Orthologs vs. Paralogs

Three events:
1. Speciation
   **Orthologs**: genes descended from a common ancestor.
2. Loss
3. Duplication
   **Paralogs**: genes related by duplication.

Distinguishing orthologs from paralogs is difficult! Sequence similarity is not enough.
Whole-Genome Phylogeny

Rearrangements can be used for phylogenetics.

Cell Division and Mutation

DNA Replication

Mitosis

Two Diploid Cells

Single nucleotide change

Deletion

Duplication

Inversion

Translocation

Copy number

Structural
History of Chromosome X


Outline

Genome Rearrangement Models

- Sorting By Reversals
  - Naïve Greedy Algorithm
  - Breakpoints and Greedy algorithm
- Breakpoint Graphs
- Multichromosomal Rearrangements
Reversals: Biology

Reversals: Simplified Representation

Blocks represent segments of the genome conserved across multiple species
1. Genes
2. Synteny blocks (identified by comparing genome sequences)
Blocks represent conserved genes.
- In the course of evolution or in a clinical context, blocks 1,...,10 could be incorrectly copied as 1, 2, 3, 8, 7, 6, 5, 4, 9, 10.

The inversion/reversal introduced two breakpoints (disruptions in order).
Reversals: Example

\[ \pi = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \]

\[ \rho(3,5) \]

\[ 1 \ 2 \ 5 \ 4 \ 3 \ 6 \ 7 \ 8 \]

Reversals: Example

\[ \pi = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \]

\[ \rho(3,5) \]

\[ 1 \ 2 \ 5 \ 4 \ 3 \ 6 \ 7 \ 8 \]

\[ \rho(5,6) \]

\[ 1 \ 2 \ 5 \ 4 \ 6 \ 3 \ 7 \ 8 \]
Reversals and Gene Orders

Gene order is represented by a permutation $\pi$:

$\pi = \pi_1 \cdots \pi_{i-1} \pi_i \pi_{i+1} \cdots \pi_{j-1} \pi_j \pi_{j+1} \cdots \pi_n$

Reversal $\rho(i, j)$ reverses (flips) the elements from $i$ to $j$ in $\pi$

Reversal Distance

For phylogenetic reconstructions, need a distance between “genomes”: permutations $\pi$ and $\sigma$.

Reversal distance $d(\pi, \sigma)$ is the minimum number of reversals $\rho_1 \ldots \rho_t$ transforming $\pi$ into $\sigma$. 
Reversal Distance Problem

Goal: Given two permutations, find the shortest series of reversals that transforms one into another

- **Input**: Permutations \( \pi \) and \( \sigma \).
- **Output**: A series of reversals \( \rho_1, \ldots, \rho_t \) transforming \( \pi \) into \( \sigma \) such that \( t \) is minimum.

**Note**: A variation: Output \( t \), reversal distance.

-------

Sorting By Reversals Problem

By relabeling, can consider one permutation to be the identity permutation \( (1 \ 2 \ \ldots \ n) \).

**Input**: Permutation \( \pi \)

**Output**: A series of reversals \( \rho_1, \ldots, \rho_t \) transforming \( \pi \) into the identity permutation:

\[
\rho_1 \ldots \rho_t \pi = (1 \ 2 \ \ldots \ n)
\]

such that \( t \) is minimum.
Sorting By Reversals: Example

- $t = d(\pi)$ - reversal distance of $\pi$
- Example:

\[
\begin{align*}
\pi &= 3 \ 4 \ 2 \ 1 \ 5 \ 6 \ 7 \ 10 \ 9 \ 8 \\
4 \ 3 \ 2 \ 1 \ 5 \ 6 \ 7 \ 10 \ 9 \ 8 \\
4 \ 3 \ 2 \ 1 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \\
1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10
\end{align*}
\]

So $d(\pi) = 3$

Pancake Flipping Problem

- The chef is sloppy; he prepares an unordered stack of pancakes of different sizes
- The waiter wants to rearrange them (so that the smallest winds up on top, and so on, down to the largest at the bottom)
- He does it by flipping over several from the top, repeating this as many times as necessary
Pancake Flipping Problem: Formulation

**Goal:** Given a stack of \( n \) pancakes, what is the minimum number of flips to rearrange them into perfect stack?

- **Input:** Permutation \( \pi \)
- **Output:** A series of prefix reversals \( \rho_1, \ldots, \rho_t \) transforming \( \pi \) into the identity permutation such that \( t \) is minimum.

A prefix reversal is \( \rho(1, k) \) for some \( k \).

Pancake Flipping Problem: Greedy Algorithm

- **Greedy approach:** 2 prefix reversals at most to place a pancake in its right position, \( 2n - 2 \) steps total

- William Gates and Christos Papadimitriou showed in the mid-1970s that this problem can be solved by at most \( 5/3 (n + 1) \) prefix reversals
Sorting By Reversals: A Greedy Algorithm

• If sorting permutation \( \pi = 1 \ 2 \ 3 \ 6 \ 4 \ 5 \), the first three elements are already in order so it does not make any sense to break them.
• The length of the already sorted prefix of \( \pi \) is denoted \( \text{prefix}(\pi) \)
  – \( \text{prefix}(\pi) = 3 \)
• This results in an idea for a greedy algorithm: increase \( \text{prefix}(\pi) \) at every step

Greedy Algorithm: An Example

• Doing so, \( \pi \) can be sorted

\[
\begin{array}{cccccc}
1 & 2 & 3 & \underline{6} & 4 & 5 \\
1 & 2 & 3 & 4 & \underline{6} & 5 \\
1 & 2 & 3 & 4 & 5 & \underline{6}
\end{array}
\]

• Number of steps to sort permutation of length \( n \) is at most \( n - 1 \)
Greedy Algorithm: Pseudocode

\textbf{SimpleReversalSort}(\pi)
\begin{itemize}
  \item [1] \textbf{for} \ i \leftarrow 1 \text { to } n - 1 \n  \item [2] \ j \leftarrow \text{position of element } i \text{ in } \pi \text{ (i.e., } \pi_j = i) \n  \item [3] \textbf{if} \ j \neq i \n  \item [4] \pi \leftarrow \pi \* \rho(i, j) \n  \item [5] \textbf{output} \pi \n  \item [6] \textbf{if} \pi \text{ is the identity permutation} \n  \item [7] \textbf{return} \n\end{itemize}

Analyzing SimpleReversalSort

\begin{itemize}
  \item SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on \( \pi = 6 \ 1 \ 2 \ 3 \ 4 \ 5 \):
  \begin{itemize}
    \item Step 1: 1 6 2 3 4 5
    \item Step 2: 1 2 6 3 4 5
    \item Step 3: 1 2 3 6 4 5
    \item Step 4: 1 2 3 4 6 5
    \item Step 5: 1 2 3 4 5 6
  \end{itemize}
\end{itemize}
Analyzing SimpleReversalSort (cont’d)

• But it can be sorted in two steps:
  \[ \pi = 6 \ 1 \ 2 \ 3 \ 4 \ 5 \]
  – Step 1:  \[ 5 \ 4 \ 3 \ 2 \ 1 \ 6 \]
  – Step 2:  \[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \]
• So, SimpleReversalSort(\pi) is not optimal

• Optimal algorithms are unknown for many problems; approximation algorithms are used

Adjacencies and Breakpoints

\[ \pi = \pi_1 \pi_2 \pi_3 \ldots \pi_{n-1} \pi_n \]
• A pair of elements \( \pi_i \) and \( \pi_{i+1} \) are adjacent if
  \[ \pi_{i+1} = \pi_i \pm 1 \]
• For example:
  \[ \pi = 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5 \]
• (3, 4) or (7, 8) and (6,5) are adjacent pairs
Breakpoints: An Example

There is a breakpoint between any adjacent element that are non-consecutive:

\[ \pi = 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5 \]

• Pairs (1,9), (9,3), (4,7), (8,2) and (2,5) form breakpoints of permutation \( \pi \)
• \( b(\pi) \) - # breakpoints in permutation \( \pi \)

Extending Permutations

• We put two elements \( \pi_0 = 0 \) and \( \pi_{n+1} = n+1 \) at the ends of \( \pi \)

Example:

\[ \pi = 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5 \]

\[ \pi = 0 \ 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5 \ 10 \]

Note: A new breakpoint was created after extending
Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.

\[
\pi = 2\ 3\ 1\ 4\ 6\ 5
\]

\[
\begin{array}{c|cccccc}
0 & 2 & 3 & 1 & 4 & 6 & 5 \\
1 & 3 & 2 & 4 & 6 & 5 & 7 \\
0 & 1 & 2 & 3 & 4 & 6 & 5 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}
\]

\[
b(\pi) = 5 \quad b(\pi) = 4 \quad b(\pi) = 2 \quad b(\pi) = 0
\]
Sorting By Reversals: A Better Greedy Algorithm

BreakPointReversalSort(\(\pi\))
1 \textbf{while} \(b(\pi) > 0\)
2 Among all possible reversals, choose reversal \(\rho\) minimizing \(b(\pi \cdot \rho)\)
3 \(\pi \leftarrow \pi \cdot \rho(i, j)\)
4 \textbf{output} \(\pi\)
5 \textbf{return}

Problem: Will this algorithm terminate?
Strips

• **Strip**: an interval between two consecutive breakpoints in a permutation
  – **Decreasing strip**: strip of elements in decreasing order (e.g. 6 5 and 3 2).
  – **Increasing strip**: strip of elements in increasing order (e.g. 7 8)

A single-element strip can be declared either increasing or decreasing. We will choose to declare them as decreasing with exception of the strips with 0 and n+1.

Reducing the Number of Breakpoints

**Theorem 1:**
If permutation $\pi$ contains at least one decreasing strip, then there exists a reversal $\rho$ which decreases the number of breakpoints (i.e. $b(\pi \cdot \rho) < b(\pi)$)
Proof & Example

• For \( \pi = 1 \ 4 \ 6 \ 5 \ 7 \ 8 \ 3 \ 2 \)

| 0 | 1 | 4 | 6 | 5 | 7 | 8 | 3 | 2 | 9 |

\[ b(\pi) = 5 \]

– Choose decreasing strip with the smallest element \( k \) in \( \pi \) (\( k = 2 \) in this case)

Proof & Example

• For \( \pi = 1 \ 4 \ 6 \ 5 \ 7 \ 8 \ 3 \ 2 \)

| 0 | 1 | 4 | 6 | 5 | 7 | 8 | 3 | 2 | 9 |

\[ b(\pi) = 5 \]

– Choose decreasing strip with the smallest element \( k \) in \( \pi \) (\( k = 2 \) in this case)

– **Claim**: \( k - 1 \) must be at end of increasing strip
Proof & Example

• For \( \pi = 1 4 6 5 7 8 3 2 \):
  
  \[
  0 \ 1 \ 4 \ 6 \ 5 \ 7 \ 8 \ 3 \ 2 \ 9 \ \ b(\pi) = 5
  \]
  
  – Choose decreasing strip with the smallest element \( k \) in \( \pi \) (\( k = 2 \) in this case)
  – Reverse the segment between \( k \) and \( k-1 \):

  \[
  0 \ 1 \ 4 \ 6 \ 5 \ 7 \ 8 \ 3 \ 2 \ 9 \ \ b(\pi) = 5
  \]

  \[
  0 \ 1 \ 2 \ 3 \ 8 \ 7 \ 5 \ 6 \ 4 \ 9 \ b(\pi) = 4
  \]

Reducing the Number of Breakpoints

Again

What if no decreasing strip?

– There may be no reversal \( \rho \) that reduces the number of breakpoints;
  
i.e. \( b(\pi \ast \rho) \geq b(\pi) \) for any reversal \( \rho \).

– In this case, reverse an increasing strip:
  
  • # of breakpoints remains unchanged
  • Creates a decreasing strip at the next step.
  • Number of breakpoints will be reduced in the next step (Theorem 1).
Example

- There are no decreasing strips in $\pi$, for:
  
  $\pi = 0 \ 1 \ 2 \ 5 \ 6 \ 7 \ 3 \ 4 \ 8 \ b(\pi) = 3$
  
  $\pi \cdot \rho(6,7) = 0 \ 1 \ 2 \ 5 \ 6 \ 7 \ 4 \ 3 \ 8 \ b(\pi) = 3$

  $\checkmark \ \rho(6,7)$ does not change the # of breakpoints
  
  $\checkmark \ \rho(6,7)$ creates a decreasing strip thus guaranteeing that the next step will decrease the # of breakpoints.

ImprovedBreakpointReversalSort

\[
\text{ImprovedBreakpointReversalSort}(\pi)
\]

\[
\begin{align*}
1 \ & \text{while } b(\pi) > 0 \\
2 \ & \text{if } \pi \text{ has a decreasing strip} \\
3 \ & \quad \text{Among all possible reversals, choose reversal } \rho \\
4 \ & \quad \text{that minimizes } b(\pi \cdot \rho) \\
5 \ & \text{else} \\
6 \ & \quad \text{Choose a reversal } \rho \text{ that flips an increasing strip in } \pi \\
7 \ & \quad \pi \leftarrow \pi \cdot \rho \\
8 \ & \text{output } \pi \\
9 \ & \text{return}
\end{align*}
\]
**ImprovedBreakpointReversalSort: Performance Guarantee**

- *ImprovedBreakPointReversalSort* is an approximation algorithm
- **Claim**: Performance guarantee of at most 4
  - Eliminates at least one breakpoint in every two steps $\rightarrow$ at most $2b(\pi)$ steps
  - Approximation ratio: $2b(\pi) / d(\pi)$
  - Optimal algorithm eliminates at most 2 breakpoints in every step: $d(\pi) \geq b(\pi) / 2$
  - Performance guarantee:
    - $(2b(\pi) / d(\pi)) \leq [2b(\pi) / (b(\pi) / 2)] = 4$

---

**Breakpoint Graph**

1) Represent the elements of the permutation
   \[ \pi = 2 \ 3 \ 1 \ 4 \ 6 \ 5 \] as vertices in a graph (ordered along a line)
2) Connect vertices in order given by $\pi$ with black edges (black path)
3) Connect vertices in order given by 1 2 3 4 5 6 with grey edges (grey path)
4) Superimpose black and gray paths
Two Equivalent Representations of the Breakpoint Graph

Showing gray path instead of black path on a horizontal line, gives identical graph

What is the Effect of the Reversal?

How does a reversal change the breakpoint graph?
- The gray paths stayed the same for both graphs
- There is a change in the graph at this point
- There is another change at this point
- Other black edges are unaffected by the reversal so they remain the same for both graphs

Before: 0 2 3 1 4 6 5 7

After: 0 2 3 5 6 4 1 7
A reversal affects 4 edges in the breakpoint graph

• A reversal removes 2 edges (red) and replaces them with 2 new edges (blue)
• Gray edges unchanged

Maximum Cycle Decomposition

• Breakpoint graph can be decomposed into edge-disjoint alternating (gray-black) cycles.
• Let \( c(\pi) \) = number of alternating cycles in maximal decomposition

\[ c(\pi) = ? \quad 4 \]
• Since the identity permutation of size \( n \) contains the maximum cycle decomposition of \( n+1 \), \( c(\text{identity}) = n+1 \)
Effects of Reversals

Case 1:
Both edges belong to the same cycle
• Remove the center black edges and replace them with new black edges (there are two ways to replace them)
  • (a) After this replacement, there now exists 2 cycles instead of 1 cycle
    \[ c(\pi\rho) - c(\pi) = 1 \]
  • (b) Or after this replacement, there still exists 1 cycle
    \[ c(\pi\rho) - c(\pi) = 0 \]

Therefore, after the reversal
\[ c(\pi\rho) - c(\pi) = 0 \text{ or } 1 \]
This is called a proper reversal since there’s a cycle increase after the reversal.

Effects of Reversals (Continued)

Case 2:
Both edges belong to different cycles
• Remove the center black edges and replace them with new black edges
• After the replacement, there now exists 1 cycle instead of 2 cycles

\[ c(\pi\rho) - c(\pi) = -1 \]

Therefore, for every permutation \( \pi \) and reversal \( \rho \),
\[ | c(\pi\rho) - c(\pi) | \leq 1 \]
Reversal Distance and Maximum Cycle Decomposition

- Since the identity permutation of size n contains the maximum cycle decomposition of $n+1$, $c(\text{identity}) = n+1$

- $c(\text{identity}) - c(\pi)$ equals the number of cycles that need to be “added” to $c(\pi)$ while transforming $\pi$ into the identity

- Based on the previous theorem, at best after each reversal, the cycle decomposition could increased by one, then:
  \[
  d(\pi) = c(\text{identity}) - c(\pi) = n+1 - c(\pi)
  \]

- Yet, not every reversal can increase the cycle decomposition

  Therefore, $d(\pi) \geq n+1 - c(\pi)$

- Reversal distance problem is NP-hard (Caprara 1997)

Signed Permutations

- Up to this point, all permutations to sort were unsigned

- But genes (and DNA) have directions... so we should consider signed permutations

\[
\pi = 5' - 1 - 2 \cdot 3 - 4 - 5 3'
\]
The inversion/reversal introduced two breakpoints (disruptions in order).

Sorting by reversals: 5 steps

Step 0: \( \pi \)

Step 1:

Step 2:

Step 3:

Step 4:

Step 5: \( \gamma \)
Sorting by reversals: 4 steps

Step 0: $\pi$ 2 -4 -3 5 -8 -7 -6 1
Step 1: 2 3 4 5 -8 -7 -6 1
Step 2: -5 -4 -3 -2 -8 -7 -6 1
Step 3: -5 -4 -3 -2 -1 6 7 8
Step 4: $\gamma$ 1 2 3 4 5 6 7 8

What is the reversal distance for this permutation? Can it be sorted in 3 steps?
Sorting Permutations by Reversals
(Sankoff et al.1990)

\( \pi = \pi_1 \pi_2 ... \pi_n \) signed permutation

Reversal \( \rho(i,j) \) [inversion]

\( \pi_1 ... \pi_{i-1} -\pi_i ... -\pi_j \pi_{j+1} ... \pi_n \)

**Problem:** Given \( \pi \), find a sequence of reversals \( \rho_1, ..., \rho_t \) with such that:
\( \pi \cdot \rho_1 \cdot \rho_2 ... \rho_t = (1, 2, ..., n) \) and \( t \) is minimal.

**Solution:** Analysis of breakpoint graph

Polynomial time algorithms

---

From Signed to Unsigned Permutation

- Begin by constructing a normal signed breakpoint graph
- Redefine each vertex \( x \) with the following rules:
  - If vertex \( x \) is positive, replace vertex \( x \) with vertex \( 2x-1 \) and vertex \( 2x \) in that order
  - If vertex \( x \) is negative, replace vertex \( x \) with vertex \( 2x \) and vertex \( 2x-1 \) in that order
  - The extension vertices \( x = 0 \) and \( x = n+1 \) are kept as it was before

![Diagram of breakpoint graph transformation](image)
From Signed to Unsigned Permutation (Continued)

- Construct the breakpoint graph as usual
- Notice the alternating cycles in the graph between every other vertex pair
- Since these cycles came from the same signed vertex, we will not be performing any reversal on both pairs at the same time; therefore, these cycles can be removed from the graph

Increasing Number of Cycles

Proper reversal acts on black edges: \( c(\rho \pi) - c(\pi) = 1 \)
Breakpoint graph
⇒ rearrangement scenario

Oriented and Unoriented Cycles

- **Oriented Cycles**

  ![Oriented Cycle Diagram]

  Proper reversal acts on black edges: $c(\rho \pi) - c(\pi) = 1$

- **Unoriented Cycles**

  ![Unoriented Cycle Diagram]

  No proper reversal acting on an unoriented cycle

  These are “impediments” in sorting by reversals.
Reversal Distance with Hurdles

- Hurdles are obstacles in the genome rearrangement problem
- They cause a higher number of required reversals for a permutation to transform into the identity permutation

\[ h(\pi) = 1 \]

Every hurdle can be transformed into oriented cycles by reversal on arbitrary cycle in hurdle.

Safe Reversals

Let \( \Delta c = c(\rho \pi) - c(\pi) \) \( \Delta h(\rho \pi) - h(\pi) \)

A reversal \( p \) is safe if \( \Delta c - \Delta h = 1 \).

- Oriented Cycles

Proper reversal acts on black edges: \( c(\rho \pi) - c(\pi) = 1 \)

- Unoriented Cycles

\[ c(\pi) = 2, h(\pi) = 1 \]
\[ c(\pi) = 2, h(\pi) = 0 \]
Interleaving Edges

- Interleaving edges are grey edges that cross each other

*Example: Edges (0,1) and (18, 19) are interleaving*

- Cycles are interleaving if they have an interleaving edge

Interleaving Graphs

An Interleaving Graph is defined on the set of cycles in the Breakpoint graph and are connected by edges where cycles are interleaved.
Interleaving Graphs

Label oriented cycles. Component oriented if contains oriented cycle.

Interleaving Graphs

Remove oriented components from interleaving graph.
Hurdles

Particular type of oriented component

Define: Containment partial order on unoriented components.

\[ U < W \text{ if } [U_{\text{min}}, U_{\text{max}}] \text{ contained in } [W_{\text{min}}, W_{\text{max}}] \]

Hurdle: Minimal or maximal element under partial order:

Red and Green unoriented components
Reversal Distance with Hurdles

- Hurdles are obstacles in the genome rearrangement problem
- They cause a higher number of required reversals for a permutation to transform into the identity permutation
- Let $h(\pi)$ be the number of hurdles in permutation $\pi$
- Taking into account of hurdles, the following formula gives a tighter bound on reversal distance:

$$d(\pi) \geq n + 1 - c(\pi) + h(\pi)$$

Every hurdle can be transformed into oriented cycles by reversal on arbitrary cycle in hurdle.

** Doing so, might cause problems with overlapping hurdles

Superhurdles

- “Protect” non-hurdles
  - Deletion of superhurdles creates another hurdle
Superhurdles

• “Protect” non-hurdles
  – Deletion of superhurdles creates another hurdle
Fortresses

- A permutation $\pi$ with an odd number of hurdles, all of which are superhurdles

**Theorem (Hannenhalli-Pevzner 1995):**

\[ d(\pi) = n + 1 - c(\pi) + h(\pi) + f(\pi) \]

where $c = \#$ cycles; $h = \#$ hurdles

$f = 1$ if $\pi$ is fortress.

---

Complexity of reversal distance

**Approximate solutions:**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Kececioglu &amp; Sankoff</td>
<td>2-approximation algorithm</td>
</tr>
<tr>
<td>(1992) (CPM '93)</td>
<td></td>
</tr>
<tr>
<td>Bafna &amp; Pevzner (FOCS '93)</td>
<td>1.5-approximation algorithm and introduced breakpoint graph</td>
</tr>
</tbody>
</table>

**Exact solutions:**

<table>
<thead>
<tr>
<th></th>
<th>Compute distance</th>
<th>Exhibit steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hannenhalli &amp; Pevzner (STOC '95)</td>
<td>$O(n^2)$</td>
<td>$O(n^4)$</td>
</tr>
<tr>
<td>Berman &amp; Hannenhalli (CPM '96)</td>
<td>$O(n^{\alpha}(n))$</td>
<td>$O(n^{2\alpha}(n))$</td>
</tr>
<tr>
<td>Kaplan, Shamir, Tarjan (SODA '97)</td>
<td>$O(n^{\alpha}(n) + d\cdot n)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Bader, Moret, Yan (WADS '01)</td>
<td>$O(n)$</td>
<td>N/A</td>
</tr>
</tbody>
</table>

$n = \#$ of blocks $\alpha(n)$ is the "Inverse Ackerman function"
**Multichromosomal Sorting**

- Concatenate chromosomes
- Translocations modeled by reversals in concatenate

![Diagram showing concatenation, reversal, and translocation of chromosomes]

**Additional Problems**

1. Other rearrangement operations
   - Duplications
2. Rearrangements and Phylogeny
   - **Multiple Genomic Distance Problem**: Given permutations $\pi^1, \ldots, \pi^k$ find a permutation $\sigma$ such that
   $$\sum_{k=1}^{k} d(\pi^1, \sigma)$$ is minimal.