Quantum information and quantum computation

John Preskill
Caltech Alumni Day Seminar, 18 May 1996

(This is a lightly edited transcript of the talk.)

Abstract. Since the physical world is fundamentally quantum mechanical, the foundations of information theory and computer science must be sought in quantum physics. In fact, there are deep differences between quantum information and the traditional "classical" concept of information, and I will discuss some of these differences. Most profoundly, a "quantum computer" seems to be capable of efficiently performing certain types of computations that are intractable on an ordinary classical computer. I will explain how quantum computers work, and will discuss some of the obstacles that must be surmounted before useful devices can be built and operated.

Quantum computation?? This will be a talk about the theory of information and the theory of computation. And perhaps the strangest thing about this topic is that a physicist is giving the talk. What do physicists know about information or computation? What am I doing here? Could it be that Milt has made a terrible mistake?

Perhaps that is possible, but please reserve judgment for a few minutes. Perhaps a physicist might have something useful to say about information or computation. After all, what is information? Well, whatever it is, it is something that can be encoded and stored in the state of a physical system. And since we live in a world that is fundamentally quantum-mechanical, that means that information is something that can be encoded in a *quantum* state.

And what is computation? Well, whatever it is, it is something that can be carried out on an actual physically realizable device. And in our quantum-mechanical world, that means a device that evolves according to the laws of *quantum* dynamics. So perhaps physicists, who know something about quantum theory, can say something enlightening about information or computation.

Now the preceding statement may seem a bit disingenuous—by similar logic I could have argued that it would be useful to have a quantum theory of the Los Angeles Dodgers: the Dodgers, after all, are fundamentally quantum mechanical. And in fact, the digital computers that are a ubiquitous part of our daily lives are (like the Dodgers) highly classical devices—you don't need to know anything about quantum mechanics to understand how they work. Still, there are incentives for thinking about how quantum theory might impact computation. For one thing, as the trend toward miniaturization of electronic circuitry proceeds to smaller and smaller scales, we will eventually enter the regime in which quantum theory *is* relevant to how computing devices function. And in any case, it might be useful to consider how intrinsically quantum-mechanical phenomena could be invoked as part of a computational strategy.

Indeed, I will argue today that there are three reasons to be excited about the idea of a *quantum computer*, a computer that makes essential use of quantum-mechanical principles. The first reason is that we now know, or think we know, that quantum computers can in principle solve *hard* problems, problems so difficult that they are beyond the reach of foreseeable digital computers. Second, the notion of a quantum computer is not merely a mathematical abstraction. Real hardware can be constructed, and an effort is underway in several laboratories to build rudimentary quantum computing devices. Finally, it seems at first that a very serious obstacle must be overcome if quantum computers are ever to be practical—quantum computers are much more prone to making *errors* than conventional computers, and a method must be found to control these errors for a quantum computer to be reliable. The third reason to be excited is that we now understand how in principle the errors made by a quantum computer can be controlled and corrected to prevent a computation from crashing.

I intend to first explain how quantum information is different than classical information, and then how a quantum computer is different than a classical computer. I'll briefly discuss *complexity*, the classification of the hardness of problems, to explain what quantum computers might be good for. I'll describe what the hardware of a quantum computer might be like, and I'll briefly describe the software. Finally, I'll return to the crucial issue of errors and how to deal with them.

Quantum information. So what is quantum information? It is familiar that classical information can be divided into indivisible units—the unit of classical information is the *bit*, it takes either one of two possible values, 0 or 1. But in quantum theory, we can envision an interesting generalization of the bit, what I will call a *quantum bit*, or a *qubit* for short. An example of a qubit is the spin of an elementary particle like an electron. You can think of the electron spin as a little vector that can point in any direction in space. But it is a peculiar vector. You can measure the spin along any axis you choose, say the vertical (z)-axis, and if you do the outcome of the measurement will be one of only two possible values—the spin either points up or down along that axis. When you measure the spin you acquire one bit of information.

But this qubit is different that a classical bit. For one thing, the outcome of the measurement is not deterministic. Suppose the electron spin is pointing along the x-axis, either to the left or to the right. And suppose I again measure the spin along the z-axis. Then I can't predict with certainty what the outcome of the measurement will be. I can only say that the probability is 1/2 that the result will be spin up and the probability is 1/2 that the result will be spin down. That's one difference between a classical bit and a qubit, but it's not the whole story.

Suppose I consider this state of the electron spin that has a probability of 1/2 of pointing to the right and a probability of 1/2 of pointing to the left. Now what happens if I measure the spin along the $z - axis$? Well, it's obvious isn't it? Whether it points right or left, the probability of spin up along the z-axis is 1/2; so this state which either points right or points left will surely also have the probability 1/2 of pointing up along the z-axis. It's obvious, but it's wrong! In this state, the spin actually points up along the z-axis with probability 1. So probabilities for qubits don't add the way we're accustomed to; this is a phenomenon that we call *quantum interference*, and it is an important way that classical information is different than quantum information.

Qubits, like classical bits, are good for storing information. Suppose I want to encode the *Encyclopedia Britannica* in qubits. It's simple. First I translate the encyclopedia into ASCII code, a string of 0's and 1's. The I line up a whole lot of spins, with each spin pointing either up (for a 0) or down (for a 1) along the z-axis. I can come back the next day and measure all the spins to read the encyclopedia.

But there is a funny thing about quantum information: to read it you have to know what you're doing. Suppose I have a friend—he doesn't know what he's doing—and he comes back the next day and measure all the spins along the x-axis. Then he just obtains a string of *random* bits, and he doesn't manage to read even a single letter of the encyclopedia. Worse than that, if I come back the day after to look up something in the encyclopedia, I can't anymore because my friend messed it up!

Hidden information. But there is a deeper difference between classical and quantum information that we can appreciate only by considering states of two or more qubits. So now imagine that we have two electrons; one is here at Caltech in Pasadena, and the other is very far away, say in the Andromeda galaxy. We can prepare a state of these two electrons with some strange properties. For this state, if I measure the spin of my electron in Pasadena along *any* axis the result will be spin up with probability 1/2 and spin down with probability 1/2. And the same is true for my friend in Andromeda when he measures his spin along any axis. So neither one of us acquires any information by making the measurement; the result is just a random bit. But we have two qubits so we should be able to store two bits of information. Where *is* the information?

The answer is that all of the information is encoded in the *correlations* between my spin and the spin in Andromeda. This state that we prepared has the property that if I measure my spin along any axis, and my friend measures his along the *same* axis, then if I obtain the result spin up my friend will *with certainty* obtain the result spin down, and if I obtain the result spin down he will obtain with certainty the result spin up—our results are perfectly *anticorrelated*. It turns out that it takes two bits of information to describe this correlation. So there *are* two bits of information, but we can't access any of the information by measuring one spin at a time. Instead, the information is spread out in a highly nonlocal way, and equally shared by the spin in Pasadena and the spin in Andromeda. This phenomenon—information that cannot be accessed through localized measurements but is instead encoded as nonlocal correlations—is what we call *quantum entanglement*, and it is a very important feature of quantum information.

It is important to appreciate how weird these nonlocal correlations are. Suppose that both my friend in Andromeda and I agree that we will each measure our spin along one of three possible axes, all rotated by 120° relative to one another. I measure my spin along axis 1, say. In performing the measurement I disturb the spin, so I can't make another measurement now to find out what would have happened if I had measured the spin along axis 2 instead. But my friend in Andromeda can measure *his* spin along axis 2, and send me a telegram reporting the result. In the special state that we have prepared, we know that his spin and mine are perfectly anticorrelated. So when I receive the telegram, I *do* find out what would have happened if I had measured the spin along axis 2 instead.

For purposes of visualization, we can think of the the value of my spin along axes 1, 2, and 3 as three "quantum coins" lying on the table. Initially, all three coins are covered, so I don't know whether the coin is heads or tails (I don't know whether the spin is up or down along this axis). But I can uncover two of the coins (I measure my spin along one axis, and my friend in Andromeda measures his coin along a different axis). When I uncover two coins, the third one always disappears before I find out if it is heads or tails (I will never know what would have happened if I had measured my spin along the third axis). Now, quantum mechanics tells us this about quantum coins: when we uncover any two coins, the probability that they come up the *same* (both heads or both tails) is 1/4. We also know this about real (classical) coins: if we uncover all 3 coins, that it is certain that at least two of the 3 coins are the same; either at least 2 are heads or at least 2 are tails. With quantum coins, we can never uncover all 3. But surely the probability that at least two coins are the same is less than the probability that two coins of a pair are the same, summed over the 3 ways of choosing the pair. But $3 \cdot \frac{1}{4} = \frac{3}{4} < 1$??!! The probability that at least two of three quantum coins are the same is less than 1. Weird!

The moral is that it's wrong, it's mathematically inconsistent, to assign simultaneous probabilities to the outcome of the measurement of my spin along two different axes, and that despite the fact that there can be a perfect correlation between my spin and a spin in Andromeda. This result, what physicist's call *Bell's theorem* has caused more consternation among thoughtful people than anything about quantum physics. But it is a fact of nature, verified by experiment. It is the most essential way that quantum information differs from classical information, and we all just need to get used to it.

Classical vs. quantum computers. You know what a classical computer is—you have one on your desktop. And you know what it does. It has a register, a long string of bits, and a CPU (a processor) that can perform simple operations on pairs of bits. There is a program that instructs the CPU what to do. By performing many simple operations in sequence, the computer can build up a very complex computation. When it is done executing its program, it halts and prints out the answer, a long string of 0's and 1's, on a piece of paper.

You may sometimes be disappointed by the performance of the machine on your desktop, but it is really quite a remarkable device. In principle, it is capable of performing any conceivable computation. In practice there are computations that you can't do—you either run out of time or you run out of memory. But if you provide an unlimited amount of memory, and you are willing to wait as long as it takes, then anything that deserves to be called a computation can be done by your little Macintosh. We say, therefore, that it is a "universal computer."

So what is a quantum computer? A classical computer processes classical information, and a quantum computer processes quantum information. We provide some qubits, say a row of electron spins that are all initially pointing up along the z-axis. The quantum computer can perform simple unitary operations on pairs of qubits. By performing many such operations in succession, it can build up an arbitrarily complex "quantum computation," a big unitary transformation acting on the many qubits in the device. If we choose our basic "quantum gates" appropriately, then in a finite number of steps we can come as close as we please to any desired unitary transformation—we then say that our machine is a "universal" quantum computer. Finally, we measure all of the spins along the z-axis to obtain the result of the computation. Thus, in the end the answer is a string of 0's and 1's that we can write down on a piece of paper, and submit for publication to a scientific journal.

Note that a quantum computer is probabilistic. If you run the exact same program twice, you might not get the same answer; there is a probability distribution of possible outcomes for the final measurement.

What is special about a quantum computer? Are there things that it can do that an ordinary digital computer cannot do? "Not at all," you might argue. Sure, a quantum computer operates according to different physical principles than a classical computer, but a classical computer can *simulate* the operation of a quantum computer to arbitrarily good accuracy. If there are N qubits in the quantum computer, the state of the qubits can be described as a vector in a vector space of dimension 2^N. The processing of the quantum information corresponds to rotating that vector in a particular way, and the final measurement can be modeled as a projection of the vector onto some set of mutually perpendicular axes. Well, classical computers know about vectors, they can rotate vectors, they can project vectors onto mutually perpendicular axes. We can evidently run a simulation of our quantum computer on a classical computer that will give precisely the same results as the quantum computation. So it is clear that there is nothing a quantum computer can calculate that cannot be calculated on a classical computer instead.

But ... how long will the simulation take? Suppose there are a modest number of qubits in our device, say $N=100$. Then the quantum information is a vector in a space of (complex) dimension $2^{100} \sim 10^{30}$. To encode that information faithfully, the classical computer would need to store 10^{30} complex numbers! No existing or forseeable digital computer will be able to do that. And performing a general rotation of a vector in a space of dimension 10^{30} is also far beyond the computational capacity of any foreseeable classical computer. While it's true that a classical computer can simulate a quantum computer, the simulation becomes extremely *inefficient* as the number of qubits N increases. This observation, that the simulation of a quantum computer by a classical computer is very inefficient, was first emphasized by Richard Feynman in 1982, and led him to speculate that quantum computers may be able to perform certain tasks that are beyond the reach of conceivable classical computers.

Quantum complexity. To sharpen this claim, let's briefly discuss the the theory of *complexity*, the classification of the hardness of problems. Computer scientists sometimes speak of *intractable* problems that have this property: once the solution to the problem is found, it is easy to verify that the solution is correct. But it is difficult to find the solution in the first place. Note that for a problem of this kind, it suffices to have a probabilistic algorithm (like an algorithm running on a quantum computer) that has a reasonable chance of finding the solution. The algorithm does not generate the correct answer every time, but we can run the algorithm many times if necessary, until we get the right answer.

An example of an intractable problem is factoring—finding the prime factors of a number. Suppose that p and q are two prime numbers (neither can be expressed as a product of smaller numbers), each 100 digits long. And n is the product $p \cdot q$. Of course, it is very easy to check that n has the prime factorization $n = p \cdot q$—we just multiply the two 100-digit number together, a snap for your Macintosh. But suppose I gave you the 200-digit number n, and didn't tell you its prime factors. Then it is a very hard problem to find the factors of n. The current practical limit for factoring is about 130 digits—numbers that long can be factored in a time of order a year by the fastest existing supercomputers. But the difficulty of the problem increases very sharply as the number of digits increases (the time it takes to solve the problem grows faster than any power of the number of digits.) To factor a 400 digit number, that same supercomputer would need about the age of the universe. It seems clear that, even with enormous advances in computer power, no digital computer available in the reasonably near future is likely to be able to factor (say) a 500-digit number.

By the way, aside from being an interesting example of an intractable problem, factoring is also of practical interest. The difficulty of factoring large numbers is the key to widely-used schemes for public-key cryptography that you may have used yourself, if you have ever made a purchase or done your banking over the Internet.

Now let's come back to Feynman's idea, that quantum computers may be able to perform computations that are beyond the reach of classical computers. This suggestion was put in a more concrete form a few years after Feynman by David Deutsch of Oxford University. Deutsch emphasized that quantum computers can best realize their computational potential by invoking what he called "massive quantum parallelism." Here's what he meant:

A quantum computer, like a classical computer, can be programmed to calculate a function $f(x)$. That is, imagine that the computer has two registers, two sets of qubits. In one register, the "input register," we encode an integer x with a value ranging from 0 to $2^N - 1$. That is, we prepare N spins in a state where each spin is either pointing up (0) or down (1) along the z-axis—the string of 0's and 1's is the binary expansion of the integer x. The other register, the output register, originally reads all 0's (all spins pointing up). Then the quantum computer calculates the function. It leaves the value x untouched in the input register, but it "writes" the number $f(x)$ in the output register; that is, it turns spins over as needed so that the binary expansion of $f(x)$ now occupies the output register.

That's what the quantum computer does when the input state is a string of spins that are all either up or down along the z-axis. But now Deutsch invites us to consider a different sort of input state—a state in which all N spins point in the positive x direction. Were we to measure all of the spins in this state along the z-axis, the result could be any possible string of 0's and 1's, each string occurring with a probability $1/2^N$. When we feed this input into the quantum computer, the result is a highly complex state with massive entanglement between the input and output registers. The intricate correlations between the registers encode the properties of the function f.

Clearly, a classical computer can compute $f(x)$ too, and the preparation of the massively entangled state could be simulated by a classical computer. However, the form of this state depends on the value of the function f for all 2^N possible values of the argument x, so the classical computer would need to compute the function 2^N times. The quantum computer has a big advantage (a truly enormous advantage if N is very large)—it can prepare this state by computing the function only once, as long as we choose the appropriate input state. All possible values of x occur with nonzero probability in the input state, so the quantum computer in a sense computes $f(x)$ for each possible x all in one go. This is what Deutsch meant by massive quantum parallelism. So here we have an example of a computation that a quantum computer can perform efficiently, while the simulation of the computation by a classical computer is hopelessly inefficient. But it is not yet clear whether the computation is *useful*—can we use the information encoded in this output state to answer an interesting question?

A way to put quantum parallelism to good use was pointed out in 1994 by Peter Shor (a former Caltech undergraduate who is now a computer scientist at Bell Laboratories). Shor observed that a quantum computer is good at finding the *period* of a function. It turns out (I won't explain why) that if you know how to find the period of a certain function you can find the prime factors of a number. So Shor discovered that quantum computers can (in principle) solve an important problem—they can factor large numbers efficiently, something beyond the capability of classical computers.

We say that a function f has period R if for any x, $f(x) = f(x+R) = f(x+2R)$ and so forth. If the period R is a very large number (of order $2^{N/2}$, say), then finding the period is a hard problem—we would need to sample many different values of x before we would be likely to have the good fortune to discover two different values of the input that are mapped to the same value of the output. Finding the period is like looking for a needle in a haystack, which is hard. But while finding a needle in a haystack is hard, finding a *periodic* needle in a haystack is much easier.

Suppose that after preparing Deutsch's massively entangled state, we measure the output register, hence picking out some particular value of the output $f(x)$. Correlated with this value of the output are all the possible input values that are mapped to $f(x)$. So if we now were to measure the input register, we could obtain any one of the results $x, x+R, x+2R$, etc., each occurring with equal probability. Of course, this measurement of the input register still won't tell us anything about the period R of the function—once we meaure the input once, we don't get another chance. A helpful analogy is: suppose that someone has arranged an array of periodically spaced atoms, and our task is to determine the spacing of the array. But we are allowed to make only a single measurement. We could measure the position of one of the scattering centers, but that wouldn't tell us anything about the spacing between the centers. A much better strategy is to take a single photon, scatter it from the array, and to measure the scattering angle. We know that the photon is highly likely to be scattered in certain preferred directions—the Bragg scattering angles. By measuring the direction of the scattered photon, we have an excellent chance of learning something about the spacing of the array.

Shor's method for finding the period of a period with a quantum computer is the moral equivalent of using Bragg scattering to find the spacing of a crystal. After measuring the output register, we perform a *fourier transform* on the input register—Shor showed that this can be done efficiently with a quantum computer. Then we measure the input register. The probability distribution for the measurement outcomes will be very strongly peaked about values that are integer multiples of one over the period. So by making that single measurement, we have a reasonably likelihood of learning something about the period of the function.

Hardware—the ion trap computer. I hope that I have convinced you that a quantum computer, if we could build one, would be a useful device. But what would the hardware of a quantum computer be like? The hardware of our device will need to meet a number of quite stringent specifications. We need to be able to store qubits for a long time, long enough to complete an interesting computation, and our qubits must be very well isolated from the environment, since interactions with the environment will cause errors. We must be able to manipulate the state of individual qubits, and to induce controlled interactions between qubits, so that we can perform quantum gates. And these gates must be implemented with very high precision if the machine is to perform reliably. Finally, we must be able to read out the device efficiently and reliably.

The existing technology that come closest to meeting these criteria is the linear ion trap. This is a Paul trap containing a linearly arranged array of ions. Because of their mutual Coulomb repulsion, the ions are held far enough apart that they can be individually addressed by pulsed lasers. We can store quantum information by preparing states in which each ion is in a superposition of two quantum states, say, the ground state (0) and a particular long-lived metastable excited state (1). By shining the laser on one of the ions, tuned to the transition between the ground state and the excited state, and by timing the pulse carefully, we can prepare any desired superposition of the ground state and excited state.

Reading out the device is simple: we shine a laser on each ion that is tuned to a transition from the ground state to a short-lived excited state that rapidly decays back down to the ground state. Then, if the ion is in the ground state, it will repeatedly absorb and reemit the laser light—it fluoresces, but if it is in the excited state, the laser is off resonance and nothing will happen. So when we turn on the laser, each ion in the state 0 will shine, and each ion in the state 1 will remain dark. That's how we read it out.

The most difficult part of designing and building quantum computing hardware is the *processing* of the quantum information—to perform a computation, we will need quantum gates in which two qubits interact. Here is how that is done in the ion trap (following an ingenious suggestion due to Ignacio Cirac and Peter Zoller of Insbrück University). The interaction between qubits arises from the mutual Coulomb interaction of the ions. Due to that interaction, the vibrational states of the ions in the trap are a set of coupled normal modes, with the mode of lowest frequency being the fundamental mode in which the ions rock back and forth in lockstep. Before our computation begins, we use laser cooling methods to drive this oscillator to its quantum-mechanical ground state.

We can exploit the vibrational modes as follows: Suppose that we shine a laser on ion number 1 that is tuned to the transition from ground to excited state. But we detune the laser slightly, reducing its frequency by an amount equal to the frequency of the "phonon"—that is, the fundamental vibrational mode. Then, if the ion is in its ground state, the laser is off resonance and nothing will happen. But if the ion is in its excited state, then a properly timed laser pulse will drive the ion to the ground state, while at the same time a phonon is excited—the *vibrational* state is changed from the ground state to the first excited state. Remember that in the normal mode we have excited, ion 1 and ion 2 oscillate together, in phase. Now we shine our laser on ion number 2, which is in its ground state, again detuning the laser from the transition by the phonon frequency. Then, if no phonon is already present, the laser is off resonance and nothing happens. But if a phonon is present, the phonon will be absorbed, and ion number 2 will be driven from its ground state to its excited state.

So look what we've achieved. If ion number 1 is initially in its ground state, then no phonon is ever excited and nothing ever happens. But if ion number 1 is initially in its excited state, then this operation will take ion 1 to its ground state and will take ion 2 from its ground to its excited state. So something has happened to ion 2 that is *conditioned* on the state of ion of ion 1. This is a simple *quantum gate*—and by performing many such gates is succession, we can build up a complex quantum computation.

What is the current experimental situation? The ion trap computer has been studied by Dave Wineland's group at NIST (the National Institute of Standards and Technology). The have a trap that contains a single ion; they have cooled the ion to its vibrational ground state, and have succeeded in exciting a phonon conditioned on the initial internal state of the ion as I have just described. They are now working on storing two ions in a new trap. An alternative arrangement is being pursued by Jeff Kimble's group at Caltech. They are trying to trap neutral atoms inside small optical cavities. In this case, processing will be done not by conditionally exciting phonons, but by exciting *photons*—that is, by exciting the cavity modes of the electromagnetic field.

Software: What would the software of a quantum computer be like? The quantum computer will be controlled by a classical computer. We will input into the classical computer the problem that we would like to solve, and it will determine what unitary transformation we should apply to the qubits in our device. It will also work out how a good approximation to this unitary transformation can be constructed using the available hardware. If the hardware is an ion trap, the classical computer will tell us just what sequence of laser pulses should be applied to the ions in the trap. The computer operator will apply these pulses, and then will finally read out the state of the ions by the method described earlier.

With a group of students (D. Beckman, A. Chari, and D. Srikrishna), I worked out last year what sort of computational resources would be needed to perform quantum factoring using an ion trap. Using all the tricks we could think of to optimize the time and space requirements, we found that the factoring of a 130 digit number (about the largest that can be factored with existing digital computers) could be done using a trap that contains 2160 ions by applying about 30 billion laser pulses to the ions. These are daunting numbers considering the current state of the technology! But of course the point is that these required resources scale up much more favorably than what a digital computer would require—the number of ions needed increases linearly with the number of digits in the number to be factored, and the number of pulses increases like the cube of the number of digits.

Quantum error correction:

Conclusions: I hope I have convinced you that quantum computation is interesting. We have found a whole new way of thinking about the complexity of problems, a way that is better founded on fundamental laws of physics than traditional classical complexity theory. and with this new view of complexity, some problems that used to seem very hard may turn out not to be so hard after all.

But it is clear that there is much work to do before quantum computers will become widely used and practical tools. For one thing, we need to understand better what quantum computers will be good for—I have told you about factoring, but what else is there? We would like to characterize more precisely the class of problems for which quantum computers have a big advantage over classical computers. Second, we need better quantum hardware. The ion trap is a good place to start, but surely the quantum computers of the future will use some quite different type of hardware. New ideas are needed for the design and fabrication of microscopic quantum computing devices. And third, while great recent progress has been made in understanding how to perform quantum error correction, much more can be done, especially in designing error-correction protocols that are tailored to practical working devices.

The inevitable question is—"When will I be able to walk into CompUSA and buy a quantum computer off the shelf?" Well, I don't know. It seems safe to say that commercial quantum computers are at least decades away. But even if quantum computers do not become useful computing devices until the distant future, the work we are doing now on quantum computation is interesting and valuable. Quantum complexity theory provides new insights into the classical classification of complexity. Thinking about quantum computers has already led us to a deeper understanding of the properties of quantum information, and especially into the nature of decoherence. And the experimental physicists who aspire to build rudimentary quantum gates may not be building very good computers, but they are doing interesting and important experimental physics—they are developing new methods for preparing highly entangled quantum states, and are studying the properties of those states.

The road to quantum computation may be a long and hard one, with many unexpected turns and bumps along the way. No one can say now where that road may lead us, but it seems highly likely that when we arrive we will be glad to be there. In any case, it is going to be a very interesting voyage.