

# **Genome Rearrangement**

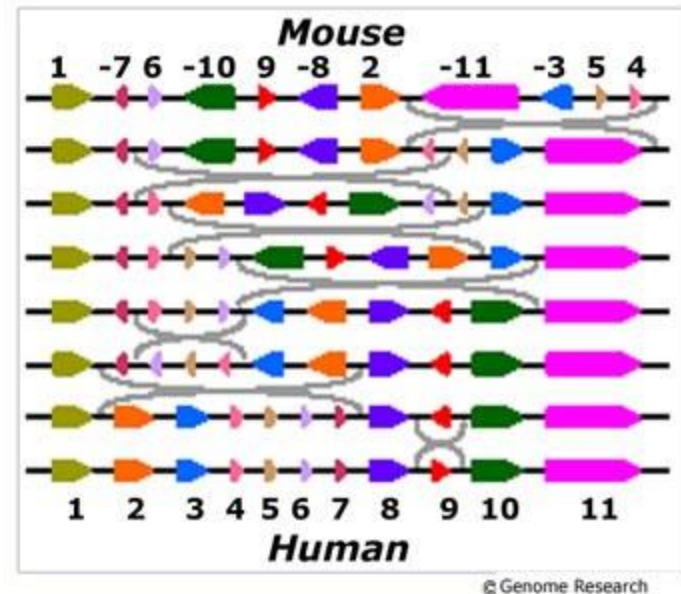
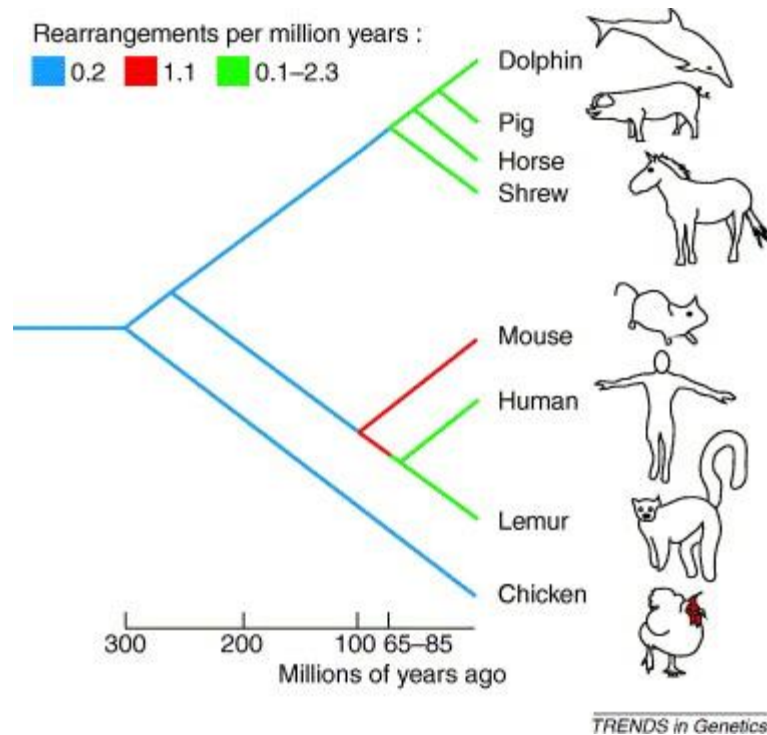


# Background



- In the late 1980's Jeffrey Palmer and colleagues discovered a remarkable and novel pattern of evolutionary change in plant organelles. They mapped the mitochondrial genomes of *Brassica oleracea* (cabbage) and *Brassica campestris* (turnip), which are very closely related (many genes are 99% ~ 99.9% identical), differ dramatically in gene order.

# Background

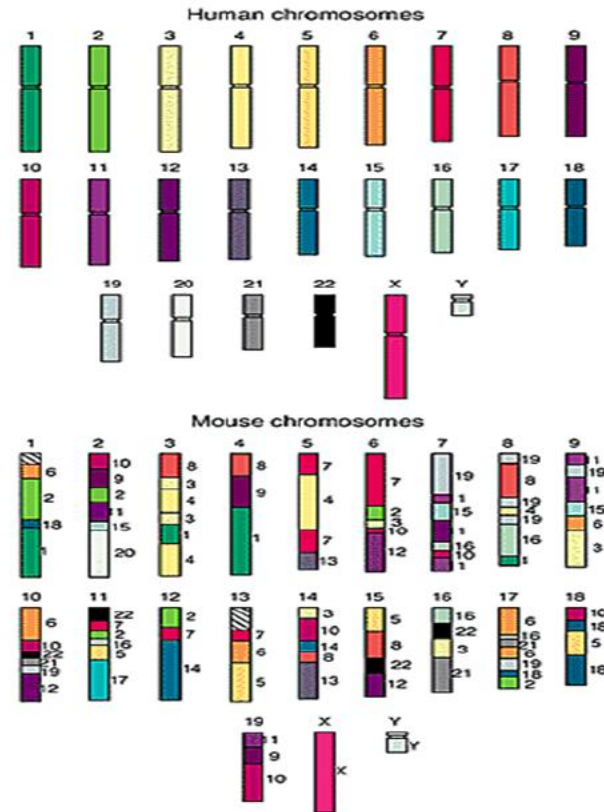


Biologists have found that the related genes in man and mouse are not chaotically distributed over the genomes, but form 'conserved blocks' instead. These conserved blocks reveal the genetic organization of the common ancestor of human and mouse, allowing Pevzner and Tesler to reconstruct a rearrangement scenario of man-mouse evolution. Genomic sequences reveal at least 11 synteny blocks (where human and mouse genes are in the same order) of one million DNA letters or longer on the X chromosome. They provide evidence of at least 7 inversions (a type of rearrangement) which emanate from a common ancestor in the middle. Two of the 11 blocks show evidence of extensive micro-rearrangements. (Graphic by Glenn Tesler, UCSD)

*UCSD Computer Science and Engineering  
Department Researchers Co-author Landmark  
Papers in 'Nature' and 'Genome Research' Journals*

# Background

- Humans and mice have similar genomes, but their genes are ordered differently.
- 245 rearrangements (reversals, fusions, fissions, translocations)



*UCSD Computer Science and Engineering  
Department Researchers Co-author Landmark  
Papers in 'Nature' and 'Genome Research' Journals*

# Background

[https://en.wikipedia.org/wiki/Waardenburg\\_syndrome](https://en.wikipedia.org/wiki/Waardenburg_syndrome)

- 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 in chromosome 2

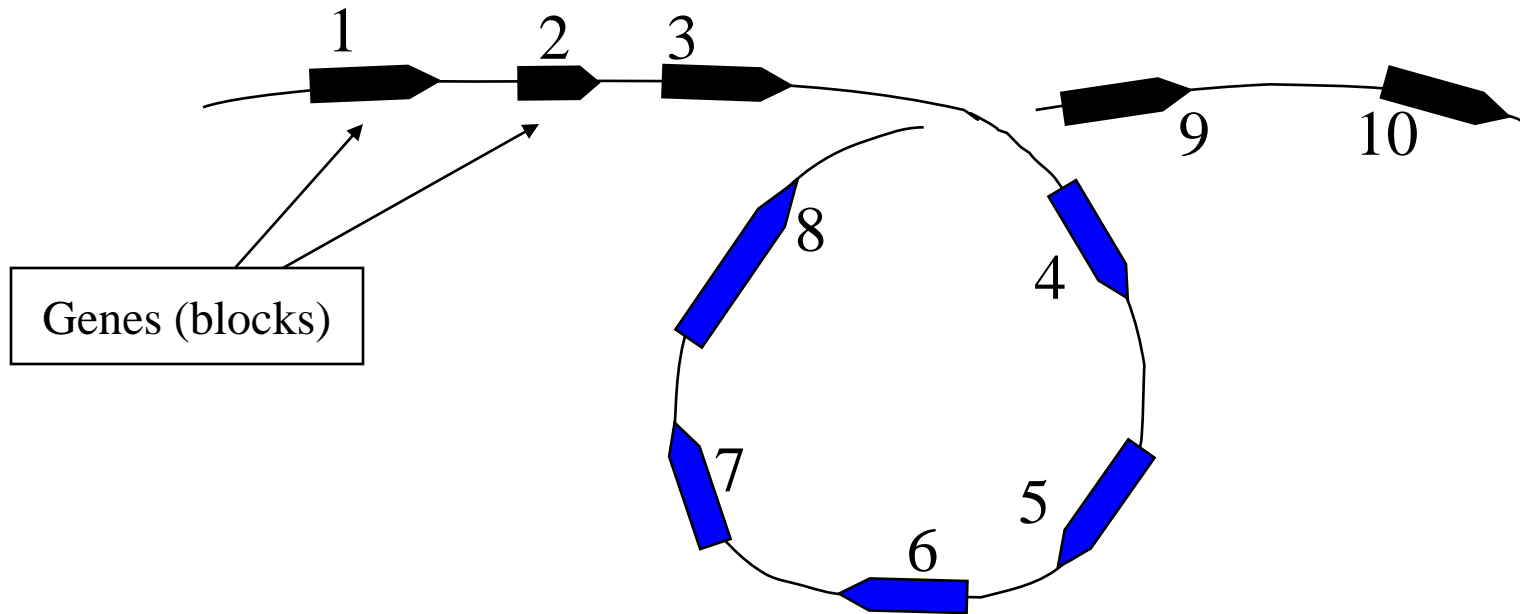


- breed of mice (with splotch gene) had similar symptoms caused by the same type of gene as in humans
- Scientists succeeded in identifying location of gene responsible for disorder in mice
- Finding the gene in mice gives clues to where the same gene is located in humans

# Genome Rearrangement

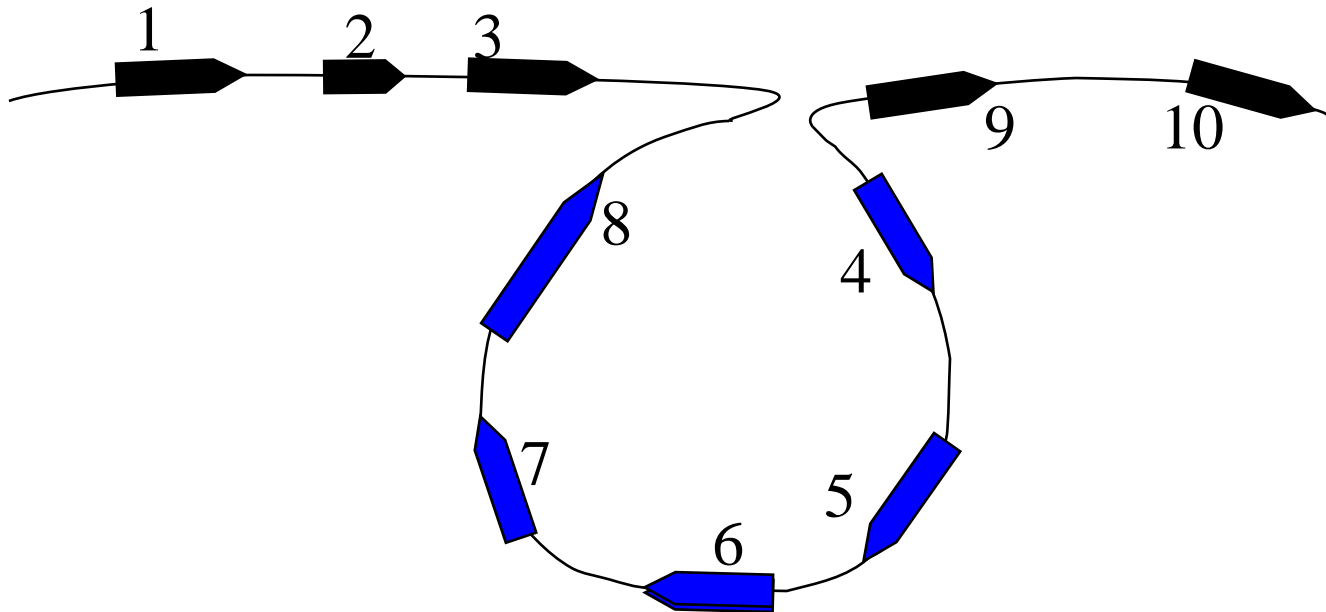
- Input: Two genomes which contains the same set of genes, but the order of genes are different.
- Goal: Find the shortest sequence of rearrange operations transforming one genome to another.
- Since we are interested in the order of genes, we label each gene a unique number, 1, 2, 3, ...,  $n$ .
- We may view the problem as a sorting problem, with some special operations (such as transposition and reversal).

# Reversals



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

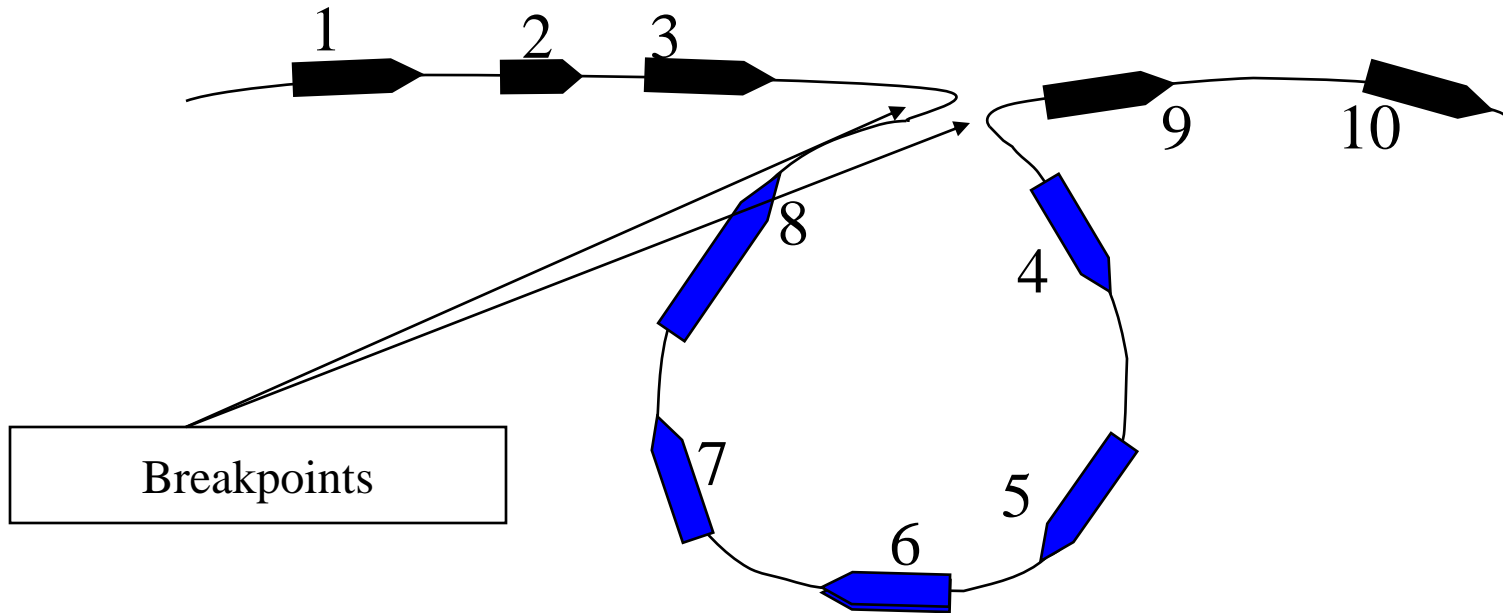
# Reversals



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



# Reversals



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

# Terminologies

- $G = "1 -5 4 -3 2"$
- $-g$ : the reverse of gene  $g$ 
  - Example: gene  $5 = "GCTGA"$ ,  $-5 = "AGTCG"$
- Transposition: swap two adjacent substrings of any length without changing the order of the two substrings
  - $3 \text{ } \underline{1\ 5} \text{ } \underline{2\ 4} \rightarrow 3 \text{ } \underline{2\ 4} \text{ } \underline{1\ 5}$
- Reversal: invert the order of a substring of any length
  - $1 \text{ } \underline{-5\ 4\ -3} \text{ } 2 \rightarrow 1 \text{ } \underline{3\ -4\ 5} \text{ } 2$

# Terminologies

- Transposition:  $\rho(i, j, k)$

$$\left( \begin{array}{ccccccc} 1 & \dots & i-1 & \boxed{i \ i+1 \ \dots \ \dots \ j-2 \ j-1} & \boxed{j \ \dots \ k-1} & k & \dots \ n \\ 1 & \dots & i-1 & \boxed{j \ \dots \ k-1} & \boxed{i \ i+1 \ \dots \ \dots \ j-2 \ j-1} & k & \dots \ n \end{array} \right)$$

e.g.  $\pi = \{4 \ 5 \ 1 \ 6 \ 3 \ 2\}$ ,  $\pi \cdot \rho(4,1,2) = \{\underline{1 \ 6 \ 3} \ \underline{4 \ 5} \ 2\}$

- Unsigned reversal:
  - $3 \ \underline{1 \ 5 \ 2} \ 4 \rightarrow 3 \ 2 \ 5 \ 1 \ 4.$
- Signed reversal:
  - $3 \ \underline{1 \ 5 \ 2} \ 4 \rightarrow 3 \ -2 \ -5 \ -1 \ 4.$

# Four types of Rearrangements

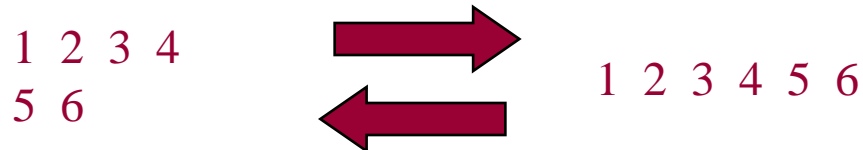
## Reversal



## Translocation



## Fusion



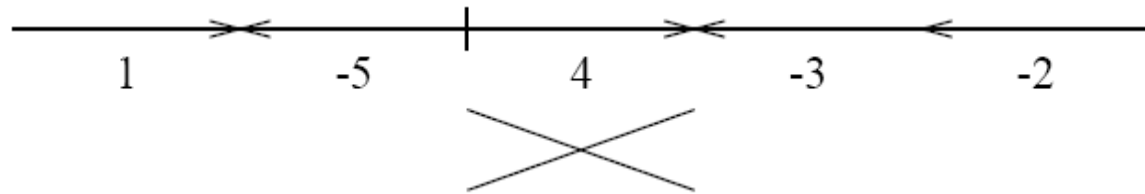
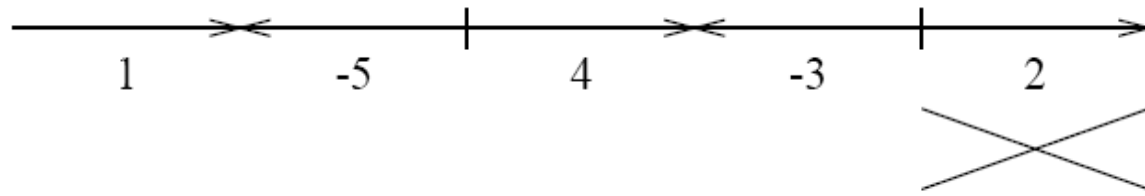
## Fission

# Genome Rearrangement

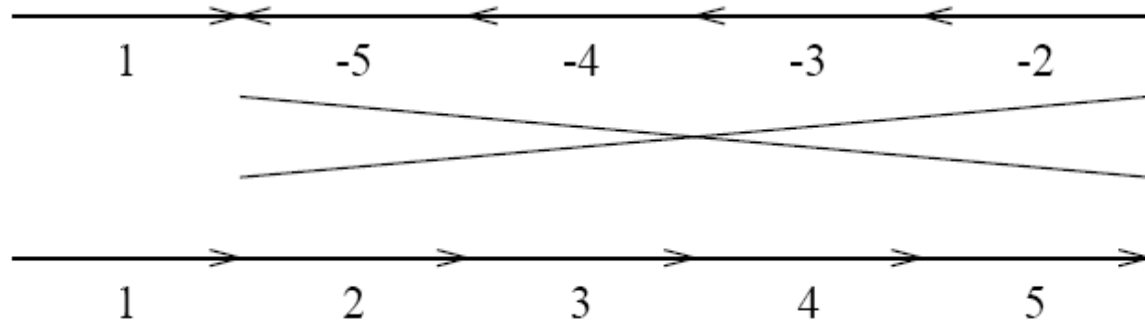
- Consider reversals only.
  - These are most common. “... *genome rearrangements is a common mode of molecular evolution in mitochondrial, chloroplast, viral and bacterial DNA*” [Hannenhalli & Pevzner 1999].
- How to transform one genome (i.e., gene ordering) to another, using the least number of reversals ?

# Sorting by Reversal

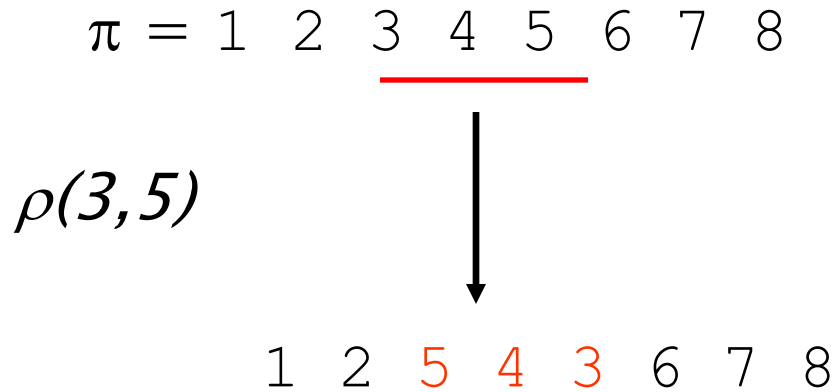
*B. oleracea*  
(cabbage)



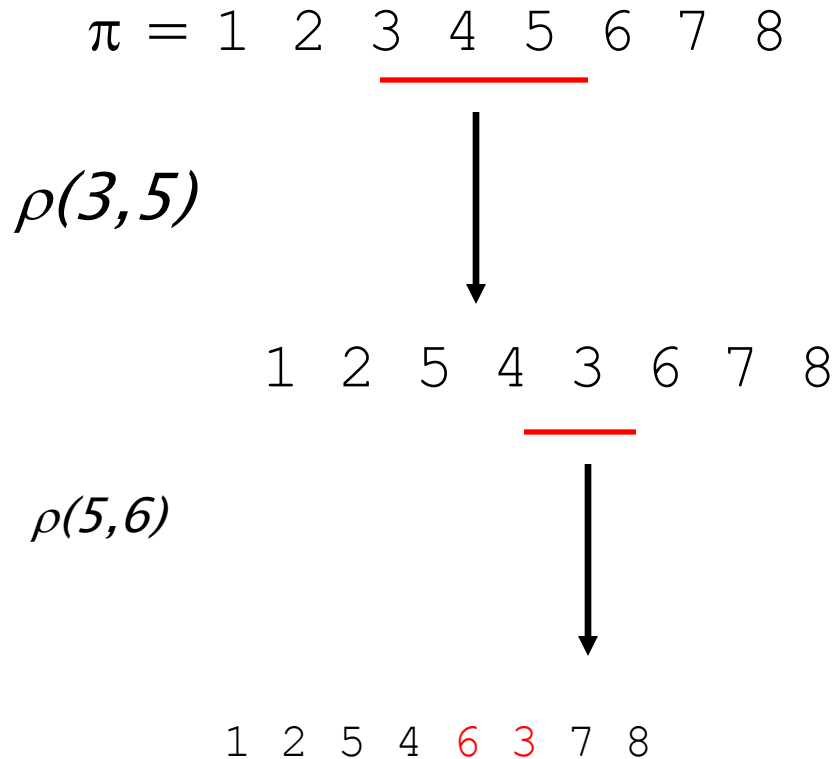
*B. campestris*  
(turnip)



# Reversals: Example



# Reversals: Example






# Reversals and Gene Orders

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

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


---

$\rho(i,j)$   


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

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

# 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  $\sigma$

# Different way of stating the same problem: Sorting By Reversals

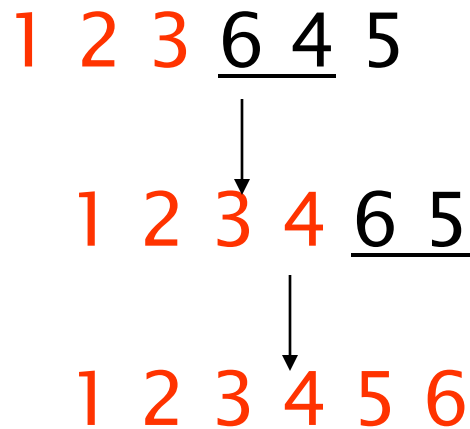
- Goal: Given a permutation, find a shortest series of reversals that transforms it into the identity permutation  $(1\ 2\ \dots\ n)$
- Input: Permutation  $\pi$
- Output: A series of reversals  $\rho_1, \dots, \rho_t$  transforming  $\pi$  into the identity permutation such that  $t$  is minimum

# 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

# Greedy Algorithm: An Example

- Doing so,  $\pi$  can be sorted



- Number of steps to sort permutation of length  $n$  is at most  $(n - 1)$

# Greedy Algorithm: Pseudocode

## SimpleReversalSort( $\pi$ )

```
1 for  $i \leftarrow 1$  to  $n - 1$ 
2    $j \leftarrow$  position of element  $i$  in  $\pi$  (i.e.,  $\pi_j = i$ )
3   if  $j \neq i$ 
4      $\pi \leftarrow \pi * \rho(i, j)$ 
5   output  $\pi$ 
6 if  $\pi$  is the identity permutation
7   return
```

# Greedy Algorithm: Example

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

# Analyzing SimpleReversalSort

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five 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



# 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

# Approximation Algorithms

- These algorithms find approximate solutions rather than optimal solutions
- The approximation ratio of **an algorithm A on input  $\pi$**  is:

$$A(\pi) / \text{OPT}(\pi)$$

where

$A(\pi)$  - solution produced by algorithm A  
 $\text{OPT}(\pi)$  - 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):
  - $\max_{|\pi| = n} A(\pi) / \text{OPT}(\pi)$
- For maximization algorithm:
  - $\min_{|\pi| = n} A(\pi) / \text{OPT}(\pi)$

# Adjacencies and Breakpoints

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

- A pair of elements  $\pi_i$  and  $\pi_{i+1}$  are **adjacent** if

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

- For example:

$$\pi = 1 \ 9 \ 3 \ \underline{4} \ \underline{7} \ \underline{8} \ \underline{2} \ 6 \ \underline{5} \ \underline{\quad}$$

- (3, 4) or (7, 8) and (6,5) are adjacent pairs

# Breakpoints: An Example

There is a **breakpoint** between any pair of adjacent elements that are non-consecutive:

$$\pi = 1 \mid 9 \mid 3 \mid 4 \mid 7 \mid 8 \mid 2 \mid 6 \mid 5$$

- Pairs  $(1,9)$ ,  $(9,3)$ ,  $(4,7)$ ,  $(8,2)$  and  $(2,6)$  form breakpoints of permutation  $\pi$
- $b(\pi)$  - # breakpoints in permutation  $\pi$

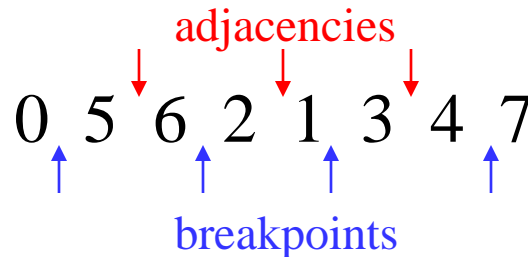
# Adjacency & Breakpoints

- An **adjacency** - a pair of adjacent elements that are **consecutive**
- A **breakpoint** - a pair of adjacent elements that are **not consecutive**

---

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

—————→ Extend  $\pi$  with  $\pi_0 = 0$  and  $\pi_7 = 7$



# Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.

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

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

$$b(\pi) = 5$$

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

$$b(\pi) = 4$$

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

$$b(\pi) = 2$$

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

$$b(\pi) = 0$$

# Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.

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

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

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

$$b(\pi) = 5$$

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

$$b(\pi) = 4$$

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

$$b(\pi) = 2$$

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

$$b(\pi) = 0$$



## Sorting By Reversals: A different greedy algorithm

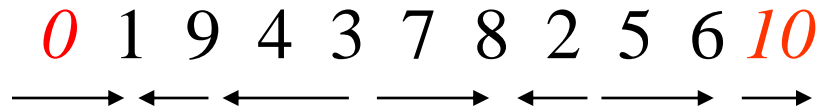
### BreakPointReversalSort( $\pi$ )

```
1  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    output  $\pi$ 
5  return
```

- One problem with this algorithm is that it is not clear why BREAKPOINTREVERSALSORT is a better approximation algorithm than SIMPLEREVERSALSORT.
- Moreover, it is not even obvious yet that BREAKPOINTREVERSALSORT terminates! How can we be sure that removing some breakpoints does not introduce others, leading to an endless cycle?

# Strips

- Strip: an interval between two consecutive breakpoints in a permutation
  - Decreasing strip: *strip* of elements in decreasing order
  - Increasing strip: *strip* of elements in increasing order



- 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  $p$  contains **at least one decreasing** strip, then there exists a reversal  $r$  which decreases the number of breakpoints (i.e.  $b(p \bullet r) < b(p)$  )

# 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

- 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)

# 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)
- Find  $k - 1$  in the permutation

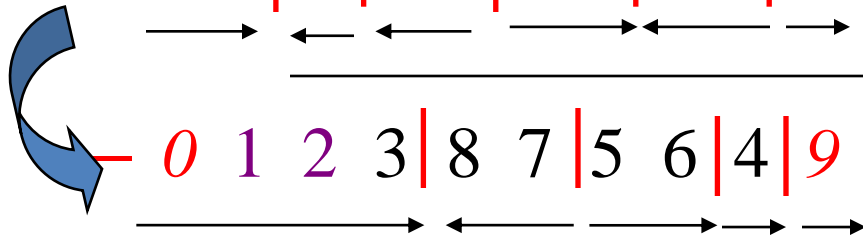
# Things To Consider

- 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)
- Find  $k - 1$  in the permutation
- Reverse the segment between  $k$  and  $k - 1$ :

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



$$b(\pi) = 4$$

## BreakPointReversalSort

- $\pi = 8\ 2\ 7\ 6\ 5\ 1\ 4\ 3$
- $\pi = 0\ 8\ 2\ 7\ 6\ 5\ 1\ 4\ 3\ 9$



## BreakPointReversalSort

- $\pi = 8\ 2\ 7\ 6\ 5\ 1\ 4\ 3$
- $\pi = \textcolor{red}{0} | 8 | 2 | 7\ 6\ 5 | 1 | 4\ 3 | \textcolor{red}{9}$

# ImprovedBreakPointReversalSort

- $\pi = 1\ 5\ 6\ 7\ 2\ 3\ 4\ 8$
- $\pi = \textcolor{red}{0}\ 1\ |5\ 6\ 7|\ 2\ 3\ 4\ |8\ \textcolor{red}{9}$

# 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.  $b(\pi \bullet \rho) \geq b(\pi)$  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).

# ImprovedBreakpointReversalSort

ImprovedBreakpointReversalSort( $\pi$ )

```
1 while  $b(\pi) > 0$ 
2   if  $\pi$  has a decreasing strip
3     Among all possible reversals, choose reversal  $\rho$ 
        that minimizes  $b(\pi \cdot \rho)$ 
4   else
5     Choose a reversal  $\rho$  that flips an increasing
      strip in  $\pi$ 
6      $\pi \leftarrow \pi \cdot \rho$ 
7 output  $\pi$ 
8 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:
    - $( 2b(\pi) / d(\pi) ) \leq [ 2b(\pi) / (b(\pi) / 2) ] = 4$