The First Chair of Computer Science at Berkeley was Richard Karp, who had just shown that the hardness of well-known algorithmic problems, such as finding the minimum cost tour for a travelling salesperson, could be related to NP-completeness—a concept proposed earlier by former Berkeley mathematics professor Steven Cook. The resulting P vs. NP question has since been accepted as one of the ten most important open problems in mathematics, along with such classics as the Riemann Hypothesis.

Berkeley computer scientists continue to lead the field of computational complexity, with work such as that on probabilistically checkable proofs and the hardness of approximation problems by Sanjeev Arora and Madhu Sudan in the early 1990s, and on quantum complexity theory by Ethan Bernstein and Umesh Vazirani a few years later. Two Turing Awards (Richard Karp, Manuel Blum) and four ACM Ph.D. Dissertation Awards (Eric Bach, Noam Nisan, Madhu Sudan, and Sanjeev Arora) are just a few of the honors garnered by the research in theoretical computer science at Berkeley.

The impact of Berkeley research on the practical end of computer science has been no less significant. The development of Reduced Instruction Set computers by David Patterson and Carlo Sequin, the Redundant Array of Inexpensive Disks project led by Randy Katz and David Patterson, and the INGRES relational database system led by Mike Stonebraker, Larry Page and Eugene Wong, can be directly connected to multi-billion dollar industries. In the area of system software, the impact of Berkeley Unix on minicomputers and subsequently on workstations and, through LINUX, on personal computers, is self-evident. Nor can we forget the role of Berkeley alumni in sparking the workstation and personal computer industry—pioneers such as Butler Lampson (Xerox PARC), Bill Joy (Sun), and Steve Wozniak (Apple). Numerical computations would not have been reliable had it not been for adoption of the IEEE 754 floating point standard, largely due to William Kahan, who received a Turing Award in 1989 for this work. In the area of programming languages and software engineering, Berkeley research has been noted to be the key for combining theory and practice, as exemplified in these pages by George Necula’s research on proof-carrying code.

While Berkeley has always led research in theory and computer systems, it was not a central player in the development of symbolic artificial intelligence, or “Good Old-Fashioned AI,” in the 1960s and 70s. Berkeley’s AI effort grew largely in the 80s and 90s, as a time when problems with this paradigm were becoming evident, and researchers at Berkeley played a major role in developing the new, more probabilistic and learning-oriented AI. This new synthesis brought traditional AI together with control theory, pattern recognition, neural networks, and statistical learning theory. Stuart Russell and Peter Norvig’s best-selling textbook has become the canonical exemplar of this synthesis, and research at Berkeley in fields such as vision, robotics and learning is bringing us ever closer to the dream of truly intelligent machines.

Berkeley was one of the first top computer science programs to invest seriously in computer graphics, and our illustrious alumni in that area have done us proud. We were not so prescient with respect to the field of human computer interaction, even though a Berkeley alumnus, Douglas Engelbart, with his 1967 demonstration, did more than any one else to lay out a futuristic vision for the field. Over the last few years, we have made up lost ground by assembling a vibrant, interdisciplinary research group in this area. Finally, computational biology, the fastest kid on the computer science block, will grow rapidly at Berkeley in the coming years under the visionary leadership of Gene Myers and Richard Karp.

This publication is the result of an outstanding team effort, ably led by Randy Katz. I am proud to share it with you.
In the spring of 2000, Berkeley EECS faculty strove across campus to offer researchers a glimpse of a bold new future. Imagine that you had your fingertips teeming with tiny, cheap “Smart Dust” sensors - sensors that could measure physical quantities, process data, and relay collected data to the outside world. What could you do with this new computational tool? The responses were as imaginative as they were diverse. Civil engineers envisioned buildings that reported structural damage after an earthquake, and smart offices that switched off light and heat in empty rooms. Other researchers proposed pulse-monitoring wristbands to alert heart specialists if a patient fainted or had a heart attack, and pollution monitors near roads to assess the environmental impact of traffic. Micro-textile sensors and microchips were bringing new possibilities to materials scientists and biologists as well.

The creation of CITRIS was a sea change in the questions Berkeley computer scientists were tackling, says James Demmel, CITRIS scientific director and a Berkeley computer science and mathematics professor. “For a long time, computer science goals have been set by the outside world. Twitter and Facebook set the trends, and engineers develop new hardware to make it easier to manage data or improve response times.”

In choosing to focus on sensor networks, CITRIS researchers had pushed a technology solidly under development. Berkeley professors Katrina Pinter and David Culler were already building prototypes networked arrays of small wireless sensors. Dating its birth between the Berkeley computer science division and the collaborative Berkeley Intel Research Laboratory in Berkeley, Culler now works on software for the sensor networks, called motes, in lab trials with motes of various shapes and sizes and control center software that enables them to transfer sensed data and communicate with one another.

The motes range in size from the one-sensor snap-on variety to mote-centered hardware to relay energy to mote-powered hardware to help motes fulfill their promise. “We've made enough progress in our research so that motes will be useful.”

The technology is up and running, but many questions remain about how to make it even more efficient and valuable. Culler's team is working on more efficient ways to route data, as well as designing ways to transmit software through the sensor network. Meanwhile, Professor David Wagner and his students are investigating how to block intruders from hacking motes, adding false nodes to a network, or reading sensitive data being beamed from mote to mote.

Beyond Berkeley, Culler estimates that about 150 research groups worldwide have adopted mote technology. Although the technology is up and running, many questions remain about how to make the technology even more efficient and valuable. Culler's team is working on more efficient ways to route data, as well as designing ways to transmit software through the sensor network. Meanwhile, Professor David Wagner and his students are investigating how to block intruders from hacking motes, adding false nodes to a network, or reading sensitive data being beamed from mote to mote.

Despite the challenges, Culler is optimistic that motes will fulfill their promise. “We've made enough progress in our research so that we can get useful data about developing real applications.”

As many new research projects start using these embedded networks, Culler believes the potential of these new promising paradigms will be huge. “In this next year or so, we will start getting to explore what data they've never been able to perceive before,” he says. “As the technology starts being deployed, I think we're going to see the revolution happen.”

In the late 1990s, researcher Kristofer Pister and David Culler were already building prototypes networked arrays of small wireless sensors. Dating its birth between the Berkeley computer science division and the collaborative Berkeley Intel Research Laboratory in Berkeley, Culler now works on software for the sensor networks, called motes, in lab trials with motes of various shapes and sizes and control center software that enables them to transfer sensed data and communicate with one another.

The motes range in size from the one-sensor snap-on variety to mote-centered hardware to relay energy to mote-powered hardware to help motes fulfill their promise. “We've made enough progress in our research so that motes will be useful.”

The technology is up and running, but many questions remain about how to make it even more efficient and valuable. Culler's team is working on more efficient ways to route data, as well as designing ways to transmit software through the sensor network. Meanwhile, Professor David Wagner and his students are investigating how to block intruders from hacking motes, adding false nodes to a network, or reading sensitive data being beamed from mote to mote.

Beyond Berkeley, Culler estimates that about 150 research groups worldwide have adopted mote technology. Although the technology is up and running, many questions remain about how to make the technology even more efficient and valuable. Culler's team is working on more efficient ways to route data, as well as designing ways to transmit software through the sensor network. Meanwhile, Professor David Wagner and his students are investigating how to block intruders from hacking motes, adding false nodes to a network, or reading sensitive data being beamed from mote to mote.

Despite the challenges, Culler is optimistic that motes will fulfill their promise. “We've made enough progress in our research so that we can get useful data about developing real applications.”

As many new research projects start using these embedded networks, Culler believes the potential of these new promising paradigms will be huge. “In this next year or so, we will start getting to explore what data they've never been able to perceive before,” he says. “As the technology starts being deployed, I think we're going to see the revolution happen.”

In the late 1990s, researcher Kristofer Pister and David Culler were already building prototypes networked arrays of small wireless sensors. Dating its birth between the Berkeley computer science division and the collaborative Berkeley Intel Research Laboratory in Berkeley, Culler now works on software for the sensor networks, called motes, in lab trials with motes of various shapes and sizes and control center software that enables them to transfer sensed data and communicate with one another.

The motes range in size from the one-sensor snap-on variety to mote-centered hardware to relay energy to mote-powered hardware to help motes fulfill their promise. “We've made enough progress in our research so that motes will be useful.”

The technology is up and running, but many questions remain about how to make it even more efficient and valuable. Culler's team is working on more efficient ways to route data, as well as designing ways to transmit software through the sensor network. Meanwhile, Professor David Wagner and his students are investigating how to block intruders from hacking motes, adding false nodes to a network, or reading sensitive data being beamed from mote to mote.

Beyond Berkeley, Culler estimates that about 150 research groups worldwide have adopted mote technology. Although the technology is up and running, many questions remain about how to make the technology even more efficient and valuable. Culler's team is working on more efficient ways to route data, as well as designing ways to transmit software through the sensor network. Meanwhile, Professor David Wagner and his students are investigating how to block intruders from hacking motes, adding false nodes to a network, or reading sensitive data being beamed from mote to mote.

Despite the challenges, Culler is optimistic that motes will fulfill their promise. “We've made enough progress in our research so that we can get useful data about developing real applications.”

As many new research projects start using these embedded networks, Culler believes the potential of these new promising paradigms will be huge. “In this next year or so, we will start getting to explore what data they've never been able to perceive before,” he says. “As the technology starts being deployed, I think we're going to see the revolution happen.”

In the late 1990s, researcher Kristofer Pister and David Culler were already building prototypes networked arrays of small wireless sensors. Dating its birth between the Berkeley computer science division and the collaborative Berkeley Intel Research Laboratory in Berkeley, Culler now works on software for the sensor networks, called motes, in lab trials with motes of various shapes and sizes and control center software that enables them to transfer sensed data and communicate with one another.
What do you decide to leave Celera and return to academics?

I want to learn and incubate new ideas. When I left Celera, I was a scientist--I created a particular vision, to see the genome, and now we’ve done it. My feeling is that coming back to a university is the right thing to do. It’s a place where I’ll have time to think about biology and computer science, and how they relate to each other. In the commercial world, when you’re at an airport and the clock behind you, you cannot move around. And we moved a mountain at Celera. But companies also have to make money. They have to keep up with being a biologist and still being a computer scientist at the same time.

I’m trying to get a feel for the Zen of experimental design. I want the freedom to speculate. Take today, for instance—I got to spend three hours just hanging out, learning how to program the Illumina sequencer, how to pipeline the sequence reads, how to turn on the centrifuge. In industry, I wouldn’t get the chance to do that.

What do you think computer science can contribute to genetics?

I don’t think computer science is going to change the way humans think about how cells work, but it will be extremely important for interpreting experimental results. You do an experiment and you don’t get exactly what you want. Take X-ray crystallography, for instance. If you send an X-ray through a crystal, you get a different pattern, and then it’s a computer that actually figures out what shape those peaks would produce that pattern. Or take sequence assembly: you use a genome into a piece of genomes, sequence about 40 million snippets, and then you have to figure out what was the genome those snippets were sampled from.

Computation is especially important for modeling and holding information. It’s clear that computation is essential to biology, because of the scale of the system we’re trying to think about. We’re talking about a system involving 10^10s of molecules. I’m trying to get a feel for the Zen of experimental design, to think about. We’re talking about a system involving 10^10s of molecules, but it will be extremely important for interpreting experimental results.

Why have you chosen Berkeley?

There are lots of reasons. Berkeley has a great computer science department, one of the best in the country, so that’s kind of a no-brainer. Biology is also extremely strong here, and there are very good people here. There’s also the whole group at Lawrence Berkeley Laboratory, great people who can do systems-level engineering. And there’s the Joint Genome Institute in Walnut Creek, that’s affiliated with us; it’s a huge base of kind of operations research at UC Berkeley is a very good place to be positioned in terms of the kinds of activities going on around here. I feel that I’m in the right place.
1980 CONTINUED VMS operating system. running 3BSD UNIX than DEC's own VMS□operating system.□GRAHAM and her students, gprof is developed. OVER THE PAST QUARTER-CENTURY, 4.2 BSD UNIX incorporates the Internet's TCP/IP. computer-aided design programs to the light years, from the bare wire-frame models of Oscar-winning films. created shock value on the movie screen in the design of campus buildings, and has even modern dance. Their work has influenced how to render buildings, cityscapes and light years, from the bare wire-frame models of Oscar-winning films.

On April 20, 1997, a team of graduate and undergraduate students under Professor Jitendra Malik into the lab to be the first to see the film. The Marquee was packed to the curb. A late night showing of a film on campus at the Berkeley campus, which has always been very popular. In 1998, the film premiered to great acclaim at the prestigious Electronic Theater Awards at the annual meeting of SIGGRAPH, the main association for computer graphics professionals.

The campus movie is a team effort, and its success is due to the collaboration of many individuals. The sound of the camera, the lighting, the photography, and the editing are all important elements in the creation of the film. The camera is used to capture the light and shadow, and the images are combined to create a seamless narrative. The computer graphics are used to enhance the images, and the final product is a stunning visual experience that captures the beauty of Berkeley's campus.

In 2000, Borshukov and two colleagues were awarded a Scientific and Technical Academy Award. "The development of a system for image-based rendering allowed character movement in computer graphics to be used by artists." Borshukov, who has made a career in the visual effects industry, says, "It's been great to be involved in a technology that's allowed filmmakers to do things they couldn't do before. The high point is for technology that's utilized in filmmaking to be used by others."
**THE SODA HALL WALKTHRU**

In the late 1980s, the Berkeley campus was gearing up to build a much-needed building for the computer science division. Carlo Sequin, the computer science professor heading the building committee, had already caught many flames in the two-dimensional architectural plans. But, he thought, could he do his job much better if he were able to take a virtual walk through a three-dimensional model of the building? At that time, that was a tall order. A detailed computer model of the new building, Soda Hall, was in the process of being developed. State-of-the-art walk-through programs could handle only the order of thousands of polygons. Even so, the idea of viewing a 3-D model of Soda Hall was too large for a computer's memory. A frame in real time would require hundreds of thousands of polygons. Polygons, the bubbles, the doors, and other objects which parts of the building could be seen. Drawing on his background in theoretical computer science and computational geometry, Teller came up with an efficient algorithm that could handle the complex shapes and objects in the model. Teller, says, graphics algorithms decided what was visible one polygon or seen one point at a time. "This idea was here to make decisions about visibility on a chunk-by-chunk basis," he says.

Teller's algorithm was so successful that it led to the development of the rendering program. With less time and effort required to render the model, the vision of Soda Hall became a reality. In 1986, the project was completed and the building was ready for use.

---

**THE SODA HALL WALKTHRU**

In the late 1980s, the Berkeley campus was gearing up to build a much-needed building for the computer science division. Carlo Sequin, the computer science professor heading the building committee, had already caught many flames in the two-dimensional architectural plans. But, he thought, could he do his job much better if he were able to take a virtual walk through a three-dimensional model of the building? At that time, that was a tall order. A detailed computer model of the new building, Soda Hall, was in the process of being developed. State-of-the-art walk-through programs could handle only the order of thousands of polygons. Even so, the idea of viewing a 3-D model of Soda Hall was too large for a computer's memory. A frame in real time would require hundreds of thousands of polygons. Polygons, the bubbles, the doors, and other objects which parts of the building could be seen. Drawing on his background in theoretical computer science and computational geometry, Teller came up with an efficient algorithm that could handle the complex shapes and objects in the model. Teller, says, graphics algorithms decided what was visible one polygon or seen one point at a time. "This idea was here to make decisions about visibility on a chunk-by-chunk basis," he says.

Teller's algorithm was so successful that it led to the development of the rendering program. With less time and effort required to render the model, the vision of Soda Hall became a reality. In 1986, the project was completed and the building was ready for use.

---

**THE SODA HALL WALKTHRU**

In the late 1980s, the Berkeley campus was gearing up to build a much-needed building for the computer science division. Carlo Sequin, the computer science professor heading the building committee, had already caught many flames in the two-dimensional architectural plans. But, he thought, could he do his job much better if he were able to take a virtual walk through a three-dimensional model of the building? At that time, that was a tall order. A detailed computer model of the new building, Soda Hall, was in the process of being developed. State-of-the-art walk-through programs could handle only the order of thousands of polygons. Even so, the idea of viewing a 3-D model of Soda Hall was too large for a computer's memory. A frame in real time would require hundreds of thousands of polygons. Polygons, the bubbles, the doors, and other objects which parts of the building could be seen. Drawing on his background in theoretical computer science and computational geometry, Teller came up with an efficient algorithm that could handle the complex shapes and objects in the model. Teller, says, graphics algorithms decided what was visible one polygon or seen one point at a time. "This idea was here to make decisions about visibility on a chunk-by-chunk basis," he says.

Teller's algorithm was so successful that it led to the development of the rendering program. With less time and effort required to render the model, the vision of Soda Hall became a reality. In 1986, the project was completed and the building was ready for use.

---

**THE SODA HALL WALKTHRU**

In the late 1980s, the Berkeley campus was gearing up to build a much-needed building for the computer science division. Carlo Sequin, the computer science professor heading the building committee, had already caught many flames in the two-dimensional architectural plans. But, he thought, could he do his job much better if he were able to take a virtual walk through a three-dimensional model of the building? At that time, that was a tall order. A detailed computer model of the new building, Soda Hall, was in the process of being developed. State-of-the-art walk-through programs could handle only the order of thousands of polygons. Even so, the idea of viewing a 3-D model of Soda Hall was too large for a computer's memory. A frame in real time would require hundreds of thousands of polygons. Polygons, the bubbles, the doors, and other objects which parts of the building could be seen. Drawing on his background in theoretical computer science and computational geometry, Teller came up with an efficient algorithm that could handle the complex shapes and objects in the model. Teller, says, graphics algorithms decided what was visible one polygon or seen one point at a time. "This idea was here to make decisions about visibility on a chunk-by-chunk basis," he says.
MOVING ABOUT: VEHICLES THAT PILOT THEMSELVES

**Michael started doing flying robots somewhat on a lark, in 1996,** says EECS department chair Shankar Sastry. “Robots on the ground were getting too passé.” People told Sastry, “That’s pie in the sky, you’ll never fly,” he recalls. “We started doing flying robots somewhat on a lark, in 1996,” says EECS department chair Shankar Sastry. “Robots on the ground were getting too passé.” People told Sastry, “That’s pie in the sky, you’ll never fly,” he recalls.

**Another important aspect of Berkeley’s expertise in the field of theoretical developments in concrete problems.** For example, “It’s not just what you measure,” says Malik. “It’s not just what you measure.” For example, “It’s not just what you measure.”

**The system has to identify cones and lane markers under all conditions.** In bad lighting and bad weather—no, not at the same time, you cannot—and then that’s a whole computational expense algorithms, says Malik. The team maintained the challenge of not finding autonomy under autonomous control has been automated, and now it is up to industry to carry on.” Mali says.

**Building a robot in the air, on the other hand, presents a whole host of challenges not found by cars on the ground.** Not least among these is how vehicles on the ground. Like the car, an airplane’s let’s us be able to understand visual images. So that it can, in general, judge the distance to the pitching deck of a ship in the dark. But unlike the car, it needs to keep moving constantly for stability, and react and respond to shifting wind speeds and unexpected obstacles. For complex maneuvers, the robot can find itself in a place where the road is not obvious, and it needs to be built, the very first Ivan van der Sterre. von Neumann Lecture, honoring his contributions to mathematics and theoretical computer science. According to Richard Karp, professor of computer science at Berkeley, the project has resulted in a “whole-merging” view of AI.

**A Berkeley researcher has made use of a variety of robots to address their revolutionary algorithms for Autonomous control flight.** After hours and hours of practice, human pilots need unflagging vigilance to monitor the endless shifts in wind direction and speed, and the speed of a surprise to avoid collision. The controls can be fast, and the work of a surprise to avoid collision. The controls can be fast.
The Internet contains pretty much everything we have—information, perhaps a website, a forum, a book or a database. It has a natural problem: how to search the vast stores of information for anything you need. The answer is often simple: ask a human. The problem is that humans store and retrieve information in entirely different ways. The internet provides a natural interface for people to interact with information. However, there are still many challenges in making the internet a truly intelligent system. The internet contains vast amounts of information in different formats and languages. The key to making the internet a truly intelligent system is to develop intelligent systems that can understand and interact with information in a way that is meaningful to people. This requires the development of intelligent systems that can learn, reason, and adapt to different contexts. The internet also presents challenges in terms of security and privacy. As the internet continues to grow and evolve, it will be crucial to develop intelligent systems that can protect personal information and prevent malicious attacks. The internet is a vast and complex system, and there is still much work to be done in order to make it truly intelligent.
In the National Academy of Sciences' 2005 report Funding a Revolution: Government Support for Computer Research and Development, the joint NASA-DoD study stressed that the main impact of government research and development dollars was in various areas of computer science. Research at Berkeley had a major impact in areas of work, including BSD (Berkeley Standard Distribution), EISC (Reduced Instruction Set Computers), and PARIS (Parallel Arrays of Instruction Set Processors).

Role of University R&D in Creation of Industries

Innovations

- BSD (Berkeley Standard Distribution)
- EISC (Reduced Instruction Set Computers)
- PARIS (Parallel Arrays of Instruction Set Processors)

Metters and Facts

- Total Degrees Awarded: 460 Ph.D., 5150 M.S.
- Masters and Ph.Ds Awarded—Computer Science
- Faculty and Ph.D. Students—Computer Science

Metrics of Excellence

- Total Degrees Awarded: 701 Ph.D., 441 M.S.
- Masters and Ph.Ds Awarded—Computer Science
- Faculty and Ph.D. Students—Computer Science

Chairs of Computer Science

1973-1975 Richard Karp
1975-1977 Elwyn Berlekamp
1977-1980 Manuel Blum
1980-1983 Carlos Seguin
1983-1987 Domenico Ferrari
1987-1990 Richard Friedman
1990-1993 David Patterson
1993-1996 Robert Milosky
1996-1999 Randy Katz
1999-2002 Christos Papadimitriou
2002-Present Jitendra Malik

Impact on Education: Influential Publications by Berkeley Faculty

- Berkeley Computer Science has always led in teaching, and not just in the classroom. The CS faculty have authored many standard textbooks to subsidiaries of Computer Science such as programming, hardware design, architectures, artificial intelligence, and computer vision.

- Berkeley has been a leader in computer science education and research for over 50 years, with a strong emphasis on interdisciplinary collaboration.

- The department has produced numerous influential publications and has been at the forefront of many major developments in the field.
TIMELINE

1990 CONTINUED

FreeBSD, NetBSD, BSDi. the foundation for several ports of UNIX to PCs.

DURING THE EARLY 1990S

VLSI-BAM

BERKELEY COMPUTER SCIENTISTS DEVELOP

VLSI-BAM: a single chip architecture for Prolog.

Pietro Perona (Berkeley Ph.D. 1990) and Jitendra Malik INTRODUCE

the anisotropic diffusion partial differential equation for image processing.


Pseudorandom Generators.”

Noam Nisan WINS ACM DISSERTATION AWARD

for “Using Hard Problems to Create

Pseudorandom Generators.”

John Hennessy, David Patterson and STANFORD PROFESSOR


AWARD

David Patterson WINS THE ACM KARL V.

Henry Karsten OUTSTANDING EDUCATOR AWARD

Elwyn Berlekamp receives the IEEE Hamming Medal.

Soft Computing): a consortium of fuzzy logic, neurocomputing, evolutionary computing, probabilistic

Towards Soft Computing:

LOFTI ZADEH LAUNCHES BISC

(Berkeley Initiative in Soft Computing): a consortium of fuzzy logic, neurocomputing, evolutionary computing, probabilistic

1991 1992

softness in visibility and importance.

Soft Computing): a consortium of fuzzy logic, neurocomputing, evolutionary computing, probabilistic

Until now, reliability efforts have been confined

Researchers in the Berkeley group have integrated all three.

founder of the Berkeley Fault-Tolerance Group in Linear Algebra. Follow-on papers by

JAMES DEMMEL AND WILLIAM KAHAN

GARTH GIBSON WINS ACM DISSERTATION AWARD

for “Redundant Disk Arrays: Reliable,

C|TRIS  |  MYERS  | GRAPHICS  |  INTELLIGENT SYSTEMS |  ROC |  CS 160  |  INTERDISCIPLINARY THEORY  |  PCC  |  COMPLEXITY THEORY

A loose and lay of following pages:

Exploration from the Berkeley Experimental Machine Room, where the ROC project is put to the test.
BERKELEY COMPUTING SYSTEMS RESEARCHERS HAVE PIONEERED THE CONCEPT OF THE RESEARCH RETREAT, WHERE PROFESSOR DAVID PATTERSON, GARTH GIBSON (PROFESSOR, CMU), PROFESSOR RICHARD FATEMAN, SHIN DAVID WOOD (PROFESSOR, UWISCONSIN), JAMES LARUS (MICROSOFT), LUIGI SEMENZANO, PROFESSOR RANDY KATZ (PACIFIC RESEARCH CENTER), DAVID CULLER AND HIS STUDENTS HAVE FOUND IT ESSENTIAL TO THEIR WORK.

In the context of computing systems research, the research retreat is a unique activity that allows researchers to step away from their day-to-day responsibilities and focus on broader, more long-term goals. This approach has been instrumental in fostering innovation and collaboration among researchers. The concept of the research retreat has been pioneered by Berkeley computing systems researchers, who recognized the importance of setting aside time to reflect on the direction of their work and the challenges they face.

The retreats are designed to provide a peaceful and productive environment for researchers to engage in deep thinking and strategic planning. During these retreats, participants are encouraged to discuss emerging trends, tackle complex problems, and explore new avenues of research. The retreats often involve small group discussions, brainstorming sessions, and personal reflection time, all aimed at fostering creativity and innovation.

One of the key benefits of these retreats is the ability to break away from the distractions of daily life and focus on the bigger picture. This can lead to the generation of new ideas, the development of innovative solutions, and the strengthening of collaborations among researchers. The retreats also provide an opportunity for researchers to revisit foundational concepts and reframe their work in a new context, which can lead to breakthroughs and new insights.

Overall, the research retreats have become an integral part of the computing systems research landscape at Berkeley and beyond. They have contributed significantly to the advancement of computing systems research and have inspired similar initiatives at other institutions around the world.
IT'S NOT EVERY COMPUTER SCIENCE COURSE THAT SENDS STUDENTS OUT TO INTERVIEW FIREFIGHTERS OR ESCORT BLIND SHoppers THROUGH A SUPERMARKET. BUT IN COMPUTER SCIENCE 160—USER INTERFACE DESIGN AND DEVELOPMENT—BLINDYSHOPS GO FACE-TO-FACE WITH A PART OF THE COMPUTING WORLD THAT TO MANY COMPUTER SCIENTISTS, IS A FACELESS ABSTRACTION. REAL, LIVE USERS.

CS 160 is a course in human-computer interaction. As an assistant professor at the University of California, Berkeley, Michael Consolvo is one of the people who teach it. And while the course is still in its infancy, he is already finding ways to make it more relevant to the real world. Recently, for example, he took his students to the fire station to interview firefighters. He wanted to learn more about how they use computers in their jobs and how they think about them. The firefighters were interested in the course and said they had never thought about it before. They told him that it was important to them and that they would like to help in any way they could.

The experience was a revelation for Consolvo. He realized that the course was just as important to the firefighters as it was to the students. He decided to make it a regular part of the course. Today, CS 160 is one of the most popular courses at Berkeley. In fact, the course has been so successful that Consolvo is now thinking about expanding it to include more students.

The course is built around a project that requires students to design and build a user interface for a real-world application. The students are divided into small groups, and each group is assigned a project that is related to a specific area of computer science. For example, one group might be asked to design a user interface for a medical diagnostic system, while another group might be asked to design a user interface for a virtual reality application. The students are given a few weeks to design and build their interfaces, and then they present them to the class. They are also required to write a paper describing their design process and the challenges they faced.

The course is taught in a classroom setting, but it is also designed to be interactive. The students are encouraged to ask questions and to participate in discussions. They are also encouraged to think about how their designs could be improved. Once the students have completed their projects, they are required to present them to a panel of judges. The judges are experts in the field, and they provide feedback on the students' work.

Consolvo believes that the course is important because it helps students understand the real-world implications of their work. He says that it is important for students to learn how to design user interfaces that are effective and easy to use. He also believes that the course is important because it helps students understand how their work can be used to solve real-world problems. Consolvo is proud of the work that his students have done, and he is looking forward to seeing what they will do in the future.
QUANTUM COMPUTING

In the early 1990s, researchers such as Peter Shor and Richard Feynman began to explore the possibility of using quantum mechanics to build a new type of computer. These computers, called quantum computers, could potentially solve certain problems much faster than classical computers.

The extended Church-Turing thesis, developed in the interwar years, due in large part to critical input on the fringe of computer science. In the intervening years, due in large part to critical input on the fringe of computer science.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—still a relatively new and poorly understood field—was not a subject of interest outside of computer science. In the intervening years, due in large part to critical input on the fringe of computer science.

Eric Brewer and Randy Katz lead the BARWAN project to develop Client-Proxy-Server Architectures and Wireless Transport Protocols.

GiST, a Vertica-based search tool, is integrated into the Oracle database engine.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.

When Umesh Vazirani became interested in quantum computation just over a decade ago, the field—it might be a field—a field—focused on the field of computer science. In the intervening years, due in large part to critical input from researchers and other researchers, there has been a recent surge in interest in quantum computing as a field.
Cool water down to 0 degrees Celsius, and it converts to ice. Heat up a homogeneous magnet, and its magnetization disappears. Cool a normal metal, and it suddenly becomes a semiconductor. At first, such phase transitions may look like a problem for physicists, but Professor Alistair Sinclair belongs to a small band of computer scientists who see a connection between these physical phenomena and computation. Like many computational problems, such as computing the optimal route for a road trip, phase transitions challenge a computer in much the same way. At high temperatures, any configuration is as likely as any other, since the orientation of an atom isn’t influenced much by its neighbors. At low temperatures, the interactions dominate, and at low temperatures this interaction is so strong that the magnet as a whole tends to take on the orientation of their neighbors. The intuition behind this is fairly straightforward. The link between phase transitions and mixing times opens up an exciting spectrum of possibilities. This promises to have far-reaching consequences in both fields.

For card shuffling, researchers showed about a decade ago that it takes approximately seven shuffles to reach a “typical” configuration. For long, complex models, it’s more complicated. The temperature that the magnet undergoes a phase transition from exponential to polynomially slow (with respect to the number of magnets) is about as likely as any other, since the orientation of a magnet isn’t influenced much by its neighbors. Somewhere in the middle of the temperature scale is a point where the magnet undergoes a phase transition from exponential to polynomially slow. The intuition behind this is fairly straightforward. The link between phase transitions and mixing times opens up an exciting spectrum of possibilities. This promises to have far-reaching consequences in both fields.

For card shuffling, researchers showed about a decade ago that it takes approximately seven shuffles to reach a “typical” configuration. For long, complex models, it’s more complicated. The temperature that the magnet undergoes a phase transition from exponential to polynomially slow (with respect to the number of magnets) is about as likely as any other, since the orientation of a magnet isn’t influenced much by its neighbors. Somewhere in the middle of the temperature scale is a point where the magnet undergoes a phase transition from exponential to polynomially slow. The intuition behind this is fairly straightforward. The link between phase transitions and mixing times opens up an exciting spectrum of possibilities. This promises to have far-reaching consequences in both fields.
To Christine Papadimitriou, the Internet stands apart from the other creations of computer science. For no individual has designed it; instead, it is the outcome of the busy, competent parleys of thousands of individuals and companies. “The Internet is the first completely artificial artifact that we must approach with humility, the same way that physicists approach the Universe, or biologists approach the cell,” Papadimitriou says.

Because the Internet is a product of the interactions—sometimes competitive and sometimes cooperative—of many agents, Papadimitriou thinks that Berkeley colleagues Karp and Scott Stecher are investigating it using the tools of game theory. Mathematicians study strategic behavior in competitive situations. One of game theory’s main conceptual contributions is that every interaction has an Nash equilibrium, which is a strategy for playing all the available games simultaneously. Papadimitriou says that an Internet equilibrium is about what he calls a “strategic equilibrium.”

Other questions abound. Why, for instance, are the connections on the Internet so well connected? Does something about the value of a connection, perhaps which machines are connected to the Internet, affect the value of a link? For this reason, the science of the Internet continues to be an active field of study.

Papadimitriou says his research continues to be useful because of the way the Internet is structured. “A lot of the power of the Internet is in its ability to scale to a vast number of users, and being able to do that in a cost-effective manner has been a lasting legacy,” he says.

In the late 1980s, Berkeley made a second contribution to the fabric of the Internet. As the ARPAnet grew, it was getting so congested that it could barely function. Jon Postel, now director of ICANN, was 所谓的“三层协议”，即TCP/IP，它使得这些计算机能够通过互联网进行通信。TCP/IP的发明者是Robert Carver and Vint Cerf，他们是ARPAnet的最早期的开发者。TCP/IP在1983年被正式采用，成为互联网的默认协议。TCP/IP的出现使得全球范围内的计算机能够通过互联网进行通信。TCP/IP是通过IP协议传输数据，并由TCP协议控制数据的传输。

Berkeley's open source distribution allowed users to adapt the protocols to suit their needs. “It was an important part of the idea of sharing the code among a competitive jostlings of thousands of individuals in an anarchic nature of the Internet,” he says. “As the Internet is getting more congested, we're getting results that seem to support socially conscious routing. More recently, researchers have started to develop a broad range of networks, the cost of anarchy is a factor of two. If we're getting results that seem to suggest that there are two different strategies of delivering packets, then maybe packets are in a new equilibrium.”

Papadimitriou says that the questions are too broad to be tackled in a full equilibrium, but Papadimitriou and other researchers are making headway on other Internet-related topics. One such problem is understanding the cost of the society’s reliance on the Internet. For instance, Quantum is a protocol for sending packets through the Internet that is designed to be secure. It allows users to establish a shared key that can be used to encrypt data. The protocol is based on the principles of quantum mechanics, which makes it theoretically impossible to break. However, the protocol is still being developed and is not yet widely used.

Papadimitriou says that the Internet is an example of how the Internet has become so complex that it is difficult to understand. “The Internet is the first completely artificial artifact that we must approach with humility, the same way that physicists approach the Universe, or biologists approach the cell,” Papadimitriou says.

Because the Internet is a product of the interactions—sometimes competitive and sometimes cooperative—of many agents, Papadimitriou thinks that Berkeley colleagues Karp and Scott Stecher are investigating it using the tools of game theory. One of game theory’s main conceptual contributions is that every interaction has an Nash equilibrium, which is a strategy for playing all the available games simultaneously. Papadimitriou says that an Internet equilibrium is about what he calls a “strategic equilibrium.”

Other questions abound. Why, for instance, are the connections on the Internet so well connected? Does something about the value of a connection, perhaps which machines are connected to the Internet, affect the value of a link? For this reason, the science of the Internet continues to be an active field of study.

Papadimitriou says his research continues to be useful because of the way the Internet is structured. “A lot of the power of the Internet is in its ability to scale to a vast number of users, and being able to do that in a cost-effective manner has been a lasting legacy,” he says.
George Necula is figuring out how to let people download and use untrusted code without fear. Necula envisions a system in which manufacturers prove to users that their code does what they claim—that it won't start digging into private files, for example, or read copies of itself over the Internet.

At first, it might seem that any explanation of how the code works would be huge—many times larger than the software itself. This was indeed the case early implementations of Necula’s “Proof-carrying code” (PCC), which he developed as a graduate student with his advisor Peter Lee at Carnegie Mellon University. On average, the proofs were three times larger than the software they explained. “People were joking, you don’t have proof-carrying code—you have code-carrying proofs,” Necula says.

Since coming to Berkeley, Necula has reappraised his approach to writing proofs and, astonishingly, arrived at the resulting explanations to about 10% of the size of the software. In the early versions of PCC, the proof wrote out, in gray text, all the safety conditions that had to be tested; “proof-checking” software on the user’s end then verified the code satisfied those conditions. Necula has since hit upon a more concise approach, in which the proof helps the user’s proof-checker only the most challenging hurdles.

The shortened proof is like a guide through a labyrinth, Necula says. Following a path through a maze can be hard even with a direct route, and what about code from the shadowy regions of the Internet—programs that are not so reliable?

This approach, the proof-checking code is only the first step on the road to Necula’s ultimate goal—an open virtual machine that sits on a user’s hard-drive and can check, verify, and run code sent to it in any language. Currently, software sent to a virtual machine must be written in Java or Microsoft Common Language Infrastructure. On the open virtual machine Necula envisions, users would first upload a verifier tailored to a specific language. The machine would then be able to accept and verify code in that language.

The challenge, Necula says, is to figure out how to verify the verifier. “How can I have a guarantee that the verification done with the verifier you just sent me is correct?” he says. “Maybe you sent me a verifier that always says yes, yes, yes—a verifier that always says, ‘This is all right! How can I prevent that from happening? ’ That’s the next thing.” Necula says. “That’s what we’re trying to develop right now.”

YOUNG FACULTY

Like many of Berkeley’s young faculty, George Necula came to Berkeley directly from graduate school. Originally leaning toward a career in industry, he was lured into academia by the high caliber of his future colleagues and the Berkeley atmosphere. “It’s a very friendly place,” he says. “The computer science division makes a point of holding only as many meetings as people who people in its type to hate,” says Division Chair Clifton Maloney. “This makes for a welcoming environment and encourages collaboration.” Necula says. “Then, you don’t feel as if you’re competing against the others for limited resources or bandwidth,” he says. “We’re all competing against the outside world, not with each other.”

DOWNLOADED HAS BECOME A NATIONAL PASTIME.
FROM PROGRAMMERS WHO ARE NOT SO RELIABLE?
WILLIAM KAHAN WINS THE IEEE EMANUEL R. LOFTI ZADEH WINS THE ACM ALLEN NEWELL KARMARKAR wins the ACM Paris Kanellakis Theory Award for his development of a highly efficient algorithm for linear programming.
JOHN KUBIATOWICZ WINS AN NSF PECASE CAREER AWARD, the highest award for young academic career promise.
IAN STOICA WINS ACM DISSERTATION AWARD for his work at CMU on “Stateless Core: A Scalable efficient algorithm for virtual machine that sits on a user’s hard-drive and can check, verify, and run code sent to it in any language. Currently, software sent to a virtual machine must be written in Java or Microsoft Common Language Infrastructure. On the open virtual machine Necula envisions, users would first upload a verifier tailored to a specific language. The machine would then be able to accept and verify code in that language.

The challenge, Necula says, is to figure out how to verify the verifier. “How can I have a guarantee that the verification done with the verifier you just sent me is correct?” he says. “Maybe you sent me a verifier that always says yes, yes, yes—a verifier that always says, ‘This is all right! How can I prevent that from happening? ’ That’s the next thing.” Necula says. “That’s what we’re trying to develop right now.”

YOUNG FACULTY
Like many of Berkeley’s young faculty, George Necula came to Berkeley directly from graduate school. Originally leaning toward a career in industry, he was lured into academia by the high caliber of his future colleagues and the Berkeley atmosphere. “It’s a very friendly place,” he says. “The computer science division makes a point of holding only as many meetings as people who people in its type to hate,” says Division Chair Clifton Maloney. “This makes for a welcoming environment and encourages collaboration.” Necula says. “Then, you don’t feel as if you’re competing against the others for limited resources or bandwidth,” he says. “We’re all competing against the outside world, not with each other.”

DOWNLOADED HAS BECOME A NATIONAL PASTIME.
FROM PROGRAMMERS WHO ARE NOT SO RELIABLE?
WILLIAM KAHAN WINS THE IEEE EMANUEL R. LOFTI ZADEH WINS THE ACM ALLEN NEWELL KARMARKAR wins the ACM Paris Kanellakis Theory Award for his development of a highly efficient algorithm for linear programming.
JOHN KUBIATOWICZ WINS AN NSF PECASE CAREER AWARD, the highest award for young academic career promise.
IAN STOICA WINS ACM DISSERTATION AWARD for his work at CMU on “Stateless Core: A Scalable efficient algorithm for linear programming.
JOHN KUBIATOWICZ WINS AN NSF PECASE CAREER AWARD, the highest award for young academic career promise.
IAN STOICA WINS ACM DISSERTATION AWARD for his work at CMU on “Stateless Core: A Scalable efficient algorithm for linear programming.

COMPLEXITY THEORY

BY THE MID 1960s, computer scientists were reaching a consensus about what makes a problem tractable: it should be solvable in an amount of time that grows only polynomially with the size of the input (such a problem is said to be in the class \( P \)). Although many problems were shown to be in \( P \), many other natural problems appeared to lie in a broader class called \( NP \). A problem in \( NP \) might, or might not, be solvable in polynomial time, but given a solution, it’s possible to check in polynomial time whether that solution is valid. All problems in \( P \) are automatically in \( NP \), but not vice versa, scientists believe. The hardest problems in \( NP \) are thought to be impossible to solve in polynomial time.

At first, it wasn’t clear whether there was any property that linked the hard problems in \( NP \). But in 1971, Stephen Cook of the University of Toronto proved that a problem called SAT was \( NP \)-complete. It could be solved in polynomial time, then so could every hard problem in \( NP \) (something computer scientists believe was until recently). At about the same time, Leonid Levin of Boston University (at the time Soviet Union) proved the same result independently.

On reading Cook’s paper, Berkeley Professor Richard Karp quickly realized that a host of important problems have the same universality property as SAT. Karp’s work in operations research—on problems such as the traveling salesman problem and the design of digital circuits—had given him an instinct for what makes problems particularly hard. “My work had revolved around the design of algorithms, so I was very familiar with these combinatorial explosions,” he said. “I was pretty cognizant of the sentiment that problems that people should talk about.” Karp, who in 1985 was awarded the ACM A.M. Turing Award, one of computer science’s highest honors, chiefly for his work on \( NP \) completeness. “I think it was important to have one foot in the theory camp and one foot in the applied camp.”

In 1972, Karp showed that 21 fundamental problems were \( NP \)-complete. Other computer scientists quickly jumped on the bandwagon, using Karp’s methods to show that thousands of problems are \( NP \)-complete. “It was like the St. Paul who popularized the Christian concept and gave it to the masses,” Papadimitriou says. The notion of \( NP \)-completeness appears to capture a remarkable dichotomy between hard and easy problems: nearly every natural problem is either \( NP \)-complete in \( P \).’ “This was one of those lucky moments in science when a particular theoretical concept had very wide applicability and could help classify practically everything,” Papadimitriou adds.

Finding out the notion of \( NP \)-completeness made computer scientists realize that for many problems, it is best to just try to do the best you can. “If you think about it,” Papadimitriou says, “you might end up with an exact answer efficiently. Researchers wondered whether they would fare better trying in first approximate answers. Amazingly, in the early 1990s, Berkeley graduate student Mika Sodini and Sanjeev Arora showed, together with several other Berkeley and other computer scientists, that for some problems, finding even a reasonably good approximate answer is as hard as solving \( NP \)-complete problems exactly. The researchers’ work landed in the theoretical computer science community like a “huge rock in the middle of the lake,” recalls Papadimitriou. “There were waves everywhere.”

Sudan and Arora were each awarded the ACM Doctoral Dissertation Award, the most prestigious award for a Ph.D. student.

It might seem that uncovering so many problems computers can’t handle, the results of Karp, Sudan, Arora and others have been depressingly negative. But the proof of a problem is irrefutable—aka, points researchers away from fruitless endeavors towards more promising directions. “If you can’t solve \( NP \)-complete problems, maybe we have to invent new ways to solve them,” says Papadimitriou.

Papadimitriou says. “Showing something is \( NP \)-complete isn’t a way to kill problems, but to create the right tools.” If computer scientists have to lower their sights about what computers can do, the resulting vision has been so keen that it has left no stone unturned.

Even before the first computers started appearing, Alan Turing proved in 1936 that there are inherent limitations of computers, with some of the defining contributions coming from Berkeley researchers.

By Richard Karp via Scientific American

Professor Karp is the Donald Bren Professor of Computer Science, Operations Research, and Mathematics at UC Berkeley. He has been recognized as a leading young scientist by Scientific American and was included in its list of the 50 most influential people in the world of theoretical computer science thirty years ago.

Even before the first computers started appearing, Alan Turing proved in 1936 that there are inherent limitations of computers, with some of the defining contributions coming from Berkeley researchers.

By Richard Karp via Scientific American

Professor Karp is the Donald Bren Professor of Computer Science, Operations Research, and Mathematics at UC Berkeley. He has been recognized as a leading young scientist by Scientific American and was included in its list of the 50 most influential people in the world of theoretical computer science thirty years ago.
1. LEN ADLEMAN, A STUDENT OF MANUEL BLUM, shares the Turing Award with Ronald Rivest and Adi Shamir for their work on encryption.

2. JONATHAN SHEWCHUK WINS THE WILKINSON PRIZE for his outstanding contributions to the field of numerical software and computational science.

3. EUGENE MYERS JOINS THE CS DIVISION from his position as Chief Computer Scientist at Celera Genomics. He developed the algorithmic breakthroughs that enabled the compilation of the human genome. He won the ACM Paris Kanellakis Theory Award in 2001 for this work.

4. PETER BARTLETT JOINS THE CS DIVISION with a joint appointment with Statistics. In 1997 he developed the first satisfactory theoretical explanation of how neural networks learn.

5. NINE NEW FACULTY MEMBERS HAVE JOINED THE CS DIVISION thus far in the decade of the 00s. The size of the CS Division faculty reaches 43.49 FTE, the largest since the Division was founded in 1973.

6. CHRISTOS PAPADIMITRIOU publishes Turing (a novel about Computation).

"HE WHO RECEIVES AN IDEA FROM ME, RECEIVES INSTRUCTION HIMSELF WITHOUT LESSENING MINE; AS HE WHO LIGHTS HIS TAPER AT MINE, RECEIVES LIGHT WITHOUT DARKENING ME."

— Thomas Jefferson, on the importance of sharing knowledge

THIS BOOK IS DEDICATED TO THE MANY INDIVIDUALS AND THEIR COLLECTIVE ACCOMPLISHMENTS—too numerous to mention—that have been affiliated with the Computer Science Division at Berkeley over the last thirty years. We could not have built this outstanding educational and research computer science institution without their contributions.

EDITOR-IN-CHIEF: Randy Katz
EDITORIAL BOARD: Alex Aiken, James Landay, Jitendra Malik, James O’Brien, David Patterson, Stuart Russell, Alistair Sinclair
EDITORIAL ASSISTANT: Kate Riley
EDITORIAL ASSOCIATE: Debra Zaller
SCIENCE WRITERS: Katie Greene, Erica Klarreich
PHOTOGRAPHER: Peg Skorpinski
BOOK DESIGN: Lisa Rounds, Shelly Designs & Illustrates