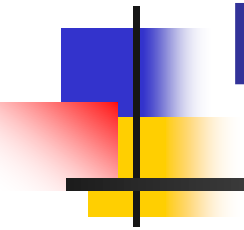


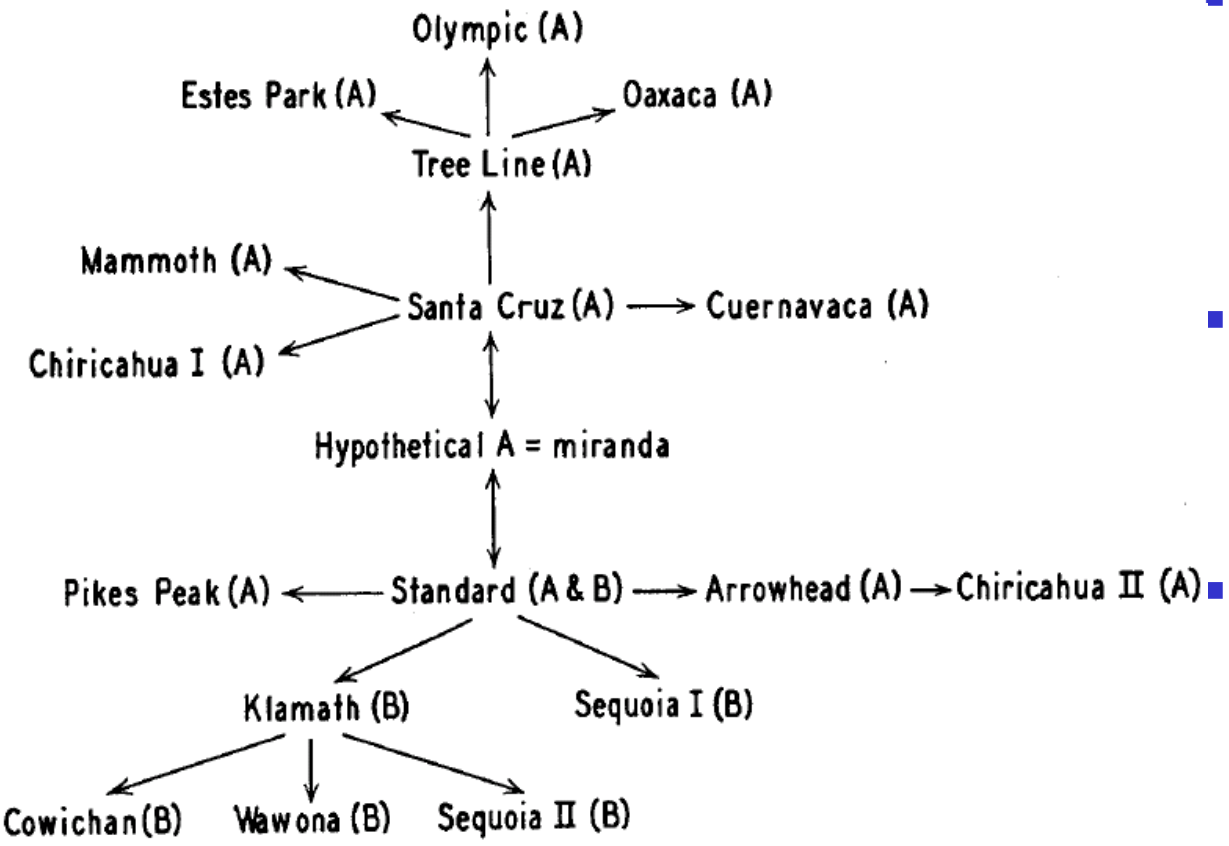
Algorithms in Bioinformatics: A Practical Introduction

Genome Rearrangement

Evidences of Genome Rearrangement

- In 1917, Sturtevant showed that strains of Drosophila melanogaster coming from the same or from distinct geographical localities may differ in having blocks of genes rotated by 180° (reversal).

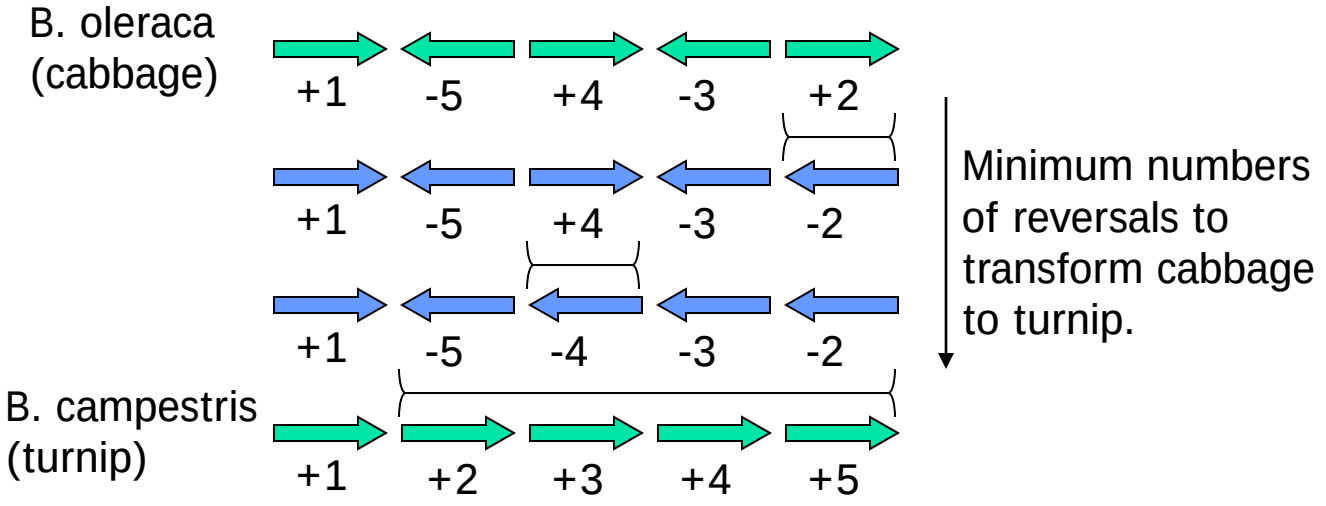
Evidences of Genome Rearrangement



- In 1938, Dobzhansky and Sturtevant studied chromosome 3 of 16 different strains of Drosophila pseudoobscura and Drosophila miranda.
- They observed that the 17 strains from a evolutionary tree where every edge corresponds to one reversal.
- Hence, Dobzhansky and Sturtevant proposed that species can evolve through genome rearrangements.

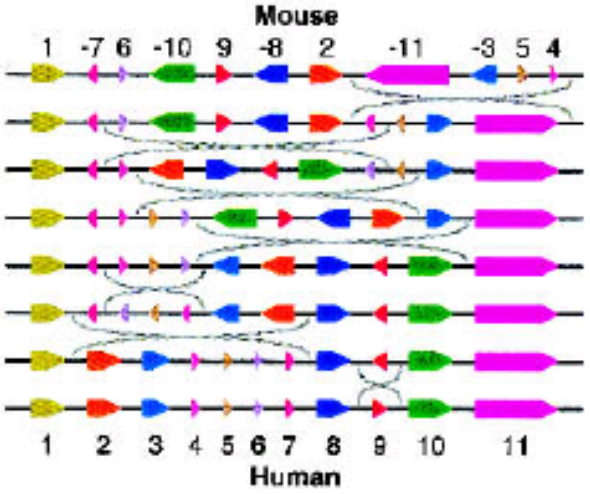
Evidences of Genome Rearrangement

- In 1980s Jeffrey Palmer and co-authors studied evolution of plant organelles by comparing the gene order of mitochondrial genomes
- They pioneered studies of the shortest (most parsimonious) rearrangement scenarios between two genomes.



Evidences of Genome Rearrangement

- Human and mouse are also highly similarity in DNA sequences (98%).
- Moreover, their DNA segments are swapped.
- For example, chromosome X of human can be transformed to chromosome X of mouse using 7 reversals.

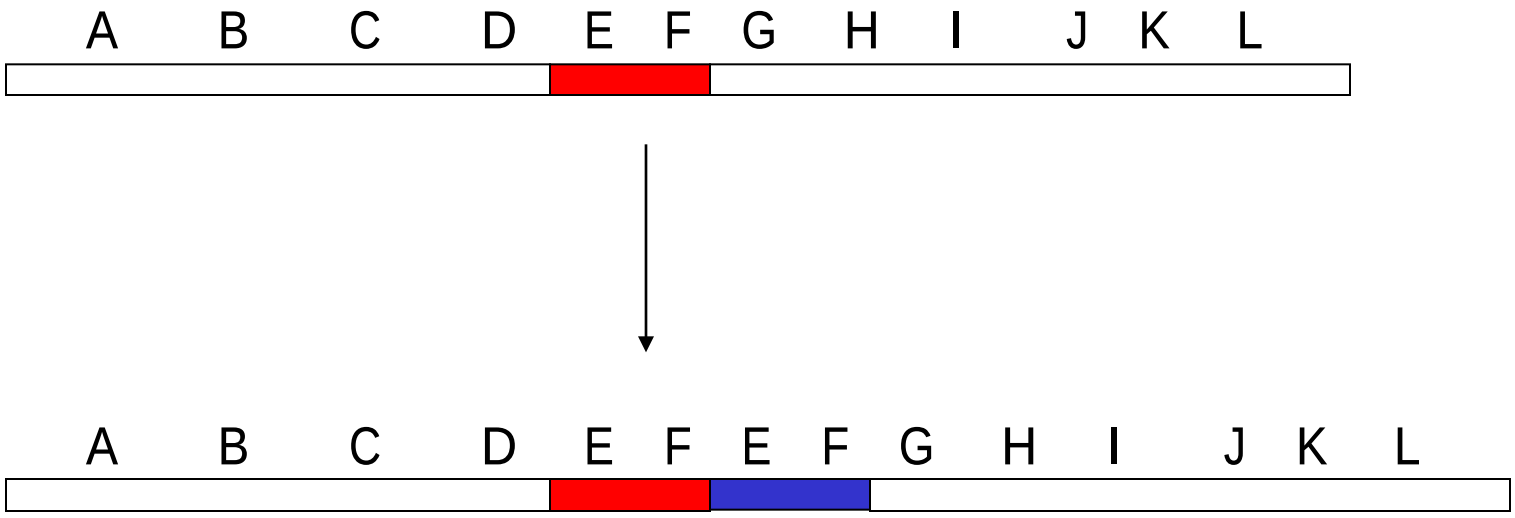


- To transfrom human to mouse, it takes 131 reversals/translocations/fusions/fissions.

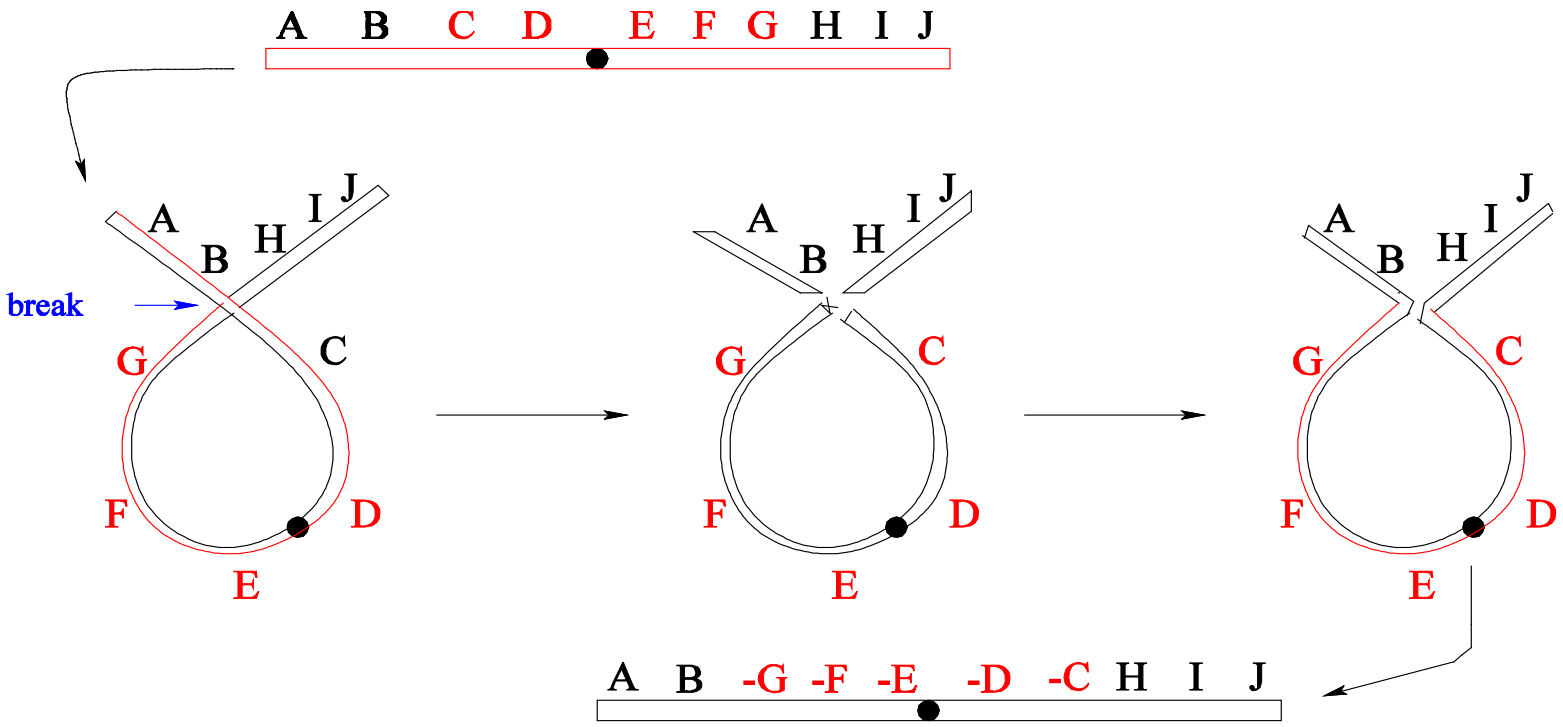
Types of genome rearrangement within one chromosome

- Reversal is just the most common rearrangement. Below, we list the known rearrangement operations within one chromosome:
  - Insertion: Inserting of a DNA segment into the genome (AC→ABC)
  - Deletion: Removal of a DNA segment from the genome (ABC→AC)
  - Duplication: A particular DNA segment is duplicated two times in the genome (ABC→ABBC, ABCD→ABCBD)
  - Reversal: Reversing a DNA segment ($Ab_1b_2b_3C \rightarrow Ab_3b_2b_1C$)
  - Transposition: cutting out a DNA segment and insert it into another location (ABCD→ACBD). This operation is believed to be rare since it requires 3 breakpoints.

Duplication

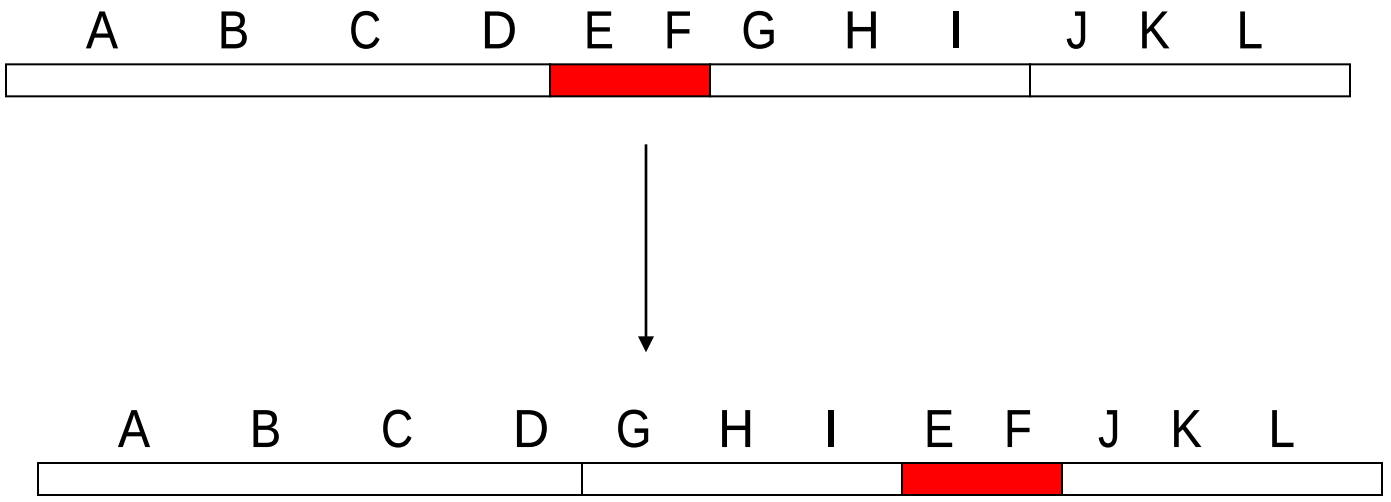


Reversal


A B C D E F G H I J
break
A B H I J
G C
F D
E
A B H I J
G C
F D
E
A B H I J
G C
F D
E
A B -G -F -E -D -C H I J

Transposition

- Transposition involves 3 breakpoints!

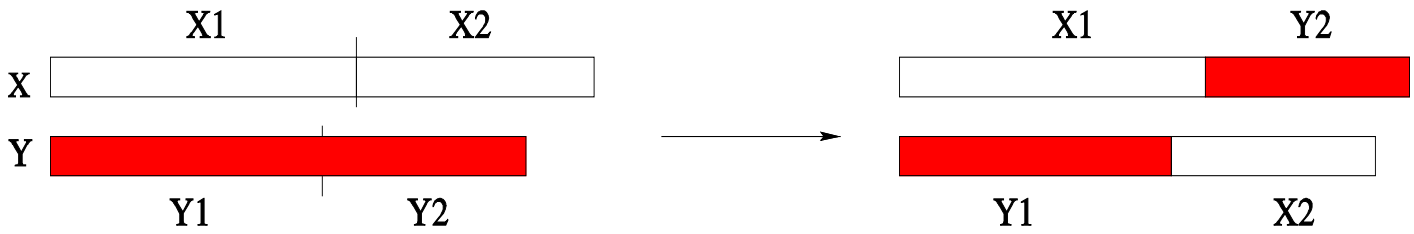


Types of genome rearrangement on two chromosomes (I)

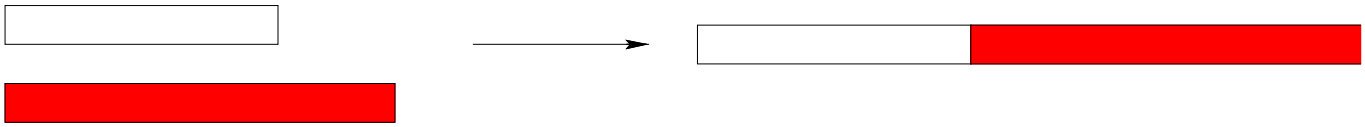
- Translocation: the transfer of a segment of one chromosome to another nonhomologous one.
- Fussion: two chromosomes merge
- Fission: one chromsome splits up into two chromosomes

Genome rearrangement on two chromosomes (II)

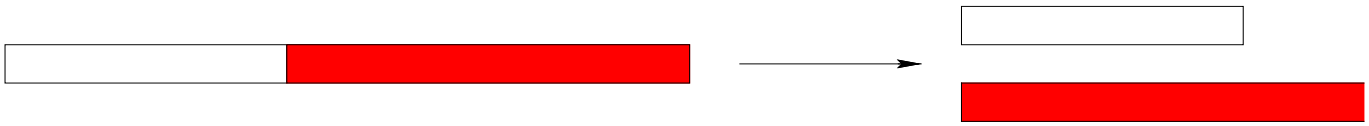
Translocation:

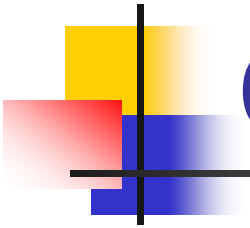


Fusion:



Fission:





Computational problems

- Given two genomes with a set common genes, those genes are arranged in different order in different genomes.
- Our aim is to understand how one genome evolves into another through rearrangements.
- By parsimony, we hope to find the shortest rearrangement path.
- Depending on the allowed rearrangement operations, literature studied the following problems:
  - Genome rearrangement by reversals
  - Genome rearrangement by translocations
  - Genome rearrangement by transpositions

- In this lecture, we focus on genome rearrangement by reversals. This problem is also called sorting by reversals.

Sorting permutation by reversals

- Consider a permutation of {1, 2, …, n}, that is, $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ representing the ordering of n genes in a genome.
- A reversal $\rho(i,j)$ is an operation applying on π, denoted as $\pi \cdot \rho(i,j)$, which reverses the order of the element in the interval [i..j].
- Thus, $\pi \cdot \rho(i,j) = (\pi_1, \ldots, \pi_{i-1}, \pi_j, \ldots, \pi_i, \pi_{j+1}, \ldots, \pi_n)$.
- Example: Let $\pi = (2, 4, 3, 5, 8, 7, 6, 1)$.
  - $\pi \cdot \rho(3,5) = (2, 4, 8, 5, 3, 7, 6, 1)$.

- Our aim is to find the minimum number of reversals that transform π to an identify permutation (1, 2, …, n).
- The minimum number of reversals need to transform π to identity permutation is called the reversal distance, denoted by $d(\pi)$.

Example: sorting unsigned permutation

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

Previous works on sorting unsigned permutation

- Kececioglu and Sankoff (1995): 2-approximation
- Bafna and Pevzner (SIAM Comp 1996): 1.75-approximation
- Caprara (RECOMB 1997, SIAM Discrete Math 2001): NP-hard
- Christie (SODA 1998): 1.5-approximation
- Berman and Karpinski (ICALP 1999): MAX-SNP hard
- Berman, Hannenhalli, Karpinski (ESA 2002): 1.375-approximation

Upper bound on unsigned reversal distance

- A way to transform π to identity permutation is by at most n reversals. The i-th reversal moves element i to position i.
- Example:
  - (4, 5, 3, 1, 2)
  - (1, 3, 5, 4, 2)
  - (1, 2, 4, 5, 3)
  - (1, 2, 3, 5, 4)
  - (1, 2, 3, 4, 5)

Lower bound on unsigned reversal distance

- Let $\pi=(\pi_1, \pi_2, \ldots, \pi_n)$ be a permutation of $\{1, 2, \ldots, n\}$
- There is a breakpoint between π_i and π_{i+1} if $|\pi_i-\pi_{i+1}|>1$.
- Denote $b(\pi)$ be the number of breakpoints in π.
- Since a reversal can reduce at most 2 breakpoints, hence $d(\pi) \geq b(\pi)/2$.

- Example: $\pi=$ • 7 6 5 4 • 1 • 9 8 • 2 3 •
  - Each • is a breakpoint. Thus, $b(\pi) = 5$

- Theorem: $b(\pi)/2 \leq d(\pi) \leq n$.

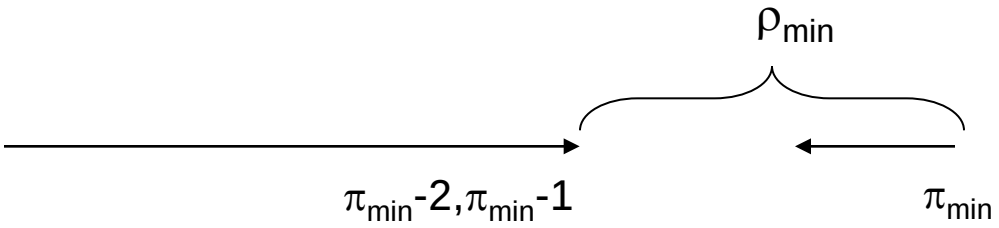
4-approximation algorithm (I)

- A strip is a maximal subsequence without breakpoints.
- A strip is either increasing or decreasing.
- Strip of size 1 is assumed to be decreasing.
  - (There is one exception. We assume there is a hidden '0' on the left of π. And a hidden 'n+1' on the right of π. If the leftmost strip is (1), we say it is increasing. If the rightmost strip is (n), we say it is increasing.)

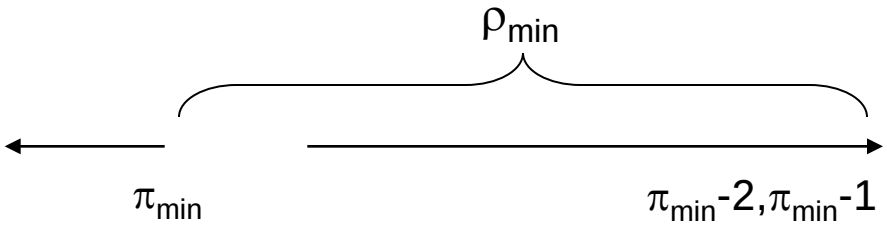
- Example: π=(7, 6, 5, 4, 1, 9, 8, 2, 3)
  - There are three breakpoints: (-,7), (4,1), (1,9), (8,2), (3,-).
  - Hence, there are 4 strips: (7,6,5,4), (1), (9,8), (2,3).
  - Among them, (2,3) is an increasing strip.

4-approximation algorithm (II)

- If π has a decreasing strip,
  - let s_{min} be the decreasing strip in π with the minimal element π_{min}.
  - Let s'_{min} be the strip containing π_{min}-1, which is increasing.
  - let ρ_{min} be the reversal which which arrange π_{min} and π_{min}-1 side by side.



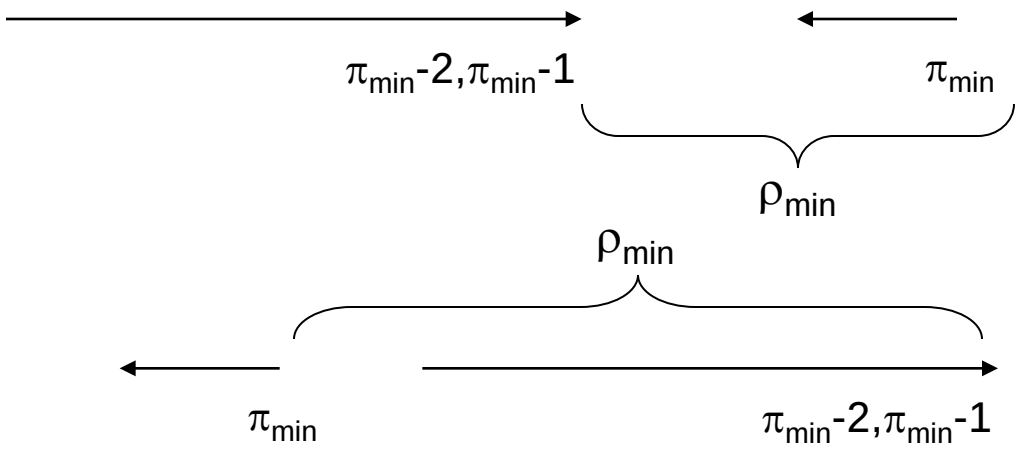
E.g. 8, 9, 14, 7, 6, 5, 1, 2, 10, 11, 3, 4, 16, 14, 13, 12, 15

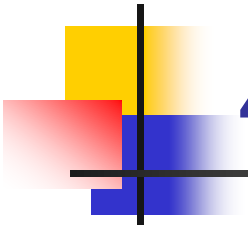


E.g. 8, 9, 3, 4, 14, 7, 6, 5, 1, 2, 10, 11, 16, 14, 13, 12, 15

4-approximation algorithm (III)

- Lemma: If π has a decreasing strip, then $b(\pi \cdot \rho_{min}) - b(\pi) \geq 1$.
- Proof:
  - There are two cases depending on whether s_{min} is to the right or to the left of s'_{min}. As shown in the figure, the reversal ρ_{min} reduces $b(\pi)$ by 1.





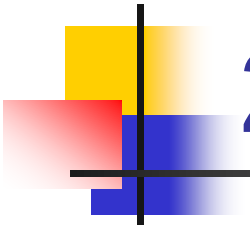
4-approximation algorithm (IV)

- Algorithm simpleApprox
  - while $b(\pi) > 0$,
    - if there exist a decreasing strip,
      - we reverse π by ρ_{min} [this reversal reduces $b(\pi)$ by at least 1];
    - else
      - reverse an increasing strip to create a decreasing strip [$b(\pi)$ does not change]

- The above algorithm will perform at most $2b(\pi)$ reversals.
- The optimal solution performs at least $b(\pi)/2$ reversals.
- Thus, algorithm simpleApprox has approximation ratio 4.

Example

- π=(8, 9, 3, 4, 7, 6, 5, 1, 2, 10, 11)
- π=(8, 9, 3, 4, 5, 6, 7, 1, 2, 10, 11)
- π=(9, 8, 3, 4, 5, 6, 7, 1, 2, 10, 11)
- π=(9, 8, 7, 6, 5, 4, 3, 1, 2, 10, 11)
- π=(9, 8, 7, 6, 5, 4, 3, 2, 1, 10, 11)
- π=(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)



2-approximation algorithm

- Previous method cannot guarantee after resolving each breakpoint, we still have some decreasing strip.

- Idea for this algorithm:
  - We try to ensure we have decreasing strip after resolving each breakpoint.
  - If we fail to ensure that there is a decreasing strip, we show that we can resolve two breakpoints.

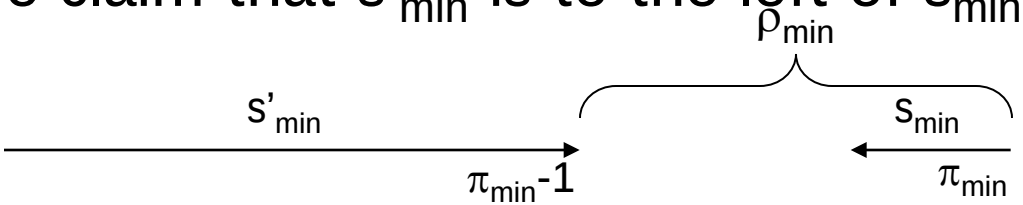
2-approximation algorithm

- If π has a decreasing strip,
  - Let s_{min} be the decreasing strip in π with the minimal element π_{min}. Let s'_{min} be the strip containing π_{min}-1, which is increasing. Let ρ_{min} be the reversal which arrange π_{min} and π_{min}-1 side by side.
  - Let s_{max} be the decreasing strip in π with the maximal element π_{max}. Let s'_{max} be the strip containing π_{max}+1, which is increasing. Let ρ_{max} be the reversal which arrange π_{max} and π_{max}+1 side by side.

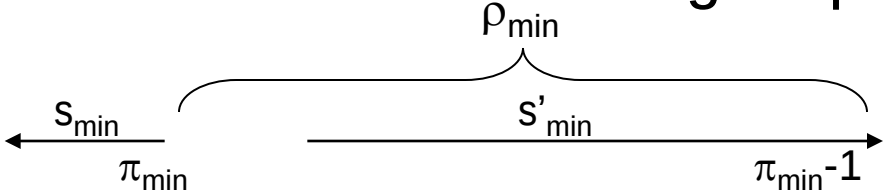
- Lemma: Consider a permutation π that has a decreasing strip. Suppose both $\pi \cdot \rho_{min}$ and $\pi \cdot \rho_{max}$ contain no decreasing strip. Then, the reversal $\rho_{min} = \rho_{max}$ removes 2 breakpoints.

2-approximation algorithm

- Proof: Assume both $\pi \cdot \rho_{min}$ and $\pi \cdot \rho_{max}$ contain no decreasing strip.
  - We claim that s'_{min} is to the left of s_{min}.



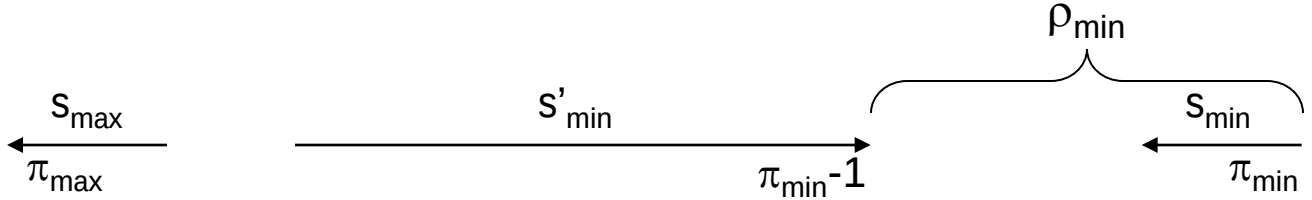
  - Otherwise, the reversal ρ_{min} removes a breakpoint and still maintains a decreasing strip.



  - Similarly, we can show that s_{max} is to the left of s'_{max}.

2-approximation algorithm

- We claim that s_{max} is in between s'_{min} and s_{min}.
- Otheriwse, if s_{max} is to the left (or right) of both s_{min} and s'_{min}, then after the reversal of ρ_{min}, we still have the decreasing strip s_{max}.



- Similarly, we can show that s_{min} is in between s_{max} and s'_{max}.

- Hence, the only possible arrangement such that there is no decreasing strip after performing either ρ_{min} or ρ_{max} is as follows.

2-approximation algorithm

- We claim that there is no element between s'_{min} and s_{max}.
- Between s'_{min} and s_{max},
  - If there is a decreasing strip, we apply the reversal of ρ_{max} and this decreasing strip retain.
  - If there is an increasing strip, we apply the reversal of ρ_{min} and this strip become decreasing.
- Similarly, we can show that there is no element between s_{min} and s'_{max}.

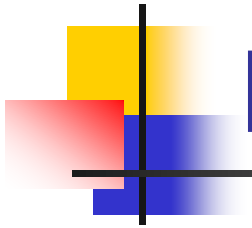
- Therefore, the reversal $\rho_{max}=\rho_{min}$ reverses the interval between π_{max} and π_{min} and removes two breakpoints.

2-approximation algorithm

- Algorithm
  - if there exist no decreasing strip in π,
    - we reverse any increasing strip to create a decreasing strip.
  - while $b(\pi) > 0$,
    - if $\pi \cdot \rho_{min}$ contains decreasing strip,
      - we reverse π by ρ_{min} [this reversal reduces $b(\pi)$ by at least 1];
    - else if $\pi \cdot \rho_{max}$ contains decreasing strip,
      - We reverse π by ρ_{max} [this reversal reduces $b(\pi)$ by at least 1];
    - else
      - We reverse π by $\rho_{max} = \rho_{min}$ [this reversal reduces $b(\pi)$ by 2];
      - We reverse any increasing strip to create a decreasing strip [$b(\pi)$ does not change]

- The above algorithm will reduce the number of breakpoints by 2 for every 2 reversals.
- Hence, it will perform $b(\pi)$ reversals.
- The optimal solution performs at least $b(\pi)/2$ reversals.
- Thus, the above algorithm has approximation ratio 2.

Example



- (11, 12, 1, 2, 3, 4, 5, 7, 6, 8, 9, 10); 5 breakpoints
- (11, 12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10); 3 breakpoints
- (11, 12, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1); 3 breakpoints
- (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 11); 2 breakpoints
- (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12); 0 breakpoint

Sorting signed permutation by reversals

- Genes have orientation. If we know the orientations, then we have the problem of sorting signed permutation.

- Given a signed permutation of {0, 1, 2, …, n}, that is, $\pi = (\pi_0, \pi_1, \pi_2, \ldots, \pi_n)$.
- We set $\pi_0=0$ and $\pi_n=n$ to denote the boundary of the genome.
- A reversal $\rho(i,j)$ is an operation applying on π, denoted as $\pi \cdot \rho(i,j)$, which reverses the order and flip the signs of the element in the interval [i..j].
- Thus, $\pi \cdot \rho(i,j) = (\pi_0, \pi_1, \ldots, \pi_{i-1}, -\pi_j, \ldots, -\pi_i, \pi_{j+1}, \ldots, \pi_n)$.
- Our aim is to find the minimum number of reversals that transform π to (0, 1, 2, …, n).
- The minimum number of reversals need to transform π to (0, 1, 2, …, n) is called the reversal distance, denoted by $d(\pi)$.

Example: sorting signed permutation

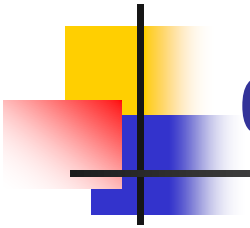
- +0,+3,+1,+6,+5,-2,+4,+7
- +0,-5,-6,-1,-3,-2,+4,+7
- +0,-5,-6,-1,+2,+3,+4,+7
- +0,-5,-6,+1,+2,+3,+4,+7
- +0,-5,-4,-3,-2,-1,+6,+7
- +0,+1,+2,+3,+4,+5,+6,+7

Previous works on sorting signed permutation

- Sankoff (1992): Introduce the problem
- Hannenhalli and Pevzner (1995): First polynomial time algorithm for sorting a signed permutation $O(n^4)$ time.
- Berman and Hannenhalli (1996): Improved to $O(n^2\alpha(n))$ time where α is the inverse Ackerman's function.
- Kaplan, Shamir, and Tarjan (1999): $O(n^2)$ time.
- Bergeron (2001): A simplifed method $O(n^3)$ time and $O(n^2)$ time on a vector-machine
- Tannier, Bergeron, and Sagot (2007): $O(n^{3/2}\text{sqrt}(\log n))$ time.

- Computing reversal distance only:
  Bader, Moret, and Yan (2001): O(n) time
  Bergeron, Mixtacki, and Stoye (2004): O(n) time

Upper bound on signed reversal distance



- A simple way to transform π to (0, 1, 2, …, n):
  - Disregarding the sign, we can create a correct sequence by n reversals
  - We can correct the sign by at most n sign flips (reversals of length 1).
- Then, the simple upper bound for the reversal distance is 2n.

- Can we get a better upper bound?

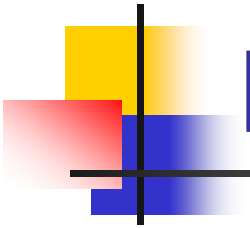
Pancake problem

- A waiter has a stack of *n* pancakes. To avoid disaster, the waiter wants to sort the pancakes in order by size. Having only one free hand, the only available operation is to lift a top portion of the stack, invert it, and replace it.
- The Pancake Problem (Goodman 1975) finds the maximum number of flips needed.

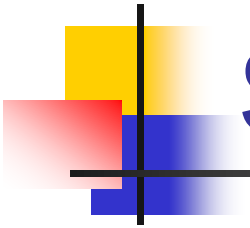
- Gate and Papadimitriou (1979) showed that the number of flips is at most (5n+5)/3.

- This problem is equivalent to sorting an unsigned permutation by prefix reversals.
- Hence, the reversal distance for sorting unsigned permutation is at most (5n+5)/3.

Burnt Pancake problem



- Gates and Papadimitriou (1979) introduced the *Burnt Pancake Problem*. Here one side of each pancake is burnt, and the pancakes must be sorted with the burnt side down.
- Heydari and Sudborough (1997) showed that the number of flips is at most 3(n+1)/2.
- This problem is equivalent to sorting a signed permutation by prefix reversals.
- Hence, the reversal distance for sorting signed permutation is at most 3(n+1)/2.



Sorting signed permutation

- Below, we discuss an $O(n^3)$ time solution for sorting signed permutation.

- First, we need to understand three concepts:
  - Interval
  - Cycle
  - Component

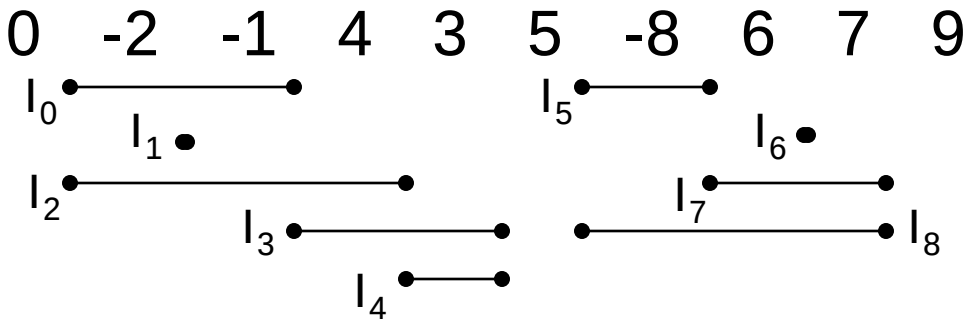
Points and breakpoints

- Consider a signed permutation $\pi=(\pi_0, \ldots, \pi_n)$ where $\pi_0=0$ and $\pi_n=n$.
- Let v_i be a point between π_i and π_{i+1} for each $0 \leq i \leq n$.
- A point v_i is a non-breakpoint if (π_i, π_{i+1}) equals either $(k,k+1)$ or $(-(k+1),-k)$ for some k.
- For example, there are two non-breakpoints in the following example.

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

Elementary interval

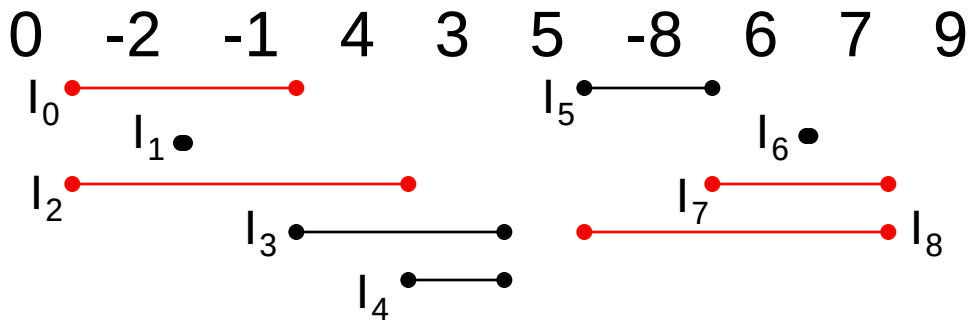
- For any (π_i, π_j) such that $\{|\pi_i|,|\pi_i|\}=\{k,k+1\}$, we define the elementary interval I_k be the interval whose endpoints are:
  - The right point of k if the sign of k is positive; otherwise its left point.
  - The left point of k+1 if the sign of k+1 is positive; otherwise its right point.



Oriented interval

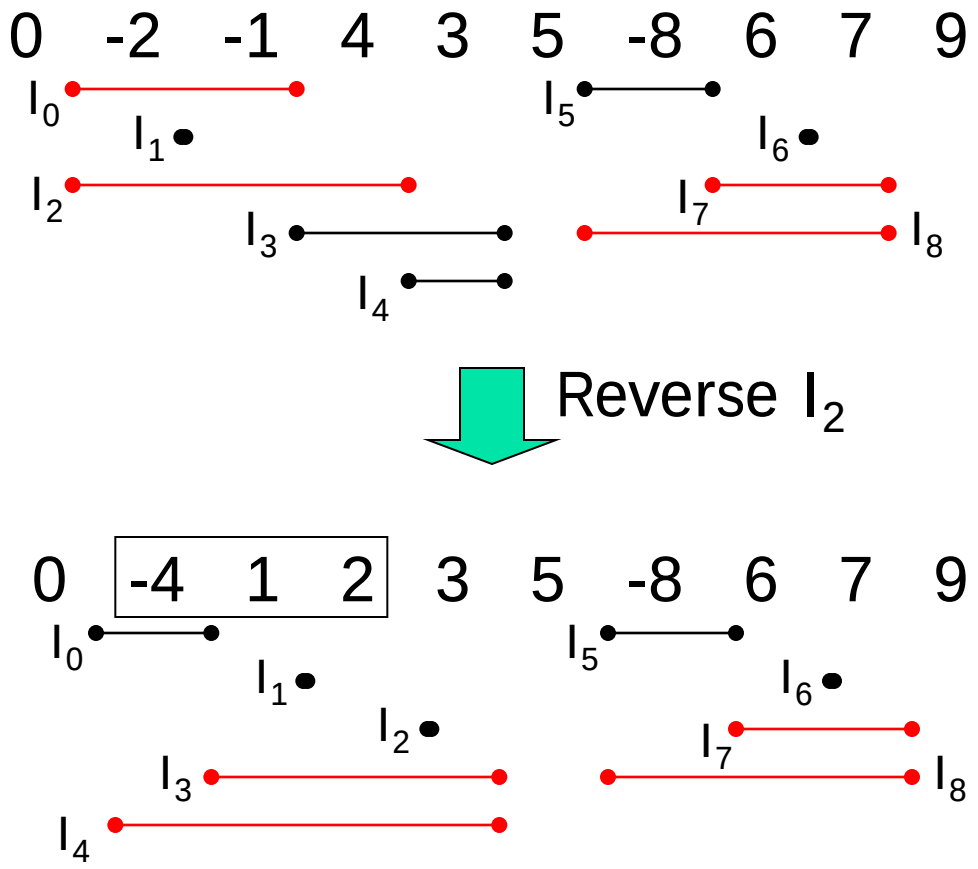
- An elementary interval I_k is oriented if the signs of k and k+1 are different; otherwise, it is unoriented.

Red color intervals are oriented intervals



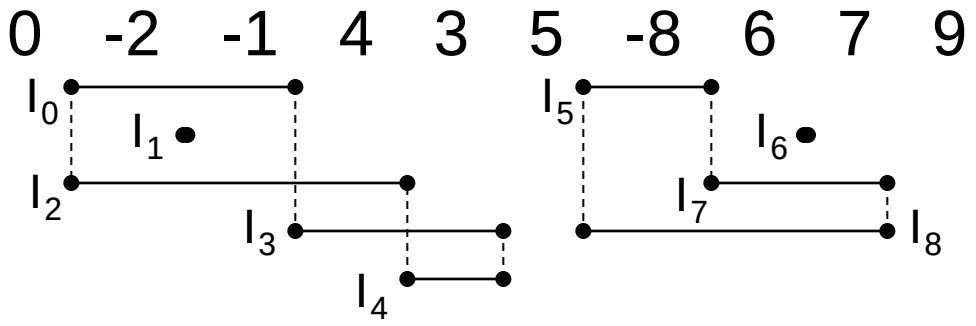
Property of oriented interval

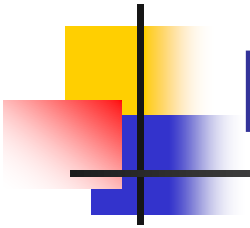
- Property: Reversing an oriented interval reduces the number of breakpoints.



Cycle

- Note that every breakpoint meet exactly two endpoints of some elementary intervals.
- Hence, the elementary intervals form disjoint cycles.
- Example: There are 4 cycles containing 1, 1, 3 and 4 elementary intervals.
- I_1 and I_6 are isolated and we call them isolated intervals.



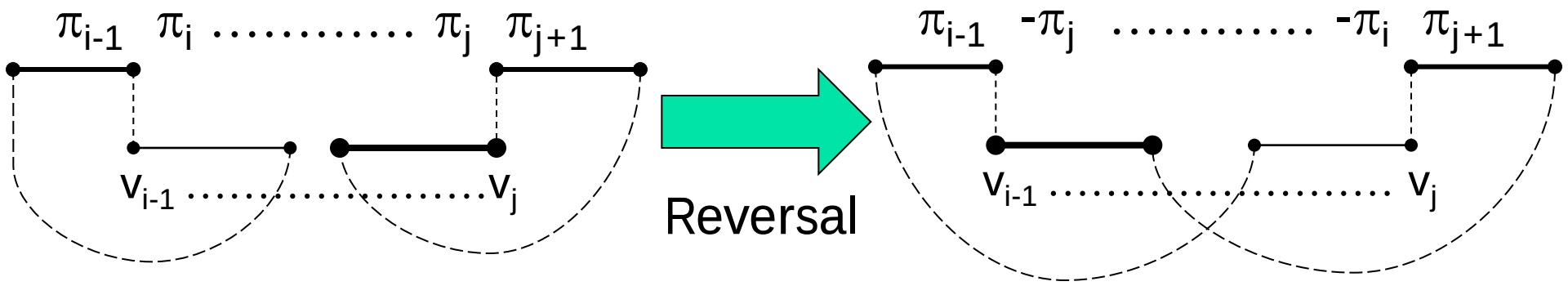


Property of Cycle (I)

- Property: Reversing any elementary intervals modifies the number of cycles by +1, 0, or -1.
- Proof:
  - Suppose we reverse $(\pi_i, \ldots, \pi_j)$.
  - Let v be the breakpoint between π_j and π_{j+1} and v' be the breakpoint between π_{i-1} and π_i.
  - The reversal will only affect the cycles passing through v and v'. There are two cases.

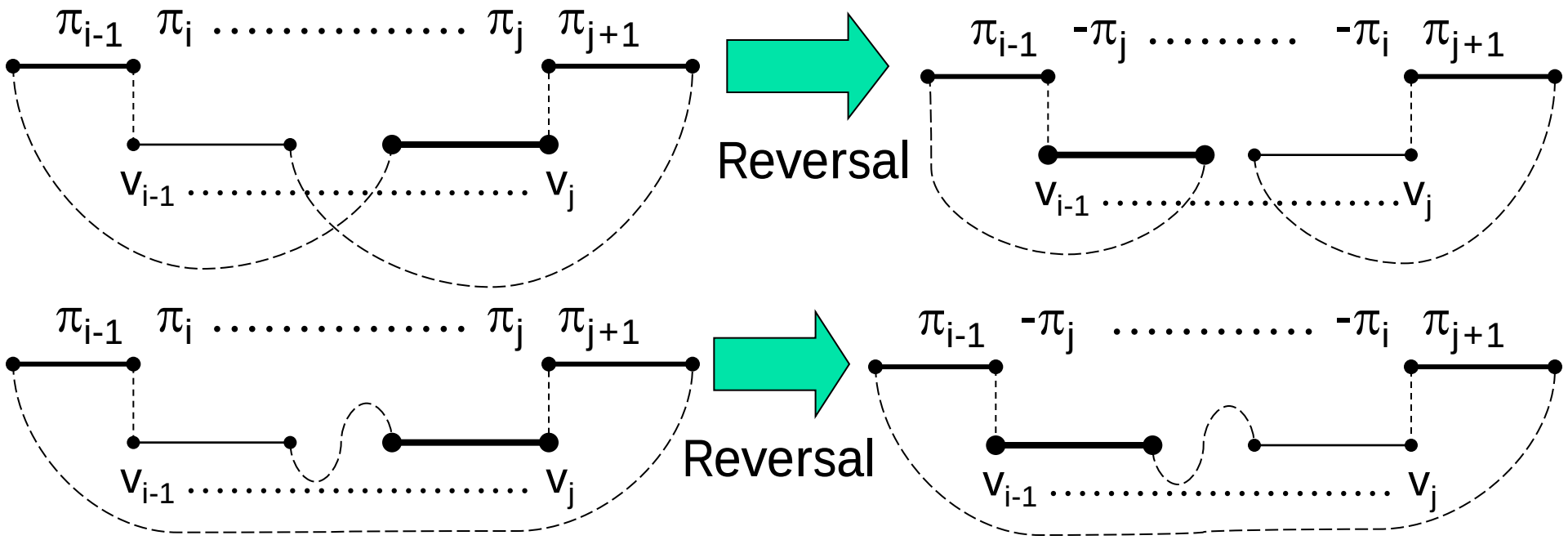
Property of Cycle (II)

- Case 1: Two distinct cycles passing through v and v'. In this case, we will merge the two cycles. Hence, the number of cycle is reduced by 1.



Property of Cycle (III)

- Case 2: One cycle passing through v and v'. In this case, we will either maintain one cycle or break the cycle into two. Hence, the number of cycle is either no change or increase by 1.

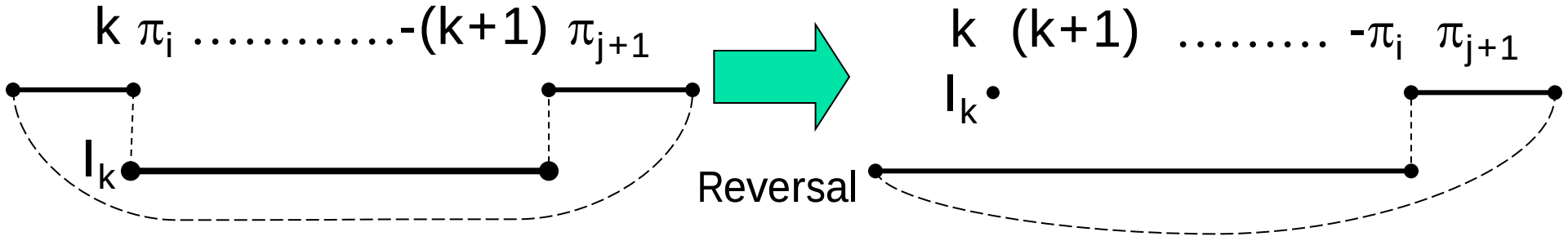


Property of Cycle (IV)

- Suppose π has c cycles.
- Note that the identity permutation (0, 1, 2, …, n) is the only permutation which has n cycles.
- By the previous property, we have the following lemma:
- Lemma: $d(\pi) \geq n-c$.

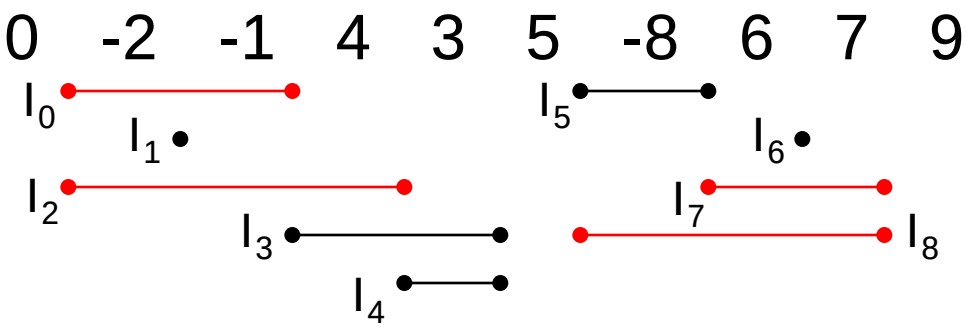
Property of Cycle (V)

- Lemma: Reversing an oriented interval increases the number of cycle by one. The new cycle is an isolated interval.
- Proof:
  - See the following example.



Component

- A component is an interval in π which
  - either starts from i and ends at j OR starts from -j and ends at -i for some i<j.
  - contains all numbers between i and j.
  - It is not the union of two or more such intervals.
- Below example has 4 components:
  - (0..5)
  - (5..9)
  - (-2..-1)
  - (6..7)



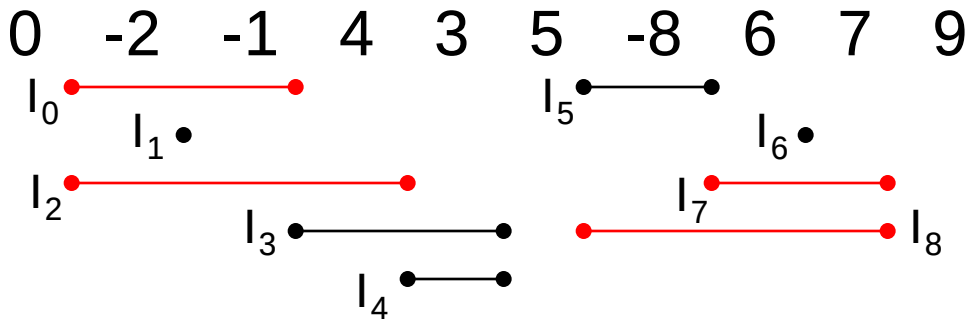
Component (II)

- Example 2:
  π=(0, -3, 1, 2, 4, 6, 5, 7, -15, -13, -14, -12, -10, -11, -9, 8, 16)
  - There are 6 components:
    - (0..4),
    - (4..7),
    - (7..16)
    - (1..2),
    - (-15..-12),
    - (-12..-9)

Oriented component

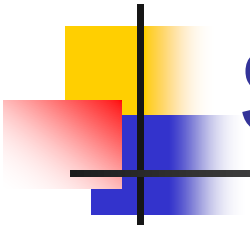
- A component is unoriented if it has breakpoint but does not have any oriented interval.

- Example:
  - (0..5): oriented
  - (5..9): oriented
  - (-2..-1) oriented
  - (6..7): oriented



Oriented component (II)

- Example:
  π=(0, -3, 1, 2, 4, 6, 5, 7, -15, -13, -14, -12, -10, -11, -9, 8, 16)
  - There are 6 components
    - (0..4) --- oriented
    - (4..7) --- unoriented
    - (7..16) --- oriented
    - (1..2) --- oriented
    - (-15..-12) --- unoriented
    - (-12..-9) --- unoriented

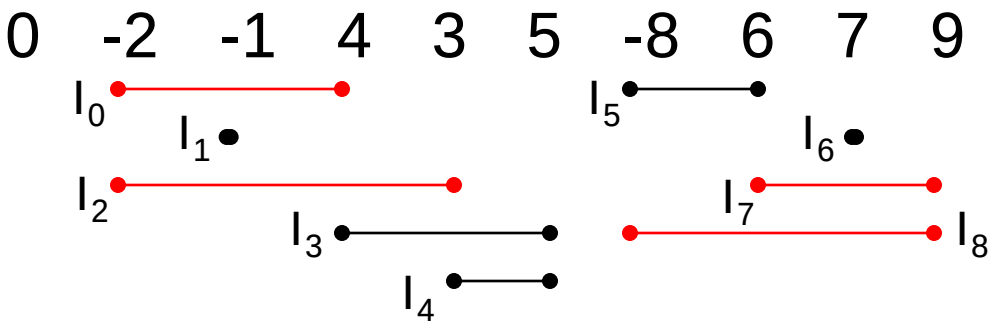


Sorting signed permutation

- When all components are oriented,
  - Bergeron's basic algorithm
- Otherwise,
  - The Hannenhalli-Pevzner Theorem

Bergeron's basic algorithm

- Define the score of a permutation π be
  - the number of oriented intervals in the permutation π.



score(π)=4

Bergeron's basic algorithm

- Input: a signed permutation with no unoriented component.

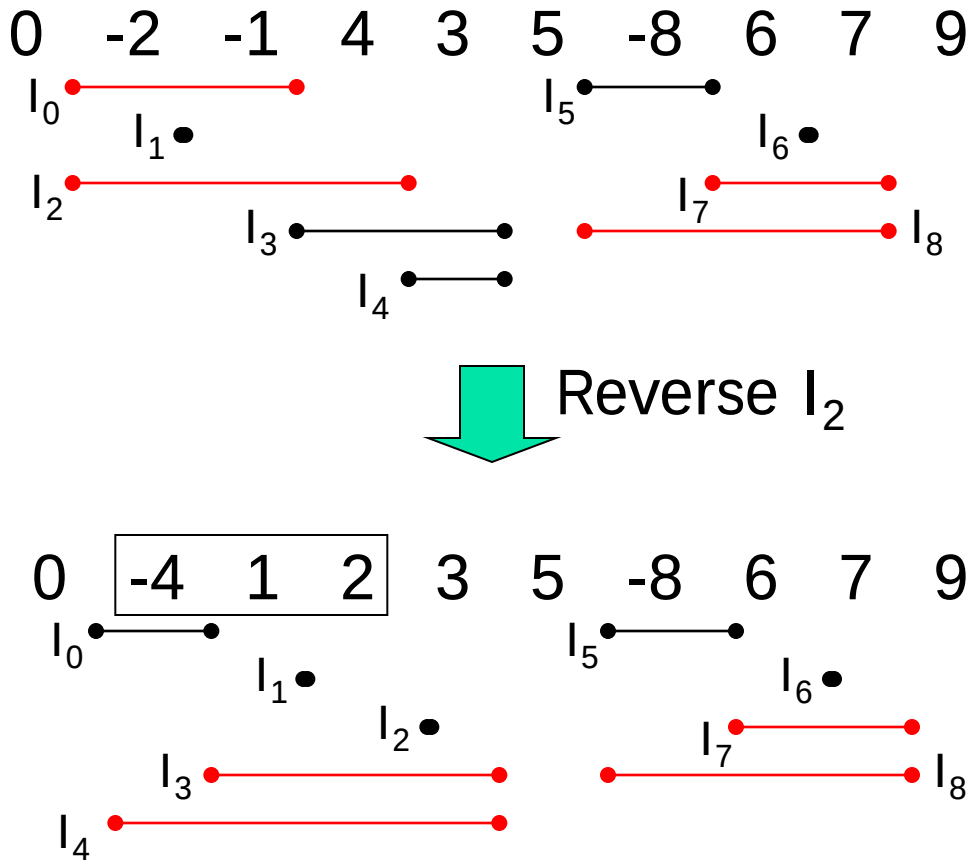
- Algorithm Bergeron_basic
  - while π has an oriented interval
    - Choose the oriented interval I that has maximum score($\pi \cdot I$)
    - Report I and set $\pi = \pi \cdot I$

Example

- π_1=(0,+3,+1,+6,+5,-2,+4,+7)
  - score($\pi_1 \cdot I_1$)=2, score($\pi_1 \cdot I_2$)=4
- π_2=(0,-5,-6,-1,-3,-2,+4,+7)
  - score($\pi_2 \cdot I_0$)=2, score($\pi_2 \cdot I_3$)=4, score($\pi_2 \cdot I_4$)=2, score($\pi_2 \cdot I_6$)=2
- π_3=(0,-5,-6,-1,+2,+3,+4,+7)
  - score($\pi_3 \cdot I_0$)=0, score($\pi_3 \cdot I_1$)=2, score($\pi_3 \cdot I_4$)=2, score($\pi_3 \cdot I_6$)=2
- π_4=(0,-5,-6,+1,+2,+3,+4,+7)
  - score($\pi_3 \cdot I_4$)=2, score($\pi_4 \cdot I_6$)=2
- π_5=(0,-5,-4,-3,-2,-1,+6,+7)
  - score($\pi_5 \cdot I_0$)=0, score($\pi_5 \cdot I_5$)=0
- π_6=(0,+1,+2,+3,+4,+5,+6,+7)

Property of intersect

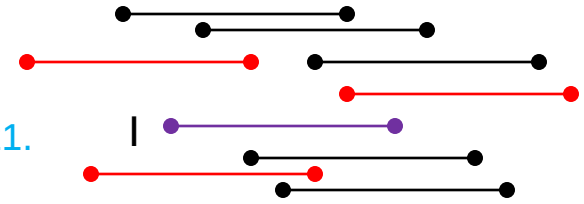
- For any intervals I_k, we says an interval $I_{k'}$ intersects with I_k if either k' or k'+1 (but not both) is within I_k.
- Property: Once we perform a reversal on an oriented interval I_k,
  - any elementary interval $I_{k'}$, where intersects with I_k, will changes its orientation.



Correctness

- Theorem: Reversing the oriented interval I of **maximal** score does not create new unoriented components.
- Proof:
  - Suppose the reversal of I introduces a new unoriented component C.
  - Note that the reversal of I only affects elementary intervals which intersects with I.
  - Let I' be an elementary interval which intersects with I and belongs to C.
  - Let T be the total number of oriented intervals before reversal.
  - Let U and O be the number of unoriented and oriented intervals, respectively, in π which intersects with I.
  - We have Score($\pi \cdot$I)=T+U-O-1.

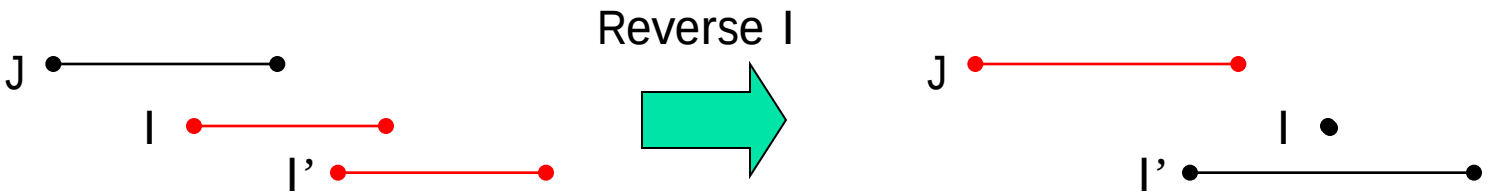
In this example, U=5, O=3.
Suppose T=20.
Then, Score($\pi \cdot$I)=20+5-3-1=21.



  - Similarly, let U' and O' be the number of unoriented and oriented intervals, respectively, in π which intersects with I'.
  - Score($\pi \cdot$I')=T+U'-O'-1.

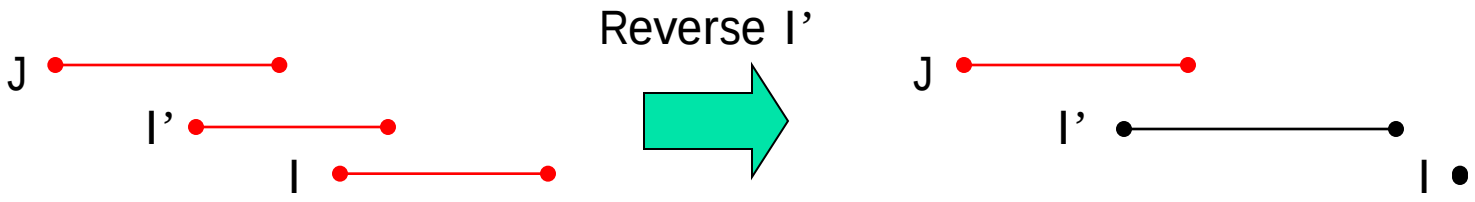
Correctness (II)

- We claim that any unoriented interval, that intersects with I, also intersects with I'.
  - Otherwise, let J be an unoriented interval that intersects with I but not I'. After reversing I, J becomes oriented and intersects with I'. This contradicts with the assumption that C is unoriented.
- Thus, U'≥U.



Correctness (III)

- We also claim that any oriented interval, that intersects with I', also intersects with I.
  - Otherwise, let J be an oriented interval that intersects with I' but not I. After reversing I, J remains oriented and intersects with I'. This also contradicts with the assumption that C is unoriented.
- Hence, O≥O'.



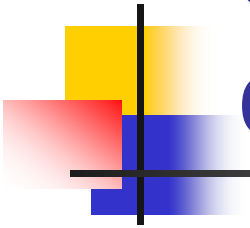
Correctness (IV)

- If U=U’ and O=O’,
  - I and I’ correspond to the same interval.
  - After reversing I, both I and I’ becomes isolated intervals. This contradicts that C is unoriented.
- This means that
  - $\text{Score}(\pi \cdot I) = T+U-O-1 < T+U'-O'-1 = \text{Score}(\pi \cdot I')$.
- This contradicts with the fact that I maximizes $\text{Score}(\pi \cdot I)$.

Summary for sorting oriented components

- Corollary: If π has c cycles and has no unoriented component, $d(\pi)=n-c$.
- Proof:
  - Recall that $d(\pi)\geq n-c$.
  - Any oriented reversal will increase the number of cycle by 1.
  - Previous theorem ensures that we always have oriented reversal.
  - Hence, after n-c oriented reversal, we get n cycles, which is an identify permutation.
  - Thus, $d(\pi)\leq n-c$.

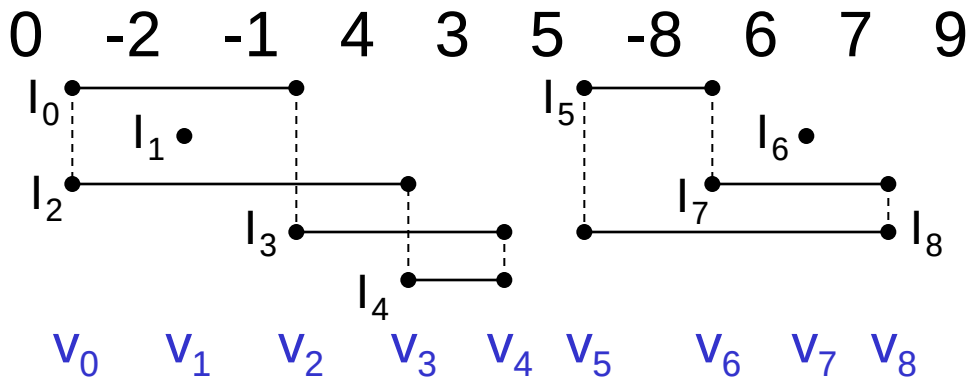
Sorting when there is unoriented component

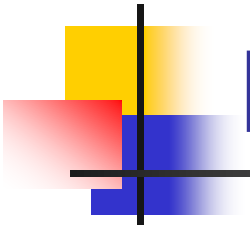


- When unoriented component exists,
  - The idea is to perform reversals to remove all the unoriented component.
  - Then, we apply the Bergeron's basic algorithm
- Below, we first give some properties of component.

More on Component (I)

- Any point v_i between π_i and π_{i+1} belongs to the smallest component which contains both π_i and π_{i+1}.
- Example:
  - (0..5) contains v_0, v_2, v_3, v_4
  - (5..9) contains v_5, v_6, v_8
  - (-2..-1) contains v_1
  - (6..7) contains v_7





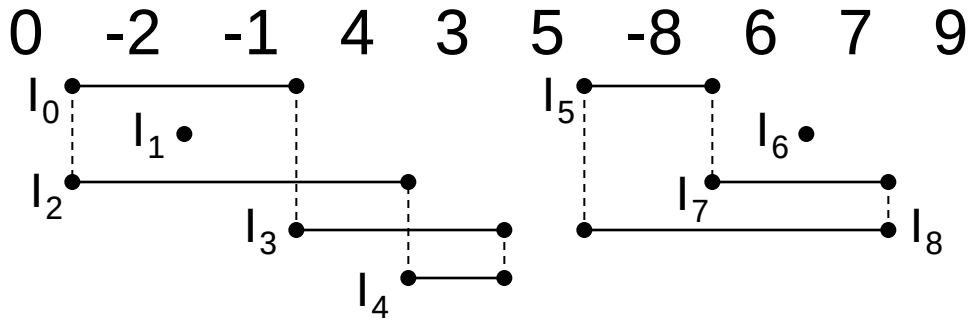
More on Component (II)

- Property: The endpoints of any elementary interval belong to the same component.

- Corollary: For any cycle, its endpoints belong to the same component.

More on Component (III)

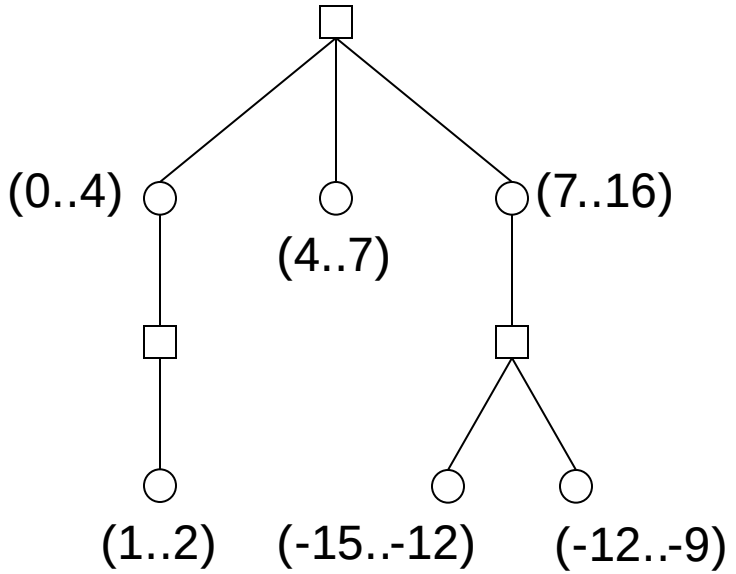
- Lemma: Two different components of a permutation are either disjoint, nested with different endpoints, or overlapping on one element.
- Example:
  - (-2..-1) and (5..9) are disjoint
  - (0..5) and (5..9) overlap on one element
  - (6..7) is nested within (5..9)



Chain and component

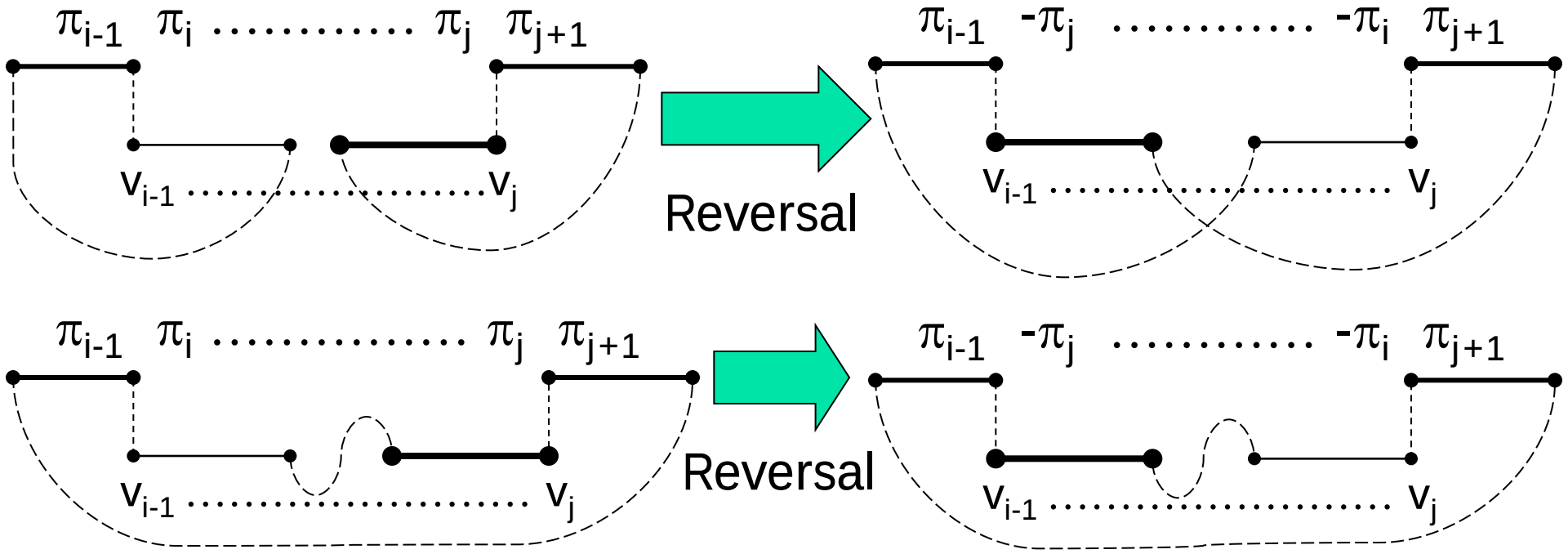
π=(0, -3, 1, 2, 4, 6, 5, 7, -15, -13, -14, -12, -10, -11, -9, 8, 16)

- When two components overlap on one element, they are said to be linked.
- Successive linked components form a chain.
- A maximal chain is a chain that cannot be extended. (It may consist of a single component.)
- The relationship among components can be represented as a tree T_π as follows.
  - Each component represents a round node
  - Each maximal chain represents a square node whose components are ordered children of it.
  - A maximal chain is a child of the smallest component that contains this maximal chain.



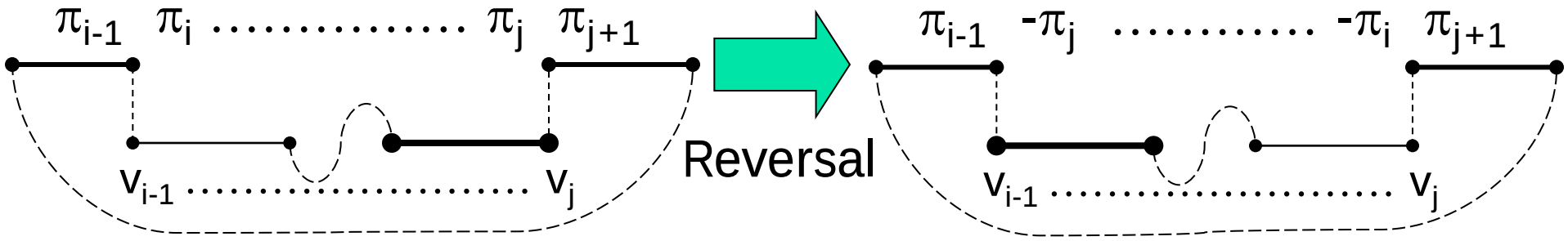
Effect of reversal on components (I)

- Lemma A: Consider an unoriented component C. The reversal of any interval in C will not increase the number of cycles. Moreover, C will become oriented.



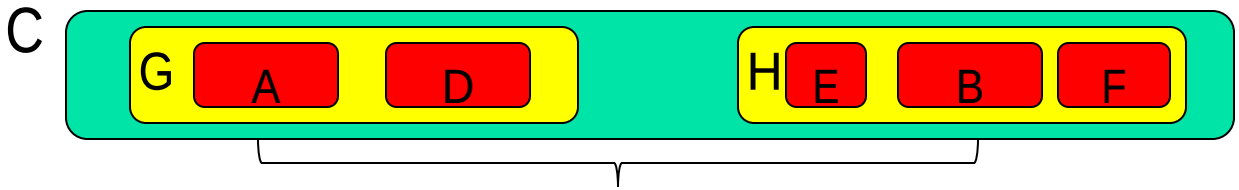
Effect of reversal on components (I)

- Lemma B: Consider an unoriented component C. The reversal of any elementary interval in C will not change the number of cycles. Moreover, C will become oriented.
- This reversal operation is denoted as the cut operation.

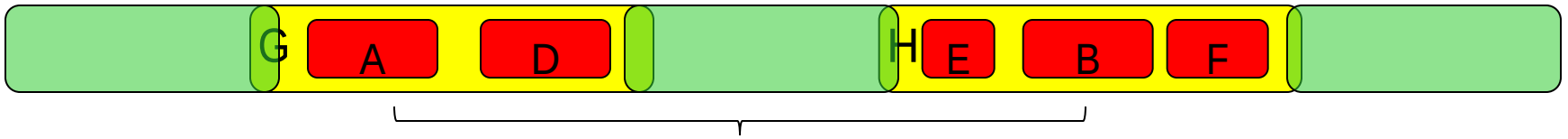


Effect of reversal on components (II)

- Lemma C: If a reversal has its two endpoints in different components A and B, then only the components on the path from A to B in T_π are affected.
  - Any component C contains either A or B but not both will be destroyed.
  - If the lowest common ancestor of A and B in T_π is a component C, if A or B is unoriented, then C become oriented after the reversal.
  - If the lowest common ancestor of A and B in T_π is a chain, a new component C is created. If either A or B is unoriented, C will be oriented.
- The reversal operation is denoted as merge operation.



After this reversal,
A, G, B, and H are destroyed.
If A or B is unoriented, C become oriented.



After this reversal,
A, G, B, and H are destroyed.
A new component C is formed.
If A or B is unoriented, C become oriented.

Cover

- A cover C of T_π is a collection of paths joining all the unoriented components of π such that no two paths end at the same node.
- A path that ends at two unoriented components is called long path.
- A path that contain only one unoriented component is called short path.

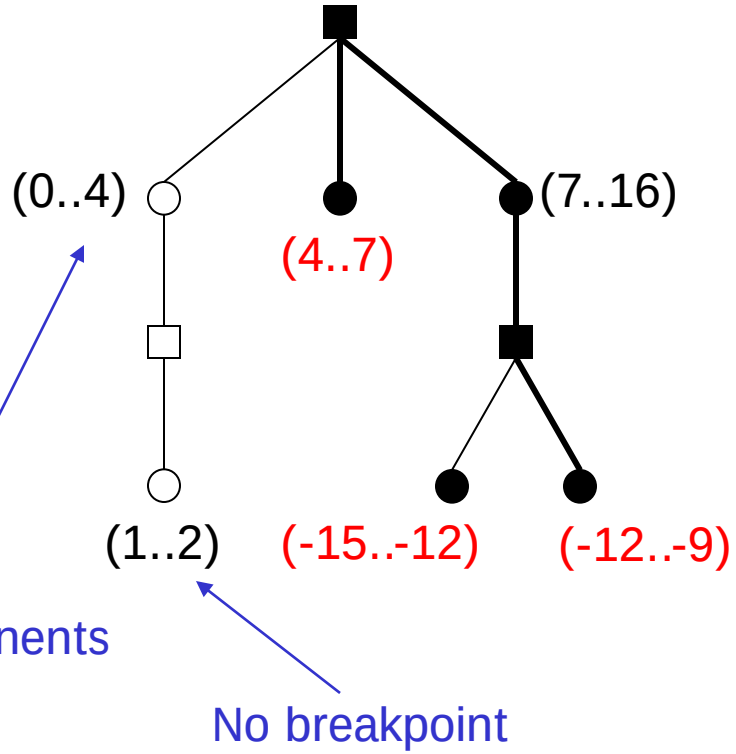
- We can generate a permutation with no unoriented component as follows:
  - For each long path, we apply merge operation on the two unoriented components at the ends of the long path.
  - For each short path, we apply cut operation on the unoriented component.

Cover (II)

π=(0, -3, 1, 2, 4, 6, 5, 7, -15, -13, -14, -12, -10, -11, -9, 8, 16)

- The cost of a long path is 2.
- The cost of a short path is 1.
- The cost of a cover is the sum of the costs of its paths.
- An optimal cover is a cover of minimal cost.

- Example: the optimal cover is
  - (4..7) to (-12..-9)
  - (-15..-12)



The Hannenhalli-Pevzner Theorem

- Theorem: Given a permutation π of {0, 1, …, n} with c cycles and the associated tree T_π has minimal cost t,
  - $d(\pi) = n - c + t$.
- Proof:
  - We claim that $d(\pi) \leq n - c + t$.
  - We apply m merges to the m long paths and q cuts to the q short paths.
  - Note that $t = 2m + q$.
  - After applying m merges and q cuts, the resulting permutation π' has c-m cycles and has no unoriented component.
  - Hence, $d(\pi') = n-(c-m)$.
  - $d(\pi) \leq d(\pi') + m + q = n - c + 2m + q = n - c + t$

The Hannenhalli-Pevzner Theorem

- We also claim that $d(\pi) \geq n - c + t$.
- Let d be the optimal reversal distance.
- d = s + m + q where
  - s is the number of reversals split cycle
  - m is the number of reversals merge cycle
  - q is the number of reversals which do not change the number of cycle
- Since identity permutation has n cycles, we have c+s-m = n.
- Thus, d = n - c + 2m + q.
- Any reversal merges a group of components on a path in T_{π}. We keep the shortest segment that includes all unoriented components of the group.
- Those paths should cover all unoriented components. Otherwise, we cannot transform π to identity permutation.
- Hence, $t \leq 2m+q$. Thus, $d \geq n - c + t$.

General algorithm for sorting by signed reversal

Algorithm Sort_Signed_Reversal

- Construct T_{π}
- Find the optimal cover C of T_{π}
- For each long path in the cover C, identify the leftmost and the rightmost unoriented components and merge them.
- For each short path in the cover C, cut the unoriented component on the short path.
- Run Bergeron_basic