20 QUESTIONS, Quantum Computers, and Cryptography

cryptography
quantum computers
cryptography
quantum computers

A mathematical metaphor for the power of quantum algorithms

Mark Ettinger

How can quantum computers do the amazing things that they are able to do, such as factoring large numbers and finding discrete logarithms? What makes them so different from classical computers? These questions are often asked, and they have proved to be surprisingly difficult to answer—at least to the satisfaction of everyone! In this short article, I'll try to address these questions by comparing the operation of a quantum computer with playing the game of 20 questions. But first, let's consider an unusual perspective on computers in general.

What Is a Computer?

Well, a computer is really just some physical machine that you prepare in a certain way, manipulate in certain ways, and then watch to observe the results it displays. That is how physicists might describe the entire physical process that mathematicians call a computation. This view seems a bit strange at first because we have become accustomed to the more abstract view of the computer scientist, who sees a computation as a certain type of process that acts on an input in order to produce an output. But our physical description is not really so different. It just emphasizes the physical nature of the computation, something that falls by the wayside in the abstracted view. The initial preparation is what a computer scientist calls an input, the actual computation is the physical manipulation, and the observation at the end results in getting the output. So, whereas a computation can be viewed abstractly as a process, its physical nature can also be emphasized. This view will help us make the transition to understanding what a quantum machine is doing in a special way. Unlike classical computers, which are physical devices manipulated according to the laws of classical physics, quantum computers are physical devices manipulated according to the laws of quantum physics.

Quantum Computers and the 20 Questions Game

Having understood that a computation is ultimately a physical process, let's go on to see how using a quantum machine is much like playing the game of 20 questions. Twenty questions is played as follows. I think of a number between 1 and 2^{20}. You try to guess my secret number by asking questions such as, "Is your secret number less than 2378?" If you ask your questions well, you can guess my secret number in, at the most, 20 questions. Why? Well, with each question, you can eliminate half of the remaining candidates. Computer scientists call this process binary search, and it allows you to find a secret number less than 2^n in $\log 2^n = n$ questions at the most. The key idea is that, by cutting the number of possibilities in half with each question, you are left with one possibility after only n questions. This principle generalizes. For example, if you are searching for a secret item among N possibilities and with each question you are able to eliminate a fraction $1 - 1/c$ of the possibilities, then you can find the secret in $\log_c N$ questions. In general, you might not be looking for a number. You might be looking for a secret element x in a set S called a search space. The key to quick success is still to be able to eliminate a constant fraction of the remaining candidates. Now, let's consider a slightly different version of this game, which we call "random 20 questions."

In playing random 20 questions, you don't get to choose your question. Instead, you randomly select a subset Q (used for the word "question") consisting of half of the N elements in the search space, and you ask, "Is the secret element in Q?" After I give you the honest answer, you choose a new random subset Q and ask again. Surprisingly, again after only about $\log N$ questions, you will almost surely have narrowed the possibilities down to the one correct answer. We say "almost surely" because there is a tiny, tiny chance that you will get unlucky and never be able to eliminate one of the elements that is not the secret element. This tiny chance is the result of each question having been selected randomly rather than deterministically, which is the case when playing the original 20 questions game. After $2 \log N$ questions, for example, that possibility is incredibly small. So, even by asking random questions, you can discover the secret element quickly. The reason is that, as in the original 20 questions game, you are able to eliminate each incorrect element as a possibility. Although in the random 20 questions game this process of elimination is only very highly probable, it is so close to being certain that, for all practical purposes, we won't worry about it. Now, let's talk about playing quantum 20 questions.

In this game, I choose a secret quantum state ρ_1 from a search space of quantum states $S = \{\rho_1, \rho_2 \ldots, \rho_N\}$, and I supply a copy of the secret state whenever you request one. Your task is to discover my secret quantum state by asking quantum questions, that is, by doing measurements on each requested quantum state and thus getting information about the state. Now, let's back up a bit and clarify these terms. What is a quantum state? A pure state ψ is simply a vector in a Hilbert space. A mixed state, or more simply a state, is a convex combination of pure states ψ_i, that is, a classical probabilistic mixture of pure states:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \quad , \quad \text{where} \sum_i p_i = 1 \quad . \tag{1}$$

Quantum Questions

What is a quantum question? A quantum question is typically called an observable. We'll think of a quantum question as simply an orthonormal basis. The answer to a quantum question will be one of the basis vectors. So, suppose the secret quantum state is a pure state $|\varphi\rangle$ and the quantum question is $\{|\phi_1\rangle, |\phi_2\rangle,\ldots, |\phi_M\rangle\}$, a basis of the M-dimensional Hilbert space. According to the basic rules of quantum mechanics, we get the answer $|\phi\rangle$ with probability $|\langle\phi|\varphi\rangle|^2$. If we have a mixed state instead of a pure state, the probability formula is extended by convexity, as usual. How many quantum questions does it take to guess the secret quantum state? That depends on lots of things. It depends on what quantum questions you are allowed to ask me. And it also depends on how different the states in S are from each other. In this context, the word "different" means how distinguishable the states are from each other. For example, two orthogonal pure states are as different as two states can be. Two very nearly parallel pure states are almost indistinguishable in that it takes many experiments and questions to tell them apart based on the outcome statistics. The standard measure of similarity between two pure states is simply their overlap $\langle\phi|\varphi\rangle$. There are measures for the similarity or overlap of mixed states as well, but we won't need the formula. We just need to know that to tell apart two similar states requires many experiments whereas to tell apart two very different states requires few experiments.

Playing search games is much like trying to break codes. If you try to break a code, you want to look for a cryptographic key. To solve classical cyptographic problems with quantum computers, you are looking for a secret key from among a known set of possible secret keys.

So, going back to quantum 20 questions, let's assume you can ask any quantum question you want; that is, you can choose any orthonormal basis as the observable. If all the states in S are sufficiently different from each other, you can find my secret state after only a few questions. Usually, when we use the word "few" in this context, we mean $\log |S|$ or $\log^2|S|$ or something like that. (A computer scientist would say that few means a polynomial function of the logarithm of the size of the search space.) The key to a fast search is that all the states must be quite different from each other.

It turns out that playing search games is much like trying to break codes. If you try to break a code, you want to look for a cryptographic key. The key is what allows you to decipher the code and read the message. One popular code is the RSA. Named after its inventors—Ronald Rivest, Adi Shamir, and Leonard Adleman—the RSA uses as its key the secret factors of a large number N. Now, suppose you are trying to break a code by finding a secret key k from among a very large set of possible keys $K = \{k_1, k_2\ldots, k_M\}$. Further suppose that, by some process and without knowing the key, you can prepare a quantum state ρ corresponding to the key k. So, you now have a state ρ, which you know comes from the search space $S = \{\rho_1,\ldots, \rho_M\}$, which is the set of states corresponding to all the possible secret keys, but you don't know exactly which of the states you have. If the states of S are all sufficiently different, then you can ask quantum questions to determine the secret state efficiently. And if you can find the secret state, then you can easily figure out the original secret key corresponding to that secret state!

Indeed, this is precisely how quantum computers would solve various classical cryptographic problems, such as factoring and finding discrete logarithms. A factoring problem is one in which you are given a very large number N (say, one with 2000 digits), which is the product of two primes $N = pq$, and your task is to find p and q. For the discrete logarithm problem, you are given a large prime number p (say, once again, one with 2000 digits) and two numbers a and b less than p. Your task is to find n such that $a^n = b \pmod p$. In both cases, you are looking for a secret key k from among a known set of possible secret keys. Also, in both cases there is a process by which you can prepare a quantum state from which k can be deduced. Significantly, this preparation process does not require knowing k.

This last point is important because, if you had to know the key first, then the code-breaking machine would not be very useful. We will later illustrate this process in an

example (see the section "Simon's Problem"). Finally, this process has the special and important quality that, for two different keys, k_1 and k_2, the resulting quantum states, ρ_1 and ρ_2, are quite different, or clearly distinguishable from one another, as discussed before. We can therefore ask quantum questions, which allow us to distinguish among states and identify secret keys. This ability to distinguish among the states is usually accomplished by eliminating the possibility of a constant fraction, say 1/2, of the remaining states. As we saw in the game of 20 questions, eliminating a constant fraction after each question allows us to narrow the possible states down to the one true state in only log N questions. However, since the quantum formula gives probabilities for certain outcomes, we eliminate the false states with high probability (not with certainty), as in the game of random 20 questions.

Identifying Secret Quantum States

Let us fill in some of the technical details of our sketch. First, can we really ask any quantum question? No, we can't, but fortunately we are able to ask the questions that let us solve factoring and discrete logarithm problems. Recalling our observation that a computation is actually a physical process, we must be sure to carry out efficiently the physical process corresponding to the quantum question we wish to ask. We accomplish this task by breaking down the observable into elementary quantum "gates." Elementary quantum gates are analogous to the basic logical gates **and**, **or**, and **not**, which are the building blocks of circuits in classical computers (for more details, see Shor 1997). In the case of factoring and discrete logarithm problems, it turns out that we have to ask only one quantum question over and over again in order to obtain enough information for identifying the secret quantum state. Called the quantum Fourier transform, this quantum question allows us to distinguish among the states that arise in the two search spaces for the factoring and discrete logarithm problems. These states are called hidden subgroup states because, in those problems, the key we are looking for corresponds to an unknown subgroup H of a finite abelian group G. The search space corresponds to the set $\{\rho_{H_1}, \rho_{H_2}, \ldots \rho_{H_i}\}$, where H_1 to H_i is a range over all the possible subgroups of G, and ρ_H is the mixed state that corresponds to a uniform mixture of the pure coset states

In factoring and discrete logarithm problems, we must ask only one quantum question over and over again in order to obtain information for identifying the secret quantum state.

$$\left|c + \mathrm{H}\right\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} \left|c + \mathrm{h}\right\rangle \ . \qquad (2)$$

It can be shown that for H_1 and H_2, two different subgroups, the corresponding states ρ_{H_1} and ρ_{H_2} are sufficiently different. Mathematically speaking, the overlap of ρ_{H_1} and ρ_{H_2} is less than 1/2 (Ettinger et al. 1999). For a discussion of the hidden subgroup problem and the reasons why the quantum Fourier transform is the right quantum question, see Ettinger and Peter Hoyer (1999).

Simon's Problem

To illustrate everything we have discussed, let's consider a concrete example known as Simon's problem. Simon's problem and the quantum algorithm to solve it contain the essence of what is going on in the factoring and discrete logarithm problems; the latter set of problems, however, also contains a number of technical twists that obscure the main ideas. The set of all bit strings of length n, denoted Z_2^n, is a commutative group if

we add bit strings using "binary add without carry." This group will be our search space. I will secretly choose an element s of this group and provide you with a function in the form of a "black box," f_s on Z_2^n, with the following special property: I guarantee that $f_s(x) = f_s(y)$ if and only if $x - y = s$. So, the function f_s encodes the secret bit string s. Because f depends entirely on s, the latter becomes a subscript on f. If you compute the function on the elements of the group $f_s(a), f_s(b), f_s(c)\ldots$, eventually you'll get a collision, which means that you'll find $f_s(g) = f_s(t)$ and then you'll know that the secret bit string is $s = g - t$. But notice that the search space, or the group, has 2^n elements, which is a very large number. In the worst case, it could take you $2^{n-1} + 1$ calculations to get a collision, and on average it will take about $2^{n/2}$ because of the so-called birthday paradox.[1] That is still a lot of time! But the quantum algorithm can solve this problem much more quickly—in about n tries only.

Our quantum algorithm for solving Simon's problem allows distinguishing among different states and thus discovering the underlying secret bit string.

Here is how Simon's problem works: You start with a quantum computer whose qubits are conceptually divided into two registers. Then you prepare the pure state $|\psi\rangle = 1/2^{n/2} \sum_b |b\rangle$, where $b \in Z_2^n$. Thus, in the first register, there is a superposition of all the bs. Now, because you have the black box function f_s, you can compute f_s(b) in the second register to obtain the pure state $|\psi_s\rangle = 1/2^{n/2} \sum_b |b\rangle|f_s(b)\rangle$, where again $b \in Z_2^n$. Notice that this procedure for preparing the state ψ_s is easily accomplished without any knowledge of the secret bit string s. Of course, for different secret bit strings, we obtain different states. In fact, this is the key point. Our quantum algorithm is really just a method used to distinguish among these different states and thus discover the underlying secret bit string.

We now observe, or perform a measurement, on the second register. Because of the way quantum mechanics works, this observation collapses $|\psi_s\rangle$, producing a specific value in the second register, say c, and the first register is left in a superposition of bit strings that map to c under f_s. Because f_s has the special property described earlier, the bit strings that map to c will differ by the secret bit string s. Therefore, the state of the computer is

$$\left|\psi\right\rangle_{a,s} = \frac{1}{\sqrt{2}}\left|a\right\rangle\left|c\right\rangle + \frac{1}{\sqrt{2}}\left|a+s\right\rangle\left|c\right\rangle \ , \tag{3}$$

where a and $a + s$ are elements of Z_2^n such that $f_s(a) = c$ and $f_s(a + s) = c$. The only use of the second register is to produce this special superposition in the first register. We will no longer use the second register or its contents, so we drop it from our notation and write

$$\left|\psi\right\rangle_{a,s} = \frac{1}{\sqrt{2}}\left|a\right\rangle + \frac{1}{\sqrt{2}}\left|a+s\right\rangle \ . \tag{4}$$

When c is chosen, the resulting mixed state can be written as

$$\rho_s = \frac{1}{2^{n-1}} \sum_a \left|\psi\right\rangle_{a,s\ \ a,s}\left\langle\psi\right| \ . \tag{5}$$

Recall that we don't know the secret bit string s, and therefore we don't know that the state we just prepared is ρ_s. All we know is that we have prepared a state that is in the search space of quantum states $\{\rho_s\}_{s \in Z_2^n}$. Each of these possible quantum states corresponds to a possible secret bit string. Our task is to identify the secret quantum state

[1] The birthday paradox derives its name from the surprising result that you only need 23 people (a slightly larger number than $365^{1/2}$) to have a 50 percent chance that at least two of them have the same birthday.

and thus the secret bit string. We now define the Fourier observable. For each bit string b in Z_2^n, define

$$|\chi_b\rangle = \frac{1}{\sqrt{2^n}} \sum_{d \in Z_2^n} (-1)^{b \cdot d} |d\rangle \ , \quad \text{where } b \cdot d = \sum_i b_i d_i \pmod 2 \ . \tag{6}$$

The orthonormal basis is $\{|\chi_b\rangle\}$, where $b \in Z_2^n$ is called the Fourier basis or the Fourier observable. Mathematicians might recognize this basis as being composed of the characters of the group Z_2^n. A character χ of a finite abelian group is a homomorphism from the group to the circle in the complex plane. Formally, the Hilbert space in which we are working is $C[G]$, the group algebra, which is the complex vector space with the canonical basis, or the point mass basis, indexed by the elements of the group. A character can be viewed as a vector in $C[G]$ via the following identification:

$$|\chi\rangle = \frac{1}{\sqrt{|G|}} \sum_{g \in G} \chi(g) |g\rangle \ . \tag{7}$$

It is a fundamental fact (Tolimieri et al. 1997) that the set of characters viewed as vectors in this way is an orthonormal basis for $C[G]$. Indeed, a Fourier transform is nothing other than a change of basis from the point mass basis, $\{|g\rangle\}_{g \in G}$, to the basis of characters, $\{|\chi\rangle\}_{\chi}$.

It is easy to show (Jozsa 1998) that, if we now observe the contents of the remaining register in the Fourier basis, we observe $|\chi_b\rangle$ with nonzero probability if and only if $s \bullet b = 0 \pmod 2$. This is the important relationship between the secret bit string s and the only possible outcomes of the experiment. Therefore, if the actual outcome of the observation is $|\chi_b\rangle$, then we have eliminated half of the possible secret states. We have therefore eliminated all states ρ_d such that $d \bullet b = 1 \pmod 2$. By repeating the state preparation procedure followed by a measurement in the Fourier basis approximately n times, we eliminate all possible states except the true secret state ρ_s. ■

Mark Ettinger graduated from the Massachusetts Institute of Technology in 1987 with Bachelor's degrees in physics and mathematics. In 1996, Mark received his Ph.D. in mathematics from the University of Wisconsin at Madison.

He first came to Los Alamos National Laboratory in 1993, as a graduate student, then entered the postdoctoral program after graduation in 1996, and became a staff member in 1999. Mark worked on the group-theoretical approach to quantum algorithms for four years and is now primarily interested in (classical) algorithmic problems in postgenomic computational biology.

Further Reading

Ettinger, M., and P. Hoyer. 1999. Quantum State Detection via Elimination [Online]: http://eprints.lanl.gov_(quant-ph/9905099).

Ettinger, M., P. Hoyer, and E. Knill. 1999. Hidden Subgroup States are Almost Orthgonal.[Online]: http://eprints.lanl.gov (quant-ph/9901034).

Jozsa, R. 1998. Quantum Algorithms and the Fourier Transform. *Proc. R. Soc. London, Ser. A* **454**: 323.

Shor, P. W. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on Logarithms on a Quantum Computer. *SIAM J. Computing* **26**: 1484.

Tolimieri, R., M. An, and C. Lu. 1997. *Mathematics of Multidimensional Fourier Transform Algorithms*. New York: Springer.