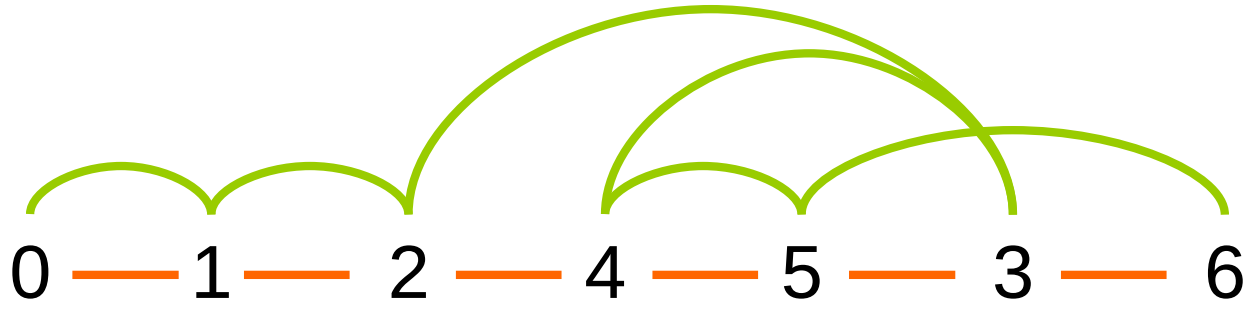
How good is simple reversal sort?

- Not so good actually
- It has to do at most n-1 reversals with permutation of length n
- The algorithm can return a distance that is as large as (n – 1)/2 times the correct result d(Π)
  - For example, if n = 1001, result can be as bad as 500 x d(Π)

Estimating reversal distance by cycle decomposition

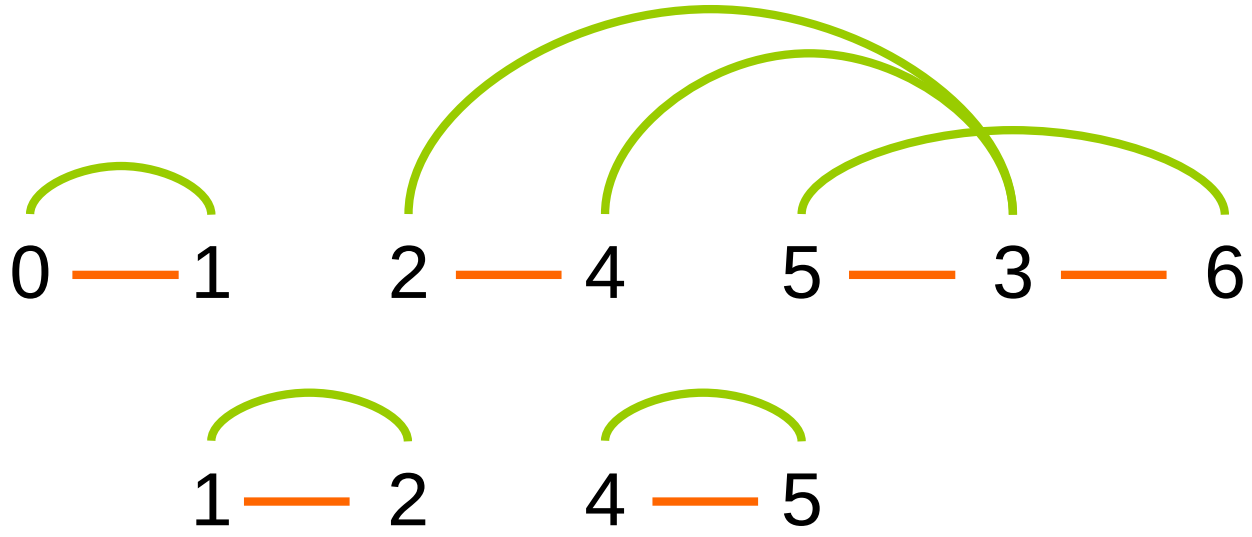
- We can estimate d(Π) by *cycle decomposition*
- Lets represent permutation Π = 1 2 4 5 3 with the following graph



where edges correspond to adjacencies (identity, permutation F)

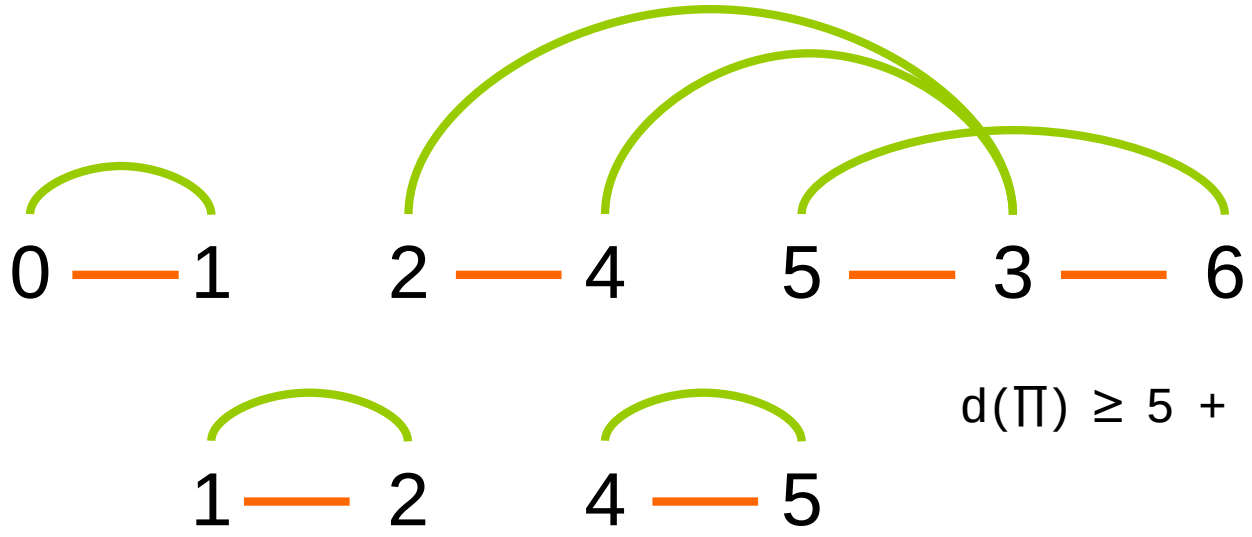
Estimating reversal distance by cycle decomposition

- Cycle decomposition: a set of cycles that
  - have edges with alternating colors
  - do not share edges with other cycles (=cycles are edge disjoint)



Cycle decompositions

- Let c(Π) the maximum number of alternating, edge-disjoint cycles in the graph representation of Π
- The following formula allows estimation of d(Π)
  - d(Π) ≥ n + 1 – c(Π), where n is the permutation length



d(Π) ≥ 5 + 1 – 4 = 2

Claim in Deonier: equality holds for ”most of the usual and interesting biological systems.

Cycle decompositions

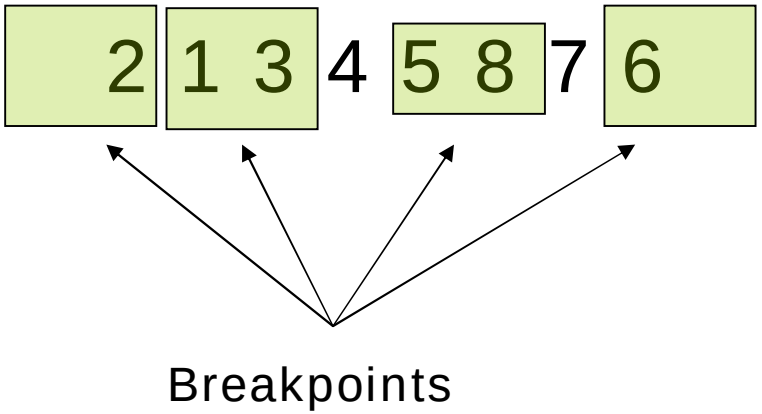
- Cycle decomposition is NP-complete
  - We cannot solve the general problem exactly for large instances
- However, with signed data the problem becomes easy
  - Before going into signed data, lets discuss another algorithm for the general case

Computing reversals with breakpoints

- Lets investigate a better way to compute reversal distance
- First, some concepts related to permutation $\Pi_1\Pi_{2,,,}\Pi_{n-1}\Pi_n$
  - Breakpoint: two elements Π_i and Π_{i+1} are a *breakpoint*, if they are not consecutive numbers
  - Adjacency: if Π_i and Π_{i+1} are consecutive, they are called *adjacency*

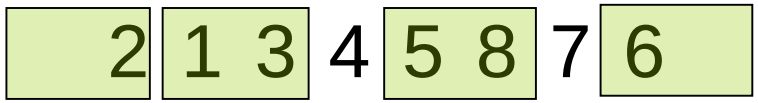
Breakpoints and adjacencies

This permutation contains
four breakpoints *begin*-2, 13, 58, 6-*end* and
five adjacencies 21, 34, 45, 87, 76



Breakpoints

- Each breakpoint in permutation needs to be removed to get to the identity permutation (=our target)
  - Identity permutation does not contain any breakpoints



$b(\Pi) = 4$

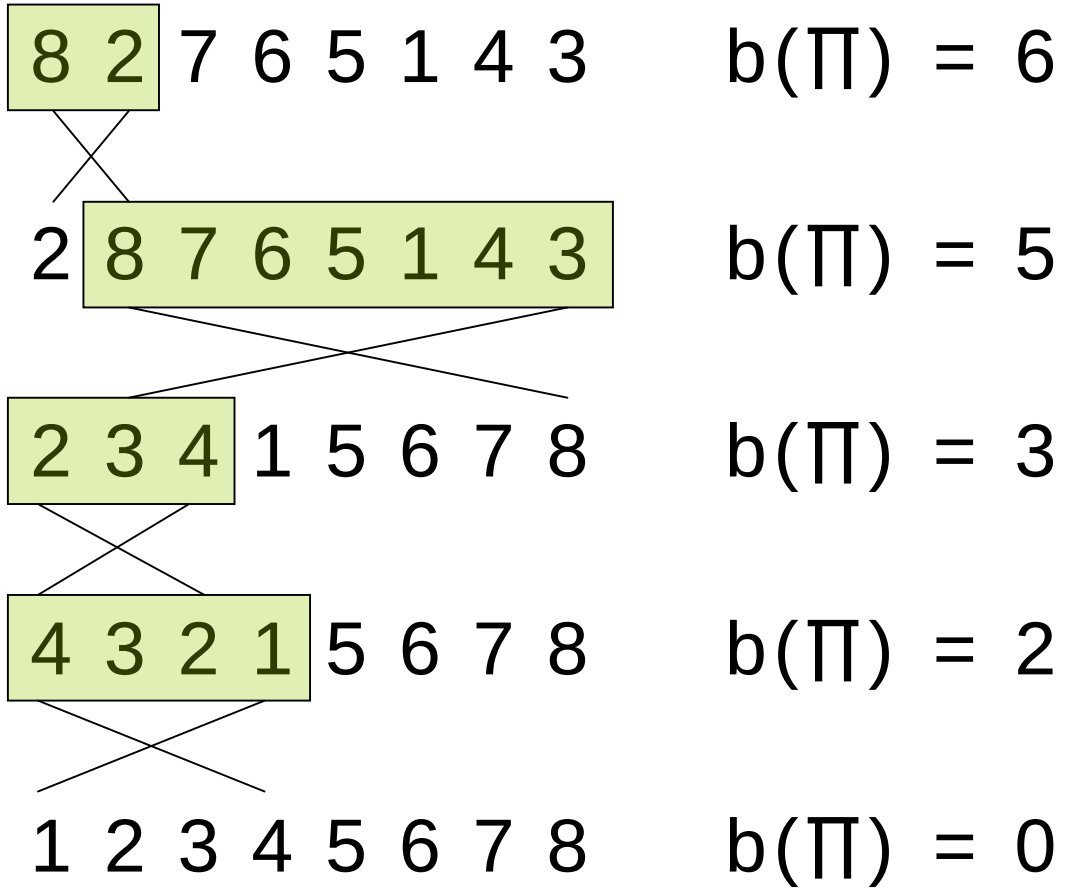
- First and last positions special cases
- Note that each reversal can remove *at most* two breakpoints
- Denote the number of breakpoints by $b(\Pi)$

Breakpoint reversal sort

- Idea: try to remove as many breakpoints as possible (max 2) in every step

1. While b(Π) > 0
2. Choose reversal p that removes most breakpoints
3. Perform reversal p to Π
4. Output Π
5. return

Breakpoint removal: example



Breakpoint removal

- The previous algorithm needs refinement to be correct
- Consider the following permutation:

1 5 6 7 2 3 4 8

- There is no reversal that decreases the number of breakpoints!
- See Jones & Pevzner for detailed description on this

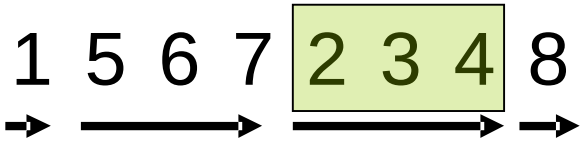
Strip: maximal segment without breakpoints

Breakpoint removal

Increasing strip

Decreasing strip

- Reversal can only decrease breakpoint count if permutation contains *decreasing strips*



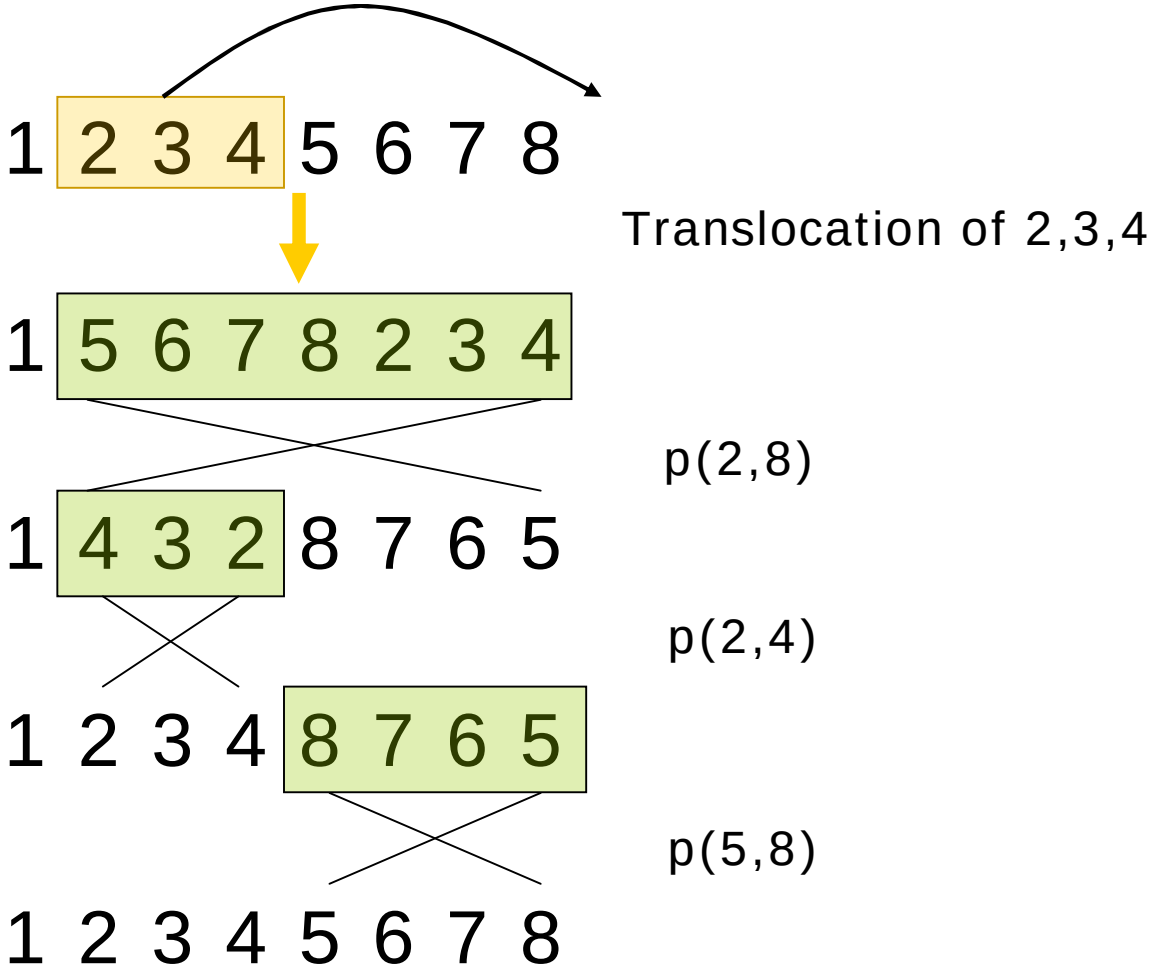
Improved breakpoint reversal sort

1. While b(Π) > 0
2. If Π has a decreasing strip
3. Do reversal p that removes most BPs
4. Else
5. Reverse an increasing strip
6. Output Π
7. return

Is Improved BP removal enough?

- The algorithm works pretty well:
  - It produces a result that is at most four times worse than the optimal result
  - ...is this good?
- We considered only reversals
- What about translocations & duplications?

Translocations via reversals



Genome rearrangements with reversals

- With *unsigned* data, the problem of finding minimum reversal distances is *NP-complete*
  - *Why is this so if sorting is easy?*
- An algorithm has been developed that achieves 1.375-approximation
- However, reversal distance in *signed data* can be computed quickly!
  - It takes linear time w.r.t. the length of permutation (Bader, Moret, Yan, 2001)

Cycle decomposition with signed data

- Consider the following two permutations that include *orientation* of markers
  - *J: +1 +5 -2 +3 +4*
  - *K: +1 -3 +2 +4 -5*

- We modify this representation a bit to include both endpoints of each marker:
  - J': 0 1a 1b 5a 5b 2b 2a 3a 3b 4a 4b 6
  - K': 0 1a 1b 3b 3a 2a 2b 4a 4b 5b 5a 6

Graph representation of J’ and K’

- Drawn online in lecture!

Multiple chromosomes

- In unichromosomal genomes, inversion (reversal) is the most common operation
- In multichromosomal genomes, inversions, translocations, *fissions* and *fusions* are most common

Multiple chromosomes

- Lets represent multichromosomal genome as a set of permutations, with $ denoting the boundary of a chromosome:

5 9 $	Chr 1
1 3 2 8 $	Chr 2
7 6 4 $	Chr 3

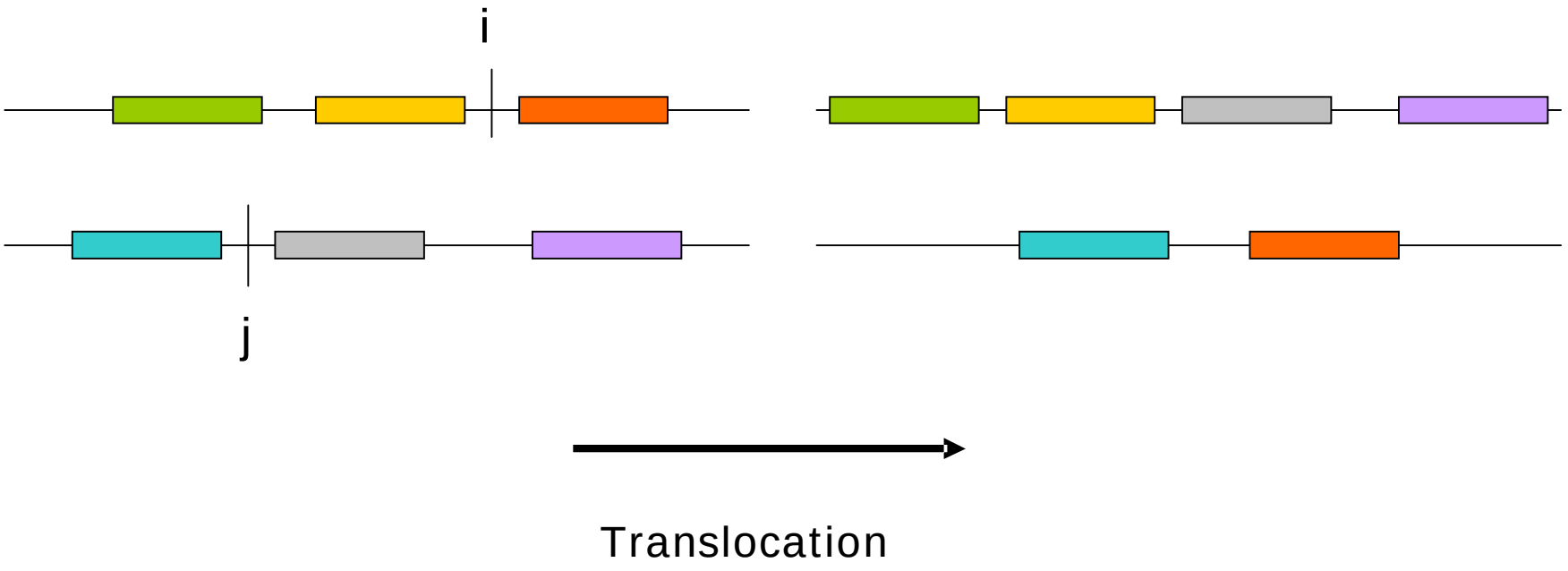
This notation is frequently used in software used to analyse genome rearrangements.

Multiple chromosomes

- Note that when dealing with multiple chromosomes, you need to specify numbering for elements on both genomes

Reversals & translocations

- Reversal p(Π, i, j)
- Translocation p(Π, σ, i, j)

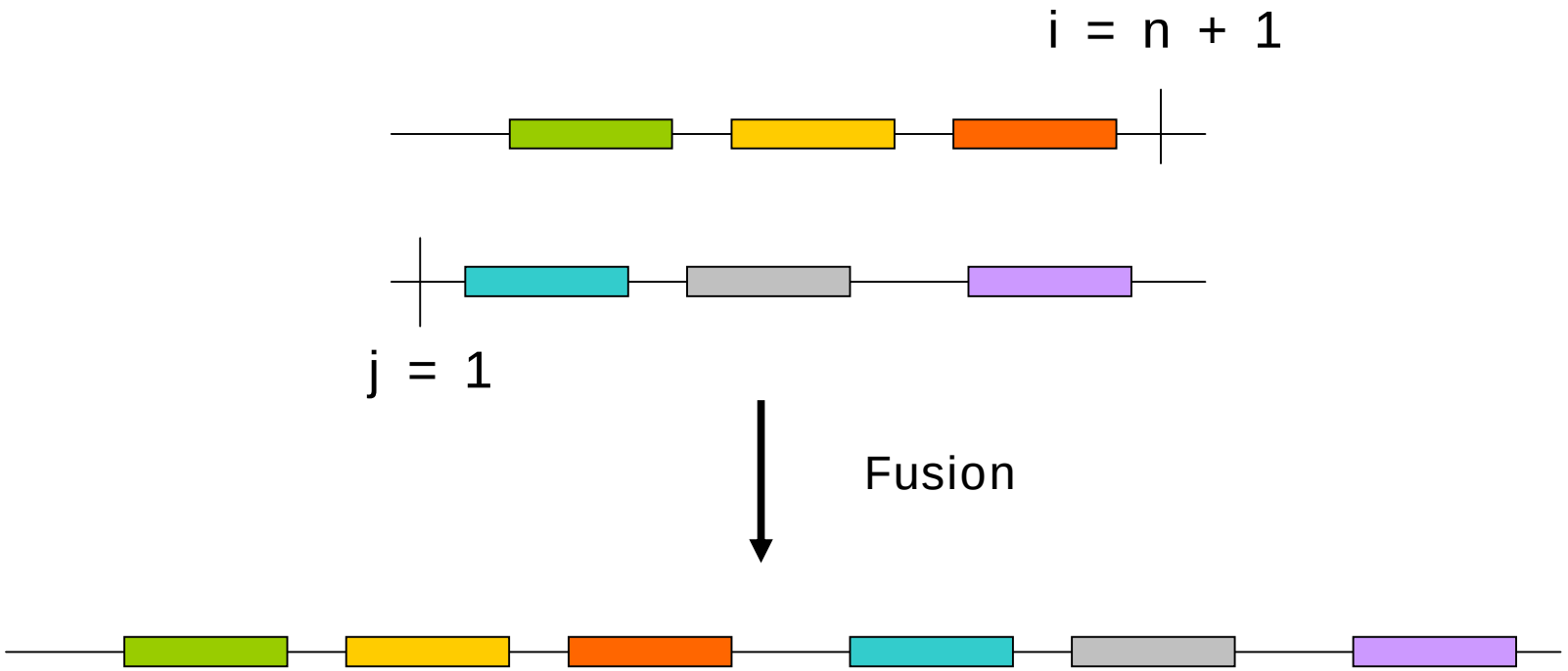


Fusions & fissions

- Fusion: merging of two chromosomes
- Fission: chromosome is split into two chromosomes
- Both events can be represented with a translocation

Fusion

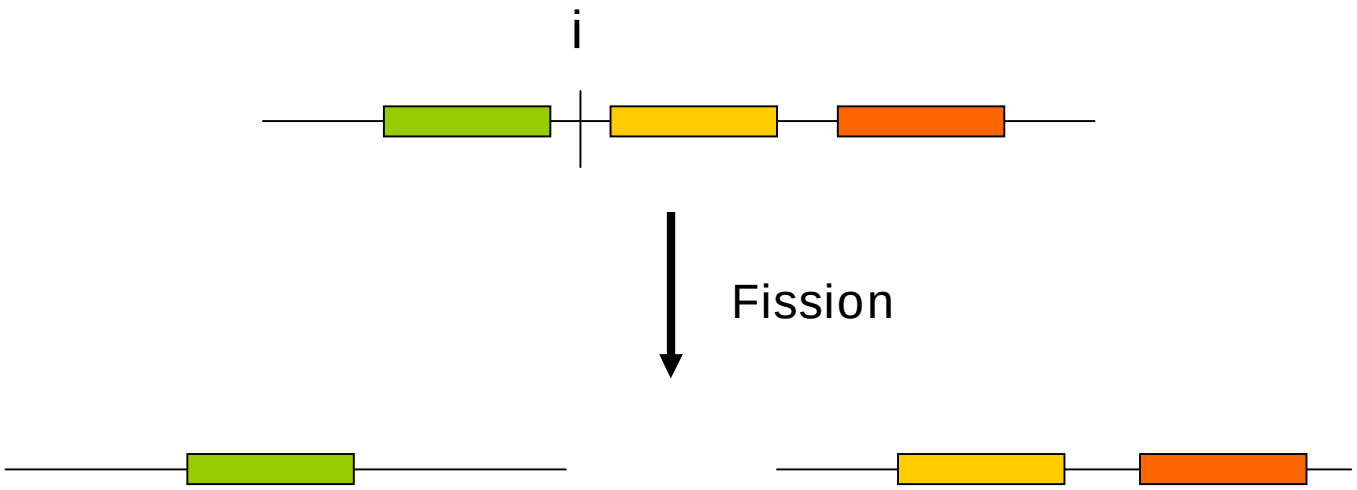
- Fusion by translocation $p(\Pi, \sigma, n+1, 1)$



Fission

Empty chromosome

- Fission by translocation p(Π, ∅, i, 1)



Algorithms for general genomic distance problem

- Hannenhalli, Pevzner: Transforming Men into Mice (polynomial algorithm for genomic distance problem), *36th Annual IEEE Symposium on Foundations of Computer Science*, 1995

Human & mouse revisited

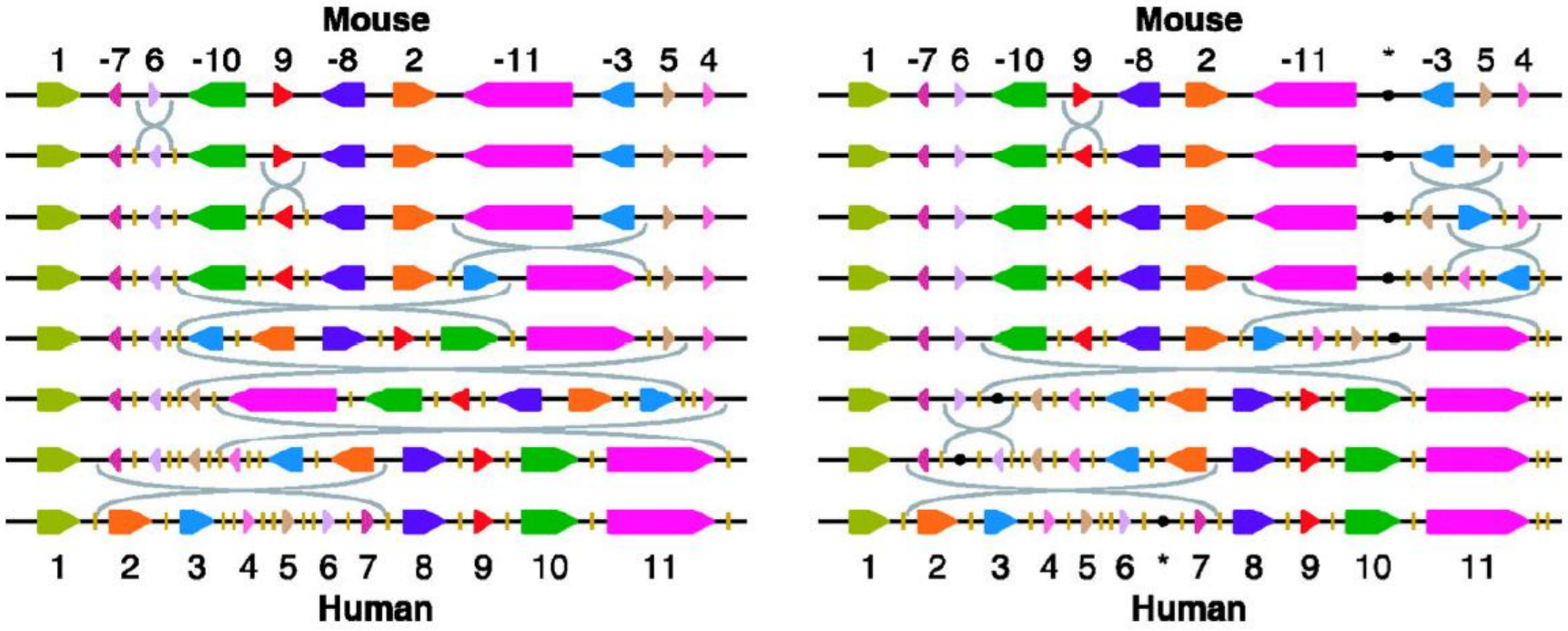
- Human and mouse are separated by about 75-83 million years of evolutionary history
- Only a few hundred rearrangements have happened after speciation from the common ancestory
- Pevzner & Tesler identified in 2003 for 281 synteny blocks a rearrangement from mouse to human with
  - 149 inversions
  - 93 translocations
  - 9 fissions

Discussion

- Genome rearrangement events are very rare compared to, e.g., point mutations
  - We can study rearrangement events further back in the evolutionary history
- Rearrangements are easier to detect in comparison to many other genomic events
- We cannot detect homologs 100% correctly so the input permutation can contain errors

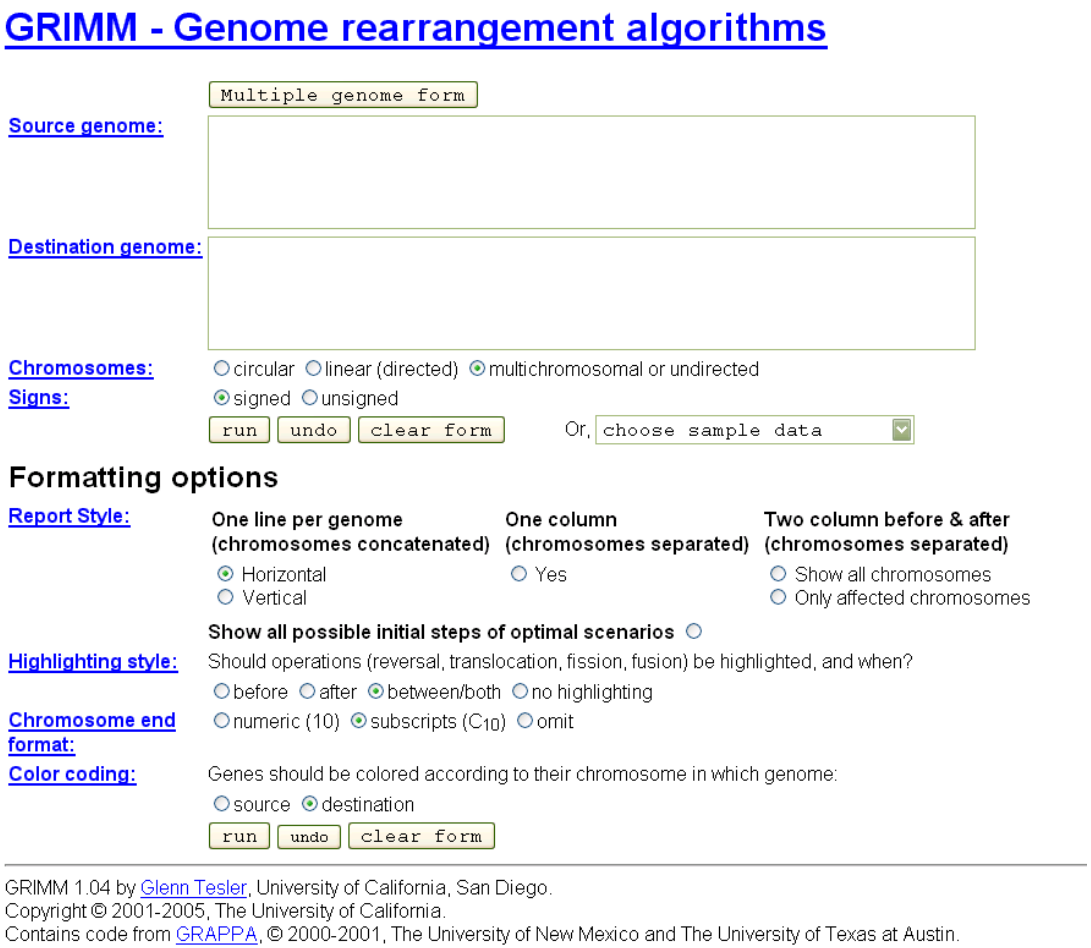
Discussion

- Genome rearrangement is to some degree constrained by the number and size of repeats in a genome
  - Notice how the importance of genomic repeats pops up once again
- Sequencing gives us (usually) signed data so we can utilize faster algorithms
- What if there are more than one optimal solution?



Two different genome rearrangement scenarios giving the same result.

GRIMM demonstration



Glenn Tesler, GRIMM: genome rearrangements web server. Bioinformatics, 2002,

GRIMM file format

```
# useful comment about first genome
# another useful comment about it
>name of first genome
1 -4 2 $ # chromosome  1
-3 5 6 # chromosome 2
>name of second genome
5 -3  $
6 $
2 -4 1 $
```

GRIMM supports analysis of one, two or more genomes

http://grimm.ucsd.edu/GRIMM/grimm_instr.html