日本語訳『Qiskit Textbook』勉強会
第3章 3.9 ショアのアルゴリズム

Kifumi Numata

3.9章 ショアのアルゴリズム
目次

ショアのアルゴリズムは、
多項式時間で
整数を因数分解できる
ことで有名。

暗号RSAは、
十分大きな整数の場合に、
古典アルゴリズムで
因数分解が不可能であることを
前提としている。

この章では、ショアのアルゴリズムの量子部分に焦点を当てます。
つまり、周期発見の問題を紹介します。

Qt

5. 周期発見から因数分解へ
例：15の素因数分解

N= 15 をショアのアルゴリズムで素因数分解する。

(1) N が偶数でないことを確認。→偶数だったら、素因数2を出力。
(2) 1 から $N-1$ の間の整数乱数 a を選択します。
(3) a と N の最大公約数が1であることを確認。→ aがNの因数を持っていたら出力。

ここを
量子計算

(4) 以下の式を満たす 位数r を見つける（a と N に対して位数発見アルゴリズムを実行すると出る。）

$$a^r \bmod N = 1$$

（左式を満たす最小の自然数 r を位数という。）

：a^rをNで割ると余りが１の意味

(5) r が偶数だったら、以下より素因数が求まる。→奇数だったら(2)に戻る。

$$(a^r - 1) \bmod N = 0$$

：a^r -1をNで割ると割り切れる。

$$a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$$

：つまりNは、$(a^{r/2} - 1)$ か $(a^{r/2} + 1)$ と
最大公約数を持つ。(pythonのmathモジュールで)
→ Nの素因数が求まる。（ユークリッドの互除法）

1. 周期発見

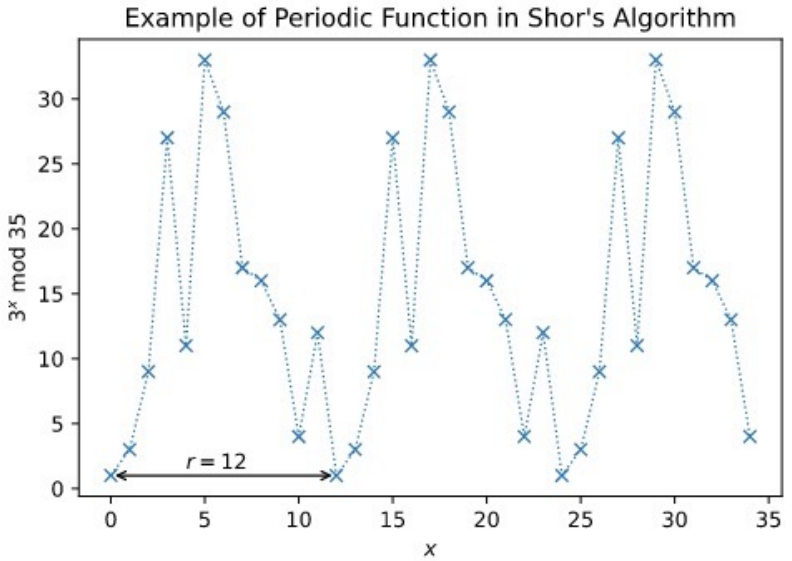
この関数は、周期関数です。

$$f(x) = a^x \bmod N$$

モジュロ演算

a^r をNで割った余り

例えば、 $f(x) = 3^x \bmod 35$ のグラフ



f(0)=1
f(1)=3
f(2)=9
f(3)=27
f(4)=11
f(5)=33
f(6)=29
f(7)=17
f(8)=16
f(9)=13
f(10)=4
f(11)=12
f(12)=1
…

x=12で周期的な関数になっていることがわかります。

つまり $3^x \bmod 35 = 1$ を満たす位数 x は12。

$$3^{12} - 1 = (3^6 - 1)(3^6 + 1) = 728 \cdot 730$$

35がこのどちらかと最大公約数を持つ

よって、Nを因数分解したい時には、

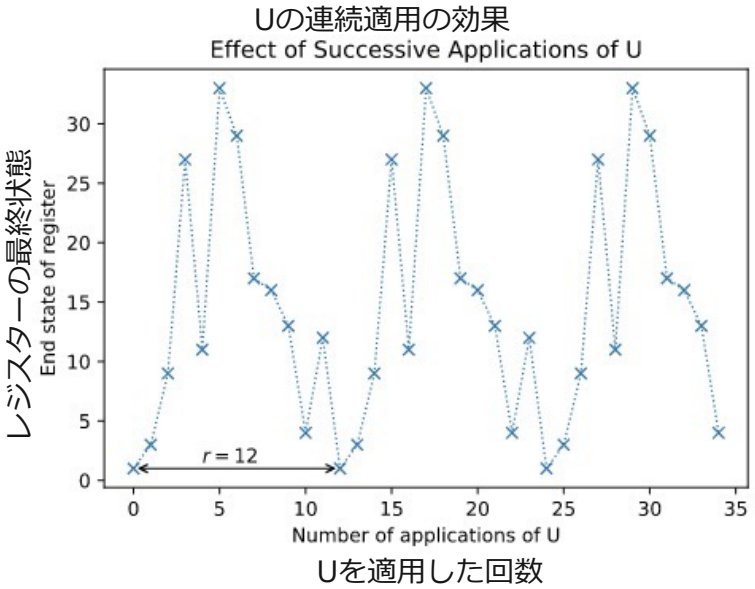
$a^r \bmod N = 1$ の位数を求めればよい。

周期関数 $f(x) = a^x \bmod N$ と同じ働きをするユニタリー演算子Uは

$$U|y\rangle \equiv |ay \bmod N\rangle$$ という形です。

このユニタリー演算子の働きを確認してみると…
a=3 およびN=35 の場合、|1⟩の状態から開始して

$$U|1\rangle = |3\rangle \quad = |3 \bmod 35>$$
$$U^2|1\rangle = |9\rangle \quad = U|3> = |3\cdot 3 \bmod 35>$$
$$U^3|1\rangle = |27\rangle \quad = U|9> = |3\cdot 9 \bmod 35>$$
$$\vdots$$
$$U^{(r-1)}|1\rangle = |12\rangle \quad = U|4> = |3\cdot 4 \bmod 35>$$
$$U^r|1\rangle = |1\rangle \quad = U|12> = |3\cdot 12 \bmod 35>$$



Uをr 回、つまり12回適用すると、再び状態|1⟩になります。
つまりこのユニタリー演算子Uの周期を求めれば良いことがわかります。

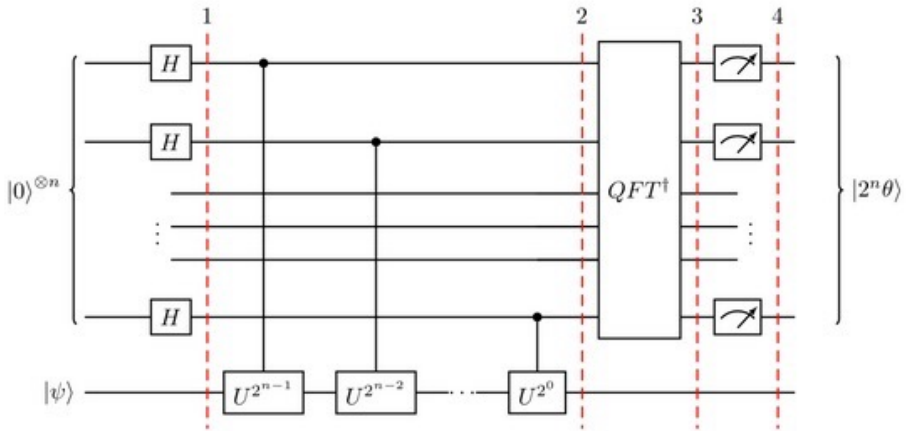
ユニタリー演算子U： $U|y\rangle \equiv |ay \bmod N\rangle$　の固有値と固有状態は、（次ページで解説）

固有値： $U|u_s\rangle = e^{\frac{2\pi i s}{r}}|u_s\rangle$　　（sは、0<s<r−1である整数, rは $a^r \bmod N = 1$の位数）

固有状態： $|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}} |a^k \bmod N\rangle$

ショアのアルゴリズムでは、このユニタリー演算子Uについて量子位相推定をして、位相 $\phi = \frac{s}{r}$ を求めます。

$U|\psi\rangle = e^{\mathbf{2\pi i\theta}}|\psi\rangle$ のθを求める位相推定の量子回路（3.8章より）



$$\phi = \frac{s}{r}$$

からrは簡単に求められる。

ユニタリー演算子U： $U|y\rangle \equiv |ay \bmod N\rangle$　の固有値と固有状態が以下のようにかけることの証明

固有値： $U|u_s\rangle = e^{\frac{2\pi i s}{r}}|u_s\rangle$　　（sは、0<s<r−1である整数, rは　$a^r \bmod N = 1$ の位数）

固有状態： $|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}}|a^k \bmod N\rangle$

a=3 、 N=35のときの例 (r=12なので)

$$|u_s\rangle = \frac{1}{\sqrt{12}}(|1\rangle + e^{-\frac{2\pi i s}{12}}|3\rangle + e^{-\frac{4\pi i s}{12}}|9\rangle \cdots + e^{-\frac{20\pi i s}{12}}|4\rangle + e^{-\frac{22\pi i s}{12}}|12\rangle)$$

$$U|u_s\rangle = \frac{1}{\sqrt{12}}(|3\rangle + e^{-\frac{2\pi i s}{12}}|9\rangle + e^{-\frac{4\pi i s}{12}}|27\rangle \cdots + e^{-\frac{20\pi i s}{12}}|12\rangle + e^{-\frac{22\pi i s}{12}}|1\rangle)$$

$$U|u_s\rangle = e^{\frac{2\pi i s}{12}}\frac{1}{\sqrt{12}}(e^{-\frac{24\pi i s}{12}}|1\rangle + e^{-\frac{2\pi i s}{12}}|3\rangle + e^{-\frac{4\pi i s}{12}}|9\rangle + e^{-\frac{6\pi i s}{12}}|27\rangle \cdots + e^{-\frac{22\pi i s}{12}}|12\rangle)$$

$$U|u_s\rangle = e^{\frac{2\pi i s}{12}}|u_s\rangle$$

$$U|1\rangle = |3\rangle$$
$$U^2|1\rangle = |9\rangle$$
$$U^3|1\rangle = |27\rangle$$
$$\vdots$$
$$U^{11}|1\rangle = |12\rangle$$
$$U^{12}|1\rangle = |1\rangle$$

これは、a=3 、 N=35の場合でなくても成り立つことから、Uの固有値と固有状態が上記のように書けることがわかります。

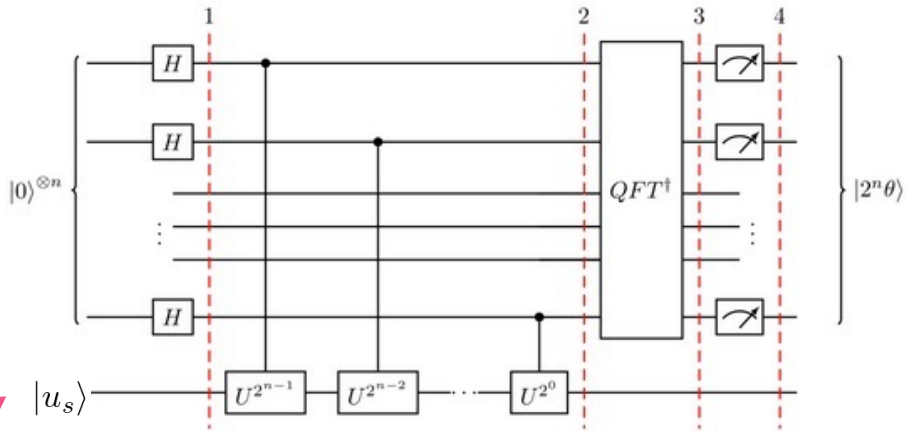
ユニタリー演算子U： $U|y\rangle \equiv |ay \bmod N\rangle$

固有値： $U|u_s\rangle = e^{\frac{2\pi i s}{r}}|u_s\rangle$　　（sは、0<s<r−1である整数, rは $a^r \bmod N = 1$の位数）

固有状態： $|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}}|a^k \bmod N\rangle$

ショアのアルゴリズムでは、このユニタリー演算子Uについて量子位相推定をして、位相 $\phi = \frac{s}{r}$ を求めます。

$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$ のθを求める位相推定の量子回路（3.8章より）



$$\phi = \frac{s}{r}$$

からrは簡単に求められる。

これで位相推定をしたいのですが、固有状態 $|u_s\rangle$が分からないと位相推定の回路が作れません。

実は、固有状態をすべて合計すると、
|1⟩ を除くすべての計算基底の状態がキャンセルされます：

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle$$

a=7 、 N=15のときの例。r=4の場合に：

$$|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-\frac{2\pi isk}{r}}|a^k \bmod N\rangle$$

k=0　　k=1　　k=2　　k=3

$$\frac{1}{2}(\quad |u_0\rangle = \frac{1}{2}(|1\rangle \quad +|7\rangle \quad +|4\rangle \quad +|13\rangle)\ldots$$

+1

s=0

$$+|u_1\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{2\pi i}{4}}|7\rangle + e^{-\frac{4\pi i}{4}}|4\rangle + e^{-\frac{6\pi i}{4}}|13\rangle)\ldots$$

-1

s=1

$$+|u_2\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{4\pi i}{4}}|7\rangle + e^{-\frac{8\pi i}{4}}|4\rangle + e^{-\frac{12\pi i}{4}}|13\rangle)\ldots$$

-i

s=2

$$+|u_3\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{6\pi i}{4}}|7\rangle + e^{-\frac{12\pi i}{4}}|4\rangle + e^{-\frac{18\pi i}{4}}|13\rangle) \quad) = |1\rangle$$

+i

s=3

$|u_s\rangle$ のかわりに|1⟩ を入れて、位相推定します。（ここで1は0,1の1ではなく、整数の1）

2.周期発見の解法

整数Nを因数分解したいとき、aを適当に選んで（a<N）

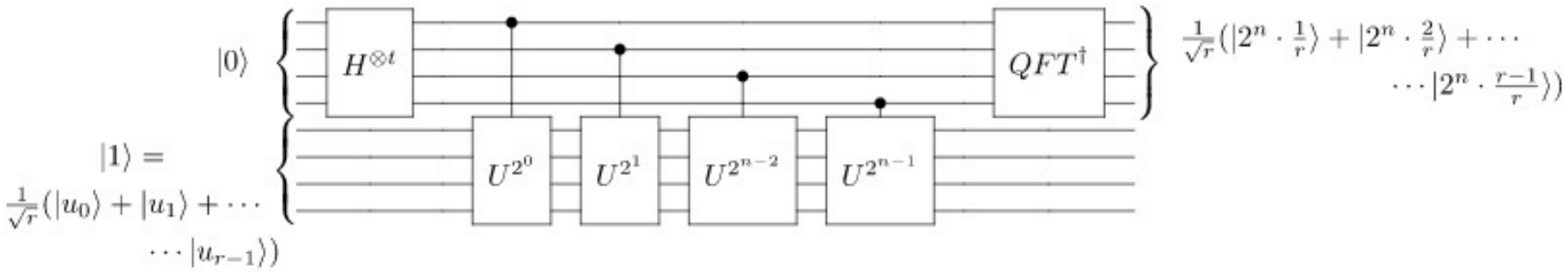
周期関数 $f(x) = a^x \bmod N$ と同じ働きをするユニタリー演算子Uについて、位相推定をする。

$U|y\rangle \equiv |ay \bmod N\rangle$ の固有状態と固有値は、

$U|u_s\rangle = e^{\frac{2\pi i s}{r}}|u_s\rangle$ （sは、0<s<r−1である整数）

$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}} |a^k \bmod N\rangle$

量子位相推定で、位相 $\phi = \frac{s}{r}$ を求めます。



$|u_s\rangle$ のかわりにその重ね合わせの|1⟩ を入れて、位相推定する。
（答えも r 個の重ね合わせになる。）

3. Qiskitでの実装

3.8章の位相推定では、UがZ軸回転で回転角θもわかっている時の例だったが、今回はUを自分で作る。

この例では、a=7 とN=15 の周期発見問題を解きます。 $U|y\rangle = |7y \bmod 15\rangle$

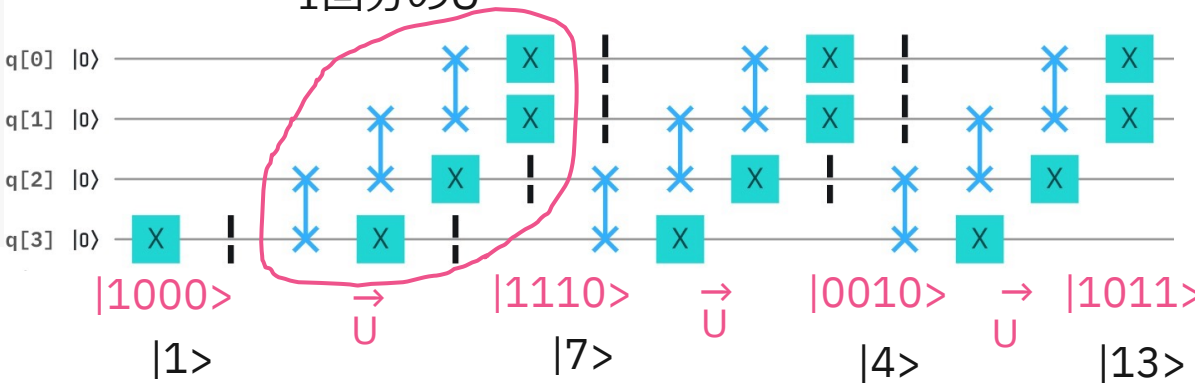
$$U_7 : \begin{cases} |1\rangle \rightarrow |7\rangle \rightarrow |4\rangle \rightarrow |13\rangle \rightarrow |1\rangle \\ |2\rangle \rightarrow |14\rangle \rightarrow |8\rangle \rightarrow |11\rangle \rightarrow |2\rangle \end{cases}$$

Multiplication by 7 modulo 15

U の実装(ビット配列は教科書順)

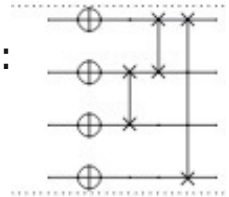
```
U = QuantumCircuit(4)
a = 7

U.swap(2,3)
U.swap(1,2)
U.swap(0,1)

for q in range(4):
    U.x(q)
```

1回分のU



ここはビット並びが教科書順！

Uの作り方は
いろいろ考えられる例：



Markov and Saeedi 2012 https://arxiv.org/pdf/1202.6614.pdf

a=7以外に拡張：

$U|y\rangle \equiv |ay \bmod N\rangle$ で、N=15の場合のUゲートを作って制御Uにする

```
def c_amod15(a, power):
    """ a mod 15の制御ゲートをかける"""
    if a not in [2,7,8,11,13]:
        raise ValueError("'a' must be 2,7,8,11 or 13")
    U = QuantumCircuit(4)
    for iteration in range(power):
        if a in [2,13]:
            U.swap(0,1)
            U.swap(1,2)
            U.swap(2,3)
        if a in [7,8]:
            U.swap(2,3)
            U.swap(1,2)
            U.swap(0,1)
        if a == 11:
            U.swap(1,3)
            U.swap(0,2)
        if a in [7,11,13]:
            for q in range(4):
                U.x(q)
    U = U.to_gate()
    U.name = "%i^%i mod 15" % (a, power)
    c_U = U.control()
    return c_U
```

15の素因数はNG。

4, 14がなぜ含まれていないか説明なし。

（14は因数が出てこない。）

Power回繰り返す

U^xを作成するには、この回路をx 回繰り返します。

a=7のとき

a=7のとき

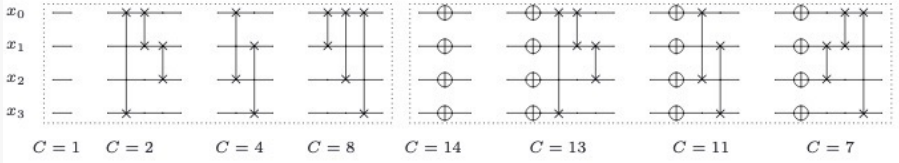
Uのゲートを作る

ゲートの名前

作ったゲートを制御ゲートにする

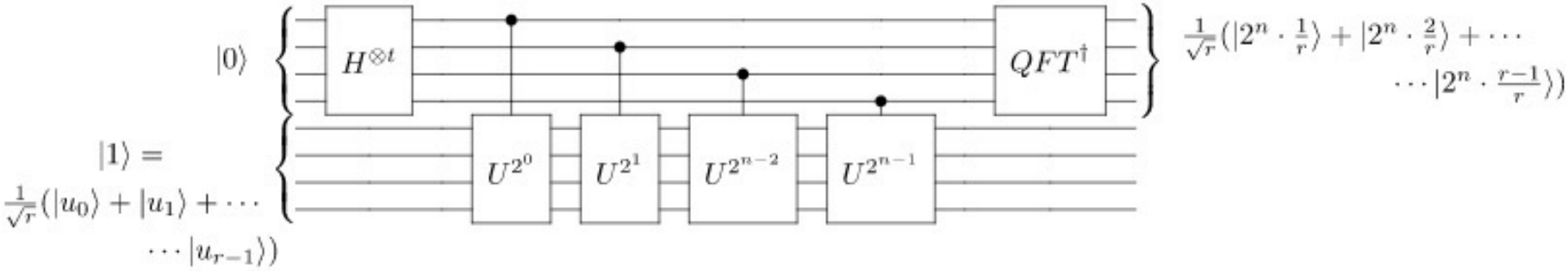
Uの回路例（色々な作り方があるようだ）

a=7



Markov and Saeedi 2012 https://arxiv.org/pdf/1202.6614.pdf

Uゲートの作り方は、
後ほど作り方を説明すると書いてあるが
特に詳しく説明はない。



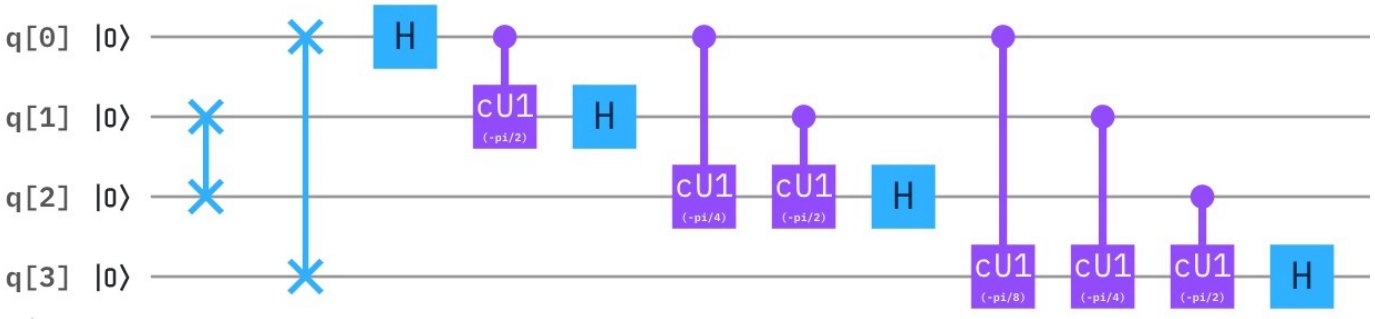
U^x を作成するには、このUの回路を x 回繰り返す。さらに制御Uにする必要がある。

逆QFTの回路（詳細は、3.7章 量子フーリエ変換を参照）：ビット配列はQiskit順

```
def qft_dagger(n):
    """n量子ビットの逆QFTを回路の最初のn量子ビットにかける"""
    qc = QuantumCircuit(n)
    # Swapsを忘れない!
    for qubit in range(n//2):
        qc.swap(qubit, n-qubit-1)
    for j in range(n):
        for m in range(j):
            qc.cu1(-np.pi/float(2**(j-m)), m, j)
        qc.h(j)
    qc.name = "QFT†"
    return qc
```

整数の割り算（切り捨て）

cU1(回転角,制御,標的)ゲート



ショアのアルゴリズム　（a=7、N=15の周期発見問題）

```
# 変数を指定
n_count = 4 # カウント用量子ビットの数
a = 7

# Create QuantumCircuit 量子回路を作成
qc = QuantumCircuit(4+n_count, n_count)

# カウント用量子ビットを|+>状態に初期化
for q in range(n_count):
    qc.h(q)

# アンシラレジスターを|1>の状態にする
qc.x(3+n_count)

# 制御Uを操作
for q in range(n_count):
    qc.append(c_amod15(a, 2**q),
              [q] + [i+n_count for i in range(4)])

# 逆QFTを操作
qc.append(qft_dagger(n_count), range(n_count))

# 回路を測定
qc.measure(range(n_count), range(n_count))
qc.draw('text')
```

Textbookより少なめにしました

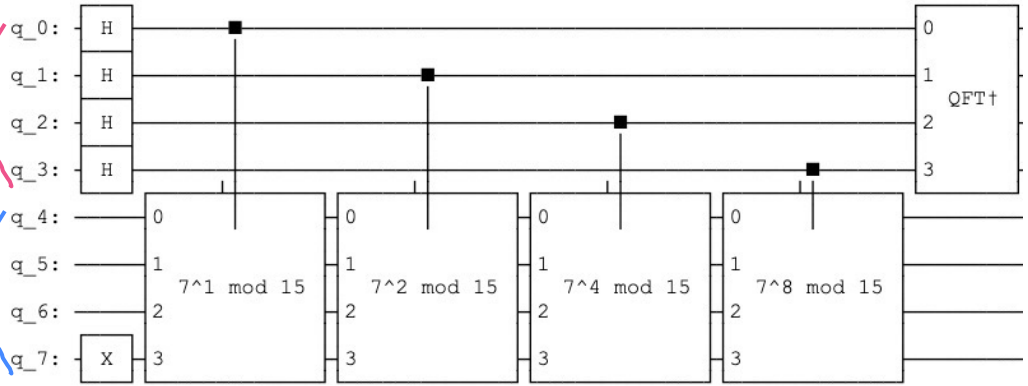
$U|y\rangle = |7y \bmod 15\rangle$ の回路なので2^4≧14

qはカウント用ビットの量子ビット数

回路図は切れているが最後に上４ビットを測定

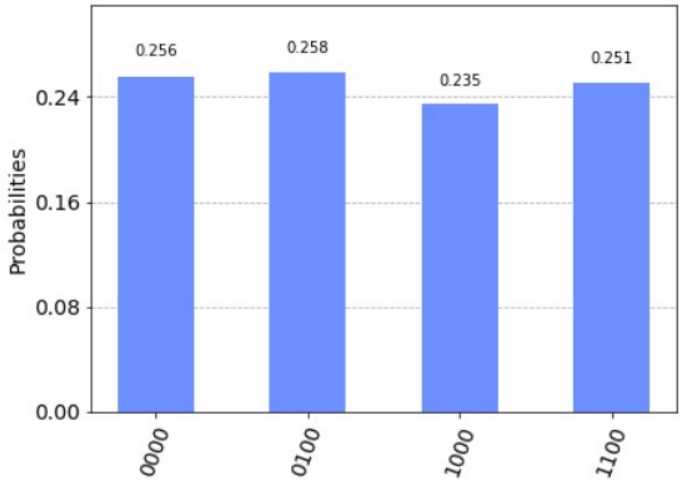
n_count = 4：ここはQiskitのビット配列

一番下が|1>：つまり教科書ビット配列



ショアのアルゴリズムのシミュレーターでの実行結果

```
backend = Aer.get_backend('qasm_simulator')
results = execute(qc, backend, shots=2048).result()
counts = results.get_counts()
plot_histogram(counts)
```



3.8章 量子位相推定と同じく
最後にN=2^nで割る

```
rows, measured_phases = [], []
for output in counts:
    decimal = int(output, 2)  # 2進数の文字列を10進数に変換
    phase = decimal/(2**n_count) # 対応する固有値を見つける
    measured_phases.append(phase)
    # これらの値をテーブルの行に追加：
    rows.append(["%s(bin) = %i(dec)" % (output, decimal),
                 "%i/%i = %.2f" % (decimal, 2**n_count, phase)])
# tabulateを使用して、これをASCIIテーブルとして行を印刷：
print(tabulate(rows, headers=["Register Output", "Phase"],
               colalign=["left","right"]))
```

Register Output	Phase
1000(bin) = 8(dec)	8/16 = 0.50
0100(bin) = 4(dec)	4/16 = 0.25
0000(bin) = 0(dec)	0/16 = 0.00
1100(bin) = 12(dec)	12/16 = 0.75

Phase	Fraction	Guess for r
0.75	3/4	4
0.5	1/2	2
0.25	1/4	4
0	0/1	1

$$U|y\rangle = |7y \bmod 15\rangle$$

今、上記のUの固有値 $e^{\frac{2\pi i s}{r}}$
の量子位相推定をして、

位相 $\phi = \frac{s}{r}$ を求めた。

ここから、
r=4と推定される。

5. 周期発見から因数分解へ
例：15の素因数分解

N= 15 をショアのアルゴリズムで素因数分解する。

(1) N が偶数でないことを確認。→偶数だったら、素因数2を出力。
(2) 1 から $N-1$ の間の整数乱数 a を選択します。→ 今回はa=7が選択された
(3) a と N の最大公約数が1であることを確認。→ aがNの因数を持っていたら出力。
(4) 以下の式を満たす 位数r を見つける（a と N に対して位数発見アルゴリズムを実行すると出る。）

$$a^r \bmod N = 1$$

（左式を満たす最小の自然数 r を位数という。）

：a^rをNで割ると余りが１の意味

→ r=4が求まった

(5) r が偶数だったら、以下より素因数が求まる。→奇数だったら(2)に戻る。

$$(a^r - 1) \bmod N = 0$$

：a^r をNで割ると割り切れる。

$$a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$$

：つまりNは、$(a^{r/2} - 1)$ か $(a^{r/2} + 1)$ と最大公約数を持つ。

→ (47と15), (49と15)の最大公約数より 3,5 が15素因数として求まった

4. 剰余指数化

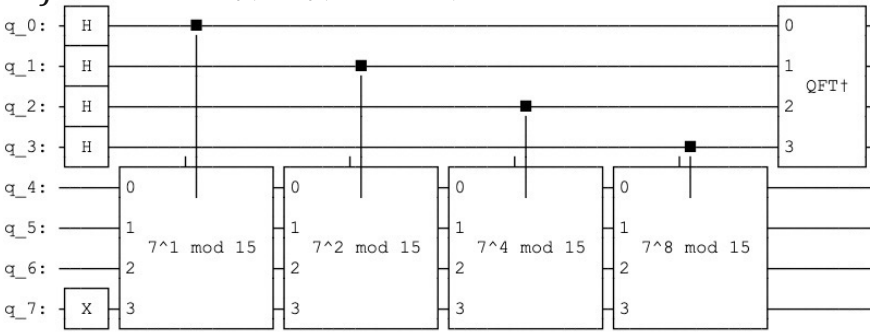
Uを繰り返すことによってU^{2^j}ゲートを作成する方法は、jとともに指数関数的に増加し、
多項式時間アルゴリズムにはならない。

$$U^{2^j}|y\rangle = |a^{2^j} y \bmod N\rangle$$

U^{2^j}の演算は　$a^{2^j} \bmod N$（剰余指数とよぶ）

をかけたものと同じ。

これは古典で簡単に計算できる。




```
In [13]: def a2jmodN(a, j, N):
             """二乗を繰り返してa^{2^j} (mod N) を計算"""
             for i in range(j):
                 a = np.mod(a**2, N)
             return a

In [14]: a2jmodN(7, 2049, 53)

Out[14]: 47
```

Pythonで効率的なアルゴリズムが可能であれば、
量子コンピューターで同じアルゴリズムを使用可能。

ただし、jで多項式にスケーリングしても、剰余指数回路は単純ではなく、
ショアのアルゴリズムのボトルネックになっています。

初心者にやさしい実装は、参考文献[1]にあります。

[1] Stephane Beauregard, *Circuit for Shor's algorithm using 2n+3 qubits,* arXiv:quant-ph/0205095

整数N (N≦2^n (n量子ビット))の素因数分解を2n+3量子ビットで行う

振り返り：位数発見アルゴリズムは、位相推定の応用版

位相推定のときは、 Uが**Z軸回転で回転角θもわかっている時の例だったが、今回はUを自分で作る。**

周期関数 $f(x) = a^x \bmod N$ と同じ働きをするユニタリー演算子U

$$U|y\rangle \equiv |ay \bmod N\rangle$$ の固有状態と固有値は、

$$U|u_s\rangle = e^{\frac{2\pi i s}{r}}|u_s\rangle$$ （sは、$0<s<r-1$である整数）

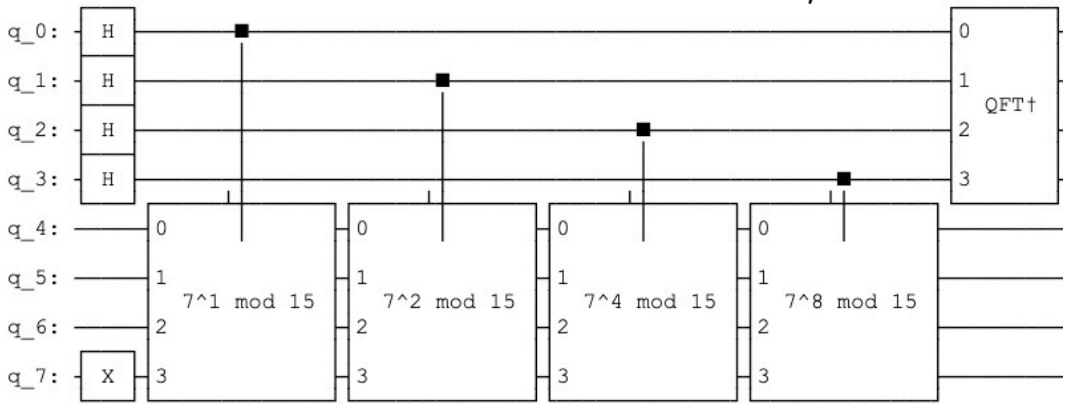
$$|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-\frac{2\pi i s k}{r}}|a^k \bmod N\rangle$$

このユニタリー演算子Uについて量子位相推定をして、

位相 $\phi = \frac{s}{r}$ を求めます。

rを求めることで、

Nを素因数分解できます。



作ったUの回路を繰り返し、さらに制御バージョンにする。

Thank you

Kifumi Numata

kifumi@jp.ibm.com

© Copyright IBM Corporation 2020. 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.