SIAM REVIEW
Vol. 41, No. 2, pp. 303–332

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract. A digital computer is generally believed to be an efficient universal computing device; that is, it is believed to be able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems that are generally thought to be hard on classical computers and that have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, for example, the number of digits of the integer to be factored.

Key words. algorithmic number theory, prime factorization, discrete logarithms, Church's thesis, quantum computers, foundations of quantum mechanics, spin systems, Fourier transforms

AMS subject classifications. 81P10, 11Y05, 68Q10, 03D10

PII. S0036144598347011

1. Introduction.

One of the first results in the mathematics of computation, which underlies the subsequent development of much of theoretical computer science, was the distinction between computable and noncomputable functions shown in the papers of Church [1936], Post [1936], and Turing [1936]. The observation that several apparently different definitions of what it meant for a function to be computable yielded the same set of computable functions led to the proposal of Church's thesis, which says that all computing devices can be simulated by a Turing machine. This thesis greatly simplifies the study of computation, since it reduces the potential field of study from any of an infinite number of potential computing devices to Turing machines. Church's thesis is not a mathematical theorem; to make it one would require a precise mathematical description of a computing device. Such a description, however, would leave open the possibility of some practical computing device that did not satisfy this precise mathematical description and thus would make the resulting theorem weaker than Church's original thesis.

With the development of practical computers, it became apparent that the distinction between computable and noncomputable functions was much too coarse; com-

*Published electronically April 23, 1999. This paper originally appeared in *SIAM Journal on Computing*, Volume 26, Number 5, 1997, pages 1484 to 1509.
http://www.siam.org/journals/sirev/41-2/34701.html

†AT&T Labs–Research, Room C237, 180 Park Avenue, Florham Park, NJ 07932 (shor@research.att.com).

puter scientists are now interested in the exact efficiency with which specific functions can be computed. This exact efficiency, on the other hand, was found to be too precise a quantity to work with easily. The generally accepted compromise between coarseness and precision distinguishes between efficiently and inefficiently computable functions by whether the length of the computation scales polynomially or superpolynomially with the input size. The class of problems that can be solved by algorithms having a number of steps polynomial in the input size is known as P.

For this classification to make sense, it must be machine independent. That is, the question of whether a function is computable in polynomial time must be independent of the type of computing device used. This corresponds to the following quantitative version of Church's thesis, which has been called the "strong Church's thesis" by Vergis, Steiglitz, and Dickinson [1986] and which makes up half of the "invariance thesis" of van Emde Boas [1990].

THESIS 1.1 (quantitative Church's thesis). *Any physical computing device can be simulated by a Turing machine in a number of steps polynomial in the resources used by the computing device.*

Readers who are not comfortable with Turing machines may think instead of digital computers that have an amount of memory that grows linearly with the length of the computation, as these two classes of computing machines can efficiently simulate each other. In statements of this thesis, the Turing machine is sometimes augmented with a random number generator, as it has not yet been determined whether there are pseudorandom number generators that can efficiently simulate truly random number generators for all purposes.

There are two escape clauses in the above thesis. One of these is the word "physical." Researchers have produced machine models that violate the above quantitative Church's thesis, but most of these have been ruled out by the fact that they are not physical, that is, they could not be built and made to work. The other escape clause in the above thesis is the word "resources," the meaning of which is not completely specified above. There are generally two resources that limit the ability of digital computers to solve large problems: time (computational steps) and space (memory). There are more resources pertinent to analog computation; some proposed analog machines that seem able to solve NP-complete problems in polynomial time have required exponentially precise parts or an exponential amount of energy. (See Vergis, Steiglitz, and Dickinson [1986] and Steiglitz [1988]; this issue is also implicit in the papers of Canny and Reif [1987] and Choi, Sellen, and Yap [1995] on three-dimensional shortest paths.)

For quantum computation, in addition to space and time, there is also a third potentially important resource: precision. For a quantum computer to work, at least in any currently envisioned implementation, it must be able to make changes in the quantum states of objects (e.g., atoms, photons, or nuclear spins). These changes clearly cannot be perfectly accurate but must contain some small amount of inherent imprecision. If this imprecision is constant (i.e., it does not depend on the size of the input), then it is not known how to compute any functions in polynomial time on a quantum computer that cannot also be computed in polynomial time on a classical computer with a random number generator.[1] However, if we let the precision grow polynomially in the input size (so the number of *bits* of precision grows logarithmically in the input size), we appear to obtain a more powerful type of computer. Allowing

[1] This was true when this article was originally written. It is no longer true; see the discussion in section 8.

the same polynomial growth in precision does not appear to confer extra computing power to classical mechanics, although allowing exponential growth in precision may [Hartmanis and Simon, 1974; Vergis, Steiglitz, and Dickinson, 1986].

As far as we know, the precision possible in quantum state manipulation is dictated not by fundamental physical laws but by the properties of the materials from which and the architecture with which a quantum computer is built. It is currently not clear which architectures, if any, will give high precision and what this precision will be. If the precision of a quantum computer is large enough to make it more powerful than a classical computer, then in order to understand its potential it is important to think of precision as a resource that can vary. Treating the precision as a large constant (even though it is almost certain to be constant for any given machine) would be comparable to treating a classical digital computer as a finite automaton: since any given computer has a fixed amount of memory, this view is technically correct; however, it is not particularly useful.

Because of the remarkable effectiveness of our mathematical models of computation, computer scientists have tended to forget that computation is dependent on the laws of physics. This can be seen in the statement of the quantitative Church's thesis in van Emde Boas [1990], where the word "physical" in the above phrasing is replaced by the word "reasonable." It is difficult to imagine any definition of "reasonable" in this context that does not mean "physically realizable," that is, that this computing machine could actually be built and would work.

Computer scientists have become convinced of the truth of the quantitative Church's thesis through the failure of all proposed counterexamples. Most of these proposed counterexamples have been based on the laws of classical mechanics; however, the universe is in reality quantum mechanical. Quantum mechanical objects often behave quite differently from how our intuition, based on classical mechanics, tells us they should. It thus seems plausible that the natural computing power of classical mechanics corresponds to that of Turing machines,[2] while the natural computing power of quantum mechanics might be greater.

The first person to look at the interaction between computation and quantum mechanics appears to have been Benioff [1980, 1982a, 1982b]. Although he did not ask whether quantum mechanics conferred extra power to computation, he showed that reversible unitary evolution was sufficient to realize the computational power of a Turing machine, thus showing that quantum mechanics is computationally at least as powerful as classical computers. This work was fundamental in making later investigation of quantum computers possible.

Feynman [1982, 1986] seems to have been the first to suggest that quantum mechanics might be computationally more powerful than Turing machines. He gave arguments as to why quantum mechanics is intrinsically computationally expensive to simulate on a classical computer. He also raised the possibility of using a computer based on quantum mechanical principles to avoid this problem, thus implicitly asking the converse question, "By using quantum mechanics in a computer can you compute more efficiently than on a classical computer?" The first to ask this question explicitly was Deutsch [1985, 1989]. In order to study this question, he defined both quantum Turing machines and quantum circuits and investigated some of their properties.

[2]See Vergis, Steiglitz, and Dickinson [1986], Steiglitz [1988], and Rubel [1989]. I believe that this question has not yet been settled and is worthy of further investigation. In particular, turbulence seems a good candidate for a counterexample to the quantitative Church's thesis because the nontrivial dynamics on many length scales appear to make it difficult to simulate on a classical computer.

The question of whether using quantum mechanics in a computer allows one to obtain more computational power was more recently addressed by Deutsch and Jozsa [1992] and Berthiaume and Brassard [1992, 1994]. These papers showed that there are problems that quantum computers can quickly solve exactly, but that classical computers can only solve quickly with high probability and the aid of a random number generator. However, these papers did not show how to solve any problem in quantum polynomial time that was not already known to be solvable in polynomial time with the aid of a random number generator, allowing a small probability of error; this is the characterization of the complexity class BPP (bounded-error probabilistic polynomial time), which is widely viewed as the class of efficiently solvable problems.

Further work on this problem was stimulated by Bernstein and Vazirani [1993]. One of the results contained in their paper was an oracle problem (that is, a problem involving a "black box" subroutine that the computer is allowed to perform but for which no code is accessible) that can be done in polynomial time on a quantum Turing machine but that requires superpolynomial time on a classical computer. This result was improved by Simon [1994], who gave a much simpler construction of an oracle problem that takes polynomial time on a quantum computer but requires *exponential* time on a classical computer. Indeed, while Bernstein and Vazirani's problem appears contrived, Simon's problem looks quite natural. Simon's algorithm inspired the work presented in this paper.

Two number theory problems that have been studied extensively but for which no polynomial-time algorithms have yet been discovered are those of finding discrete logarithms and factoring integers [Pomerance, 1987; Gordon, 1993; Lenstra and Lenstra, 1993; Adleman and McCurley, 1994]. These problems are so widely believed to be hard that several cryptosystems based on their difficulty have been proposed, including the widely used RSA public key cryptosystem developed by Rivest, Shamir, and Adleman [1978]. We show that these problems can be solved in polynomial time on a quantum computer with a small probability of error.

Currently, nobody knows how to build a quantum computer, although it seems as though it might be possible within the laws of quantum mechanics. Some suggestions have been made as to possible designs for such computers [Teich, Obermayer, and Mahler, 1988; Lloyd, 1993, 1994; Cirac and Zoller, 1995; DiVincenzo, 1995; Sleator and Weinfurter, 1995; Barenco et al., 1995b; Chuang and Yamamoto, 1995; Cory, Fahmy, and Havel, 1997; Gershenfeld and Chuang, 1997; Kane 1998], but there will be substantial difficulty in building any of these [Landauer, 1995, 1997; Unruh, 1995; Chuang et al., 1995; Palma, Suominen, and Ekert, 1996]. The most difficult obstacles appear to involve the decoherence of quantum superpositions through the interaction of the computer with the environment, and the implementation of quantum state transformations with enough precision to give accurate results after many computation steps. Both of these obstacles become more difficult as the size of the computer grows, so it may turn out to be possible to build small quantum computers, while scaling up to machines large enough to do interesting computations may present fundamental difficulties.

Even if no useful quantum computer is ever built, this research does illuminate the problem of simulating quantum mechanics on a classical computer. Any method of doing this for an arbitrary Hamiltonian would necessarily be able to simulate a quantum computer. Thus, any general method for simulating quantum mechanics with at most a polynomial slowdown would lead to a polynomial-time algorithm for factoring.

The rest of this paper is organized as follows. In section 2, we introduce the model of quantum computation, the *quantum gate array,* that we use in the rest of the paper. In sections 3 and 4, we explain two subroutines that are used in our algorithms: reversible modular exponentiation in section 3 and quantum Fourier transforms in section 4. In section 5, we give our algorithm for prime factorization, and in section 6, we give our algorithm for extracting discrete logarithms. In section 7, we give a brief discussion of the practicality of quantum computation and suggest possible directions for further work. Section 8 has been added to the original paper for this *SIAM Review* reprint. It briefly mentions some developments that happened after the original paper was written, and surveys some related work.

2. Quantum Computation. In this section we give a brief introduction to quantum computation, emphasizing the properties that we will use. We will describe only *quantum gate arrays,* or *quantum acyclic circuits,* which are analogous to acyclic circuits in classical computer science. This model was originally studied by Yao [1993] and is closely related to the quantum computational networks discussed by Deutsch [1989]. For other models of quantum computers, see references on quantum Turing machines [Deutsch, 1985; Bernstein and Vazirani, 1993; Yao, 1993] and quantum cellular automata [Feynman, 1986; Margolus, 1986, 1990; Lloyd, 1993; Biafore, 1994]. If they are allowed a small probability of error, quantum Turing machines and quantum gate arrays can compute the same functions in polynomial time [Yao, 1993]. This may also be true for the various models of quantum cellular automata, but it has not yet been proved. This gives evidence that the class of functions computable in quantum polynomial time with a small probability of error is robust in that it does not depend on the exact architecture of a quantum computer. By analogy with the classical class BPP, this class is called BQP (bounded error probability quantum polynomial time).

Consider a system with n components, each of which can have two states. Whereas in classical physics, a complete description of the state of this system requires only n bits, in quantum physics, a complete description of the state of this system requires $2^n - 1$ complex numbers. To be more precise, the state of the quantum system is a point in a 2^n-dimensional vector space. For each of the 2^n possible classical positions of the components, there is a basis state of this vector space that we represent, for example, by $|011\cdots0\rangle$, meaning that the first bit is 0, the second bit is 1, and so on. Here, the *ket* notation $|x\rangle$ means that x is a (pure) quantum state. (Mixed states will not be discussed in this paper, and thus we do not define them; see a quantum theory book such as Peres [1993] for their definition.) The *Hilbert space* associated with this quantum system is the complex vector space with these 2^n states as basis vectors, and the state of the system at any time is represented by a unit-length vector in this Hilbert space. Because multiplying this state vector by a unit-length complex phase does not change any behavior of the state, we need only $2^n - 1$ complex numbers to completely describe the state. We represent this superposition of states by

$$\sum_{i=0}^{2^n-1} a_i|S_i\rangle, \tag{2.1}$$

where the amplitudes a_i are complex numbers such that $\sum_i |a_i|^2 = 1$ and each $|S_i\rangle$ is a basis vector of the Hilbert space. If the machine is measured (with respect to this basis) at any particular step, the probability of seeing basis state $|S_i\rangle$ is $|a_i|^2$; however, measuring the state of the machine projects this state to the observed basis vector $|S_i\rangle$. Thus, looking at the machine during the computation will invalidate the rest