Sorting Out the Genome

Brian Hayes

All through the 1930s, members of the famous *Drosophila* group at Caltech roamed the American West collecting fruit flies for genetic analysis. One discovery to come from these expeditions was an abundance of genetic inversions: blocks of genes that had flipped end-over-end. Flies from one geographic region might have a certain set of genes in the order *abcdefg*, but a population elsewhere could harbor the sequence *dcbaefg*, with the first four genes inverted. A further reversal, affecting a different block of genes, could produce an ordering such as *dcfeabg*. Theodosius Dobzhansky and Alfred H. Sturtevant, two of the leading Drosophilists, pointed out that such genetic rearrangements could help in reconstructing the family tree of the flies. More reversals would indicate greater evolutionary distance.

The variations discovered by Dobzhansky and Sturtevant could be explained by reversing just one or two blocks of genes. Later, when gene order was studied in a broader range of organisms, more complex patterns emerged. In the 1980s Jeffrey D. Palmer and Laura A. Herbon of the University of Michigan were measuring the pace of evolutionary change in plants of the cabbage family. Looking at the DNA in mitochondria (the energy-producing organelles), they found that the genes had been jumbled by multiple random reversals. Transforming cabbage into turnip took at least three reversals. More distant relatives such as cabbage and mustard appeared to be separated by a dozen or more reversal events—they could only estimate how many.

If these genetic flip-flops are to serve as an evolutionary clock, we need a reliable way to count them. Given two arrangements of a set of genes—say *abcdefg* and *febagcd*—how do you determine what sequence of reversals produced the transformation? This example has a three-step solution, which you can probably find in a few minutes with pencil and paper. For larger genomes and longer chains of reversals, however, trial-and-error methods soon falter. Is there an efficient algorithm for identifying a sequence of reversals that converts one permutation into another?

To put your genes in order, flip them like pancakes

The genetic reversal problem lies at the intersection of biology, mathematics and computer science. For some time, the prospects for finding a simple and efficient solution seemed dim, even with the most powerful tools of all three disciplines. But the story has a happy ending. A little more than a decade ago, computing gene reversals was still a subtle research problem; now it can be done with such ease that it's a matter of routine technology. If you need to know the "reversal distance" between two genomes, you can go to a Web site and get the answer in seconds.

Ordering Breakfast

Before getting tangled up in loopy strands of DNA, consider a warm-up exercise: flipping pancakes. You are given a stack of n pancakes, no two the same diameter, and asked to sort them in order of size, with the smallest on top and the largest on the bottom. At each step in the sorting process, you insert a spatula anywhere you choose within the stack, then flip over all the pancakes above the spatula. No other manipulations of the stack are allowed. How many flips are required to get the pancakes in order?

Here's one pancake-sorting algorithm. First find the largest pancake, put the spatula under it, and turn over the part of the stack above that point. Now the largest pancake is on top, so flipping the entire stack of n pancakes puts it in its proper position at the bottom. From here on, the bottommost pancake will never be moved again. Next, choose the second-largest pancake, make a flip to bring it to the top, and then flip $n-1$ pancakes to place the second-largest in its permanent place atop the largest. After bringing the third-largest to the top, flip $n-2$ pancakes, and so on until the whole stack is sorted. Placing each pancake takes two flips, and so the entire procedure requires $2n$ steps. (Or maybe $2n-1$. Or $2n-3$. Some of the final flips are not really needed.)

This algorithm provides a naive upper bound: It shows that sorting the stack can't require more than about $2n$ steps, but the possibility remains that some other method could do better. In the 1970s such an improved algorithm was devised by Christos H. Papadimitriou, then of Harvard University, and William H. Gates, an undergraduate at that institution (who soon dropped out to start a software company). The Gates-Papadimitriou algorithm sorts any stack of pancakes in $(5n+5)/3$ moves, which for large n is essentially equal to $\frac{5}{3}n$. Gates and Papadimitriou also established a lower bound, showing that some pancake stacks cannot be sorted in fewer than $\frac{17}{16}n$ steps; the lower bound has since been raised slightly to $\frac{15}{14}n$. The range defined by these upper and lower bounds is probably narrow enough

Brian Hayes is Senior Writer for American Scientist. *Additional material related to the "Computing Science" column appears in Hayes's Weblog at http://bit-player.org. Address: 211 Dacian Avenue, Durham, NC 27701. Internet: bhayes@amsci.org*

to cope with any practical problems arising at the pancake griddle. From a mathematical point of view, however, the pancake problem remains unsolved: The exact number of flips needed to sort *n* pancakes is unknown.

Gene flipping has a lot in common with the pancake problem, but there are differences as well. With pancakes, you are not allowed to reach into the middle of the stack and change the internal order; you can only reverse subsequences of pancakes at the top of the pile. In the case of genes lined up along a chromosome, there's no reason to enforce such a constraint. Any block of genes can be inverted, whether it's at either end of the chromosome or in the interior. This additional freedom ought to make permuting genes somewhat easier than flipping flapjacks.

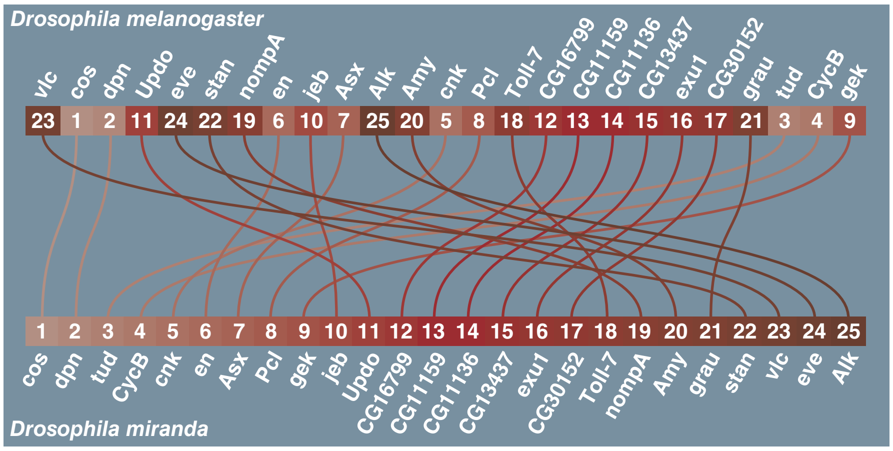
On the other hand, the genetic problem is harder in a different way. Pancake sorting has mostly been treated as a quest for worst-case results. The quantity of primary interest is not the number of flips needed for any particular stack of *n* pancakes but rather the maximum for all possible stacks of height *n*. Solving the biological problem calls for a finer level of detail: A geneticist would like to know the reversal distance between particular gene configurations seen in nature. Knowing the maximum for all *n*-gene chromosomes would not be nearly as helpful.

Gene Sorting

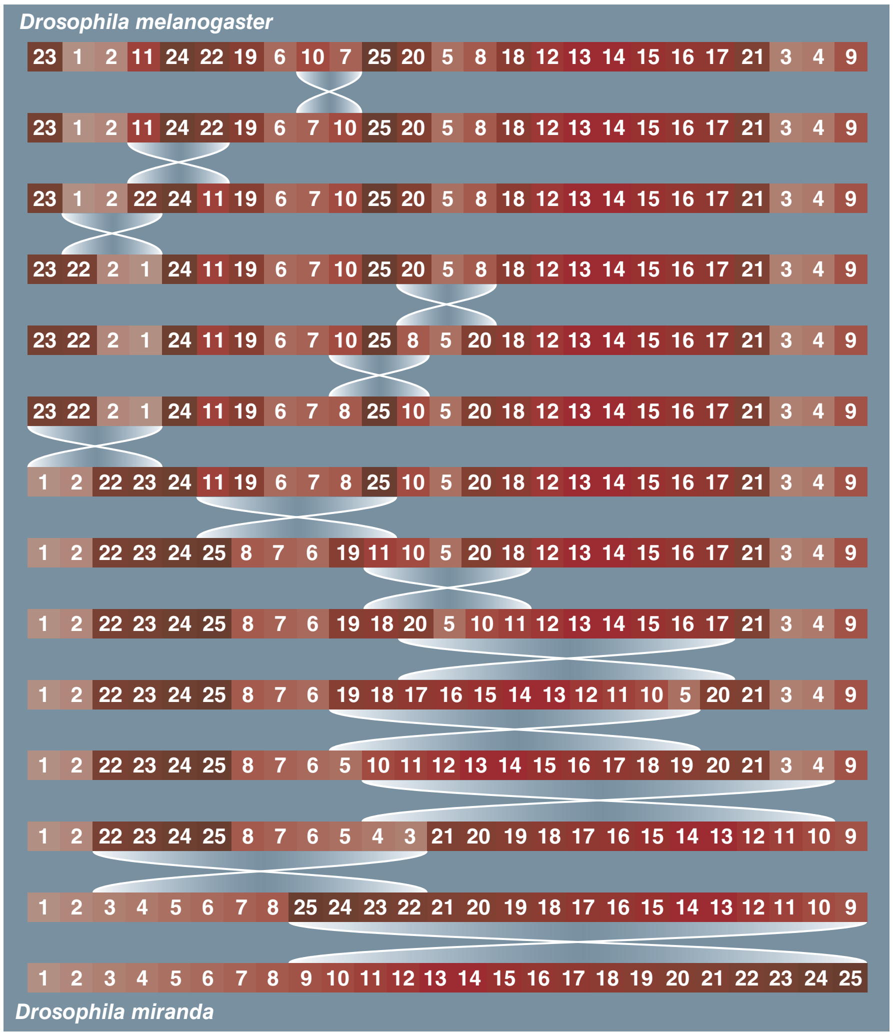
Geneticists give *Drosophila* genes charming names like *groucho*, *kojak* and *dreadlocks*, but for algorithmic purposes it's a lot easier to just number them. From this point of view a chromosome is nothing more than a permutation of the numbers from 1 through *n*.

The goal is a method for finding the reversal distance between any two permutations, such as 31524 and 41235. Because the labeling of the genes is arbitrary, however, we can always assign the numbers in such a way that one configuration is the *canonical* permutation 123…*n*, with the integers in ascending order. Then the task is simply to *sort* the other permutation, finding a sequence of reversals that restores it to the canonical order. Instead of converting 41235 into 31524, we can sort the sequence 52413 into its canonical order 12345; the operations required are identical.

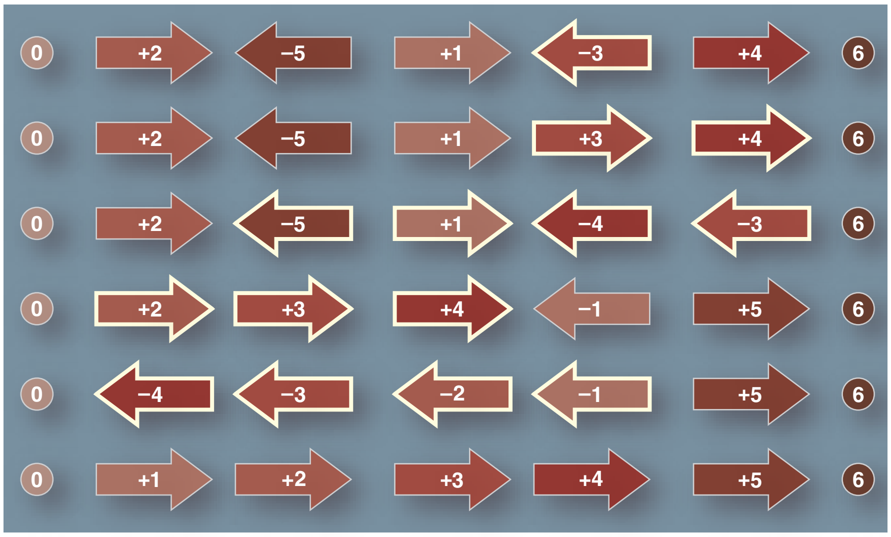
Can every permutation be sorted by some series of reversal moves, with-


Drosophila melanogaster
vlc cos dpn Updo eve stan nompA en jeb Asx Alk Amy cnk Pcl Toll-7 CG16799 CG11159 CG11136 CG13437 exu1 CG30152 grau tud CycB gek
23 1 2 11 24 22 19 6 10 7 25 20 5 8 18 12 13 14 15 16 17 21 3 4 9
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
cos dpn tud CycB cnk en Asx Pcl gek jeb Updo CG16799 CG11159 CG11136 CG13437 exu1 CG30152 Toll-7 nompA Amy grau stan vlc eve Alk
Drosophila miranda

Nature reshuffles the deck: Two species of the fruit fly *Drosophila* have the same complement of genes, but they are arranged very differently. The mixing is caused mainly by occasional reversal events, in which a block of contiguous genes turns 180 degrees. But it's not obvious just what sequence of reversals created the particular permutation seen here. The genes have been numbered arbitrarily to put the *D. miranda* genome in canonical order. The illustration is based on an analysis by Carolina Bartolomé and Brian Charlesworth.



The reversal distance between two genomes—the minimum number of reversals needed to convert one permutation into the other—can serve as a measure of evolutionary divergence. The calculated reversal distance between *D. melanogaster* and *D. miranda* is 13. The sequence of 13 reversals shown here is one possible pathway for the transformation, although it is probably not unique. The pathway was generated by the GRIMM software of Glenn Tesler.



Signed reversals are rearrangements that take into account not only the position of each gene but also its orientation. Genome-sorting algorithms incorporate the directionality of DNA by giving each gene a plus or a minus sign; reversing a block of genes inverts their order and changes their signs. Here a five-element signed permutation is sorted in five steps. In each row the genes to be reversed next are highlighted with a bright outline. Note that the first step inverts a single gene, something that would never be done in sorting an unsigned permutation. The flanking elements 0 and 6 serve to anchor the ends of the genome, breaking the left-right symmetry.

out any need for other kinds of operations? A simple variation on the naive pancake-flipping algorithm supplies an affirmative answer. From any starting configuration, there must be some block of genes that can be rotated 180 degrees to bring gene 1 into its correct, leftmost position. Thereafter some other block can be flipped to put gene 2 in its proper place; moreover, this can be done without disturbing gene 1. Here is the full sequence of reversals that sorts our five-digit example (the underline designates the segment that is about to be reversed):

$$\begin{gathered}\underline{5\,2\,4\,1}\,3\\ 1\,\underline{4\,2}\,5\,3\\ 1\,2\,\underline{4\,5\,3}\\ 1\,2\,3\,\underline{5\,4}\\ 1\,2\,3\,4\,5\end{gathered}$$

Using this left-to-right algorithm, the five genes are sorted with four reversals. It's not hard to see that any *n*-element array can be sorted by the same method in at most *n*−1 reversals. The procedure is similar to the bottom-up pancake algorithm, but because we don't have to work only from one end of the chromosome, each gene can be put in its place with a single flip rather than two.

Unfortunately, the left-to-right algorithm cannot always be counted on to find the *shortest* sequence of reversals for sorting a chromosome. In this example, four steps is not the minimum; there is a three-step sequence that accomplishes the same task:

$$\begin{gathered}5\,2\,\underline{4\,1}\,3\\ \underline{5\,2\,1}\,4\,3\\ 1\,2\,\underline{5\,4\,3}\\ 1\,2\,3\,4\,5\end{gathered}$$

But there is no obvious logic behind this series of moves, so the question becomes: How do you find such shorter pathways, and how do you know when you've found the shortest route of all?

The importance of identifying the shortest reversal sequence is not that we need a high-performance algorithm for sorting genes on chromosomes. Sorting the genes is not really the point. The shortest pathway matters because it is the best available estimate of the true reversal distance between two genomes—the measure of their evolutionary kinship.

In general, we can't reconstruct the actual course of an evolutionary process with any certainty, but parsimony argues that short and direct pathways are more likely than long, zigzag ones. A change in gene order from 123 to 321 *might* have come about through a multistep transformation such as $\underline{12}3 \rightarrow \underline{213} \rightarrow 3\underline{12} \rightarrow 321$, but the single reversal $\underline{123} \rightarrow 321$ is usually a better guess. Thus the shortest pathway is always the starting point for inferring a phylogeny. (The shortest series of reversals is not necessarily unique; there may be several reversal pathways of the same minimal length.)

Breakpoints

In sorting out the process of sorting by reversals, a good place to begin is with the work of John Kececioglu, now of the University of Arizona, and David Sankoff of the University of Ottawa. They were not the first to write about the problem, but they were the first to make substantial progress on it, starting in the early 1990s. Their principal tool was the concept of a breakpoint.

A breakpoint appears wherever a permutation brings together two nonconsecutive numbers. For example, this eight-element permutation has two breakpoints, marked with carets:

$$2\,1_{\wedge}6\,5\,4\,3_{\wedge}7\,8.$$

A segment *not* interrupted by a breakpoint is called a strip; it consists of consecutive numbers in either ascending or descending order. The importance of breakpoints is that the canonical permutation has none. Thus any procedure that keeps reducing the number of breakpoints until the count reaches zero is a method for sorting a permutation. (A complication is that both the forward permutation 123…*n* and the backward one *n*…321 are without breakpoints. The remedy is to augment the sequence with "anchoring" elements 0 and *n*+1, which compel the sorting algorithm to choose the forward direction.)

Breakpoint analysis suggests a lower bound on the complexity of sorting by reversals. A reversal can create or destroy breakpoints only at the ends of the segment being reversed; it follows that a single reversal can eliminate at most two breakpoints. If the initial permutation has *m* breakpoints, no sorting procedure can possibly eliminate all of them in fewer than *m*/2 reversals.

Kececioglu and Sankoff devised a sorting algorithm that comes within a factor of two of this limit; in other words, the algorithm's maximum number of reversals will not exceed the number of breakpoints in the original permutation. The algorithm employs a "greedy" strategy: At each stage in the computation it maximizes the number of breakpoints removed. If there is an opportunity to eliminate two breakpoints, that move is given higher priority than all reversals that remove just one breakpoint, etc.

A focus on breakpoints makes intuitive sense in sorting permutations. After all, the breakpoints are the positions where the numbers are out of order, whereas the strips are already sorted. This line of reasoning leads to the conjecture that an optimal algorithm for sorting by reversals should operate only at breakpoints, never cutting the permutation within a strip. Unfortunately, the conjecture does not hold. Consider the permutation 3412, which has a single breakpoint, between the 4 and the 1. Sorting with an algorithm that never cuts a strip takes three reversals: $\underline{34}12 \rightarrow 43\underline{12} \rightarrow \underline{4321} \rightarrow 1234$. But another sequence gets to the goal in just two reversals: $\underline{341}2 \rightarrow 1\underline{432} \rightarrow 1234$.

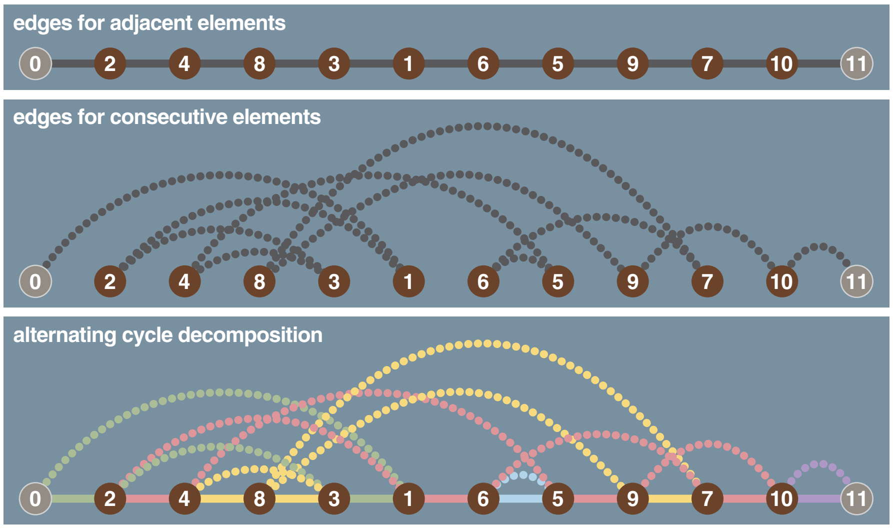
Kececioglu and Sankoff's greedy algorithm runs efficiently but gives only approximate results. They also described a program that makes the opposite tradeoff, finding the optimum sorting sequence but paying a penalty in running time and memory requirements. At the heart of this algorithm is a routine that may have to examine every possible sequence of reversals for sorting the given permutation. The number of such reversal sequences grows exponentially with n; indeed, the worst-case rate of growth exceeds n^n. By means of a "branch and bound" strategy Kececioglu and Sankoff were able to greatly reduce the number of sequences considered, but the worst-case performance remains exponential, and their program was able to find exact solutions only up to about $n = 30$.

In 1995 Kececioglu and Sankoff speculated that optimal sorting by reversals belongs to the class of computational problems designated "NP-hard." These are problems for which—barring miracles—only exponential algorithms will ever be found. Two years later Alberto Caprara of the University of Bologna proved that finding the shortest sequence of reversals for sorting an arbitrary permutation is indeed NP-hard.

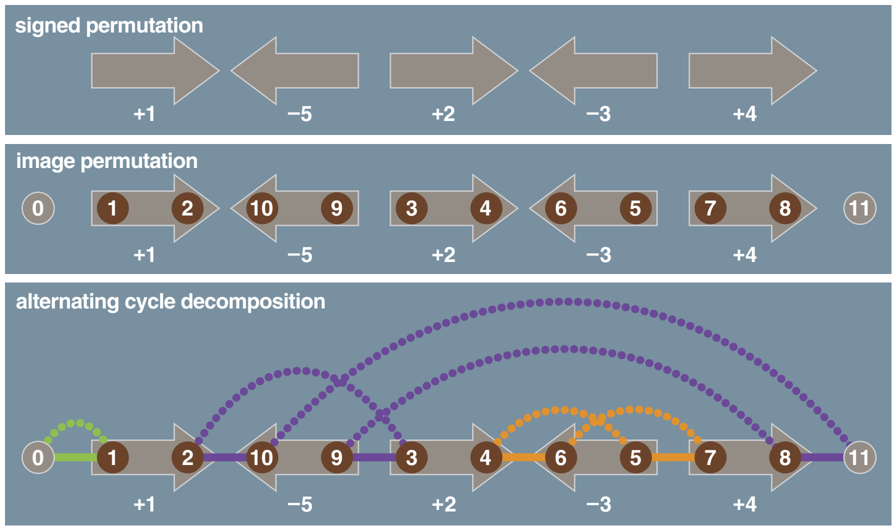
Signs of Change

If a problem is too hard to solve, do you look for a simpler one? In this case the response was just the opposite. Attention turned to a more elaborate variant of the sorting-by-reversals problem. And, surprisingly, efficient solutions to this variant soon emerged. Furthermore, as a bonus, the variant problem offers a more faithful model of the biological process by which genes get jumbled.

To explain this strange twist in the plot, I need to introduce a crucial fact about genetic information that I have neglected mentioning up to now. Each gene on a chromosome has not only a position but also an orientation; think of a gene as an arrow rather than a line segment. Like the letters in a word of


edges for adjacent elements
0 2 4 8 3 1 6 5 9 7 10 11
edges for consecutive elements
0 2 4 8 3 1 6 5 9 7 10 11
alternating cycle decomposition
0 2 4 8 3 1 6 5 9 7 10 11

Mathematical graph theory is the primary tool for calculating reversal distance. A graph consists of vertices (or dots) and edges (lines that connect the vertices). In this graph of a 10-element unsigned permutation (with additional anchoring elements 0 and 11) there are two kinds of edges. Solid edges *(top)* join each element to its immediate neighbors in the permutation; dotted edges *(middle)* join numerically consecutive elements. Of special importance in the analysis of reversals are alternating cycles in the graph: closed loops formed by traversing first a solid edge, then a dotted edge, and so on. A reversal can create or destroy at most one cycle, and so the number of cycles provides an estimate of the reversal distance. Here five cycles are identified *(bottom)*. However, for unsigned permutations, finding the maximum cycle decomposition is itself a hard problem, and so this technique does not lead to an efficient reversal-distance algorithm.



The cycle graph for a signed permutation is more elaborate but more tractable than the unsigned structure. The first step in building the graph is to replace the signed permutation of *n* elements with a new unsigned "image" permutation of 2*n* elements (augmented by 0 and 2*n*+1). A signed element *x* becomes a pair of elements 2*x*–1 and 2*x*. If x is envisioned as an arrow pointing left for –*x* and right for +*x*, then 2*x* always goes in the head of the arrow, and 2*x*–1 in the tail. Solid edges join physically adjacent vertices, and dotted edges connect numerically consecutive elements, but with a special proviso: No edges connect the 2*x* and 2*x*–1 vertices derived from a single signed element. In other words, no edges are drawn inside an arrow. Counting the cycles in the graph is easy. Because each vertex receives one solid edge and one dotted edge, there is only one way to decompose the graph into alternating cycles.

text, the sequence of nucleotides in a strand of DNA reads correctly in only one direction. Thus when two genomes are compared, it's necessary to match both the order of the genes along the chromosome and the direction of each gene. Early methods of mapping genes gave no information about their direction, and so the issue was ignored. With instruments that can read nucleotide sequences directly from a molecule of DNA, data on gene orientation is becoming more readily available.

A chromosome with oriented genes corresponds to a signed permutation, one where each element in the set 123…*n* has either a plus or a minus sign. Genes facing one way are positive, those with the opposite direction are negative. Reversing a segment of the chromosome inverts the order of the genes and also changes all their signs. For example, the signed permutation +3 –1 –2 +4 has the inverse –4 +2 +1 –3. To sort a signed permutation, you find a series of reversals that arranges the genes in ascending order and makes all the signs positive. Going back to the breakfast table again, the corresponding task is known as the burnt pancake problem: The pancakes have to be stacked in order of size with the blackened sides down.

Common sense argues that signed permutations must surely be harder to sort than unsigned ones. There are more constraints and more variables. The trouble with this reasoning is not that it gives the wrong answer but that it answers the wrong question. In general, sorting a signed permutation does take more work than sorting an unsigned one—but the efficiency of sorting is not at issue. What's wanted is not a sorting of the genome but a measure of the number of reversals needed to sort it, and that is indeed easier to calculate with a signed permutation.

Leaping Hurdles

The idea that overturned everyone's thinking about reversal sorting came from Pavel A. Pevzner of the University of California, San Diego, working in collaboration first with Vineet Bafna, also of San Diego, and then with Sridhar Hannenhalli of the University of Pennsylvania. They presented their sorting scheme in the language of mathematical graph theory—the science of dots and lines. The illustrations on page 389 show the construction of some of the graphs that enter into the sorting method. The process has its intricacies, including the replacement of a signed permutation of the numbers 1 through *n* by an unsigned permutation of 1 through 2*n*.

Recently Anne Bergeron of the Université du Québec à Montréal has given an alternative explanation of the same algorithm, presenting it more directly in terms of the original signed permutation. The account that follows is based on Bergeron's work.

The key concept is that of an "oriented pair": two numbers, found anywhere within the permutation, that have opposite signs but are consecutive when you ignore the signs. Bergeron observes that a reversal that brings these numbers together will also give them like signs. The process is easier to understand with an example. Consider the six-element permutation 0 +3 +1 +6 +5 –2 +4 +7 (shown flanked by the "anchoring" elements 0 and 7, which will never move). The permutation has two oriented pairs, namely +3 –2 and +1 –2, and each of these can be reunited in two ways. For example, reversing the segment +6 +5 –2 brings the 1 and the 2 together in their correct order and leaves both with a plus sign, whereas reversing +3 +1 +6 +5 creates the subsequence –3 –2.

Bergeron derives deterministic rules for identifying oriented pairs and choosing which of the pairs to reunite next. The rules favor reversals that leave positive numbers in ascending order and negative numbers in descending order. They also give precedence to whichever reversal creates the largest number of new oriented pairs. In this case, the +3 +1 +6 +5 reversal scores highest, generating four oriented pairs. The rules can now be applied again to choose another optimal reversal. Continuing in this way brings the original permutation to canonical order in five steps:

$$\begin{array}{l} 0\ \underline{+3\ +1\ +6\ +5}\ -2\ +4\ +7 \\ 0\ -5\ -6\ -1\ \underline{-3\ -2}\ +4\ +7 \\ 0\ -5\ -6\ \underline{-1}\ +2\ +3\ +4\ +7 \\ 0\ -5\ \underline{-6\ +1\ +2\ +3\ +4}\ +7 \\ 0\ \underline{-5\ -4\ -3\ -2\ -1}\ +6\ +7 \\ 0\ +1\ +2\ +3\ +4\ +5\ +6\ +7 \end{array}$$

This is the shortest possible reversal-sorting sequence.

The procedure explained by Bergeron is guaranteed to reach a permutation in which all elements have plus signs. Usually, that permutation is the canonical one, and so the task of sorting is completed. However, there are exceptional permutations for which the algorithm fails, and further steps are needed. This is not an artefact of Bergeron's simplified version; the problem is with the underlying algorithm and can be traced to subtle features of the graphs that describe the permutations. Pevzner and his colleagues named these disruptive structures hurdles, and there are even rarer impediments known as superhurdles and fortresses. In an early version of the algorithm the hurdles could not be overcome. Hannenhalli and Pevzner later found a series of intricate maneuvers that can clear all the varieties of hurdles, and so the measurement of reversal distance is now exact.

The Happy Ending

The algorithm published by Hannenhalli and Pevzner finds the optimum sequence of reversals in an amount of time proportional to n^4. Others soon improved this bound to n^3 and n^2. Eric Tannier and Marie-France Sagot of the Université Claude Bernard in France apparently hold the current record: $n\sqrt{n \log n}$. If you simply want to know the length of the shortest sequence of reversals (without seeing the sequence itself), there is a linear method, taking time proportional to the first power of *n*. Even the slowest of these algorithms is immeasurably more efficient than an exponential method, and they can all cope with any realistic biological data.

Several authors have implemented reversal-sorting algorithms in publicly available software. Of particular note is a Web site called GRIMM, created by Glenn Tesler of the University of California, San Diego, which calculates the optimum sequence of reversals for any pair of signed permutations. On a lighter note, the online version of this column includes an interactive illustration that will allow you to try your own hand at finding an optimum series of reversals for small random permutations.

The decade-long effort to solve the sorting-by-reversals problem stands as a model of interdisciplinary collaboration. The question that inspired the work in the first place was of genuine importance to biology, but it also turned out to hold real interest for mathematicians and computer scientists, who might well have tackled the problem even if it didn't have an appealing application. I suspect that some of the competitive fine-tuning of algorithms goes beyond the actual needs of biologists, but those whose main interests are computational deserve to have their fun too.

In any case, there are plenty of biological challenges that remain to be mastered. The abstract version of the sorting problem implicitly assumes that all segments of a chromosome are equally likely to be reversed. Biologists know that there are actually biases favoring some segments, or some junctions between segments. Furthermore, reversals of chromosomal segments are not the only mechanism for mixing up the genome. In organisms with multiple chromosomes, there are also transpositions, fissions and fusions to be taken into account. Genes are duplicated or deleted. DNA is not, in the end, just a permutation of signed numbers, and yet it's impressive how much can be gleaned from such simple models.

Bibliography

Bader, David A., Bernard M. E. Moret and Mi Yan. 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Proceedings of the Seventh Workshop on Algorithms and Data Structures* (WADS '01), pp. 365–376. Berlin: Springer-Verlag.

Bafna, Vineet, and Pavel Pevzner. 1996. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing* 25:272–289.

Bartolomé, Carolina, and Brian Charlesworth. 2006. Rates and patterns of chromosomal evolution in *Drosophila pseudoobscura* and *D. miranda*. *Genetics* 173:779–791.

Bergeron, Anne. 2005. A very elementary presentation of the Hannenhalli-Pevzner theory. *Discrete Applied Mathematics* 146:134–145.

Bergeron, Anne, and François Strasbourg. 2001. Experiments in computing sequences of reversals. *Proceedings of the First International Workshop on Algorithms in Bioinformatics*, pp. 164–174. Berlin: Springer-Verlag.

Berman, Piotr, and Sridhar Hannenhalli. 1996. Fast sorting by reversal. *Proceedings of the Seventh Symposium on Combinatorial Pattern Matching*, pp. 168–185. Berlin: Springer-Verlag.

Caprara, Alberto. 1997. Sorting by reversals is difficult. *Proceedings of RECOMB '97: The First International Conference on Computational Molecular Biology*, pp. 75–83. New York: ACM Press.

Dobzhansky, Th., and A. H. Sturtevant. 1938. Inversions in the chromosomes of *Drosophila pseudoobscura*. *Genetics* 23:28–64.

Gates, William H., and Christos H. Papadimitriou. 1979. Bounds for sorting by prefix reversal. *Discrete Mathematics* 27:47–57.

Hannenhalli, Sridhar, and Pavel A. Pevzner. 1999. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM* 48:1–27.

Heydari, Mohammad H., and I. Hal Sudborough. 1997. On the diameter of the pancake network. *Journal of Algorithms* 25:67–94.

Kaplan, Haim, Ron Shamir and Robert E. Tarjan. 1997. Faster and simpler algorithm for sorting signed permutations by reversals. *Proceedings of the Eighth ACM-SIAM Symposium on Discrete Algorithms*, pp. 344–351. New York: ACM Press.

Kaplan, Haim, and Elad Verbin. 2005. Sorting signed permutations by reversals, revisited. *Journal of Computer and System Sciences* 70:321–341.

Kececioglu, J. D., and D. Sankoff. 1995. Exact and approximation algorithms for sorting by reversal, with application to genome rearrangement. *Algorithmica* 13:180–210.

Palmer, Jeffrey D., and Laura A. Herbon. 1988. Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution* 28:87–97.

Pevzner, Pavel A. 2000. *Computational Molecular Biology: An Algorithmic Approach*. Cambridge, Mass.: MIT Press.

Tannier, Eric, and Marie-France Sagot. 2004. Sorting by reversals in subquadratic time. *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching*, pp. 1–13. Berlin: Springer-Verlag.

Tesler, Glenn. 2002. GRIMM: genome rearrangements web server. *Bioinformatics* 18:492–493. See also http://www-cse.ucsd.edu/groups/bioinformatics/GRIMM/index.html

Watterson, G. A., W. J. Ewens, T. E. Hall and A. Morgan. 1982. The chromosome inversion problem. *Journal of Theoretical Biology* 99:1–7.