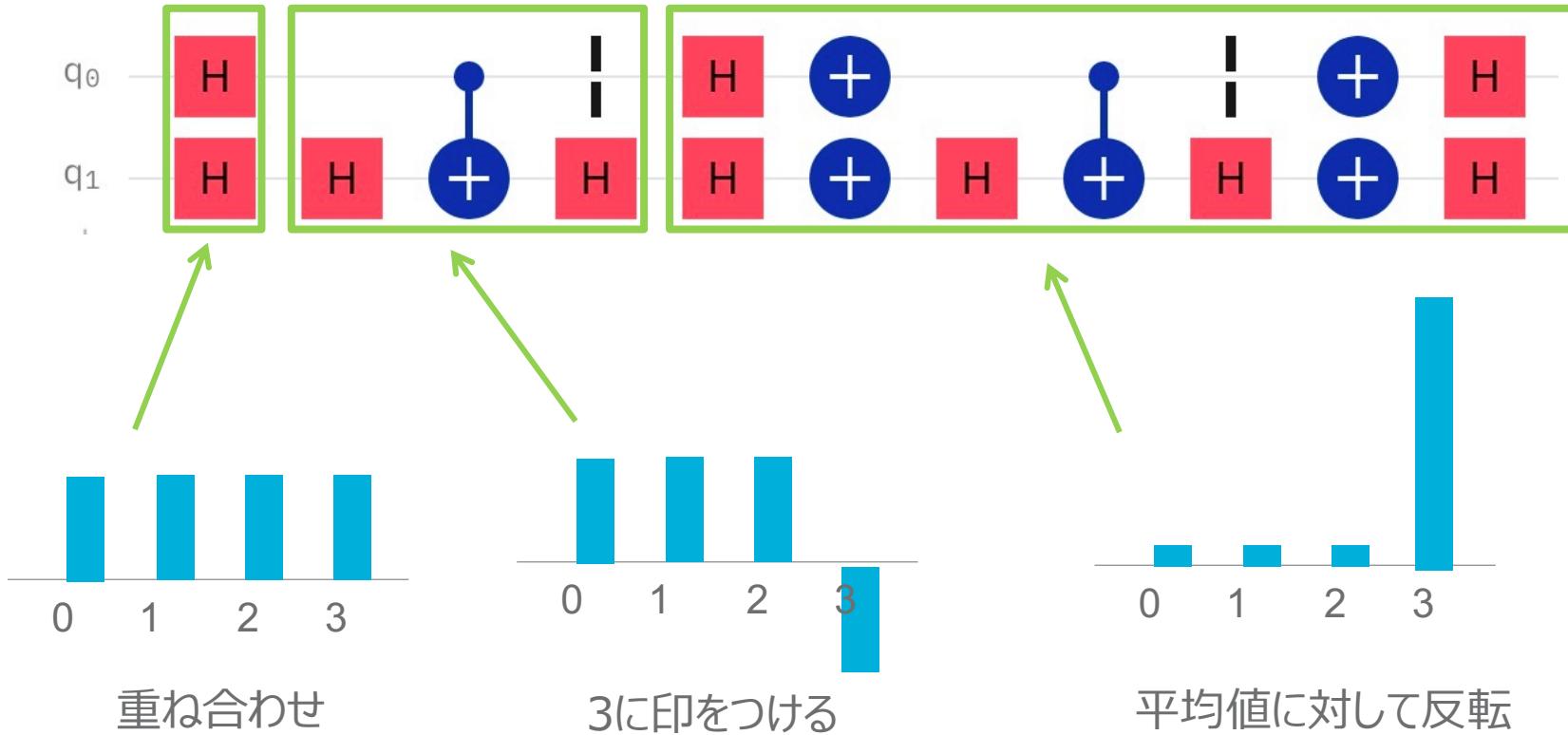
裏返しで置いたトランプをめくって目的のカードを探すとき、古典的手法では、総当たりで一つ一つ探す方法のため、平均して、データの数Nの半分は問い合わせをかけないといけません。



一方、量子コンピューターでは、
おおよそ \sqrt{N} ステップで、目的のカードを探し出すことができます。

グローバー探索（デモ）

0, 1, 2, 3 から 3 を選び出す場合のアルゴリズム



IBM Quantum Challenge 2019

最終問題：自治体のコンビニ出店プランを提示せよ

東京のZ市は11の区域からなる自治体で、すでに4社のコンビニ(A社,B社,C社,D社)が本社の1店舗を別々の区域に展開しています。

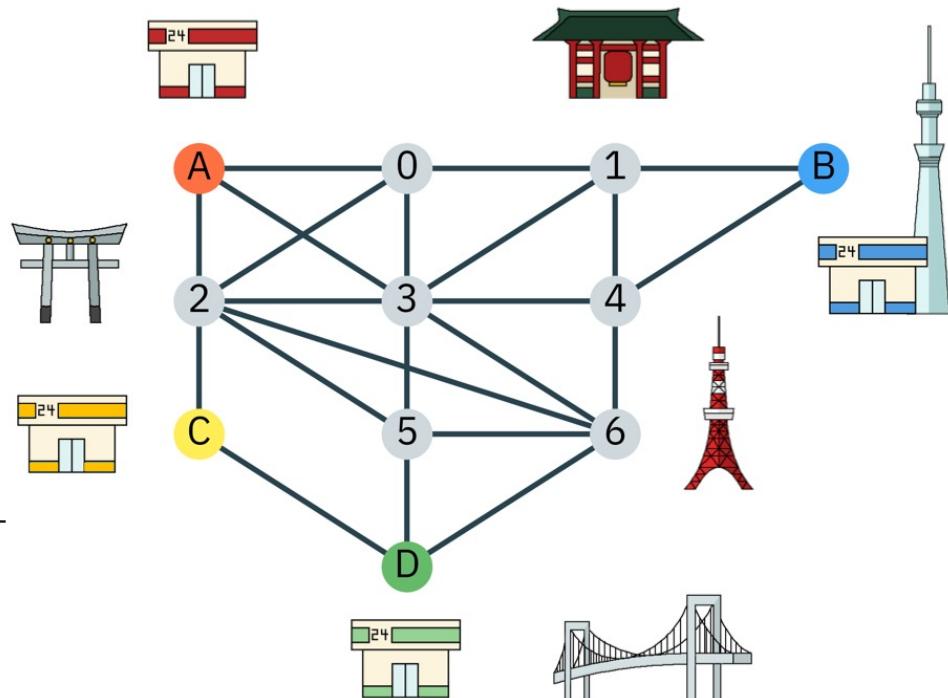
市長のあなたは、残りの7区域にもコンビニを誘致しようとしたが、4社から以下の条件が提示されました。

- ・1つの区域に出店出来るのは1社のコンビニのみ
- ・自社のコンビニは、隣接する区域に自社のコンビニが既に出店している場合は出店しない。

あなたはこれらの条件を満たす出店案を提示できるでしょうか？

グローバーのアルゴリズムを用いて、条件を満たす全ての出店案を列挙してみてください。

正解かつ回路を最も小さくできたチームが優勝！



続きは、Qiskitテキストブックで！



Qiskitを使った量子計算の学習

量子とは？

0. 前提条件

1. 量子状態と量子ビット

1.1 はじめに

1.2 計算の原子

1.3 量子ビット状態を表現する

1.4 単一量子ビットゲート

1.5 量子コンピューターの場合

2. 複数量子ビットともつれ状態

2.1 はじめに

2.2 複数量子ビットともつれ状態

2.3 位相キックバック

2.4 さらなる回路の等価性

2.5 普遍性の証明

2.6 量子コンピューター上の古典計算

3. 量子プロトコルと量子アルゴリズム

3.1 量子回路

3.2 ドイチ-ジョサのアルゴリズム

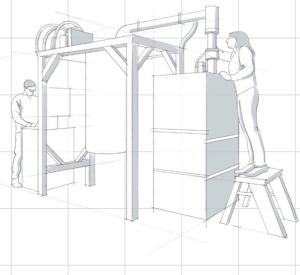
3.3 ペルンシュタイン・ヴァジラニアル

Japanese

The new Qiskit Textbook beta is now available. Try it out now.

Qiskit を使った量子計算の学習

Qiskit Communityチームからのご挨拶です！
Qiskitをベースとした大学の量子アルゴリズム/計算コースの補足教材となるよう、このテキストブックを作り始めました：



1. 量子アルゴリズムの基礎となる数学
2. 今日の非フォールトトレラントな量子デバイスの詳細
3. IBMのクラウド型量子システムに量子アルゴリズムを実装するためのQiskitでのコーディング

テキストブックを読む



Overview

Learn

Community

Document



Overview Learn

5. グローバーのアルゴリズムで数独を解く

この章でこれまで使われていたオラクルは、事前にその解が分かっているものから作成されています。ここでは、グローバーのアルゴリズムを使用して、事前に解を知らないでも解ける単純な問題を解きます。その問題は2x2のバイナリーの数独で、以下の2つのシンプルなルールに基づいています：

- 同じ値を2回含む列はない
- 同じ値を2回含む行はない

数独の各正方形を次の図のような変数に割り当てて：

V_0	V_1
V_2	V_3

巡回路にこの数独の解を出力させたいと思います。

<https://www.youtube.com/@QuantumTokyo>



Quantum Tokyo
日本語解説



ホーム 動画 再生リスト チャンネル フリートーク 概要

アップロード動画 すべて再生

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その1
Kohei Nakamura
4:14 4章 4.1.5章
34:31

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その2
Etsu Arai
3:11 3章 3.11章
6:12

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その3
Etsu Arai
1:07:14

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その4
Kohei Nakamura
4:14 4章 4.1.6章
1:09:06

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その5
Takumi Yamamoto
4:14 4章 4.1.7章
1:29:52

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その6
花畠和也
4:14 4章 4.1.8章
50:05

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その7
花畠和也
1:29:52

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その8
花畠和也
1:29:52

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その9
花畠和也
1:29:52

Quantum Textbook 日本語解説
量子コンピューターハンズオン
アルゴリズム入門編その10
花畠和也
1:29:52

<https://qiskit.org/textbook/ja/preface.html>



Qiskit：より高度なアルゴリズムの実装に

IBM Quantum Composer

Operations

Search

q[0] H

q[1] +

c2

OpenQASM 2.0

```
1 import qiskit
2 import QuantumRegister, ClassicalRegister, QuantumCircuit
3
4 qreg_q = QuantumRegister(2, 'q')
5 creg_c = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(q, 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])
```

Visualizations seed 3623

Setup and run

Qiskit

Read only

Open in Quantum Lab

IBM Quantum Lab

File Edit View Run Kernel Tabs Settings Help

Launcher Untitled circuit_Dec 10, 2023

Code

```
[ ]: from ibm_quantum_widgets import CircuitComposer
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(2, 'q')
creg_c = ClassicalRegister(2, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)
```

IBM Quantum Lab
量子計算用開発キットQiskitを
インストールなしで実行できる
Jupyter Notebook環境

47

本日の内容

- ・量子コンピューターとは
- ・量子情報入門
- ・ハンズオン：IBM Quantum Composerでの実験
『IBM Quantum Composerの登録（くわしいバージョン）』
URL : <https://qiita.com/kifumi/items/7ac33ab7939d2dd796d0>

Qiskitコミュニティによる おすすめイベント・教材

最新のご案内はTwitterから  @qiskit

日本語リンク集：<https://github.com/quantum-tokyo/introduction>

特別デバイス

- [研究者プログラム](#)
- [教育用プログラム](#)

イベント

- [量子プログラミングコンテスト](#)
- [Qiskit夏の学校](#)
- [オープン・サイエンス・プライズ](#)

オンライン教材・翻訳活動

- [Qiskit テキストブック](#)
- [Qiskit ドキュメントチュートリアル](#)
- 翻訳プロジェクト([テキストブック](#)・[ドキュメント](#))

認定制度

- [Qiskit Advocate プログラム](#)
- [Qiskit デベロッパー認定制度](#)

Thank you

Kifumi Numata

kifumi@jp.ibm.com

© Copyright IBM Corporation 2022. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and ibm.com are trademarks of IBM Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at [Copyright and trademark information](#).