

# Quiz 1

1 Which of the following statements about the inversion model are true?

- A The inversion distance depends only on the number of cycles in the relational diagram.
- B Every bad component in the diagram is a hurdle.
- C A good component can always be sorted with (safe) split inversions.
- D A super hurdle can be optimally sorted with a neutral inversion.
- E A diagram with an even number of bad components can be a fortress.

# Topics of today:

Canonical inversion distance and sorting:

1. Component tree
2. Circularizing linear chromosomes

Singular DCJ-indel distance and sorting:

1. Indels: insertions and deletions
2. Relational graph of singular genomes
3. Runs

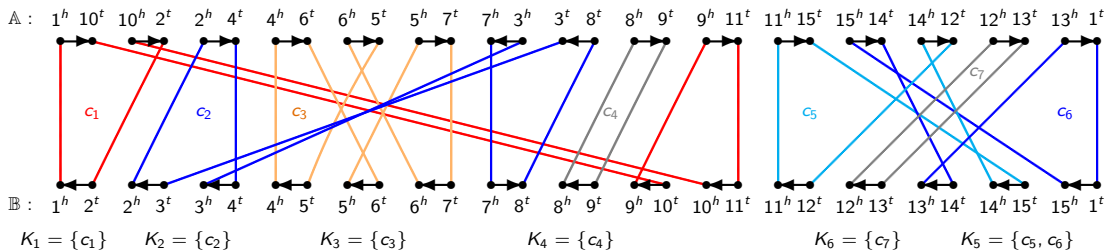
# Chained and nested components on the relational diagram

Alternative to component separation: **chaining** and **nesting** relationships between components

**Chain:**  $\left\{ \begin{array}{l} \text{sequence of components } K_1, K_2, \dots, K_\ell \\ \text{the rightmost adjacency-edge of } K_i \text{ is succeeded by} \\ \text{the leftmost adjacency-edge of } K_{i+1}, \text{ for } 1 \leq i \leq \ell \end{array} \right.$

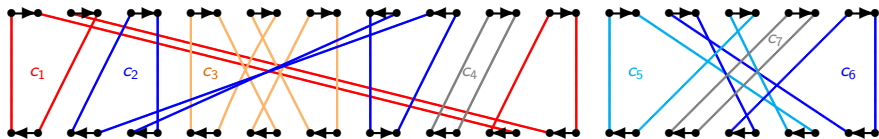
**Maximal chain:** cannot be extended to the left nor to the right.

A maximal chain  $H$  is **nested** in a component  $K$  when the leftmost adjacency-edge of  $H$  is preceded by an adjacency-edge of  $K$  and the rightmost adjacency-edge of  $H$  is succeeded by an adjacency-edge of  $K$ .



Maximal chains:  $\left\{ \begin{array}{l} H_1 = K_1 \bowtie K_5, \\ H_2 = K_2 \bowtie K_4 \text{ (nested in component } K_1), \\ H_3 = K_3 \text{ (nested in component } K_2), \\ H_4 = K_6 \text{ (nested in component } K_5) \end{array} \right.$

# Chained component tree $\Upsilon_{\blacksquare}$ (rooted)



$$K_1 = \{c_1\} \quad K_2 = \{c_2\}$$

$$K_3 = \{c_3\}$$

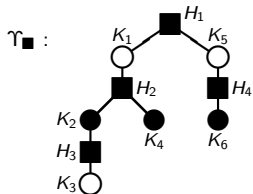
$$K_4 = \{c_4\}$$

$$K_6 = \{c_7\}$$

$$K_5 = \{c_5, c_6\}$$

$$\text{Maximal chains: } \begin{cases} H_1 = K_1 \bowtie K_5, \\ H_2 = K_2 \bowtie K_4 \text{ (nested in component } K_1), \\ H_3 = K_3 \text{ (nested in component } K_2), \\ H_4 = K_6 \text{ (nested in component } K_5) \end{cases}$$

1. One round node per component  $K_i$ :  $\begin{cases} \text{bad node } (\circ): K_i \text{ is a bad component;} \\ \text{good node } (\bullet): K_i \text{ is a trivial or a good component.} \end{cases}$
2. One square ( $\blacksquare$ ) node per maximal chain  $H_i$ , whose children are the round nodes corresponding to the components of  $H_i$ . A square node is either the root or a child of the component in which  $H_1$  is nested.



$P$ : path connecting two distinct round nodes  $u_1$  and  $u_2$  in  $\Upsilon_{\blacksquare}(\mathbb{A}, \mathbb{B})$

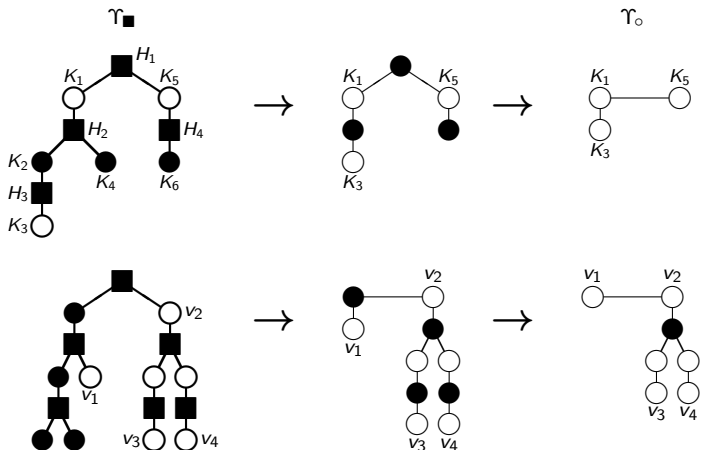
round nodes in  $P \setminus \{u_1, u_2\}$ : components that separate  $u_1$  and  $u_2$  in  $RG(\mathbb{A}, \mathbb{B})$ .

# Component tree $\Upsilon_\circ$ (unrooted)

**Max-flower:** maximal connected subgraph composed of good and/or square nodes only.

**Contraction** of  $\Upsilon_\blacksquare$  into unrooted  $\Upsilon_\circ$ : for each max-flower  $F$  of  $\Upsilon_\blacksquare$ :

1. Replace  $F$  by a single good round node  $g$ , such that  $g$  is connected to all bad nodes connected to  $F$
2.  $\left\{ \begin{array}{l} \text{If } g \text{ has exactly two neighbors } b_1 \text{ and } b_2: \text{ remove } g \text{ from the tree and connect } b_1 \text{ to } b_2; \\ \text{If } g \text{ is a leaf: simply remove } g \text{ from the tree (all leaves in } \Upsilon_\circ \text{ are bad).} \end{array} \right.$



# Cost of covering the component tree $\Upsilon_0$

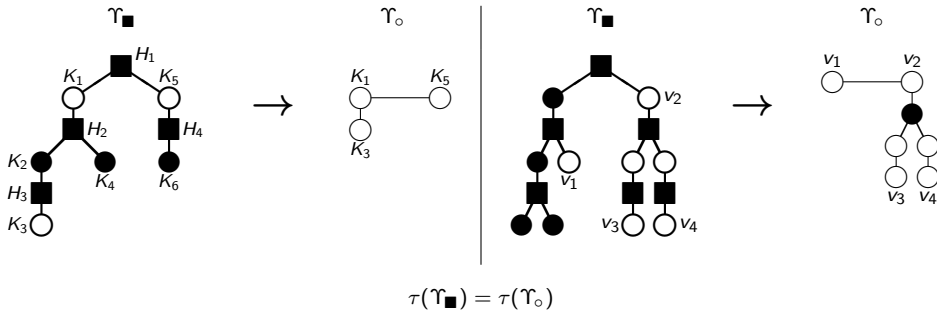
path  $P$  in  $\Upsilon_0$ :  $\begin{cases} \text{short: contains a single bad node} \\ \text{long: contains at least two bad nodes} \end{cases}$

cost of path  $P$ :  $\tau(P) \begin{cases} P \text{ is short : } \tau(P) = 1 \text{ (cut a bad component)} \\ P \text{ is long : } \tau(P) = 2 \text{ (merge two or more components)} \end{cases}$

Cover of  $\Upsilon_0$ : set of paths  $\hat{\mathcal{P}}$  such that each bad node of  $\Upsilon_0$  is contained in at least one path  $P \in \hat{\mathcal{P}}$

Cost of cover  $\hat{\mathcal{P}}$ :  $\tau(\hat{\mathcal{P}}) = \sum_{P \in \hat{\mathcal{P}}} \tau(P)$

Cost of an optimal cover of  $\Upsilon_0$ :  $\tau(\Upsilon_0) = \min_{\hat{\mathcal{P}} \text{ is a cover of } \Upsilon_0} \tau(\hat{\mathcal{P}})$



## Covering the component tree $\Upsilon_o$

$\mathcal{L}$ : # of leaves in  $\Upsilon_o$ ; **Branching node** of  $\Upsilon_o$ : any node whose degree is  $\geq 3$

**Leaf-branch** of  $\Upsilon_o$ :  $\begin{cases} \text{if } \mathcal{L} \leq 2: \text{ the complete tree } \Upsilon_o \\ \text{if } \mathcal{L} \geq 3: \text{ maximal path } u_1, u_2, \dots, u_k, \text{ such that } u_1 \text{ is a leaf of } \Upsilon_o \text{ and,} \\ \text{for } i = 2, \dots, k, \text{ the degree of internal node } u_i \text{ in } T \text{ is two} \end{cases}$

A leaf-branch may be a path of length 1 (a leaf directly connected to a branching node of  $\Upsilon_o$ )

**Traversal**: path connecting two leaves of  $\Upsilon_o$

Suppose  $\mathcal{L} = 2, 4, 6, \dots$ :

$\widehat{\mathcal{P}}_T(\Upsilon_o)$ : smallest set of traversals covering all nodes of  $\Upsilon_o$  :  $|\widehat{\mathcal{P}}_T(\Upsilon_o)| = \frac{\mathcal{L}}{2}$

COVERTREETWITHTRAVERSALS

**Input**: unrooted tree  $\Upsilon_o$  with  $\mathcal{L} = 2n$  leaves

**Output**: set  $\widehat{\mathcal{P}}_T$  of  $n$  traversals covering all nodes of  $\Upsilon_o$

Based on any planar view of  $\Upsilon_o$ , enumerate the leaves from 1 to  $2n$  in circular order;

$\widehat{\mathcal{P}}_T = \emptyset$ ;

**for**  $i = 1$  **to**  $n$  **do**

$\widehat{\mathcal{P}}_T = \widehat{\mathcal{P}}_T \cup \{\text{traversal connecting leaves } i \text{ and } i + n\}$ ;

**Return**  $\widehat{\mathcal{P}}_T$ ;

# Computing $\tau(\Upsilon_o)$

Lower bound for the cost of an optimal cover of  $\Upsilon_o$ :  $\tau(\Upsilon_o) \geq \mathcal{L}$

Each traversal  $T$  has cost  $\tau(T) = 2$

If  $\mathcal{L}$  is even,  $\widehat{\mathcal{P}}_T(\Upsilon_o)$  is an optimal cover:

$$\Rightarrow \tau(\Upsilon_o) = \tau(\widehat{\mathcal{P}}_T(\Upsilon_o)) = 2 \frac{\mathcal{L}}{2} = \mathcal{L}$$

If  $\mathcal{L}$  is odd and  $\Upsilon_o$  has a short leaf-branch  $s$  ( $\tau(s) = 1$ ):

$$\Rightarrow \tau(\Upsilon_o) = \tau(\widehat{\mathcal{P}}_T(\Upsilon_o \setminus s)) + \tau(s) = 2 \frac{\mathcal{L}-1}{2} + 1 = \mathcal{L}$$

If  $\mathcal{L}$  is odd and  $\Upsilon_o$  has no short leaf-branch ("fortress"); let  $\ell$  be any long leaf-branch of  $\Upsilon_o$  ( $\tau(\ell) = 2$ ):

$$\Rightarrow \tau(\Upsilon_o) = \tau(\widehat{\mathcal{P}}_T(\Upsilon_o \setminus \ell)) + \tau(\ell) = 2 \frac{\mathcal{L}-1}{2} + 2 = \mathcal{L} + 1$$

The cost of any optimal cover of  $\Upsilon_o$  is:

$$\tau(\Upsilon_o) = \begin{cases} \mathcal{L} + 1 & \text{if } \mathcal{L} \text{ is odd and all leaf-branches are long ("fortress"),} \\ \mathcal{L} & \text{otherwise.} \end{cases}$$



# Canonical inversion distance

$$d_{\text{INV}}(\mathbb{A}, \mathbb{B}) = n - |\mathcal{C}| + \tau_*$$

where

$$\tau_* = \tau(\Upsilon_{\circ}(\mathbb{A}, \mathbb{B})) = h + f$$

# Components are framed conserved intervals

Assuming that  $\mathbb{B} = (1 \ 2 \ 3 \ \dots \ 16)$ , let us identify its **framed conserved intervals** with respect to

$$\mathbb{A} = (1 \ \bar{4} \ 2 \ 3 \ 5 \ 7 \ 6 \ 8 \ \bar{16} \ \bar{14} \ \bar{15} \ \bar{13} \ \bar{11} \ \bar{12} \ \bar{10} \ 9)$$

For given  $i \geq 1$  and  $j \geq 1$  such that  $i+j \leq n+1$ :

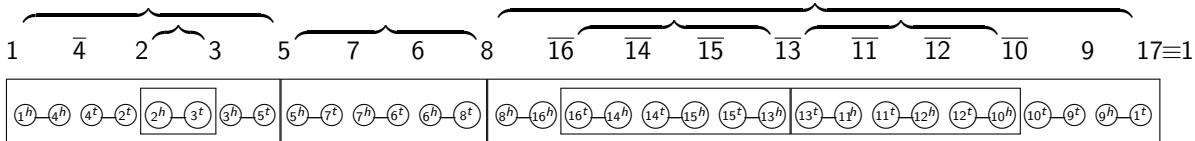
**Conserved interval:** interval of  $\mathbb{A}$  composed of values  $i, i+1, \dots, i+j$  (assuming  $n+1 \equiv 1$ )

**Framed conserved interval**  $\begin{cases} \text{direct: first element is } i \text{ and last element is } i+j; \text{ or} \\ \text{reverse: first element is } \bar{i+j} \text{ and last element is } \bar{i} \end{cases}$

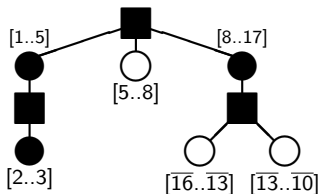
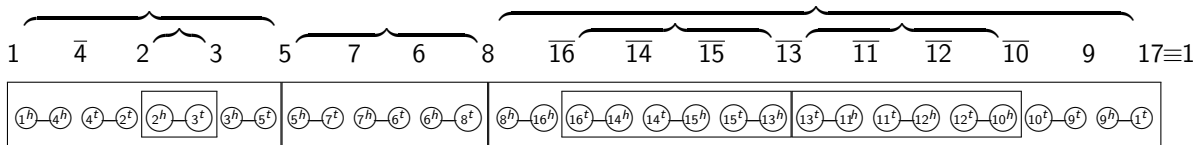
Direct:  $[1..5]; [2..3]; [5..8]; [8..17]$       Reverse:  $[\bar{16}..\bar{13}]; [\bar{13}..\bar{10}], [\bar{16}..\bar{10}]$

**Component:** framed conserved interval that is not a union of framed conserved intervals

Direct:  $[1..5]; [2..3]; [5..8]; [8..17]$       Reverse:  $[\bar{16}..\bar{13}]; [\bar{13}..\bar{10}]$



## Components are framed conserved intervals



The inversion distance can be computed in linear time, by efficiently identifying chains of framed conserved intervals (Bergeron *et al.*, 2002: Common intervals and sorting by reversals: a marriage of necessity)

An optimal inversion sorting scenario can be computed in subquadratic time.  
(Tannier and Sagot, 2004: Sorting by reversals in subquadratic time)

# Canonical inversion distance of linear chromosomes

Given canonical linear chromosomes  $\mathbb{A}$  and  $\mathbb{B}$ :

Add one new family (e.g. 0) and circularize chromosome  $\mathbb{B}$  into  $\mathbb{B}' = (0 \mathbb{B})$

$$d_{\text{INV}}(\mathbb{A}, \mathbb{B}) = \min \begin{cases} d_{\text{INV}}((0 \mathbb{A}), \mathbb{B}') \\ d_{\text{INV}}((\bar{0} \mathbb{A}), \mathbb{B}') \end{cases}$$

Example:

$$\mathbb{A} = [\bar{5} \ 1 \ 2 \ \bar{3} \ 4] \quad \text{and} \quad \mathbb{B} = [1 \ 2 \ 3 \ 4 \ 5]$$

$$\mathbb{B}' = (0 \ 1 \ 2 \ 3 \ 4 \ 5)$$

$$d_{\text{INV}}((0 \ \bar{5} \ 1 \ 2 \ \bar{3} \ 4), \mathbb{B}') = 3$$

$$d_{\text{INV}}((\bar{0} \ \bar{5} \ 1 \ 2 \ \bar{3} \ 4), \mathbb{B}') = 2$$

$$d_{\text{INV}}(\mathbb{A}, \mathbb{B}) = 2$$

## Quiz 2

1 What is the bottleneck of the running time of inversion sorting?

A Finding inversions that fix bad components.

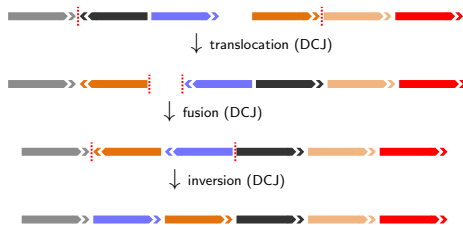
B Finding split inversions.

C Finding safe split inversions.

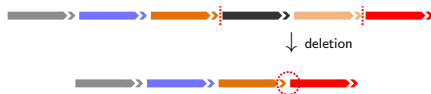
D Finding inversions that merge bad components.

# DCJ and indels

## ▶ DCJ: structural rearrangements



## ▶ Modifying the content: insertions and deletions (**indels**)



# Singular DCJ-indel model

Recall that  $\mathcal{G}_* = \mathcal{G}(\mathbb{A}) \cap \mathcal{G}(\mathbb{B})$

Let  $\begin{cases} \mathcal{A} = \mathcal{G}(\mathbb{A}) \setminus \mathcal{G}_* \text{ (set of genes exclusive to genome } \mathbb{A}) \\ \mathcal{B} = \mathcal{G}(\mathbb{B}) \setminus \mathcal{G}_* \text{ (set of genes exclusive to genome } \mathbb{B}) \end{cases}$

Restrictions for indel operations:

- ▶ At most one chromosome can be deleted or inserted at once
- ▶ Only genes of set  $\mathcal{A}$  can be deleted
- ▶ Only genes of set  $\mathcal{B}$  can be inserted

# Singular DCJ-indel model

Given two singular genomes  $\mathbb{A}$  and  $\mathbb{B}$ ,...

**Singular DCJ-indel Distance Problem:** Compute the minimum number of DCJ and indel operations required to transform  $\mathbb{A}$  into  $\mathbb{B}$ .

Denote by  $d_{\text{DCJ}}^{\text{ID}}(\mathbb{A}, \mathbb{B})$  the DCJ-indel distance of  $\mathbb{A}$  and  $\mathbb{B}$ .

**Singular DCJ-indel Sorting Problem:** Find a sequence of  $d_{\text{DCJ}}^{\text{ID}}(\mathbb{A}, \mathbb{B})$  DCJ and indel operations that transform  $\mathbb{A}$  into  $\mathbb{B}$ .

First upper bound:

$$d_{\text{DCJ}}^{\text{ID}}(\mathbb{A}, \mathbb{B}) \leq d_{\text{DCJ}}(\mathbb{A}_c, \mathbb{B}_c) + |\mathcal{A}| + |\mathcal{B}|$$

where  $\begin{cases} \mathbb{A}_c \text{ is the genome obtained from } \mathbb{A} \text{ by simply removing the genes of } \mathcal{A} \\ \mathbb{B}_c \text{ is the genome obtained from } \mathbb{B} \text{ by simply removing the genes of } \mathcal{B} \end{cases}$



# Relational graph of singular genomes

Given two singular genomes  $\mathbb{A}$  and  $\mathbb{B}$ , their **relational graph**  $RG(\mathbb{A}, \mathbb{B}) = (V, E)$  is described as follows:

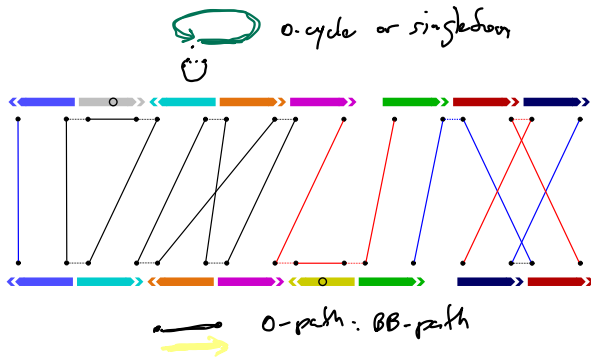
1.  $V = V(\xi(\mathbb{A})) \cup V(\xi(\mathbb{B}))$  : there is a vertex for each extremity of each gene in  $\mathbb{A}$   
and a vertex for each extremity of each gene in  $\mathbb{B}$

Each vertex  $v$  has a label  $\ell(v)$ , that corresponds to the extremity it represents.

2.  $E = E_\alpha(\mathbb{A}) \cup E_\alpha(\mathbb{B}) \cup E_\xi \cup E_{\text{ID}}(\mathbb{A}) \cup E_{\text{ID}}(\mathbb{B})$ , where:

- ▶ **Adjacency edges:** 
$$\begin{cases} E_\alpha(\mathbb{A}) = \{uv : u, v \in V(\xi(\mathbb{A})) \text{ and } \ell(u)\ell(v) \in \alpha(\mathbb{A})\} \\ E_\alpha(\mathbb{B}) = \{uv : u, v \in V(\xi(\mathbb{B})) \text{ and } \ell(u)\ell(v) \in \alpha(\mathbb{B})\} \end{cases}$$
- ▶ **Extremity edges:**  $E_\xi = \{uv : u \in V(\xi(\mathbb{A})) \text{ and } v \in V(\xi(\mathbb{B})) \text{ and } \ell(u) = \ell(v)\}$
- ▶ **Indel edges:** 
$$\begin{cases} E_{\text{ID}}(\mathbb{A}) = \{uv : \ell(u) = g^t \text{ and } \ell(v) = g^h \text{ and } g \in \mathcal{A}\} \\ E_{\text{ID}}(\mathbb{B}) = \{uv : \ell(u) = g^t \text{ and } \ell(v) = g^h \text{ and } g \in \mathcal{B}\} \end{cases}$$

# Relational graph of singular genomes



components can be **indel-inclosing** or **indel-free**

$$n = |g_*|$$

If  $A_c = B_c$ ,  
 $RG(A, B)$  has only 2-cycles and 1-paths:

$$2n = 2|C| + |P_{AB}| \Rightarrow n = |C| + \frac{|P_{AB}|}{2}$$

Every vertex has degree one or two:  
 $RG(A, B)$  is a collection of paths and cycles

cycle with  $k$  edges in  $E_\xi$ :  $k$ -cycle or  $c_k$   
 path with  $k$  edges in  $E_\xi$ :  $k$ -path or  $p_k$

if  $k = 0$  the component is a **singleton**

$$\left\{ \begin{array}{l} C = \{c_k : k \geq 2\} : \text{set of cycles (} k \text{ is even)} \\ S = \{c_k : k = 0\} : \text{set of circular singletons} \\ P_{AA} = \{p_k : \text{starts and ends in } A\} : \\ \quad \text{set of } AA\text{-paths (} k \text{ is even)} \\ P_{BB} = \{p_k : \text{starts and ends in } B\} : \\ \quad \text{set of } BB\text{-paths (} k \text{ is even)} \\ P_{AB} = \{p_k : \text{starts in } A \text{ and ends in } B\} : \\ \quad \text{set of } AB\text{-paths (} k \text{ is odd)} \end{array} \right.$$

$|P_{AB}|$  is even ( $E_\xi$  has  $2n$  edges)

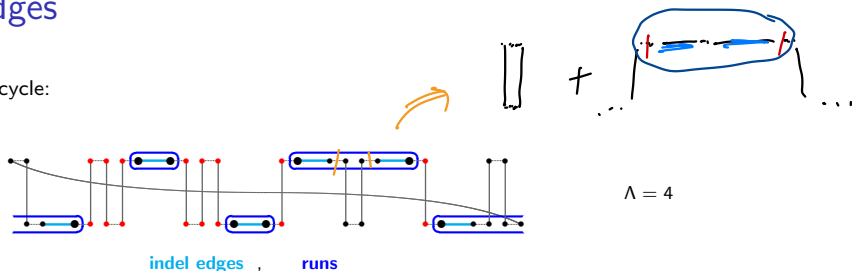
$$|P_{AA}| + |P_{BB}| + |P_{AB}| = \kappa(A) + \kappa(B)$$

Otherwise, if  $A_c \neq B_c$ :

$$n > |C| + \frac{|P_{AB}|}{2}$$

# Runs of indel-edges

One indel-enclosing cycle:



$\Lambda(C)$  is the number of **runs** in component  $C$

$\Lambda$		
0	cycles or paths	<i>indel-free</i>
1	cycles, paths and singletons	
2	cycles, paths	<i>indel-enclosing</i>
3	paths	
4	cycles, paths	
5	paths	
6	cycles, paths	
⋮	⋮	
⋮	⋮	

Each **run** can be inserted/deleted at once

⇒ Second upper bound:

$$d_{\text{DCJ}}^{\text{ID}}(\mathbb{A}, \mathbb{B}) \leq n - |\mathcal{C}| - \frac{|\mathcal{P}_{\mathbb{A}\mathbb{B}}|}{2} + \sum_{C \in \mathcal{R}\mathcal{G}} \Lambda(C)$$

# References

The Inversion Distance Problem

(Anne Bergeron, Julia Mixtacki and Jens Stoye)

In: Mathematics of Evolution and Phylogeny. Gascuel O (Ed); (2005)

Double Cut and Join with Insertions and Deletions

(Marília D.V. Braga, Eyla Willing and Jens Stoye)

JCB, Vol. 18, No. 9 (2011)