Greedy Algorithms and Genome Rearrangements

Outline

1. Transforming Cabbage into Turnip
2. Genome Rearrangements
3. Sorting By Reversals
4. Pancake Flipping Problem
5. Greedy Algorithm for Sorting by Reversals
6. Approximation Algorithms
7. Breakpoints: a Different Face of Greed
8. Breakpoint Graphs

Motivating Example: Turnip vs. Cabbage

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

Motivating Example: Turnip vs. Cabbage

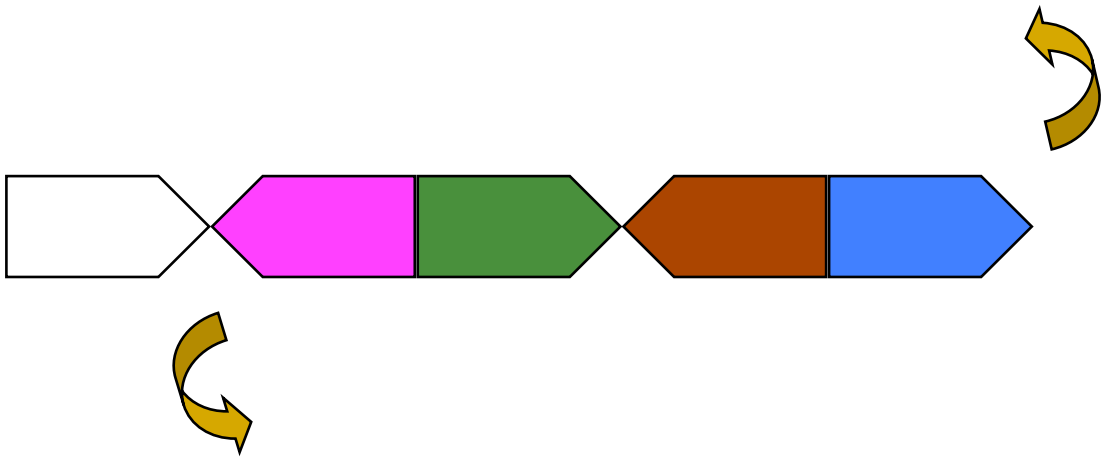
- In the 1980s Jeffrey Palmer studied evolution of plant organelles by comparing mitochondrial genomes of cabbage and turnip.
- He found 99% similarity between genes.
- These surprisingly similar gene sequences differed in gene order.
- This study helped pave the way to analyzing genome rearrangements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

Motivating Example: Turnip vs. Cabbage

- Gene order comparison:

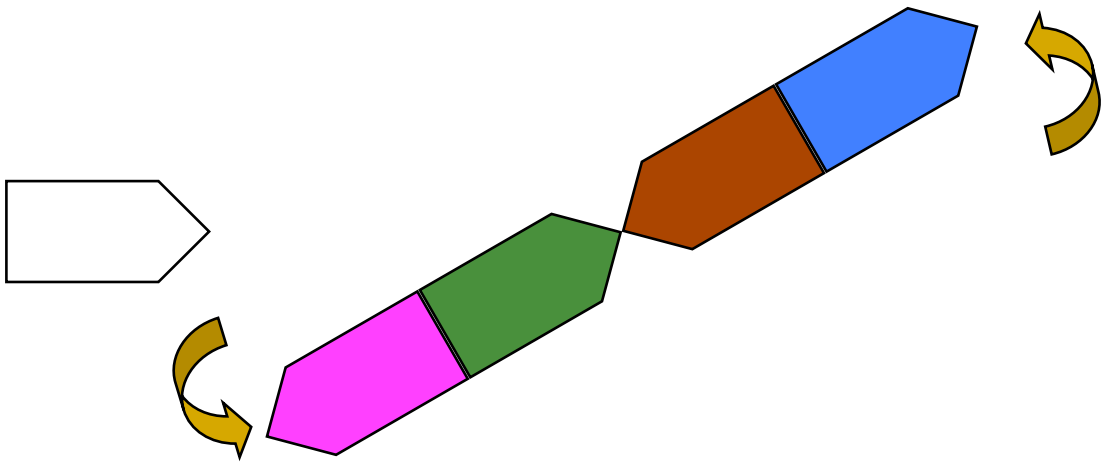
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



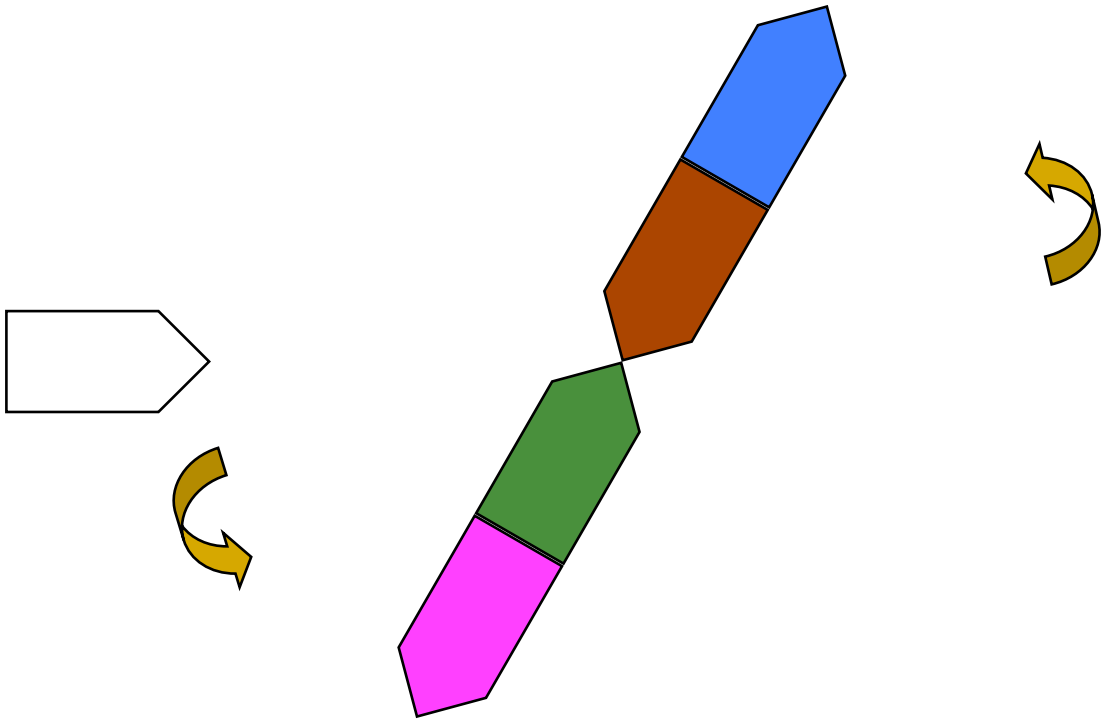
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



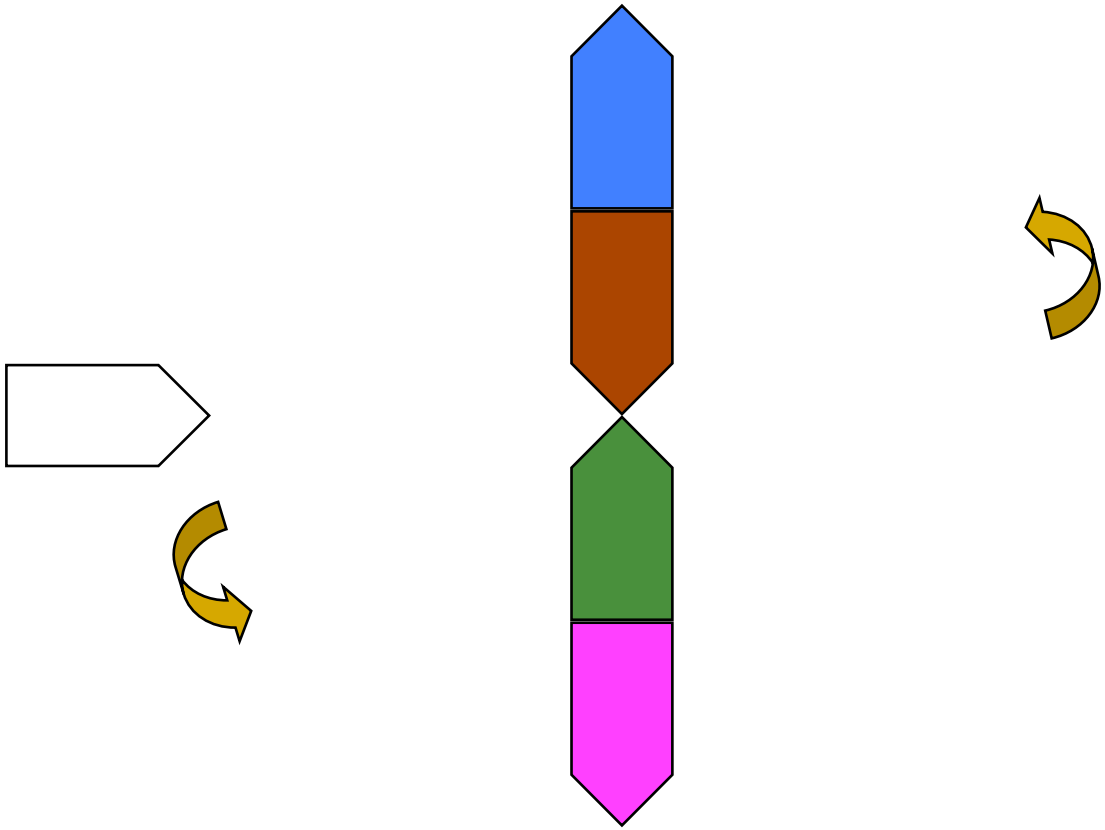
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



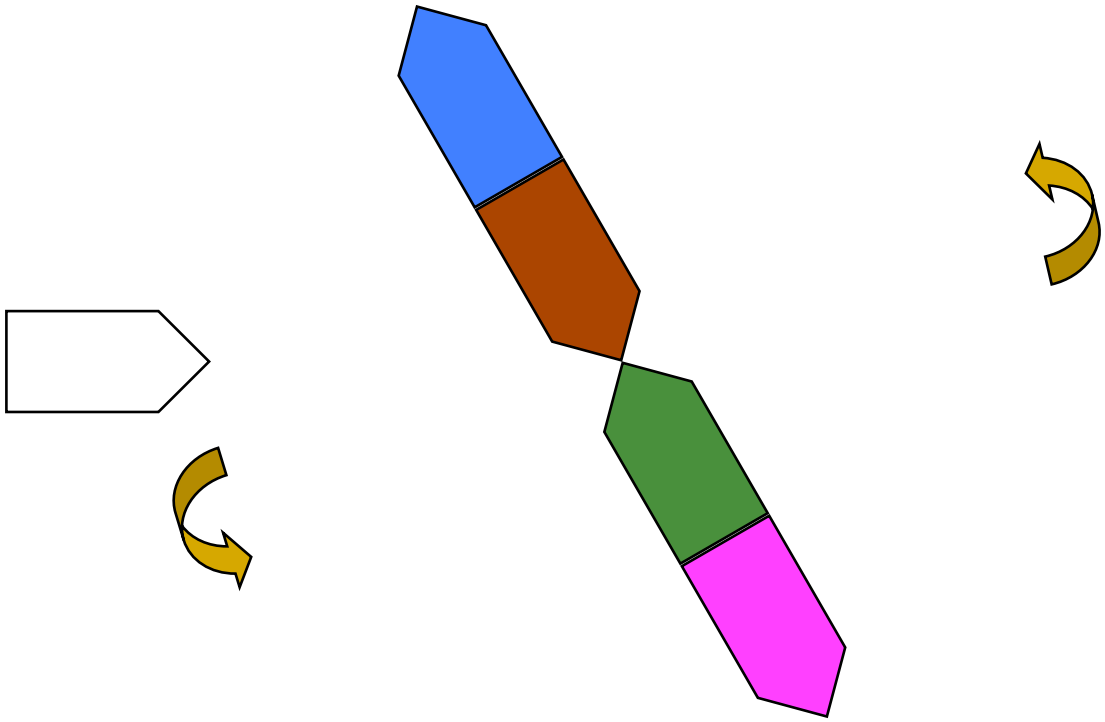
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



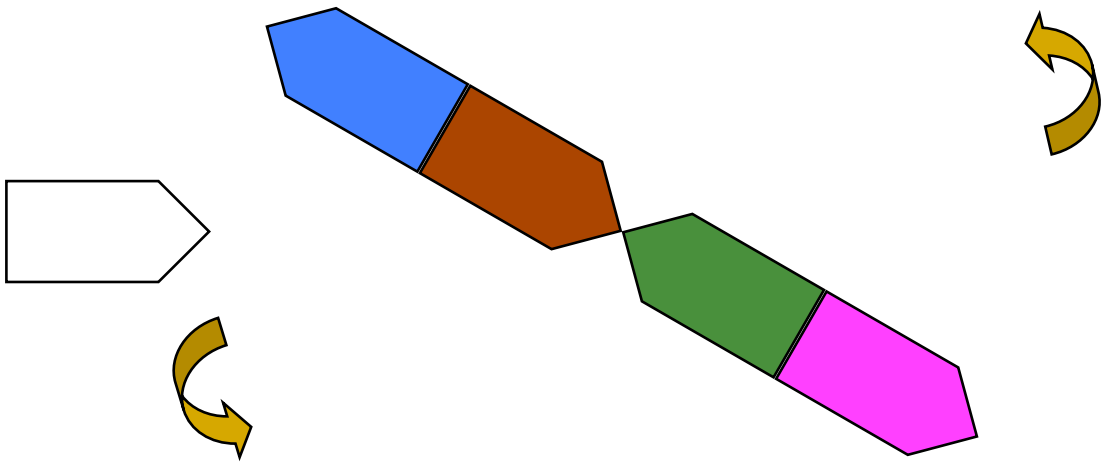
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



Motivating Example: Turnip vs. Cabbage

- Gene order comparison:

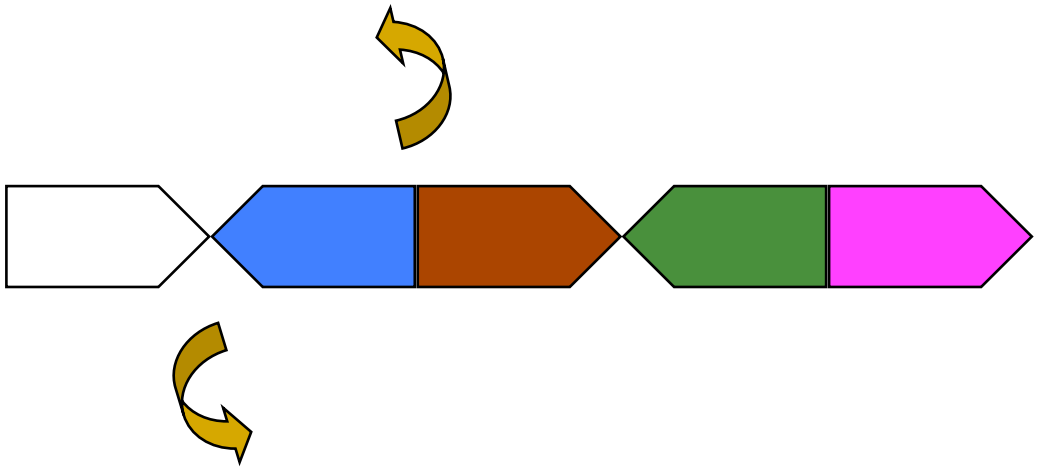


Motivating Example: Turnip vs. Cabbage

- Gene order comparison:

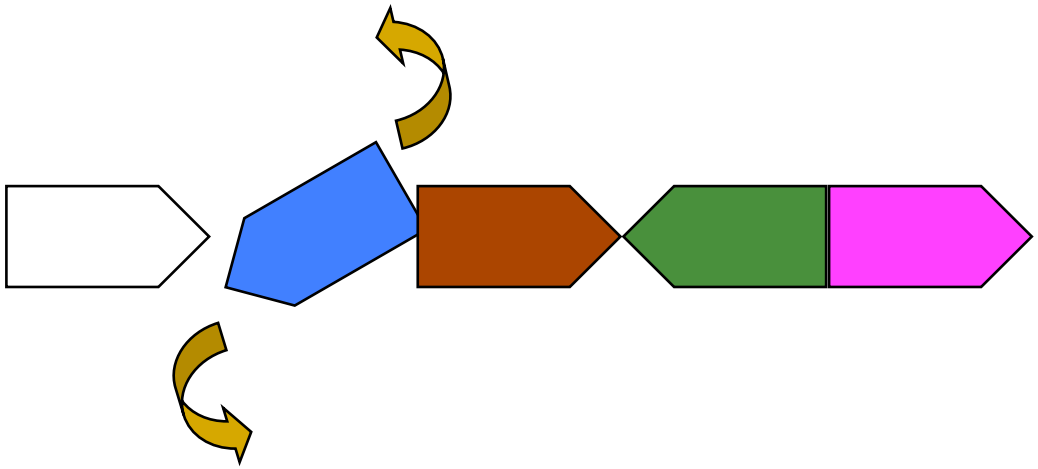
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



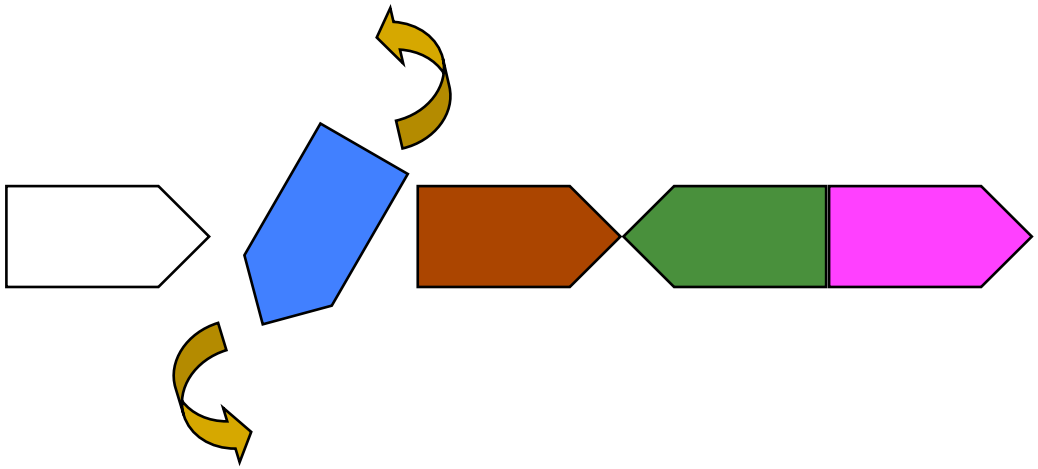
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



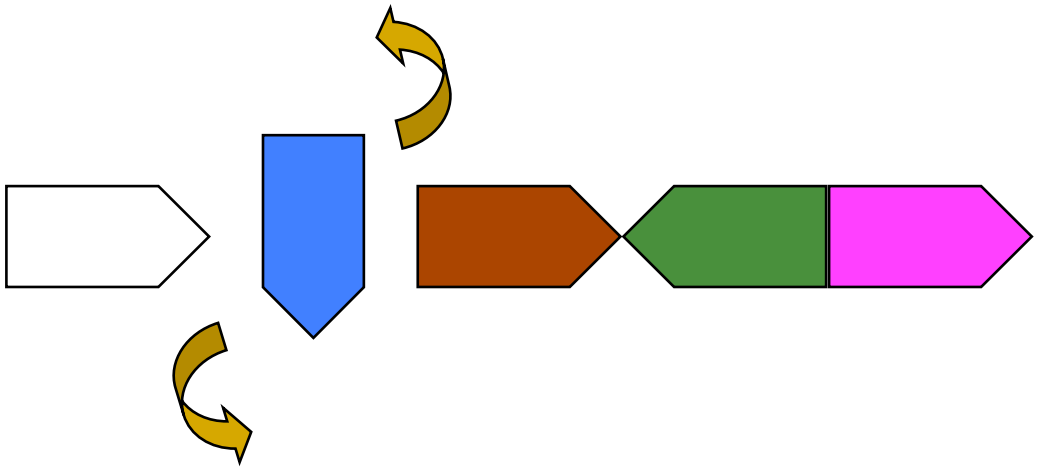
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



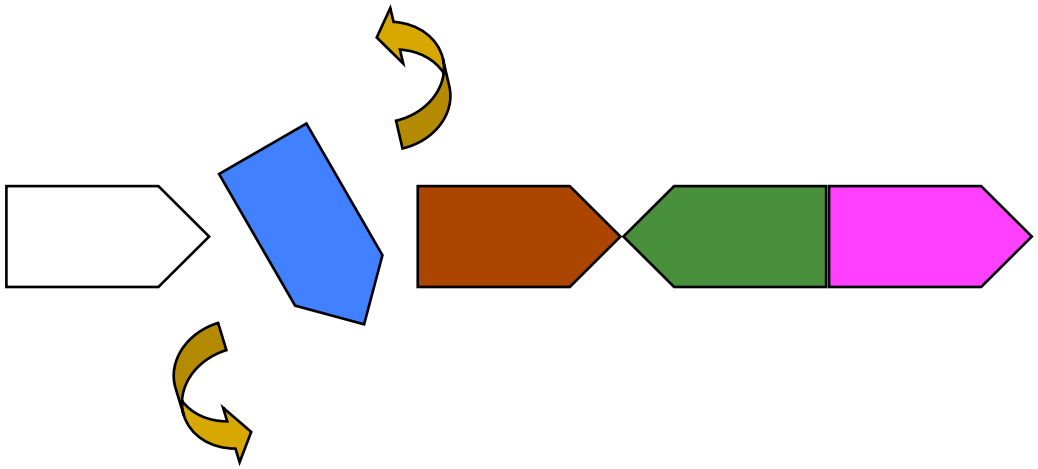
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



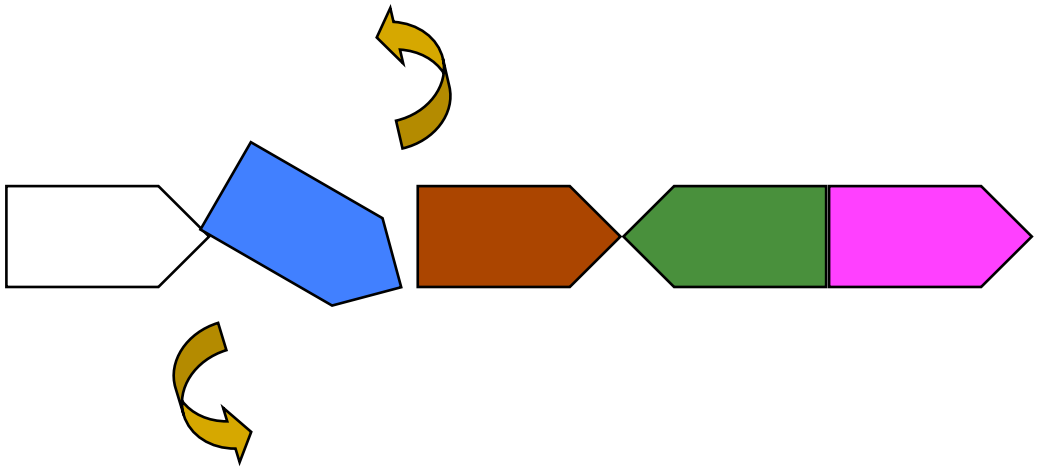
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



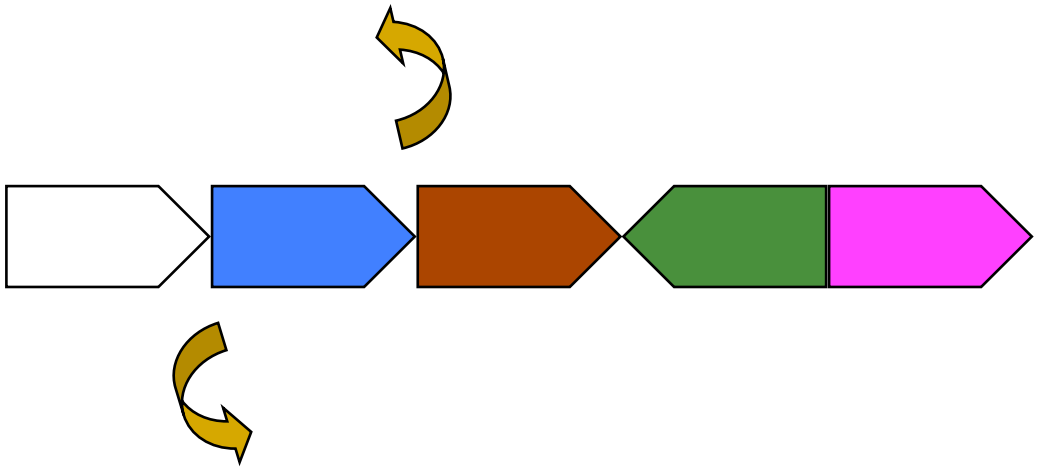
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



Motivating Example: Turnip vs. Cabbage

- Gene order comparison:

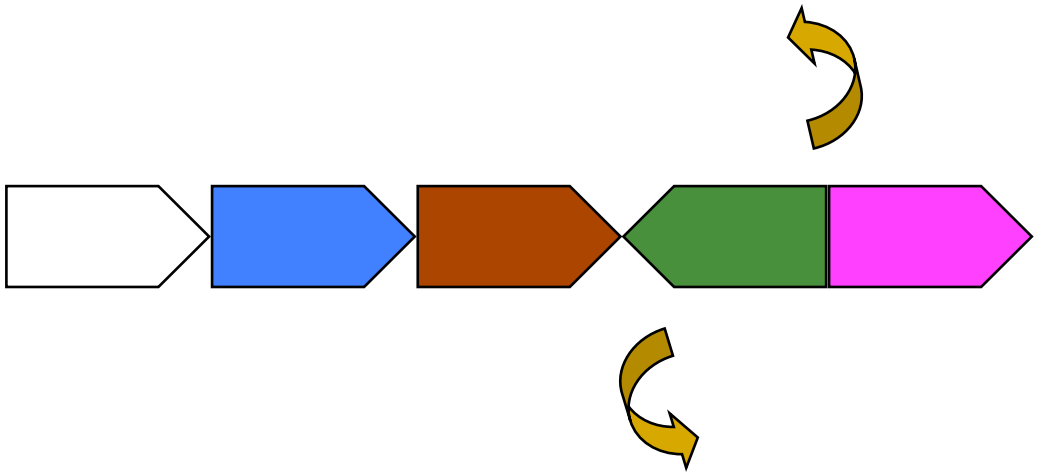


Motivating Example: Turnip vs. Cabbage

- Gene order comparison:

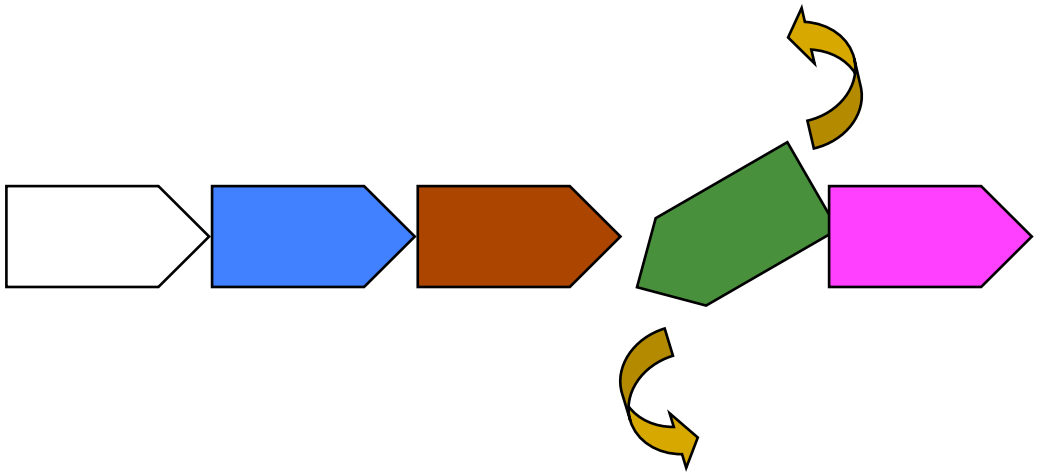
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



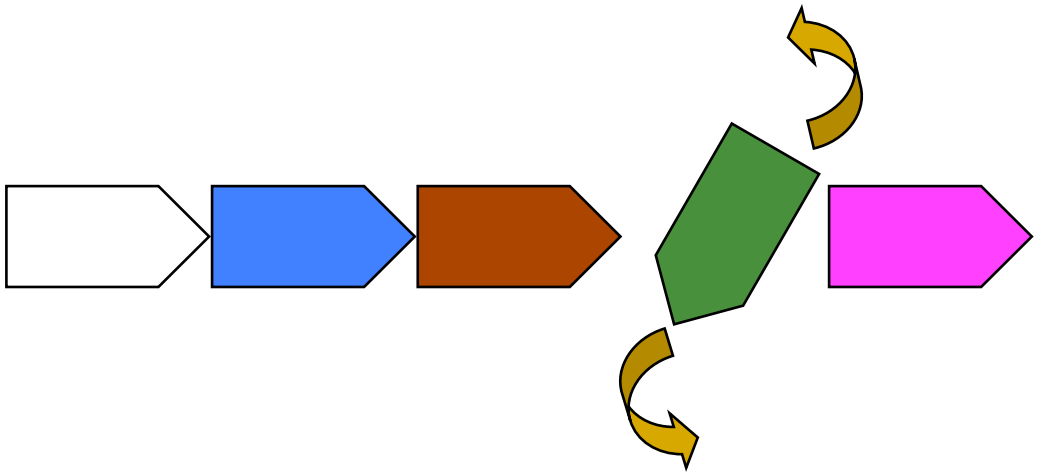
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



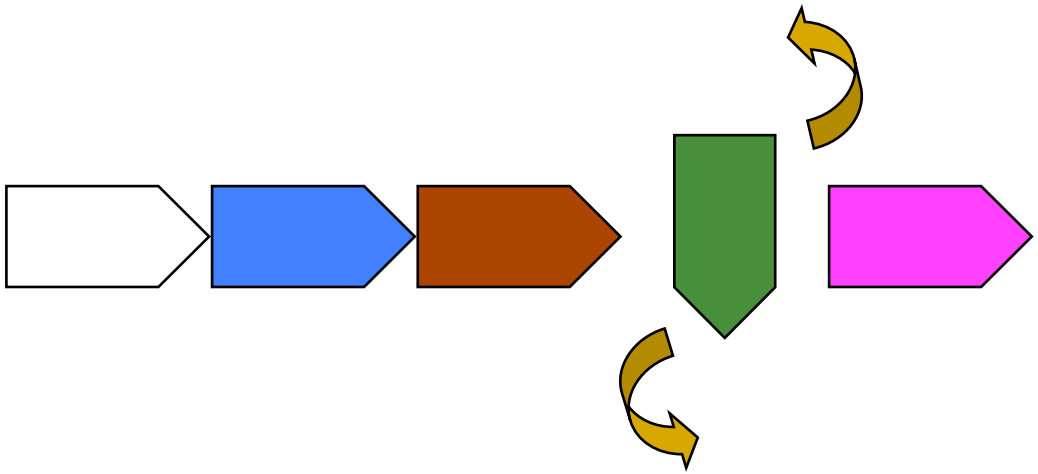
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



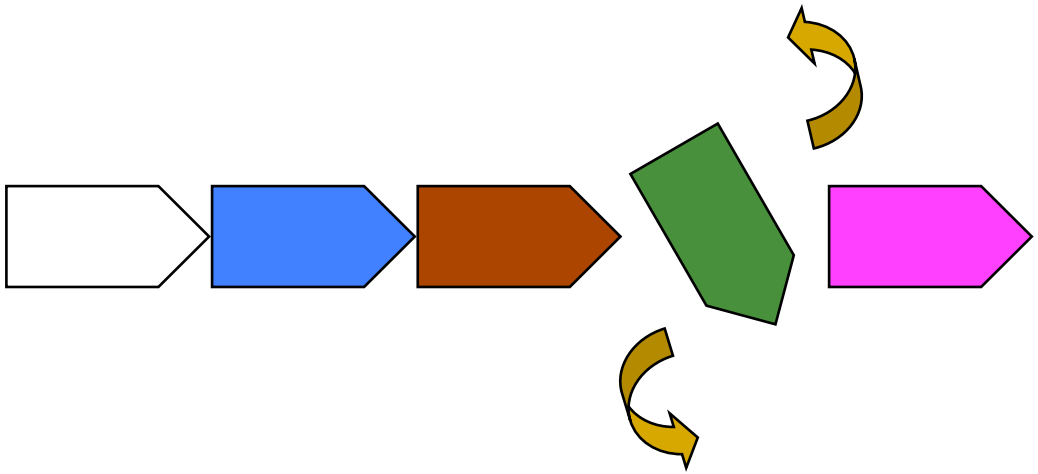
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



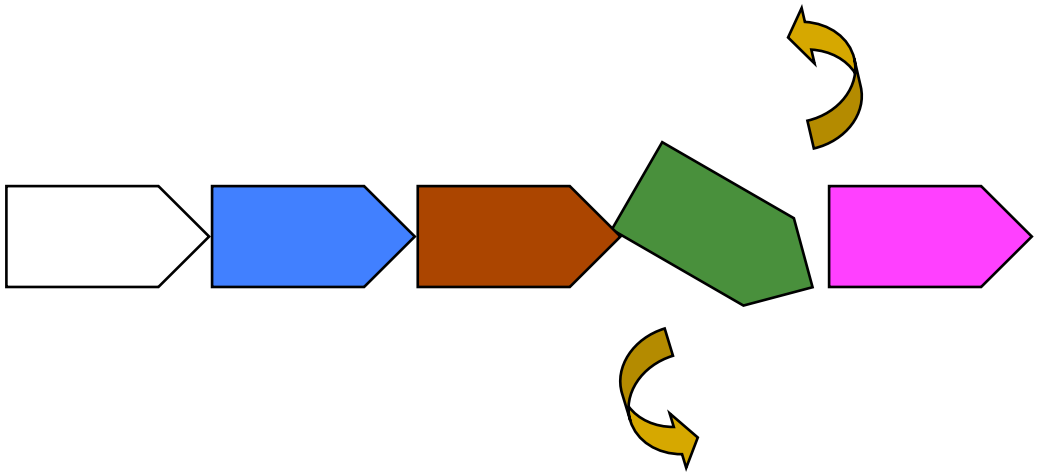
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



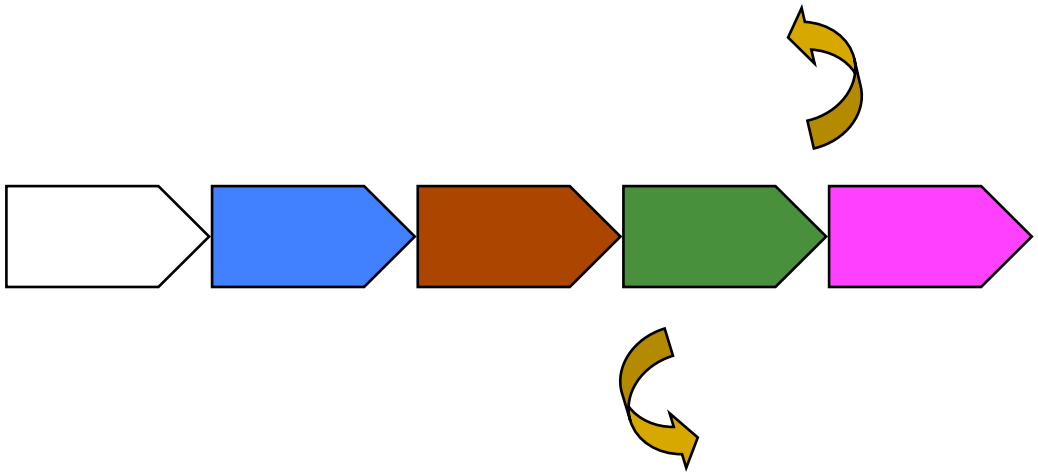
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:



Motivating Example: Turnip vs. Cabbage

- Gene order comparison:

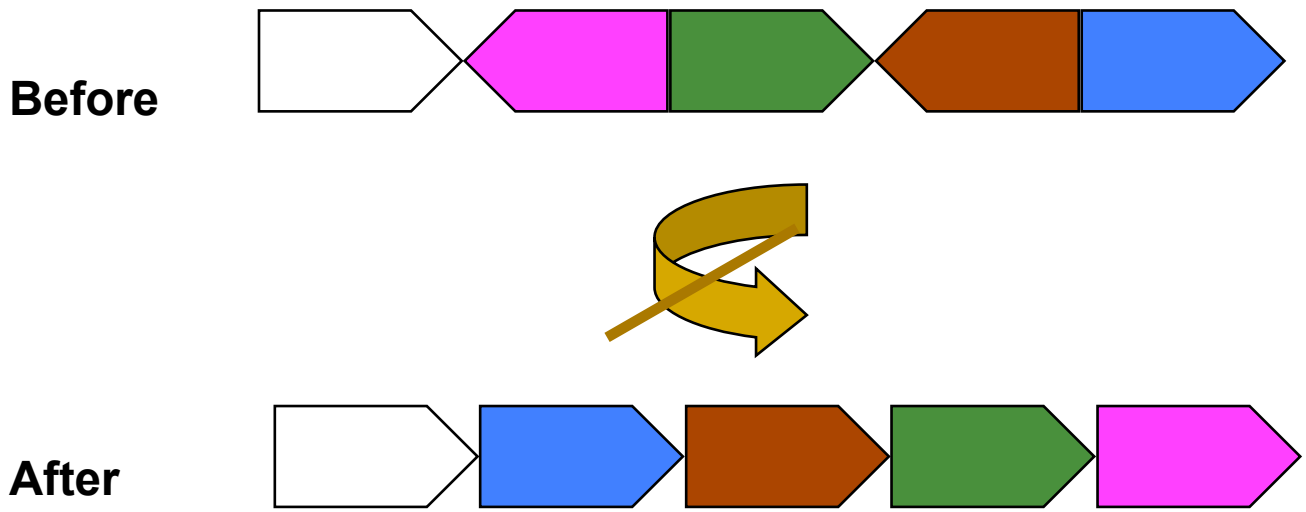


Motivating Example: Turnip vs. Cabbage

- Gene order comparison:

Motivating Example: Turnip vs. Cabbage

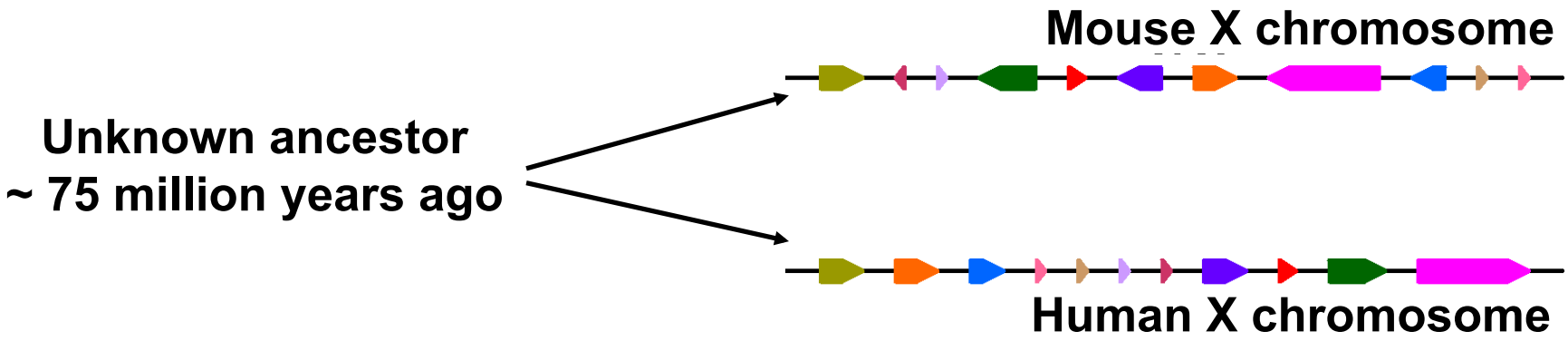
- Gene order comparison:



- Evolution is manifested as the divergence in gene order and orientation caused by these inversions of segments of genes

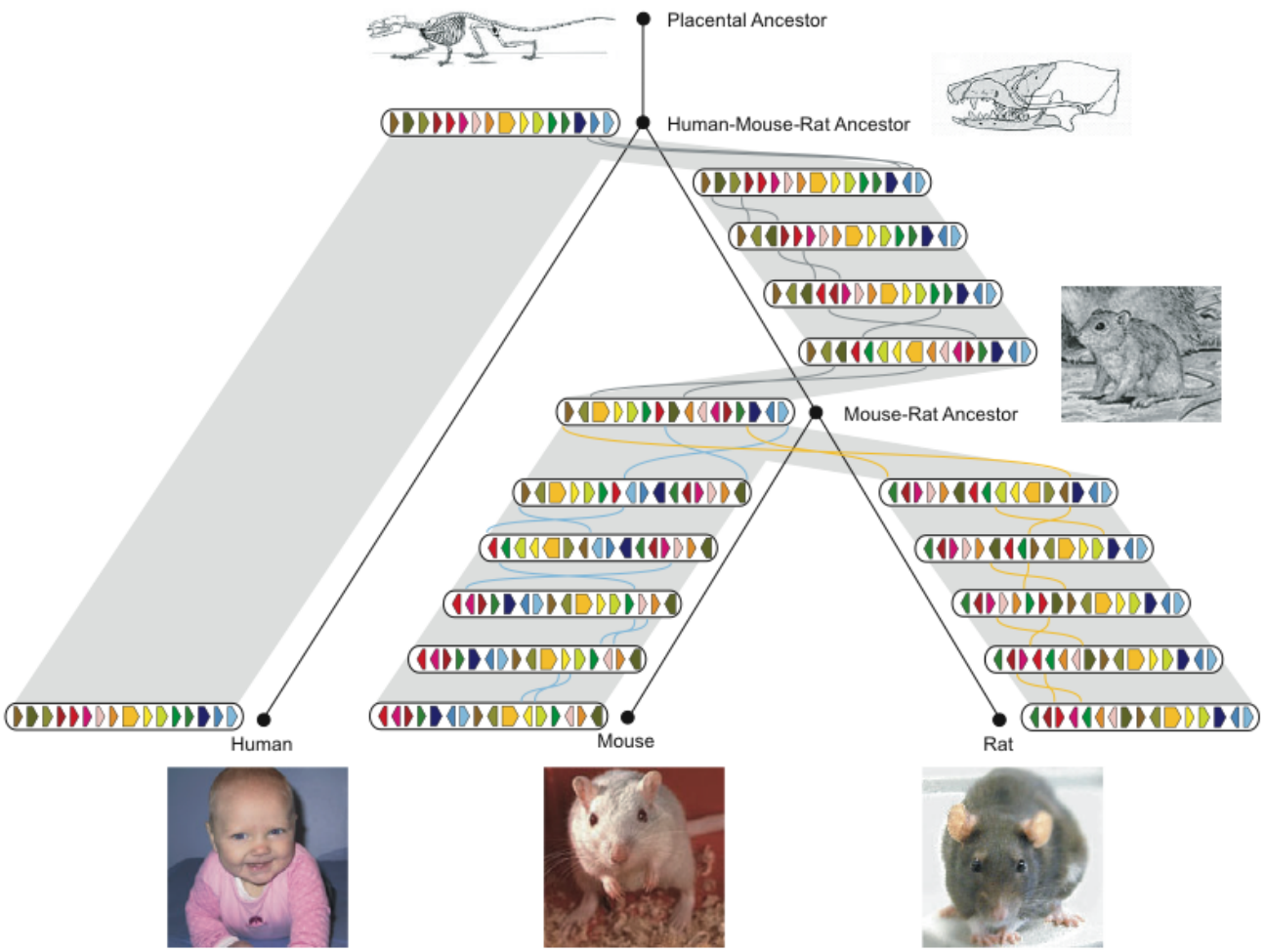
Another Example: Mice and Humans

- The X chromosomes of mice and humans give another example.



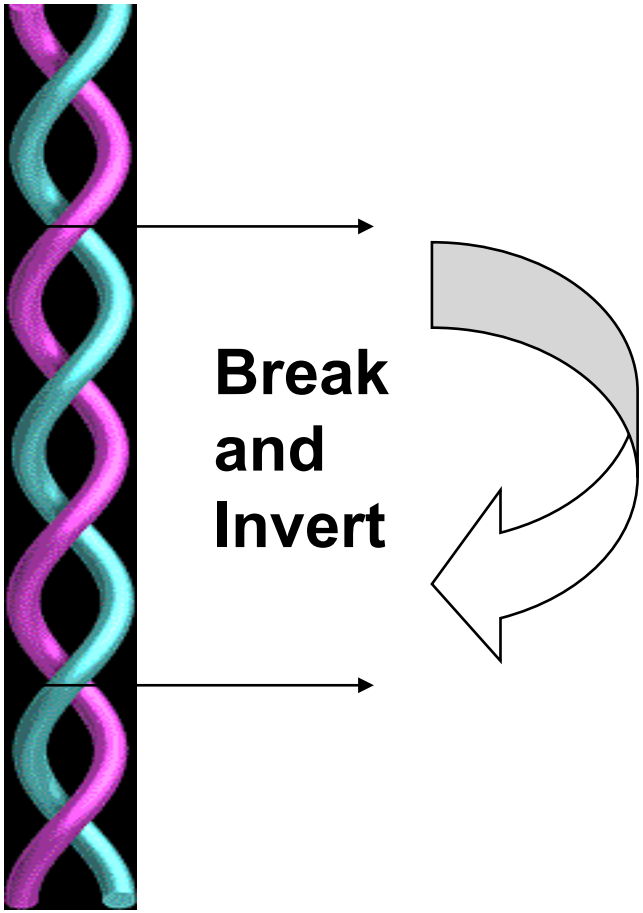
- What are the similarity blocks and how do we find them?
- What is the ordering of the blocks?
- What is the scenario for transforming one genome into another?

History of Chromosome X



Rat Consortium, *Nature*, 2004

Reversals: Example



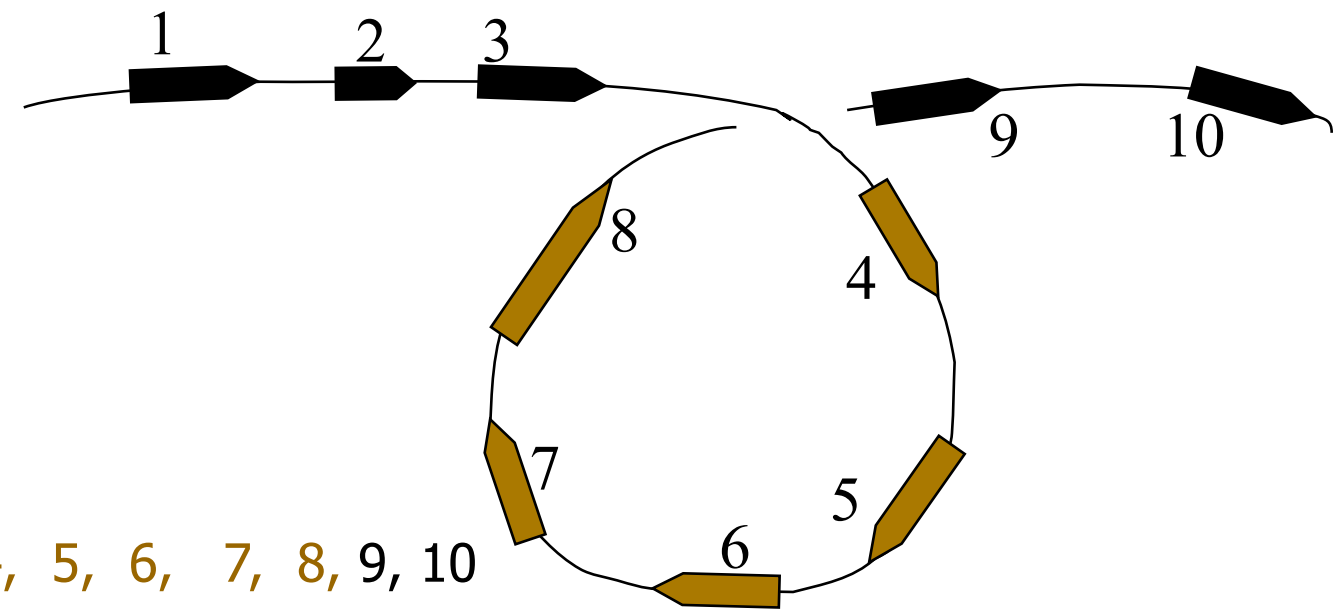
5' ATGCCTGTACTA 3'

3' TACGGACATGAT 5'

5' ATGTACAGGCTA 3'

3' TACATGTCCGAT 5'

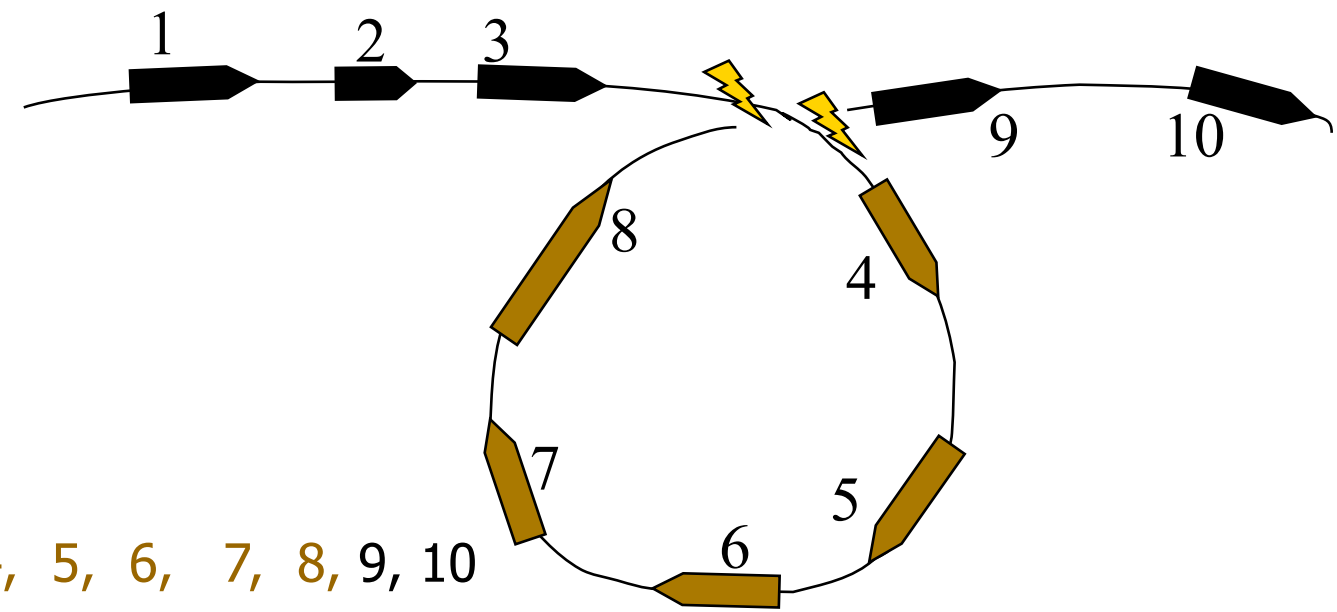
Reversals



1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- Blocks represent conserved genes.

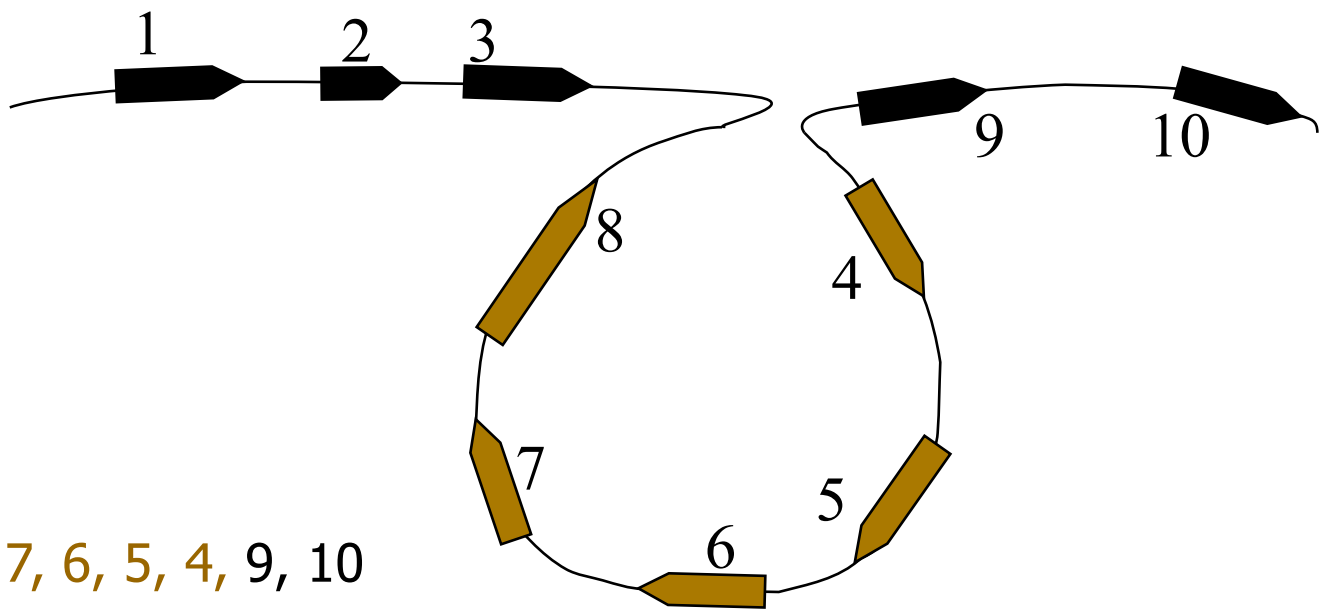
Reversals



1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- Blocks represent conserved genes.
- A reversal introduces two breakpoints, represented by

Reversals



1, 2, 3, 8, 7, 6, 5, 4, 9, 10

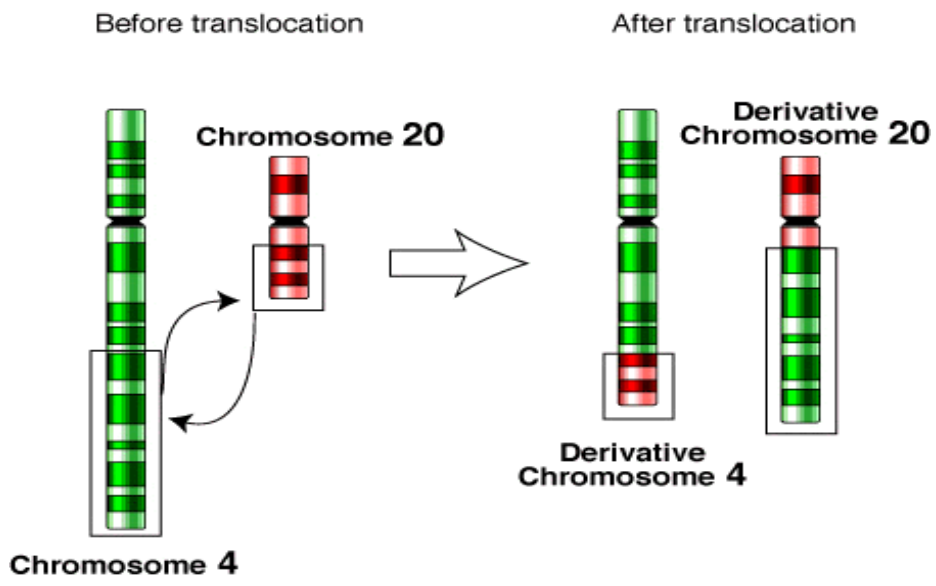
- Blocks represent conserved genes.
- A reversal introduces two breakpoints, represented by
- As a result of the reversal, the gene ordering has changed to 1, 2, 3, 8, 7, 6, 5, 4, 9, 10.

Additional Genome Rearrangements

- Besides reversals, we also have:
  1. Fusion and fission of chromosomes

Additional Genome Rearrangements

- Besides reversals, we also have:
  1. Fusion and fission of chromosomes
  2. Translocations



Rearrangements: Mathematical Representation

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

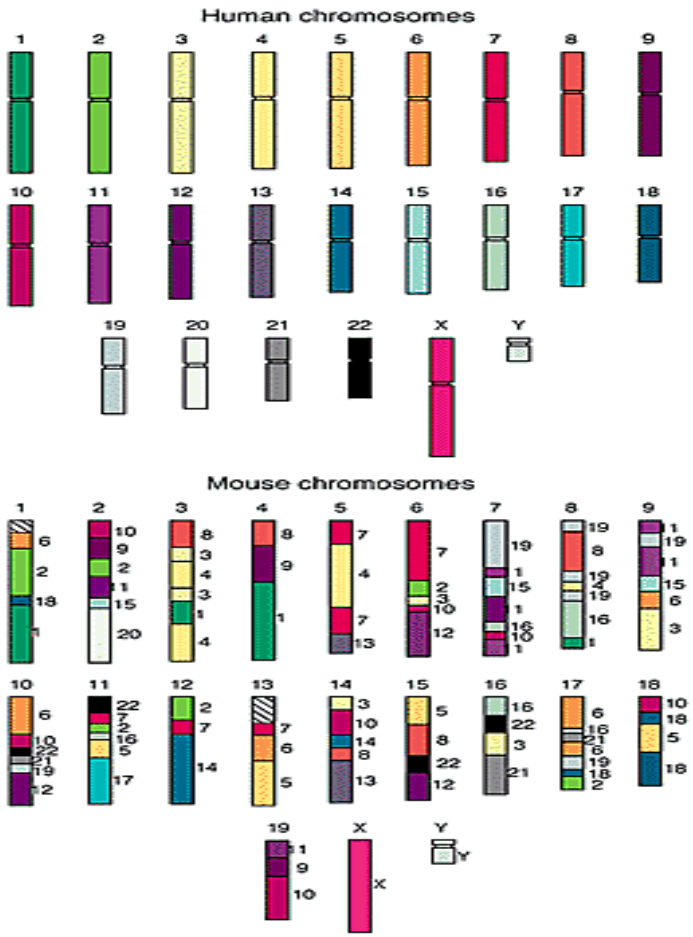
Fission

Each number represents a conserved region; +/- represents orientation.

Mice vs. Humans Revisited

- Humans and mice have similar genomes, but their genes are ordered differently.

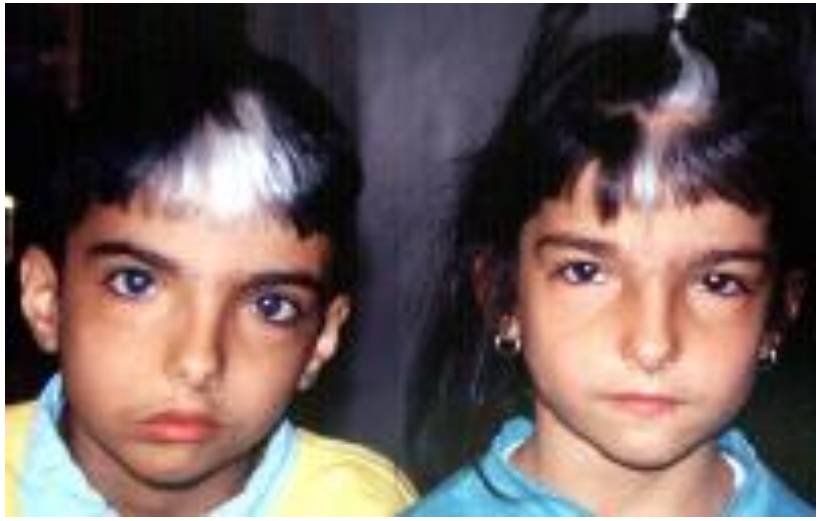
- ~245 rearrangements
  - Reversals
  - Fusions
  - Fissions
  - Translocations



Mice vs. Humans Revisited

- Waardenburg's syndrome is characterized by pigmentary dysphasia.

- A gene implicated in the disease was linked to human chromosome 2 but it was not clear where exactly it is located on chromosome 2.



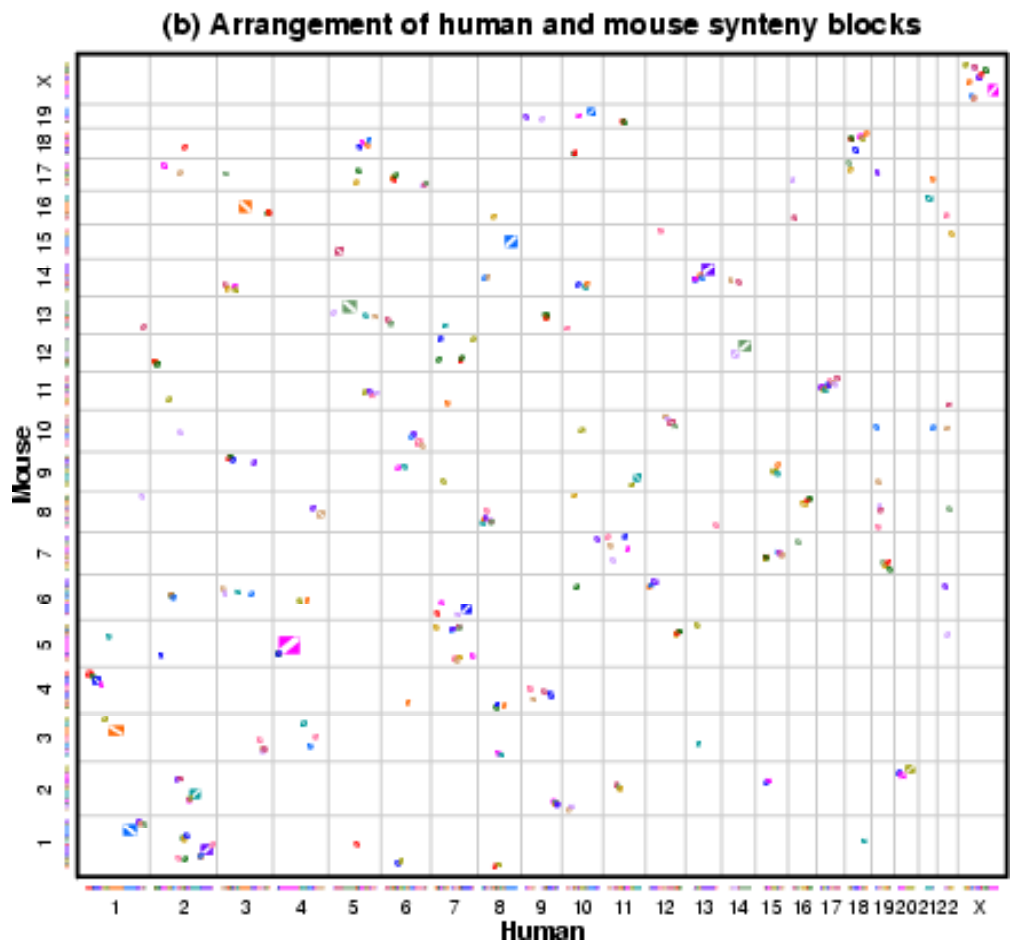
Mice vs. Humans Revisited

- 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; this gives clues to where the gene is located in humans.

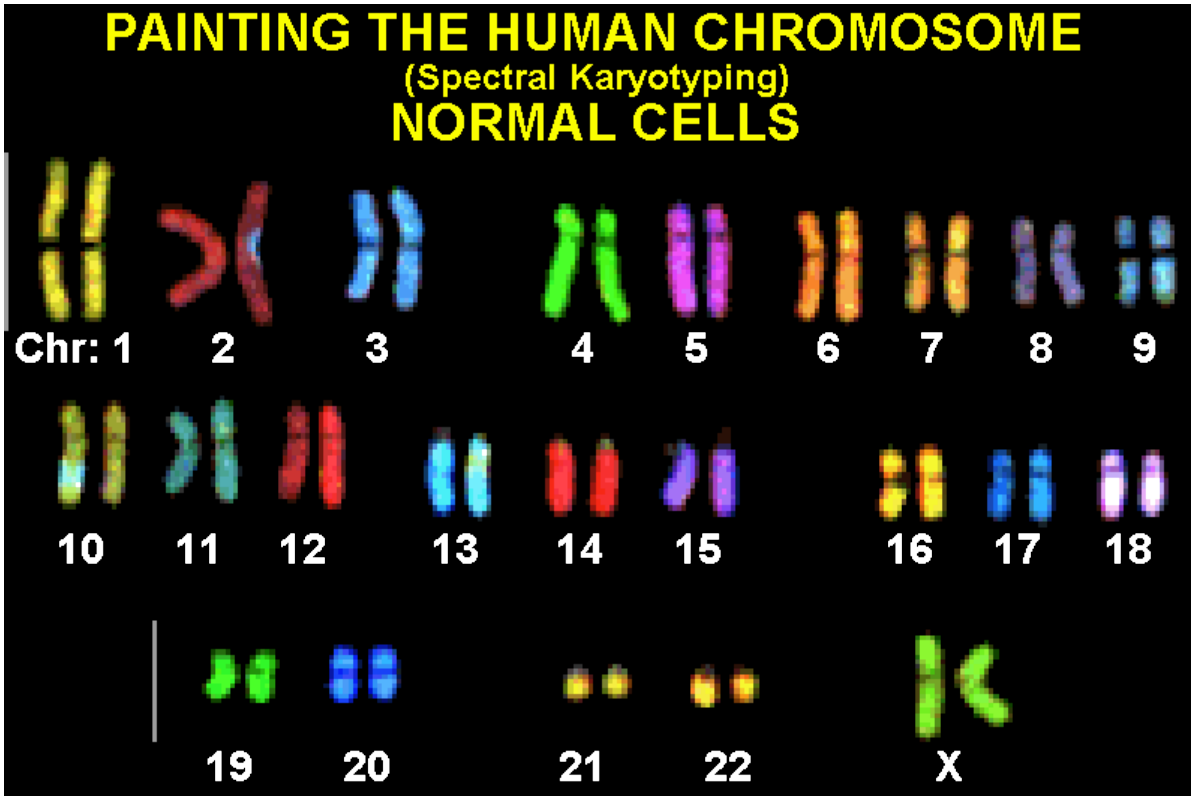
Mice vs. Humans Revisited

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



Additional Application: Cancer

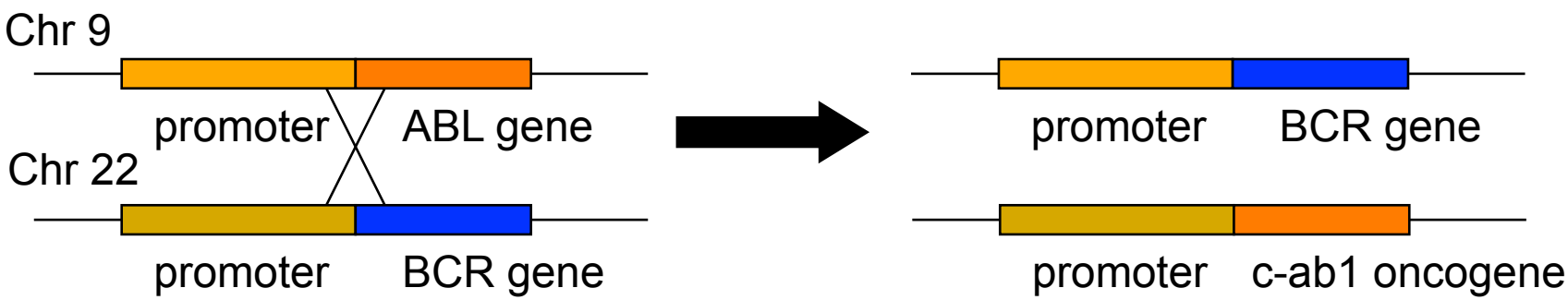
- Normal cells will have a certain makeup of chromosomes, as revealed by "chromosome painting."



Additional Application: Cancer

- Rearrangements may disrupt genes and alter gene regulation.

- **Example**: translocation in leukemia yields "Philadelphia" chromosome:

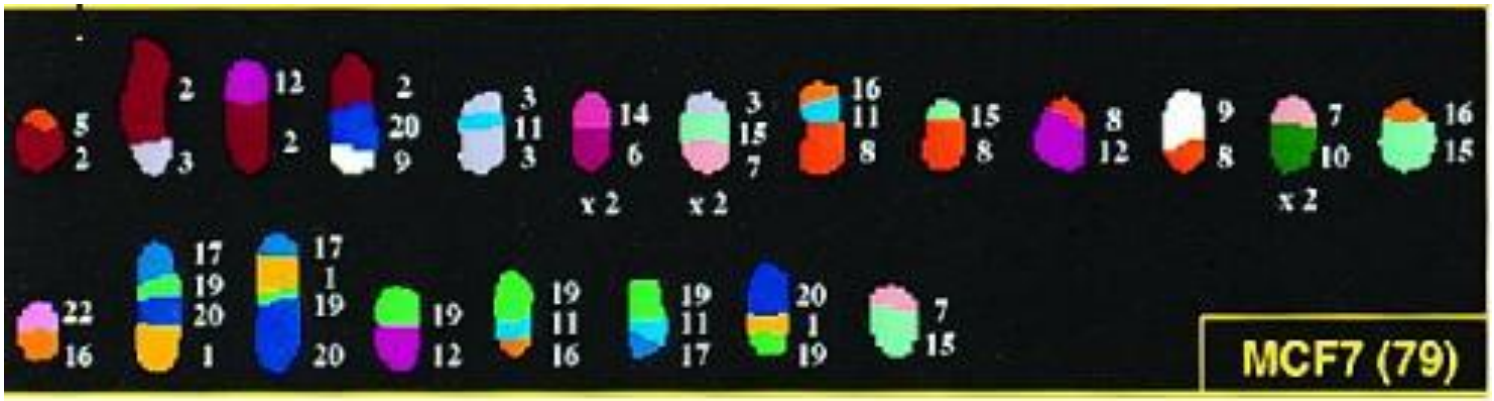


- There are thousands of individual rearrangements known for different tumors.

Additional Application: Cancer

- MCF7 is the human breast cancer cell line.

- Cytogenetic analysis (low-resolution) suggests a complex architecture with many translocations.
  - What is the detailed architecture of MCF7 tumor genome?
  - What sequence of rearrangements produced MCF7?



Reversals: A Formal Motivation

- A sequence of n genes is represented by a permutation $\boldsymbol{\pi}$, where a permutation is an ordering of the integers 1 to n*:*

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

$$\rho(i,j)$$

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

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

Note: we don't consider the orientation (sign) of genes here

Reversals: Example

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

$\rho(3,5)$ ↓

$1\ 2\ 5\ 4\ 3\ 6\ 7\ 8$

Reversals: Example

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

$\rho(3,5)$ ↓

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

$\rho(5,6)$ ↓

$1\ 2\ 5\ 4\ 6\ 3\ 7\ 8$

Reversal Distance Problem

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

- Input: Permutations $\boldsymbol{\pi}$ and $\boldsymbol{\sigma}$

- Output: A series of reversals $\rho_1,\ldots\rho_t$ transforming $\boldsymbol{\pi}$ into $\boldsymbol{\sigma}$, such that t is minimized

- The minimal such t is called the **reversal distance** between π and σ and is often written as $d(\pi, \sigma)$.

Sorting By Reversals Problem

- We will often simply assume that one of the permutations being considered is fixed as the most natural permutation, the identity I = (1 2 3 … n); we now restate the problem.

- Sorting By Reversals Problem: Given a permutation, find a shortest series of reversals that transforms it into the identity.

- Input: Permutation $\boldsymbol{\pi}$

- Output: A series of reversals $\rho_1, \ldots \rho_t$ transforming $\boldsymbol{\pi}$ into the identity permutation such that t is minimum. We call the minimum such t the "reversal distance" of π, denoted $d(\pi)$

Sorting by Reversals: Example

- Say we are given $\pi = (2\ 4\ 3\ 5\ 8\ 7\ 6\ 1)$

- We can sort π in four steps as follows:

Step 0:	$\pi = (2\ \underline{4\ 3}\ 5\ 8\ 7\ 6\ 1)$
Step 1:	$(\underline{2\ 3\ 4\ 5}\ 8\ 7\ 6\ 1)$
Step 2:	$(5\ 4\ 3\ 2\ \underline{8\ 7\ 6\ 1})$
Step 3:	$(\underline{5\ 4\ 3\ 2\ 1}\ 6\ 7\ 8)$
Step 4:	$(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$

- But can we sort π in three steps? Two? How can we know?

Sorting by Reversals: Example

- Say we are given $\pi = (2\ 4\ 3\ 5\ 8\ 7\ 6\ 1)$

- We can sort π in four steps as follows:

Step 0:	$\pi = (2\ \underline{4\ 3}\ 5\ 8\ 7\ 6\ 1)$
Step 1:	$(\underline{2\ 3\ 4\ 5}\ 8\ 7\ 6\ 1)$
Step 2:	$(5\ 4\ 3\ 2\ \underline{8\ 7\ 6\ 1})$
Step 3:	$(\underline{5\ 4\ 3\ 2\ 1}\ 6\ 7\ 8)$
Step 4:	$(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$

- But can we sort π in three steps? Two? How can we know?

Pancake Flipping Problem

- We take a short detour to discuss a slightly more specific problem, which first arose as a mathematics problem in the 1970s.
- A chef is sloppy; he prepares an unordered stack of pancakes of different sizes.
- The waiter wants to quickly rearrange them (so that the smallest winds up on top, down to the largest at the bottom).
- He does it by flipping over several from the top of the stack.

Pancake Flipping Problem

- Goal: Given a stack of *n* pancakes, what is the minimum number of flips needed to rearrange them into the perfect stack?
- Input: Stack of *n* pancakes
- Output: A minimal sequence of flips transforming the stack into the perfect stack.
- We should note, however, that this isn't very mathematical…

Pancake Flipping Problem

- Let's label the smallest pancake by 1, the biggest by n, etc.; a stack of pancakes can then be represented by a permutation.
- We can view a flip of the stack as a special reversal (called a "prefix reversal") which must involve the first element.
- With this mathematical framework, we will restate the pancake flipping problem as the "Sorting by Prefix Reversals Problem."

Sorting by Prefix Reversals

- Sorting by Prefix Reversals Problem: Given a permutation π, find the shortest sequence of prefix reversals transforming π into the identity permutation.

- Input: Permutation π

- Output: A series of prefix reversals $\rho_1, \ldots \rho_t$ transforming $\boldsymbol{\pi}$ into the identity permutation and such that t is minimized.

Pancake Flipping Problem: Greedy Algorithm

- Greedy approach: at most 2 prefix reversals at most to place the biggest pancake on the bottom, at most 2 prefix reversals to place the second-biggest pancake in the second position from the bottom…
- This results in an algorithm which requires $2n - 2$ prefix reversals.
- Bill Gates (!) and Christos Papadimitriou showed in the mid-1970s that this problem can actually be solved by at most $5/3\ (n + 1)$ prefix reversals.

Christos Papadimitrou and Bill Gates flip pancakes

Note: This means that our greedy method is "close" to optimal

Sorting By Reversals: A Greedy Algorithm

- We now return to sorting by reversals.
- If the first three elements in permutation π= 1 2 3 6 4 5 are already in order, it does not make any sense to break them.
- The length of the already sorted prefix of π is denoted *prefix* (π)
  - In our example above, *prefix*(π) = 3
- This results in an idea for a greedy algorithm: increase *prefix* (π) at every step.

Greedy Algorithm: Example

- Doing so, π can be sorted in two steps:

$$1\ 2\ 3\ \underline{6\ 4}\ 5$$

$$\downarrow$$

$$1\ 2\ 3\ 4\ \underline{6\ 5}$$

$$\downarrow$$

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

- The number of steps needed to sort a permutation of length n is at most $(n - 1)$.

Note: Why is it $n - 1$ and not n? Think about the final step…

Greedy Algorithm: Pseudocode

<u>SimpleReversalSort(p)</u>

```
for i ← 1 to n – 1
    j ← position of element i in p (i.e., p_j = i)
    if j ≠ i
        p ← p * r(i, j)
        output p
    if p is the identity permutation
        return
```

SimpleReversalSort: Analysis

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

SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: $\underline{6\ 1}\ 2\ 3\ 4\ 5$

SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: <u>6 1</u> 2 3 4 5
  - Step 1: 1 <u>6 2</u> 3 4 5

SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: $\underline{6\ 1}$ 2 3 4 5
  - Step 1: 1 $\underline{6\ 2}$ 3 4 5
  - Step 2: 1 2 $\underline{6\ 3}$ 4 5

SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: $\underline{6\ 1}$ 2 3 4 5
  - Step 1: 1 $\underline{6\ 2}$ 3 4 5
  - Step 2: 1 2 $\underline{6\ 3}$ 4 5
  - Step 3: 1 2 3 $\underline{6\ 4}$ 5

SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: $\underline{6\ 1}$ 2 3 4 5
  - Step 1: 1 $\underline{6\ 2}$ 3 4 5
  - Step 2: 1 2 $\underline{6\ 3}$ 4 5
  - Step 3: 1 2 3 $\underline{6\ 4}$ 5
  - Step 4: 1 2 3 4 $\underline{6\ 5}$

SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: $\underline{6\ 1}$ 2 3 4 5
  - Step 1: 1 $\underline{6\ 2}$ 3 4 5
  - Step 2: 1 2 $\underline{6\ 3}$ 4 5
  - Step 3: 1 2 3 $\underline{6\ 4}$ 5
  - Step 4: 1 2 3 4 $\underline{6\ 5}$
  - Step 5: 1 2 3 4 5 6

SimpleReversalSort: Analysis

- But π can be sorted in two steps:

SimpleReversalSort: Analysis

- But π can be sorted in two steps:
  - Step 0: <u>6 1 2 3 4 5</u>

SimpleReversalSort: Analysis

- But π can be sorted in two steps:
  - Step 0: <u>6 1 2 3 4 5</u>
  - Step 1: <u>5 4 3 2 1</u> 6

SimpleReversalSort: Analysis

- But π can be sorted in two steps:
  - Step 0: <u>6 1 2 3 4 5</u>
  - Step 1: <u>5 4 3 2 1</u> 6
  - Step 2: 1 2 3 4 5 6

Approximation Algorithms

- The previous slide has shown that SimpleReversalSort is not optimal for sorting by reversals.

- Often when we cannot find an optimal algorithm, we will use approximation algorithms instead, which find good approximate solutions rather than optimal ones.

- Define the approximation ratio of an algorithm A on input π as:

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

  where

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

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

Approximation Ratio/Performance Guarantee

- The approximation ratio (performance guarantee) of a minimization algorithm A is the maximum approximation ratio of all inputs of size *n*.

- Formally, for a minimization algorithm A, the approximation ratio of A is as follows:

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

- <u>Big Question</u>: What is the best approximation ratio we can find for sorting by reversals?

Note: What is the approximation ratio for our greedy algorithm for pancake flipping?

Adjacencies and Breakpoints

- Let $\pi = \pi_1\pi_2\pi_3 \ldots \pi_{n-1}\pi_n$ be a permutation. A pair of elements π_i and π_{i+1} is called an **adjacency** if

$$\pi_{i+1} = \pi_i \pm 1$$

  The remaining pairs are called **breakpoints**.

- **Example**:

  π= 1 9 3 4 7 8 2 6 5

  - (3, 4), (7, 8) and (6,5) are adjacent pairs
  - (1,9), (9,3), (4,7), (8,2) and (2,5) are breakpoints.

Extending Permutations

- We put two elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$ at the ends of π.

- **Example**:

$$\pi = 1 \,|\, 9 \,|\, 3 \;\; 4 \,|\, 7 \;\; 8 \,|\, 2 \,|\, 6 \;\; 5$$

Extending with *0* and *10*

$$\pi = 0 \;\; 1 \,|\, 9 \,|\, 3 \;\; 4 \,|\, 7 \;\; 8 \,|\, 2 \,|\, 6 \;\; 5 \,|\, 10$$

- Note that after extension, both a new breakpoint (5, 10) and a new adjacency (0, 1) have been created.

Sorting by Reversals = Breakpoint Elimination

- Define b(π) to be the number of breakpoints in the extension of π.

- Note that sorting by reversals appears to correspond to removal of breakpoints.
  - **Example**:

π = 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		$b(\pi) = 0$

Sorting by Reversals = Breakpoint Elimination

- How many breakpoints can each reversal eliminate?

π = 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	$b(\pi) = 0$

Sorting by Reversals = Breakpoint Elimination

- How many breakpoints can each reversal eliminate?
- Each reversal eliminates at most 2 breakpoints.

π = 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		$b(\pi) = 0$

Sorting by Reversals = Breakpoint Elimination

- How many breakpoints can each reversal eliminate?
- Each reversal eliminates at most 2 breakpoints.
- This implies:
  - $d(\pi) \geq b(\pi) / 2$
  - We will use this idea to create a new greedy algorithm.

π = 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		$b(\pi) = 0$

A Better Greedy Algorithm

<u>BreakPointReversalSort(π)</u>

```
while b(π) > 0
  Among all possible reversals,   choose reversal r
        minimizing b(π · ρ)
  π ← π · ρ(i, j)
  output π
  return
```

A Better Greedy Algorithm

<u>BreakPointReversalSort(π)</u>

```
while b(π) > 0
  Among all possible reversals,   choose reversal r
        minimizing b(π · ρ)
  π ← π · ρ(i, j)
  output π
  return
```

- **Problem**: this algorithm may never terminate.

Strips

- **Strip**: an interval between two consecutive breakpoints in a permutation.
  - **Decreasing strip**: a strip of elements in decreasing order (e.g. 6 5 4 and 3 2).
  - **Increasing strip**: a strip of elements in increasing order (e.g. 6 7 8 9)

0 1 9 4 3 7 8 2 5 6 *10*

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

Reducing the Number of Breakpoints

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

- We will use this result to help adapt our algorithm to one which is guaranteed to terminate.

Idea for New Algorithm

- For π = 1 4 6 5 7 8 3 2

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

  - Choose decreasing strip with the smallest element k in π (k = 2 in this case)

Idea for New Algorithm

- For π = 1 4 6 5 7 8 3 2

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

  - Choose decreasing strip with the smallest element k in π (k = 2 in this case)
  - Find $k - 1$ in π

Idea for New Algorithm

- For π = 1 4 6 5 7 8 3 2

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

  - Choose decreasing strip with the smallest element k in π (k = 2 in this case)
  - Find $k - 1$ in π
  - Reverse the segment between $k - 1$ and k:

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

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

Idea for New Algorithm

- For π = 1 4 6 5 7 8 3 2

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

  - Choose decreasing strip with the smallest element k in π (k = 2 in this case)
  - Find $k - 1$ in π
  - Reverse the segment between $k - 1$ and k:

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

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

  - This gives us a way of decreasing the number of breakpoints, but what if there are no decreasing strips?

The Case of No Decreasing Strips

- If there is no decreasing strip, there may be no reversal that reduces the number of breakpoints (i.e. $b(\pi \bullet \rho) \geq b(\pi)$ for any reversal ρ).
- 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 following step (by Theorem 1).
- Example:

$$\boldsymbol{\pi} = 0 \;\; 1 \;\; 2 \mid 5 \;\; 6 \;\; 7 \mid 3 \;\; 4 \mid 8 \qquad b(\pi) = 3$$

$$\pi \bullet \rho(6,7) = 0 \;\; 1 \;\; 2 \mid 5 \;\; 6 \;\; 7 \mid 4 \;\; 3 \mid 8 \qquad b(\pi) = 3$$

- $\rho(6,7)$ creates a decreasing strip thus guaranteeing that the next step will decrease the # of breakpoints.

ImprovedBreakpointReversalSort

```
ImprovedBreakpointReversalSort(p)
while b(π) > 0
    if π has a decreasing strip
        Among all possible reversals, choose reversal ρ
            that minimizes b(π · ρ)
    else
        Choose a reversal r that flips an increasing strip in
            π
    π ← π · ρ
output π
return
```

Performance Guarantee

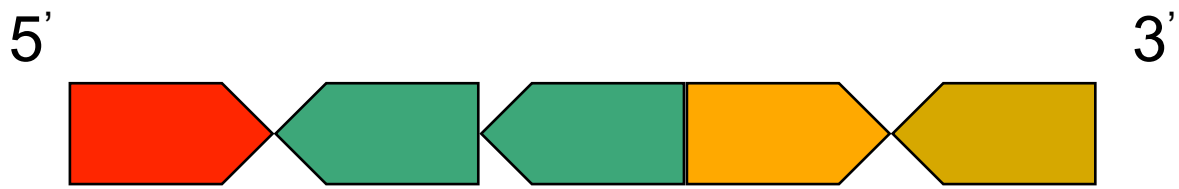
- *ImprovedBreakPointReversalSort* is an approximation algorithm with a performance guarantee of at most 4.
  - It eliminates at least one breakpoint in every two steps; at most $2b(\pi)$ steps.
  - Approximation ratio: $2b(\pi) \ / \ d(\pi)$

- Optimal algorithm eliminates at most 2 breakpoints in every step: $d(\pi) \geq b(\pi) / 2$

- Performance guarantee:
  - $(2b(\pi) / d(\pi)) \leq [2b(\pi) / (b(\pi) / 2)] = 4$

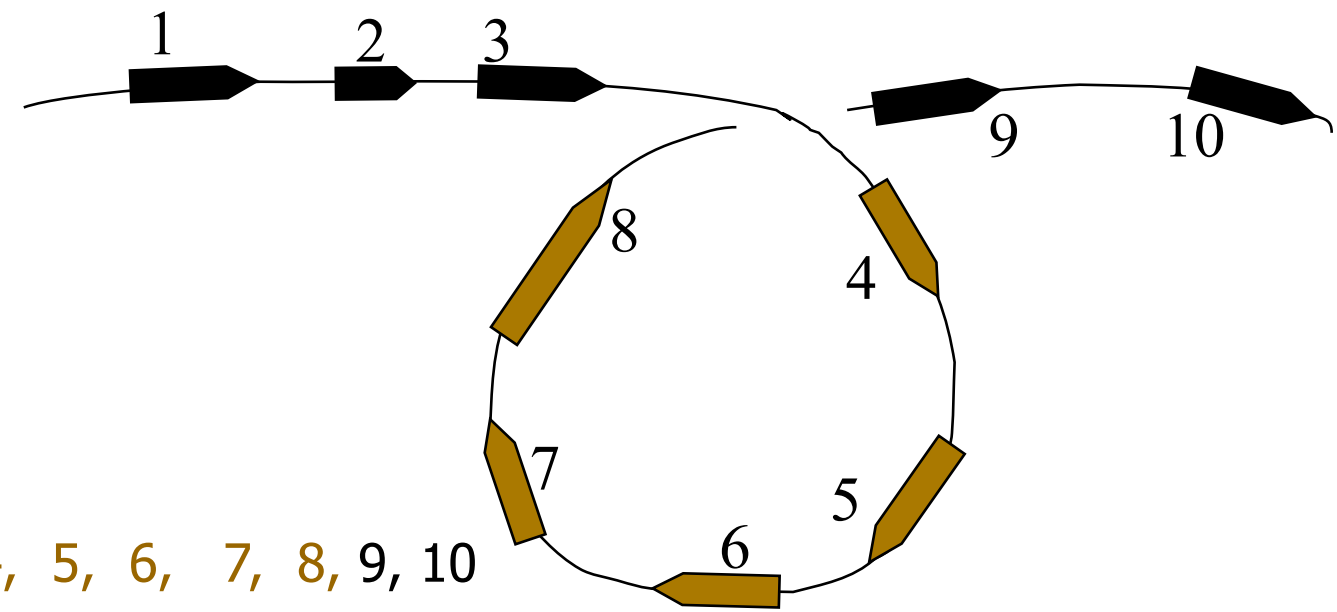
Signed Permutations

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



π = 1 -2 -3 4 -5

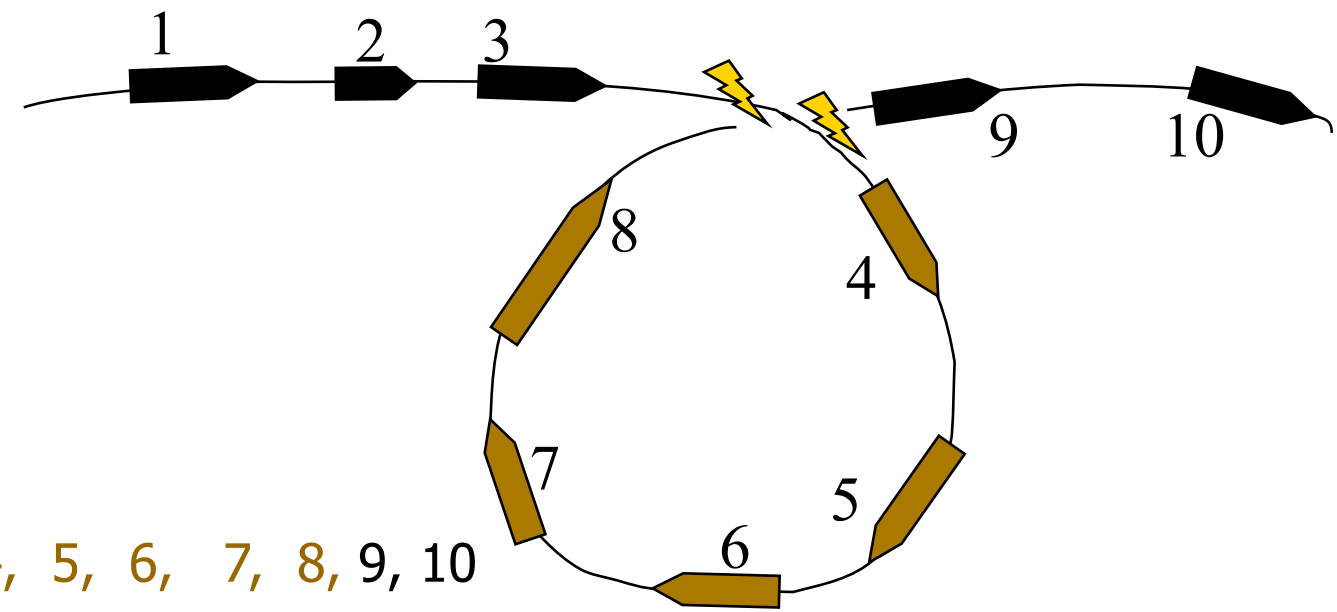
SIGNED Reversals



1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- Blocks represent conserved genes.

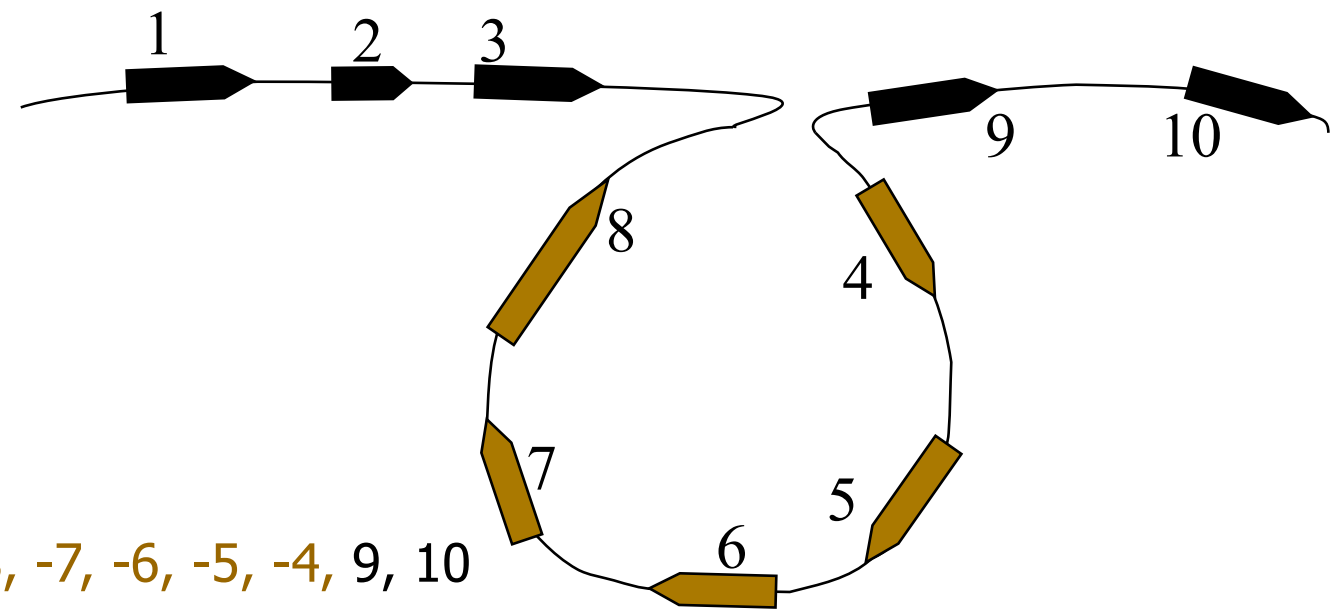
SIGNED Reversals



1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- Blocks represent conserved genes.
- A reversal introduces two breakpoints, represented by

SIGNED Reversals



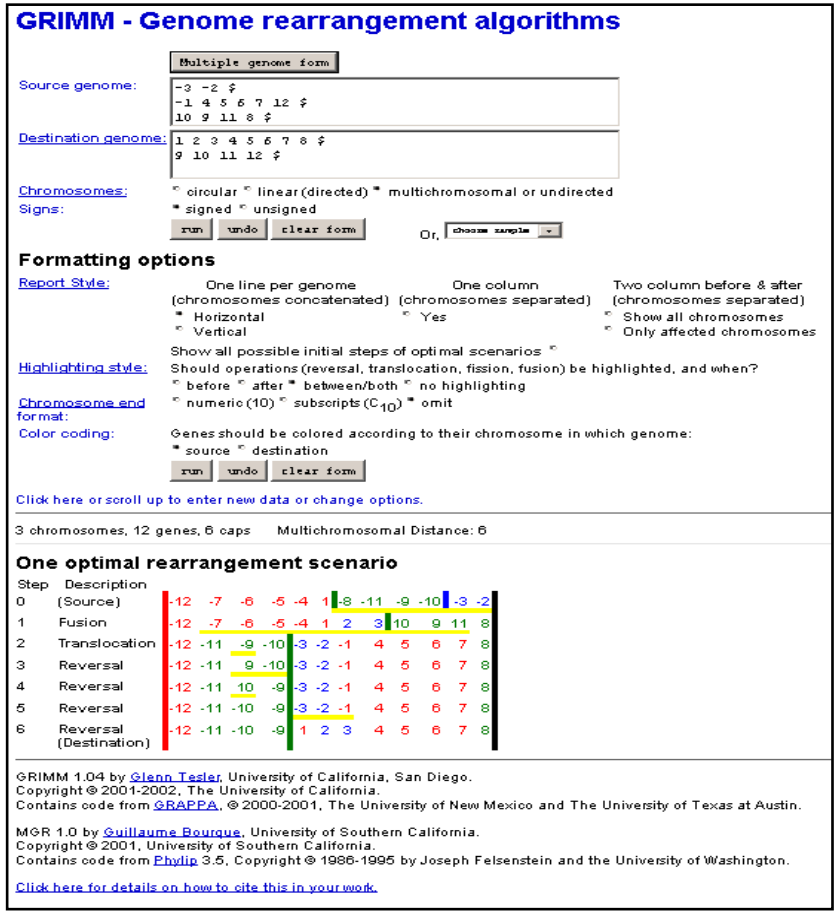
1, 2, 3, -8, -7, -6, -5, -4, 9, 10

- Blocks represent conserved genes.
- A reversal introduces two breakpoints, represented by
- As a result of the reversal, the gene ordering has changed to 1, 2, 3, -8, -7, -6, -5, -4, 9, 10.

GRIMM Web Server

- Real genome architectures are represented by signed permutations.
- Efficient algorithms to sort signed permutations have been developed.
- GRIMM web server computes the reversal distances between signed permutations.

GRIMM Web Server



GRIMM - Genome rearrangement algorithms

Multiple genome form

Source genome: -3 -2 $
-1 4 5 6 7 12 $
10 9 11 8 $

Destination genome: 1 2 3 4 5 6 7 8 $
9 10 11 12 $

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

run undo clear form Or, choose sample

Formatting options

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

Show all possible initial steps of optimal scenarios

Highlighting style: Should operations (reversal, translocation, fission, fusion) be highlighted, and when? before after between/both no highlighting

Chromosome end format: numeric (10) subscripts (C_{10}) omit

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

run undo clear form

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	
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.

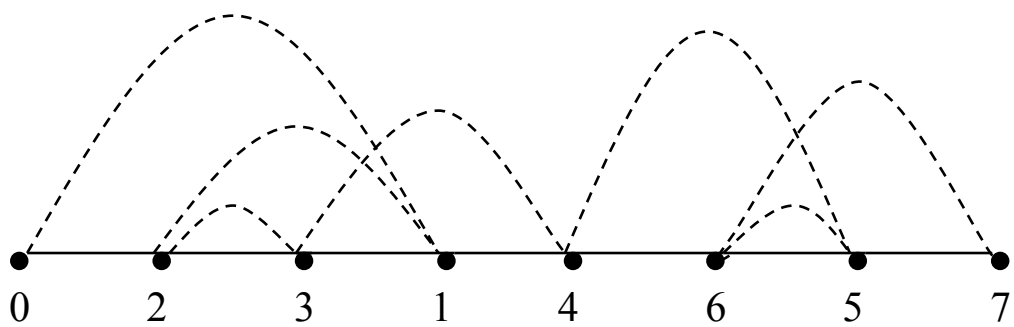
MGR 1.0 by Guillaume Bourque, University of Southern California.
Copyright © 2001, University of Southern California.
Contains code from Phylip 3.5, Copyright © 1986-1995 by Joseph Felsenstein and the University of Washington.

Click here for details on how to cite this in your work.

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

Additional Application: Breakpoint Graphs

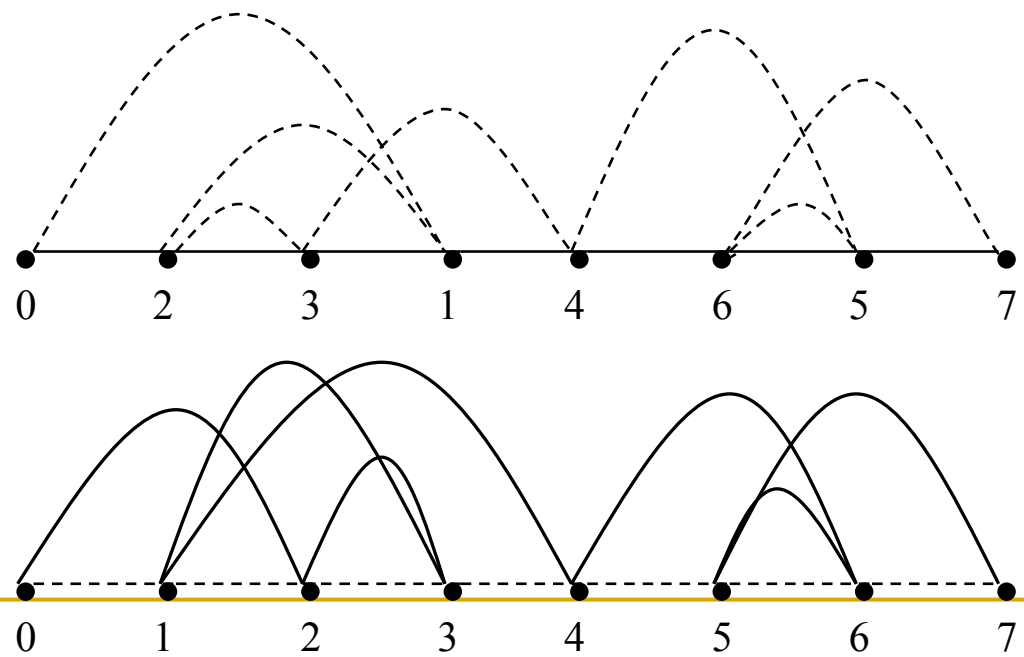
1) Represent the elements of the permutation π = 2 3 1 4 6 5 as vertices in a graph (ordered along a line).
2) Connect vertices in order given by π with solid edges.
3) Connect vertices in order given by 1 2 3 4 5 6 with dotted edges.
4) Superimpose solid and dotted paths.



Additional Application: Breakpoint Graphs

- If we line up the dotted path (instead of the solid path) on a horizontal line, then we would get the following graph.

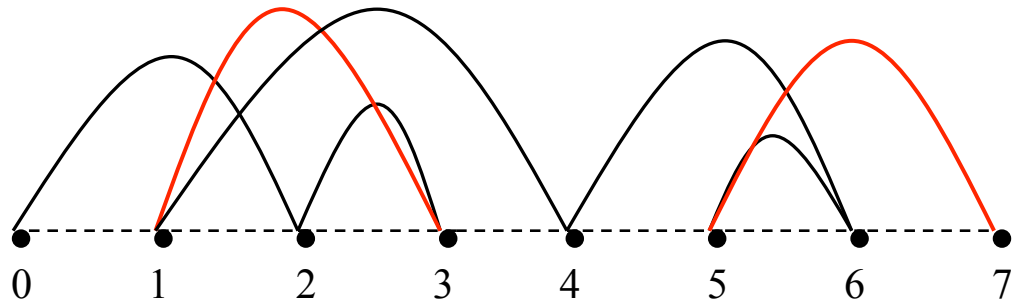
- Although they look different, these graphs are the same.



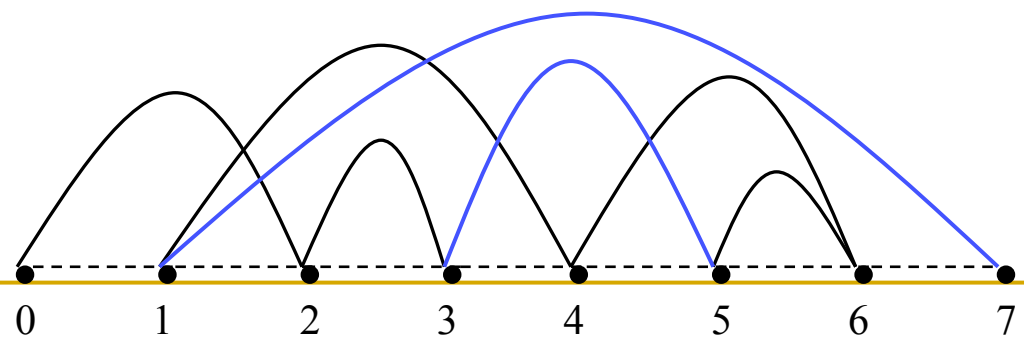
Additional Application: Breakpoint Graphs

- How does a reversal change the breakpoint graph?
  - The dotted paths stay the same for both graphs.
  - However, the red (solid) edges are replaced with blue ones.

Before: 0 2 3 1 4 6 5 7

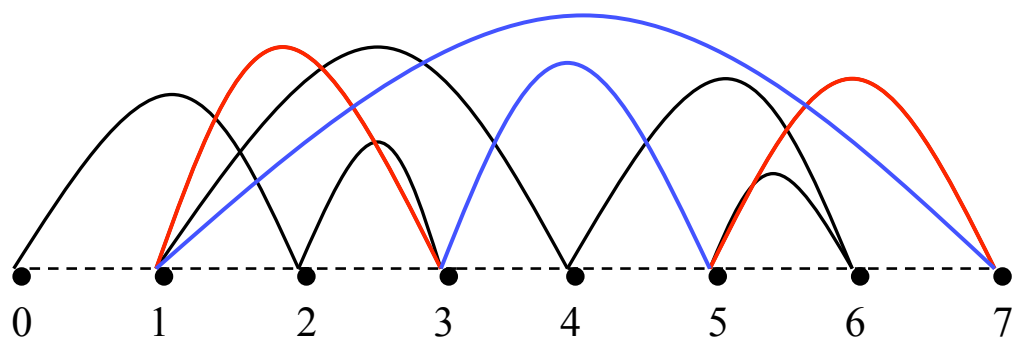


After: 0 2 3 5 6 4 1 7



Additional Application: Breakpoint Graphs

- A reversal removes 2 edges (red) and replaces them with 2 new edges (blue).



Additional Application: Breakpoint Graphs

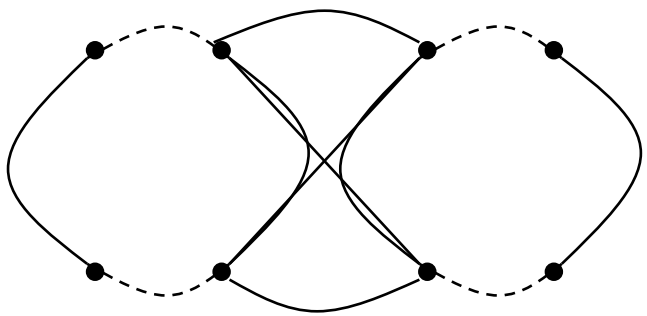
Case 1:

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
- (b) Or after this replacement, there still exists 1 cycle

$\Rightarrow$ $c(\pi\rho) - c(\pi) = 0$

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



Additional Application: Breakpoint Graphs

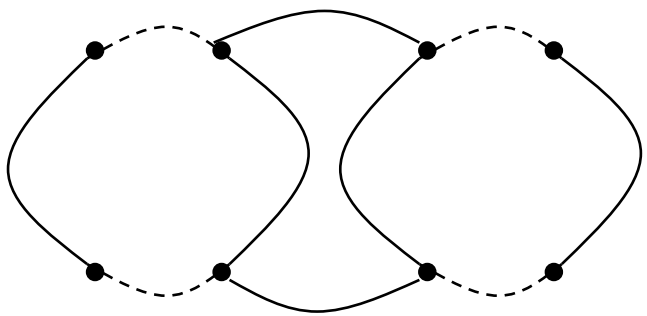
<u>Case 2</u>:

The 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

$\Rightarrow$ $c(\pi\rho) - c(\pi) = -1$

Therefore, for every permutation π and reversal ρ, $c(\pi\rho) - c(\pi) \leq 1$



Additional Application: Breakpoint Graphs

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

- $c(identity) - c(\pi)$ equals the number of cycles that need to be "added" to $c(\pi)$ while transforming π into the identity

- Based on the previous theorem, at best after each reversal, the cycle decomposition could increased by one, then:

$$d(\pi) = c(identity) - c(\pi) = n+1 - c(\pi)$$

- Yet, not every reversal can increase the cycle decomposition

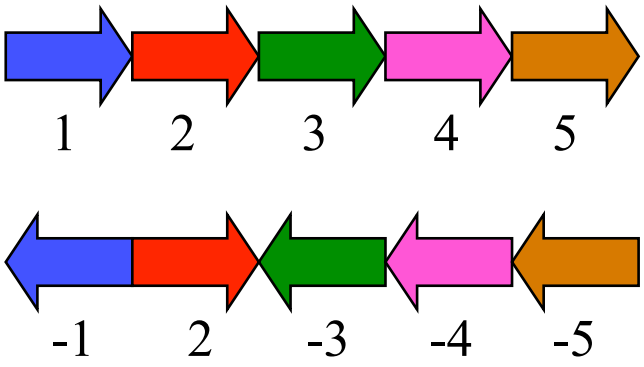
➡ *Therefore,* $d(\pi) \geq n+1 - c(\pi)$

From Signed to Unsigned Permutations

- Recall that genes are *directed* fragments of DNA and we represent a genome by a signed permutation

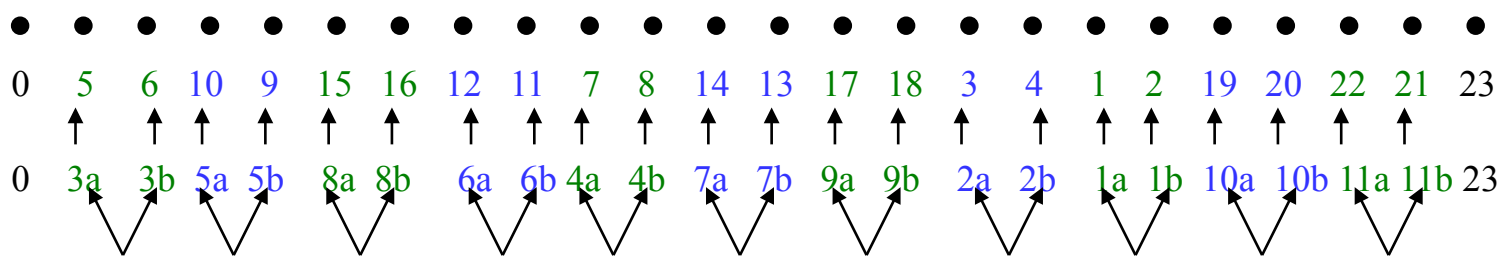
- If genes are in the same position but there orientations are different, they do not have the equivalent gene order

- For example, these two permutations have the same order, but each gene's orientation is the reverse; therefore, they are not equivalent gene sequences



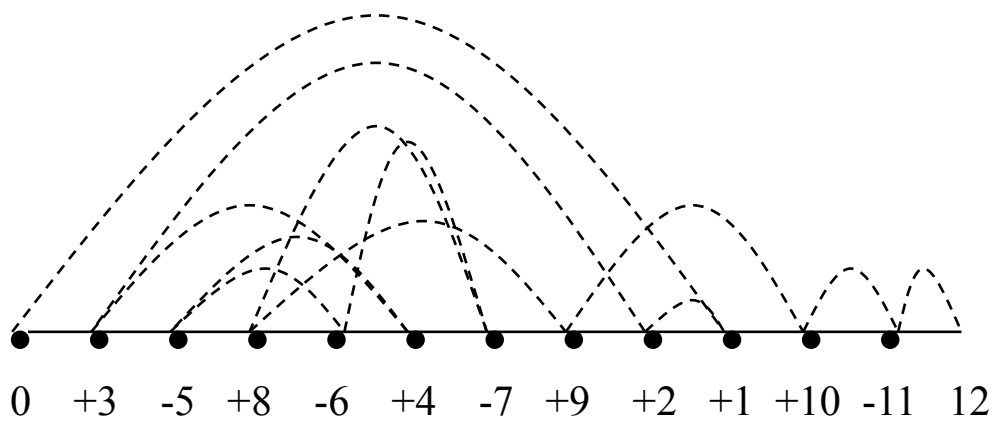
From Signed to Unsigned Permutations

- 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*
  - *If vertex x is negative, replace vertex x with vertex 2x and vertex 2x-1*
  - *The extension vertices x = 0 and x = n+1 are kept as they were before*



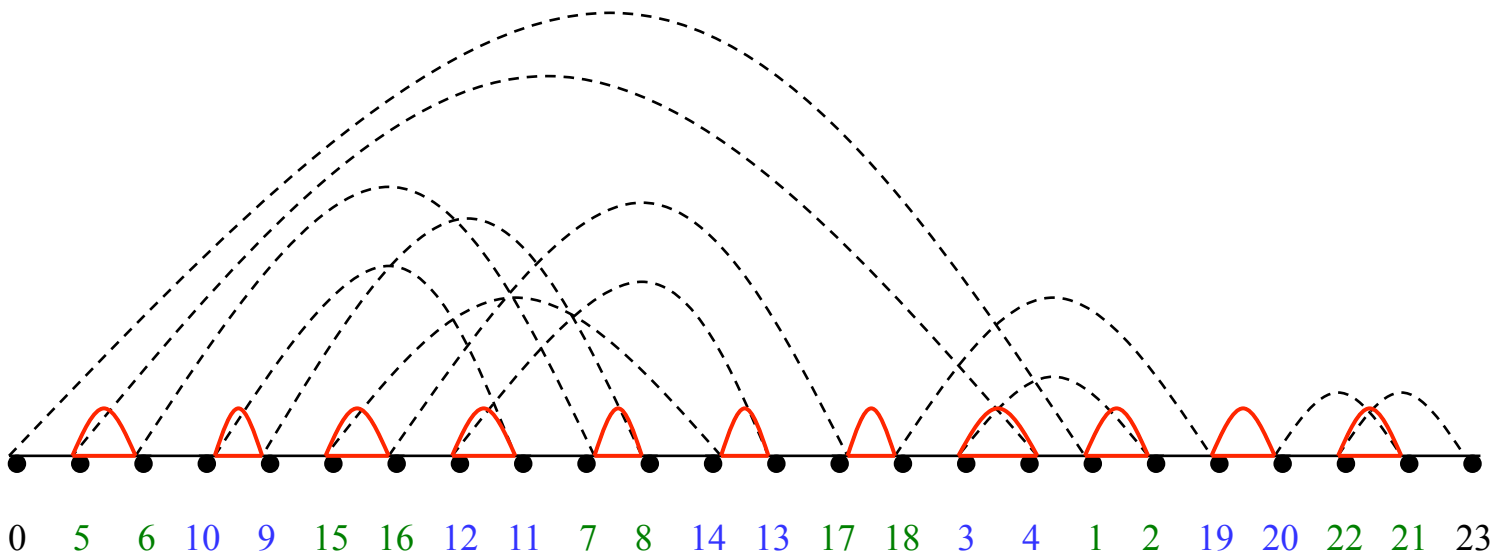
From Signed to Unsigned Permutations

- 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*
  - *If vertex x is negative, replace vertex x with vertex 2x and vertex 2x-1*
  - *The extension vertices x = 0 and x = n+1 are kept as they were before*



From Signed to Unsigned Permutations

- 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

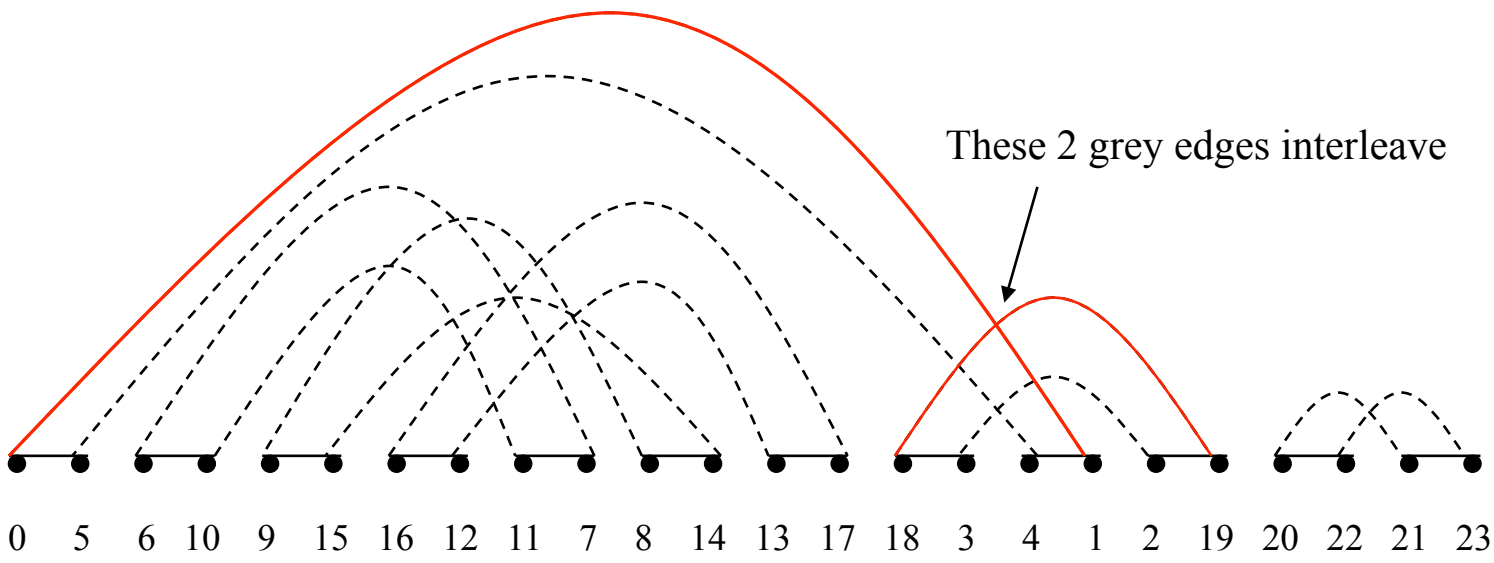


Interleaving Edges

- Interleaving edges are grey edges that cross each other

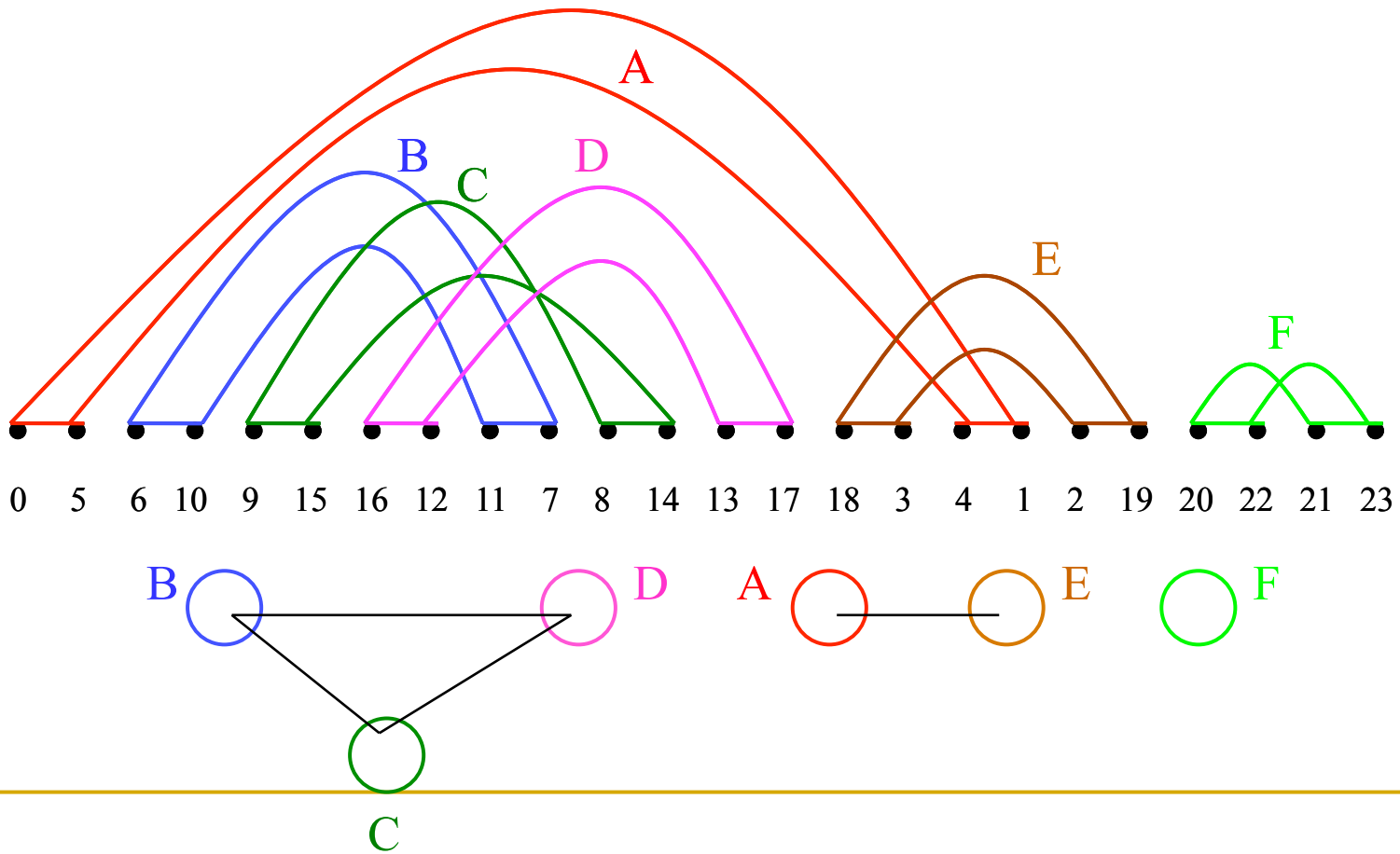
  Example: Edges (0,1) and (18, 19) are interleaving

- Cycles are interleaving if they have an interleaving edge



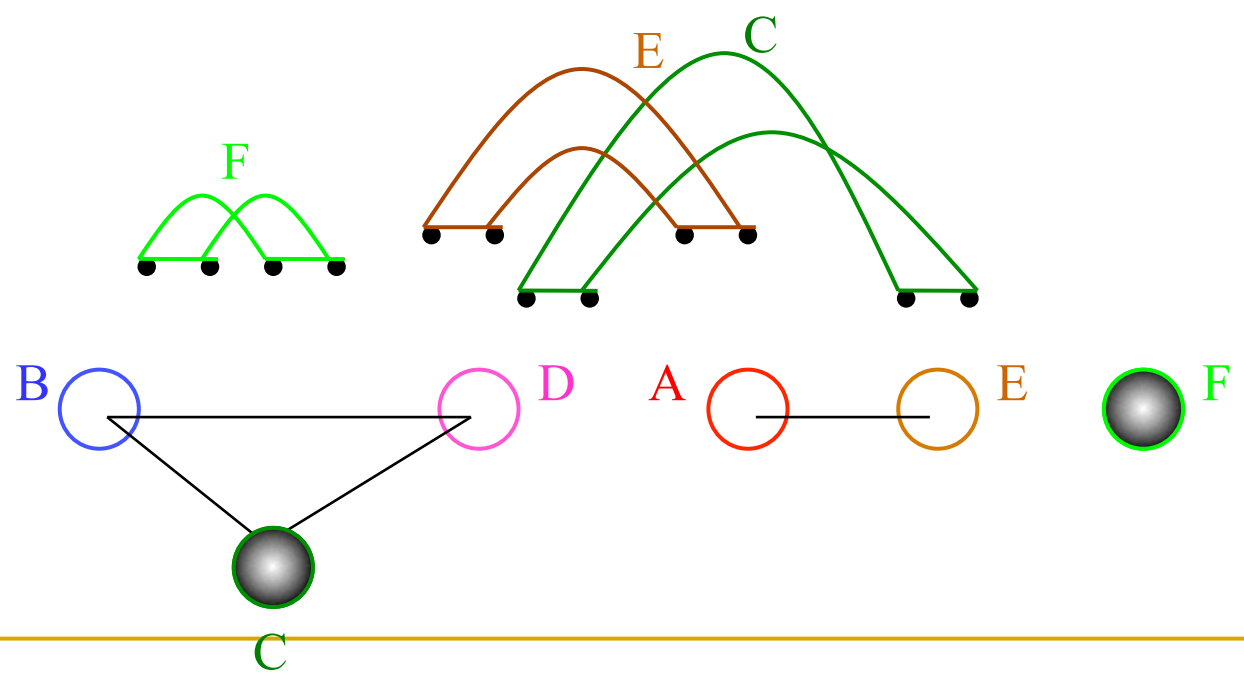
Interleaving Graphs

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



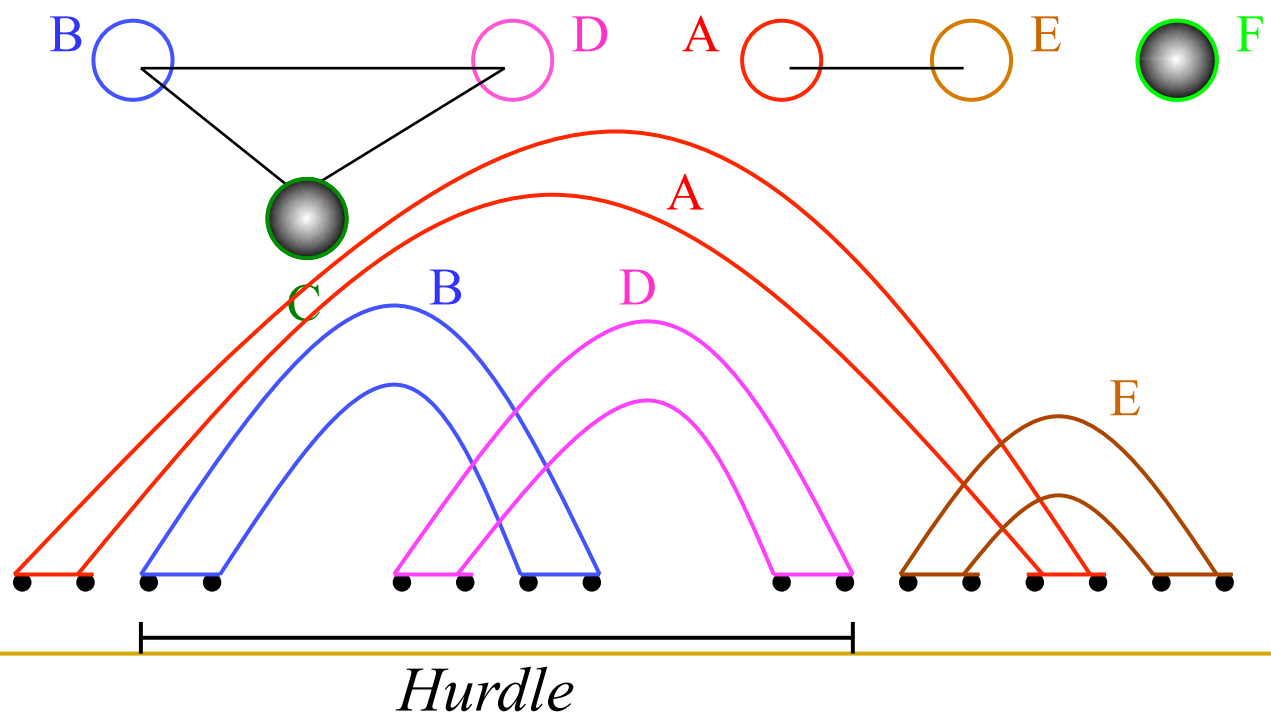
Interleaving Graphs

- Oriented cycles are cycles that have the following form
- Mark them on the interleave graph
- Unoriented cycles are cycles that have the following form
- In our example, A, B, D, E are unoriented cycles while C, F are oriented cycles



Hurdles

- Remove the oriented components from the interleaving graph
- The following is the breakpoint graph with these oriented components removed
- Hurdles are connected components that do not contain any other connected components within them



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(π) be the number of hurdles in permutation π
- Taking into account of hurdles, the following formula gives a tighter bound on reversal distance:

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