Quantum Computing and Quantum Algorithms

DUKE VAISHNAV
Department of Computer Science & Engineering, Poornima College of Engineering, Jaipur, Rajasthan

Abstract -- Generally, a classical computer (which works on a layer of millions of transistors) is said to be an efficient computing device which can find out the solution of any computational problem in polynomial order of time .But with the advent of quantum computers and quantum algorithms it was found that such computational problems and the other problems which cannot be solved even by the classical computers were solved in much lesser time by these quantum computers. Introductory Topics covered by this paper include: introduction to quantum computing and quantum algorithms, a brief description about vector complex numbers, complex vector spaces, quantum states, quantum systems, bits, qbits and cbits .the further sections will include LAS VEGAS algorithm for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, these problems are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. This review paper is written with the aim to lay out a foundation for quantum computing in the minds of readers.

I. INTRODUCTION

The fundamental basis of quantum computation is Landauer's observation that all information is ultimately physical [1, 2]. Information, the 1's and 0's of classical computers, must inevitably be recorded by some physical system - be it paper or silicon. Which brings us to the key point. As far as we know today, all matter is composed of atoms - nuclei and electrons - and the interactions and time evolution of atoms are governed by the laws of quantum mechanics. Although the peculiarities of the quantum world may not seem readily apparent at first glance, a closer look reveals that applications of quantum mechanics are all around us (see for example Ref. [3]). As has been emphasized by Minsky [4], the very existence of atoms owes everything not to the chaotic uncertainties of classical mechanics, but rather to the *certainties* of quantum mechanics with the Pauli exclusion principle and well-defined and stable atomic energy levels! Indeed without our quantum understanding of the solid state and the band theory of metals, insulators and semiconductors, the whole of the semiconductor industry with its transistors and integrated circuits - and hence the computer on which I am writing this lecture - could not have developed. The same can be said about quantum optics and lasers: huge industries - from optical communications to music and video CDs - have their basis in these intrinsically quantum technologies.

At bottom then, everything is quantum mechanical and, like Feynman in his visionary 1959 'Plenty of Room at the Bottom' talk [5], we can certainly envisage storing bits of information on single atoms or electrons. However, these microscopic objects do *not* obey Newton's Laws of classical mechanics: instead, they evolve and interact according to the Schroedinger equation, the 'Newton's Law' of quantum mechanics. In fact, we know now that even this is only a suitable approximation for everyday speeds and energies: at high speeds and energies, we must use the Dirac equation and Einstein's relativity, with its predictions of relativistic mass increase and particle antiparticle creation, must be taken into account. However, for most of our everyday concerns, it is safe for us to ignore these complications and use the non-relativistic version of quantum mechanics embodied by Schroedinger's equation. Information is ultimately not an abstract concept - it must be recorded and stored on media that are fundamentally quantum mechanical. We must therefore broaden our definition of information as merely a string of 1's and 0's and examine the consequences of the quantum nature of media for information. The implications of this new field of quantum information theory are still being explored and may yet deliver more surprises. However, to introduce quantum computing, we shall only need a few quantum concepts and principles. But before we turn to a discussion of qubits and the like, we must now make an apparently puzzling diversion and introduce some ideas of Ed Fredkin and Charles Bennett about reversible computing and reversible logic gates.

II. REVERSIBLE COMPUTING

In 1973, Charles Bennett of IBM Research made a remarkable discovery [6]. Classical computation can be broken down into a series of steps, each logically reversible, and this in turn allows physical reversibility of the computation. This result has implications for the energy dissipated by the computation. Rolf Landauer, Bennett's long-term colleague and mentor, had earlier shown that it is the act of discarding information that incurs an unavoidable energy loss. This is Landauer's Principle and, for example, this is now central to our current understanding of the problem of Maxwell's Demon as given by Bennett [7, 8]. Bennett's result means that we can arrange our computer to calculate reversibly, very slowly, with an energy as small as we please. In his lectures on computation in the 1980's [9], Feynman discusses a reversible computer that calculates for a few steps, then drifts back a bit, 'uncalculating' as it goes, before it drifts forward again to eventually complete the calculation with almost zero energy loss.

To build such a reversible computer requires us to use new types of logic gates that are reversible, i.e. from the output of the gate one can reconstruct the input. It is easy to see that a conventional AND gate is not reversible. If the output of an AND gate is 0, the signals on the two input wires could be any one of three possibilities - 00, 01 and 10. The possibility of reversible logic gates was considered by Fredkin and Toffoli nearly 20 years ago [10]. Let us consider a simple example. The truth table for a classical NOT gate is shown below (Fig. 1). It is clearly reversible: from its output we can deduce its input. For this reason Feynman prefers to use the symmetrical notation for a NOT gate shown in Fig. 2. Two NOT gates put back to back evidently bring us back to the same place and manifestly demonstrate the reversibility. Consider now the two-input gate shown in Fig.3. This is called a 'Controlled NOT' or CN gate, since the NOT operation on the lower input line is only operative when there is a '1' on the upper input: a '0' on the upper input means that the lower bit passes through unchanged. In effect, what appears on the lower output is just the XOR operation on the two input bits (Fig. 4). However, the CN gate is more than just an XOR gate since we retain information about the control bit. This is a general feature of reversible gates: the price for reversibility is that we need to carry round extra bits of information. But, because we are not discarding any information, such a gate is, in principle, more energy efficient than a classical XOR

A	NOT A
0	1
1	0

Fig. 1 Truth Table for NOT gate.

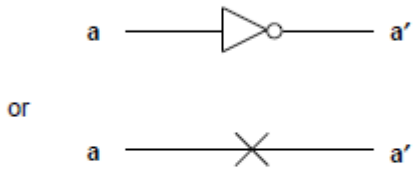

Fig. 2 Alternative symbols for NOT gate.

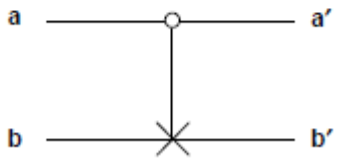

Fig. 3 Controlled NOT or CN gate.

gate.Again, as shown in Fig. 5, the CN gate can be shown to be manifestly reversible by putting two CN gates back to back. Any logical operation can be built from one of several complete sets of classical logic gates - a choice from NOT, AND, OR, XOR, NAND and so on. Similarly, one can show that there are complete sets of reversible gates that allow us to perform any logic operation. In fact, we need more than just the CN gate: we can add a Controlled Controlled NOT (CCN) or 'Toffoli' gate (Fig. 6) or a more complicated Fredkin exchange gate (Fig. 7). Why do we care about all this? Well for one thing it is possible that use of such gates may one day be needed to reduce power consumption of microprocessors implemented in CMOS silicon technology. At present, the Intel Pentium discards something like 100,000 bits per flop with each discarded bit incurring at least the minimum Landauer energy loss [11]. In our case,

however, we are interested because the laws of quantum physics are reversible in time. This guarantees that probability is conserved as a state evolves with time. Technically speaking, the Schroedinger time evolution operator is unitary and preserves the norm of quantum mechanical states (see below). To build a quantum computer with quantum states evolving according to the Schroedinger equation therefore necessarily requires us to use realisations of reversible logic gates.

a	b	a′	b′
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Fig. 4 Truth Table for Controlled NOT gate.

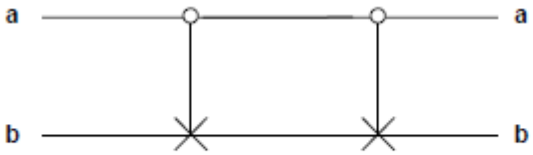

Fig. 5 CN gates are reversible.

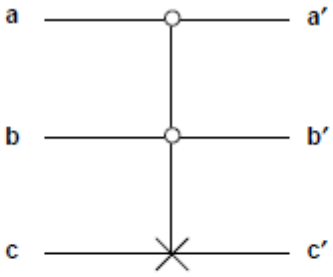

Fig. 6 Controlled Controlled NOT, CCN or Toffoli gate.

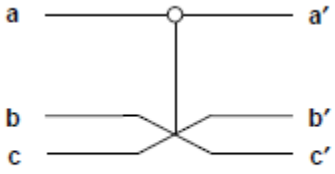

Fig. 7 Fredkin Exchange gate.

III. QUANTUM COMPLEXITY

Complexity is the study of algorithms. The 'universality' of Turing Machines makes it possible for computer scientists to classify algorithms into different 'complexity classes'. For example, multiplication of two N x N matrices requires an operation count that grows like N3 with the size of the matrix. This can be analysed in detail for a simple Turing machine implementation of the algorithm. However, the important point about 'universality' is that although you may be able to multiply matrices somewhat faster than on a Turing machine, you cannot change from an N3 growth of operations no matter what Pentium chip or special purpose matrix multiply hardware you choose to use. Thus algorithms, such as matrix multiply, for which execution time and resources grow polynomially with problem size, are said to be 'tractable' and in the complexity class 'P'. Algorithms for which time and resources are found to grow exponentially with problem size are said to be 'intractable'. There are many subtleties to this classification scheme: the famous 'Travelling Salesperson Problem', for example, is in the rather mysterious complexity class 'NP'. The book by David Harel [12] contains an excellent introduction to this subject.

What has this to do with quantum information and quantum computers? In 1985 David Deutsch pointed out that since a quantum computer was not a Turing machine there was the possibility of new complexity classification of algorithms [13]. As we will see, quantum computers evolve a coherent superposition of quantum states so that each of these states could follow a distinct computational path until a final measurement is made at the output. It is therefore certainly conceptually possible that at least for some problems, quantum computers could surpass the power of classical Turing computers. The first speculation that

this might be so is probably due to Feynman in 1981 [14]. However, it was not until 1994 that interest in this subject exploded after Peter Shor's discovery of a new quantum algorithm for factorizing large numbers [15].

Mathematicians believe (although it has yet to be proved) that the number of steps required on a classical computer to factorize a number with N decimal digits grows exponentially with N. Since the computational work required grows very rapidly, the difficulty of factorizing very large numbers has been made the basis of the security of the RSA encryption method (see Ref. [13] for a good review of encryption techniques). This system is widely used to protect electronic bank accounts, for example. The significance of Shor's result was that his algorithm, running on a quantum computer, could solve the factorization problem in polynomial time. What this could mean for the RSA cryptographic system may be illustrated by the time required to factorize a 129 digit number known as RSA129 [16]. In 1994 this required 5000 MIPS-years of computer time to factorize into its 64 and 65 bit prime factors, using over 1000 workstations over a period of 8 months. A quantum computer using Shor's algorithm with a clock speed of 100 MHz could factor RSA129 in a few seconds. This explains the interest of various 'secret' government agencies around the world in the feasibility of building quantum computers!

IV. QUBITS AND QUANTUM GATES

Instead of using high and low voltages to represent the 1's and 0's of binary data, there is no reason in principle for us not to be able to any two state quantum system. Two commonly discussed possibilities are the two spin states of an electron:

$|1\rangle$ and $|0\rangle$ as $\uparrow$ and $\downarrow$

or two polarization states of a photon:

$|1\rangle$ and $|0\rangle$ as H and V

The time evolution of a quantum system is usually well approximated by the Schroedinger equation. In a coordinate space representation, for example, the Schroedinger equation is a linear partial differential equation with the property that any linear superposition of eigenfunctions is also a solution. This superposition property of quantum mechanics means that the general state may be written as a superposition of eigenstates. In the case of our 2-state quantum system the general state may be written as:

$$|\psi\rangle = \alpha|1\rangle + \beta|0\rangle$$

According to the standard interpretation of quantum mechanics, any measurement (of spin or polarization) made on this state will always yield one of the two eigenvalues with no way of knowing which one. Normalization of the state to unity guarantees:

$$|\alpha|^2 + |\beta|^2 = 1$$

and this normalization and hence the probability interpretation is maintained by any unitary operator U defined by the property:

$$U^+ U = 1$$

Information stored in a 2-state quantum system is called a quantum bit or 'qubit': besides storing classical '1' and '0' information there is also the possibility of storing information as a superposition of '1' and '0' states. We can define quantum analogues of classical reversible gates by means of unitary operators acting on the qubit basis states. For example, a quantum version of the NOT operator may be defined as follows:

$$U_{NOT}|1\rangle = |0\rangle$$

$$U_{NOT}|0\rangle = -|1\rangle$$

The phase is chosen for consistency of interpretation in terms of rotations of a spin half particle. The NOT gate corresponds to a 180 degree spin rotation. An overall phase makes no difference to the probability of measuring the particular basis state although any relative phase difference does affect measurements which depend on the interference between the two basis states.We now see two possible quantum generalisations compared to computation with classical bits. First, we can perform unitary operations

on coherent linear combinations of the two basis states:

$$U_{NOT}\frac{1}{\sqrt{2}}(|1\rangle+|0\rangle)=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

Second, we can consider operations on qubits that have no classical analogue. For example,
Deutsch introduces the 'Square Root of NOT' operator defined by:

$$(U_{SRN})^2=U_{NOT}$$

$$U_{SRN}|1\rangle=\frac{1}{\sqrt{2}}(|1\rangle+|0\rangle)$$

$$U_{SRN}|0\rangle=\frac{1}{\sqrt{2}}(-|1\rangle+|0\rangle)$$

In physical terms, such an operation merely corresponds to a 90 degree spin rotation1. Generalizing away from this specific spin interpretation, a transformation that takes a basis state and transforms it into a linear combination of the two basis states is very useful in the construction of quantum algorithms and is called a 'Hadamard' transformation.
We have considered a single electron system for storing a single qubit. By considering multiparticle systems we can construct quantum registers. Thus an n-bit register may be written as:

$$|\psi_n\rangle=|1\rangle\otimes|1\rangle\ldots\otimes|1\rangle\equiv|11\ldots1\rangle$$

If we now apply our SRN or Hadamard transformation to this state we now generate a superposition of all 2n states:

$$|\psi_n\rangle'=U_{SRN}\otimes U_{SRN}\ldots\otimes U_{SRN}|11\ldots1\rangle$$

$$=\frac{1}{2^{n/2}}(|1\rangle+|0\rangle)\otimes(|1\rangle+|0\rangle)\ldots\otimes(|1\rangle+|0\rangle)$$

$$=\frac{1}{2^{n/2}}\{|11\ldots1\rangle+|11\ldots0\rangle+\ldots+|00\ldots0\rangle\}$$

In other words, by applying a linear number of operations to the quantum register we are able to generate a register state with an exponential (2n) number of terms. The ability to create such superpositions is one of the key properties that gives quantum parallel processing its power.We now seem to have all the ingredients - logic gates and registers - to construct a quantum computer. However, neither reversible gates nor superpositions are specifically quantum mechanical.Quantum algorithms derive their remarkable power from one intrinsically quantum phenomenon that we have not so far considered. This is the property called quantum entanglement and, as we shall see, takes us to the very heart of the peculiarities of quantum mechanics.

V. PRIME FACTORIZATION

It has been known since before Euclid that every integer n is uniquely decomposable into a product of primes. Mathematicians have been interested in the question of how to factor a number into this product of primes for nearly as long. It was only in the 1970's, however, that researchers applied the paradigms of theoretical computer science to number theory, and looked at the asymptotic running times of factoring algorithms [Adleman 1994]. This has resulted in a great improvement in the efficiency of factoring algorithms. The best factoring algorithm asymptotically is currently the number field sieve [Lenstra et al. 1990, Lenstra and Lenstra 1993], which in order to factor an integer n takes asymptotic running time $\exp(c (\log n)^{1/3} (\log (\log n))^{2/3})$ for some constant c.
Since the input, n, is only log n bits in length, this algorithm is an exponential-time algorithm. Our quantum factoring algorithm takes asymptotically $O((\log n)^2(\log \log n) (\log \log \log n))$ steps on a quantum computer, along with a polynomial (in log n) amount of post-processing time on a classical computer that is used to convert the output of the quantum computer to factors of n. While this post-processing could in principle be done on a quantum computer, there is no reason not to use a classical computer if they are more efficient in practice. Instead of giving a quantum computer algorithm for factoring n directly, we give a quantum computer algorithm for finding the order of an element x in the multiplicative group (mod n); that is, the least integer r such that $x_r \equiv$

1 (mod n). It is known that using randomization, factorization can be reduced to finding the order of an element [Miller 1976]; we now briefly give this reduction. To find a factor of an odd number n, given a method for computing the order r of x, choose a random x (mod n), find its order r, and compute gcd($x_{r/2}$ − 1, n). Here, gcd(a, b) is the greatest common divisor of a and b, i.e., the largest integer that divides both a and b. The Euclidean algorithm [Knuth 1981] can be used to compute gcd(a, b) in polynomial time. Since ($x_{r/2}$−1)($x_{r/2}$+1) = x_r−1 ≡ 0 (mod n), the gcd($x_{r/2}$−1, n) fails to be a non-trivial divisor of n only if r is odd or if $x_{r/2}$ ≡ −1 (mod n). Using this criterion, it can be shown that this procedure, when applied to a random x (mod n), yields a factor of n with probability at least 1−1/2^{k-1}, where k is the number of distinct odd prime factors of n. A brief sketch of the proof of this result follows. Suppose that

$$n = \prod_{i=1}^{k} p_i^{a_i}$$

Let ri be the order of x (mod pi^{ai}). Then r is the least common multiple of all the ri. Consider the largest power of 2 dividing each ri. The algorithm only fails if all of these powers of 2 agree: if they are all 1, then r is odd and r/2 does not exist; if they are all equal and larger than 1, then $x_{r/2}$ ≡ −1 (mod n) since $x_{r/2}$ ≡ −1 (mod $pi^{\alpha i}$) for every i. By the Chinese remainder theorem [Knuth 1981, Hardy and Wright 1979, Theorem 121], choosing an x (mod n) at random is the same as choosing for each i a number x_i (mod pi^{ai}) at random, where pi^{ai} is the ith prime power factor of n. The multiplicative group (mod p^{α}) for any odd prime power p^{α} is cyclic [Knuth 1981], so for any odd prime power pi^{ai} , the probability is at most 1/2 of choosing an xi having any particular power of two as the largest divisor of its order r_i. Thus each of these powers of 2 has at most a 50% probability of agreeing with the previous ones, so all k of them agree with probability at most 1/2k−1, and there is at least a 1 − 1/2k−1 chance that the x we choose is good. This scheme will thus work as long as n is odd and not a prime power; finding factors of prime powers can be done efficiently with classical methods.

We now describe the algorithm for finding the order of x (mod n) on a quantum computer. This algorithm will use two quantum registers which hold integers represented in binary. There will also be some amount of workspace.This workspace gets reset to 0 after each subroutine of our algorithm, so we will not include it when we write down the state of our machine. Given x and n, to find the order of x, i.e., the least r such that xr ≡ 1 (mod n), we do the following. First, we find q, the power of 2 with $n^2 \le q < 2n^2$. We will not include n, x, or q when we write down the state of our machine, because we never change these values. In a quantum gate array we need not even keep these values in memory, as they can be built into the structure of the gate array. Next, we put the first register in the uniform superposition of states representing numbers a (mod q). This leaves our machine in state

$$\frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a\rangle\,|0\rangle\,.$$

This step is relatively easy, since all it entails is putting each bit in the first register into the superposition 1/√2(|0> + |1>). Next, we compute x^a (mod n) in the second register as described in §3. Since we keep a in the first register this can be done reversibly. This leaves our machine in the state

$$\frac{1}{q^{1/2}} \sum_{a=0}^{q-1} |a\rangle\,|x^a \pmod n\rangle\,.$$

We then perform our Fourier transform Aq on the first register, as described in §4, mapping |a> to

$$\frac{1}{q^{1/2}} \sum_{c=0}^{q-1} \exp(2\pi i ac/q)\,|c\rangle\,.$$

That is, we apply the unitary matrix with the (a, c) entry equal to

$$\tfrac{1}{q^{1/2}} \exp(2\pi i ac/q).$$

This leaves our machine in state

$$\frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} \exp(2\pi i ac/q)\,|c\rangle\,|x^a \pmod n\rangle\,.$$

Finally, we observe the machine. It would be sufficient to observe solely the value of |c> in the first register,

but for clarity we will assume that we observe both |c> and |xa (mod n)>. We now compute the probability that our machine ends in a particular state |c, x^k (mod n)> , where we may assume $0 \le k < r$. Summing over all possible ways to reach the state |c, x^k (mod n)> , we find that this probability is

$$\left| \frac{1}{q} \sum_{a:\, x^a \equiv x^k} \exp(2\pi i a c/q) \right|^2 . \qquad (5.5)$$

where the sum is over all a, $0 \le a < q$, such that $x^a \equiv x^k \pmod n$. Because the order of x is r, this sum is over all a satisfying $a \equiv k \pmod r$. Writing a = br + k, we find that the above probability is

$$\left| \frac{1}{q} \sum_{b=0}^{\lfloor (q-k-1)/r \rfloor} \exp(2\pi i (br+k) c/q) \right|^2 .$$

We can ignore the term of exp(2∏ikc/q), as it can be factored out of the sum and has magnitude 1. We can also replace rc with $\{rc\}_q$, where $\{rc\}_q$ is the residue which is congruent to rc (mod q) and is in the range $-q/2 < \{rc\}_q \le q/2$. This leaves us with the expression

$$\left| \frac{1}{q} \sum_{b=0}^{\lfloor (q-k-1)/r \rfloor} \exp(2\pi i b \{rc\}_q/q) \right|^2 . \qquad (5.7)$$

We will now show that if $\{rc\}_q$ is small enough, all the amplitudes in this sum will be in nearly the same direction (i.e., have close to the same phase), and thus make the sum large. Turning the sum into an integral, we obtain

$$\frac{1}{q} \int_0^{\left\lfloor \frac{q-k-1}{r} \right\rfloor} \exp(2\pi i b \{rc\}_q/q) db + O\left(\frac{\lfloor (q-k-1)/r \rfloor}{q} \left(\exp(2\pi i \{rc\}_q/q) - 1 \right) \right). \qquad (5.8)$$

If $|\{rc\}_q| \le r/2$, the error term in the above expression is easily seen to be bounded by O(1/q). We now show that if $|\{rc\}_q| \le r/2$, the above integral is large, so the probability of obtaining a state |c, x_k (mod n)> is large. Note that this condition depends only on c and is independent of k. Substituting u = rb/q in the above integral, we get

$$\frac{1}{r} \int_0^{\frac{r}{q}\left\lfloor \frac{q-k-1}{r} \right\rfloor} \exp\left(2\pi i \frac{\{rc\}_q}{r} u \right) du.$$

Since k < r, approximating the upper limit of integration by 1 results in only a O(1/q) error in the above expression. If we do this, we obtain the integral

$$\frac{1}{r} \int_0^1 \exp\left(2\pi i \frac{\{rc\}_q}{r} u \right) du. \qquad (5.10)$$

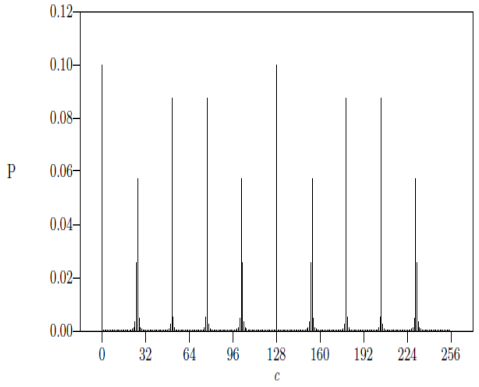

Figure 5.1: The probability P of observing values of c between 0 and 255, given $q = 256$ and $r = 10$.

Letting $\{rc\}_q/r$ vary between −1/2 and 1/2 , the absolute magnitude of the integral (5.10) is easily seen to be minimized when $\{rc\}_q/r = \pm 1/2$, in which case the absolute value of expression (5.10) is 2/(∏r). The square of this quantity is a lower bound on the probability that we see any particular state |c, x_k (mod n)> with $\{rc\}_q \le r/2$; this probability is thus asymptotically bounded below by $4/(\prod^2 r^2)$, and so is at least $1/3r^2$ for sufficiently large n. The probability of seeing a given state

$$\frac{-r}{2} \le \{rc\}_q \le \frac{r}{2}, \qquad (5.11)$$

i.e., if there is a d such that

$$\frac{-r}{2} \le rc - dq \le \frac{r}{2}. \qquad (5.12)$$

Dividing by rq and rearranging the terms gives

$$\left|\frac{c}{q} - \frac{d}{r}\right| \leq \frac{1}{2q}. \qquad (5.13)$$

We know c and q. Because $q > n^2$, there is at most one fraction d/r with r < n that satisfies the above inequality. Thus, we can obtain the fraction d/r in lowest terms by rounding c/q to the nearest fraction having a denominator smaller than n. This fraction can be found in polynomial time by using a continued fraction expansion of c/q, which finds all the best approximations of c/q by fractions [Hardy and Wright 1979, Chapter X, Knuth 1981]. The exact probabilities as given by equation (5.7) for an example case with r = 10 and q = 256 are plotted in Figure 5.1. The value r = 10 could occur when factoring 33 if x were chosen to be 5, for example. Here q is taken smaller than 33^2 so as to make the values of c in the plot distinguishable; this does not change the functional structure of P(c). Note that with high probability the observed value of c is near an integral multiple of q/r = 256/10.If we have the fraction d/r in lowest terms, and if d happens to be relatively prime to r, this will give us r. We will now count the number of states $|c, x_k \pmod n\rangle$ which enable us to compute r in this way. There are ø(r) possible values of d relatively prime to r, where ø is Euler's totient function [Knuth 1981, Hardy and Wright 1979, §5.5]. Each of these fractions d/r is close to one fraction c/q with $|c/q - d/r| \leq 1/2q$. There are also r possible values for x_k, since r is the order of x. Thus, there are r ø(r) states $|c, x_k \pmod n|$ which would enable us to obtain r. Since each of these states occurs with probability at least $1/3r^2$, we obtain r with probability at least ø (r)/3r. Using the theorem that _(r)/r > _/ log log r for some constant _ [Hardy and Wright 1979,Theorem 328], this shows that we find r at least a _/ log log r fraction of the time, so by repeating this experiment only O(log log r) times, we are assured of a high probability of success. In practice, assuming that quantum computation is more expensive than classical computation, it would be worthwhile to alter the above algorithm so as to perform less quantum computation and more post processing. First, if the observed state is |c>, it would be wise to also try numbers close to c such as c ± 1, c ± 2, . . ., since these also have a reasonable chance of being close to a fraction qd/r. Second, if c/q ≈ d/r, and d and r have a common factor, it is likely to be small. Thus, if the observed value of c/q is rounded off to d'/r′ in lowest terms, for a candidate r one should consider not only r′ but also its small multiples 2r′, 3r′, . . . , to see if these are the actual order of x. Although the first technique will only reduce the expected number of trials required to find r by a constant factor, the second technique will reduce the expected number of trials for the hardest n from O(log log n) to O(1) if the first $(\log n)^{1+\epsilon}$ multiples of r′ are

considered [Odylzko 1995]. A third technique is, if two candidate r's have been found, say r1 and r2, to test the least common multiple of r1 and r2 as a candidate r. This third technique is also able to reduce the expected number of trials to a constant [Knill 1995], and will also work in some cases where the first two techniques fail. Note that in this algorithm for determining the order of an element, we did not use many of the properties of multiplication (mod n). In fact, if we have a permutation f mapping the set {0, 1, 2, . . . , n − 1} into itself such that its kth iterate, $f^{(k)}(a)$, is computable in time polynomial in log n and log k, the same algorithm will be able to find the order of an element a under f, i.e., the minimum r such that $f^{(r)}(a) = a$.

VI. DISCRETE LOGARITHMS

For every prime p, the multiplicative group (mod p) is cyclic, that is, there are generators g such that 1, g, g^2, g^{p-2} comprise all the non-zero residues (mod p) [Hardy and Wright 1979, Theorem 111, Knuth 1981]. Suppose we are given a prime p and such a generator g. The discrete logarithm of a number x with respect to p and g is theinteger r with $0 \leq r < p-1$ such that $g^r \equiv x \pmod p$. The fastest algorithm known for finding discrete logarithms modulo arbitrary primes p is Gordon's [1993] adaptation of the number field sieve, which runs in time $\exp(O(\log p)^{1/3}(\log\log p)^{2/3}))$. We show how to find discrete logarithms on a quantum computer with two modular exponentiations and two quantum Fourier transforms.

This algorithm will use three quantum registers. We first find q a power of 2 such that q is close to p, i.e., with p < q < 2p. Next, we put the first two registers in our quantum computer in the uniform superposition of all |a> and |b> (mod p − 1), and compute $g^a x^{-b} \pmod p$ in the third register. This leaves our machine in the state

$$\frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}\left|a,b,g^a x^{-b} \pmod p\right\rangle .$$

As before, we use the Fourier transform Aq to send |a> → |c> and |b> → |d> with probability amplitude 1/q exp(2∏i(ac+bd)/q). This is, we take the state |a, b> to the state

$$\frac{1}{q}\sum_{c=0}^{q-1}\sum_{d=0}^{q-1}\exp\left(\tfrac{2\pi i}{q}(ac+bd)\right)|c,d\rangle .$$

This leaves our quantum computer in the state

$$\frac{1}{(p-1)q}\sum_{a,b=0}^{p-2}\ \sum_{c,d=0}^{q-1}\exp\left(\tfrac{2\pi i}{q}(ac+bd)\right)$$

Finally, we observe the state of the quantum computer. The probability of observing a state |c, d, y> with y ≡ g^k (mod p) is

$$\left|\frac{1}{(p-1)q}\sum_{\substack{a,b\\ a-rb\equiv k}}\exp\left(\tfrac{2\pi i}{q}(ac+bd)\right)\right|^2$$

where the sum is over all (a, b) such that a − rb ≡ k (mod p − 1). Note that we now have two moduli to deal with, p − 1 and q. While this makes keeping track of things more confusing, it does not pose serious problems. We now use the relation

$$a = br + k - (p-1)\left\lfloor\tfrac{br+k}{p-1}\right\rfloor$$

and substitute (6.5) in the expression (6.4) to obtain the amplitude on | c, d, gk (mod p) >, which is

$$\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\exp\left(\tfrac{2\pi i}{q}\left(brc+kc+bd-c(p-1)\left\lfloor\tfrac{br+k}{p-1}\right\rfloor\right)\right). \tag{6.6}$$

The absolute value of the square of this amplitude is the probability of observing the State | c, d, gk (mod p) > . We will now analyze the expression (6.6). First, a factor of exp(2∏ikc/q) can be taken out of all the terms and ignored, because it does not change the probability. Next, we split the exponent into two parts and factor out b to obtain

$$\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\exp\left(\tfrac{2\pi i}{q}bT\right)\exp\left(\tfrac{2\pi i}{q}V\right), \tag{6.7}$$

where

$$T = rc+d-\tfrac{r}{p-1}\{c(p-1)\}_q, \tag{6.8}$$

and

$$V = \left(\tfrac{br}{p-1}-\left\lfloor\tfrac{br+k}{p-1}\right\rfloor\right)\{c(p-1)\}_q. \tag{6.9}$$

Here by $\{z\}_q$ we mean the residue of z (mod q) with $-q/2 < \{z\}_q \le q/2$, as in equation (5.7). We next classify possible outputs (observed states) of the quantum computer into "good" and "bad." We will show that if we get enough "good" outputs, then we will likely be able to deduce r, and that furthermore, the chance of getting a "good" output is constant. The idea is that if

$$\left|\{T\}_q\right| = \left|rc+d-\tfrac{r}{p-1}\{c(p-1)\}_q - jq\right| \le \frac{1}{2}, \tag{6.10}$$

where j is the closest integer to T/q, then as b varies between 0 and p − 2, the phase of the first exponential term in equation (6.7) only varies over at most half of the unit circle. Further, if

$$\left|\{c(p-1)\}_q\right| \le q/12, \tag{6.11}$$

then |V | is always at most q/12, so the phase of the second exponential term in equation (6.7) never is farther than exp(∏i/6) from 1. If conditions (6.10) and (6.11) both hold, we will say that an output is "good." We will show that if both conditions hold, then the contribution to the probability from the corresponding term is significant. Furthermore, both conditions will hold with constant probability, and a reasonable sample of c's for which condition (6.10) holds will allow us to deduce r. We now give a lower bound on the probability of each good output, i.e., an output that satisfies conditions (6.10) and (6.11). We know that as b ranges from 0 to p − 2, the phase of exp(2∏ibT/q) ranges from 0 to 2∏iW where

$$W = \frac{p-2}{q}\left(rc+d-\frac{r}{p-1}\{c(p-1)\}_q - jq\right) \tag{6.12}$$

and j is as in equation (6.10). Thus, the component of the amplitude of the first exponential in the summand of (6.7) in the direction

$$\exp(\pi i W) \tag{6.13}$$

is at least cos(2∏ |W/2 −Wb/(p − 2)|). By condition (6.11), the phase can vary by at most ∏i/6 due to the second exponential exp(2∏iV/q). Applying this variation in the manner that minimizes the component in the direction (6.13), we get that the component in this direction is at least

$$\cos(2\pi\,|W/2 - Wb/(p-2)| + \tfrac{\pi}{6}).$$

Thus we get that the absolute value of the amplitude (6.7) is at least

$$\frac{1}{(p-1)q}\sum_{b=0}^{p-2}\cos\left(2\pi\,|W/2 - Wb/(p-2)| + \tfrac{\pi}{6}\right). \tag{6.15}$$

Replacing this sum with an integral, we get that the absolute value of this amplitude is at least

$$\frac{2}{q}\int_0^{1/2}\cos(\tfrac{\pi}{6} + 2\pi|W|u)du \;+\; O\left(\tfrac{W}{pq}\right). \tag{6.16}$$

From condition (6.10), |W| ≤ 1/2 , so the error term is O(1/pq). As W varies between −1/2 and 1/2 , the integral (6.16) is minimized when |W| = 1/2 . Thus, the probability of arriving at a state |c, d, y> that satisfies both conditions (6.10) and (6.11) is at least

$$\left(\frac{1}{q}\frac{2}{\pi}\int_{\pi/6}^{2\pi/3}\cos u\;du\right)^2,$$

or at least .054/q^2 > 1/(20q^2). We will now count the number of pairs (c, d) satisfying conditions (6.10) and (6.11). The number of pairs (c, d) such that (6.10) holds is exactly the number of possible c's, since for every c there is exactly one d such that (6.10) holds. Unless gcd(p−1, q) is large, the number of c's for which (6.11) holds is approximately q/6, and even if it is large, this number is at least q/12. Thus, there are at least q/12 pairs (c, d) satisfying both conditions. Multiplying by p−1, which is the number of possible y's, gives approximately pq/12 good states |c, d, y>.. Combining this calculation with the lower bound 1/(20q^2) on the probability of observing each good state gives us that the probability of observing some good state is at least p/(240q), or at least 1/480 (since q < 2p). Note that each good c has a probability of at least (p − 1)/(20q^2) ≥ 1/(40q) of being observed, since there p − 1 values of y and one value of d with which c can make a good state |c, d, y>. We now want to recover r from a pair c, d such that

$$-\frac{1}{2q} \le \frac{d}{q} + r\left(\frac{c(p-1) - \{c(p-1)\}_q}{(p-1)q}\right) \le \frac{1}{2q} \pmod 1, \tag{6.18}$$

where this equation was obtained from condition (6.10) by dividing by q. The first thing to notice is that the multiplier on r is a fraction with denominator p − 1, since q evenly divides c(p−1)−{c(p − 1)}q. Thus, we need only round d/q off to the nearest multiple of 1/(p − 1) and divide (mod p − 1) by the integer

$$c' = \frac{c(p-1) - \{c(p-1)\}_q}{q} \tag{6.19}$$

to find a candidate r. To show that the quantum calculation need only be repeated a polynomial number of times to find the correct r requires only a few more details. The problem is that we cannot divide by a number c′ which is not relatively prime to p − 1. For the discrete log algorithm, we do not know that all possible values of c′ are generated with reasonable likelihood; we only know this about one-twelfth of them. This additional difficulty makes the next step harder than the corresponding step in the algorithm for factoring. If we knew the remainder of r modulo all prime powers dividing
p−1, we could use the Chinese remainder theorem to recover r in polynomial time. We will only be able to prove that we can find this remainder for primes larger than 18, but with a little extra work we will still be able to recover r.

Recall that each good (c, d) pair is generated with probability at least 1/(20q^2), and that at least a twelfth of the possible c's are in a good (c, d) pair. From equation (6.19), it follows that these c's are mapped from c/q to c′/(p − 1) by rounding to the nearest integral multiple of 1/(p − 1). Further, the good c's are exactly those in which c/q is close to c′/(p − 1). Thus, each good c corresponds with exactly one c′. We

would like to show that for any prime power $p^{\alpha i}$ dividing $p - 1$, a random good c' is unlikely to contain p_i. If we are willing to accept a large constant for our algorithm, we can just ignore the prime powers under 18; if we know r modulo all prime powers over 18, we can try all possible residues for primes under 18 with only a (large) constant factor increase in running time. Because at least one twelfth of the c's were in a good (c, d) pair, at least one twelfth of the c''s are good. Thus, for a prime power $p_i^{\alpha i}$, a random good c' is divisible by $p_i^{\alpha i}$ with probability at most $12/p_i^{\alpha i}$. If we have t good c''s, the probability of having a prime power over 18 that divides all of them is therefore at most

$$\sum_{18 < p_i^{\alpha_i} \mid (p-1)} \left(\frac{12}{p_i^{\alpha_i}} \right)^t , \qquad (6.20)$$

where a|b means that a evenly divides b, so the sum is over all prime powers greater than 18 that divide $p - 1$. This sum (over all integers > 18) converges for t = 2, and goes down by at least a factor of 2/3 for each further increase of t by 1; thus for some constant t it is less than 1/2. Recall that each good c' is obtained with probability at least 1/(40q) from any experiment. Since there are q/12 good c''s, after 480t experiments, we are likely to obtain a sample of t good c''s chosen equally likely from all good c''s. Thus, we will be able to find a set of c''s such that all prime powers $p_i^{\alpha i} > 20$ dividing $p-1$ are relatively prime to at least one of these c''s. To obtain a polynomial time algorithm, all one need

do is try all possible sets of c''s of size t; in practice, one would use an algorithm to find sets of c''s with large common factors. This set gives the residue of r for all primes larger than 18. For each prime p_i less than 18, we have at most 18 possibilities for the residue modulo $p_i^{\alpha i}$, where αi is the exponent on prime p_i in the prime factorization of $p-1$. We can thus try all possibilities for residues modulo powers of primes less than 18: for each possibility we can calculate the corresponding r using the Chinese remainder theorem and then check to see whether it is the desired discrete logarithm. If one were to actually program this algorithm there are many ways in which the efficiency could be increased over the efficiency shown in this paper. For example, the estimate for the number of good c''s is likely too low, especially since weaker conditions than (6.10) and (6.11) should suffice. This means that the number of times the experiment need be run could be reduced. It also seems improbable that the distribution of bad values of c' would have any relationship to primes under 18; if this is true, we need not treat small prime powers separately.

This algorithm does not use very many properties of Zp, so we can use the same algorithm to find discrete logarithms over other fields such as Zp^{α}, as long as the field has a cyclic multiplicative group. All we need is that we know the order of the generator, and that we can multiply and take inverses of elements in polynomial time. The order of the generator could in fact be computed using the quantum order-finding algorithm given in §5 of this paper. Boneh and Lipton [1995] have generalized the algorithm so as to be able to find discrete logarithms when the group is abelian but not cyclic.

REFERENCES

[1] P. Benioff, "Quantum mechanical Hamiltonian models of Turing machines," J. Stat. Phys. Vol. 29, pp. 515- 546 (1982).

[2] P. Benioff, "Quantum mechanical Hamiltonian models of Turing machines that dissipate no energy," Phys. Rev. Lett. Vol. 48, pp. 1581-1585 (1982).

[3] C. H. Bennett, "Logical reversibility of computation," IBM J. Res. Develop. Vol. 17, pp. 525-532 (1973).