Bacterial Computing:

Using *E. coli* to Solve the Burnt Pancake Problem

Laurie J. Heyer, Jeffrey L. Poet, Marian L. Broderick, Phillip E. C. Compeau, James O. Dickson, and W. Lance Harden

Living Computers

Imagine a computer so small that a billion of them fit in a Petri dish. Now imagine that same computer replicating itself every 20 minutes. It sounds like science fiction, but such microscopic machines, in the form of genetically engineered *E. coli* cells, are being designed and built to solve interesting mathematical problems in a new field called *synthetic biology*. Synthetic biology seeks to use engineering principles, mathematical modeling, and molecular biology techniques to design and construct novel biological devices with applications in medicine, the environment, energy, and various other fields. Undergraduate students in biology, engineering, computer science, mathematics, physics, and chemistry are major players in synthetic biology thanks to an annual competition (affectionately called a Jamboree) at MIT. Successful projects at the international Genetically Engineered Machines (iGEM) Jamboree mix creativity with logic, lab work with mathematical modeling, and utility with whimsy. Here is the story of one such project: a living computer to solve a version of the Pancake Problem, first introduced in the *American Mathematical Monthly* by a "harried waiter":

> The chef in our place is sloppy, and when he prepares a stack of pancakes they come out all different sizes. Therefore, when I deliver them to a customer, on the way to the table I rearrange them (so that the smallest winds up on top, and so on, down to the largest on the bottom) by grabbing several from the top and flipping them over, repeating this (varying the number I flip) as many times as necessary. If there are n pancakes, what is the maximum number of flips (as a function of n) that I will ever have to use to rearrange them?
> *American Mathematical Monthly*, 1975, 10, p.101.
> Proposed by Harry Dweighter (a.k.a. Jacob Goodman, The City College of the City University of New York).

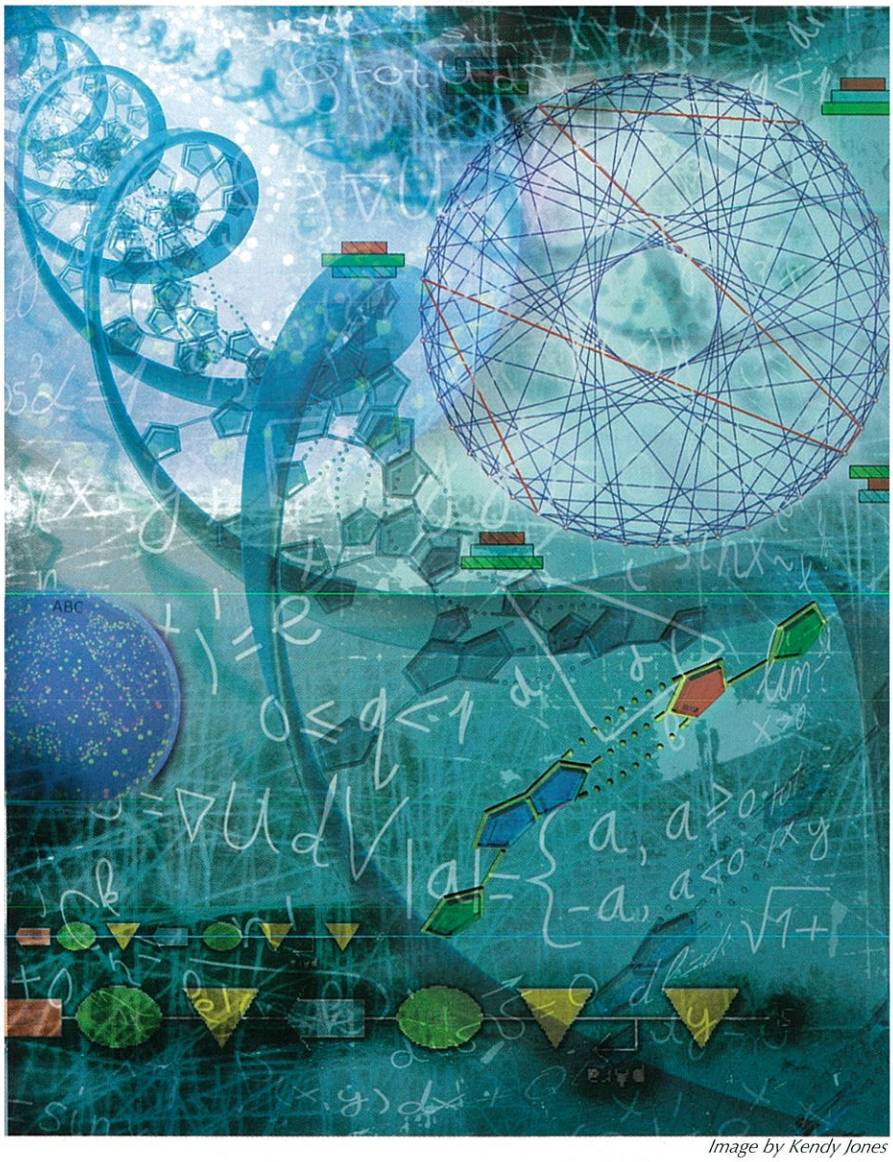
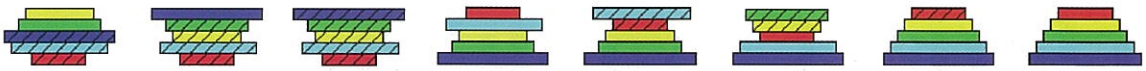


Image by Kendy Jones

The original pancake problem allowed only prefix reversals, corresponding to flips of the top portion of the stack with a single spatula. Variations of the Pancake Problem have since been studied, including the Burnt Pancake Problem (BPP), introduced in 1979 by Bill Gates (yes, that Bill Gates!) and his undergraduate research mentor in what turned out to be Gates's only academic paper [3]. In the BPP, each of the pan-



(2,3,-5,-4,-1) → (5,-3,-2,-4,-1) → (-5,-3,-2,-4,-1) → (1,4,2,3,5) → (-4,-1,2,3,5) → (-3,-2,1,4,5) → (-1,2,3,4,5) → (1,2,3,4,5)

Figure 1a. Sequence of flips when following brute force algorithm, requiring 7 flips. Hashed pancakes are burnt side up; solid pancakes are burnt side down.

cakes is burnt on one side, and in addition to sorting the pancakes by size (with smallest on top and largest on bottom) each pancake must also be oriented burnt side down (so that the customer cannot see that it is burnt). Another variation of the Pancake Problem allows the waiter to flip pancakes anywhere in the stack while leaving the top of the stack undisturbed. These flips are called *reversals*, and represent the use of two spatulas (the first spatula lifts away the top portion of the stack, while the second flips a set of adjacent pancakes underneath, then the top portion is set back down as it was). The ultimate goal of any version of the Pancake Problem is to determine which stack of n pancakes requires the most reversals to sort. As a first step in this direction, one might start with an arbitrary stack of n pancakes and determine the minimum number of flips to sort this one stack. *Try out zip-line problem 242 on page 30 to try your hand at this*.

(2,3,-5,-4,-1) → (4,5,-3,-2,-1) → (-5,-4,-3,-2,-1) → (1,2,3,4,5)

Figure 1b. An optimal sequence using 3 flips.

Mathematically, the BPP corresponds to sorting *signed permutations*. Like a permutation, a signed permutation can be thought of as an ordered n-tuple indicating position *(where 1 represents the smallest pancake, 2 the second smallest, etc.)*, with the additional feature of recording the orientation of each element: a negative sign indicates that a pancake is burnt side up. A reversal changes both the order and orientation of a set of adjacent pancakes, and a stack of burnt pancakes is sorted when the resulting signed permutation is the identity (1, 2, …, n).

For the one-spatua BPP, it is relatively straightforward to see that the minimum number of flips for any given starting stack is no greater than $3(n-1)+1$. For any stack, it takes at most three flips to *(i)* move the largest pancake to the top of the stack, *(ii)* flip it over to be burnt side up if necessary, and *(iii)* invert the *entire* stack to put the largest pancake on the bottom in proper orientation. This process can be repeated for each of the pancakes from the second largest to the second smallest, at which point the smallest pancake is on top, and can be inverted if necessary. This *"brute force"* algorithm is illustrated in Figure 1a on the beginning stack corresponding to (2, 3, –5, –4, –1) and gives an upper bound on the number of flips necessary.

However, this naïve upper bound is not very tight. For example, Figure 1b shows a way to sort the same stack as in Figure 1a in just three flips. Cohen and Blum [4] found the best upper bound to be $2n-2$, and researchers continue to work to improve both lower and upper bounds on the number of flips required to sort the most difficult stack.

By building bacterial computers to attack the BPP, we were not trying to solve a problem that could not be solved by conventional computers. In fact, there is an efficient graph theoretical algorithm for finding an optimal sequence of steps to sort a stack of burnt pancakes, and the interested

From DNA Computing to Living Computers

DNA computing was introduced by Leonard Adleman in 1994 [6]. He was able to get segments of DNA to assemble themselves, through a series of test tube experiments, to find the unique Hamiltonian path (a sequence of directed edges that visit each node exactly once) in the directed graph shown in Figure 2. What's new about the synthetic biology approach is that living cells do the computations with no human intervention. However, designing a living computer that behaves as expected is a tremendous challenge. The synthetic biology design process is like studying a huge pile of parts and deciding whether to build a nuclear reactor or a washing machine. In this case, the pile of parts is the existing cellular machinery in any number of compatible organisms, and the list of "machines" that could be built is nearly infinite.

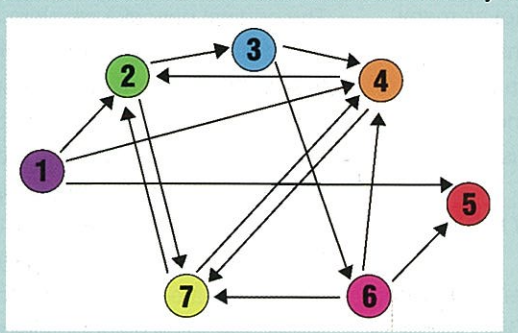


Figure 2. The Adleman Graph

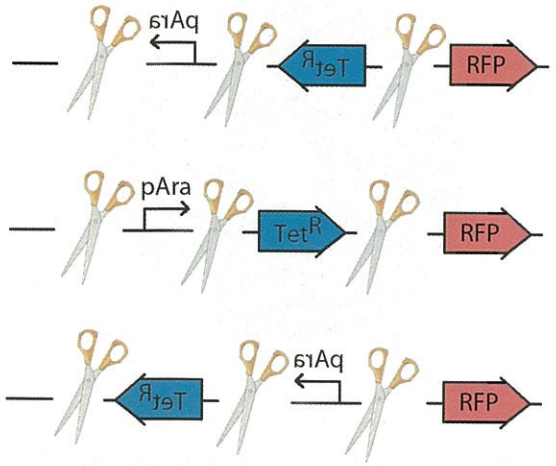


Figure 3. Biological construction of (a) an unsorted stack (–1, –2), (b) the sorted stack (1,2), and (c) the unsorted stack (–2, –1). Scissors represent hix sites. The Hin protein can cut the DNA at a pair of these hix sites, and reverse the pancake(s) between them. TetR produces resistance to the antibiotic tetracycline when preceded by the pAra promoter in the same orientation. Reverse lettering indicates that the corresponding DNA segment is in reverse orientation, and will be read from right to left.

reader is referred to [5] to learn more about this algorithm that manipulates so-called Reality and Desire diagrams. We also were not the first to think of using DNA to perform computations. Leonard Adleman (the "A" in RSA cryptography) introduced DNA computing in 1994 [6] when he used inherent base-pairing properties of DNA to self-assemble segments of DNA *in vitro* (in a test tube) to solve a 7-node Hamiltonian Path Problem (see *the* Side Panel *"From DNA Computing to Living Computers" on page 6*). This was the subject of a 1995 Math Horizons article [7] by Keith Devlin. Rather, our goal was to demonstrate that living cells could be used to solve the same problems as conventional computers. The next section describes how we solved a two-spatula, two-pancake BPP *in vivo*, using *E. coli* as our living computers.

Biological Implementation

DNA is the basic building block of organisms, a macromolecule that contains all the information needed for each cell to function as a part of the organism as a whole. The entire DNA of an organism (its genome) can be thought of simply as a very long string of letters from a four-letter alphabet: A, C, G and T. Biologists call these strings *DNA sequences*. Some DNA sequences form functional units. For example, to make a protein, two functional units (a promoter and a protein-coding sequence) must appear in that order. DNA is double stranded and each strand can only be read in one direction – known in biological terms as 5' to 3' (read "5 prime to 3 prime"). The natural order and orientation of DNA sequences form the basis for our biological implementation of a BPP.

The biologists on our iGEM team spent many weeks piecing together and testing the DNA parts required to implement this approach. The side panel *"How Do You Build Biological Parts" on this page* outlines the methods they used to build basic parts and combine parts into functional devices. Our approach to solving the two-spatula BPP in living cells required three feats of genetic engineering:

1. Encode a signed permutation within the DNA of an E. coli bacterium. A "pan-

How Do You Build Biological Parts?

In synthetic biology, a *part* is a DNA sequence with a particular function. Examples include protein-coding genes like red fluorescent protein and tetracycline resistance, promoters like pAra, and short motifs like *hix* sites. Parts are typically put into cells by first placing them on a *plasmid*, a circular piece of double-stranded DNA that can be maintained inside *E. coli*, essentially a mini-chromosome.

The Registry of Standard Biological Parts, housed at MIT, contains several thousand parts. These parts, called BioBricks™, are interchangeable with each other through a system of *restriction sites*, which allow DNA sequences to be cut in particular locations and two sequences to be spliced together. The restriction site system makes it straightforward to cut part A out of one plasmid and *ligate* (attach) it in front of or behind part B in another plasmid. Our two-pancake constructs, shown in Figure 3, were built with this method, successively ligating *hix* site, promoter, *hix* site, TetR coding sequence, *hix* site, and RFP coding sequence.

Before we could assemble the two-pancake constructs, we had to build the basic parts. A big surprise for most non-biologists is that DNA sequences can be custom ordered from a company for about $0.50 per letter. This is the easiest, but certainly not the cheapest, way to make a DNA part. Sequences that exist in other organisms, for example the Hin gene in *Salmonella*, can be obtained through the polymerase chain reaction (PCR), a standard method for making millions of copies of any sequence for which the letters comprising the beginning and end of the sequence are known.

Although all these methods are conceptually simple, each step of part construction and assembly must be verified, and often repeated multiple times. Life seems to have a way of resisting change, and *E. coli* often spontaneously mutates to partially or completely avoid the DNA we want to give it.

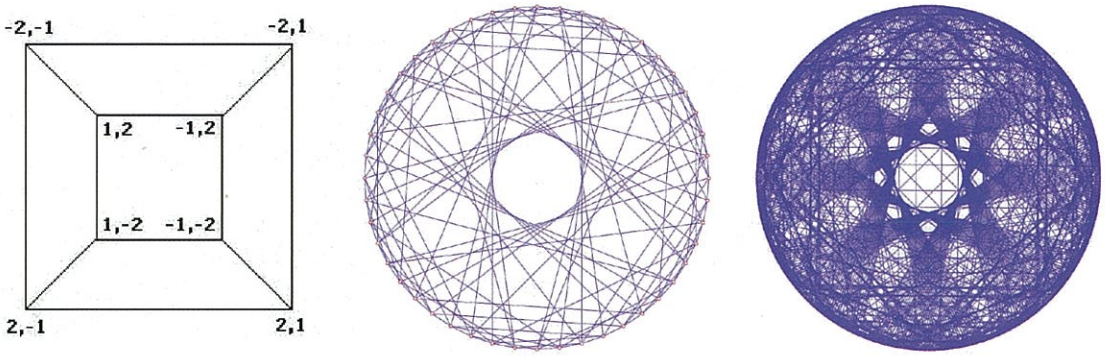


Figure 4. Burnt pancake graphs for 2, 3, and 4 pancakes.

cake" is represented by a particular sequence of DNA, which can be inserted into a population of *E. coli* cells. As a proof of principle, we constructed a two-pancake problem. Because of the natural 5' to 3' directionality of DNA, it is easy to represent a pancake as burnt side up or burnt side down by inserting the pancakes in the desired orientation.

2. Endow the bacterium with the ability to flip particular segments of its DNA. The flipping function is achieved by transferring into *E. coli* a gene that already exists naturally in *Salmonella typhimurium*. In *Salmonella*, this gene has the ability to invert a particular segment of DNA to change the characteristics of the cell's flagella. The protein made by this gene, called Hin, binds to a pair of specific DNA sequences, called *hix* sites, breaks the DNA strand in both places, reverses the orientation of the DNA between the *hix* sites, and reattaches the strand to keep the DNA intact. Therefore, any segment of DNA flanked by two *hix* sites can be inverted when the Hin protein is present. We encoded a signed permutation by concatenating two DNA pancakes with the sequence for *hix* both before and after each pancake. Then we added Hin to allow the pancakes to be reversed.

3. Signal the presence of the identity permutation. We designed our bacterium to be resistant to tetracycline only when the pancakes are sorted in the correct order and orientation. Thus, the bacterium itself can tell us when the problem has been solved, like a traditional computer might make a loud "ding" when it has finished a long computation. In our case, the only cells that can survive in a Petri dish laced with tetracycline are those that have successfully solved the puzzle. Cells that have not found the solution will die, a severe penalty for not solving a problem!

Figure 3a shows a schematic of our biological implementation of the signed permutation (–1, –2), where pancake 1 is the promoter pAra, and pancake 2 is the gene for tetracycline resistance. Figure 3b shows a schematic of the sorted stack, corresponding to a solution to the BPP, which confers tetracycline resistance to the bacterium. Note that the DNA downstream of the last hix site encodes a red fluorescent protein (RFP), which will only be expressed when the pAra promoter is pointing toward it. Tetracyline resistance could also be produced by (–2, –1), as shown in Figure 3c, but only the true solution will be both antibiotic resistant and glowing red. We built and tested all possible stacks of two burnt pancakes.

Traditional computers work serially, completing one step before moving to the next, but a biological approach to a mathematical problem like the BPP applies massive parallel processing to the problem by having millions of bacteria work simultaneously but independently on the given problem. Utilizing the reproductive capabilities of bacteria, we can encode the problem in a single *E. coli* cell and have tens of millions of genetically identical bacteria within 24 hours. When these bacteria are supplied with the Hin protein, we have tens of millions of independent opportunities to solve the problem. Using standard wetlab techniques, we can screen for bacteria that have found the solution by exposing them to the antibiotic. Those bacteria that have found the forward (1, 2) or backward (–2, –1) solution will survive and all others will die. Those bacteria that have found the forward solution (the true solution) will additionally glow red when observed under ultraviolet light. By dividing the time required to reach a solution by the time it takes to do a single flip, we can determine the number of flips required to sort the stack.

Mathematical Modeling of the Biological Computer

While the biological part of the team worked in the lab to build first one pancake, then another, and put them together with the Hin "spatula," the mathematical part of the team worked on predicting the behavior of our biological computer. We asked the mathematical question "Given an arbitrary stack of burnt pancakes subjected to the random flipping

process of our biological computer, what is the probability that the stack is sorted after k reversals?" To answer this question, we modeled the random flipping process as a Markov Chain in which the states are the $n!2^n$ possible signed permutations. Because there are $n + 1$ *hix* sites from which to select the beginning and ending of a particular reversal, there are

$$\binom{n+1}{2} = \frac{n(n+1)}{2}$$

ways to perform a reversal on a given signed permutation. We assume that all reversals are equally likely (a big modeling assumption that we are still trying to verify biologically). Therefore, the transition probabilities in our Markov Chain are

$$\frac{2}{n(n+1)}$$

for all states that can be reached with a single reversal from the current state, and 0 for all other states. Because this Markov Chain is ergodic, the biological computer is equally likely to be in any one of the possible signed permutations after an extended period of time. The k^{th} order transition probabilities tell us how likely the stack is to be sorted after k reversals. The minimum number of reversals required to sort the stack is given by the smallest k for which the k^{th} order transition probability from the given state to the identity state is non-zero.

One way to visualize the Markov Chain model is as a random walk on a *burnt pancake graph*. The nodes of the burnt pancake graph are the states of the chain (all possible signed permutations) and an edge connects nodes i and j if the signed permutation at node i can be reached from the signed permutation at node j (and vice versa) with a single reversal. There are $n!2^n$ nodes in the burnt pancake graph for n pancakes, and each node of the graph has degree $n(n + 1)/2$. The regularity of these graphs leads to some beautiful symmetries as illustrated in Figure 4. Observe the n-fold symmetry and $2n$-gons in each diagram. The construction and properties of these graphs are described in [10]. Our biological computer is trying to find the shortest path from the given stack to the sorted stack. Note that the solution to the original pancake problem is the diameter of the pancake graph.

The collaborative iGEM team that worked on the Burnt Pancake Problem worked full time all summer on two different campuses, and met for the first time in November at the Jamboree at MIT. Student researchers from Missouri Western State University, Davidson College, Hampton University and St. Joseph's Central High School are shown here with faculty and postdoctoral mentors and with the four trophies they received at the competition. Thirty-eight teams from around the world presented their work at the iGEM Jamboree that year. More than 100 teams participated in 2009.

There are many challenges facing this biological computing system. First, flipping continues until we physically remove the Hin from the cell. As a consequence, a particular bacterium might achieve the solution but flip again (out of the solution) before we do our antibiotic screening. That is, the solution state is not an absorbing state of the Markov chain. Second, flips do not occur in strict time intervals. The biological reality introduces variability into the problem. While standard Markov chains can be used to model the problem, they are only approximating reality insofar as we have no standard "step." The Hin/*hix* system has only recently been introduced into *E. coli* and there is still much to discover about the mechanism and kinetics of flipping. Still, a mathematical model in which we assume ideal steps and a uniformly random distribution of flips is a good starting point for understanding the system.

Synthetic Biology and the Future of Computing

After a full summer of research, the entire iGEM team traveled to MIT in November to present our work at the Jamboree. At that point, we were only able to demonstrate flipping of a single DNA pancake in *E. coli*, though we had built a stack of two DNA pancakes, and modeled the behavior of much bigger stacks. Our project made a big splash. We had built a working computer in living cells, a feat that no other team had attempted. Teamwork, across disciplines and between institutions, was key to our success. We continued to work on our project after the Jamboree, until we could demonstrate flipping of two pancakes, both individually and the whole stack. Eventually, we published our work in

the *Journal of Biological Engineering* [2], and this article made waves of its own in the popular press, landing the first author, Karmella Haynes, on NPR's *Science Friday* with Ira Flatow.

The field of synthetic biology is full of opportunities for undergraduates from many disciplines, including the mathematical sciences, to work as part of interdisciplinary teams. The interested reader is encouraged to visit the iGEM website [1] to learn more about the breadth and variety of undergraduate research in the field of synthetic biology. We have used iGEM as a springboard to investigate living, biological computers that launch new attacks on some old problems, including the Hamiltonian Path Problem [11], cryptographic hash functions [12], and Boolean satisfiability problems [13].

The natural exponential growth of bacterial cells and the ability to replicate information within those cells, gives hope that a biological computer may someday be able to tackle some of the mathematical giants.

Acknowledgements

Collaboration with biologists is essential in this highly interdisciplinary field, and we are grateful for our biology colleagues, Malcolm Campbell (Davidson College) and Todd Eckdahl (Missouri Western State University). Karmella Haynes performed many of the wetlab experiments to confirm our solution to the BPP while she was a teaching postdoctoral fellow at Davidson College. We appreciate the efforts of our many student researchers and the continuing support from our institutions. We are grateful for the support of iGEM founders Drew Endy, Tom Knight and Randy Rettburg, and the inspiration we have drawn from many other members of the worldwide iGEM community. The zip-line challenge problem is due to Rob Hochberg of East Carolina University. Finally, we thank the National Science Foundation (grants DMS-0733952 and DMS-0733955) and the Howard Hughes Medical Institute for supporting our efforts to mold and shape the next generation of synthetic biology researchers.

Further Reading

1. International Genetically Engineered Machines (iGEM) website http://2009.igem.org/Main_Page.

2. Haynes, K.A., M.L. Broderick*, A.D. Brown*, T.L. Butner*, J.O. Dickson*, W.L. Harden*, L.H. Heard*, E.L. Jessen*, K.J. Malloy*, B.J. Ogden*, S. Rosemond*, S. Simpson*, E. Zwack*, A.M. Campbell, T.T. Eckdahl, L.J. Heyer, J.L. Poet (2007). Engineering bacteria to solve the burnt pancake problem. *Journal of Biological Engineering* **2:8**. www.jbioleng.org/content/2/1/8.

3. Gates W.H.; Papadimitriou, C.H. Bounds for sorting by prefix reversal. *Discrete Math*. **27** (1979), 47–57.

4. D.S. Cohen and M. Blum, On the problem of sorting burnt pancakes, *Discrete Applied Mathematics*, 61 (1995) 105-120.

5. Jones, N.C. and Pevzner, P.A. *Introduction to Bioinformatics Algorithms*. The MIT Press (2004).

6. Adleman, L.M., "Molecular Computation Of Solutions To Combinatorial Problems." *Science* **266** (1994), 1021–1024.

7. Devlin, Keith, Test tube computing with DNA. *Math Horizons* **2** (April 1995), 14-21.

8. http://grimm.ucsd.edu/GRIMM/index.html.

9. http://www.americanscientist.org/issues/pub/sorting-out-the-genome/1.

10. Compeau, P.E.C. Cycles in Pancake Graphs. Undergraduate honors thesis, Davidson College Mathematics Department.

11. Baumgardner, J.*, K. Acker*, O. Adefuye*, S. T. Crowley*, W. DeLoache*, J. O. Dickson*, L. Heard*, A. T. Martens*, N. Morton*, M. Ritter*, A. Shoecraft*, J. Treece*, M. Unzicker*, A. Valencia*, M. Waters*, A. M. Campbell , L. J. Heyer , J. L. Poet , and T. T. Eckdahl, Solving a Hamiltonian Path Problem with a bacterial computer. *Journal of Biological Engineering* 3:11, (2009), http://www.jbioleng.org/content/3/1/11.

12. http://2008.igem.org/Team:Davidson-Missouri_Western.

13. http://2009.igem.org/Team:MoWestern_Davidson.

About the authors: *Laurie Heyer is an associate professor of mathematics at Davidson College. Jeff Poet is an associate professor of mathematics at Missouri Western State University. Marian Broderick is an undergraduate at Missouri Western, and Phillip Compeau, Jim Dickson, and Lance Harden are undergraduates at Davidson.*

email: laheyer@davidson.edu
email: poet@missouriwestern.edu

DOI: 10.4169/194762110X489242