Quantum Computing 6 – Quantum Fourier Transform and Applications

Angelo Bassi

is to *transform* it into some other problem for which a solution is known. There are a fiest which appear the term and in so many different contexts and in so many different contexts. a quantum computer than on a classical computer, a discovery which has enabled the The Quantum Fourier Transform

The discrete Fourier transform takes as input a vector of complex numbers, x₀, . . . , x_{N −1} where the length N of the vector is a fixed parameter. It outputs the transformed data, a vector of complex numbers as in put a vector of complex numbers, *numbers* y_0 , ..., y_{N-1} , defined by a quantum computer than on a classical computer than on a classical computer, a discovery which has enabled th The **discrete Fourier transform** takes as input *x*0*,...,x^N*−¹ where the length *N* of the vector is a fixed parameter. It outputs the The **discrete Fourier transform** takes as input a vector of cor nambers, x_0 , ..., x_N ₋₁ where the length is or the vector is a fixed parameter. It outputs the transformed data, a vector of complex ——
∟ *numbers* y_0 , ..., y_{N-1} , defined by *that for which it was intended.* albortete Tourne

$$
y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i jk/N}
$$

conventional notation for the quantum Fourier transform is somewhat different. The *n* and **quantum Fourier transform**, we do a bill on the amplitudes of a quantum state of a quantum state states, with the basis states, $\frac{1}{2}$ *|* In the **quantum Fourier transform, we do a DFT on the amplitudes** *N* $\frac{1}{100}$ *In the quantum Fourier transform, we do a DFT on the amplitudes* in the **quantum Fourier transform)**, we do a DFT on the amplitudes *x*N−1 where the length *N* of the vector is a fixed parameter. It outputs the vector i

$$
\sum_{j=0}^{N-1} x_j \overline{\underline{\left|j\right\rangle}} \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle
$$

 N_k are the dist amplitudes x_j . On the **computational basis** it reads where the amplitudes y_k are the discrete Fourier transform of the $\frac{1}{\sqrt{2}}$ where the amplitudes y_k are the discrete Fourier transform of the

$$
|j\rangle \longrightarrow \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}e^{2\pi ijk/N}|k\rangle
$$

It is not obvious from the definition, but this transformation is a

lupitary transformation, and thus san he implemented as the unitary transformation, and thus can be implemented as the namics for a quantum c dynamics for a quantum computer.

Since the output amplitudes are a linear combination of the input amplitudes, it is a linear operator. Its form is easy to write down

$$
\hat{F} = \sum_{j,k=0}^{N-1} \frac{e^{2\pi i jk/N}}{\sqrt{N}} |k\rangle\langle j|
$$

It is easy to check that it is unitary isv to chec \overline{c} λ nat it is unitary

$$
\hat{F}^{\dagger}\hat{F} = \frac{1}{N} \sum_{j,k,j',k'} e^{2\pi i (j'k'-jk)/N} |j\rangle\langle j'|\delta_{kk'}
$$

$$
= \frac{1}{N} \sum_{j,k,j'} e^{2\pi i (j'-j)k/N} |j\rangle\langle j'|
$$

$$
= \sum_{j,j'} |j\rangle\langle j'|\delta_{jj'} = \sum_j |j\rangle\langle j| = \hat{I}.
$$

The Fourier transform lets us define a new basis: $|\tilde{x}\rangle$ = F $|x\rangle$, where $\{ |x\rangle \}$ is the usual computational basis. This basis has a number of interesting properties. Every vector $|\tilde{x}\rangle$ is an equally weighted superposition of all the computational basis states: all the computational basis states:

$$
|\langle \tilde{x} | y \rangle|^2 = \langle y | \tilde{x} \rangle \langle \tilde{x} | y \rangle = \langle y | \hat{F} | x \rangle \langle x | \hat{F}^{\dagger} | y \rangle
$$

=
$$
\frac{e^{2\pi i x y/N}}{\sqrt{N}} \frac{e^{-2\pi i x y/N}}{\sqrt{N}} = \frac{1}{N}.
$$

From the point of view of physics, the relationship of this basis to $\frac{1}{11}$ the point of view of priysics, the relationship of this the computational basis is analogous to that between the momentum and position bases of a particle.

Recall that the Hadamard transform could also turn computational basis states into equally weighted superpositions of all states. But it left all amplitudes real, while the amplitudes of $|\tilde{x}\rangle$ are complex. 2

Circuit for the Quantum Fourier Transform

At this point we specialize to the case of n qubits, so the dimension is $N = 2ⁿ$.

We have seen that the quantum Fourier transform is a unitary operator. Therefore, by our earlier results, there is a quantum circuit which implements it. However, there is no guarantee that this circuit will be efficient! A general unitary requires a circuit with a number of gates exponential in the number of bits.

Very fortunately, in this case an efficient circuit does exist. (Fortunately, because the Fourier transform is at the heart of the most impressive quantum algorithms!)

Binary expression: $j \rightarrow j_1 j_2 ... j_n$

$$
j = j_1 2^{n-1} + j_2 2^{n-2} + \dots + j_n
$$

For example, for $n = 2$ we have $j = 2 j_1 + j_2$, therefore

$$
0 \rightarrow 00
$$

$$
1 \rightarrow 01
$$

$$
2 \rightarrow 10
$$

$$
3 \rightarrow 11
$$

 α consider the **hinary fraction** o consider the binary fraction by expression above, σ as usual. We also consider the binary fraction

$$
0.j_1j_2...j_n = j_1/2 + j_2/4 + ... + j_n/2^n = j/2^n
$$

The key insight into designing a circuit for the Fourier transform is to notice that it can be written in a **product form** state *[|]j*! using the binary representation *^j* ⁼ *^j*1*j*² *...jn*. More formally, *^j* ⁼ *^j*12*ⁿ*−¹ ⁺ *f* he key insight into designing a circuit for the Fourier transform is *product representation*: \mathbf{g} nt mto utsigning a theuit ior the roundr transion in me acsigning a circuit for the rouncr transformer and the manufacturer. product_iorm ____________

$$
|j_1,\ldots,j_n\rangle \to \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1j_2\cdots j_n}|1\rangle\right)}{2^{n/2}}
$$

definition of the quantum Fourier transform. As we explain shortly this representation The proof is the following transform, and the proof is the following

$$
|j\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^{n}-1} e^{2\pi i j k/2^{n}} |k\rangle
$$

\n
$$
= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} e^{2\pi i j} (\sum_{l=1}^{n} k_l 2^{-l}) |k_1 \dots k_n\rangle
$$

\n
$$
= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle
$$

\n
$$
= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right]
$$

\n
$$
= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right]
$$

\n
$$
= \frac{\left(|\Theta\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0 \cdot j_{n-1} j_n} |1\rangle \right) \dots \left(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle \right)}{2^{n/2}}
$$

2*n/*² *l*=1 on the time to the fourth line, we used the factury quantum Fourier transform. Such a circuit is shown in Figure 5.1. The gate *R^k* denotes In going from the third to the fourth line, we used the identity

(

$$
\sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} f_{k_l} = \sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} f_{k_1} f_{k_2} \dots f_{k_n}
$$

$$
= \bigotimes_{l=1}^{n} \sum_{k_l=0}^{1} f_{k_l} = \sum_{k_1=0}^{1} f_{k_1} \sum_{k_2=0}^{1} f_{k_2} \dots \sum_{k_n=0}^{1} f_{k_n}
$$

1 0

)

The product representation (5.4) makes it easy to derive an effective and the first circuit for the first for

²*n/*² *.*(5.10)

The unitary equation for the quantum Fourier transform. As an incidental bonus we obtain the classical $\frac{1}{\sqrt{1-\frac{1}{2}}}$

⁼ ¹

⁼ ¹

⁼ ¹

⁼ ¹

fast Fourier transform, in the exercises!

[|]˜j! = 2−n/² !

$$
|0,1\rangle\rightarrow(|0\rangle\pm\exp(i\theta)|1\rangle)/\sqrt{2}
$$

followed by a Z-rotation by a Z-rotation by expression above, and the expression above, and the expression abov

 $t = t$ rotation depends on the values of the values of the other bits. So the other bits. is a Hadamard followed by a Z-rotation by θ. In the expression above the rotation depends on the values of the other bits. So we should expect to be able to build the Fourier transform out of Hadamard and controlled-phase rotation gates. Define the rotation able to build the Fourier transform out of Hadamard and on depends on the values of the other bits. So we should exp quantum Fourier transform. Such a circuit is shown in Figure 5.1. The gate *R^k* denotes **be able to build the Fourier**
pro∏ed-phase rotation gates De *a e**l***_{1 ***l***_{1**} *ln l***_{1**} *l ln l***_{1**} *l l<i>l l l l***** *l*** ***l l l l l l l l l l* *****l l l l l l l*} *k*1=0

$$
R_k \equiv \left[\begin{array}{cc} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{array} \right]
$$

To see that the picture distribution $\mathcal{L}_\mathcal{A}$ and $\mathcal{L}_\mathcal{A}$

 $\frac{1}{\sqrt{2}}$ $\frac{1}{\sqrt{2}}$ The circuit implementing the QFT then is **The contact of the parameter of the quantum and the quantum and the quantum** $\frac{1}{\sqrt{2\pi}}$ $\frac{1}{2}$ $\frac{1}{2}$

 p_n

 $|j_n\rangle$ **produces the state e bitter of the first bitter of the first bitter** $|j_n\rangle$ **e c h** $|0\rangle + e^{2\pi i 0.j_n} |1\rangle$

 $\int_{-\infty}^{\infty}$ $\int_{-\infty}^{\in$ *[|]*0" ⁺ *^e*²π*i*0*.j*1*j*² *[|]*1" " 21*/*² $\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)|j_2\ldots j_n\rangle$ $\mathcal{L}^{1/2}$ for the smallest symmetry. Note shown are swap gates at the end of the end of the circuit which whi reverse the order of the qubits, or normalization factors of 1*/* [√]2 in the output.

21*/*² since $e^{2\pi i 0.j_1} = -1$ when $j_1 = 1$, and is +1 otherwise. Applying the controlled-R₂ gate produces the state

$$
\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1j_2}|1\rangle\right)|j_2\ldots j_n\rangle
$$

6

(5.9)

We continue applying the controlled-R₃, R_4 through R_n gates, each of which adds an extra bit to the phase of the co-efficient of the first $|1\rangle$. At the end of this procedure we have the state $\begin{array}{ccc} \cdots & H & R_2 & |0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \end{array}$ produces the state of the state y_1
 $\begin{array}{ccc}\n|j_1\rangle & |j_2\rangle \\
|j_2\rangle & |j_3\rangle\n\end{array}$ $|j_n\rangle$ 1 21*/*² $\left\langle \ket{0} + e^{2\pi i 0.j_1j_2...j_n}\ket{1} \right\rangle \ket{j_2 \dots j_n}$ We continue a the end of this procedure we have the state $\begin{array}{ccc} \cdots & H & R_2 & |0\rangle + e^{2\pi i 0. j_{n-1} j_n} |1\rangle \end{array}$ \cdots *H* R_2 $|0\rangle + e^{2\pi i 0 \cdot j_{n-1} j_n} |1\rangle$
 \cdots \bullet *H* $|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle$ We continue applying the controlled-*R*3, *R*⁴ through *Rⁿ* gates, each of which adds an $\frac{1}{2^{1/2}}\left(\ket{0}+e^{2\pi i 0.j_1j_2...j_n}\ket{1} \right) \ket{j_2...j_n}$ $\begin{array}{ccccccc}\n\langle j_{n-1}\rangle & & & & \bullet & & \cdots & H & R_2 & & |0\rangle + e^{2n\omega_{j}j_{n-1}}. \\
\langle j_{n-1}\rangle & & & & \Omega & & P & & \cdots & H & R_2 & & |0\rangle + e^{2n\omega_{j}j_{n-1}}. \end{array}$ r_{ref} for the ϵ for the ϵ transform. Not shown are supported to the end of the end of the circuit which $\overline{u}^{[j_n]}_{\lambda}$ $\overline{u}^{[j_2]}_{\lambda}$ in the controlled R^{R_n} , $\overline{R}^{R_{n-1}}_{\lambda}$ in the output of R^{R_n} $\frac{1}{2^{1/2}}\left(0\right) + e^{2\pi i 0.j_{1}j_{2}...j_{n}}|1\rangle\right)|j_{2}...j_{n}\rangle$ reverse the order of the qubits, or normalization factors of 1*/* $\frac{1}{2}$ in the output of $\frac{1}{2}$

 \mathcal{L} for the quantum Fourier transform. Note shown are swap gates at the end of the end of the circuit which which

[√]2 in the output.

Next, we perform a similar procedure on the second qubit. The **Hadamard gate puts us in the state on the state on the Hadamard gate puts us in the state** " !*|*0" ⁺ *^e*²π*i*0*.j*² *[|]*1" Next, we perform a similar procedure on the second qubit. The *n*₂, and *p* = 1, and is the state *example the controlled-party the controlled-*We continue applying the controlled-*R*3, *R*⁴ through *Rⁿ* gates, each of which adds an produces the state of the state
In the state of the

$$
\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i 0.j_1j_2...j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right)|j_3...j_n\rangle
$$

 $W = \frac{1}{\sqrt{2\pi}}$ for each $\frac{1}{\sqrt{2\pi}}$ for each $\frac{1}{\sqrt{2\pi}}$ for each $\frac{1}{\sqrt{2\pi}}$ final states a final state and the controlled-R₂ through R_{n-1} gates yield the state
 and the controlled- R_2 through R_2 , gates vield the state *|j*³ *...jn*"*,* (5.15) aind the e \mathbf{h} the state state \mathbf{h} 10 ugii n_{n-1} gales yieiu
— *[|]*0" ⁺ *^e*²π*i*0*.j*1*j*2*...jⁿ [|]*1"

and the controlled-*R*² through *Rⁿ*−¹ gates yield the state

21*/*²

*[|]*0" ⁺ *^e*²π*i*0*.j*1*j*2*...jⁿ [|]*1"

operations, the state of the qubits is

$$
\frac{1}{2^{2/2}}\left(\left|0\right\rangle+e^{2\pi i0.j_1j_2...j_n}\left|1\right\rangle\right)\left(\left|0\right\rangle+e^{2\pi i0.j_2...j_n}\left|1\right\rangle\right)|j_3...j_n\rangle
$$

/e_<u>con</u>tinue in this fashion for each qubit, giving a final state
— We continue in this fashion for each qubit, giving a final state $\frac{1}{\sqrt{1-\frac{1$ 2*n/*² ≀ <u>con</u>tinue in this fashion for each qubit, giving a

We continue in this fashion for each qubit, giving a final state α final state α

$$
\frac{1}{2^{n/2}}\left(|0\rangle+e^{2\pi i0.j_1j_2...j_n}|1\rangle\right)\left(|0\rangle+e^{2\pi i0.j_2...j_n}|1\rangle\right)...\left(|0\rangle+e^{2\pi i0.j_n}|1\rangle\right)
$$

If necessary, swap operations omitted from the Figure for clarity, can be used to reverse the order of the qubits. After the swap operations, the state of the qubits is set of the qubits is gate and *n* − 2 conditional rotations on the second qubit, for a total of *n* + (*n* − 1) gates. for the quantum Fourier transform on three qubits is given in Box 5.1. Fourier transform. This construction also proves that the quantum Fourier transform is α then used to reverse the order of the qubits. After the swap the swap of the qubits. After the swap swap α operations, the state of the state of the state of the quantity

$$
\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i 0.j_1j_2\cdots j_n}|1\rangle\right)
$$

Box 5.1: Three qubit quantum Fourier transform

For concreteness it may help to look at the explicit circuit for the three qubit quantum Fourier transform:

Recall that *S* and *T* are the phase and $\pi/8$ gates (see page xxiii). As a matrix the quantum Fourier transform in this instance may be written out explicitly, using $\omega = e^{2\pi i/8} = \sqrt{i}$, as

$$
\frac{1}{\sqrt{8}}\begin{bmatrix}\n1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\
1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\
1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\
1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\
1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\
1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\
1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1\n\end{bmatrix}.
$$
\n(5.19)

gate and n − 1 conditional rotations on the first qubit – a total of n gates. This is followed by a Hadamard gate and $T_n - 2$ conditional rotations on the econd qubit, for a total of n + (n − 1) gates. Continuing in this way, we see that n+(n−1)+···+1 = n(n+1)/2 gates are required plus the gates involved in the swaps. At most n/2 swaps are required, and each swap can be accomplished using three CNOT gates. Therefore, this circuit provides a more operations to compute the Fourier operations to compute the Fourier transform t_{t} is a quantum for performing the quantum Fourier transform σ How many gates does this circuit use? We start by doing a Hadamard $\Theta(n^2)$ algorithm for performing the quantum Fourier transform.

In contrast, the best classical algorithms for computing the discrete Fourier transform on 2ⁿ elements are algorithms such as the Fast Fourier surfer the first step in phonements are digitized sounds the Fourier transform that ϵ Transform (FFT), which compute the discrete Fourier transform using
 Ω' 3N $U(12)$ gates. Θ(n2n) gates.

But remember, we cannot measure particular values of the amplitudes! The quantum Fourier Transform is useful only as a piece of another algorithm.

Exercise on IBM Quantum Composer.

 $N = 2$

Opuput state $[0.5+0j, 0.5+0j, 0.5+0j, 0.5+0j]$

Opuput state $[0.5+0j,-0.5+0j,0.5+0j,-0.5+0j]$

Opuput state $[0.5+0j, 0+0.5j, -0.5+0j, 0-0.5j]$

Opuput state $[0.5+0j, 0-0.5j, -0.5+0j, 0+0.5j]$

This corresponds to the matrix seen before

 $N = 3$

Phase estimation algorithm

The problem: Suppose we are given a unitary operator U on n qubits, which has a known eigenstate |u⟩ with an unknown eigenvalue $e^{2\pi i\phi}$, with ϕ in [0,1]. We wish to find the phase ϕ with some precision.

By itself, the phase estimation algorithm is a solution to a rather artificial problem. But this solution turns out to be useful as a piece of several other algorithms, to solve much more natural and important problems. $\mathsf{problems.}$

The first thing we might try is to prepare n q-bits in the state $|u\rangle$, and carry out the unitary transformation U on them: $\frac{1}{2}$

Is there a measurement on the bits which will give us information
 Expectation Value ¹⁹⁷ about the phase φ? The answer, or course, is no. O just produces an
overall phase on the state, with no observable consequences. overall phase on the state, with ho observable consequences. about the phase φ? The answer, of course, is no: Û just produces an

We need to generate relative phases, which can be measured. This can be done in the following way. Here is a more sophisticated approach to the problem. we need to generate relative phases, which can be measured. Th
he done in the following way We need to generate relative phases, which can be meas

10

The initial state changes as follows:

$$
|0\rangle|u\rangle \rightarrow \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]|u\rangle
$$

$$
\rightarrow \frac{1}{\sqrt{2}}[|0\rangle|u\rangle + |1\rangle e^{2\pi i\varphi}|u\rangle] = \frac{1}{\sqrt{2}}[|0\rangle + e^{2\pi i\varphi}|1\rangle]|u\rangle
$$

Now we have a relative phase among the two qubits of the computational basis, which can be measured for example by first applying an Hadamard gate to the first qubit, whose state then becomes:

$$
\frac{1+e^{2\pi i\varphi}}{2}|0\rangle+\frac{1-e^{2\pi i\varphi}}{2}|1\rangle
$$

and by making a measurement on the computational basis we have the output probabilities:

$$
\mathbb{P}[0] = \frac{1 + \cos 2\pi \varphi}{2}, \qquad \mathbb{P}[1] = \frac{1 - \cos 2\pi \varphi}{2}
$$

We can run this circuit several times to recover the phase ϕ . Unfortunately, the convergence of this algorithm is very slow. After N repetitions, the accuracy in the estimate of the phase ϕ is N^{-1/2}. If we wish to know ϕ with m bits accuracy, then N^{1/2} = 2^m, which means $N = 2^{2m}$: the number of repetitions grows exponentially with the number of bits of accuracy.

A smarter solution is provided by the following algorithm.

The first stage of the algorithm is: *IU* in the stays in the state *in the stage* of the algorithm is:

⁼ ¹

accuracy we wish to have in our estimate for φ , and with what showing, as you show that the phase estimation procedure to be successful operations like that in Figure 5.2 is to take the state *|j*!*|u*! to *|j*!*U^j|u*!. (Note How we choose t depends on two things: the number of digits of probability we wish the phase estimation procedure to be successful.

The final state of the first register is easily seen to be

$$
\frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{t-2}\varphi}|1\rangle\right)\dots\left(|0\rangle + e^{2\pi i 2^{0}\varphi}|1\rangle\right)
$$
\n
$$
= \frac{1}{2^{t/2}}\sum_{k_1=0}^1 \dots \sum_{k_t=0}^1 e^{2\pi i \varphi(k_1 2^{t-1} + k_2 2^{t-2} + \dots + k_t 2^0)}|k_1 k_2 \dots k_t\rangle = \frac{1}{2^{t/2}}\sum_{k=0}^{2^{t}-1} e^{2\pi i \varphi k}|k\rangle.
$$

Suppose φ may be expressed exactly in t bits, as φ = 0. φ_1 φ_t . Then with its contract us suppose that the there is an exact term is an exact to bit the there is an exact t-bit uppose ϕ may be expressed exactly in t bits, as $\phi = 0.\phi_1 \dots \phi_n$

$$
e^{2\pi i \phi} = e^{2\pi i 0 \cdot \phi_1 \dots \phi_t}.
$$

\n
$$
e^{4\pi i \phi} = e^{2\pi i \phi_1 \cdot \phi_2 \dots \phi_t} = e^{2\pi i \phi_1 + 2\pi i 0 \cdot \phi_2 \dots \phi_t} = e^{2\pi i 0 \cdot \phi_2 \dots \phi_t}
$$

\n
$$
e^{2^j \pi i \phi} = e^{2\pi i 0 \cdot \phi_j \dots \phi_t}.
$$

⁽normalization factors 2^{-1/2} have been omintted)

The output state can be rewritten as Theorem and the output state can be rewritten as

$$
\frac{1}{2^{t/2}}\left(\overline{|0\rangle}+e^{2\pi i0.\varphi_t}|1\rangle\right)\left(|0\rangle+e^{2\pi i0.\varphi_{t-1}\varphi_t}|1\rangle\right)\dots\left(|0\rangle+e^{2\pi i0.\varphi_1\varphi_2\cdots\varphi_t}|1\rangle\right)
$$

which is the Quantum Fourier Transform of the state $|\varphi_1 \dots \varphi_t \rangle$ $|\varphi_1\ldots\varphi_t\rangle$ eigenvalue of a unitary operator *U*, given the corresponding eigenvector *|u*!. An essential

Therefore the second stage of the algorithm is to apply the inverse Quantum Fourier Transform to the output of the first state, and one recovers exactly the bits of the binary fraction for φ.

The full phase estimation algorithm is: $\frac{1}{2}$ full phase estimation algorithm is: *j*=0

wires, as usual) are the first register, and the bottom qubits are the second register, numbering as many as required The above analysis applies to the ideal case, where φ can be written
The above analysis applies to the ideal case, where φ can be written The above analysis applies to the ideal case, where ϕ can be written
exactly with a t bit binary expansion. What happens when this is not the case? exactly with a t bit binary expansion. What happens when this
the case? ϕ and *b/*2*^t* satisfies 0 ≤ δ ≤ 2−*^t* . We are the observation at the observation at the observation at the end of \mathcal{A}

Applying the inverse QFT to the state in the previous page produces eigenvalue of a unitary operator \mathcal{U} , given the corresponding eigenvalue of a unitary operator \mathcal{U} , \mathcal{U} the state $t_{\rm t}$ and inverse α , with stately in the probability probability. The state

$$
\frac{1}{2^t}\sum_{k,l=0}^{2^t-1}e^{\frac{-2\pi i k l}{2^t}}e^{2\pi i \varphi k}|l\rangle = \frac{1}{2^t}\sum_{k,l=0}^{2^t-1}e^{-\frac{2\pi i}{2^t}(l-2^t\varphi)k}|l\rangle
$$

we see again that if $\phi = 0.\phi_1 \dots \phi_t$, then $Z^c \phi$ is an integer and the vertical to χ ^t The <u>III</u> all state is $|\varphi_1 \cdots \varphi_t \rangle = |z| \varphi$ We see again that if $\phi = 0.\phi_1 \ldots \Phi_t$, then 2^t ϕ is an integer and the .
Fil sum over k returns a Kronecker delta, forcing it to be equal to 2^t φ. al st *^t*−¹ The <u>fin</u>al state is $|\phi_1 \cdots \phi_t \rangle = |2^t \phi \rangle$ $\langle \varphi \rangle$ ϕ - θ ϕ = ϕ = ϕ + ϕ and the ϕ

p(*t* the case, the coeffici -1 *l*2 ted to If this is not the case, the coefficient associated to the state |l> is:

$$
= \frac{1}{2^t}\sum_{k=0}^{2^t-1}e^{-\frac{2\pi i}{2^t}(l-2^t\varphi)k} = \frac{1}{2^t}\frac{1-e^{2\pi i(l-2^t\varphi)}}{1-e^{2\pi i(l-2^t\varphi)/2^t}}
$$

And its square modulus is

$$
\frac{1}{2^{2t}}\frac{1-\cos[2\pi(l-2^t\varphi)]}{1-\cos[2\pi(l-2^t\varphi)/2^t]}
$$

dl 1

*^l*² (5.33)

The above function is sharply peaked around the closest l to $2^t \phi$. More precisely, it can be shown (see Nielsen & Chuang) that to successfully obtain φ accurate to *n* bits with probability of success at least $1 - ε$, we choose Suppose we wish to approximate ϕ to an accuracy 2−*ⁿ*, that is, we choose *e* = 2*^t*−*ⁿ* − 1. The above function is sharply peaked around the closest l to 2^t φ.

$$
t = n + \left\lceil \log \left(2 + \frac{1}{2\epsilon} \right) \right\rceil
$$

The number of qubits needed to run the algorithm with the desired accuracy grows linearly. Assuming that the controlled-U^{2j} unitaries are given by oracles (and hence free), the complexity of the algorithm is basically that of the Quantum Fourier Transform, O(t²). We have an exponential advantage with respect to the naïve algorithm we first tried.

However, if we have to perform circuits for the controlled- U^{2j} unitaries, than things change. Even if we have an efficient circuit for controlled-U, we need efficient circuits for all the controlled- U^{2j} gates as well; just repeating the controlled-U 2^{j} times will make the complexity exponential.

There is another somewhat artificial assumption as well. It is assumed that we don't know the eigenvalue $e^{2\pi i\phi}$, but that we can prepare the eigenvector $|u\rangle$. While this may sometimes be true, in most cases it will not be.

The phase estimation algorithm then is:
 μ is second register contains and register contains . Quantum Phase estimation first prepares the state prepares the state prepare the state prepare the state prepare the state prepare of the state prepare the state pre *m m θ*

Inputs: (1) A black box wich performs a controlled- U^j operation, for integer j , (2) an eigenstate $|u\rangle$ of U with eigenvalue $e^{2\pi i\varphi_u}$, and (3) $t = n + \lceil \log \left(2 + \frac{1}{2\epsilon}\right) \rceil$ qubits initialized to $|0\rangle$. **indicipalizion in the first of and encoding in the seconding in the second register.**

Outputs: An *n*-bit approximation $\widetilde{\varphi_u}$ to φ_u .

Runtime: $O(t^2)$ operations and one call to controlled- U^j black box. Succeeds with probability at least $1 - \epsilon$. **u u** *with probability at 1*

\blacksquare Procedure: 2¹ |1⟩)1 (|0⟩ + *e*²*πi^θ*

*e*²*πiθ^k* |*k*⟩] |*ψ*⟩

e |*k*⟩

estimation will be seen as \sim

Example 1. Consider the unitary operator (X gate)

 $U = \left[\begin{matrix} 0 & 1\ 1 & 0 \end{matrix} \right]$ phase (). A calculation for the eigenvalues of U gives and . So and . The] *U λ* = *e*²*πi^θ* The state of the second register doesn't change during computation, so the final state of the system before $U = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

with eigenstate $|+\rangle = \frac{1}{2}||0\rangle + |1\rangle$ and we know the relative eigenvalue is a phase $\lambda = e^{2\pi i \theta}$ V^2 The goal is to find the phase, with 1 bit precision. with eigenstate $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + |1\rangle|$ and we know the relative eigenvalue with eigenstate $|+\rangle = \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$ and we know the relative eigenvalue is a phase $|+\rangle = \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$ δ is a phase $\lambda = e^{2\pi i \theta}$, $\lambda = e^{2\pi i \theta}$ \overline{a} calculation for the eigenvalues of U g *θ λ*¹ = 1 *λ*² = −1 *θ*¹ = 0 *θ*² = ¹

1. The initial state is: phases can be calculated on a quantum computer where corresponds to the measured result and to because it's the only other option (using more qubits to estimate is not neccesary in this case). *θ*₂ 1 The initial state is:

https://vtomole.com/blog/2018/05/20/pea Page 1 of 4 2018/05/20/pea

 $|0\rangle |+\rangle$ $|0\rangle|+\rangle$

2. We apply a Hadamard to the first qubit to create the superposition:

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

3. We apply the controlled-U gate once: the state remains unchanged

4. The inverse Fourier transform, which amounts to an Hadamard, brings the state back to >

 $|0\rangle|+\rangle$

5. Measuring the first register gives 0, from which we learn that the phase, in binary fraction, is 0.0 (to 1 bit accuracy). A single qubit is a vector *|*ψ! = *a|*0! + *b|*1! parameterized by two complex numbers

This is correct: we know that the eigenvalue is 1, therefore the phase is Ω . σ . $\overline{}$

Exercise 1. Consider the unitary operator (T gate) *<u>Fuenciae 1</u>* Concid</u> $\sum_{i=1}^n$ and $\sum_{i=1}^n$ consider the differences $\sum_{i=1}^n$

2 and *S* = *T*²

 $T = \left[\begin{array}{cc} 1 & 0 \\ 0 & \exp(i\pi/4) \end{array} \right]$

You might wonder why the *T* gate is called the π*/*8 gate when it is π*/*4 that appears in with eigenstate $|1$ >. Write down the circuit to find the associated *eigenvalue (which is assumed to be a phase) with 3 bit precision.*

Solution: since the associated eigenvalue is e^{iπ/4}, the desired phase is 2.4 (8) which corresponds to the binary fraction 0.001. Therefore the algorithm returns 001, corresponding to the exact value of the phase. 1/8, which corresponds to the binary fraction 0.001. Therefore the

Recall also that a single qubit in the state *a|*0!+*b|*1! can be visualized as a point (θ*,* ϕ) On the IBM Quantum Composer, it looks as follows because the overall phase of the state is unobservable. This is called the Bloch sphere

Remember that qubits are ordered from bottom to top. (use "freeform alignment" to place the gate as desired)

The output probabilities are

Exercise 2. Consider the same case as before, with a phase $\phi = 1/3 = 1/3$ 0,33333 (not binary fraction). With 3-bit precision the algorithm is similar to the one before:

The output probabilities are

Most likely outcome: $011 \rightarrow b$ inary fraction 0.011 corresponding to a phase $\phi = 1/4 + 1/8 = 0.375 \rightarrow 0.042$ difference from exact result (off by 13%).

The second most likely outcome is: $010 \rightarrow b$ inary fraction 0.010 corresponding to a phase $\phi = 1/4 = 0.25 \rightarrow 0.083$ difference from exact result (off by 25%).

The true phase lies in between, closer to the most likely outcome.

The circuit can be written more shortly as follows

The output probabilities are

Most likely outcome: $0101 \rightarrow b$ inary fraction 0.0101 corresponding to a phase $\phi = 1/4 + 1/16 = 0.313 \rightarrow 0.02$ difference from exact result (off by 6%).

The second most likely outcome is: $0110 \rightarrow b$ inary fraction 0.0110 corresponding to a phase $\phi = 1/4 + 1/8 = 0.375 \rightarrow 0.042$ difference from exact result (off by 13%).

We see an improvement with respect to the case with 3 bits

Exercise 4. Same as before, but with 5-bit precision. The outcome probabilities are

Most likely outcome: $01011 \rightarrow$ binary fraction 0.01011 corresponding to a phase $\phi = 1/4 + 1/16 + 1/32 = 0.344 \rightarrow 0.011$ difference from exact result (off by 3%).

The second most likely outcome is: $01010 \rightarrow b$ inary fraction 0.01010 corresponding to a phase $\phi = 1/4 + 1/16 = 0.313 \rightarrow 0.02$ difference from exact result (off by 6%).

Again, we see an improvement with respect to the previous cases.

Order finding algorithm For positive integers *x* and *N*, *x<N*, with no common factors, the *order* of *x* modulo *N* is defined to be the least positive integrals positive integrals α *r*, α *r* α

Definition of order: For positive integers x and N, $x < N$, with no common factors, the order of x modulo N is defined to be the least
positive integer r, such that y^r = 1(mod N) positive integer, r, such that $x^r = 1 \pmod{N}$. to be a hard problem on a classical computer, in the sense that no algorithm is known **Definition of order:** For positive integers **x** and N , **x** < N , with no

Order finding problem: to determine the order for some specified x and N. for order-finding. Exercise 5.10: Show that the order of *x* = 5 modulo *N* = 21 is 6.

Order-finding is believed to be a hard problem on a classical computer.
The guarature also *x* that for a relating is in the phase activation. The quantum algorithm for order-finding is just the phase estimation algorithm applied to the unitary operator **comes algorithm** applied to the unitary operator

 $U|y\rangle \equiv |xy \pmod{N}$

with $y \in \{0, 1, \ldots 2^{\mathsf{L}}\text{-}1\}$, with L to be defined later. Note that here and below, when $N \le y \le 2^L - 1$, we use the convention that xy(mod N) is just y again. That is, U only acts non-trivially when $0 \le y \le N - 1$.

Example: $N = 15$, $x = 7$. Then:

A simple calculation shows that the states defined by − A simple calculation shows that the states defined by
← Applications: order-finding and factoring and factoring 2277 and f

$$
|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i sk}{r}\right] |x^k \bmod N\rangle
$$

for integer $0 \leq s \leq r - 1$ are eigenstates of U, since $f(x)$ integer $0 \leq x \leq r-1$ are eigenstates of \cup sin *r*−1 for integer 0 ≤ *s* ≤ *r* − 1 are eigenstates of *U*, since

$$
U|u_s\rangle = \frac{\overline{1}}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i sk}{r}\right] |x^{k+1} \text{ mod } N\rangle
$$

$$
= \exp\left[\frac{2\pi i s}{r}\right] |u_s\rangle.
$$

(this because x^r mod $N = 1$, by definition). responding eigenvalues exp(2π*is/r*), from which we can obtain the order *r* with a little (this because x^r mod $N = 1$, by definition).

Using the phase estimation procedure allows us to obtain, with high accuracy, the corresponding eigenvalues exp^{2πis/r}, from which we can obtain the order r with a little bit more work. Using the phase estimation procedure allows us to obtain, with ingitially S_{S} is unitary that α is unitary the corresponding eigenvalues exp^{2nis/r} from which we can ccuracy, the correspor
htain the order r with

There are three important requirements to be met in order for the algorithm to be efficient:

- We must have efficient procedures to implement a controlled-Figure implement the entire sequence of controlled and the entire sequence of controlled and a vertice of controlled and U^{2j} operation for any integer j. omplement a controlled estimation procedure using *O*² Operation for any integer j.
- We must be able to efficiently prepare an eigenstate $|u_s\rangle$ with a non-trivial eigenvalue. The second requirement is a little tricker: prepare an eigenstate $|u_s|$ with a report we know *require* that we know r
- We must be able to obtain the desired answer, r, from the result of the phase estimation algorithm, $φ ≈ s/r$. !*r*−1

We analyse the three eleme<u>nts</u> separately. nt<u>s</u> sep ─=
e<u>nts</u> sep *u* $\frac{1}{2}$ = $\frac{1}{2}$ = *|us*# = *|*1#*.* (5.44)

 $\frac{1}{2}$

 $\frac{1}{2}$

Implementation of the controlled- $\overline{U^{2j}}$ operation: modular exponentiation. The following relation holds: *k*=0 −2π*isk ______*
|
|
| approximation: modular **for the second of the set of** *u***, since** $\frac{1}{2}$

0 ≤ *y ≤ 2 × 1.*
Define calculation shows that the states defined by the states defined by states defined by the states defined

$$
|z\rangle|y\rangle \rightarrow |\underline{z}\rangle U^{z_t 2^{t-1}} \cdot \underline{U^{z_1 2^0}}|y\rangle
$$

= $|z\rangle|x^{z_t 2^{t-1}} \times \cdots \times x^{z_1 2^0} y \pmod{N}$
= $|z\rangle|x^z y \pmod{N}$.

Thus the sequence of controlled- U^{2j} operations used in phase estimation is equivalent to multiplying the contents of the second register by the modular exponential x^2 (mod N), where z is the contents of the first register. to reversibly multiply the contents of the second register by *x^z*(mod *N*), using the U sing the phase estimation procedure allows us to obtain, with high accuracy, the cor-

This operation may be accomplished classically using $O(L^3)$ gates. The classical circuit can be transformed into a reversible circuit, which can be translated into a quantum circuit of similar complexity, computing the transformation $|z\rangle|y\rangle \rightarrow |z\rangle|x^2y \pmod{N}$. The book of Nakahara & *x*₁₂*j*₁₂*j*₁₂*j*₁₂*j*₁₂*j*₁₄*y*₁*l*₁₁, *y*₁(11100 *N₎₁. The book of Nukundru & <i>Ohimi (p. 156)* explains in detail how to do it. Somm p. 150 capital of actual now to do *it. Ohimi (p. 156) explains in detail how to do it.* the dimension and $\vert z \vert \vert y \vert > \vert z \vert \vert x \vert y \vert$ in our type the soon of rundrian α . is satisfied by using a procedure with which we are *n*.

Prepare an eigenstate $|u_s\rangle$. Preparing $|u_s\rangle$ requires that we know r, so this is out of the question. Fortunately, there is a clever observation $t_{\rm{min}}$ is determined statistical standard statistic statistic statistic upon the observation which allows us to circumvent the problem of preparing $|u_s\rangle$, which is that that can implement the entire sequence of controlled-
operations applications applications applications applications applications and provide the phase of control of $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ which allows us to chequity the problem of preparing $\begin{bmatrix} a_{s} \\ b_{s} \end{bmatrix}$, which is $\frac{1}{\sqrt{2}}$ and $\frac{1}{\sqrt{2}}$ and

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

 $\frac{1}{\pi}$ gates means that if we propera the second register in the state $\frac{13}{\pi}$ p_{max} more efficient and proposes into second register in the state $|12\rangle$ pust before ineasurement the state of the two registers will be.
 $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ qubits in the first register (referring to Figure 5.3), and prepare the second register in This means that if we prepare the second register in the state |1>, just before measurement the state of the two registers will be:

in Figure 5.4.

Exercise 5.13: Prove (5.44). (*Hint:* (*^r*−¹

√*r*

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

if we initialize the second register as *|*1!. Show that the same state is obtained if

 $\frac{1}{2}$

&'

operation

A measurement of the first register will collapse the state of the second register to the eigenstate $|u_s\rangle$, and the first register will end up in the state $|\phi_s\rangle$, from which the phase $\phi \approx s/r$ can be read.

> $\frac{1}{\pi}$ Therefore, if we use :
...

√*r*

$$
t = 2L + 1 + \left\lceil \log \left(2 + \frac{1}{2\epsilon} \right) \right\rceil
$$

qubits in the first register and prepare the second register in the state (1), which is trivial to construct, it follows that for each s in the range 0
→ through r – 1, we will obtain an estimate of the phase φ ≈ s/r accurate to algorithm is strated algorithm is schematically departed algorithm is strated algorithm in the order of the order of the order of the order of th 2L + 1 bits, with probability at least $(1 - \varepsilon)/r$.

The order finding algorithm the is: Exercise 5.13: Prove (5.44). (*Hint:* (*^r*−¹ *^s*=0 exp(−2π*isk/r*) = *r*δ*k*0.) In fact, prove that

How to extract r from $\phi \approx s/r$. We only know ϕ to 2L + 1 bits, but we also *The continued fraction expansion* if we could compute the nearest such fraction to φ we might obtain r. The reduction of order-finding to phase estimation is completed by describing how to know a priori that it is a rational number – the ratio of two integers – and

ore is an algorithm, which accomplishes this task officiently known as There is an algorithm which accomplishes this task efficiently, known as the continued fractions algorithm: given φ the continued fractions algorithm efficiently produces numbers s' and r' with no common factor, such that s'/r' = s/r. The number r' is our candidate for the order. We can the *continued fractions algorithm*. An example of how this works is described in Box 5.3. check to see whether it is the order by calculating $x^{r'}$ mod N, and seeing if the result is 1. If so, then r' is the order of x modulo N, and we are done.

^r [−] ^ϕ

²*r*² *.* (5.48)

Performance. How can the order-finding algorithm fail? There are two possibilities. The first and superposition of superposition of superposition of such eigenstates and superposition of su is satisfied by using a procedure known as *modular exponentiation*, with which we accomplished easily using the techniques of techniques of reversibilities. The basic ideas of reversion σ is to reversibly compute the function *x^z*(mod *N*) of *z* in a third register, and then

First, the phase estimation procedure might produce a bad estimate to s/r. This occurs with probability at most ε, and can be made small with a spit rims ocears with probability at most *c*, and earnod made small with a
negligible increase in the size of the circuit. I had, the pridate eatherwith procedure imight produce a bad eatherwith to the grigible filered of the size of the circuit. Spiriths occurs with probability at most c, and can be made small with a meghgibit multiplicant multiplication to (mod *N*), by squaring *x* modulo *N*, then

More seriously, it might be that s and r have a common factor, in which *rivice* seriously, it might be that *s* and *r* have a common ractor, in which factor of r, and not r itself. Fortunately, there are at least three ways √*r s*=0 around this problem. Perhaps the most straightforward way is to note that for randomly chosen s in the range 0 through r − 1, it's actually pretty likely that s and r are co-prime, in which case the continued fractions algorithm must return r. Specifically, one can show that by repeating the algorithm 2log(N) times we will, with high probability, observe a phase s/r
with that south were as written and therefore the continued frestions. such that s and r are co-prime, and therefore the continued fractions algorithm produces r, as desired. *ase the number F returne<u>d</u> by the continued iractions algorithm be a*
factor of r and not ritself Fortunately there are at least three ways around this problem. Ferriaps the most straightforward way is to note
that for randomly chosen s in the range 0 through r = 1 it's actually pretty $\frac{1}{2}$, and first register (referring to Figure 5.3), and prepare the second register in the second stage. The second stage of the algorithm is based upon the second stage of the algorithm is based upon the observation inery that stand it are co-prime, in which case the continued iractions
algorithm must return r Specifically, one can show that by repeating the through through research *return in an estimate of the can show that by repeating the* and property and an estimate to 2
Also sither $2\log(N)$ times we will with high probability, absence a phase of a $\frac{1}{2}$ actor of i, and not integral to tunately, there are at least three ways algorithm we all learn as children for multiplication), for a total cost of *O*(*L*3) for *x*^{*x*} algorithm 2log(N) times we will, with high probability, observe a phase Performing *^t* [−] 1 modular multiplications with a cost *^O*(*L*2) each, we see that this

See Nielsen and Chuang for further details. 1 !*r*−1 product can be computed using *O*(*L*3) gates. This is sufficiently efficient for our p urposes, but more efficient algorithms are possible based on more effects are possible based on more efficient algorithms are possible based on more effects as p

← –
Note. The quantum state produced in the order-finding algorithm, before the inverse Fourier transform, is stated in the order-finding algorithm and the order**e**
 Note The quantum state produced in the order-finding algorithm hefore of Section 3.2.5, it is not start for the straightforward to construct a reversible construction and the construction of t *the medical caller and and* α) outputs α

$$
|\psi\rangle = \sum_{j=0}^{2^t-1} |j\rangle U^j |1\rangle = \sum_{j=0}^{2^t-1} |j\rangle |x^j \text{ mod } N\rangle
$$

if we initialize the second register as |1⟩. The same state is obtained if we replace U^j with a different unitary transform V, which computes we replace *U^j* with a *different* unitary transform *V* , which computes

$$
V|j\rangle|k\rangle \equiv |j\rangle|k + x^j \text{ mod } N\rangle
$$

and start the second register in the state $|0\rangle$. Moreover, V can be constructed also using $O(L^3)$ gates.

Inputs: (1) A black box $U_{x,N}$ which performs the transformation $|j\rangle|k\rangle \rightarrow |j\rangle|x^{j}k$ mod *N* \rangle , for *x* co-prime to the *L*-bit number *N*, (2) $t = 2L + 1 + \left[\log \left(2 + \frac{1}{2\epsilon} \right) \right]$ qubits initialized to $|0\rangle$, and (3) *L* qubits initialized to the state $|1\rangle$.

Outputs: The least integer $r > 0$ such that $x^r = 1 \pmod{N}$.

Runtime: $O(L^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

Next, we apply the Hadamard transformation to the first register

$$
\frac{1}{\sqrt{256}}\Big(\left| \textbf{00...0}\right\rangle _8+\dots \left| \textbf{11...1}\right\rangle _8 \Big) \left| \textbf{00...0}\right\rangle _8 \\ = 255
$$

The next step is to perform the modular exponentiation. One gets
\n
$$
R_{0} | R_{1} \rangle = \frac{1}{\sqrt{256}} \Big(|0\rangle |1\rangle + |1\rangle |7\rangle + |2\rangle |4\rangle + |3\rangle |13\rangle + |4\rangle |1\rangle + |5\rangle |7\rangle ... |255\rangle |13\rangle \Big)
$$
\n
$$
\frac{1}{\sqrt{256}} \Big(|0\rangle |1\rangle + |1\rangle |1\rangle + |2\rangle |4\rangle + |3\rangle |13\rangle + |4\rangle |1\rangle + |5\rangle |7\rangle ... |255\rangle |13\rangle \Big)
$$
\nwhich can be rewritten as
\n
$$
= \frac{1}{\sqrt{4}} \Big(\frac{|0\rangle + |4\rangle + ... + |252\rangle}{\sqrt{64}} \Big) |1\rangle + \frac{1}{\sqrt{4}} \Big(\frac{|1\rangle + |5\rangle + ... + |253\rangle}{\sqrt{64}} \Big) |7\rangle
$$
\n
$$
= \frac{1}{\sqrt{4}} \Big(\frac{|2\rangle + |6\rangle + ... + |254\rangle}{\sqrt{64}} \Big) |4\rangle + \frac{1}{\sqrt{4}} \Big(\frac{|3\rangle + |7\rangle + ... + |255\rangle}{\sqrt{64}} \Big) |13\rangle
$$
\n
$$
+ \frac{1}{\sqrt{4}} \Big(\frac{|2\rangle + |0\rangle + ... + |254\rangle}{\sqrt{64}} \Big) |4\rangle + \frac{1}{\sqrt{4}} \Big(\frac{|3\rangle + |7\rangle + ... + |255\rangle}{\sqrt{64}} \Big) |13\rangle
$$
\nDetermine C11.CILUTE

DA CONTINUARE SU QISKIT

x α = 1(mode α) none of α = 10 α = to the equation *^x*² = 1(mod *^N*) in the range 1 [≤] *^x* [≤] *^N*, that is, neither gcd(*x* − 1*, N*) and gcd(*x* + 1*, N*) is a non-trivial factor of *N* that can be

The factoring problem turns out to be equivalent to the order-finding problem we just studied, in the sense that a fast algorithm for orderfinding can easily be turned into a fast algorithm for factoring. The reduction of factoring to order-finding proceeds in two basic steps. *x* duction or nactor

The first step is to show that we can compute a factor of N if we can find
2 non-trivial solution x nog + 1(mod N) to the equation x² = 1(mod N) a non-trivial solution x neq \pm 1(mod N) to the equation $x^2 = 1$ (mod N).

The second step is to show that a randomly chosen y co-prime to N is quite likely to have an order r which is even, and such that $y^{r/2}$ neq \pm $1(\text{mod N})$. Thus $x \equiv y^{r/2}(\text{mod N})$ is a non-trivial solution to $x^2 = 1(\text{mod N})$. N). we may find a complete prime factorization of *N. Theory is a non-dividi solution to x* $-$ 1(mod
1)

The algorithm runs as follows.

Inputs: A composite number *N*

Outputs: A non-trivial factor of *N*.

Runtime: $O((\log N)^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

- 1. If *N* is even, return the factor 2.
- 2. Determine whether $N = a^b$ for integers $a > 1$ and $b > 2$, and if so return the factor *a* (uses the classical algorithm of Exercise 5.17).
- 3. Randomly choose *x* in the range 1 to $N-1$. If $gcd(x, N) > 1$ then return the factor $gcd(x, N)$.
- Use the order-finding subroutine to find the order r of x modulo N
- 5. If *r* is even and $x^{r/2} \neq -1$ (mod *N*) then compute gcd($x^{r/2} 1$, *N*) and $gcd(x^{r/2} + 1, N)$, and test to see if one of these is a non-trivial factor, returning that factor if so. Otherwise, the algorithm fails.

Steps 1 and 2 of the algorithm either return a factor, or else ensure that *N* is an

Steps 1 and 2 of the algorithm either return a factor, or else ensure that N is an odd integer with more than one prime factor. These steps may be performed using $O(1)$ and $O(L^3)$ operations, respectively.

Step 3 either returns a factor, or produces a randomly chosen element x of $\{0, 1, 2, \ldots, N-1\}$, co-prime to N.

Step 4 calls the order-finding subroutine, computing the order r of x modulo N.

Step 5 completes the algorithm, since Theorem 5.3 of Nielsen & Chuang guarantees that with probability at least one-half, r will be even and $x^{r/2}$ neq − 1(mod N), and then Theorem 5.2 of Nielsen & Chuang guarantees that either gcd($x^{r/2}$ – 1, N) or gcd($x^{r/2}$ + 1,N) is a non-trivial factor of N.

Example: Factoring $N = 15$.

15 is neither even, nor of the form a^b with a greater or equal to 1 and b greater or equal to 2. Therefore steps 1 and 2 do not return anything.

Step 3 requires to pick randomly a number between 1 and d 14. Following the previous example, suppose we choose $x = 7$. It is co-prime to 15.

Step 4 makes use of the order finding algorithm to find the order r of $x =$ 7 mod N = 15. We saw that the output is $r = 4$.

By chance, 4 is even, and more over $x^{r/2}$ mod $15 = 4$ differ from -1 mod 15, so the algorithm works. Computing the greatest common divisor $gcd(x^2 - 1, 15) = 3$ and $gcd(x^2 + 1, 15) = 5$ tells us that $15 = 3 \times 5$.