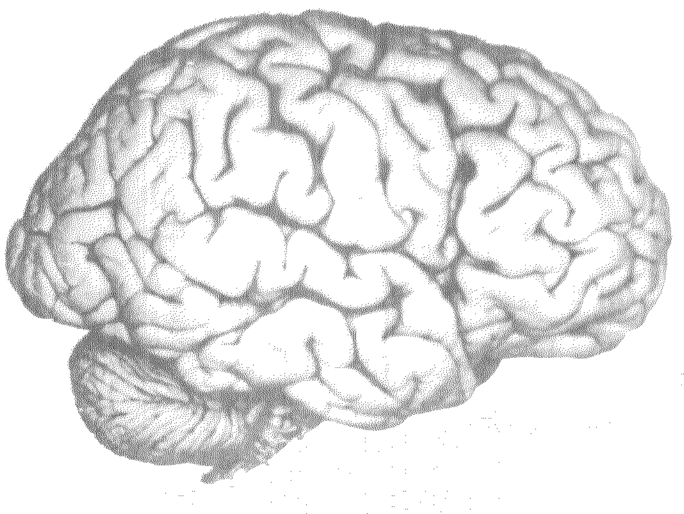
Brain, Computer, and Memory

by John Hopfield



BOTH OBJECTS above are computers. If you put reliable information into each of them, and if previously each has received appropriate programming, each will deliver what appear to be intelligent responses. At this macroscopic scale, however, it's impossible to determine whether the microprocessor and the human brain store and retrieve memories and make decisions on the same principles or on different ones.

While it isn't obvious from looking at their architecture whether or not the brain and the computer work on the same principles, it is clear that they don't carry out all tasks equally well. If you want to remember a million things accurately, it is better to do it on a computer than to trust your brain. On the other hand, your brain is capable of recognizing the face of somebody you haven't seen in ten years, even though both the face and the context may have changed considerably. It is *very* hard to write a computer program to make such a recognition, and these routine abilities of animals are astounding by comparison.

A closer look at their processing elements — a processing chip and a stained section of a brain (opposite page) — reveals more about how they work. This chip, which is one-quarter inch on a side, contains about 5,000 individual devices called gates. This is small compared to modern chips, which can have 50,000 or 100,000 individual devices on them. There are also many input and output leads to a chip — so many that you can't figure out from the circuit diagram of the chip itself how to use it. There are too many different ways of applying inputs to be able to try them all. If you don't know what the designer intended you won't find a useful function for a processor chip.

The human neural system also has processing elements — the individual nerve cells (or neurons), each with long branching arms. These branches have still finer branches, where connections are made from other cells. There are about 1,000 to 100,000 connections, or synapses, for each neuron.

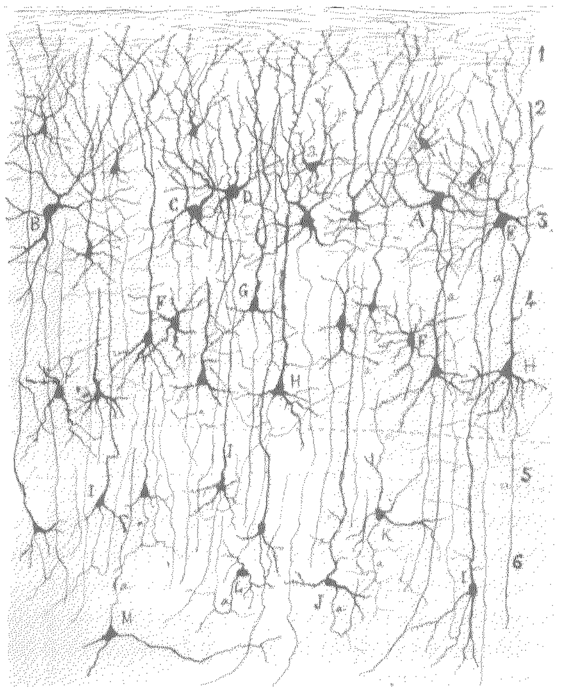
A modern computer has about ten million individual silicon gates or processing elements, each typically connected to three or four other elements. In comparison, a human brain has about 1,000 times as many cells as the modern computer does computer elements, and each cell has about 1,000 times as many synapses. So the brain is about a million times more complex than a large computer, though the computers are growing rapidly and will eventually rival a brain in complexity.

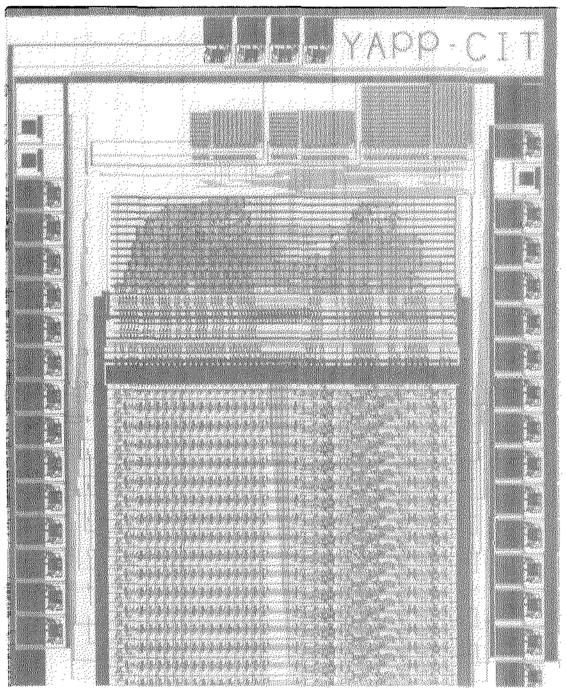
There is a major difference in the importance of the individual computing element for these systems. This difference is most visible in the sensitivity of these two computers to the failure of individual elements and suggests the occurrence of differing modes of operation. The brain of a 12-year-old child has a certain number of cells in it. By the time that child reaches 50, a few percent of the cells will have died. Yet his brain will still work very well, perhaps even better than it did at 12. In contrast, if one percent of the transistors in a computer go bad, it won't do anything at all. The reason for this difference is that the layout of the electronic computer is carefully planned. There isn't a transistor in it that wasn't put there with a purpose in mind. If that transistor goes bad, that purpose doesn't get carried out. The brain had no such grand overall planning done for it. Evolution didn't figure out ahead of time that if you build a brain exactly one certain way it will all work. Fortunately, since the biological computer must operate in a somewhat more holistic fashion than an electronic computer, it's also more fail-safe. How the brain actually manages to achieve such a mode of operation is among the most interesting problems of biology.

In physical science the customary and best way to study the fundamental interaction between simple objects is to put a few of them together and observe what happens. But new properties emerge as a result of having a very large number of those simple objects or elements. Take as an example the collision of molecules. To study simple collisions we can put two molecules in a large box. Every once in a while they collide, and that's an exciting event in the life of someone studying molecular collisons. After a while it gets boring waiting for them to collide, so we'll put in ten molecules or even 1,000. All that happens then is more frequent collisions between two objects, and the collisions will look the same as they did when there were only two molecules present.

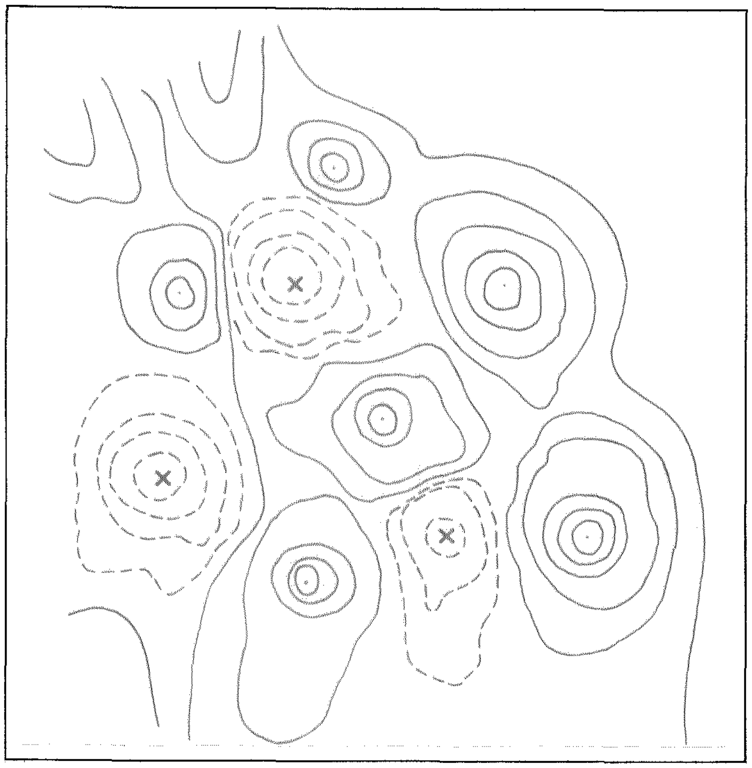
But if we put a billion billion (10^{18}) molecules in the box, there's a new phenomenon — sound waves. Sound waves wouldn't exist without collisions, which keep the sound waves organized. There was nothing in the behavior of two molecules in the box — or ten or 1,000 molecules — that would suggest to you that a billion billion molecules would be able to produce sound waves. Sound waves are a collective phenomenon, which takes place only when there are huge numbers of molecules present. Many of the laws of physics are collective in nature, including thermodynamics, hydrodynamics, magnetism, and the fact that materials have solid, liquid, and gas phases.

The brain has a huge number of elements — about 10^{13}. Do large collections of neurons have collective properties like other large physical systems? And if they do, are these collective effects used in the computations that a brain can do? One thing that the brain can do is keep memories intact. For instance, we all remember dozens of telephone numbers, with names, faces, places, and sets of experiences associated with each one. How do those telephone numbers and personal associations remain distinct without getting scrambled? Could a collective property keep a memory as an entity, unmixed with other memories?





Staining one neuron in 100 produces this picture (far left) of the brain's individual cell bodies with their branching arms that connect to other neurons. Each neuron has between 1,000 and 10,000 such connections. If the other 99 percent of the neurons were stained, this picture would be completely black. The 18-bit microprocessor (left) designed by graduate student John Wawrzynek (YAPP stands for "yet another processor project") is a one-quarter inch silicon chip containing about 5,000 individual devices, or gates. Modern chips can have 20 times this number.



A content-addressable memory can be likened to a simple physical system such as this contour map, which describes a terrain in terms of altitude. Mountain peaks are represented by dots, valley bottoms by Xs; the solid lines describe high altitude contours and the dotted lines, low contours. The valleys are like salt lakes with no connections with other valleys or escape routes worn by water. Another interpretation of this terrain appears on the following page.

A computer keeps memories distinct and held together in a way that we can liken to a very tall, very skinny library — 100,000 stories tall — with one book stored per floor. If we write the information we want to keep connected together in one book, and store the book on one particular floor, all we have to know to get that information out is the floor it's stored on. The information has obviously stayed together because it's all in one book.

That kind of memory doesn't work in biology. For one thing, as far as we know, there is no such thing as the address of a memory in the sense that there is a particular book on a particular floor. For another, a biological memory is content addressable; that is, a big memory — of a telephone number, a name, a person, the experiences you've had with that person, some places you've been together, and so on — holds together. Any part of that memory can be used to try to retrieve the whole thing. If you're reminded of any bit — the telephone number, or the name, or a common experience — all parts of the memory come together and can be retrieved. There is no one single part of the memory that plays the role of the library floor number and can be used to reach the memory. Rather, the memory can be retrieved by *any* reasonable part of the content.

Another difference between computer memory and biological memory is the locality of storage. It is generally believed that memories in our heads are much less locally stored than those in computers. What is "local" or "nonlocal" storage of memory? If I want to remember the word "Caltech," I can write "Caltech" on one page of a book. That's a local memory. If I rip the page out, the memory is not in the book any longer. If I had ripped out some other page, the memory would still be there. There is, however, another way of storing "Caltech" in a book. If I write it across the fore edge of the book (across the edges of the pages), ripping out one page or *any* particular page doesn't degrade the memory of "Caltech" written across the edge. Even if I ripped out 5 percent of the pages, I would still be able to retrieve that memory. This is nonlocal storage of information.

The trouble is, a book doesn't have many edges. If I want to store a lot of memories that way, I'm going to have to write one over another. If I do that and then try to recall a particular memory, it's going to be difficult to pull it out accurately and unmixed with others because it has many other memories written on top of it. Holding memories together in nonlocal storage is, I think, one of the fundamental problems that biological memories have had to solve.

Essential to the properties of content addressability and nonlocal storage is the fact that biological memory is not a linear system. Relations between input signals and output responses can be either linear or nonlinear. A great deal of physics and engineering is done with linear systems, and usually linear systems are easier to analyze than are nonlinear systems. But they do not always generate the desired outcome. For example, if I give a particular name to a linear memory system storing telephone numbers and names in memory, it will evoke the corresponding telephone number. If I give it a different one of its remembered names, it produces a different corresponding phone number. But if I give it a confusion of names, say, a 50-50 mixture of the two names, it will produce a 50-50 mixture, or an average of the two outputs. Since an averaged telephone number is of little use to anyone, it is fortunate that our brains don't work this way.

The essence of a content-addressable memory can be described in terms of the spontaneous behavior of appropriate simple physical systems. As an example we can use a contour map of a simple terrain, perhaps a lunar terrain, with mountain peaks and with valleys that don't connect with one another or lead down toward the ocean as they would on earth. A contour map defines a terrain in terms of altitude, but there is also another way to describe it — by a flow map showing which paths raindrops landing at any point would follow downhill, eventually to accumulate at the lowest places.

A flow map has all the information that the contour map contains but in a different representation. In looking at the flow map by itself without any knowledge of altitude differences, you would think that the raindrops were motivated by some miraculous laws of forces of attraction to specific locations. But knowing that there's a contour map behind it you know the secret — that this map doesn't come from mysterious forces. It comes from the flow pattern of water flowing downhill, and there is a terrain underneath causing the particular flow pattern.

In some sense this flow map is like a content-addressable memory. Just as a water droplet falling somewhere near the position of a valley bottom will flow to exactly the bottom position, so partial information can ultimately generate total information. Think of a particular valley location as being precise information. If you start anywhere vaguely near that position (having been given only partial information about the location of the valley) and simply follow the raindrops along, you'll come to the precise valley location. Then you'll stop moving because you've reached the lowest point around. Any one of these lowest points might be thought of as memory.

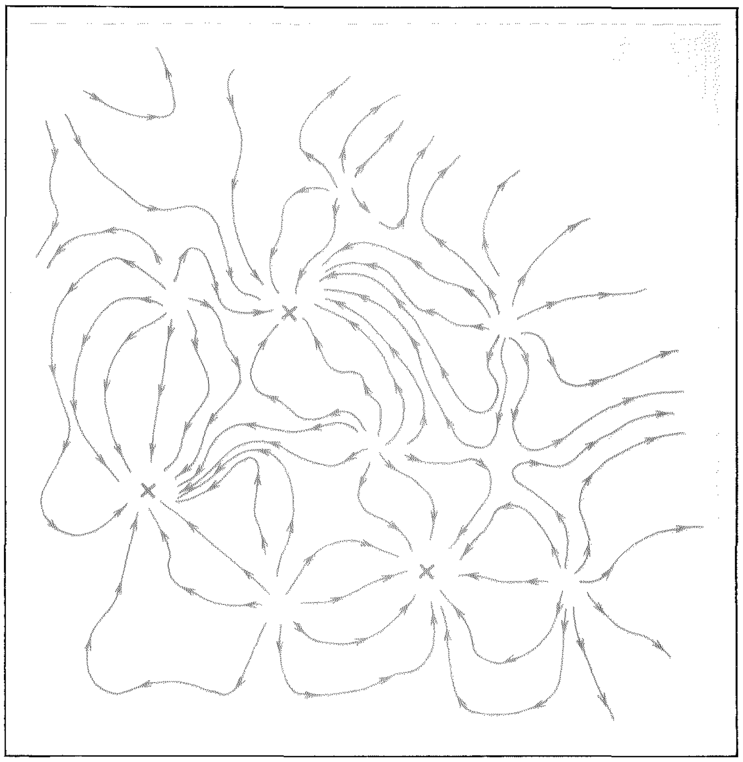
We can also describe a content-addressable memory in a computer or brain in such a way as to be able to understand it by a flow map of this sort. A computer is basically a large number of switches. At any moment each switch is either on or off. Each switch can be represented as a one, if it's on, or as a zero, if it's off. The present state of the computer can be represented by a long string of ones and zeroes, simply listing what its switch positions are at the moment. With time, the computer changes the switches around, but at any particular time you can represent what the computer is like by how the switches are set.

The same is presumably true of a set of neurons, though neural switches are more complicated — for example, they are not simply on or off. But conceptually the idea in brains is very similar. What your brain is like at any moment is described both by its connections and by which of the neurons are at the moment active (ones) and which are inactive (zeroes).

A particular memory state in a computer can be thought of as a long string of ones and zeroes — a long word in an alphabet with only two symbols. Different memories are simply different strings of ones and zeroes. In a content-addressable memory, an initial clue to a memory that somewhat resembles a particular one of the memories, say a slight "misspelling" of the ones and zeroes, will be able to change the "wrong" ones and zeroes to achieve exactly the desired memory. For example, the two states below differ in the five places indicated by arrows.

```
memory state . . . 11010110110001 . . .
initial clue   . . . 11100110101000 . . .
                     ▲▲      ▲▲  ▲
```

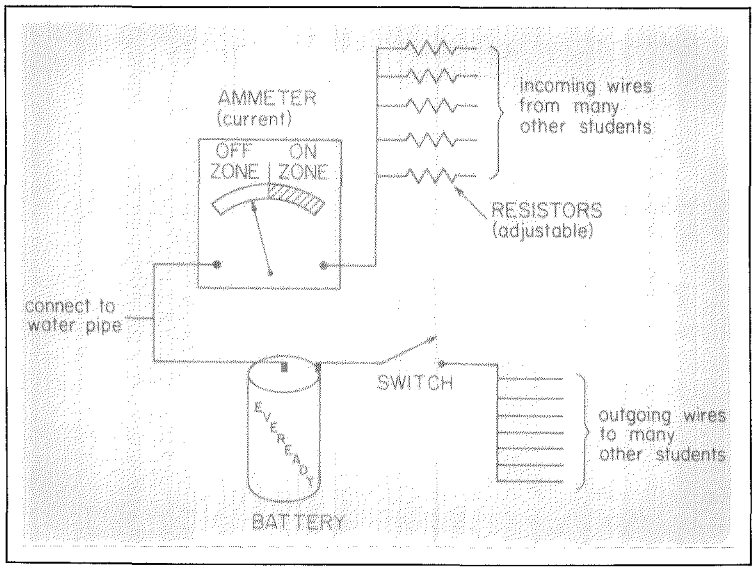
If the computer starting in a state represented by the initial clue could change these five ones to zeroes (or zeroes to ones) to match the first state, it would be behaving as a content-addressable memory.



The information of the contour map on the preceding page can also be described by a flow map. The arrow paths indicate how a drop of water would flow from higher ground ending up at an X, which marks a valley bottom. Without knowing the "secret" that the flow is downhill, it would be easy to conclude that the flow is motivated by some mysterious force.

In order to turn this description into a kind of physical map, we could say that these two states are five units apart in distance. We could then draw a picture of a space in which every point represents a possible state of the computer, a possible computer word, and some particular ones of those states are the things that we have stored as memories. What we need for a content-addressable memory is for initial states of the computer near one memory but distant from others to change with time to become the nearby memory.

If we think of a content-addressable memory in these terms, we end up with a flow map like the one generated by raindrops flowing downhill. The flow in state space tends to accumulate at the points representing the memories. If you start with information somewhat like the memory, you will be able to retrieve the whole thing. If you start halfway between two points, you will end up at one or the other, but not in a linear combination of the two.



In the "Caltech neuronal computer," a model representing 1,000 neurons as 1,000 Caltech students, each student would have on his desk an apparatus similar to this. Each has many wires coming in from and going out to other students, but it does not matter which students are connected to each other. Switches are opened and closed according to the meter at random time intervals. Surprisingly, the system will quickly settle down to one of 100 stable states out of a possible 10^{300}.

We derived the flow pattern of the figure from a very simple physical system, so it's clear that real physical systems can have the kind of flow patterns necessary for a content-addressable memory. Nor did we have to plan how to obtain content-addressability. This feature arose spontaneously. In that same sense we might ask whether the physical system consisting of a collection of neurons is of such a nature that it will spontaneously produce for its states a flow pattern like this.

I've been doing mathematical modeling of such systems, trying to abstract essential features of neurons and see if this kind of flow pattern will happen. To see what sort of simple system is capable of generating this flow, we can imagine a hypothetical system representing 1,000 neurons as 1,000 Caltech students busy working on their calculus problem sets. Each is sitting at a desk, in the corner of which is a little apparatus with a battery, a switch, and an ammeter. The switch connects to wires going out *to* many other students, but no student needs to know exactly where his own wires go. And each student, of course, has wires coming in *from* many other students through resistors and through the ammeter that measures the current. It's all hooked to a water pipe to get a "ground" return current path.

Each student is given the following instructions: Every once in a while look up from your problem set and observe your meter. If the needle is pointing to ON on the dial, *close* your switch; if your switch is already closed, leave it closed. If the needle is pointing to OFF, *open* the switch, or if the switch is already open, leave it open. Then go back to your homework.

This system preserves crudely some of the essential features of interacting nerve cells. The apparatus of each student represents a neuron. The switch being *open* or *closed* corresponds to whether each neuron is making an output or not. Each of the student neurons puts out leads to many others and gets inputs from a comparable number. The input connections from the other students are the synapses in the case of neurons, and the strengths correspond to adjustments of the resistors.

What happens if you give these instructions and then simply turn the system loose? You can describe the state of the system at any moment by writing down the names of all the students and opposite each name a zero for *open* and a one for *closed* — 1,000 ones and zeroes. There are 10^{300} different states this system can be in, which is more than the number of atoms in the universe. So it's not obvious what's going to happen when the system is turned loose. It could keep going from state to state and never run out of new ones to go to.

But this does not happen. Of that nearly infinite number of states, only about 100 will be stable. The system will quickly settle down to one of 100 memories in it. Each of these memories corresponds to a particular set of students' switch settings. Each memory consists of a 1,000-bit "word" — the amount of information in 200 typewritten characters or a little more than four of these lines of type.

This memory turns out to be content addressable. You can create a particular memory state by telling the first 50 students which way to set their switches for that memory state. Then if the other students follow the usual procedure of looking up occasionally and throwing the switch, they will actually reconstruct the particular stable memory that corresponds to the switch setting of the first 50 students. Any 50 students would suffice. So this memory is content addressable and addressable by any part of the information in it.

How is it that this system (which has no particular design principles and which might conceivably wander anywhere in its huge number of states) actually goes to stable states? In the case of the pattern of water flowing, looking only at water moving across the surface produced a mystery. Understanding that there was really a hidden contour map and that the water was flowing downhill made the whole thing understandable. There was a secret in understanding the cause of flow pattern for water.

Sometimes flows in physical systems are directed by collective principles. For example, the flow of heat from a high temperature object to a low temperature object is a collective effect, and comes about only because of the large number of molecules in each object. If the objects were tiny enough, heat would no longer always flow from the hot to the cold. Similarly, when I wrote down the mathematics that describes how on the average the students will change the positions of their switches with time, I found a quantity that plays the role of the height and contour map in the water flow problem or the temperature difference in heat flow, and drives the system in a collective fashion inexorably toward stable states.

In the Caltech student "neuronal" computer system the stable memories are determined by the particular values of the resistors that lie in the connections between students. Each connection between one student and another has one of these resistors in it, just as in neurobiology each connection between a neuron and another neuron has a synapse in the pathway. Many neurobiologists think that the place where information is stored in a biological memory is in the strength of the synaptic connections between particular neurons.

In order to have a useful memory, you need to have a way of establishing what is to be remembered. If this system (with its 100 stable states) remembers an arbitrary 400 lines of type, it's not much help. You would like to be able to specify exactly which 400 lines of type are stored out of the 10^{300} possibilities. You can do this by adjusting the resistors appropriately. There turns out to be a very simple way of adjusting them that involves no global information about the memory. You need only local information. Each resistor actually participates in many of the memories but has no idea of the total memory. There are so many resistors that if you were to take one of them out, you wouldn't forget anything. If one of the students were to go to sleep, one bit of information in the 1,000-bit word (or state) would disappear, but the memory presented by the other 999 students would remain intact and correct. Because the system is based on very large numbers and is collective in nature, it is robust against failure in a way that is highly desirable in neurobiology.

The system operates through parallel processing, doing many things together. A student doesn't have to wait and see what another student does. Each student flips a switch or not according to what his meter is doing at the moment he happens to look up. Parallel processing is one of two different kinds of processing in computer science; the other is serial processing, doing things one after another. Most computers rely heavily on serial processing.

The importance of "parallel" and "serial" in computers can be illustrated by considering two problems involving the sentences in a book. If I try to find the location of one particular sentence in a book, I can do it in parallel by tearing out the pages, distributing them to a group of people, and asking each one to read his page to see if the sentence is present. That way I'll find out very rapidly where the sentence is. Each of the tasks of looking at a particular page can be done in parallel, independently of all the others.

The position of a sentence in a book can also present a different problem. I could break up the book into its individual sentences and then try to reassemble it with the help of the same group of people. Each of them is given a page number and told to pick out a set of sentences appropriate for his particular page. If you are given page 75, you will have trouble making your choice unless you have seen the contents of pages 1 to 74. This is clearly a sequential or serial task.

Biology is different from most modern computers in that biology heavily emphasizes parallel processing. The collective processing that my simple array of students (or neurons) did is also highly parallel in nature. It accomplishes a memory retrieval task that from the point of view of planned computers might be most readily done by a sequential operation, and yet it manages to do so effectively even though operating in parallel.

Useful computational properties, which one might expect to have to carefully design, can generate themselves (in this case, at least) as collective properties of a large system of simple elements. The fascinating intactness of memories seems to come about in a spontaneous way as a collective behavior of a system with simple elements that have the plausible behavior of neurons. All the functions of the devices needed for the student computer system algorithm could easily be duplicated with neurons (or with silicon devices!). The evolution of higher nervous function and intelligent behavior would be easier to understand if some of it is based on collective properties, rather than based on the precise design necessary to make conventional computers having the capabilities of higher nervous function.

The same principles could be used to design a silicon-based computer system very different from current ones. It would have ten times as many elements but could tolerate the failure of one percent of its devices. This design, which might be called biological design, would use parallel processing in a natural way and should have corresponding speed advantages in many tasks.

Thus, I think the two computers discussed at the beginning of this article achieve their similar computational abilities rather differently. I believe that thinking about these differences will enhance our understanding of each. □