Genome Rearrangements and Greedy Algorithms

Genome Rearrangement Problem

When genomes of different species were compared, biologists noticed that *groups of genes* appeared together in the different genomes, however, the *global arrangement* of these "blocks" of genes differed from genome to genome.

Biologists hypothesize that such rearrangements are due to *gene reversals*: groups of genes being "transferred" from one DNA strand to the other, and thus reversing their order.

Reversal. More formally, let $\pi = \pi_1, \ldots \pi_i, \ldots \pi_j, \ldots \pi_n$ be a permutation of $\{1, \ldots, n\}$. A **reversal** of positions i and j in π, denoted $\rho(i,j)$ is a transformation that results in the following permutation π':

$$\pi' \pi \circ \pi_1, \ldots \pi_{i-1}, \underline{p_j, p_{j-1}, \ldots, \pi_{i+1}, \pi_i}, \pi_{j+1}, \ldots, \pi_n.$$

Example. Consider a permutation $\pi = 1,2,3,4,5,6,7$. A reverasal $\rho(3,6)$ applied to π will produce the following permutation:

$$\begin{aligned} \pi = &\quad 1,2,3,4,5,6,7 \\ \pi \circ \rho(3,6) = &\quad 1,2,6,5,4,3,7 \end{aligned}$$

Biological meaning. In a permutation π, each component π_i represents a block of genes that persisted through a variety of genomes. Biologists construct such *maps* for different organisms - each can be represented by some permutation. The key study question here is, what is the sequence of reversals that leads from one genome to another.

Genome reversal problem. Given two *permutations* π and σ of n numbers, find a series of *reversals* $\rho_1, \ldots, \rho_t$, such that:

1. $\pi \circ rho_1 \circ \ldots \circ \rho_t = \sigma$;
2. t is minimized.

t is called the *reversal distance* between π and σ.

Note. Without loss of generality, we assume that $\sigma = 1, 2, 3, \ldots, n$. The problem of genome reversal to the identity permutation is sometimes called the **sorting by reversal problem**.

Greedy Algorithm: prefix maximization

Prefixes. Let $\pi = \pi_1, \ldots, \pi_n$ and let $\pi_1, \ldots \pi_i = 1, 2, \ldots i$. We call $\pi_1, \ldots, \pi_i$, the *prefix* of π and denote it $prefix(\pi)$.

Idea. Let $prefix(\pi) = \pi_1, \ldots \pi_i$. Find position j in π, such that $\pi_j = i + 1$. Apply reversal $\rho(i + 1, j)$ to π.

$\pi \circ \rho(i + 1, j)$ will increase its prefix by *at least* one position (possibly more, if, e.g., $\pi_{j-1} = i + 2$). Thus, on each step, we are approaching monotonically the sorted permutation.

Greed. The algorithm that extends the $prefix(\pi)$ is *greedy*, because on each step it goes for the *low-hanging fruit* of increasing the already sorted part of the permutation.

Algorithm. The pseudocode for the algorithm is below.

```
Algorithm SimpleReveraslSort(π)
 begin
   for i ← 1  to n − 1  do
     j ← FindPosition(π,i);
     if  j ≠ i  then
       π ← Reversal(π,i,j);
       print  π;
     end if
     if π = 1, 2, ..., n  then
       return;
     end if
   end for
 end
```

Function FindPosition(). FindPosition finds a position j in the permutation (array) π, such that $\pi_j = i$.

Without preprocessing, it can be done in $O(n)$ for each call. To speed up FindPosition() we can create an array $P[1..n]$, such that $P[i]$ is such that $\pi_{P[i]} = i$, i.e., the position in π, which contains number i. If an *up-to-date* version of P is available prior to calling FindPosition, then, FindPosition() works in $O(1)$.

Function Reversal(). Reversal(π, i, j) performs the reversal operation $\rho(i, j)$ on permutation π. This can be done in $O(n)$ time, with the use of a supplemental variable which would be used for value exchange.

Because a reversal changes the locations of various values in the permutations, a new array P needs to be computed. This computation, however, can be done in a straightforward manner - each time a new assignment to a position in π

is made, the appropriate update of the array P is performed. This doubles the number of operations, but the running time of this algorithm will remain $O(n)$.

Analysis. Straightforward implementation of SimpleReversalSort() takes $O(n^2)$ time: the outer loop repeats $O(n)$ times, and each loop repeat involves, in worst case, an $O(n)$ operation.

SimpleReversalSort is NOT optimal. Consider the following permutation $\pi = 7, 6, 1, 2, 3, 4, 5$. SimpleReveralSort with π as input will produce the following output:

```
7 6 1 2 3 4 5    Reversal(1,3)
1 6 7 2 3 4 5    Reversal(2,4)
1 2 7 6 3 4 5    Reversal(3,5)
1 2 3 6 7 4 5    Reversal(4,6)
1 2 3 4 7 6 5    Reversal(5,7)
1 2 3 4 5 6 7
```

That is, SimpleReversalSort sorts the permutation using **five** reversals. Yet, the following shows that there is a sequence of reversals that takes fewer steps:

```
7 6 1 2 3 4 5    Reversal(3,7)
7 6 5 4 3 2 1    Reversal(1,7)
1 2 3 4 5 6 7
```

Greedy Algorithm: Breakpoints

Breakpoints. Let $\pi = \pi_1, \ldots, \pi_n$ be a permutation. Without loss of generality we extend it to $\pi = \pi_0, \pi_1, \ldots, \pi_n, \pi_{n+1}$. A pair positions π_i, π_{i+1} is called an *adjacency* if $|\pi_i - \pi_{i+1}| = 1$. Otherwise, the pair π_i, π_{i+1} is called a **breakpoint**.

Example. Let $\pi = 6, 5, 2, 3, 1, 4$. π has two (2) adjacencies: *6,5* and *2,3*. It has three (3) breakpoints: *5,2*, *3,1* and *1,4*.

Idea. $\sigma = 1, 2, \ldots, n$ has **no** breakpoints and $n-1$ adjacencies. All adjacencies are *increasing*. If π has $b(\pi)$ breakpoints, then an algorithm that monotonically decreases the number of breakpoints until it reaches 0 will solve the problem.

How bad can it get? Each reversal can potentially eliminate two breakpoints. Therefore, the total number of reversals needed to sort, $d(\pi) \geq \frac{b(\pi)}{2}$.

Strips. A *strip* in a permutation π is an interval between two consecutive breakpoints. A *strip* $\pi_i, \ldots, \pi_j$ is *increasing* if $\pi_i < \pi_{i+1} < \ldots < \pi_j$. A *strip* is *decreasing* otherwise. (Strips of length 1 are both increasing and decreasing).

Example. Let $\pi = 0, 7, 5, 4, 3, 1, 2, 8$. There is a decreasing strip $5, 4, 3$ between the breakpoints $7, 5$ and $3, 1$ in π. Similarly, there is an increasing strip $1, 2$ between breakpoints $3, 1$ and $2, 8$.

Theorem. If a permutation π has a decreasing strip, then there exists a reversal ρ that decreases the number of breakpoints in π: $b(\pi \circ \rho) < b(\pi)$.

Proof. Consider a decreasing strip $\pi_i, \ldots, \pi_j$ in π, such that $\pi_j = k$ is the smallest number terminating a decreasing strip. The number $k-1$ must therefore terminate an *increasing strip*: (k is the smallest terminus of a decreasing strip, it is NOT followed by $k-1$, hence $k-1$ either is surrounded by two breakpoints somewhere, or is the end of an increasing strip of length ≥ 2. Let s be the position of k in π. Let $\rho = \rho(s+1, j)$ (or $\rho(j+1, s)$ - depending on which number if greater).

$\pi \circ \rho$ will eliminate put k and $k-1$ on the same strip.

Example. Consider $\pi = 0, 4, 3, 7, 6, 2, 1, 5, 8$. Here, the decreasing strip with the smallest terminus is $2, 1$, and $k - 1 = 0$, at position 0. The total number of breakpoints is $5((0,4), (3,7), (6,2), (1,5), (5,8)$. We use the reversal $\rho(1,6)$.

$$\pi \circ \rho(1,6) = 0, \underline{1, 2, 6, 7, 3, 4}, 5, 8.$$

Here, the number of breakpoints is $3 : (2,6), (7,3)$ and $(5,8)$.

Example. Consider $\pi = 0, 5, 4, 6, 7, 1, 2, 3, 8$. This permutation has 4 breakpoints $((0,5), (4,6), (7,1), (3,8)$ and only one decreasing strip, $5, 4$. We apply the trasformation $\rho(3,7)$:

$$\pi \circ \rho(1,7) = 0, 5, 4, \underline{3, 2, 1}, 7, 6, 8.$$

Here, the number of breakpoints is $3 : (0,5), (1,7)$ and $(6,8)$.

What if there is no decreasing strip? If π has no decreasing strip, then we pick any increasing strip and reverse it. This will create a decreasing strip in π and we can apply our theorem.

Lemma. If π has no decreasing strips, then reversing any increasing strip **does not** change the total number of breakpoints.

Example. Recall our permutation $\pi = 0, 4, 3, 7, 6, 2, 1, 5$. We applied a reversal $\rho(1,6)$ to it to get $\pi \circ \rho(1,6) = 0, 1, 2, 6, 7, 3, 4, 5, 8$. This permutation has no decreasing strips. We pick an increasing strip $6, 7$ and reverse it using $\rho(3,4)$:

$$\pi \circ \rho(1,6) \circ \rho(3,4) = 0, 1, 2, \underline{7, 6}, 3, 4, 5, 8.$$

There is now a decreasing strip and we can proceed with reversing it:

$$\pi \circ \rho(1,6) \circ \rho(3,4) \circ \rho(5,7) = 0, 1, 2, 7, 6, \underline{5, 4, 3}, 8$$

$$\pi \circ \rho(1,6) \circ \rho(3,4) \circ \rho(5,7) \circ \rho(3,7) = 0, 1, 2, \underline{3, 4, 5, 6, 7}, 8$$

Algorithm. The outline of the algorithm is:

1. If π has decreasing strips, find the decreasing strip with the smallest terminus k, and merge it with $k-1$.

2. If π has no decreasing strips, reverse any increasing strip.

References

[1] John Kececioglu, David Sankoff, Exact and Approximation Algorithms for Sorting by Reversals with Applications to Genome Rearrangement, *Algorithmica*, Vo. 1/2: pp. 180-210 (1995).