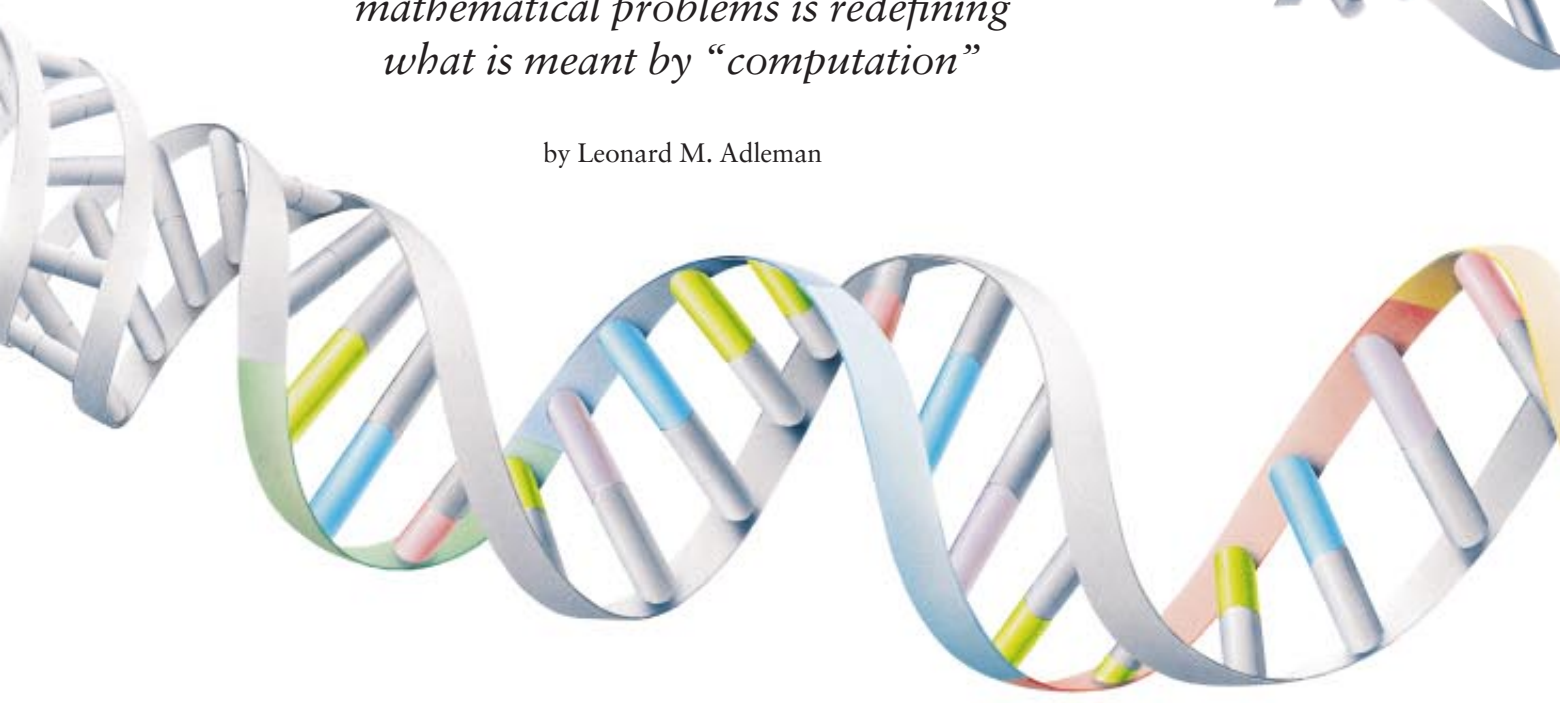
Computing with DNA

The manipulation of DNA to solve mathematical problems is redefining what is meant by "computation"

by Leonard M. Adleman



Computer. The word conjures up images of keyboards and monitors. Terms like "ROM," "RAM," "gigabyte" and "megahertz" come to mind. We have grown accustomed to the idea that computation takes place using electronic components on a silicon substrate.

But must it be this way? The computer that you are using to read these words bears little resemblance to a PC. Perhaps our view of computation is too limited. What if computers were ubiquitous and could be found in many forms? Could a liquid computer exist in which interacting molecules perform computations? The answer is yes. This is the story of the DNA computer.

Rediscovering Biology

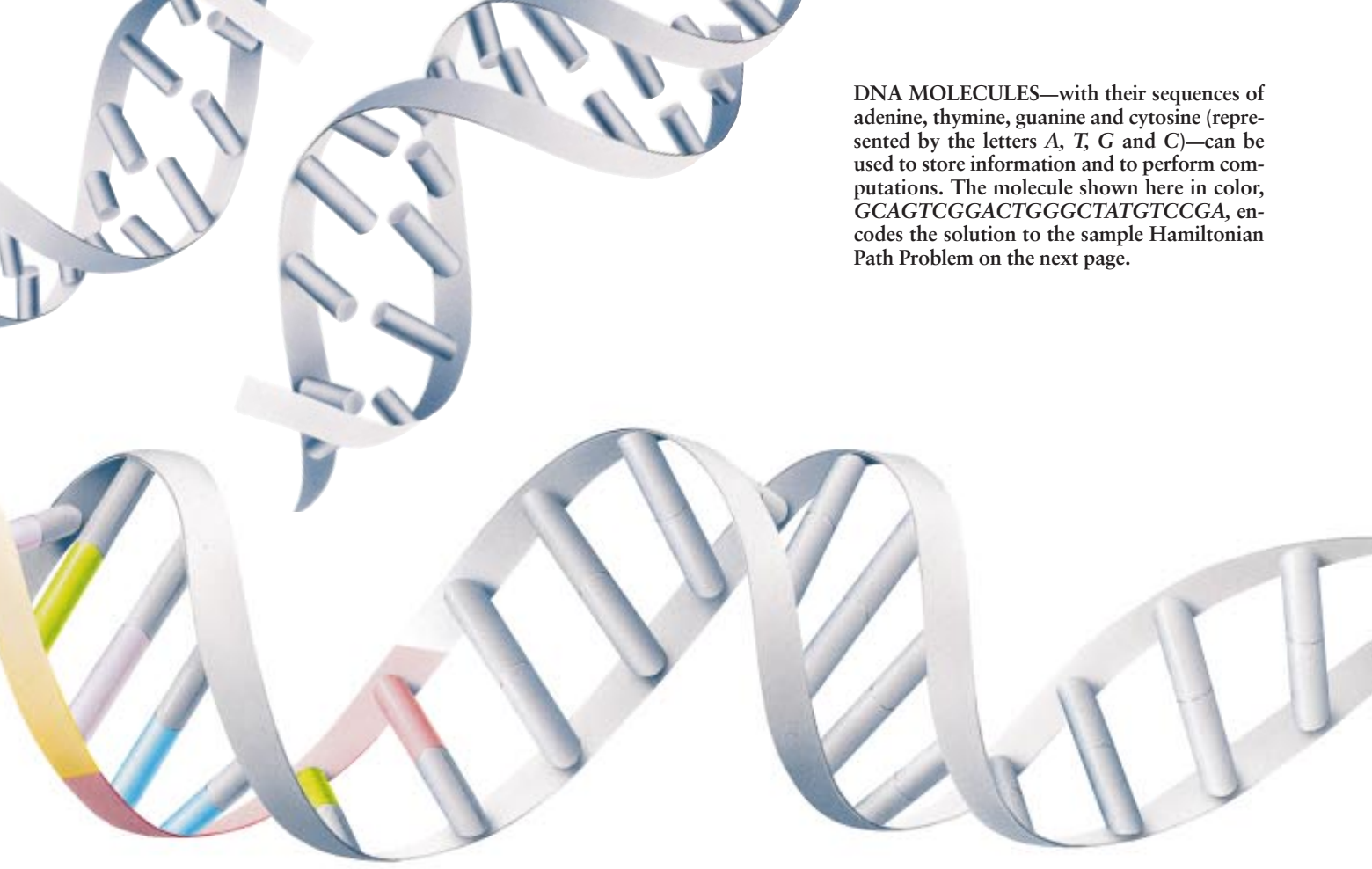
My involvement in this story began in 1993, when I walked into a molecular biology lab for the first time. Although I am a mathematician and computer scientist, I had done a bit of AIDS research, which I believed and still believe to be of importance [see "Balanced Immunity," by John Rennie; Scientific American, May 1993]. Unfortunately, I had been remarkably unsuccessful in communicating my ideas to the AIDS research community. So, in an effort to become a more persuasive advocate, I decided to acquire a deeper understanding of the biology of HIV. Hence, the molecular biology lab. There, under the guidance of Nickolas Chelyapov (now chief scientist in my own laboratory), I began to learn the methods of modern biology.

I was fascinated. With my own hands, I was creating DNA that did not exist in nature. And I was introducing it into bacteria, where it acted as a blueprint for producing proteins that would change the very nature of the organism.

During this period of intense learning, I began reading the classic text *The Molecular Biology of the Gene,* co-authored by James D. Watson of Watson-Crick fame. My concept of biology was being dramatically transformed. Biology was no longer the science of things that smelled funny in refrigerators (my view from undergraduate days in the 1960s at the University of California at Berkeley). The field was undergoing a revolution and was rapidly acquiring the depth and power previously associated exclusively with the physical sciences. Biology was now the study of information stored in DNA—strings of four letters: *A, T, G* and *C* for the bases adenine, thymine, guanine and cytosine—and of the transformations that information undergoes in the cell. There was mathematics here!

Late one evening, while lying in bed reading Watson's text, I came to a description of DNA polymerase. This is the king of enzymes—the maker of life. Under appropriate conditions, given a strand of DNA, DNA polymerase produces a second "Watson-Crick" complementary strand, in which every *C* is replaced by a *G,* every *G* by a *C,* every *A* by a *T* and every *T* by an *A*. For example, given a molecule with the sequence *CATGTC,* DNA polymerase will produce a new molecule with the sequence *GTACAG*. The polymerase enables DNA to reproduce, which in turn allows cells to reproduce and ultimately allows you to reproduce. For a strict reductionist, the replication of DNA by DNA polymerase is what life is all about.

DNA polymerase is an amazing little nanomachine, a single molecule that



DNA MOLECULES—with their sequences of adenine, thymine, guanine and cytosine (represented by the letters *A, T, G* and *C*)—can be used to store information and to perform computations. The molecule shown here in color, *GCAGTCGGACTGGGCTATGTCCGA*, encodes the solution to the sample Hamiltonian Path Problem on the next page.

"hops" onto a strand of DNA and slides along it, "reading" each base it passes and "writing" its complement onto a new, growing DNA strand. While lying there admiring this amazing enzyme, I was struck by its similarity to something described in 1936 by Alan M. Turing, the famous British mathematician. Turing—and, independently, Kurt Gödel, Alonzo Church and S. C. Kleene—had begun a rigorous study of the notion of "computability." This purely theoretical work preceded the advent of actual computers by about a decade and led to some of the major mathematical results of the 20th century [see "Unsolved Problems in Arithmetic," by Howard DeLong; Scientific American, March 1971; and "Randomness in Arithmetic," by Gregory J. Chaitin; Scientific American, July 1988].

For his study, Turing had invented a "toy" computer, now referred to as a Turing machine. This device was not intended to be real but rather to be conceptual, suitable for mathematical investigation. For this purpose, it had to be extremely simple—and Turing succeeded brilliantly. One version of his machine consisted of a pair of tapes and a mechanism called a finite control, which moved along the "input" tape reading data while simultaneously moving along the "output" tape reading and writing other data. The finite control was programmable with very simple instructions, and one could easily write a program that would read a string of *A, T, C* and *G* on the input tape and write the Watson-Crick complementary string on the output tape. The similarities with DNA polymerase could hardly have been more obvious.

But there was one important piece of information that made this similarity truly striking: Turing's toy computer had turned out to be universal—simple as it was, it could be programmed to compute anything that was computable at all. (This notion is essentially the content of the well-known "Church's thesis.") In other words, one could program a Turing machine to produce Watson-Crick complementary strings, factor numbers, play chess and so on. This realization caused me to sit up in bed and remark to my wife, Lori, "Jeez, these things could compute." I did not sleep the rest of the night, trying to figure out a way to get DNA to solve problems.

My initial thinking was to make a DNA computer in the image of a Turing machine, with the finite control replaced by an enzyme. Remarkably, essentially the same idea had been suggested almost a decade earlier by Charles H. Bennet and Rolf Landauer of IBM [see "The Fundamental Physical Limits of Computation"; Scientific American, July 1985]. Unfortunately, while an enzyme (DNA polymerase) was known that would make Watson-Crick complements, it seemed unlikely that any existed for other important roles, such as factoring numbers.

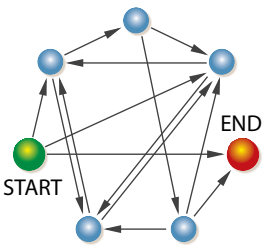
This brings up an important fact about biotechnologists: we are a community of thieves. We steal from the cell. We are a long way from being able

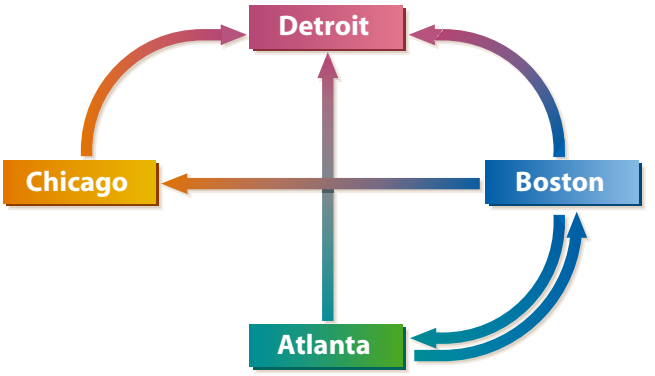
TOMO NARASHIMA

Hamiltonian Path Problem

Consider a map of cities connected by certain nonstop flights (*top right*). For instance, in the example shown here, it is possible to travel directly from Boston to Detroit but not vice versa. The goal is to determine whether a path exists that will commence at the start city (Atlanta), finish at the end city (Detroit) and pass through each of the remaining cities exactly once. In DNA computation, each city is assigned a DNA sequence (*ACTTGCAG* for Atlanta) that can be thought of as a first name (*ACTT*) followed by a last name (*GCAG*). DNA flight numbers can then be defined by concatenating the last name of the city of origin with the first name of the city of destination (*bottom right*).

The complementary DNA city names are the Watson-Crick complements of the DNA city names in which every *C* is replaced by a *G*, every *G* by a *C*, every *A* by a *T*, and every *T* by an *A*. (To simplify the discussion here, details of the 3′ versus 5′ ends of the DNA molecules have been omitted.) For this particular problem, only one Hamiltonian path exists, and it passes through Atlanta, Boston, Chicago and Detroit in that order. In the computation, this path is represented by *GCAGTCGGACTGGGCTATGTCCGA*, a DNA sequence of length 24. Shown at the left is the map with seven cities and 14 nonstop flights used in the actual experiment. —*L.M.A.*





CITY	DNA NAME	COMPLEMENT
ATLANTA	ACTTGCAG	TGAACGTC
BOSTON	TCGGACTG	AGCCTGAC
CHICAGO	GGCTATGT	CCGATACA
DETROIT	CCGAGCAA	GGCTCGTT

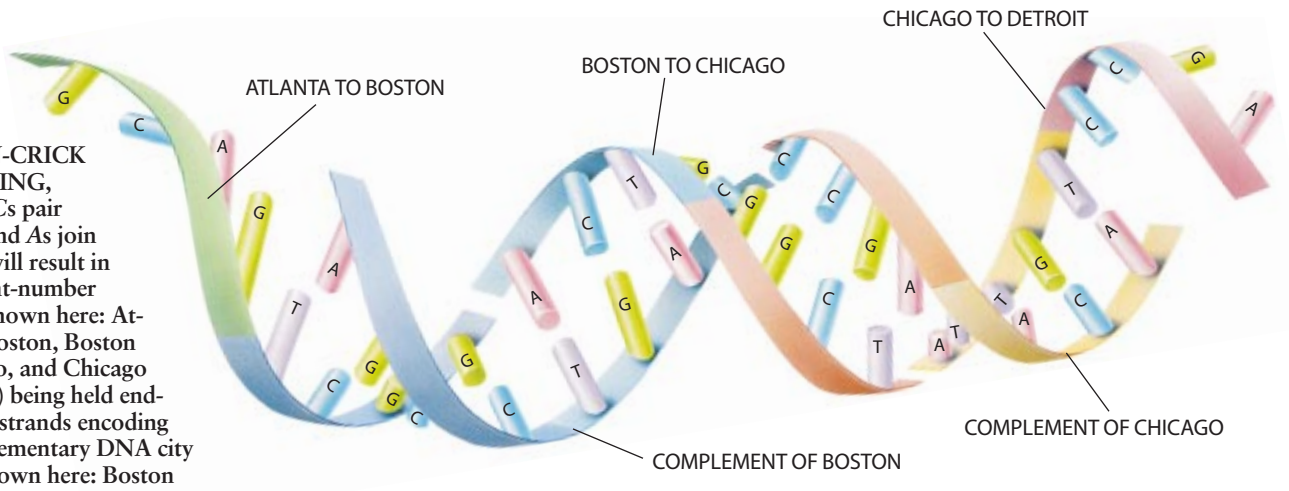
FLIGHT	DNA FLIGHT NUMBER
ATLANTA - BOSTON	GCAGTCGG
ATLANTA - DETROIT	GCAGCCGA
BOSTON - CHICAGO	ACTGGGCT
BOSTON - DETROIT	ACTGCCGA
BOSTON - ATLANTA	ACTGACTT
CHICAGO - DETROIT	ATGTCCGA

SLIM FILMS

to create de novo miraculous molecular machines such as DNA polymerase. Fortunately, three or four billion years of evolution have resulted in cells that are full of wondrous little machines. It is these machines, stolen from the cell, that make modern biotechnology possible. But a molecular machine that would play chess has apparently never evolved. So if I were to build a DNA computer that could do something interesting, I would have to do it with the tools that were at hand. These tools were essentially the following:

1. **Watson-Crick pairing.** As stated earlier, every strand of DNA has its Watson-Crick complement. As it happens, if a molecule of DNA in solution meets its Watson-Crick complement, then the two strands will anneal—that is, twist around each other to form the famous double helix. The strands are not covalently bound but are held together by weak forces such as hydrogen bonds. If a molecule of DNA in solution meets a DNA molecule to which it is not complementary (and has no long stretches of complementarity), then the two molecules will not anneal.

2. **Polymerases.** Polymerases copy information from one molecule into another. For example, DNA polymerase will make a Watson-Crick complementary DNA strand from a DNA template. In fact, DNA polymerase needs a "start



WATSON-CRICK ANNEALING, in which *C*s pair with *G*s and *A*s join with *T*s, will result in DNA flight-number strands (shown here: Atlanta to Boston, Boston to Chicago, and Chicago to Detroit) being held end-to-end by strands encoding the complementary DNA city names (shown here: Boston and Chicago).

signal" to tell it where to begin making the complementary copy. This signal is provided by a primer—a (possibly short) piece of DNA that is annealed to the template by Watson-Crick complementarity. Wherever such a primer-template pair is found, DNA polymerase will begin adding bases to the primer to create a complementary copy of the template.

3. **Ligases.** Ligases bind molecules together. For example, DNA ligase will take two strands of DNA in proximity and covalently bond them into a single strand. DNA ligase is used by the cell to repair breaks in DNA strands that occur, for instance, after skin cells are exposed to ultraviolet light.

4. **Nucleases.** Nucleases cut nucleic acids. For example, restriction endonucleases will "search" a strand of DNA for a predetermined sequence of bases and, when found, will cut the molecule into two pieces. EcoRI (from *Escherichia coli*) is a restriction enzyme that will cut DNA after the *G* in the sequence *GAATTC*—it will almost never cut a strand of DNA anywhere else. It has been suggested that restriction enzymes evolved to protect bacteria from viruses (yes, even bacteria have viruses!). For example, *E. coli* has a means (methylation) of protecting its own DNA from EcoRI, but an invading virus with the deadly *GAATTC* sequence will be cut to pieces. My DNA computer did not use restriction enzymes, but they have been used in subsequent experiments by many other research groups.

5. **Gel electrophoresis.** This is not stolen from the cell. A solution of heterogeneous DNA molecules is placed in one end of a slab of gel, and a current is applied. The negatively charged DNA molecules move toward the anode, with shorter strands moving more quickly than longer ones. Hence, this process separates DNA by length. With special chemicals and ultraviolet light, it is possible to see bands in the gel where the DNA molecules of various lengths have come to rest.

6. **DNA synthesis.** It is now possible to write a DNA sequence on a piece of paper, send it to a commercial synthesis facility and in a few days receive a test tube containing approximately 10^{18} molecules of DNA, all (or at least most) of which have the described sequence. Currently sequences of length approximately 100 can be reliably handled in this manner. For a sequence of length 20, the cost is about $25. The molecules are delivered dry in a small tube and appear as a small, white, amorphous lump.

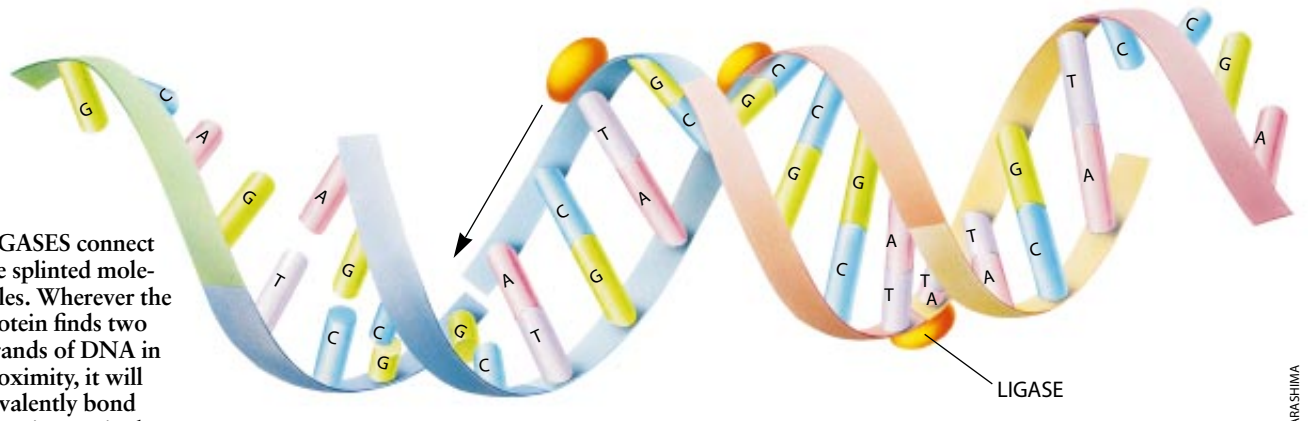
None of these appeared likely to help play chess, but there was another important fact that the great logicians of the 1930s taught us: computation is easy. To build a computer, only two things are really necessary—a method of storing information and a few simple operations for acting on that information. The Turing machine stores information as sequences of letters on tape and manipulates that information with the simple instructions in the finite control. An electronic computer stores information as sequences of zeros and ones in memory and manipulates that information with the operations available on the processor chip. The remarkable thing is that just about any method of storing information and any set of operations to act on that information are good enough.

Good enough for what? For universal computation—computing anything that can be computed. To get your computer to make Watson-Crick complements or to play chess, you need only start with the correct input information and apply the right sequence of operations—that is, run a program. DNA is a great way to store information. In fact, the cell has been using this method to store the "blueprint for life" for billions of years. Further, enzymes such as polymerases and ligases have been used to operate on this information. Was there enough to build a universal computer? Because of the lessons of the 1930s, I was sure that the answer was yes.

The Hamiltonian Path Problem

The next task was choosing a problem to solve. It should not appear to be contrived to fit the machine, and it should demonstrate the potential of this novel method of computation. The problem I chose was the Hamiltonian Path Problem.

William Rowan Hamilton was Astronomer Royal of Ireland in the mid-19th century. The problem that has come to bear his name is illustrated in the box on the opposite page. Let the arrows (directed edges) represent the nonstop flights between the cities (vertices) in the map (graph). For example, you can fly nonstop from Boston to Chicago but not from Chicago to Boston. Your job (the Hamiltonian Path Problem) is to determine if a sequence of connecting flights (a path) exists that starts in Atlanta (the start vertex) and ends in Detroit (the end vertex), while passing through each of the remaining cities (Boston and Chicago) exactly once. Such a path is called a Hamiltonian path. In the example shown on



LIGASES connect the splinted molecules. Wherever the protein finds two strands of DNA in proximity, it will covalently bond them into a single strand.

TOMO NARASHIMA

page 56, it is easy to see that a unique Hamiltonian path exists, and it passes through the cities in this order: Atlanta, Boston, Chicago, Detroit. If the start city were changed to Detroit and the end city to Atlanta, then clearly there would be no Hamiltonian path.

More generally, given a graph with directed edges and a specified start vertex and end vertex, one says there is a Hamiltonian path if and only if there is a path that starts at the start vertex, ends at the end vertex and passes though each remaining vertex exactly once. The Hamiltonian Path Problem is to decide for any given graph with specified start and end vertices whether a Hamiltonian path exists or not.

The Hamiltonian Path Problem has been extensively studied by computer scientists. No efficient (that is, fast) algorithm to solve it has ever emerged. In fact, it seems likely that even using the best currently available algorithms and computers, there are some graphs of fewer than 100 vertices for which determining whether a Hamiltonian path exists would require hundreds of years.

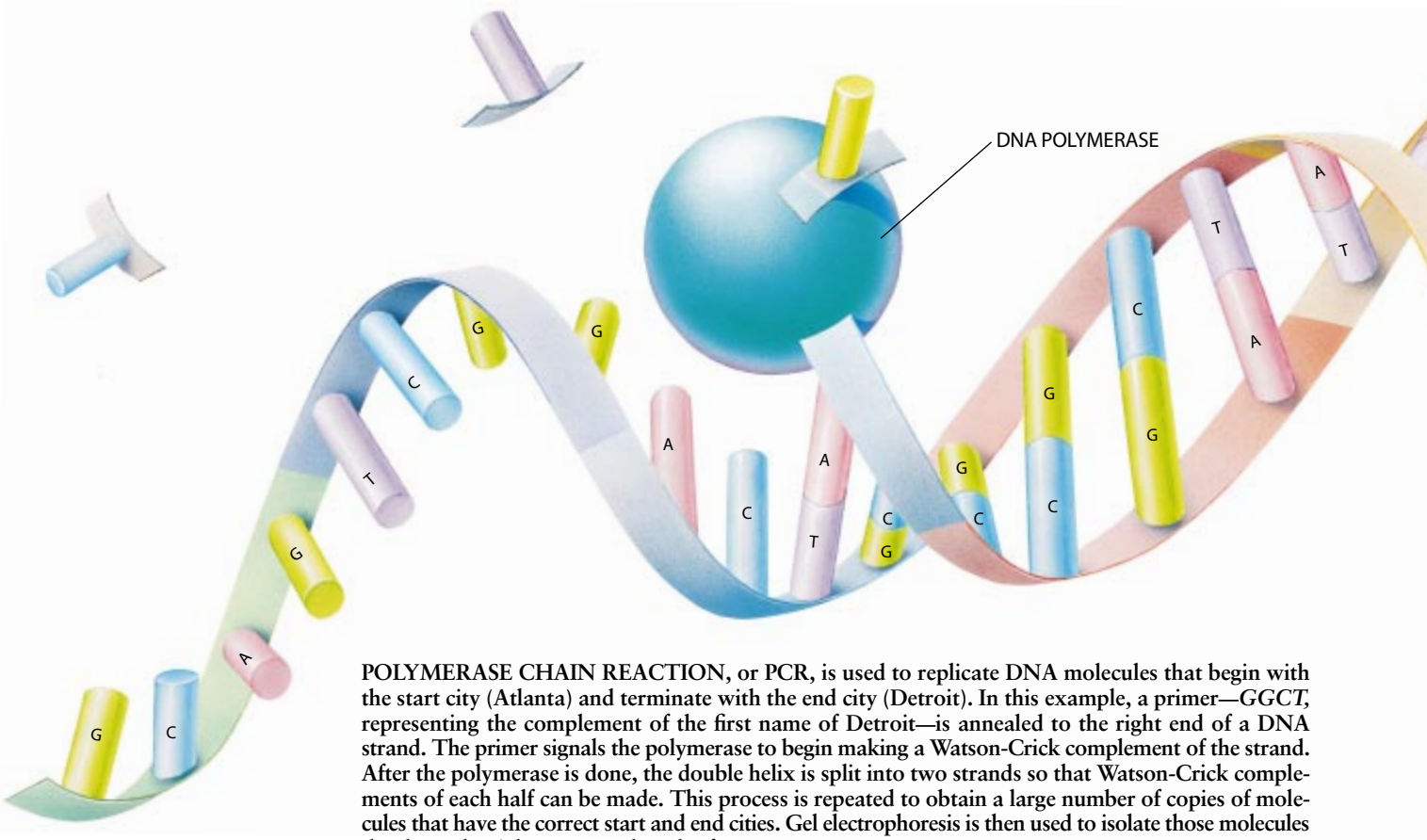
In the early 1970s the Hamiltonian Path Problem was shown to be "NP-complete." Without going into the theory of NP-completeness, suffice it to say that this finding convinced most theoretical computer scientists that no efficient algorithm for the problem is possible at all (though proving this remains the most important open problem in theoretical computer science, the so-called NP = P? problem [see "Turing Machines," by John E. Hopcroft; Scientific American, May 1984]). This is not to say that no algorithms exist for the Hamiltonian Path Problem, just no efficient ones. For example, consider the following algorithm:

Given a graph with n vertices,
1. Generate a set of random paths through the graph.
2. For each path in the set:
   a. Check whether that path starts at the start vertex and ends with the end vertex. If not, remove that path from the set.
   b. Check if that path passes through exactly n vertices. If not, remove that path from the set.
   c. For each vertex, check if that path passes through that vertex. If not, remove that path from the set.
3. If the set is not empty, then report that there is a Hamiltonian path. If the set is empty, report that there is no Hamiltonian path.

This is not a perfect algorithm; nevertheless, if the generation of paths is random enough and the resulting set large enough, then there is a high probability that it will give the correct answer. It is this algorithm that I implemented in the first DNA computation.

Seven Days in a Lab

For my experiment, I sought a Hamiltonian Path Problem small enough to be solved readily in the lab, yet large enough to provide a clear "proof of principle" for DNA computation. I chose the seven-city, 14-flight map shown in the inset on page 56. A nonscientific study has shown that it takes about 54



POLYMERASE CHAIN REACTION, or PCR, is used to replicate DNA molecules that begin with the start city (Atlanta) and terminate with the end city (Detroit). In this example, a primer—*GGCT*, representing the complement of the first name of Detroit—is annealed to the right end of a DNA strand. The primer signals the polymerase to begin making a Watson-Crick complement of the strand. After the polymerase is done, the double helix is split into two strands so that Watson-Crick complements of each half can be made. This process is repeated to obtain a large number of copies of molecules that have the correct start and end cities. Gel electrophoresis is then used to isolate those molecules that have the right sequence length of 24.

seconds on average to find the unique Hamiltonian path in this graph. (You may begin now….)

To simplify the discussion here, consider the map on page 56, which contains just four cities—Atlanta, Boston, Chicago and Detroit—linked by six flights. The problem is to determine the existence of a Hamiltonian path starting in Atlanta and ending in Detroit.

I began by assigning a random DNA sequence to each city. In our example, Atlanta becomes *ACTTGCAG,* Boston *TCGGACTG* and so on. It was convenient to think of the first half of the DNA sequence as the first name of the city and the second half as the last name. So Atlanta's last name is *GCAG,* whereas Boston's first name is *TCGG*. Next, I gave each nonstop flight a DNA "flight number," obtained by concatenating the last name of the city of origin with the first name of the city of destination. In the example on page 56, the Atlanta-to-Boston flight number becomes *GCAGTCGG*.

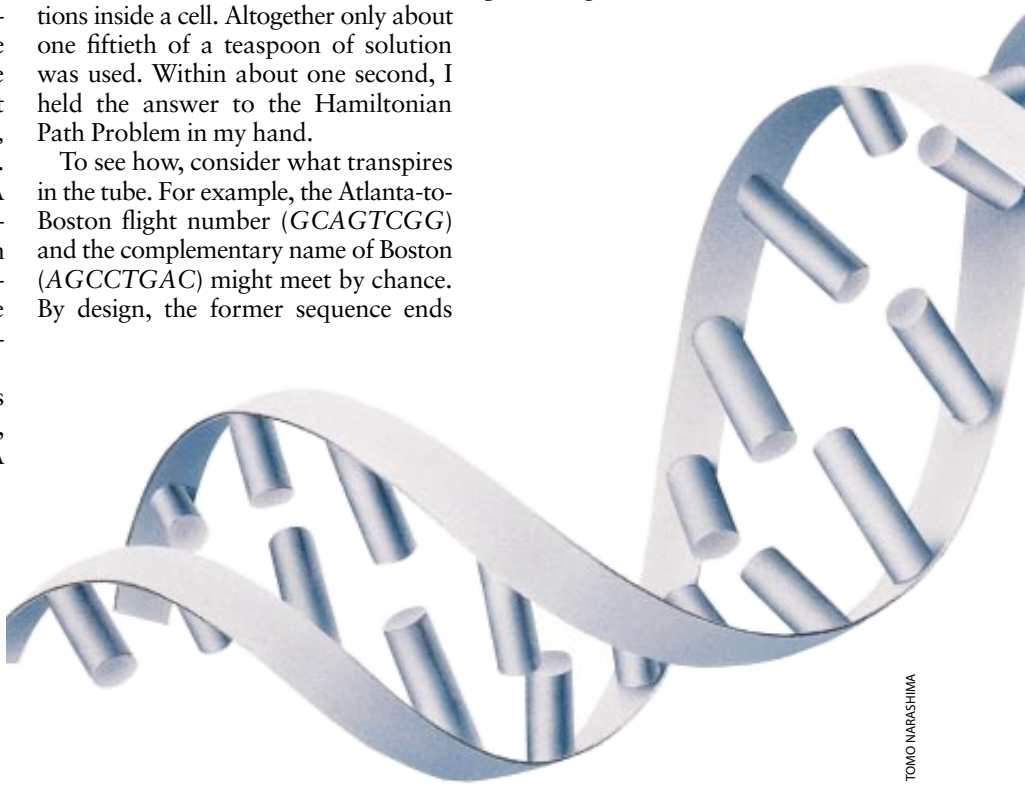
Recall that each strand of DNA has its Watson-Crick complement. Thus, each city has its complementary DNA name. Atlanta's complementary name becomes, for instance, *TGAACGTC*.

After working out these encodings, I had the complementary DNA city names and the DNA flight numbers synthesized. (As it turned out, the DNA city names themselves were largely unnecessary.) I took a pinch (about 10^{14} molecules) of each of the different sequences and put them into a common test tube. To begin the computation, I simply added water—plus ligase, salt and a few other ingredients to approximate the conditions inside a cell. Altogether only about one fiftieth of a teaspoon of solution was used. Within about one second, I held the answer to the Hamiltonian Path Problem in my hand.

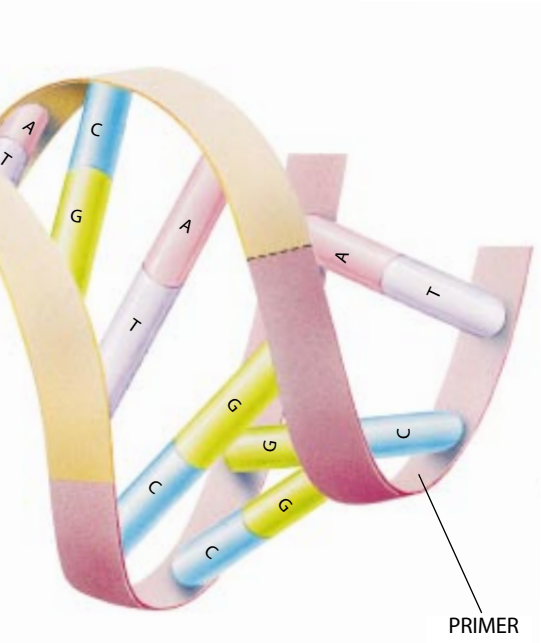
To see how, consider what transpires in the tube. For example, the Atlanta-to-Boston flight number (*GCAGTCGG*) and the complementary name of Boston (*AGCCTGAC*) might meet by chance. By design, the former sequence ends with *TCGG,* and the latter starts with *AGCC*. Because these sequences are complementary, they will stick together. If the resulting complex now encounters the Boston-to-Chicago flight number (*ACTGGGCT*), it, too, will join the complex because the end of the former (*TGAC*) is complementary to the beginning of the latter (*ACTG*). In this manner, complexes will grow in length, with DNA flight numbers splinted together by complementary DNA city names. The ligase in the mixture will then permanently concatenate the chains of DNA flight numbers. Hence, the test tube contains molecules that encode random paths through the different cities (as required in the first step of the algorithm).

Because I began with such a large number of DNA molecules and the problem contained just a handful of cities, there was a virtual certainty that at least one of the molecules formed would encode the Hamiltonian path. It was amazing to think that the solution to a mathematical problem could be stored in a single molecule!

Notice also that all the paths were created at once by the simultaneous interactions of literally hundreds of trillions of molecules. This biochemical reaction represents enormous parallel processing.

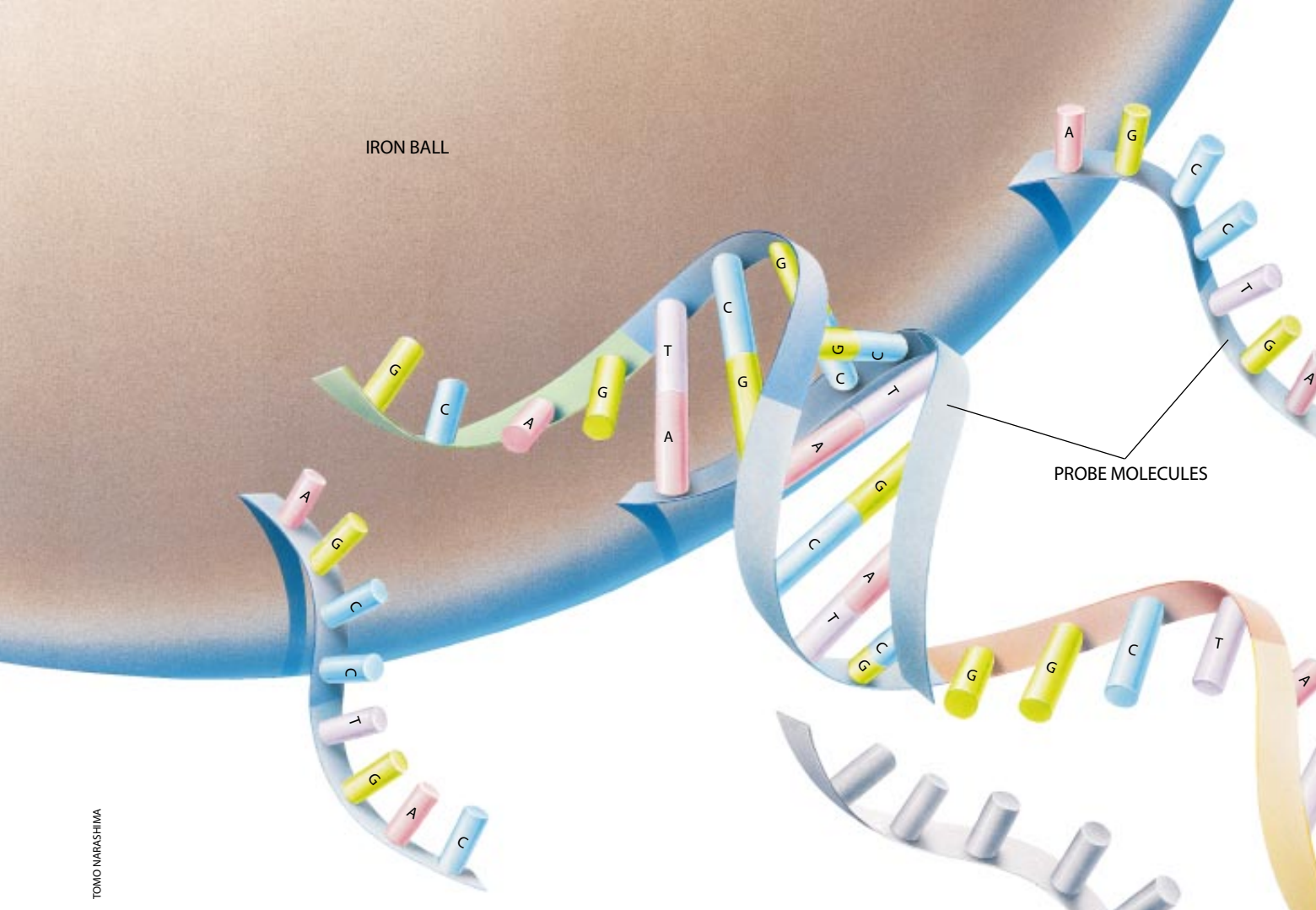


TOMO NARASHIMA



For the map on page 56, there is only one Hamiltonian path, and it goes through Atlanta, Boston, Chicago and Detroit, in that order. Thus, the molecule encoding the solution will have the sequence *GCAGTCGGACTGGGCTATGTCCGA*.

Unfortunately, although I held the solution in my hand, I also held about 100 trillion molecules that encoded paths that were not Hamiltonian. These had to be eliminated. To weed out molecules that did not both begin with the start city and terminate with the end city, I relied on the polymerase chain reaction (PCR). This important technique requires many copies of two short pieces of DNA as primers to signal the DNA polymerase to start its Watson-Crick replication. The primers used were the last name of the start city (*GCAG* for



PROBE MOLECULES are used to locate DNA strands encoding paths that pass through the intermediate cities (Boston and Chicago). Probe molecules containing the complementary DNA name of Boston (*AGCCTGAC*) are attached to an iron ball suspended in liquid. Because of Watson-Crick affinity, the probes capture DNA strands that contain Boston's name (*TCGGACTG*). Strands missing Boston's name are then discarded. The process is repeated with probe molecules encoding the complementary DNA name of Chicago. When all the computational steps are completed, the strands left will be those that encode the solution *GCAGTCGGACTGGGCTATGTCCGA*.

Atlanta) and the Watson-Crick complement of the first name of the end city (*GGCT* for Detroit). These two primers worked in concert: the first alerted DNA polymerase to copy complements of sequences that had the right start city, and the second initiated the duplication of molecules that encoded the correct end city.

PCR proceeds through thermocycling, repeatedly raising and lowering the temperature of the mixture in the test tube. Warm conditions encourage the DNA polymerase to begin duplicating; a hotter environment causes the resulting annealed strands to split from their double-helix structure, enabling subsequent replication of the individual pieces.

The result was that molecules with both the right start and end cities were reproduced at an exponential rate. In contrast, molecules that encoded the right start city but an incorrect end city, or vice versa, were duplicated in a much slower, linear fashion. DNA sequences that had neither the right start nor end were not duplicated at all. Thus, by taking a small amount of the mixture after the PCR was completed, I obtained a solution containing many copies of the molecules that had both the right start and end cities, but few if any molecules that did not meet this criterion. Thus, step 2a of the algorithm was complete.

Next, I used gel electrophoresis to identify those molecules that had the right length (in the example on page 56, a length of 24). All other molecules were discarded. This completed step 2b of the algorithm.

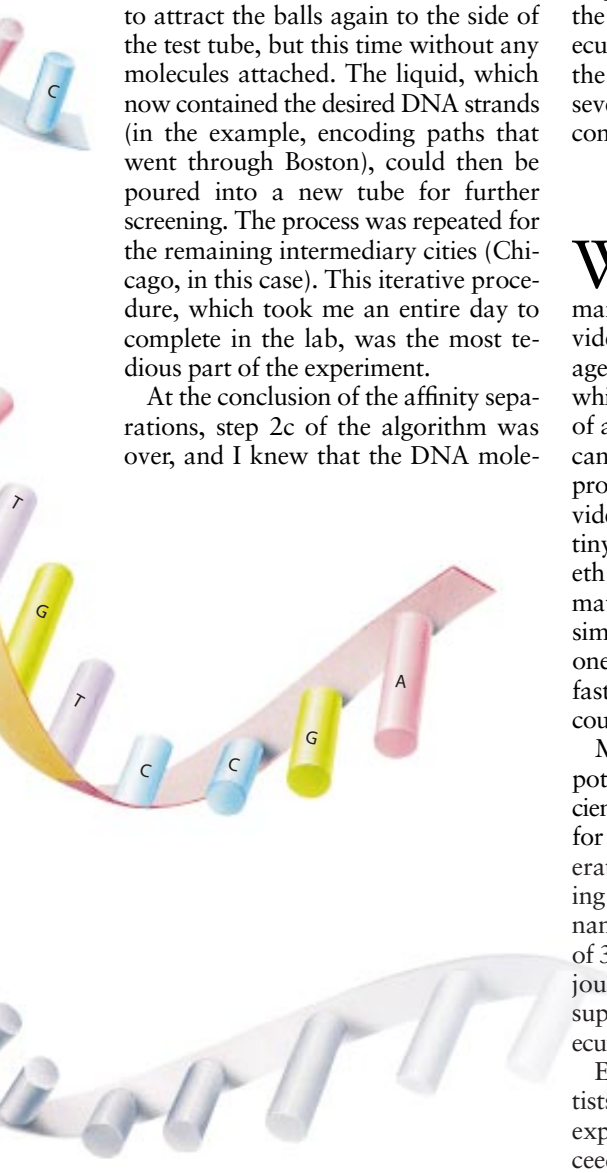
To check the remaining sequences for whether their paths passed through all the intermediary cities, I took advantage of Watson-Crick annealing in a procedure called affinity separation. This process uses multiple copies of a DNA "probe" molecule that encodes the complementary name of a particular city (for example, Boston). These probes are attached to microscopic iron balls, each approximately one micron in diameter.

I suspended the balls in the tube containing the remaining molecules under conditions that encouraged Watson-Crick pairing. Only those molecules that contained the desired city's name (Boston) would anneal to the probes. Then I placed a magnet against the wall of the

test tube to attract and hold the metal balls to the side while I poured out the liquid phase containing molecules that did not have the desired city's name.

I then added new solvent and removed the magnet in order to resuspend the balls. Raising the temperature of the mixture caused the molecules to break free from the probes and redissolve in the liquid. Next, I reapplied the magnet to attract the balls again to the side of the test tube, but this time without any molecules attached. The liquid, which now contained the desired DNA strands (in the example, encoding paths that went through Boston), could then be poured into a new tube for further screening. The process was repeated for the remaining intermediary cities (Chicago, in this case). This iterative procedure, which took me an entire day to complete in the lab, was the most tedious part of the experiment.

At the conclusion of the affinity separations, step 2c of the algorithm was over, and I knew that the DNA molecules left in the tube should be precisely those encoding Hamiltonian paths. Hence, if the tube contained any DNA at all, I could conclude that a Hamiltonian path existed in the graph. No DNA would indicate that no such path existed. Fortunately, to make this determination I could use an additional PCR step, followed by another gel-electrophoresis operation. To my delight, the final analysis revealed that the molecules that remained did indeed encode the desired Hamiltonian path. After seven days in the lab, the first DNA computation was complete.



A New Field Emerges

What about the future? It is clear that molecular computers have many attractive properties. They provide extremely dense information storage. For example, one gram of DNA, which when dry would occupy a volume of approximately one cubic centimeter, can store as much information as approximately one trillion CDs. They provide enormous parallelism. Even in the tiny experiment carried out in one fiftieth of a teaspoon of solution, approximately 10^{14} DNA flight numbers were simultaneously concatenated in about one second. It is not clear whether the fastest supercomputer available today could accomplish such a task so quickly.

Molecular computers also have the potential for extraordinary energy efficiency. In principle, one joule is sufficient for approximately 2×10^{19} ligation operations. This is remarkable considering that the second law of thermodynamics dictates a theoretical maximum of 34×10^{19} (irreversible) operations per joule (at room temperature). Existing supercomputers are far less efficient, executing at most 10^9 operations per joule.

Experimental and theoretical scientists around the world are working to exploit these properties. Will they succeed in creating molecular computers that can compete with electronic computers? That remains to be seen. Huge financial and intellectual investments over half a century have made electronic computers the marvels of our age—they will be hard to beat.

But it would be shortsighted to view this research only in such practical terms. My experiment can be viewed as a manifestation of an emerging new area of science made possible by our rapidly developing ability to control the molecular world. Evidence of this new "molecular science" can be found in many places. For example, Gerald F. Joyce of Scripps Research Institute in La Jolla, Calif., "breeds" trillions of RNA molecules, generation after generation, until "champion" molecules evolve that have the catalytic properties he seeks [see "Directed Molecular Evolution," by Gerald F. Joyce; Scientific American, December 1992]. Julius Rebek, Jr., of the Massachusetts Institute of Technology creates molecules that can reproduce—informing us about how life on the earth may have arisen [see "Synthetic Self-Replicating Molecules," by Julius Rebek, Jr.; Scientific American, July 1994]. Stimulated by research on DNA computation, Erik Winfree of the California Institute of Technology synthesizes "intelligent" molecular complexes that can be "programmed" to assemble themselves into predetermined structures of arbitrary complexity. There are many other examples. It is the enormous potential of this new area that we should focus on and nurture.

For me, it is enough just to know that computation with DNA is possible. In the past half-century, biology and computer science have blossomed, and there can be little doubt that they will be central to our scientific and economic progress in the new millennium. But biology and computer science—life and computation—are related. I am confident that at their interface great discoveries await those who seek them. SA

The Author

LEONARD M. ADLEMAN received a Ph.D. in computer science in 1976 from the University of California, Berkeley. In 1977 he joined the faculty in the mathematics department at the Massachusetts Institute of Technology, where he specialized in algorithmic number theory and was one of the inventors of the RSA public-key cryptosystem. (The "A" in RSA stands for "Adleman.") Soon after joining the computer science faculty at the University of Southern California, he was "implicated" in the emergence of computer viruses. He is a member of the National Academy of Engineering.

Further Reading

Molecular Computation of Solutions to Combinatorial Problems. Leonard M. Adleman in *Science,* Vol. 266, pages 1021–1024; November 11, 1994.

On the Path to Computation with DNA. David K. Gifford in *Science,* Vol. 266, pages 993–994; November 11, 1994.

DNA Solution of Hard Computational Problems. Richard J. Lipton in *Science,* Vol. 268, pages 542–545; April 28, 1995.

Additional information on DNA computing can be found at http://users.aol.com/ibrandt/dna_computer.html on the World Wide Web.