SORTING BY REVERSALS

based on chapter 7 of Setubal, Meidanis:
Introduction to Computational molecular biology

Motivation

When comparing genomes across species insertions, deletions and substitutions of bases (point mutations) are often *not* as interesting as changes on a larger scale.

Genome rearrangements is when longer pieces of chromosone are moved or copied to other locations in the same chromosone or even to other chromosones.

The *reversal* seems to be the most important type of genome rearrangement.

The oriented case

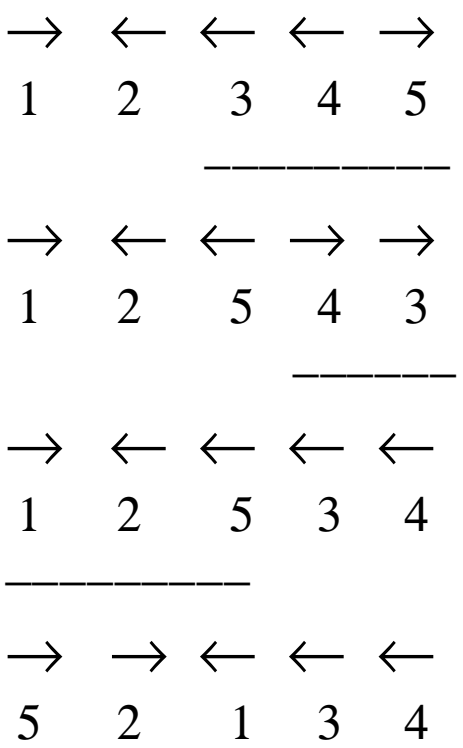
Given: Two permutaions of n *oriented* genes
taken from chromosones of two related organisms

Find: Minimum number of reversals needed to transfer one permutation to the other.
(Minimum because of the *parsimony assumption*)

Solvable in polynomial time

Example

Permutation 1



Permutation 2

Distance ≤ 3 reversals

The unoriented case

Given: Two permutaions of n genes taken from chromosones of two related organisms

Find: Minimum number of reversals needed to transfer one permutation to the other.

NP-hard

Example

1 2 3 4 5

1 2 5 4 3

1 2 5 3 4

5 2 1 3 4

The canonical case

Any instance of sorting by reversal can be transformed to a canonical case where one permutation is the identity permutation

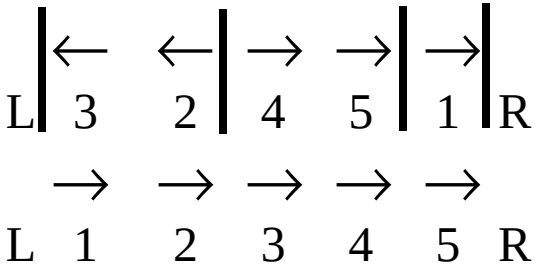
Initial permutation:	$\rightarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\rightarrow$
	a	b	c	d	e
Target permutation:	$\rightarrow$	$\rightarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
	e	b	a	c	d

Initial permutation:	$\leftarrow$	$\leftarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
	3	2	4	5	1
Target permutation:	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
	1	2	3	4	5

Breakpoints

A breakpoint is a point between two consecutive oriented labels that must be separated by at least one reversal.

Each reversal removes at most two breakpoints



At least two reversals are needed (but in fact three are needed in this case)

Breakpoints – Lower bound

$d(\pi)$ = minimum number of traversals needed to bring π to the identity permutation.

$b(\pi)$ = number of breakpoints in π.

$d(\pi) \geq b(\pi) / 2$ (not tight)

A reversal is sorting if it reduces the distance to the identity permutation (by 1).

A reversal can remove two breakpoints without being sorting.

Diagram of Reality and Desire

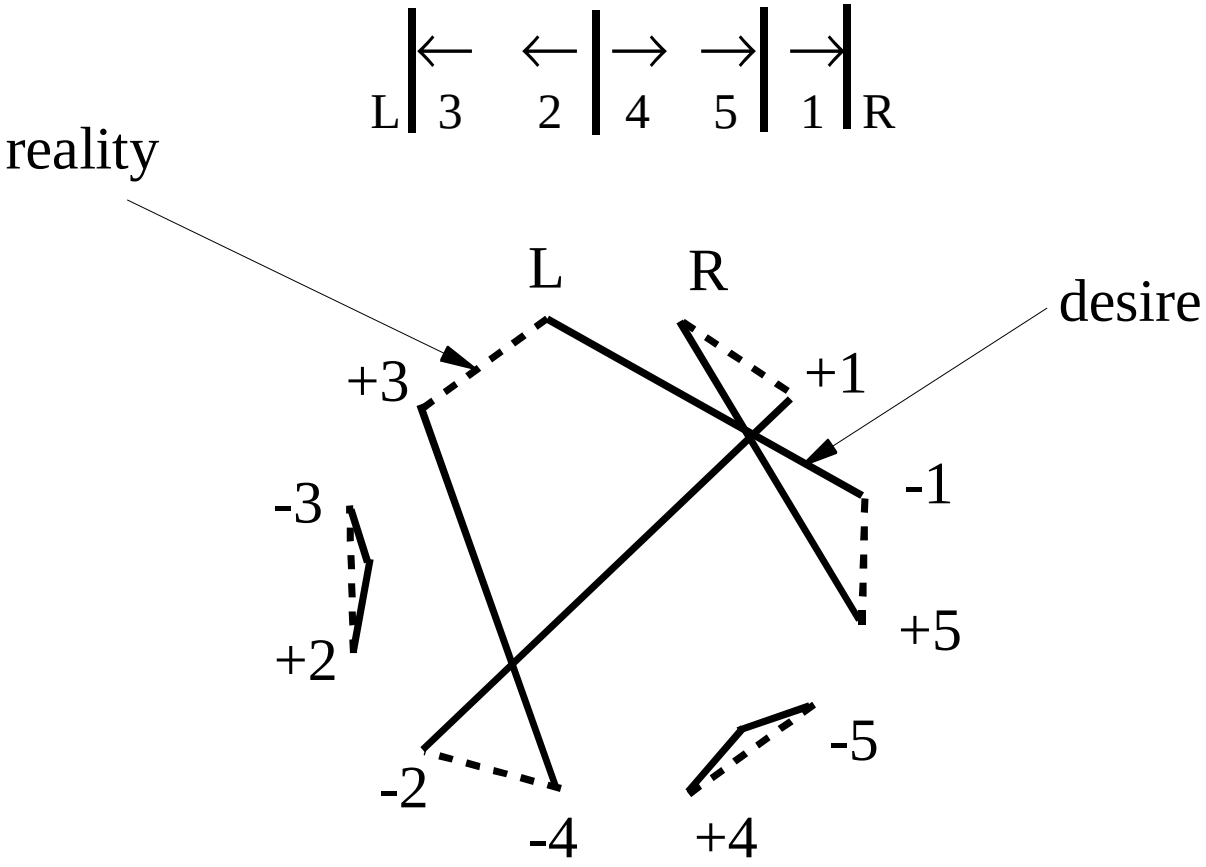
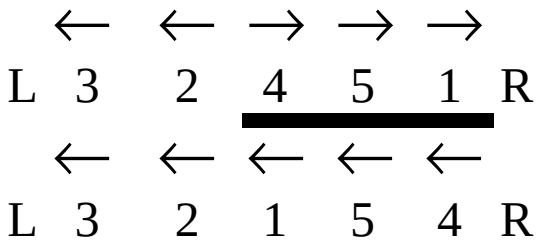
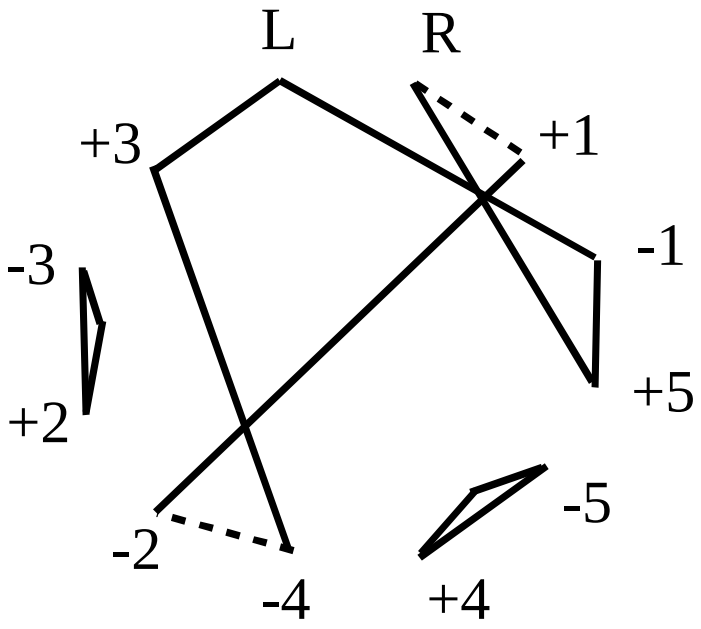
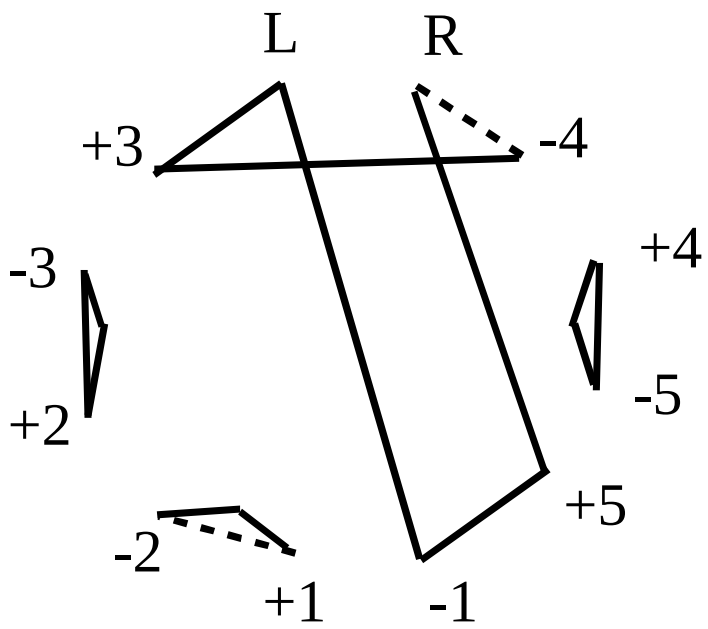


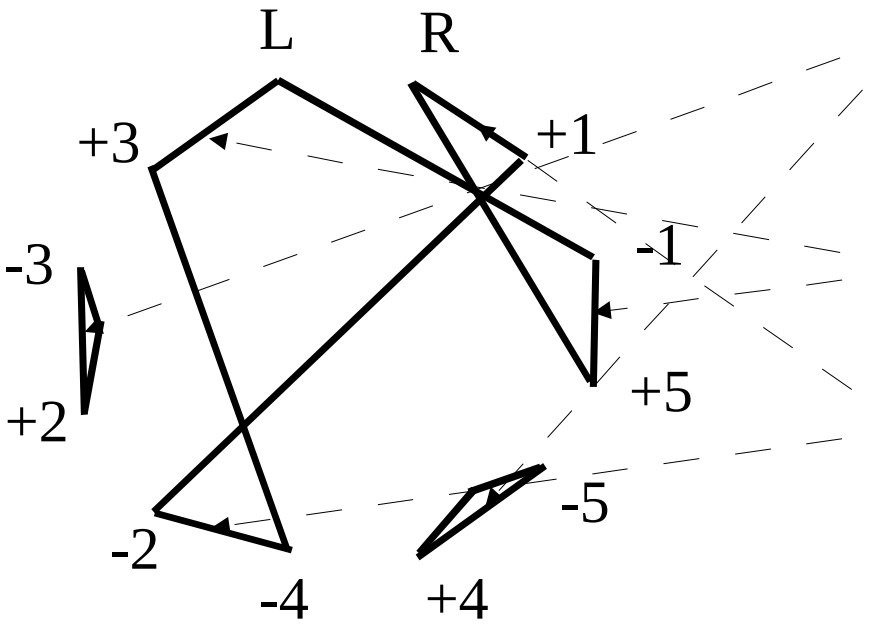
Diagram changes caused by reversal





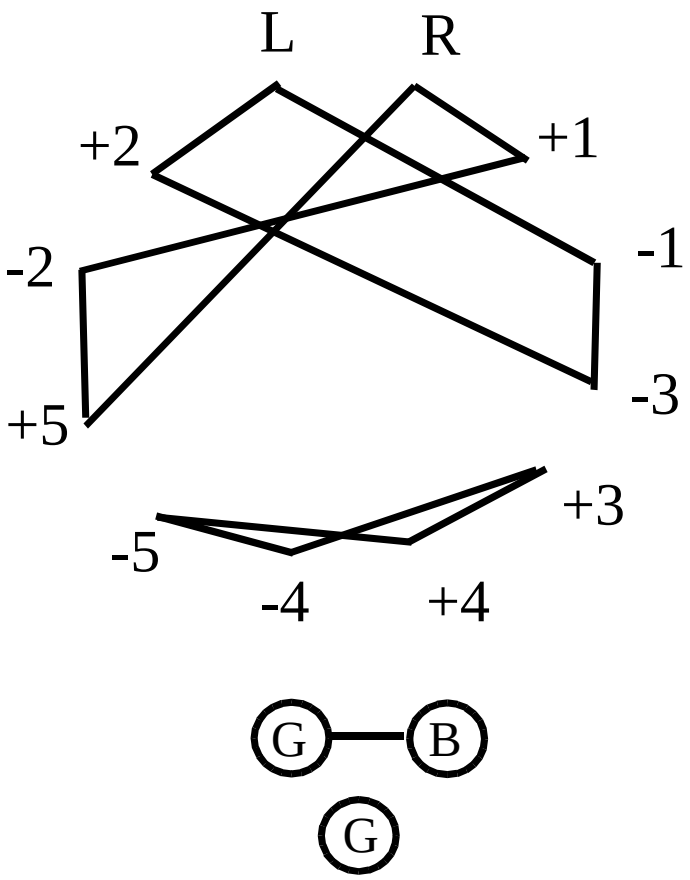


Number of cycles



- $c(\pi)$ = number of cycles
- $c(\pi) = n-1$ if π id identity permutation
- reversal defined by two reality edges from different cycles decreases the number of cycles by one
- converging edges from same cycle does not number of cycles
- diverging edges from same cycle increases number of cycles by one
- $d(\pi) \geq n+1-c(\pi)$

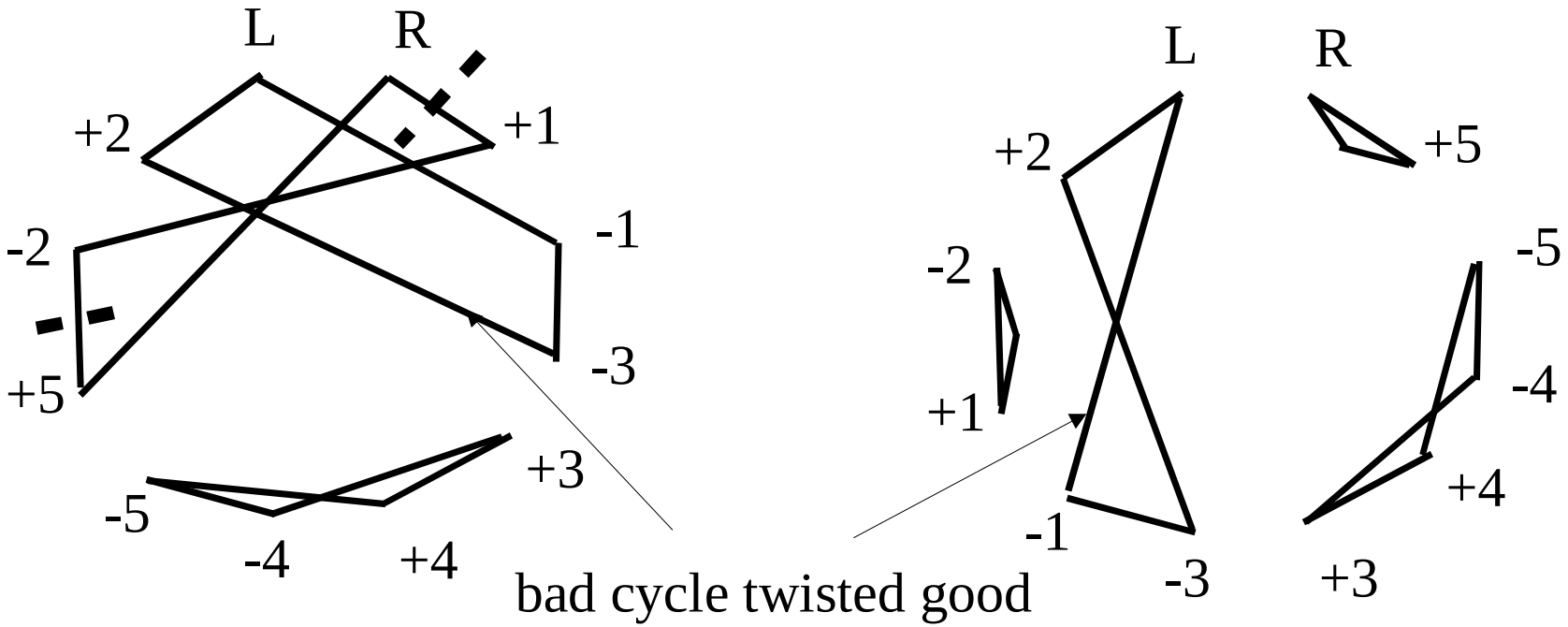
Good and bad cycles and interleaving graph



- A cycle is *good* if it has diverging edges
- Otherwise it's *bad*
- Two cycles *interleave* if any pair of edges cross
- Interleaving graph has cycles as nodes. Two nodes are connected if corresponding cycles interleave, cycles of leangth 2 are excluded.
- A connected component of the interleaving graph is good if it contains at least one good cycle otherwise it's bad

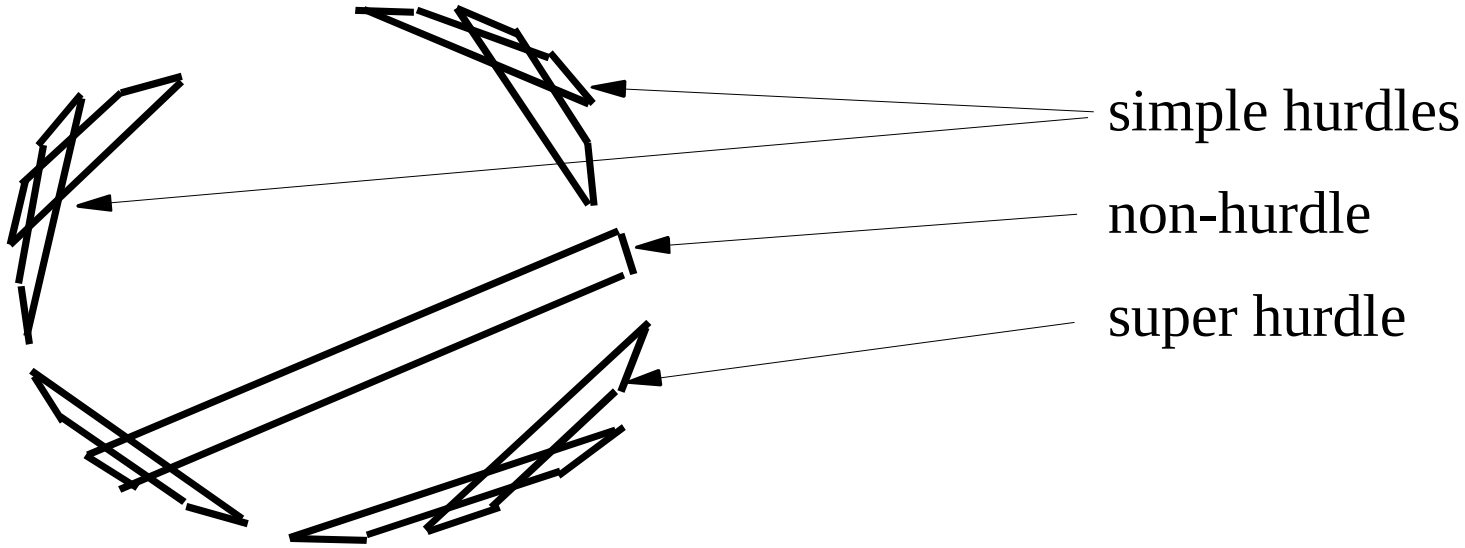
Good components

- A reversal defined by two converging edges of a good cycle is a sorting reversal if and only if its application does not lead to the creation of any bad components
- Such a reversal always exists



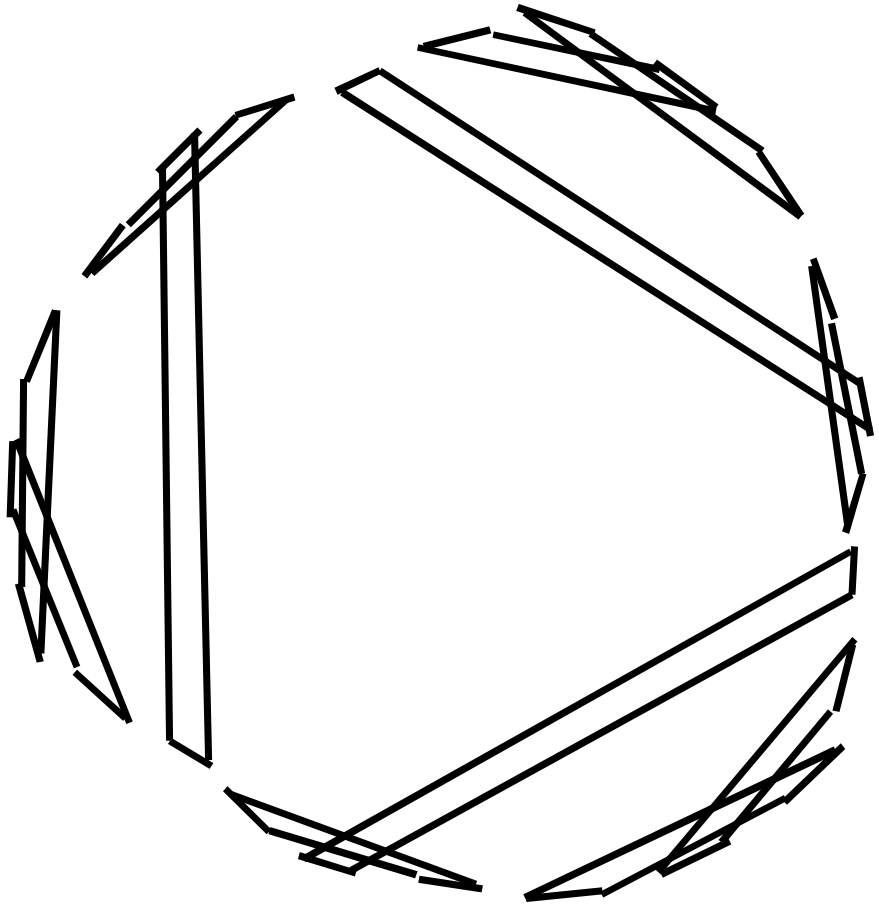
Bad components and hurdles

- A component B *separates* components A and C if every edge between A and C has to cross an edge of B
- A *hurdle* is a bad component that does not separate any other two bad components, the other bad componets are non-hurdles
- A hurdle is a *super-hurdle* if its removal would cause a non-hurdle to become a hurdle
- All other hurdles are called *simple hurdles*



Fortress

•A *fortress* is a permutation which contains an odd number of hurles and all of them are super hurdles



Lower bound on the reversal distance

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

- $c(\pi)$ is number of cycles
- $h(\pi)$ is number of hurdles
- $f(\pi)$ is 1 if π is a fortress and 0 otherwise

Algorithm

Pick a sorting reversal and perform it until target permutation reached.

- If a good cycle exist pick a pair of diverging edges making sure that the corresponding reversal does not create any bad components. (decreases number of cycles by one
- If $h(\pi)$ is odd and there is a simple hurdle, cut this hurdle. (decreases number of hurdles by one without creating a fortress as $h(\pi)$ is odd)
- If $h(\pi)$ is odd and no simple hurdle exists, then π is a fortress, merge any two hurdles (either two less hurdles and one less cycle or fortress gone and one less hurdle (two removed, one created) and one less cycle)
- If $h(\pi)$ is even then merge two opposite hurdles. Choosing opposite hurdles will not create a new hurdle. (two less hurdles and one less cycle)

Unoriented case – strips

- A breakpoint exists between every pair of nonconsecutive label
- A sequence of consecutive labels surrounded by breakpoints is a *strip*
- Strips are either *increasing*, *decreasing* or both
- L and R is always part of a single increasing strip

Example:

L 1 2 | 8 7 | 3 | 5 6 | 4 | R

Increasing strips: RL12, 56
Decreasing strip: 87
Both: 3, 4

Decreasing strips

•If a permutation contains a decreasing strip, then it is always possible to decrease the number of breakpoints.

•A permutation always contains at least one increasing strip (RL)

•Pick the lowest lable *k* in a decreasing strip

... k-2 k-1 | ... k+1 k | ...

... k-2 k-1 k k+1 ... | ...

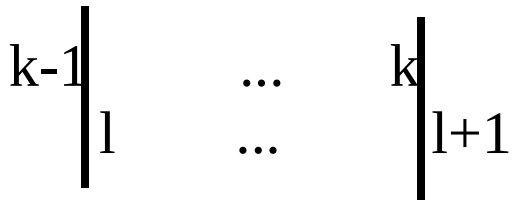
or

... k+1 k | ... k-2 k-1 | ...

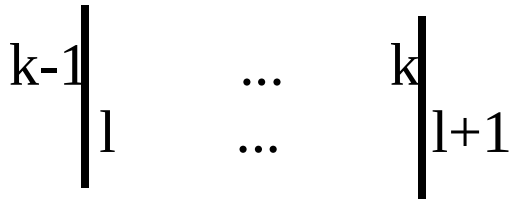
... k+1 k k-1 k-2 ... | ...

Decreasing strips

- If no breakpoint-removing reversal leaves a decreasing strip, then there is a reversal that removes two breakpoints
- k smallest label involved in any decreasing strip
- k-1 must be at the end of an increasing strip
- If k-1 is to the right of k, then the reversal leaves a decreasing strip
- Assume that k-1 is to the left of k
- Let l be the largest label involved in any decreasing strip
- If l+1 is to the left of l a reversal on these gives a decreasing strip



Decreasing strips



•if k is outside of l...l+1 och if l is outside k-1...k a decreasing strip will survive
•if k and l is inside k-1 and l-1, and they are not identical, then there is a strip that can be either reversed or not, thus a decreasing strip will survive choosing the right reversal
•if they are identical, tho breakpoints are removed

Algorithm

Until done:

- Apply reversals to decreasing strips with the smallest possible label provided that the resulting permutation has a decreasing strip
- If the resulting permutation doesn't have a decreasing strip, pick instead the decreasing strip with the largest label
- If there are no decreasing strips, do any reversal that cuts two breakpoints

Algorithm analysis

- For every reversal (but the first) that does not decrease the number of breakpoint, the previous one decreased it by two
- The last reversal decreases the number of breakpoints by two
- Thus the number rate of breakpoints reductions is not less than half the optimum and we have a 2-approximation