

---

# Greedy Algorithms And Genome Rearrangements

---

---

# Outline

- Comparing genomes
  - Transforming cabbage into turnip
  - Genome rearrangements
  - Sorting by reversals
  - Pancake flipping problem
  - Greedy algorithm for sorting by reversals
  - Approximation algorithms
  - Breakpoints: a different face of greed
  - Breakpoint graphs
-

---

# Comparing genome

- **What is a genome?** The complete inventory of all heritable nucleic acids that determines the genetic identity of an organism is called **genome**.
    - **Viral genomes** – DNA and RNA viruses.
    - **Bacteria** – Circular DNA.
    - **Eukaryotes** – distributed over linear DNA pieces (chromosomes).
-

---

# Types of comparison

- **Within-genome comparisons** focus on the genome of a single species. Variations on base composition,  $k$ -tuple frequency, gene density, numbers and kinds of transposable elements and segmental duplications.
  - **Between-genome comparisons** employ closely related species for identifying conserved genes, gene structure and organization and control elements. More distantly related species are used for phylogenetic profiling.
-

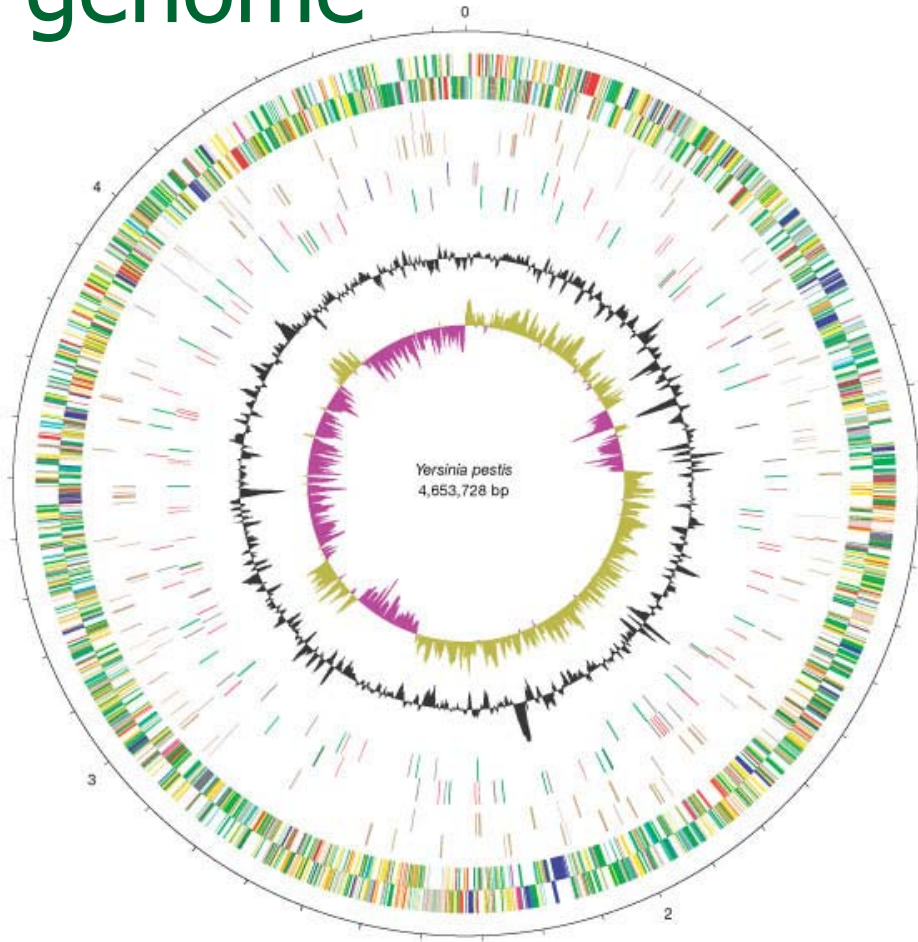
# Compositional measures

- $k$ -tuple compositions of genomes are not uniformly distributed along the genome.
- Take  $k = 1$  as an example: In the human genome, gene-rich regions typically have a higher %**G+C** content than gene-poor regions.
- A statistics for prokaryotic genomes is the **GC skew**:

$$GC_w = \frac{\#G_w - \#C_w}{\#G_w + \#C_w}$$

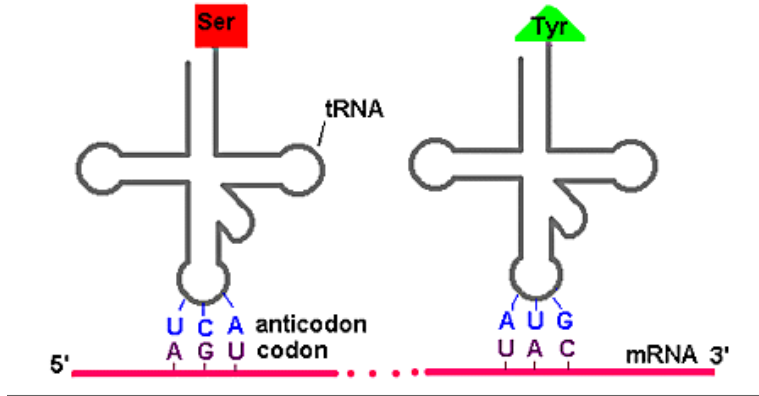
where  $w$  is a sequence window.

# Properties of the *Yersinia pestis*<sup>(mor)</sup> genome

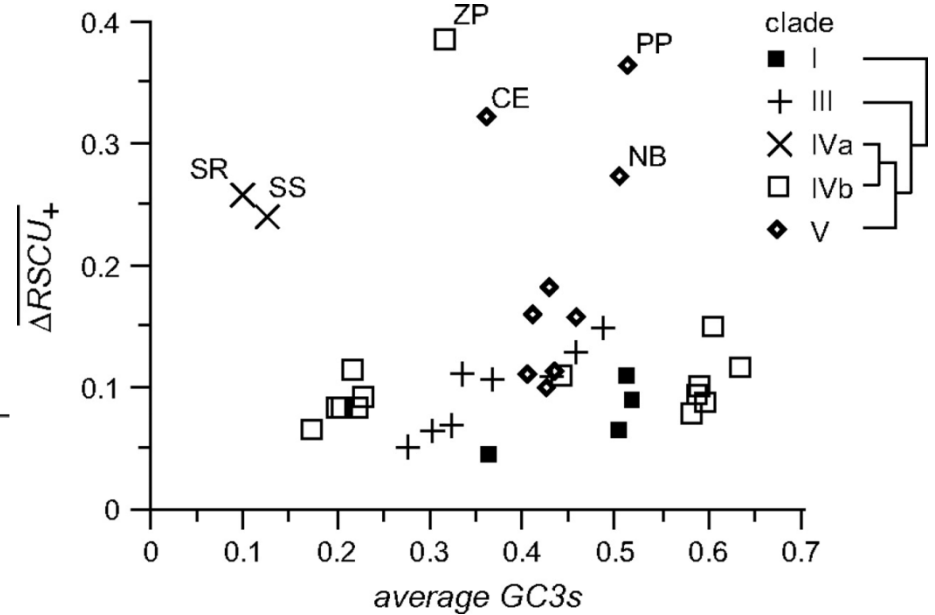


The innermost circle represents GC skew. There are regions where the GC skew reverses sign. This indicates recent inversions.

# Codon usage



		2nd base in codon					
		U	C	A	G		
1st base in codon	U	Phe Phe Leu Leu	Ser Ser Ser Ser	Tyr Tyr STOP STOP	Cys Cys STOP Trp	U C A G	3rd base in codon
	C	Leu Leu Leu Leu	Pro Pro Pro Pro	His His Gln Gln	Arg Arg Arg Arg	U C A G	
	A	Ile Ile Ile Met	Thr Thr Thr Thr	Asn Asn Lys Lys	Ser Ser Arg Arg	U C A G	
	G	Val Val Val Val	Ala Ala Ala Ala	Asp Asp Glu Glu	Gly Gly Gly Gly	U C A G	



Differences among species in selection on codon usage. Average of positive  $\Delta RSCU$  values (differences in codon usage for highly and lowly expressed genes) per species indicate that 6 species have particularly strong selection on codon bias, spanning low, medium, and high GC-content genomes. Symbols indicate different clades within the nematode phylogeny.

## The Genetic Code

# Gene content

- Prerequisite: Gene annotation (usually via HMMs; will be later).
  - Total no. of predicted genes
  - No. of genes duplicated
  - Total no. of distinct families

	<i>H. influenzae</i>	<i>S. cerevisiae</i>	<i>C. elegans</i>	<i>D. melanogaster</i>
Total no. of predicted genes	1709	6241	18424	13601
No. of genes duplicated <sup>1</sup>	284	1858	8971	5536
Total no. of distinct families <sup>2</sup>	1425	4383	9453	8065



# Gene content

- Coding or non-coding sequence (RNA versus protein coding genes).
  - Number of genes
  - % of coding
  - Gene size (average bp)
  - Exon size (average bp)
  - Exons/gene (average)
  - ...

	<i>S. cerevisiae</i>	<i>C. elegans</i>	<i>D. melanogaster</i>	<i>A. thaliana</i>	<i>H. sapiens</i>
No. of genes	5,500	18,400	13,600	26,400	25,000
% Coding	70	27	20	26.3	1.2
Gene size (average bp)	1450	2700	3250	1970	27,000
Exon size (average bp)	1450	240	425	164	145
Exons/gene (average)	1	6	4	5.2	9.5

# Clustering Gene Content

- Vectors of the same length could be clustered.
- Based on a  $n \times n$  distance matrix, a hierarchical clustering can be performed. This method builds the hierarchy from the individual elements by progressively merging "closest" clusters. Distance between clusters  $A$ ,  $B$  can be computed in various ways:
  - complete linkage clustering:  $\max \{ d(x, y) \mid x \in A, y \in B \}$
  - single linkage clustering:  $\min \{ d(x, y) \mid x \in A, y \in B \}$
  - average linkage clustering:

$$\frac{1}{\text{card}(A) \text{card}(B)} \sum_{x \in A} \sum_{y \in B} d(x, y)$$

# Turnip vs. Cabbage: Look and Taste Different

- Although cabbages and turnips share a recent common ancestor, they look and taste different



---

# Turnip vs Cabbage: Almost Identical mtDNA gene sequences

- In 1980s Jeffrey Palmer studied evolution of plant organelles by comparing mitochondrial genomes of the cabbage and turnip
  - 99% similarity between genes
  - These surprisingly identical gene sequences **differed in gene order**
  - This study helped pave the way to analyzing genome rearrangements in molecular evolution
-

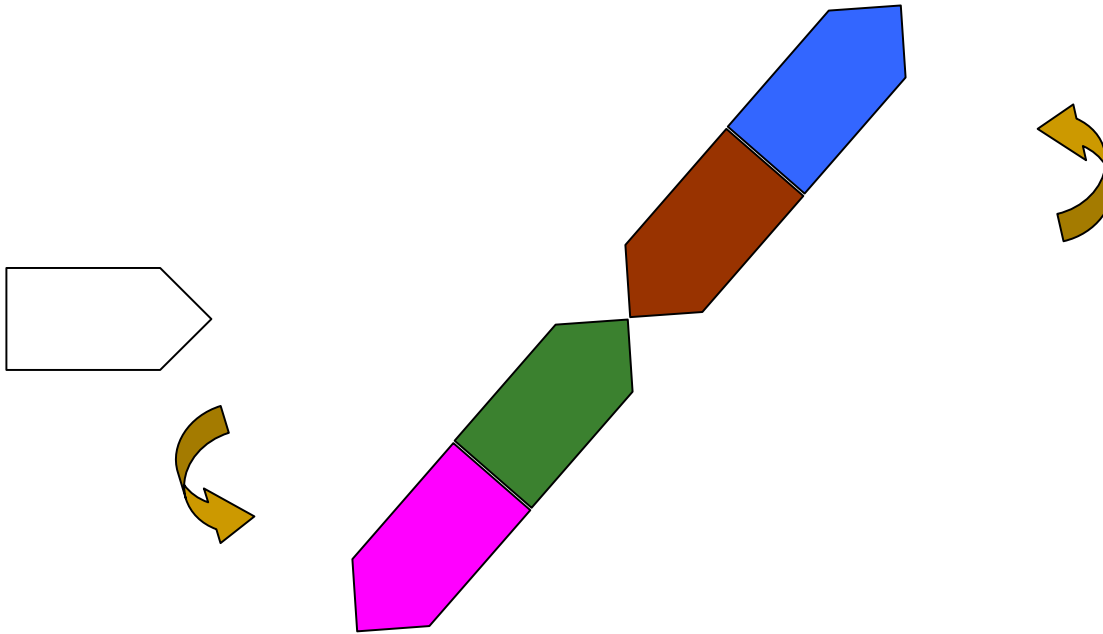
# Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:



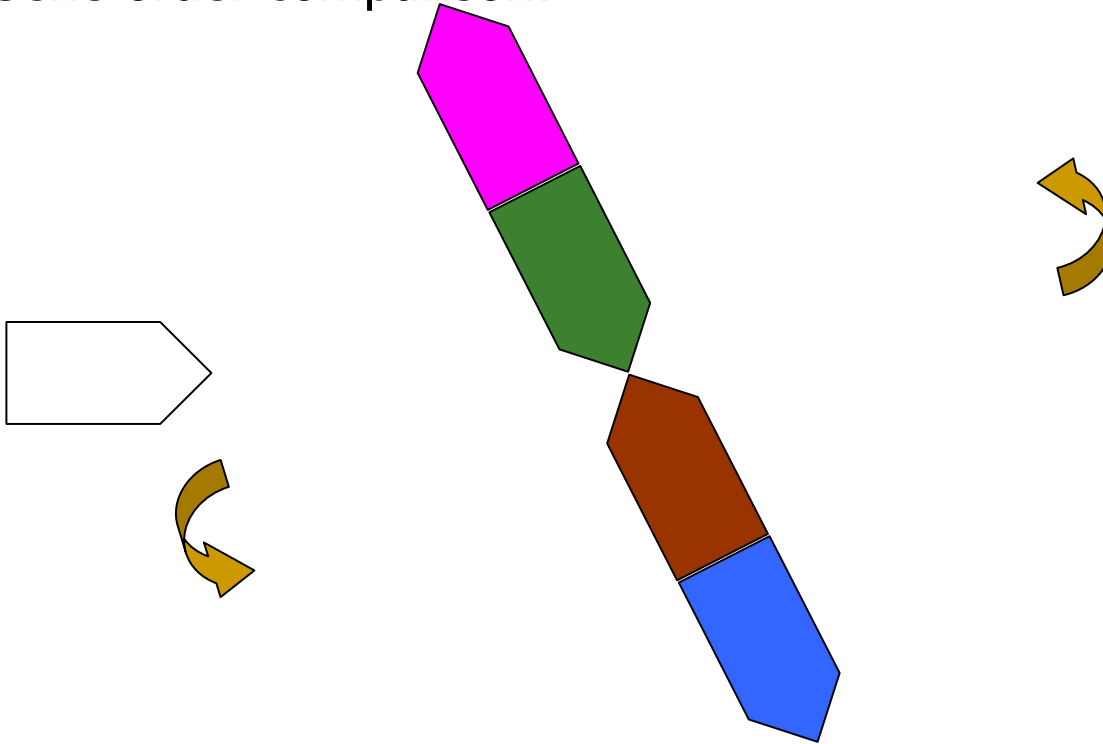
# Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:



# Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:



# Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:





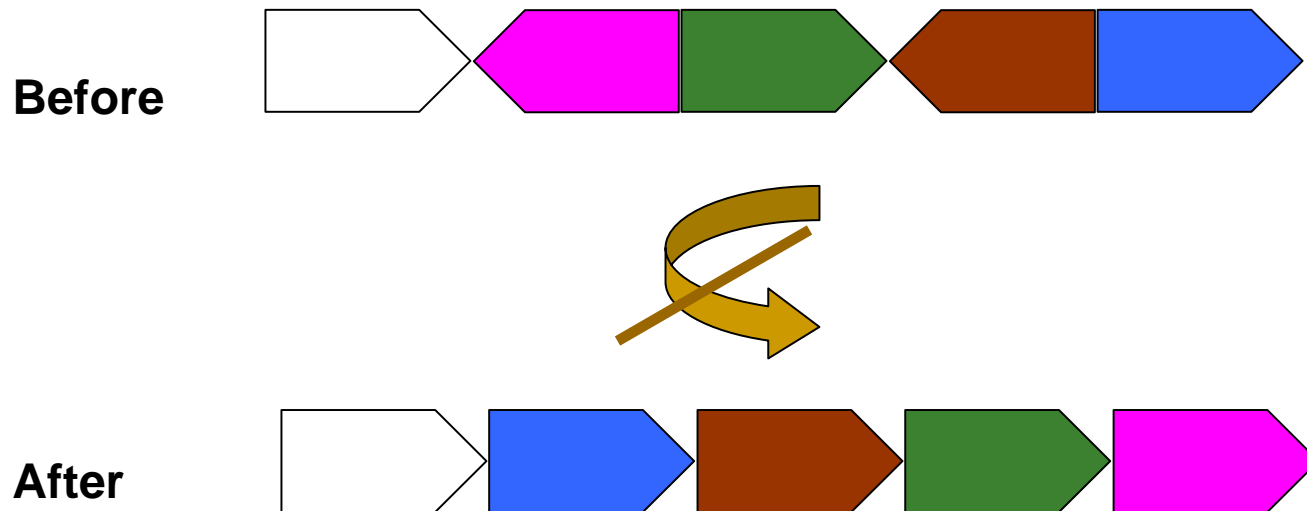
# Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:



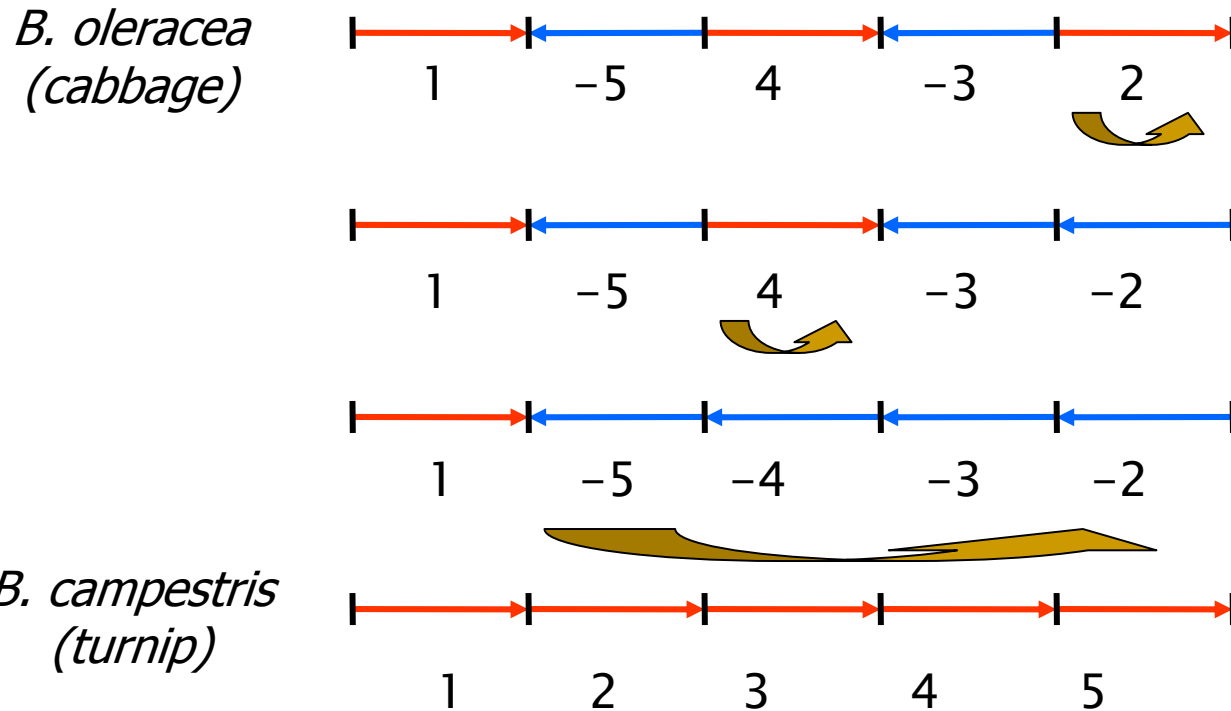
# Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:

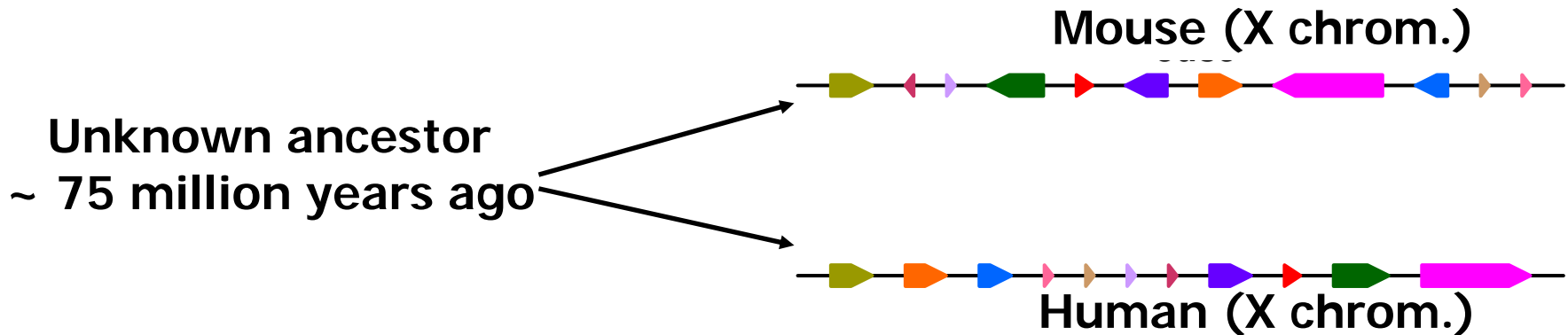


- Evolution is manifested as the divergence in gene order

# Transforming Cabbage into Turnip

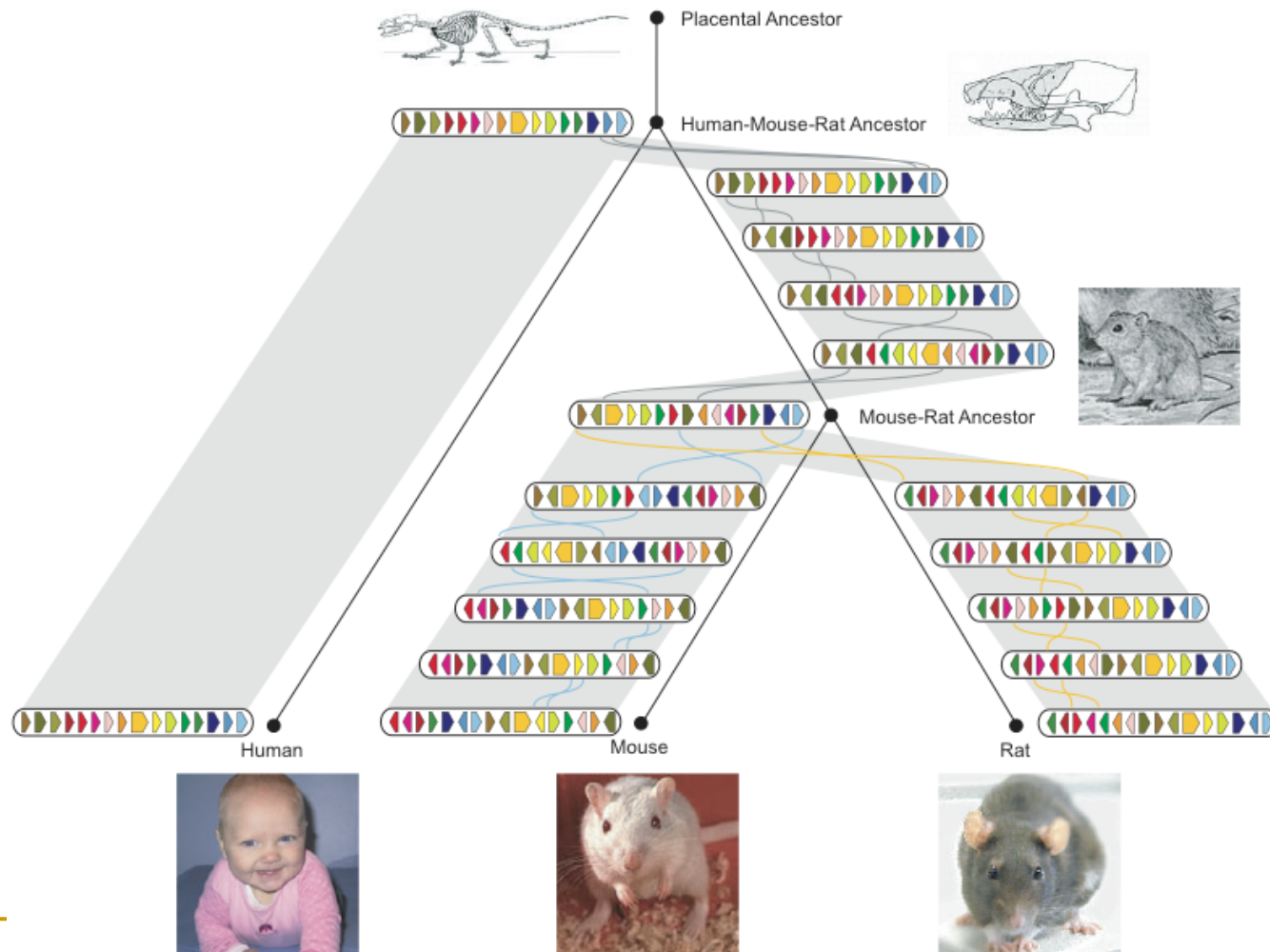


# Genome rearrangements

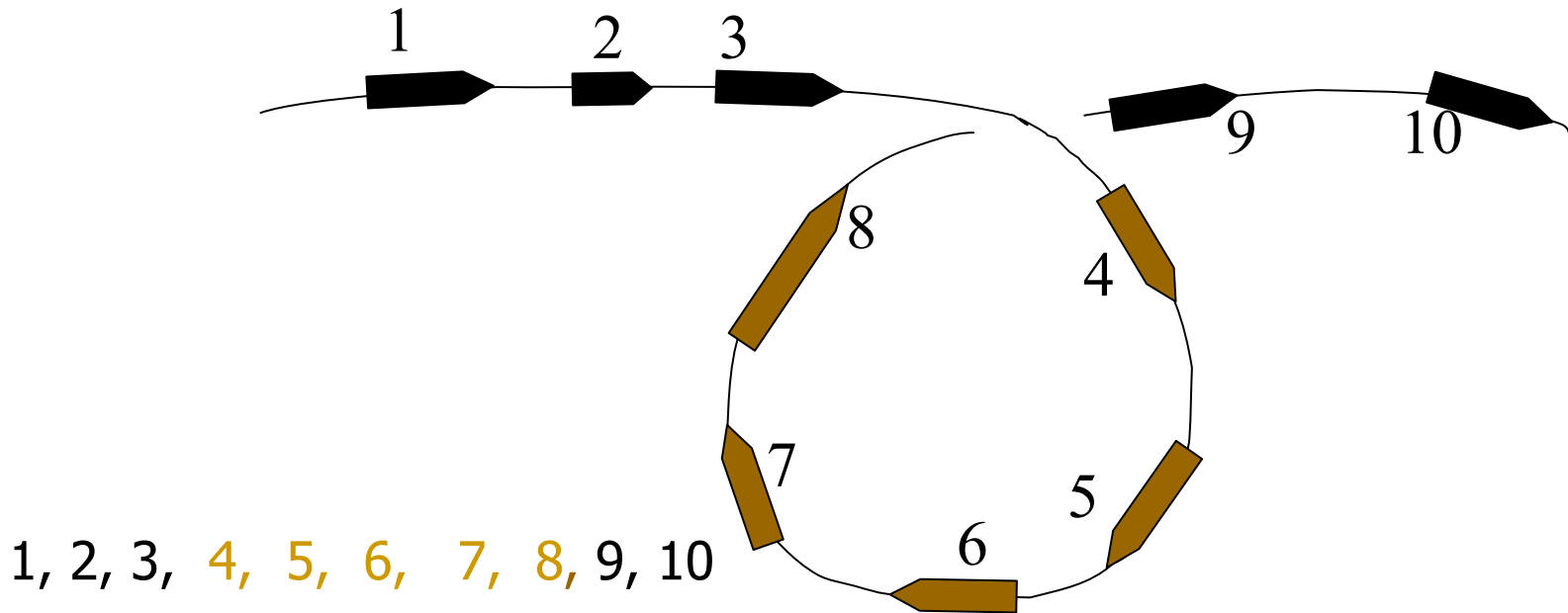


- What are the similarity blocks and how to find them?
- What is the architecture of the ancestral genome?
- What is the evolutionary scenario for transforming one genome into the other?

# History of Chromosome X

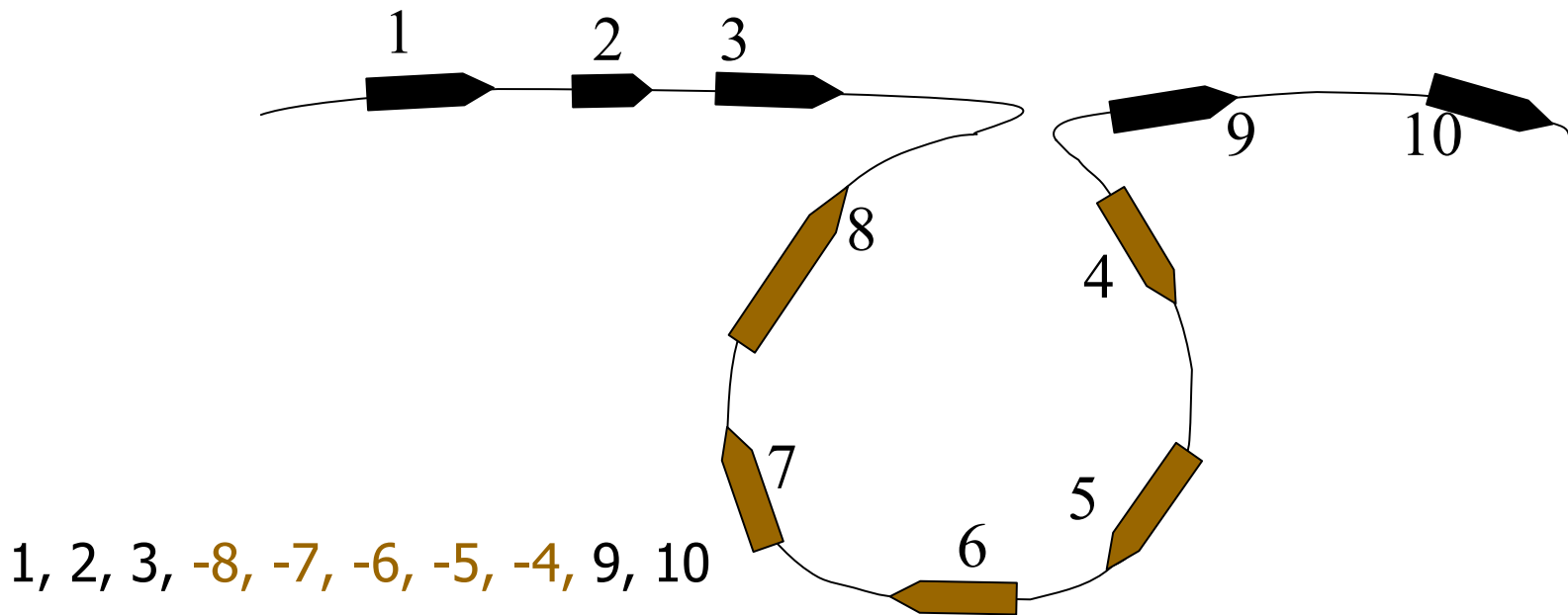


# Reversals



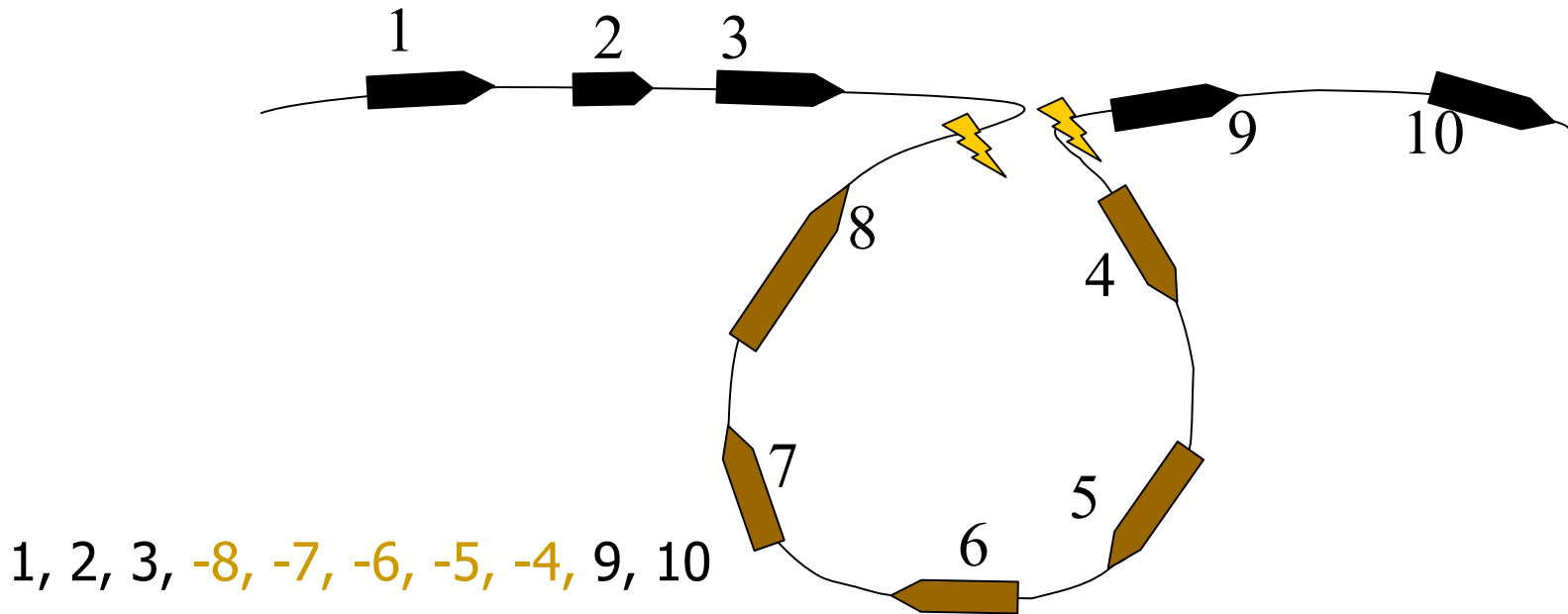
- Blocks represent conserved genes.

# Reversals



- Blocks represent conserved genes.
- In the course of evolution or in a clinical context, blocks 1,...,10 could be misread as 1, 2, 3, -8, -7, -6, -5, -4, 9, 10.

# Reversals and Breakpoints



The reversion introduced two *breakpoints* (disruptions in order).

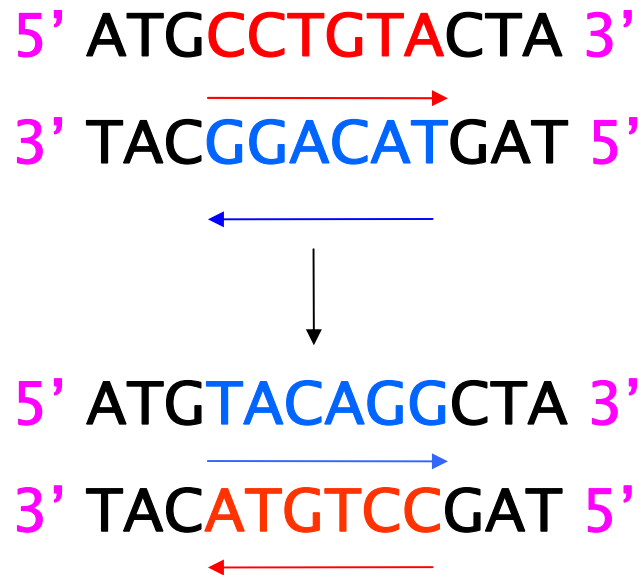
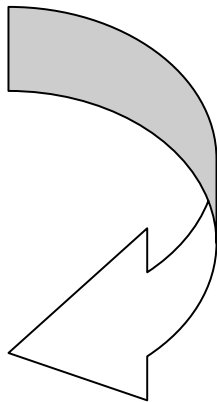




# Reversals: Example



Break  
and  
Invert



# Types of Rearrangements



Reversal

1 2 3 4 5 6  1 2 -5 -4 -3 6

Translocation

1 2 3  
4 5 6  1 2 6  
4 5 3

Fusion

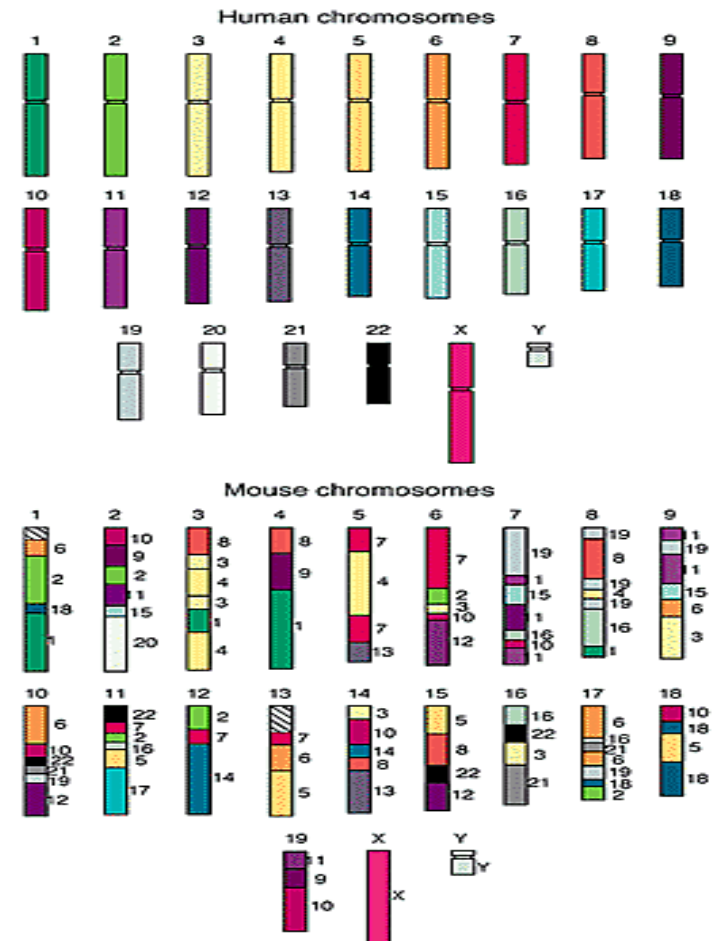
1 2 3 4  
5 6   1 2 3 4 5 6

Fission

štiepenie

# Comparative Genomic Architectures: Mouse vs Human Genome

- Humans and mice have similar genomes, but their genes are ordered differently
- ~245 rearrangements
  - Reversals
  - Fusions
  - Fissions
  - Translocation



# Waardenburg's Syndrome: Mouse Provides Insight into Human Genetic Disorder

- Waardenburg's syndrome is characterized by pigmentary dysphasia
- Gene implicated in the disease was linked to human chromosome 2 but it was not clear where exactly it is located on chromosome 2



---

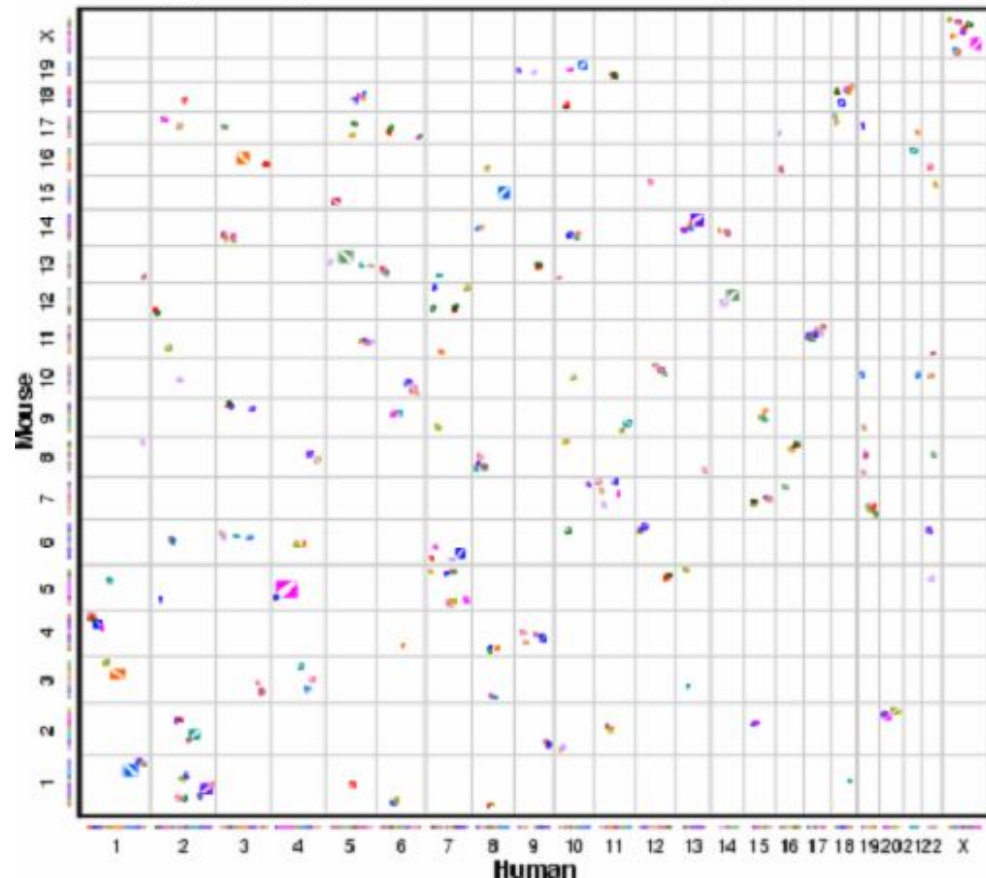
# Waardenburg's syndrome and splotch mice

- A breed of mice (with splotch gene) had similar symptoms caused by the same type of gene as in humans
  - Scientists succeeded in identifying location of gene responsible for disorder in mice
  - Finding the gene in mice gives clues to where the same gene is located in humans
-

# Comparative Genomic Architecture of Human and Mouse Genomes

To locate where corresponding gene is in humans, we have to analyze the relative architecture of human and mouse genomes

(b) Arrangement of human and mouse synteny blocks



# Reversals and Gene Orders

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

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

$$\rho(i, j) \downarrow$$

$$\pi_1 \dots \pi_{i-1} \pi_j \pi_{j-1} \dots \pi_{i+1} \pi_i \pi_{i+1} \dots \pi_n$$

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

# Reversals: Example

$$\begin{array}{cccccccc} \pi = & 1 & 2 & \underline{3} & \underline{4} & \underline{5} & 6 & 7 & 8 \\ & & & \downarrow & & & & & \\ & 1 & 2 & 5 & 4 & 3 & 6 & 7 & 8 \end{array}$$



# Reversals: Example

$\pi = 1\ 2\ \underline{3\ 4\ 5}\ 6\ 7\ 8$

$\rho(3,5)$



1 2 5 4 3 6 7 8

$\rho(5,6)$



1 2 5 4 **6 3** 7 8

# 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  $\rho_1, \dots, \rho_t$  transforming  $\pi$  into  $\sigma$ , such that  $t$  is minimum
- $t$  - reversal distance between  $\pi$  and  $\sigma$
- $d(\pi, \sigma)$  - smallest possible value of  $t$ , given  $\pi$  and  $\sigma$

# Sorting By Reversals Problem

- **Goal**: Given a permutation, find a shortest series of reversals that transforms it into the identity permutation (  $1\ 2\ \dots\ n$  )
  - **Input**: permutation  $\pi$
  - **Output**: a series of reversals  $\rho_1, \dots, \rho_t$  transforming  $\pi$  into the identity permutation such that  $t$  is minimum
  - $t = d(\pi)$  - reversal distance of  $\pi$
-

# Sorting By Reversals: Example

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

$$\begin{array}{rcccccccccc} \pi & = & \underline{3} & \underline{4} & 2 & 1 & 5 & 6 & 7 & 10 & 9 & 8 \\ & & 4 & 3 & 2 & 1 & 5 & 6 & 7 & \underline{10} & \underline{9} & \underline{8} \\ & & \underline{4} & \underline{3} & \underline{2} & \underline{1} & 5 & 6 & 7 & 8 & 9 & 10 \\ & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array}$$

So  $d(\pi) = 3$

**Question:** how to find  $d(\pi)$

# 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  $prefix(\pi)$   
 $prefix(\pi) = 3$
- This results in an idea for a greedy algorithm: increase  $prefix(\pi)$  at every step
- $1\ 2\ 3\ \underline{6}\ 4\ 5 \rightarrow 1\ 2\ 3\ 4\ \underline{6}\ 5 \rightarrow 1\ 2\ 3\ 4\ 5\ 6$
- Number of steps to sort permutation of length  $n$  is at most  $(n - 1)$

# Analyzing SimpleReversalSort

- **SimpleReversalSort** does not guarantee the smallest number of reversals and takes 5 steps on  $\pi = 6\ 1\ 2\ 3\ 4\ 5$  :

Step 1: 1 6 2 3 4 5

Step 2: 1 2 6 3 4 5

Step 3: 1 2 3 6 4 5

Step 4: 1 2 3 4 6 5

Step 5: 1 2 3 4 5 6

Step 1: 5 4 3 2 1 6

Step 2: 1 2 3 4 5 6

**SimpleReversalSort**

5 steps

optimal solution – 2 steps

So, **SimpleReversalSort( $\pi$ )** is not optimal

# Approximation Algorithms

- Optimal algorithms are unknown for many problems; approximation algorithms are used.
- These algorithms find approximate solutions rather than optimal solutions.
- The approximation ratio of an algorithm  $A$  on input  $\pi$  is:

$$A(\pi) / OPT(\pi)$$

where

$A(\pi)$  - solution produced by algorithm  $A$

$OPT(\pi)$  - optimal solution of the problem

---

# Approximation Ratio/Performance Guarantee

- Approximation ratio (**performance guarantee**) of algorithm  $A$ : max approximation ratio of all inputs of size  $n$ 
  - For algorithm  $A$  that **minimizes** objective function (minimization algorithm):
    - $\max_{|\pi| = n} A(\pi) / OPT(\pi)$



# Approximation Ratio/Performance Guarantee

- Approximation ratio (**performance guarantee**) of algorithm  $A$ : max approximation ratio of all inputs of size  $n$ 
  - For algorithm  $A$  that minimizes objective function (minimization algorithm):

$$\max_{|\pi| = n} A(\pi) / OPT(\pi)$$

- For **maximization** algorithm:

$$\min_{|\pi| = n} A(\pi) / OPT(\pi)$$

# Approximation Ratio/Performance Guarantee

- Approximation ratio (**performance guarantee**) of algorithm  $A$ : max approximation ratio of all inputs of size  $n$ 
  - For algorithm  $A$  that minimizes objective function (minimization algorithm):

$$\max_{|\pi| = n} A(\pi) / OPT(\pi)$$

- For **maximization** algorithm:

$$\min_{|\pi| = n} A(\pi) / OPT(\pi)$$

- For  $A = \text{SimpleReversalSort}()$

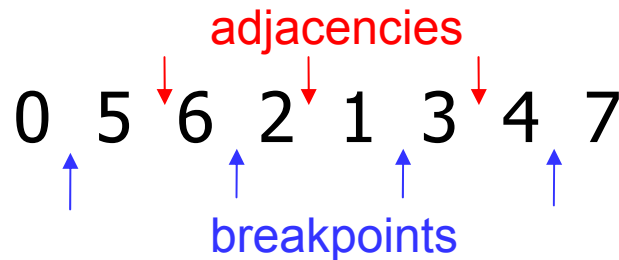
$$\max_{|\pi| = n} \text{SimpleReversalSort}(\pi) / OPT(\pi) \geq (n-1) / 2$$

# Adjacency & Breakpoints

$$\pi = \pi_1 \pi_2 \pi_3 \dots \pi_{n-1} \pi_n$$

- An **adjacency** – a pair of adjacent elements  $\pi_i$  and  $\pi_{i+1}$  that are **consecutive**  $\pi_{i+1} = \pi_i \pm 1$
- A **breakpoint** – a pair of adjacent elements that are **not consecutive**

$\pi = 5 \ 6 \ 2 \ 1 \ 3 \ 4$        $\longrightarrow$       Extend  $\pi$  with  $\pi_0 = 0$  and  $\pi_7 = 7$



# Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.

$\pi = 2\ 3\ 1\ 4\ 6\ 5$

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

$b(\pi) = 5$

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

$b(\pi) = 4$

0 | 1 2 3 4 | 6 5 | 7

$b(\pi) = 2$

0 | 1 2 3 4 5 6 | 7

$b(\pi) = 0$

# Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.
- This implies:

$$\text{reversal distance} \geq \# \text{breakpoints} / 2$$

$$\pi = 2 \ 3 \ 1 \ 4 \ 6 \ 5$$

$$0 \ | \ \underline{2 \ 3} \ | \ 1 \ | \ 4 \ | \ 6 \ 5 \ | \ 7$$

$$b(\pi) = 5$$

$$0 \ 1 \ | \ \underline{3 \ 2} \ | \ 4 \ | \ 6 \ 5 \ | \ 7$$

$$b(\pi) = 4$$

$$0 \ 1 \ 2 \ 3 \ 4 \ | \ \underline{6 \ 5} \ | \ 7$$

$$b(\pi) = 2$$

$$0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7$$

$$b(\pi) = 0$$

# Sorting By Reversals: A Better Greedy Algorithm

## BreakPointReversalSort( $\pi$ )

**while**  $b(\pi) > 0$

    Among all possible reversals, choose reversal  $\rho$

        minimizing  $b(\pi \circ \rho)$

$\pi \leftarrow \pi \circ \rho(i, j)$

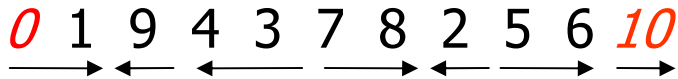
**output**  $\pi$

**return**

**Problem:** how good it approximates  $d(\pi)$

# 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.  $b(\pi \circ \rho) < b(\pi)$ ).



# Things To Consider

- For  $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$   
 $0\ 1\ |4|\ 6\ 5\ |7\ 8\ |3\ 2\ |9\ \quad b(\pi) = 5$
- Choose decreasing strip with the smallest element  $k$  in  $\pi$   
(  $k = 2$  in this case)

# Things To Consider (cont'd)

- For  $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$   
 $0\ 1\ |4|\ 6\ 5\ |7\ 8\ |3\ 2\ |9\ \quad b(\pi) = 5$
- Choose decreasing strip with the smallest element  $k$  in  $\pi$   
(  $k = 2$  in this case)

# Things To Consider (cont'd)

- For  $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$   
 $0\ 1\ | 4\ | 6\ 5\ | 7\ 8\ | 3\ 2\ | 9\ \quad b(\pi) = 5$
- Choose decreasing strip with the smallest element  $k$  in  $\pi$   
(  $k = 2$  in this case)
- Find  $k - 1$  in the permutation – it is in an increasing strip!
  - Where are breakpoints adjacent to  $k$  and  $k - 1$  ?


# Things To Consider (cont'd)

- For  $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$

$$0\ 1\ |4|\ 6\ 5\ |7\ 8|\ 3\ 2\ |9\quad b(\pi) = 5$$

- Choose decreasing strip with the smallest element  $k$  in  $\pi$  ( $k = 2$  in this case)
- Find  $k - 1$  in the permutation – it is in an increasing strip!
  - Where are breakpoints adjacent to  $k$  and  $k - 1$  ?

- Reverse the segment between  $k$  and  $k - 1$ :

$$0\ 1\ |4|\ 6\ 5\ |7\ 8|\ 3\ 2\ |9\quad b(\pi) = 5$$


$$0\ 1\ 2\ 3\ |8\ 7|\ 5\ 6\ |4\ |9\quad b(\pi) = 4$$

# Reducing the Number of Breakpoints Again

- If there is no decreasing strip, there may be no reversal  $\rho$  that reduces the number of breakpoints (i.e.  $b(\pi \circ \rho) \geq b(\pi)$  for any reversal  $\rho$ ).
- By reversing an increasing strip (# of breakpoints stay unchanged), we will create a decreasing strip at the next step. Then the number of breakpoints will be reduced in the next step (**Theorem 1**).

# Things To Consider (cont'd)

- There are no decreasing strips in  $\pi$ , for:

$$\pi = 0 \ 1 \ 2 \mid 5 \ 6 \ 7 \mid 3 \ 4 \mid 8 \quad b(\pi) = 3$$

$\rho(6,7)$

$$\pi = 0 \ 1 \ 2 \mid 5 \ 6 \ 7 \mid 4 \ 3 \mid 8 \quad b(\pi) = 3$$

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

# ImprovedBreakpointReversalSort

## ImprovedBreakpointReversalSort( $\pi$ )

**while**  $b(\pi) > 0$

**if**  $\pi$  has a decreasing strip

Among all possible reversals, choose reversal  $\rho$  that minimizes  $b(\pi \circ \rho)$

**else**

Choose a reversal  $\rho$  that flips an increasing strip in  $\pi$

$\pi \leftarrow \pi \circ \rho$

**output**  $\pi$

**return**

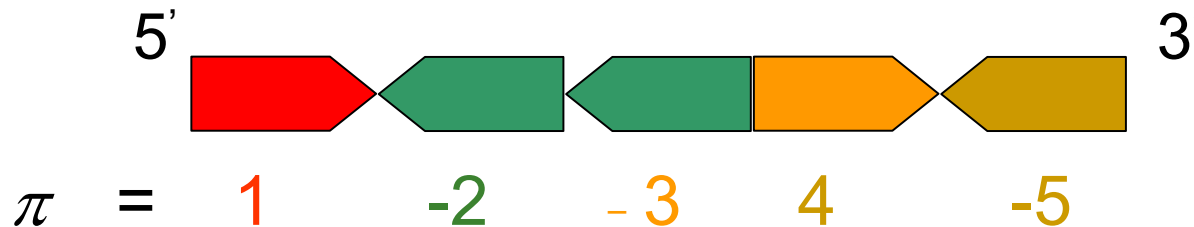
# ImprovedBreakpointReversalSort: Performance Guarantee

- **ImprovedBreakPointReversalSort** is an approximation algorithm with a performance guarantee of at most 4
  - It eliminates at least one breakpoint in every two steps; 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:  
 $(2b(\pi) / d(\pi)) \geq [2b(\pi) / (b(\pi) / 2)] = 4$



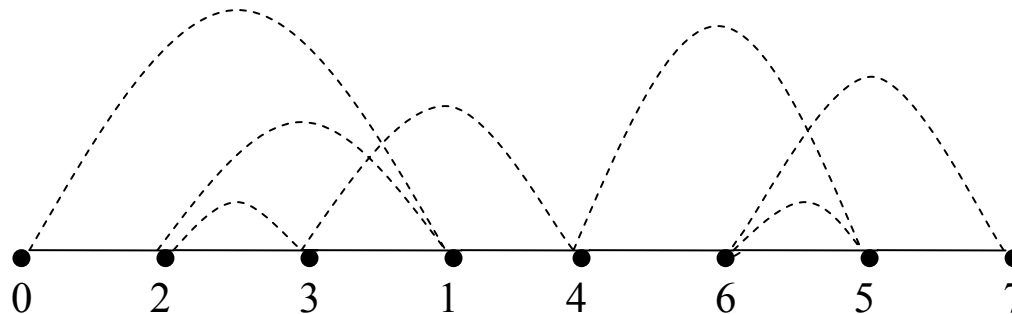
# Signed Permutations

- Up to this point, all permutations to sort were unsigned
- But genes have directions... so we should consider signed permutations



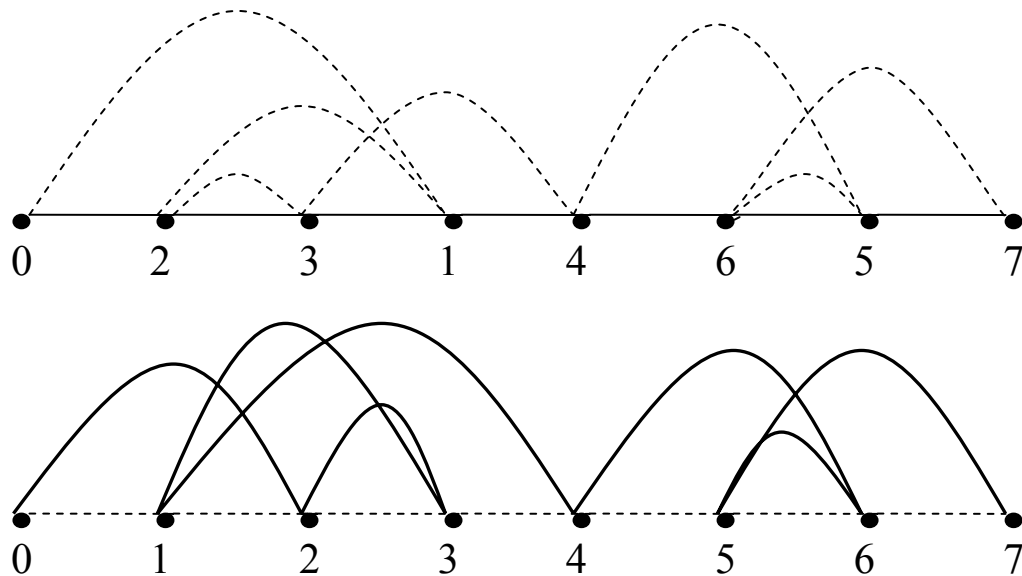
# Breakpoint Graph

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



# Two Equivalent Representations of the Breakpoint Graph

- Consider the following Breakpoint Graph
- If we line up the gray path (instead of black path) on a horizontal line, then we would get the following graph
- Although they may look different, these two graphs are the same

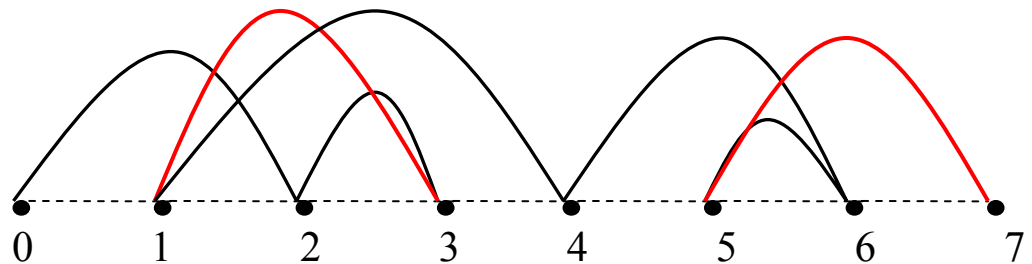


# What is the Effect of the Reversal ?

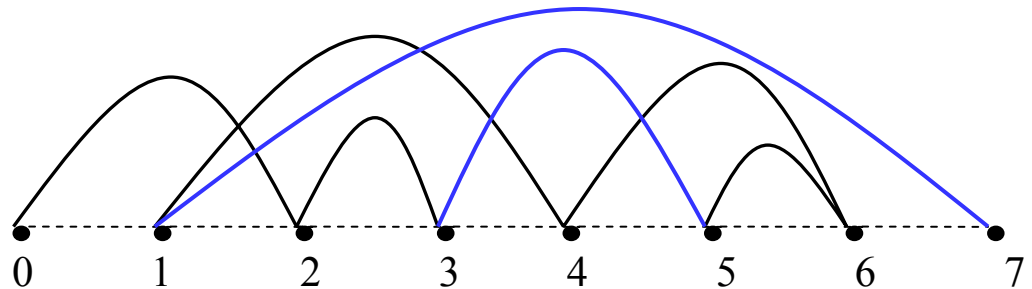
How does a reversal change the breakpoint graph?

- 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
- The black edges are unaffected by the reversal so they remain the same for both graphs

Before: 0 2 3 1 4 6 5 7

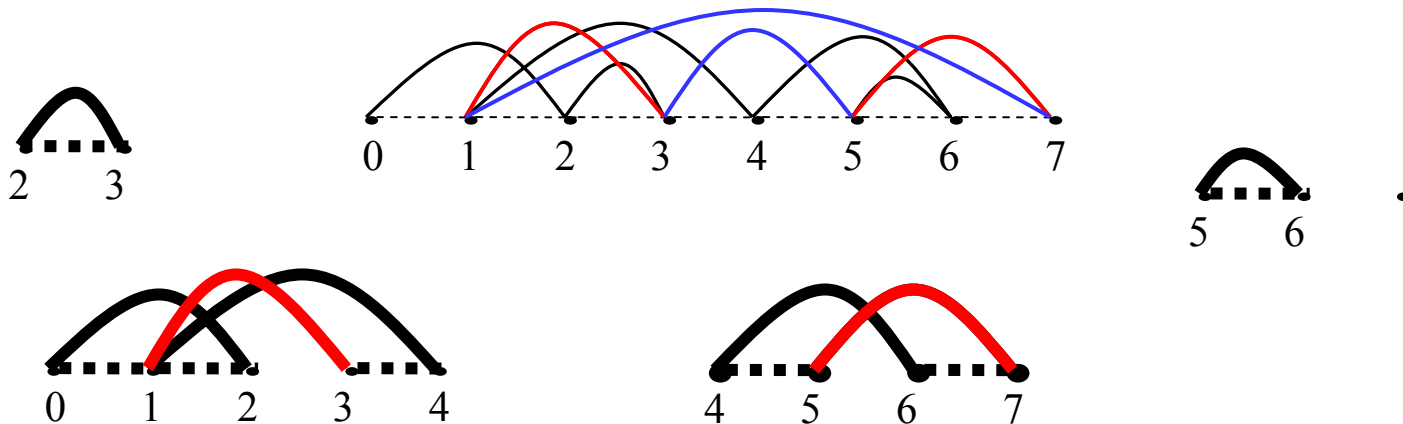


After: 0 2 3 5 6 4 1 7



# Estimating reversal distance by Cycle Decomposition

- A reversal removes 2 edges (red) and replaces them with 2 new edges (blue)
- A breakpoint graph can be decomposed into cycles that have edges with **alternating** patterns (solid / dashed).
- What effects have reversal on these cycles ?



# Effects of Reversals

## Case 1:

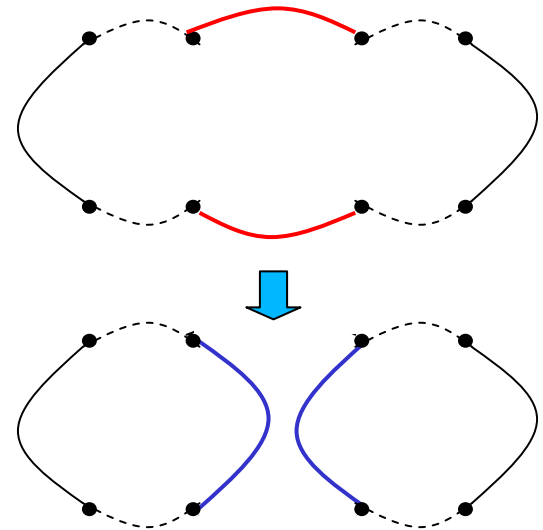
Both edges belong to the same cycle

- 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

➔  $c(\pi\rho) - c(\pi) = 1$

*This is called a **proper reversal** since there's a cycle increase after the reversal.*



# Effects of Reversals

## Case 1:

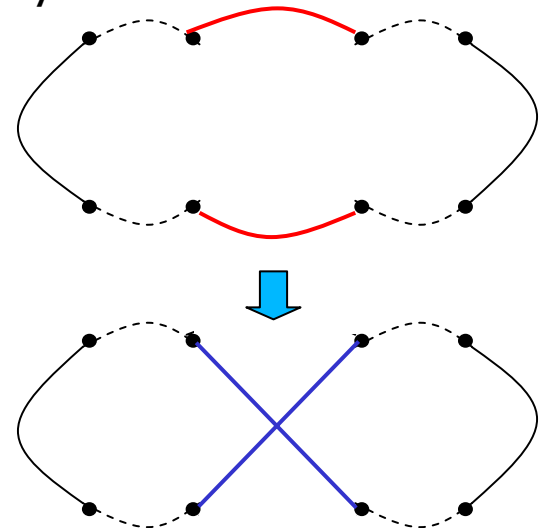
Both edges belong to the same cycle

- Remove the center black edges and replace them with new black edges (there are two ways to replace them)

- After this replacement, there now exists 2 cycles instead of 1 cycle
- Or after this replacement, there still exists 1 cycle

➔  $c(\pi\rho) - c(\pi) = 0$

*Therefore, after the reversal  
 $c(\pi\rho) - c(\pi) = 0$  or  $1$*



# Effects of Reversals (Continued)

## Case 2:

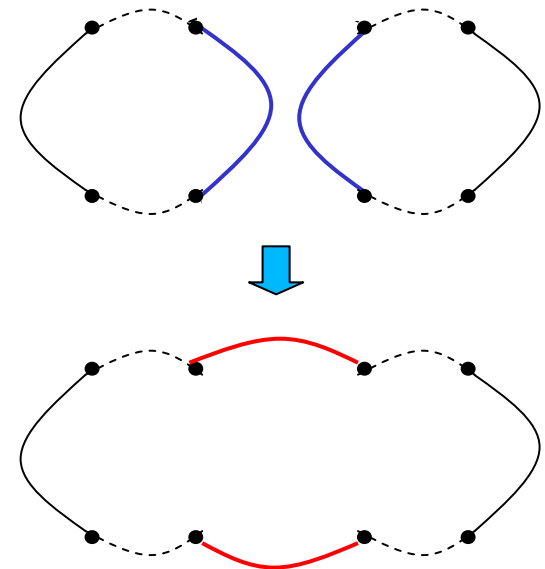
Both edges belong to different cycles

- 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  $n + 1$ ,  $c(\textit{identity}) = n + 1$
- $c(\textit{identity}) - c(\pi)$  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 increase by one, then:  
$$d(\pi) = c(\textit{identity}) - c(\pi) = n + 1 - c(\pi)$$
- Yet, not every reversal can increase the cycle decomposition

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

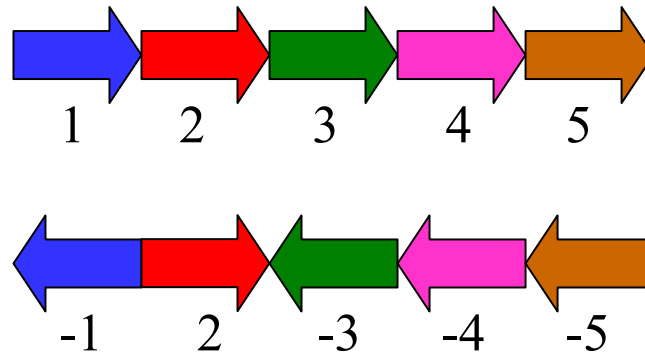
*For most biological systems the equality holds*

# The Complexity Reversal Distance

- 1997 – Alberto Caprara: *Sorting by reversals is difficult*. RECOMB 1997, ACM Press, 75-83.
- **Computing reversal distance is NP-hard!**
- Surprisingly, signed version of the problem is of polynomial complexity

# Signed Permutation

- Genes are *directed* fragments of DNA and we represent a genome by a signed permutation
- If genes are in the same position but their orientations are different, they do not have the equivalent gene order
- For example, these two permutations have the same order, but each gene's orientation is the reverse; therefore, they are not equivalent gene sequences



---

# Signed Permutation

- The polynomial algorithm for computing signed reversal sorting
    1. **Basic sorting** until we get a positive permutation.
    2. If the permutation is not sorted then continue with **hurdles removal**.
-

# Basic sorting

- As usual, we will assume that  $\pi$  is framed by 0 and  $n + 1$ , and that those extra elements are always positive:

$$\pi = (0 \ \pi_1 \ \pi_2 \ \dots \ \pi_n \ n + 1)$$

- An **oriented pair**  $(\pi_i, \pi_j)$  is a pair of consecutive integers, that is  $|\pi_i| - |\pi_j| = \pm 1$ , with opposite signs, i.e.  $\pi_i + \pi_j = \pm 1$ .
- Example
  - $(0 \ 3 \ 1 \ 6 \ 5 \ -2 \ 4 \ 7)$
  - $(0 \ 3 \ 1 \ \underline{6 \ 5} \ -2 \ 4 \ 7)$  # pair  $(1, -2)$  induces reversal
  - $(0 \ 3 \ 1 \ 2 \ -5 \ -6 \ 4 \ 7)$

# Basic sorting

- An **oriented pair**  $(\pi_i, \pi_j)$  is a pair of consecutive integers, that is  $|\pi_i| - |\pi_j| = \pm 1$ , with opposite signs, i.e.  $\pi_i + \pi_j = \pm 1$ .

- Example

- $(0\ 3\ 1\ 6\ 5\ -2\ 4\ 7)$

- $(0\ 3\ 1\ \underline{6\ 5}\ -2\ 4\ 7)$  # pair  $(1,-2)$  induces reversal

- $(0\ 3\ 1\ 2\ -5\ -6\ 4\ 7)$

- In general, the reversals by an oriented pair will be:

- $\rho(i, j - 1),$  if  $\pi_i + \pi_j = +1$

$(0\ \underline{3\ 1\ 6\ 5}\ -2\ 4\ 7) \rightarrow (0\ -5\ -6\ -1\ -3\ -2\ 4\ 7)$

$(0\ 3\ 1\ \underline{-6\ 5}\ -2\ 4\ 7) \rightarrow (0\ 3\ 1\ 4\ 2\ -5\ 6\ 7)$

- $\rho(i + 1, j),$  if  $\pi_i + \pi_j = -1$

$(0\ 3\ 1\ \underline{6\ 5}\ -2\ 4\ 7) \rightarrow (0\ 3\ 1\ 2\ -5\ -6\ 4\ 7)$

$(0\ -3\ \underline{1\ 6\ 5}\ 2\ 4\ 7) \rightarrow (0\ -3\ -2\ -5\ -6\ -1\ 4\ 7)$

# Reversal score and basic sorting

- The **score** of an (oriented) reversal is defined as the number of oriented pairs in the resulting permutation.
- **Example**  
 $(0 \ 3 \ 1 \ 6 \ 5 \ -2 \ 4 \ 7)$  reversal  $\rho(1,4)$   
 $(0 \ -5 \ -6 \ -1 \ -3 \ -2 \ 4 \ 7)$  score 4 !
- **Basic sorting:** As long as  $\pi$  has an oriented pair, choose the oriented reversal that has maximal score.
- **Example**
  - Step 1:  $(0 \ 3 \ 1 \ 6 \ 5 \ -2 \ 4 \ 7)$  two oriented pairs  $(1,-2)$  and  $(3,-2)$  with score 2 and 4.
  - Step 2:  $(0 \ -5 \ -6 \ -1 \ -3 \ -2 \ 4 \ 7)$  pairs  $(0,-1), (-3,4), (-5,4)$  and  $(-6,7)$
  - Step 3:  $(0 \ -5 \ -6 \ -1 \ 2 \ 3 \ 4 \ 7)$  pairs  $(0,-1), (-1,2), (-5,4)$  and  $(-6,7)$

## Basic sorting cont.

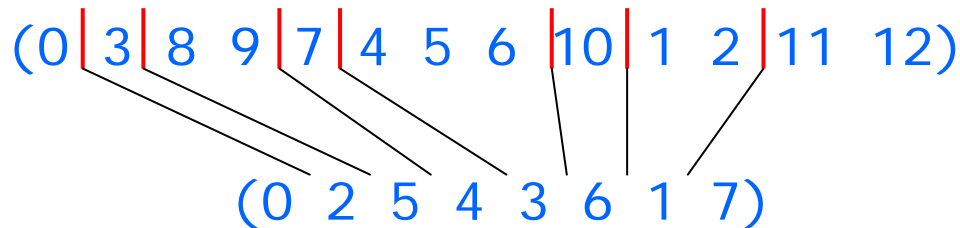
- (0 -5 -6 1 2 3 4 7)
- (0 -5 -4 -3 -2 -1 6 7)
- (0 1 2 3 4 5 6 7)
- This elementary strategy of Basic sorting is sufficient to optimally sort almost all permutations that arise from biological data!
- **Claim 1:** Basic sorting applies  $k$  reversals to a permutation  $\pi$ , yielding a permutations  $\pi'$  such that  $d(\pi) = d(\pi') + k$ .

- 
1. **Basic sorting** until we get a positive permutation
  2. If the permutation is not sorted then continue with **hurdles removal**
-



# Sorting positive permutations

- Such permutations are called **reduced** if they do not contain consecutive elements.
  - How to reduce a permutation?*



- We suppose circular order by setting  $0$  to be successor of  $n+1$
- Framed interval:** encompasses all integers between  $i$  and  $i+k$  belong to the interval  $[i \dots i+k]$ .
- Consider permutation:  $(02543617)$ . The whole permutation is a framed interval, as well as  $25436$  and, by circularity,  $61702$ .

# Tough regions: Hurdles

- A **hurdle** in  $\pi$  is a framed interval that contains no shorter framed interval.
  - When a permutation has only one or two hurdles, one reversal is sufficient to create enough oriented pairs to completely sort the permutation with *Basic sorting*.
  - Two operations break hurdles: *hurdle cutting* and *hurdle merging*.
-

# Breaking Hurdles

- **Hurdle cutting:** Reversing segment between  $i$  and  $i + 1$  of a hurdle:

$$i \dots \underline{i + 1} \dots i + k$$

$$(0 \ 2 \ 4 \ 3 \ 1 \ 5) \rightarrow (0 \ -3 \ -4 \ -2 \ 1 \ 5)$$

which can be sorted in 4 reversals.

- **Hurdle merging:** Merging the end points of two hurdles.

$$i \dots \underline{i + k} \dots i' \dots i' + k'$$

$$(0 \ 2 \ 5 \ 4 \ 3 \ 6 \ 1 \ 7) \rightarrow (0 \ 2 \ 5 \ 4 \ 3 \ -6 \ 1 \ 7)$$

which can be sorted in 5 reversals.

# Super Hurdles

- A **simple hurdle** is a hurdle whose cutting decreases the number of hurdles. Hurdles that are not simple are called **super hurdles**.
- **Example**
  1.  $(0\ 2\ 5\ 4\ 3\ 6\ 1\ 7)$  has two hurdles; after cutting and sorting the hurdle
    - $2\ \underline{5}\ 4\ 3\ 6$
    - $2\ -4\ \underline{-5}\ 3\ 6$
    - $2\ \underline{-4}\ \underline{-3}\ 5\ 6$
    - $2\ 3\ 4\ 5\ 6$
 we get  $(0\ 2\ 3\ 4\ 5\ 6\ 1\ 7)$  – it collapses to  $(0\ 2\ 1\ 3)$  (a reduction!) and has only one hurdle.
  2.  $(0\ 2\ 4\ 3\ 5\ 1\ 6\ 8\ 7\ 9)$  also contains two hurdles; after cutting and sorting the hurdle  $2\ 4\ 3\ 5$  the resulting reduced permutation has still two hurdles  $(0\ 2\ 3\ 4\ 5\ 1\ 6\ 8\ 7\ 9) \xrightarrow{\text{reduction}} (0\ 2\ 1\ 3\ 5\ 4\ 6)$

# Breaking Hurdles

- **Hurdles removal:** If a permutation has  $2k$  hurdles,  $k \geq 2$ , merge any two non-consecutive hurdles. If a permutation has  $2k + 1$  hurdles,  $k \geq 1$ , then if it has one simple hurdle, cut it; If it has none, merge two non-consecutive hurdles, or consecutive ones if  $k = 1$ .
- For proofs of all the algorithms and claims – see:
  - A very elementary presentation of the Hannenhalli–Pevzner Theory by Anne Bergeron  
<http://citeseer.ist.psu.edu/599900.html>
  - Maximal exposure can be obtained from: Efficient algorithms for multichromosomal genome rearrangements by Glen Tesler  
[http://math.ucsd.edu/gptesler/pub\\_jcss.html](http://math.ucsd.edu/gptesler/pub_jcss.html)

# GRIMM Web Server

- GRIMM web server computes the reversal distances between signed permutations:

**GRIMM - Genome rearrangement algorithms**

Multiple genome form

Source genome:

Destination genome:

Chromosomes:  circular  linear (directed)  multichromosomal or undirected  
 Signs:  signed  unsigned

Or,

**Formatting options**

Report Style:  One line per genome (chromosomes concatenated)  One column (chromosomes separated)  Two column before & after (chromosomes separated)  
 Horizontal  Vertical  Yes  Show all chromosomes  Only affected chromosomes

Highlighting style:  Show all possible initial steps of optimal scenarios  
 before  after  between/both  no highlighting

Chromosome end format:  numeric (10)  subscripts (C<sub>10</sub>)  omit

Color coding:  Genes should be colored according to their chromosome in which genome:  
 source  destination

[Click here or scroll up to enter new data or change options.](#)

3 chromosomes, 12 genes, 6 caps    Multichromosomal Distance: 6

**One optimal rearrangement scenario**

Step	Description	Permutation
0	(Source)	-12 -7 -6 -5 -4 1 -8 -11 -9 -10 -3 -2
1	Fusion	-12 -7 -6 -5 -4 1 2 3 10 9 11 8
2	Translocation	-12 -11 -9 -10 -3 -2 -1 4 5 6 7 8
3	Reversal	-12 -11 9 -10 -3 -2 -1 4 5 6 7 8
4	Reversal	-12 -11 10 -9 -3 -2 -1 4 5 6 7 8
5	Reversal	-12 -11 -10 -9 -3 -2 -1 4 5 6 7 8
6	Reversal (Destination)	-12 -11 -10 -9 1 2 3 4 5 6 7 8

GRIMM 1.04 by [Glenn Tesler](#), University of California, San Diego.  
 Copyright © 2001-2002, The University of California.  
 Contains code from [GRAPPA](#), © 2000-2001, The University of New Mexico and The University of Texas at Austin.

MGR 1.0 by [Guillaume Bourque](#), University of Southern California.  
 Copyright © 2001, University of Southern California.  
 Contains code from [Phylip](#) 3.5, Copyright © 1986-1995 by Joseph Felsenstein and the University of Washington.

[Click here for details on how to cite this in your work.](#)

<http://www-cse.ucsd.edu/groups/bioinformatics/GRIMM>