Quantum Computation and Shor's Factoring Algorithm

Ronald de Wolf
CWI and University of Amsterdam
rdewolf@cwi.nl

January 12, 1999

Abstract

The field of *quantum computation* studies the power of computers that are based on quantum-mechanical principles. We give a brief introduction to the model of quantum computation and to its main success so far: Peter Shor's efficient quantum algorithm for factoring integers.

1 Introduction

Today's computers—both in theory (Turing machines) and practice (PCs)—are based on classical physics. They are limited by locality (operations have only local effects) and by the classical fact that systems can be in only one state at the time. However, modern quantum physics tells us that the world behaves quite differently: some operations can have non-local effects and in some sense quantum systems can be in several states simultaneously.

Quantum computation is the field that investigates the computational power and other properties of computers based on quantum-mechanical principles. Its main objective is to find quantum algorithms that are significantly faster than any classical algorithm solving the same problem. The field started in the early 1980s with suggestions by Paul Benioff [Ben82] and Richard Feynman [Fey82, Fey85] and reached more rigorous ground when in 1985 David Deutsch defined the universal quantum Turing machine [Deu85]. The following years saw only sparse activity, but the field accelerated explosively after Peter Shor's 1994 discovery of efficient quantum algorithms for the problems of integer factorization and discrete logarithms [Sho97]. Since most of current classical cryptography is based on the assumption that these two problems are computationally hard, the ability to actually build and use a quantum computer would allow to break most current cryptographic systems (notably the RSA system [RSA78, Riv90]).

This paper is intended to be a brief and incomplete introduction to the model of quantum computation and Shor's factoring algorithm, which is widely considered to be quantum computing's biggest success so far.[1] The paper is aimed at mathematicians and computer scientists. Some familiarity with computational complexity theory will be useful, but is not necessary for understanding the paper. We start with an abstract explanation of quantum mechanics in Section 2. Section 3 explains what quantum bits and quantum memory look like, and Section 4 shows how we can compute with quantum memory. Finally, Section 5 explains the factoring algorithm in some detail.

[1] Shor's discrete log algorithm is similar to the factoring algorithm. Important topics related to quantum computation that will go unmentioned here are Grover's algorithm for database search, quantum information and communication complexity, teleportation, quantum error-correcting codes, quantum cryptography, and potential physical implementations of quantum computers. The interested reader is referred to the wealth of papers available at the Los Alamos preprint archive http://xxx.lanl.gov/archive/quant-ph.

2 Quantum Mechanics

Here we give a brief introduction to quantum mechanics. In short: a quantum state is a *superposition* of classical states, to which we can apply either a *measurement* or a *unitary operation*.

2.1 Superposition

Consider some physical system which can be in N different, mutually exclusive classical states. Call these states $|1\rangle, |2\rangle, \ldots, |N\rangle$. Roughly, by a "classical" state we mean a state in which the system can be found if we observe it. A *quantum* state $|\phi\rangle$ is a *superposition* of classical states, written

$$|\phi\rangle = \alpha_1|1\rangle + \alpha_2|2\rangle + \ldots + \alpha_N|N\rangle.$$

Here α_i is a complex number which is called the *amplitude* of $|i\rangle$ in $|\phi\rangle$. Intuitively, a system in quantum state $|\phi\rangle$ is in *all* classical states *at the same time*! It is in state $|1\rangle$ with amplitude α_1, in state $|2\rangle$ with amplitude α_2, and so on. Mathematically, the states $|1\rangle, \ldots, |N\rangle$ form an orthonormal basis of a *Hilbert space* of dimension N, and a quantum state is a vector (of norm 1, see below) in this space.

2.2 Measurement

There are two things we can do with a quantum state: observe (measure) it, or let it evolve unitarily without measuring it. Suppose we observe state $|\phi\rangle$. We cannot "see" a superposition itself, but only classical states. Accordingly, if we observe state $|\phi\rangle$ we will see one and only one classical state $|j\rangle$. Which specific $|j\rangle$ will we see? This is not determined in advance; the only thing we can say is that we will see state $|j\rangle$ with probability $|\alpha_j|^2$. Thus observing a quantum state induces a probability distribution on the classical states, given by the squared amplitudes. Note that we must have $\sum_{j=1}^{N} |\alpha_j|^2 = 1$, so the vector of amplitudes has (Euclidean) norm 1.

Suppose we observe $|\phi\rangle$ and see classical state $|j\rangle$ as a result. Then $|\phi\rangle$ itself has "disappeared", and all that is left is $|j\rangle$. In other words, observing $|\phi\rangle$ "collapses" the quantum superposition $|\phi\rangle$ to the classical state $|j\rangle$ which we saw, and all "information" that might have been contained in the amplitudes α_j is gone.

2.3 Unitary Evolution

Instead of measuring $|\phi\rangle$, we can also apply some operation on it, i.e. change the state to some

$$|\psi\rangle = \beta_1|1\rangle + \beta_2|2\rangle + \ldots + \beta_N|N\rangle.$$

Quantum mechanics only allows *linear* operations to be applied to quantum states. What this means is: if we view a state like $|\phi\rangle$ as an N-dimensional vector $(\alpha_1, \ldots, \alpha_N)^T$, then applying an operation which changes $|\phi\rangle$ to $|\psi\rangle$ correspond to multiplying $|\phi\rangle$ with an $N \times N$ matrix U:

$$U \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_N \end{pmatrix}.$$

Because measuring $|\psi\rangle$ should also give a probability distribution, we have $\sum_{j=1}^{N} |\beta_j|^2 = 1$. This implies that the operation U must preserve the norm of vectors, and hence must be a *unitary*

transformation.A matrix U is *unitary* if its inverse U^{-1} equals its conjugate transpose U^*. This is equivalent to saying that U always maps a vector of norm 1 to a vector of norm 1. Because a unitary transformation always has an inverse, it follows that any (non-measuring) operation on quantum states must be reversible. On the other hand, a measurement is clearly non-reversible, because we cannot reconstruct $|\phi\rangle$ from the observed classical state $|j\rangle$.

3 Quantum Memory

In classical computation the unit of information is a *bit*, which can be 0 or 1. In *quantum* computation, this unit is a *quantum* bit (*qubit*), which is a superposition of 0 and 1. Consider a system with 2 basis states, call them $|0\rangle$ and $|1\rangle$. A single qubit can be in any superposition

$$\alpha_0|0\rangle + \alpha_1|1\rangle, \ |\alpha_0|^2 + |\alpha_1|^2 = 1.$$

Similarly we can think of systems of more than 1 qubit. For instance, a 2-qubit system has 4 basis states: $|0\rangle|0\rangle$, $|0\rangle|1\rangle$, $|1\rangle|0\rangle$, $|1\rangle|1\rangle$. Here for instance $|1\rangle|0\rangle$ means that the first qubit is in its basis state $|1\rangle$, the second is in its basis state $|0\rangle$.

More generally, a register of n qubits has 2^n basis states, each of the form $|b_1\rangle|b_2\rangle \ldots |b_n\rangle$, with $b_i \in \{0,1\}$. We can abbreviate this to $|b_1 b_2 \ldots b_n\rangle$. Since bitstrings of length n can be viewed as numbers between 0 and 2^n-1, we can also write the basis states as numbers $|0\rangle, |1\rangle, |2\rangle, \ldots, |2^n-1\rangle$. A quantum register of n qubits can be in any superposition

$$\alpha_0|0\rangle + \alpha_1|1\rangle + \ldots + \alpha_{2^n-1}|2^n-1\rangle, \ \sum_{j=0}^{2^n-1} |\alpha_j|^2 = 1.$$

Note that we need 2^n complex numbers to completely specify the state of an n-qubit system, whereas we need only n bits to specify the state of a classical n-bit system. Thus quantum memory can contain vastly more information (in some sense) than classical memory. The art of quantum computing is to use this information for interesting computational purposes.

4 Quantum Computation

Below we explain how a quantum computer can apply computational steps to its register of qubits. Two equivalent models exist for this: the quantum Turing machine [Deu85, BV97] and the quantum circuit model [Deu89, Yao93]. We only explain the latter, which is more popular among researchers.

4.1 Classical Circuits

In classical complexity theory, a *Boolean circuit* is a finite directed acyclic graph with AND, OR, and NOT gates. It has n input nodes, which contain the n input bits ($n \geq 0$). The internal nodes are AND, OR, and NOT gates, and there are one or more designated output nodes. The initial input bits are fed into AND, OR, and NOT gates according to the circuit, and eventually the output nodes assume some value (see figure 1). We say that a circuit *computes* some Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ if the output nodes get the right value $f(x)$ for every input $x \in \{0,1\}^n$.

A *circuit family* is a set $\mathcal{C} = \{C_n\}$ of circuits, one for each input size n. Each circuit has one output bit. Such a family *recognizes* or *decides* a language $L \subseteq \{0,1\}^*$ if, for every n and every input $x \in \{0,1\}^n$, the circuit C_n outputs 1 if $x \in L$ and outputs 0 otherwise. Such a circuit family is *uniformly polynomial* if there is a deterministic Turing machine that outputs C_n given

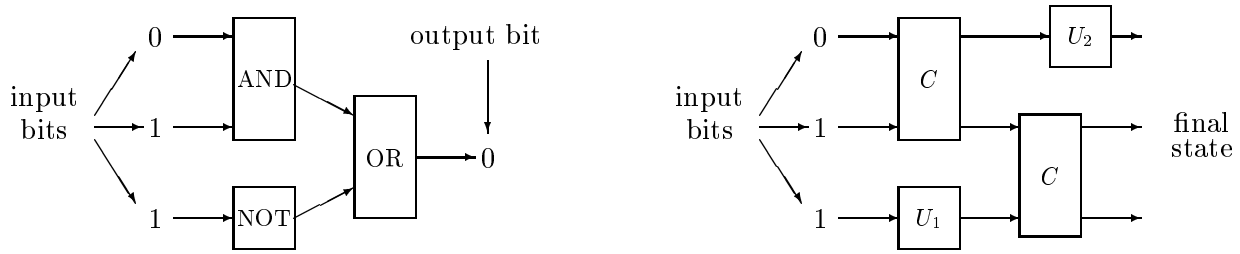

Figure 1: A classical (left) and a quantum (right) circuit. U_1 and U_2 are 1-qubit gates.

n as input using space logarithmic in n (this implies time polynomial in n). Note that the size (number of gates) of the circuits C_n can then grow at most polynomially with n. It is known that uniformly polynomial circuit families are equal in power to polynomial-time deterministic Turing machines: a language L can be decided by a uniformly polynomial circuit family iff $L \in \mathbf{P}$ [Pap94, Theorem 11.5].

Similarly we can consider *randomized* circuits. These receive, in addition to the n input bits, also some random bits ("coin flips") as input. A randomized circuit computes a function f if it successfully outputs the right answer $f(x)$ with probability at least $2/3$ for every x (probability taken over the values of the random bits; the $2/3$ may be replaced by any $1/2 + \varepsilon$). Randomized circuits are equal in power to randomized Turing machines: a language L can be decided by a uniformly polynomial randomized circuit family iff $L \in \mathbf{BPP}$, where $\mathbf{BPP}$ ("Bounded-error Probabilistic Polynomial time") is the class of languages that can efficiently be recognized by randomized Turing machines with small error probability. Clearly $\mathbf{P} \subseteq \mathbf{BPP}$. It is unknown whether this inclusion is strict.

4.2 Quantum Circuits

A *quantum circuit* (also called quantum network or quantum gate array) generalizes the idea of classical circuit families, replacing the AND, OR, and NOT gates by *quantum gates*. A quantum gate is a unitary transformation on a small (usually 1, 2, or 3) number of qubits. Mathematically, these gates can be composed by taking tensor products (if gates are applied in parallel to different parts of the register) and ordinary products (if gates are applied sequentially).

A simple but widely used example of a 1-qubit gate is the *Hadamard* transform, specified by:

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

If we apply H to initial state $|0\rangle$ and then observe, we have equal probability of observing $|0\rangle$ or $|1\rangle$. Similarly, applying H to $|1\rangle$ and observing gives equal probability of $|0\rangle$ or $|1\rangle$. However, if we apply H to the superposition $(1/\sqrt{2})|0\rangle + (1/\sqrt{2})|1\rangle$ then we obtain $|0\rangle$: the positive and negative amplitudes for $|1\rangle$ cancel out! This effect is called *interference*, and is analogous to interference patterns between light or sound waves. Note that if we apply H to each bit in a register of n zeroes, we obtain $(1/\sqrt{2^n}) \sum_{x \in \{0,1\}^n} |x\rangle$, which is a superposition of all n-bit strings.

An example of a 2-qubit gate is the *controlled-not* gate C. This negates the second bit b of its input if the first bit is 1, and does nothing if the first bit is 0:

$$C|0\rangle|b\rangle = |0\rangle|b\rangle$$
$$C|1\rangle|b\rangle = |1\rangle|1-b\rangle$$

As in the classical case, a quantum circuit is a finite directed acyclic graph of input leaves, gates, and output nodes (see figure 1). There are n leaves that contain the input (as classical bits); in addition we may have some more input leaves which are initially 0 ("workspace"). The internal nodes of the quantum circuit are quantum gates that each operate on at most 2 qubits of the state. It is known that the set of all 1-qubit operations together with the 2-qubit controlled-not gate is universal [BBC$^+$95], meaning that any other unitary transformation can be built from them. Allowing all 1-qubit gates is not very realistic from an implementational point of view, as there are uncountably many of them. However, the model is usually restricted, only allowing a small finite set of 1-qubit gates from which all other 1-qubit gates can be well approximated. The gates in the circuit transform the initial state vector into a final state, which will generally be a superposition. We observe some dedicated output bits of this final state to (probabilistically) obtain an answer.

The classical classes **P** and **BPP** can now be generalized as follows. **EQP** ("Exact Quantum Polynomial time") is the class of languages that can be recognized with success probability 1 by uniformly polynomial quantum circuits. **BQP** ("Bounded-error Quantum Polynomial time") is the class of languages that can be recognized with success probability at least 2/3 by uniformly polynomial quantum circuits. It can be shown that $\mathbf{P} \subseteq \mathbf{EQP}$ and $\mathbf{BPP} \subseteq \mathbf{BQP}$. The main open question of quantum complexity theory is whether these inclusions are strict.

One uniquely quantum-mechanical effect that we can use for building quantum algorithms is *quantum parallelism*. Suppose we have a classical algorithm that computes some function $f : \{0,1\}^n \to \{0,1\}^m$. Then we can build a quantum circuit U that maps $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ for every $x \in \{0,1\}^n$. Suppose we apply U to a superposition of *all* inputs x (which is easy to build using the Hadamard transform):

$$U\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|f(x)\rangle.$$

We applied U just once, but the final superposition contains $f(x)$ for *all* 2^n input values x! However, by itself this is not very useful, since observing the final superposition will give just one random $|x\rangle|f(x)\rangle$. All other information will be lost.

A second important effect that can be used is *entanglement*, which refers to quantum correlations between different qubits. For instance, consider a 2-qubit register that is in the state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle.$$

Initially neither of the two qubits has a classical value $|0\rangle$ or $|1\rangle$. However, if we measure the first qubit and observe, say, a $|0\rangle$, then the whole state collapses to $|00\rangle$. Thus observing only the first qubit immediately fixes also the second, unobserved qubit to a classical value. Therefore this system is called *entangled*. Since the two qubits that make up the register may be far apart, this example illustrates some of the non-local effects that quantum systems can exhibit.

5 Shor's Factoring Algorithm

Probably the most important quantum algorithm so far is Shor's factoring algorithm [Sho97]. It can find a factor of a composite number N in $\widetilde{O}((\log N)^2)$ steps, which is polynomial in the length $\log N$ of the input (the $\widetilde{O}$-notation ignores some $\log\log$-factors). On the other hand, it is widely conjectured that a classical (deterministic or randomized) computer cannot factor N in polynomial

time—in fact, much of modern cryptography is based on this conjecture. The best known classical randomized algorithms run in time roughly

$$2^{(\log N)^\alpha},$$

where $\alpha = 1/3$ for a heuristic upper bound [LL93] and $\alpha = 1/2$ for a rigorous upper bound [LP92]. In terms of complexity classes: factoring (rather, the decision problem equivalent to it) is in **BQP** but is widely believed not to be in **BPP**. If the latter belief is true, the quantum computer would be the first counterexample to the "strong" Church-Turing thesis, which states that all "reasonable" models of computation are polynomially equivalent (see [EB90] and [Pap94, p.31,36]).

5.1 Reduction to Period-Finding

Shor's algorithm finds a factor by finding the period of some sequence. We first show how efficient period-finding suffices for efficient factoring. Suppose we want to find factors of the composite number $N > 1$. Randomly choose some integer $x \in \{2, \ldots, N-1\}$. If the greatest common divisor of x and N is greater than 1, then this gcd will be a non-trivial factor of N, so then we are done. If $gcd(x, N) = 1$, then consider the sequence

$$1 = x^0 \bmod N, x^1 \bmod N, x^2 \bmod N, \ldots$$

This sequence will cycle after a while: there is a least $0 < r \leq N$ such that $x^r = 1 \bmod N$. This r is called the *period* of the sequence. It can be shown that with probability $\geq 1/4$, r is even and $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N. In that case:

$$\begin{aligned} x^r &\equiv 1 \bmod N \quad \Longleftrightarrow \\ (x^{r/2})^2 &\equiv 1 \bmod N \quad \Longleftrightarrow \\ (x^{r/2}+1)(x^{r/2}-1) &\equiv 0 \bmod N \quad \Longleftrightarrow \\ (x^{r/2}+1)(x^{r/2}-1) &= kN \text{ for some } k. \end{aligned}$$

Note that $k > 0$ because both $x^{r/2} + 1 > 0$ and $x^{r/2} - 1 > 0$ ($x > 1$). Hence $x^{r/2} + 1$ or $x^{r/2} - 1$ will share a factor with N. Because $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N this factor will be $< N$, and in fact *both* these numbers will share a non-trivial factor with N. Accordingly, if we have r then we can efficiently (in $\widetilde{O}(\log N)$ steps) compute the greatest common divisors $gcd(x^{r/2} + 1, N)$ and $gcd(x^{r/2} - 1, N)$, and both of these two numbers will be non-trivial factors of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

Thus the problem of factoring reduces to finding r. We will show how the quantum Fourier transform enables us to do this.

5.2 The Quantum Fourier Transform

For some number q, let $Z_q = \{0, \ldots, q-1\}$. For each $a \in Z_q$ define a function $\chi_a : Z_q \to \mathbf{C}$ by

$$\chi_a(b) = e^{2\pi i \frac{ab}{q}}.$$

The set of basis states $\{|a\rangle \mid a \in Z_q\}$ is called the *standard basis*. An alternative orthonormal basis, called the *Fourier basis*, is the set $\{|\chi_a\rangle \mid a \in Z_q\}$ defined by

$$|\chi_a\rangle = \frac{1}{\sqrt{q}} \sum_{b \in Z_q} \chi_a(b)|b\rangle.$$

The quantum Fourier transform (QFT) is the unitary transformation that maps the standard basis to the Fourier basis:

$$\text{QFT:}\ \ |a\rangle \to |\chi_a\rangle.$$

It is known that if q is smooth (meaning that all factors of q are $O(\log q)$, for instance q is a power of 2), then the QFT can be implemented on a quantum computer using $O((\log q)^2)$ elementary gates [Cop94, Cle94, CEMM98].

5.3 Easy Case: r Divides q

Assume we have picked a random x as in Section 5.1, and we want to find the corresponding period r. We can always efficiently pick some smooth q such that $N^2 < q \leq 2N^2$ (for instance take q a power of 2). The QFT for Z_q can be implemented using $O((\log q)^2) = O((\log N)^2)$ elementary gates.

We will first assume that the unknown r divides q, in which case everything works out smoothly.

It is known that in $\widetilde{O}((\log N)^2)$ steps we can compute the transformation $|a\rangle|0\rangle \to |a\rangle|x^a \bmod N\rangle$ using the Schönhage-Strassen algorithm for fast multiplication (see [Knu97]). We now find r as follows. Start with $|0\rangle|0\rangle$, two registers of $\lceil \log q \rceil$ and $\lceil \log N \rceil$ zeroes, respectively. Apply the QFT to the first register to build

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0\rangle.$$

Then compute $x^a \bmod N$ in quantum parallel:

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|x^a \bmod N\rangle.$$

Observing the second register gives some $x^s \bmod N$, with $s < r$. Note that because r divides q, the a of the form $a = jr+s$ $(0 \leq j < q/r)$ are exactly the a for which $x^a \bmod N$ equals the observed value $x^s \bmod N$. Thus the first register collapses to a superposition of $|s\rangle, |r+s\rangle, |2r+s\rangle, \ldots, |q-r+s\rangle$ and the second register collapses to the classical state $|x^s \bmod N\rangle$. We can now ignore the second register, and have in the first:

$$\sqrt{\frac{r}{q}} \sum_{j=0}^{q/r-1} |jr+s\rangle.$$

Applying the QFT again gives

$$\sqrt{\frac{r}{q}} \sum_{j=0}^{q/r-1} \frac{1}{\sqrt{q}} \sum_{b=0}^{q-1} e^{2\pi i \frac{(jr+s)b}{q}} |b\rangle = \frac{\sqrt{r}}{q} \sum_{b=0}^{q-1} e^{2\pi i \frac{sb}{q}} \left(\sum_{j=0}^{q/r-1} e^{2\pi i \frac{jrb}{q}} \right) |b\rangle.$$

Using that $\sum_{j=0}^{n-1} a^j = (1-a^n)/(1-a)$ for $a \neq 1$, we compute:

$$\sum_{j=0}^{q/r-1} e^{2\pi i \frac{jrb}{q}} = \sum_{j=0}^{q/r-1} \left(e^{2\pi i \frac{rb}{q}} \right)^j = \begin{cases} q/r & \text{if } e^{2\pi i \frac{rb}{q}} = 1 \\ \frac{1-\left(e^{2\pi i \frac{rb}{q}}\right)^{q/r}}{1-e^{2\pi i \frac{rb}{q}}} = \frac{1-e^{2\pi i b}}{1-e^{2\pi i \frac{rb}{q}}} = 0 & \text{if } e^{2\pi i \frac{rb}{q}} \neq 1 \end{cases}$$

Note that $e^{2\pi i rb/q} = 1$ iff rb/q is an integer iff b is a multiple of q/r. Accordingly, we are left with a superposition where only the multiples of q/r have non-zero amplitude. Observing this final

superposition gives some random multiple $b = cq/r$, with c a random number $0 \leq c < r$. Thus we get a b such that

$$\frac{b}{q} = \frac{c}{r},$$

where b and q are known and c and r are unknown. There are $\phi(r) \in \Omega(r/\log\log r)$ numbers smaller than r which are coprime to r [HW79, Theorem 328], so c will be coprime to r with probability $\Omega(1/\log\log r)$. Accordingly, an expected number of $O(\log\log N)$ repetitions of the procedure of this section suffices to obtain a $b = cq/r$ with c coprime to r. Once we have such a b, we can obtain r as the denominator by writing b/q in lowest terms.

5.4 Hard Case: r Does not Divide q

In case r does not divide q (which is actually quite likely), it can be shown that applying exactly the same algorithm will still yield with high probability a b such that

$$\left|\frac{b}{q} - \frac{c}{r}\right| \leq \frac{1}{2q},$$

with b, q known and c, r unknown. Two distinct fractions, each with denominator $\leq N$, must be at least $1/N^2 > 1/q$ apart.[2] Therefore c/r is the only fraction with denominator $\leq N$ at distance $\leq 1/2q$ from b/q. Applying continued-fraction expansion (see [HW79, Chapter X]) to b/q efficiently gives us the fraction with denominator $\leq N$ that is closest to b/q. This fraction must be c/r. Again, with good probability c and r will be coprime, in which case writing c/r in lowest terms gives r. The whole algorithm finds a factor of N in expected time $\widetilde{O}((\log N)^2)$.

References

[BBC+95] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995. quant-ph/9503016.

[Ben82] P. A. Benioff. Quantum mechanical Hamiltonian models of Turing machines. *Journal of Statistical Physics*, 29(3):515–546, 1982.

[BV97] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. Earlier version in STOC'93.

[CEMM98] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. In *Proceedings of the Royal Society of London*, volume A454, pages 339–354, 1998. quant-ph/9708016.

[Cle94] R. Cleve. A note on computing Fourier transforms by quantum programs. Unpublished. Available at `http://www.cpsc.ucalgary.ca/~cleve/publications.html`, 1994.

[Cop94] D. Coppersmith. An approximate Fourier transform useful in quantum factoring. IBM Research Report No. RC19642, 1994.

[2] Consider two fractions $z = x/y$ and $z' = x'/y'$ with $y, y' \leq N$. If $z \neq z'$ then $|xy' - x'y| \geq 1$, and hence $|z - z'| = |(xy' - x'y)/yy'| \geq 1/N^2$.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle, and the universal quantum Turing machine. In *Proceedings of the Royal Society of London*, volume A400, pages 97–117, 1985.

[Deu89] D. Deutsch. Quantum computational networks. In *Proceedings of the Royal Society of London*, volume A425, pages 73–90, 1989.

[EB90] P. van Emde Boas. Machine models and simulations. In van Leeuwen [vL90], pages 1–66.

[Fey82] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982.

[Fey85] R. Feynman. Quantum mechanical computers. *Optics News*, 11:11–20, 1985.

[HW79] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, New York, fifth edition, 1979.

[Knu97] D. E. Knuth. *The Art of Computer Programming. Volume 2: Seminumerical Algorithms.* Addison-Wesley, third edition, 1997.

[LL93] A. K. Lenstra and H. W. Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics.* Springer, 1993.

[LP92] H. W. Lenstra and C. Pomerance. A rigorous time bound for factoring integers. *Journal of the American Mathematical Society*, 5:483–516, 1992.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[Riv90] R. L. Rivest. Cryptography. In van Leeuwen [vL90], pages 717–755.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. Earlier version in FOCS'94; also quant-ph/9508027.

[vL90] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity.* MIT Press, Cambridge, MA, 1990.

[Yao93] A. C-C. Yao. Quantum circuit complexity. In *Proceedings of 34th FOCS*, pages 352–360, 1993.