Topics of today:

Canonical inversion distance and sorting:

1. Breakpoint diagram
2. Split / Neutral / Joining inversions
3. Good / bad components
4. Safe inversions and overlap graph
5. Hurdles and fortress / component tree

Canonical inversion model - circular chromosomes

(Unichromosomal genomes $\equiv$ chromosomes)

Given two canonical circular chromosomes $\mathbb{A}$ and $\mathbb{B}$,...

Canonical Inversion Distance Problem:	Compute the minimum number of inversions required to transform $\mathbb{A}$ into $\mathbb{B}$. Denote by $\mathrm{d}_{\mathrm{INV}}(\mathbb{A},\mathbb{B})$ the inversion distance of $\mathbb{A}$ and $\mathbb{B}$.
Canonical Inversion Sorting Problem:	Find a sequence of $\mathrm{d}_{\mathrm{INV}}(\mathbb{A},\mathbb{B})$ inversions that transform $\mathbb{A}$ into $\mathbb{B}$.

Breakpoint diagram of canonical circular chromosomes

Let $\mathbb{A}$ and $\mathbb{B}$ be canonical circular chromosomes, with $n = |\mathcal{G}_\star|$.

The **breakpoint diagram** $BD(\mathbb{A}, \mathbb{B}) = (V, E)$ is described as follows:

1. $V = \bigcup_{\mathtt{X} \in \mathcal{G}_\star} \{\mathtt{X}^h, \mathtt{X}^t\} \quad \Rightarrow V = \xi(\mathbb{A}) = \xi(\mathbb{B}) \;\; ; \quad |V| = 2n$

   there is a vertex for each extremity of each gene in $\mathcal{G}_\star$

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

   The vertices are drawn in one line, next to each other.

   The vertices must follow the same (circular) order of the corresponding extremities in chromosome $\mathbb{A}$, according to one of the two reading directions.

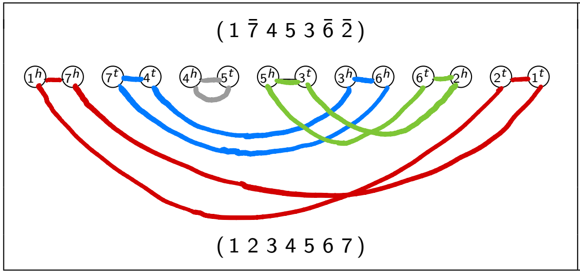
2. $E = E_\Gamma(\mathbb{A}) \cup E_\Gamma(\mathbb{B})$, where:

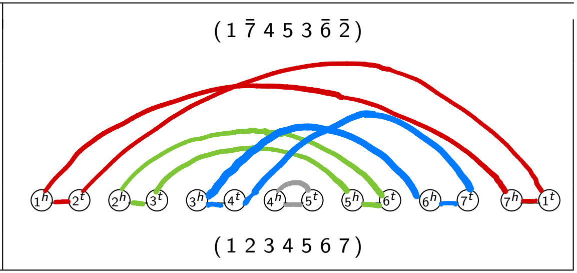
   - **Adjacency edges:** $\begin{cases} E_\Gamma(\mathbb{A}) = \{uv : u, v \in V(\xi(\mathbb{A})) \text{ and } \ell(u)\ell(v) \in \Gamma(\mathbb{A})\} \\ E_\Gamma(\mathbb{B}) = \{uv : u, v \in V(\xi(\mathbb{B})) \text{ and } \ell(u)\ell(v) \in \Gamma(\mathbb{B})\} \end{cases}$

     The number of edges is $|E| = 2n$ (n adjacency edges per chromosome)

Two equivalent breakpoint diagrams

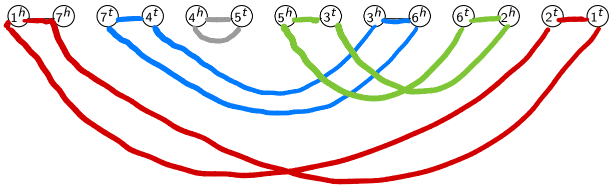
$$BD(\mathbb{A}, \mathbb{B}) \quad \cong \quad BD(\mathbb{B}, \mathbb{A})$$


(1 $\bar{7}$ 4 5 3 $\bar{6}$ $\bar{2}$)
1^h 7^h 7^t 4^t 4^h 5^t 5^h 3^t 3^h 6^h 6^t 2^h 2^t 1^t
(1 2 3 4 5 6 7)


(1 $\bar{7}$ 4 5 3 $\bar{6}$ $\bar{2}$)
1^h 2^t 2^h 3^t 3^h 4^t 4^h 5^t 5^h 6^t 6^h 7^t 7^h 1^t
(1 2 3 4 5 6 7)

Properties of the breakpoint diagram

$\mathbb{A} =$ $(1\ \bar{7}\ 4\ 5\ 3\ \bar{6}\ \bar{2})$



$\mathbb{B} =$ $(1\ 2\ 3\ 4\ 5\ 6\ 7)$

$n = |\mathcal{G}_\star| = 7$

Every vertex has degree two:

$BD(\mathbb{A}, \mathbb{B})$ is a collection of (even) cycles (alternating edes in $E_\Gamma(\mathbb{A})$ and in $E_\Gamma(\mathbb{B})$)

cycle with k edges: k-cycle (always even)

$\mathcal{C}$ = set of cycles in $BD(\mathbb{A}, \mathbb{B})$

If $\mathbb{A} = \mathbb{B}$,
$RG(\mathbb{A}, \mathbb{B})$ has only 2-cycles:

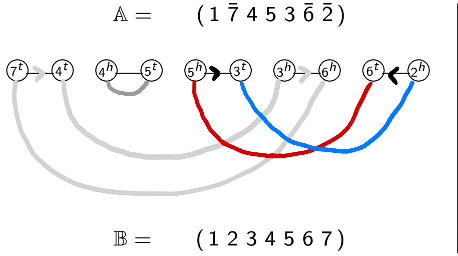
$$2n = 2|\mathcal{C}| \quad \Rightarrow \quad n = |\mathcal{C}|$$

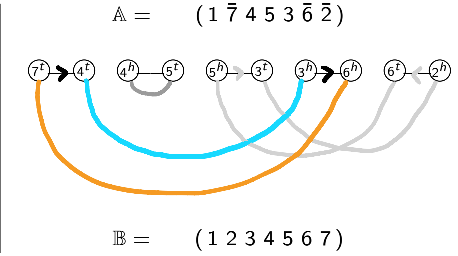
Otherwise, if $\mathbb{A} \neq \mathbb{B}$:

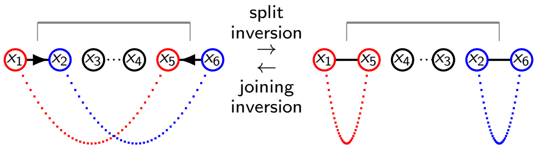
$$n > |\mathcal{C}|$$

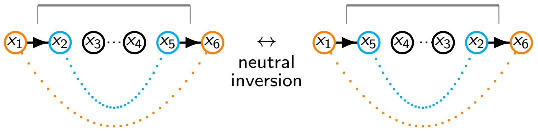
Types of inversion and lower bound for the inversion distance

Assign one (arbitrary) direction to each cycle of $BD(\mathbb{A}, \mathbb{B})$


$\mathbb{A} = \; (1\ \bar{7}\ 4\ 5\ 3\ \bar{6}\ \bar{2})$
7^t 4^t 4^h 5^t 5^h 3^t 3^h 6^h 6^t 2^h
$\mathbb{B} = \; (1\ 2\ 3\ 4\ 5\ 6\ 7)$


$\mathbb{A} = \; (1\ \bar{7}\ 4\ 5\ 3\ \bar{6}\ \bar{2})$
7^t 4^t 4^h 5^t 5^h 3^t 3^h 6^h 6^t 2^h
$\mathbb{B} = \; (1\ 2\ 3\ 4\ 5\ 6\ 7)$


x_1 x_2 x_3 x_4 x_5 x_6
split inversion
$\rightarrow$
$\leftarrow$
joining inversion
x_1 x_5 x_4 x_3 x_2 x_6


x_1 x_2 x_3 x_4 x_5 x_6
$\leftrightarrow$
neutral inversion
x_1 x_5 x_4 x_3 x_2 x_6

Lower bound for the inversion distance: $\mathrm{d}_{\mathrm{INV}}(\mathbb{A}, \mathbb{B}) \geq n - |\mathcal{C}|$

Types of cycles

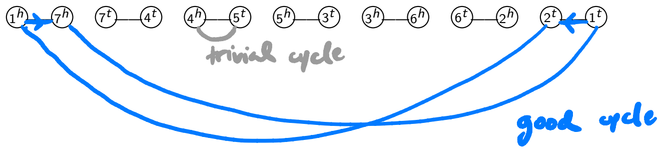
Trivial cycle: one adjacency in each chromosome

2-cycle (sorted)

Good cycle: at least one pair of adjacencies with opposite directions

Can be split into two cycles by applying an inversion

$(1\ \bar{7}\ 4\ 5\ 3\ \bar{6}\ \bar{2})$



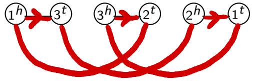
$(1\ 2\ 3\ 4\ 5\ 6\ 7)$

Types of cycles

Bad cycle: all adjacencies have the same direction

Cannot be split into two cycles

(1 3 2)

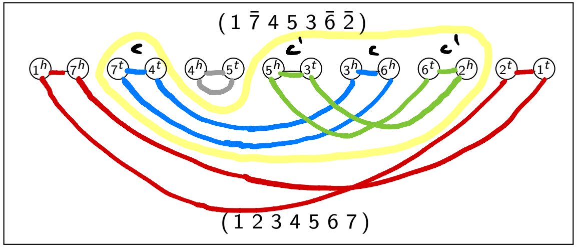


bad cycle

(1 2 3)

(Interleaving) components

Breakpoint diagram:



(1 $\bar{7}$ 4 5 3 $\bar{6}$ $\bar{2}$)
c
c'
c
c'
1^h 7^h 7^t 4^t 4^h 5^t 5^h 3^t 3^h 6^h 6^t 2^h 2^t 1^t
(1 2 3 4 5 6 7)

Two interleaving cycles: $c \dots c' \dots c \dots c'$ (crossing edges)

Interleaving sequence of cycles:

$c_1, c_2, \dots, c_k$ such that c_i and c_{i+1} are interleaving for all $1 \leq i \leq k-1$

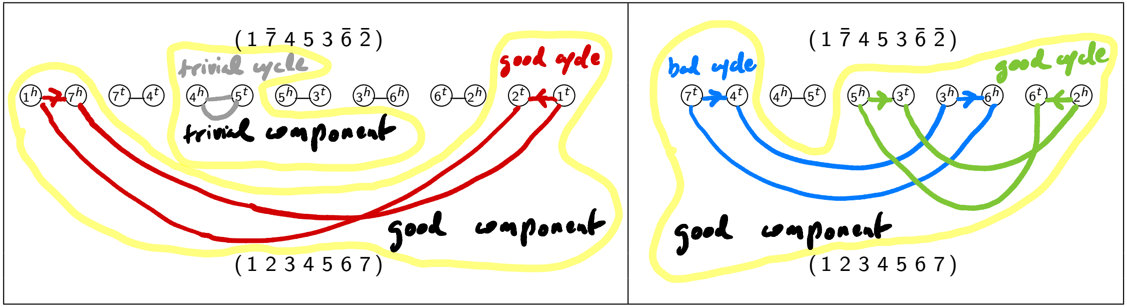
Interleaving component or simply **component** K:

$$\begin{cases} \text{either a cycle } c \text{ that does not interleave with any other cycle} \\ \text{or } \begin{cases} \text{for each pair of cycles } c, c' \in K \text{ there is an interleaving sequence from } c \text{ to } c' \\ K \text{ is maximal} \end{cases} \end{cases}$$

Types of (interleaving) components

Trivial component: only one trivial 2-cycle

Good component: at least one good cycle

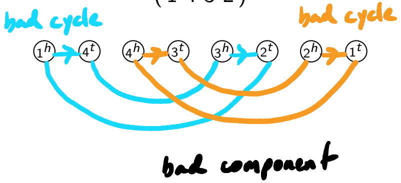


$(1\ \bar{7}\ 4\ 5\ 3\ \bar{6}\ \bar{2})$
trivial cycle
good cycle
1^h 7^h 7^t 4^t 4^h 5^t 5^h 3^t 3^h 6^h 6^t 2^h 2^t 1^t
trivial component
good component
$(1\ 2\ 3\ 4\ 5\ 6\ 7)$
$(1\ \bar{7}\ 4\ 5\ 3\ \bar{6}\ \bar{2})$
bad cycle
good cycle
7^t 4^t 4^h 5^t 5^h 3^t 3^h 6^h 6^t 2^h
good component
$(1\ 2\ 3\ 4\ 5\ 6\ 7)$

Types of (interleaving) components

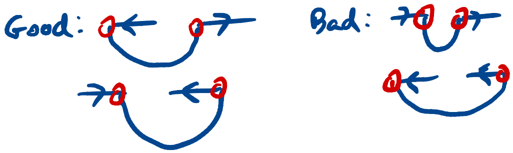
Bad component: only bad cycles

(1 4 3 2)

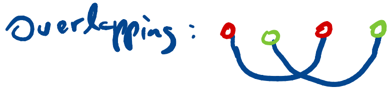


(1 2 3 4)

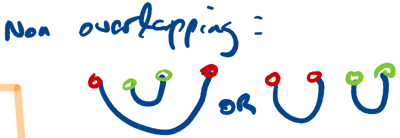
Overlap graph of a component

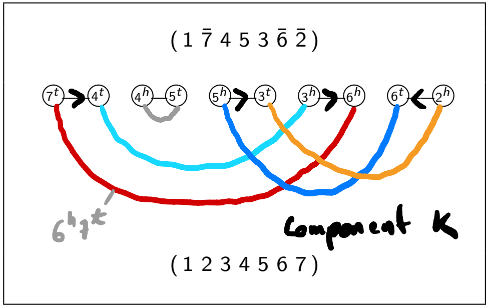


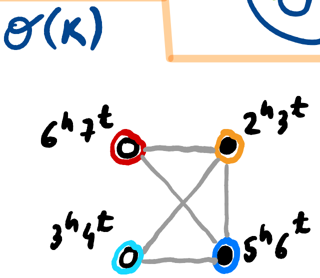
Target adjacency: $\begin{cases} \text{good: black vertex} \\ \text{bad: white vertex} \end{cases}$



Target adjacencies can be $\begin{cases} \text{overlapping: connected in the graph} \\ \text{non overlapping: disconnected in the graph} \end{cases}$

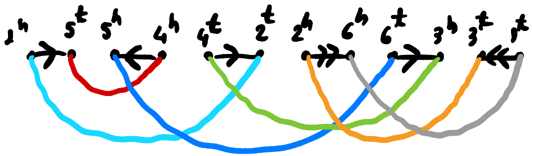






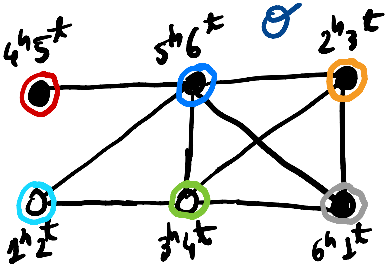
Another example

$(\ 1\ \ 5\ \ \bar{4}\ \ 2\ \ \bar{6}\ \ \bar{3}\ )$



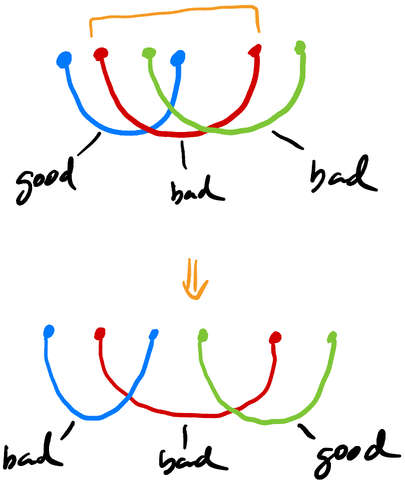
$(\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ )$

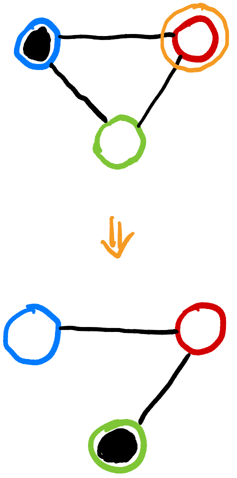
Overlap graph:



Effects on the overlap graph by inverting a bad adjacency

Three overlapping adjacencies :

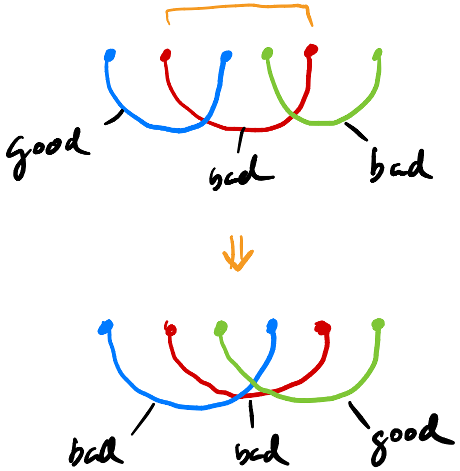


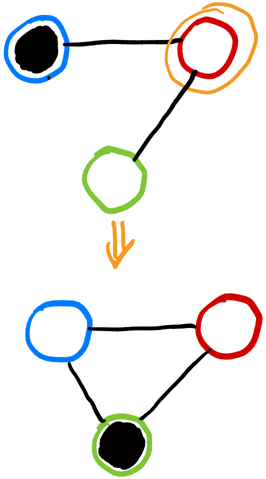


flip the types of the other two vertices + complement the edge between the other two vertices

Effects on the overlap graph by inverting a bad adjacency

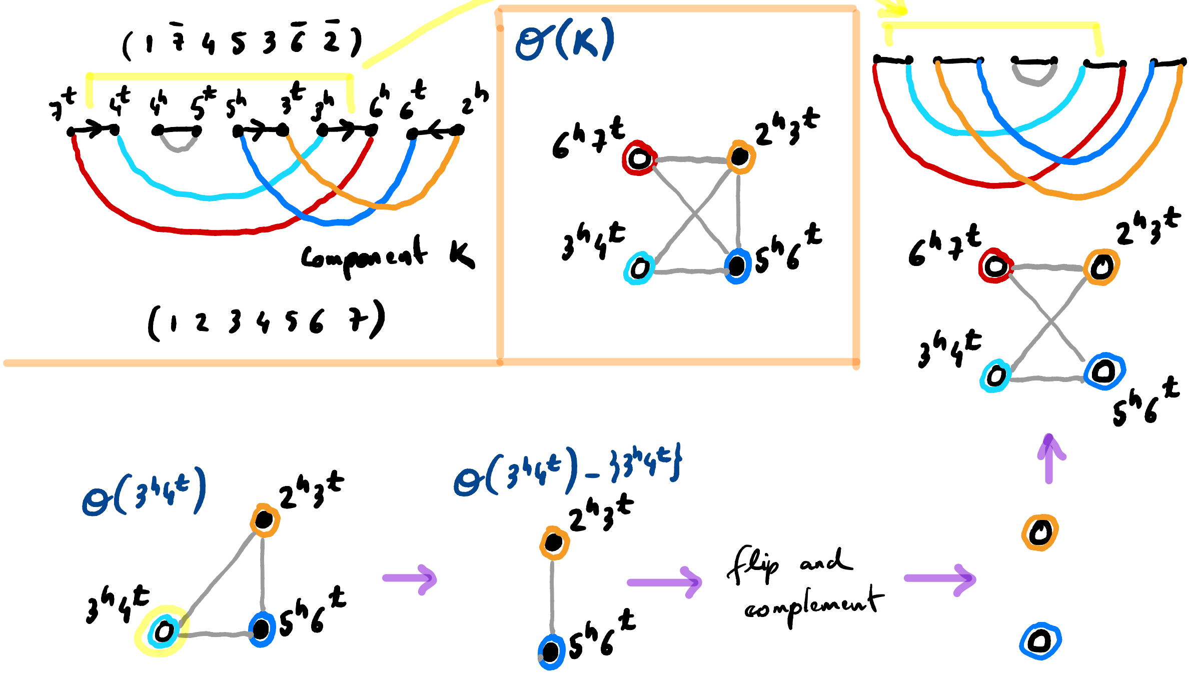
Three overlapping adjacencies :





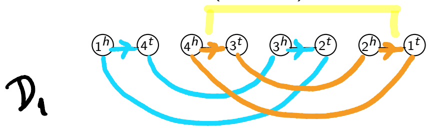
flip the types of the other two vertices + complement the edge between the other two vertices

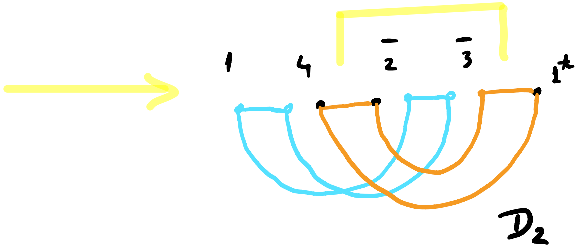
Effects on the overlap graph by inverting a bad adjacency


(1 $\bar{7}$ 4 5 3 $\bar{6}$ $\bar{2}$)
7^t 4^t 4^h 5^t 5^h 3^t 3^h 6^h 6^t 2^h
Component K
(1 2 3 4 5 6 7)
$\mathcal{O}(K)$
$6^h 7^t$
$2^h 3^t$
$3^h 4^t$
$5^h 6^t$
$\mathcal{O}(3^h 4^t)$
$\mathcal{O}(3^h 4^t) - \{3^h 4^t\}$
flip and complement

Sorting a bad component with a neutral inversion

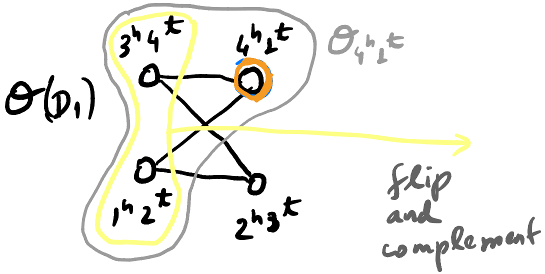
(1 4 3 2)

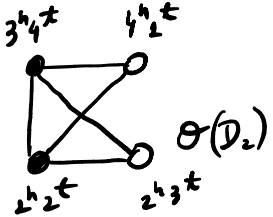




(1 2 3 4)

Overlap graph:





Sorting a bad component with a neutral inversion

Any neutral inversion applied to a bad adjacency of a bad component K turns K into a good component

b: # of bad components

Upper bound: one extra inversion per bad component:

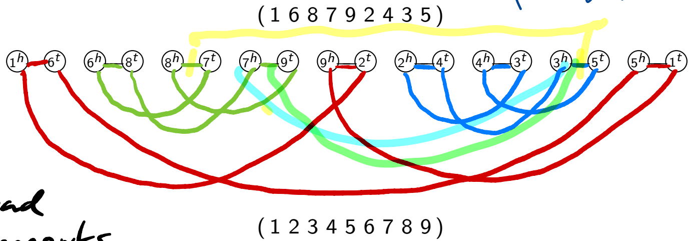
$$d_{inv}(\mathbb{A},\mathbb{B}) \leq n - |\mathcal{C}| + b$$

Sorting bad components with a joining inversion

K_1, K_2 and K_3 are three distinct components in $BD(\mathbb{A}, \mathbb{B})$ so that $K_3 \dots K_1 \dots K_1 \dots K_3 \dots K_2 \dots K_2$

$\Rightarrow$ K_3 **separates** K_1 and K_2

(Merging hurdles)



(1 6 8 7 9 2 4 3 5)
1^h 6^t 6^h 8^t 8^h 7^t 7^h 9^t 9^h 2^t 2^h 4^t 4^h 3^t 3^h 5^t 5^h 1^t
(1 2 3 4 5 6 7 8 9)

3 bad components

the red component separates the green and the blue components

$\Downarrow$

only green and blue are hurdles

By joining with an inversion two cycles c_1 and c_2, that belong to two distinct components K_1 and K_2 respectively, we merge not only the components K_1 and K_2, but also all components that separate K_1 and K_2 , into a single **good** component K.

Hurdle: a bad component that does not separate two bad components

Sorting bad components with a joining inversion

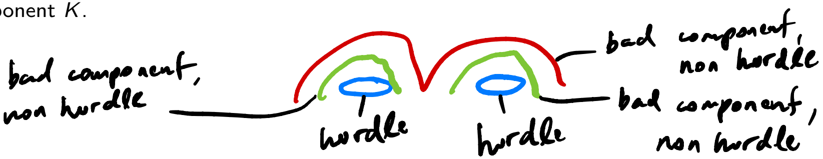
h: # of hurdles

One joining inversion "fixes" two hurdles

Lower bound: $d_{INV} \geq n - |C| + h$

By joining with an inversion two cycles c_1 and c_2, that belong to two distinct components K_1 and K_2 respectively, we merge not only the components K_1 and K_2, but also all components that separate K_1 and K_2, into a single **good** component K.

Another example:


bad component, non hurdle
hurdle
hurdle
bad component, non hurdle
bad component, non hurdle

Quiz 1

1 Which of the following statements about the breakpoint diagram are true?

~~A~~ A cycle can always be split into two cycles with an inversion.

~~B~~ A neutral inversion cannot be optimal.

~~C~~ A joining inversion cannot be optimal.

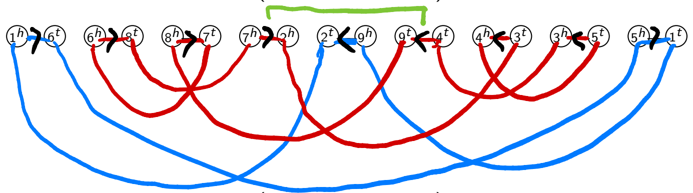
[✓ D] It is always possible to split a good cycle into two.

[✓ E] A bad cycle cannot be split by an inversion.

Unsafe inversions

A split inversion applied to a good cycle of a good component can create bad components

$(1\ 6\ 8\ 7\ \bar{2}\ \bar{9}\ 4\ 3\ 5)$



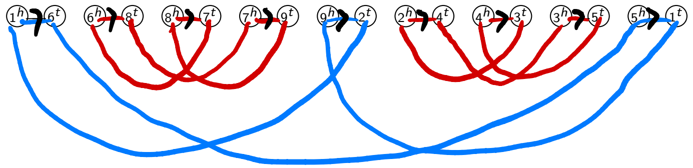
1^h 6^t 6^h 8^t 8^h 7^t 7^h 2^h 2^t 9^h 9^t 4^t 4^h 3^t 3^h 5^t 5^h 1^t

$(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$

– One good component

$\Downarrow$

$(1\ 6\ 8\ 7\ 9\ 2\ 4\ 3\ 5)$



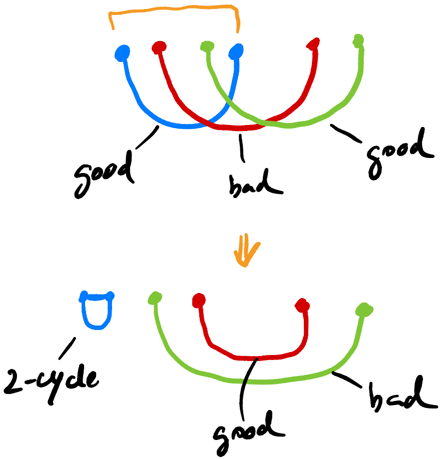
1^h 6^t 6^h 8^t 8^h 7^t 7^h 9^t 9^h 2^t 2^h 4^t 4^h 3^t 3^h 5^t 5^h 1^t

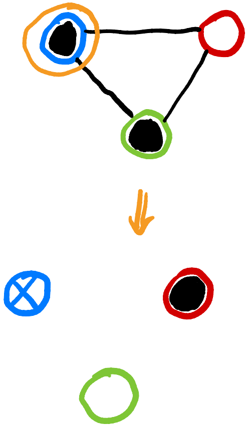
$(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$

– 3 bad components

Effects on the overlap graph by inverting a good adjacency

Three overlapping adjacencies:

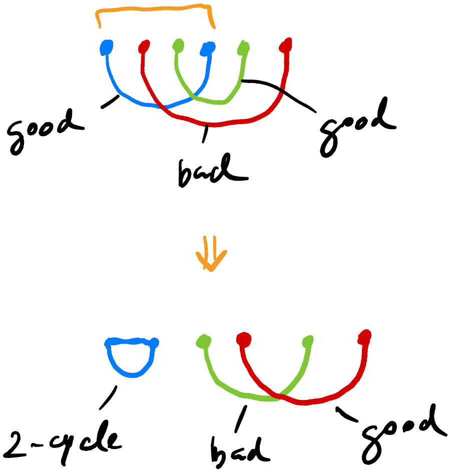


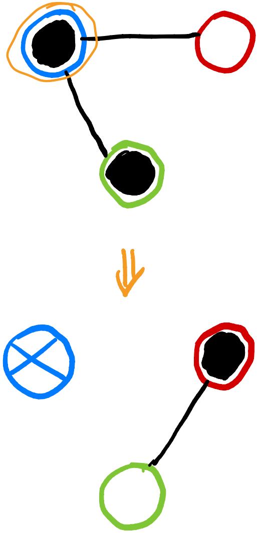


flip the types of the vertices + complement the edges

Effects on the overlap graph by inverting a good adjacency

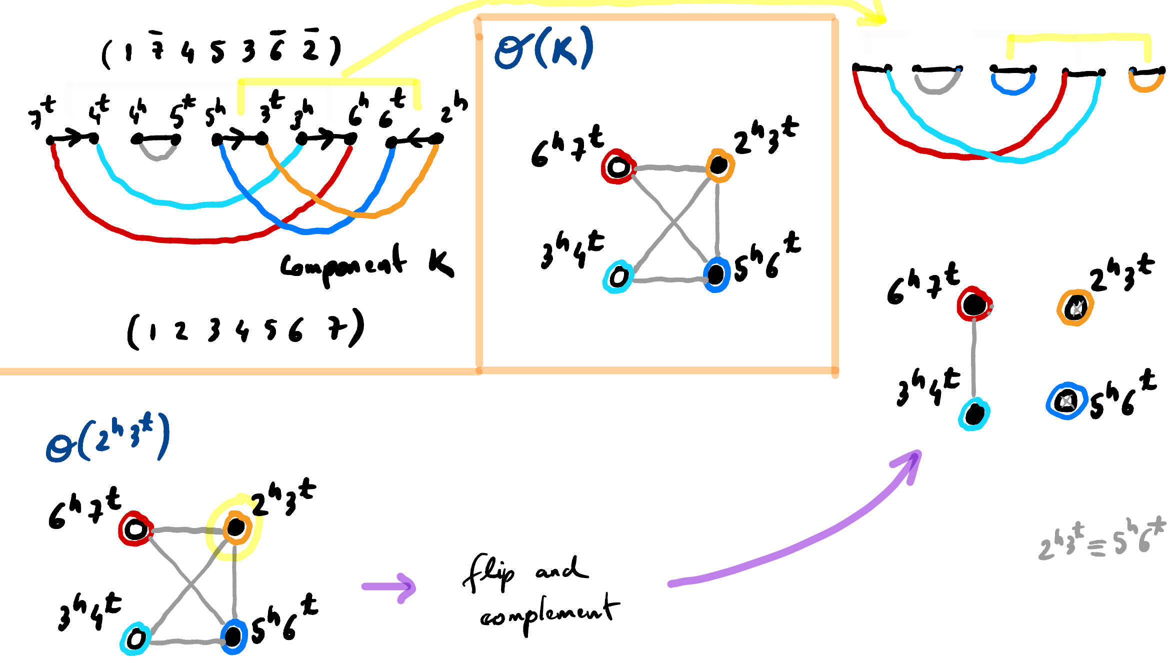
Three overlapping adjacencies :





flip the types of the vertices + complement the edges

Effects on the overlap graph by inverting a good adjacency



Sorting a good component - finding safe split inversions

G: # of good adjacencies in $BD(\mathbb{A},\mathbb{B})$

$g(xy)$: # of good adjacencies overlapping xy in $BD(\mathbb{A},\mathbb{B})$

$b(xy)$: # of bad adjacencies overlapping xy in $BD(\mathbb{A},\mathbb{B})$

score(xy): # good adjacencies in the diagram <u>after</u> fixing xy

$$\boxed{\text{score}(xy) = G + b(xy) - g(xy) - 1}$$

An inversion that fixes a good target adjacency with maximal score is <u>SAFE</u>. (does not create new bad components)

Sorting a good component - finding safe split inversions

Let Θ be the overlap graph of component K

Suppose $\left\{ \begin{array}{l} xy \text{ is a good adjacency with maximal score in } \Theta \\ \text{inversion fixing } xy \text{ creates a bad component } K_B \end{array} \right.$

At least one (bad) adjacency $zw \in K_B$ was adjacent to xy in Θ $\Rightarrow$ zw was good in Θ

Scores in Θ $\left\{ \begin{array}{l} \text{score}(xy) = G + b(xy) - g(xy) - 1 \\ \text{score}(zw) = G + b(zw) - g(zw) - 1 \end{array} \right.$

Sorting a good component - finding safe split inversions

set of bad target adjacencies connected to xy in Θ

But : $B(xy) \subseteq B(zw) \Rightarrow b(xy) \leq b(zw)$

$G(zw) \subseteq G(xy) \Rightarrow g(zw) \leq g(xy)$

$\Downarrow$

we cannot have $\begin{cases} b(xy) = b(zw) \text{ and} \\ g(zw) = g(xy) \end{cases}$

$\Downarrow$

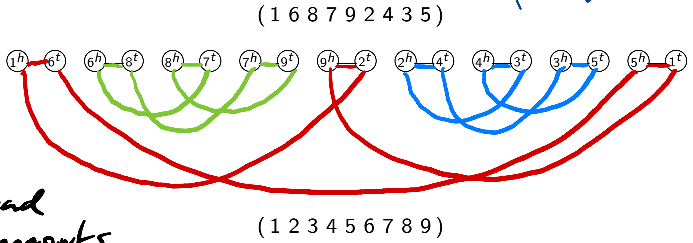
$\text{score}(zw) > \text{Score}(xy)$

Sorting bad components with a joining inversion

K_1, K_2 and K_3 are three distinct components in $BD(\mathbb{A}, \mathbb{B})$ so that $K_3 \dots K_1 \dots K_1 \dots K_3 \dots K_2 \dots K_2$

$\Rightarrow$ K_3 **separates** K_1 and K_2

(Merging hurdles)



(1 6 8 7 9 2 4 3 5)
1^h 6^t 6^h 8^t 8^h 7^t 7^h 9^t 9^h 2^t 2^h 4^t 4^h 3^t 3^h 5^t 5^h 1^t
(1 2 3 4 5 6 7 8 9)

3 bad components

the red component separates the green and the blue components

$\Downarrow$

only green and blue are hurdles

By joining with an inversion two cycles c_1 and c_2, that belong to two distinct components K_1 and K_2 respectively, we merge not only the components K_1 and K_2, but also all components that separate K_1 and K_2 , into a single **good** component K.

Hurdle: a bad component that does not separate two bad components

Sorting bad components - simple hurdles and super hurdles

h : number of hurdles in $BD(\mathbb{A}, \mathbb{B})$

hurdle : bad component that does not separate 2 bad components

super hurdle K : fixing K by a neutral inversion creates a new hurdle

On the previous page, both green and blue are super hurdles
fixing only the green or only the blue component with a neutral inversion would turn the red component into a hurdle

Sorting bad components - simple hurdles and super hurdles

<u>simple hurdle</u>: a hurdle that is not a super hurdle

* each simple hurdle can be fixed with a neutral inversion

(cutting a hurdle) $\Delta_{INV} = +1$

* each pair of super hurdles can be fixed with a joining inversion

(merging hurdles) $\Delta_{INV} = +2$

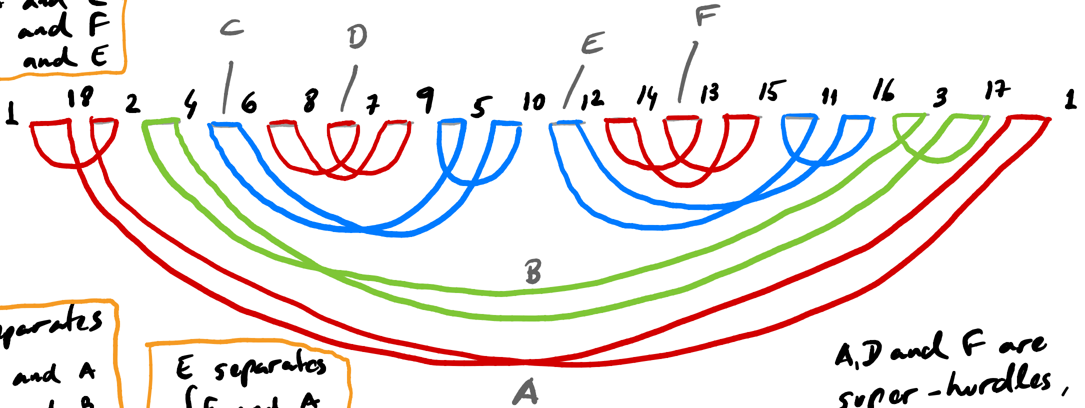
Sorting bad components - fortress

A, B, C, D, E, F are bad components

fortress $\begin{cases} \text{* there is an odd number of hurdles} \\ \text{* all hurdles are super hurdles} \end{cases}$

$$f : \begin{cases} 0 & BD(\mathbb{A}, \mathbb{B}) \text{ is not a fortress} \\ 1 & BD(\mathbb{A}, \mathbb{B}) \text{ is a fortress} \end{cases}$$

B separates
- A and D
- A and C
- A and F
- A and E



C separates
- D and A
- D and B
- D and E
- D and F

E separates
- F and A
- F and B
- F and C
- F and D

⇒ only A, D and F are hurdles

A, D and F are super-hurdles, but joining any pair among A, D, F creates a new hurdle

Canonical inversion distance of circular chromosomes

$$\boxed{\mathrm{d}_{\mathrm{INV}}(\mathbb{A}, \mathbb{B}) = n - |\mathcal{C}| + h + \underline{f}}$$

$$f : \left\{ \begin{array}{l} 0 \\ 1 \end{array} \right.$$

Quiz 2

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 breakpoint diagram.

~~B~~ Every bad component in the diagram is a hurdle.

~~C~~ A split inversion is always optimal.

[✓ D] A good component can always be sorted with (safe) split inversions.

~~E~~ A super hurdle can be optimally sorted with a neutral inversion. true if BD is a fortress

[✓ F] A diagram with an even number of bad components can be a fortress.

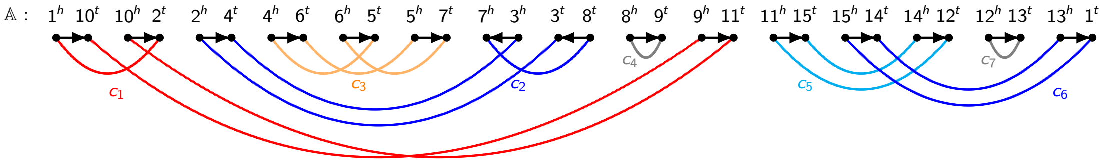
Chained and nested components on the breakpoint diagram

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

Chain: $\begin{cases} \text{sequence of components } K_1, K_2, \ldots, K_\ell \\ \text{the rightmost adjacency-edge of } K_i \text{ is succeeded by} \\ \text{the leftmost adjacency-edge of } K_{i+1}, \text{ for } 1 \leq i \leq \ell \end{cases}$

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.

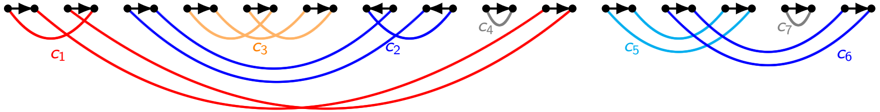


$\mathbb{A}$:
1^h 10^t 10^h 2^t 2^h 4^t 4^h 6^t 6^h 5^t 5^h 7^t 7^h 3^h 3^t 8^t 8^h 9^t 9^h 11^t 11^h 15^t 15^h 14^t 14^h 12^t 12^h 13^t 13^h 1^t
c_1
c_2
c_3
c_4
c_5
c_6
c_7

$K_1 = \{c_1\}$ $K_2 = \{c_2\}$ $K_3 = \{c_3\}$ $K_4 = \{c_4\}$ $K_5 = \{c_5, c_6\}$ $K_6 = \{c_7\}$

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

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

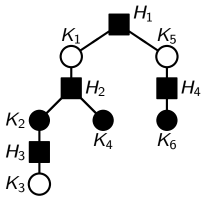


$K_1 = \{c_1\}$ $K_2 = \{c_2\}$ $K_3 = \{c_3\}$ $K_4 = \{c_4\}$ $K_5 = \{c_5, c_6\}$ $K_6 = \{c_7\}$

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

1. One round node per component K_i: $\begin{cases} \text{bad node } (\circ)\text{: } K_i \text{ is a bad component;} \\ \text{good node } (\bullet)\text{: } K_i \text{ is a trivial or a good component.} \end{cases}$
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.

$\Upsilon_{\blacksquare}$:



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 $RG(\mathbb{A}, \mathbb{B})$.

Contraction of $\Upsilon_{\blacksquare}$ into unrooted component tree $\Upsilon_{\circ}$

Max-flower:

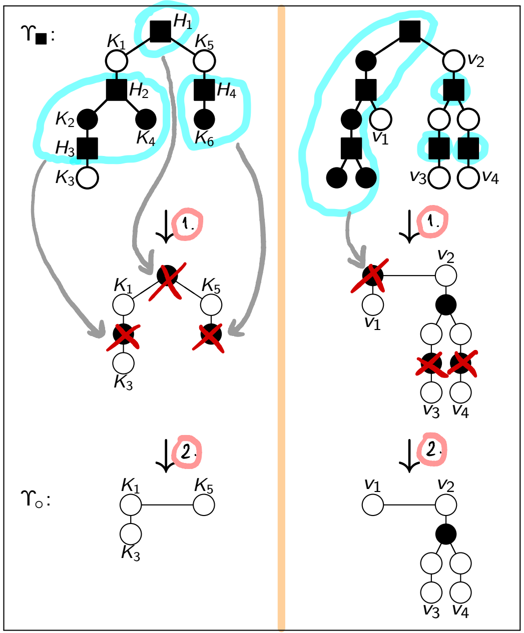
maximal connected subgraph of $\Upsilon_{\blacksquare}$
composed of good and/or square nodes only

Obtaining $\Upsilon_{\circ}$ from $\Upsilon_{\blacksquare}$

For each max-flower F of $\Upsilon_{\blacksquare}$:

1. Replace F by a single good round node g
(g is connected to all bad nodes connected to F)

2.
- If g has exactly two neighbors b_1 and b_2: remove g from the tree and connect b_1 to b_2;
- If g is a leaf: simply remove g from the tree
  ($\Rightarrow$ in the end, all leaves in $\Upsilon_{\circ}$ are bad)

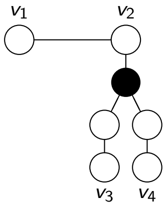


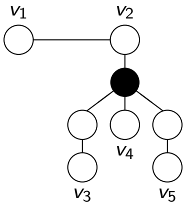
Topology and paths in the component tree $\Upsilon_\circ$

All leaves in $\Upsilon_\circ$ are bad nodes ($\equiv$ hurdles)

$\mathcal{L}$: # of leaves in $\Upsilon_\circ$

Traversal: path connecting two leaves of $\Upsilon_\circ$





Branching node of $\Upsilon_\circ$: any node whose degree is ≥ 3

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

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

Cost of covering the component tree $\Upsilon_\circ$

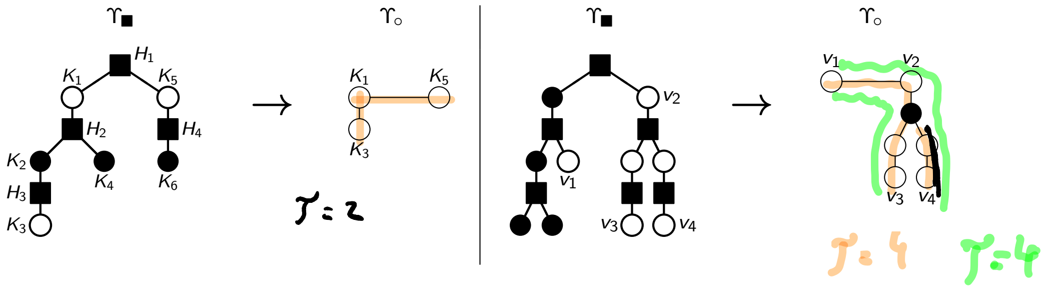
path P in $\Upsilon_\circ$: $\begin{cases} \textbf{short:} \text{ contains a single bad node} & \Rightarrow \text{if } P \text{ is a leaf-branch it corresponds to a simple hurdle} \\ \textbf{long:} \text{ contains at least two bad nodes} & \Rightarrow \text{if } P \text{ is a leaf-branch it corresponds to a super hurdle} \end{cases}$

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

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

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

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



$$\tau(\Upsilon_\blacksquare) = \tau(\Upsilon_\circ)$$

Covering the component tree $\Upsilon_\circ$

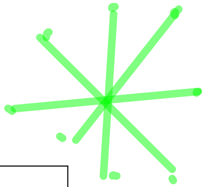
$\mathcal{L} = n$

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

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

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

Traversal: path connecting two leaves of $\Upsilon_\circ$



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

$\widehat{\mathcal{P}}_{\mathrm{T}}(\Upsilon_\circ)$: smallest set of traversals covering all nodes of $\Upsilon_\circ$: $|\widehat{\mathcal{P}}_{\mathrm{T}}(\Upsilon_\circ)| = \frac{\mathcal{L}}{2}$

```
CoverTreeWithTraversals
Input: unrooted tree $\Upsilon_\circ$ with $\mathcal{L} = 2n$ leaves
Output: set $\widehat{\mathcal{P}}_{\mathrm{T}}$ of $n$ traversals covering all nodes of $\Upsilon_\circ$

  Based on any planar view of $\Upsilon_\circ$, enumerate the leaves from 1 to $2n$ in circular order;
  $\widehat{\mathcal{P}}_{\mathrm{T}} = \emptyset$;
  for $i = 1$ to $n$ do
      $\widehat{\mathcal{P}}_{\mathrm{T}} = \widehat{\mathcal{P}}_{\mathrm{T}} \cup \{$traversal connecting leaves $i$ and $i + n\}$;
  Return $\widehat{\mathcal{P}}_{\mathrm{T}}$;
```

Computing $\tau(\Upsilon_\circ)$

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

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

If $\mathcal{L}$ is even, $\widehat{\mathcal{P}}_{\mathrm{T}}(\Upsilon_\circ)$ is an optimal cover:

$$\Rightarrow \tau(\Upsilon_\circ) = \tau\left(\widehat{\mathcal{P}}_{\mathrm{T}}(\Upsilon_\circ)\right) = 2\tfrac{\mathcal{L}}{2} = \mathcal{L}$$

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

$$\Rightarrow \tau(\Upsilon_\circ) = \tau\left(\widehat{\mathcal{P}}_{\mathrm{T}}(\Upsilon_\circ \setminus s)\right) + \tau(s) = 2\tfrac{\mathcal{L}-1}{2} + 1 = \mathcal{L}$$

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

$$\Rightarrow \tau(\Upsilon_\circ) = \tau\left(\widehat{\mathcal{P}}_{\mathrm{T}}(\Upsilon_\circ \setminus \ell)\right) + \tau(\ell) = 2\tfrac{\mathcal{L}-1}{2} + 2 = \mathcal{L} + 1$$

The cost of any optimal cover of $\Upsilon_\circ$ is:

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

Canonical inversion distance

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

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

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

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

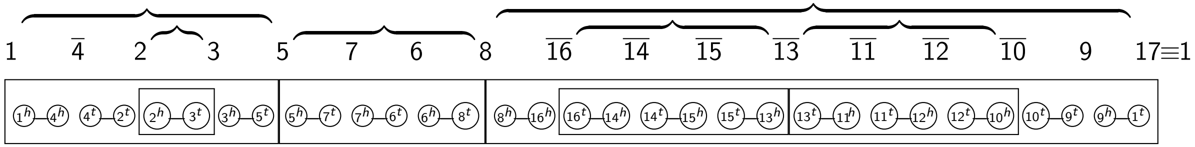
Conserved interval: interval of $\mathbb{A}$ composed of values $i, i+1, \dots, i+j$ (assuming $n+1 \equiv 1$)

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

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

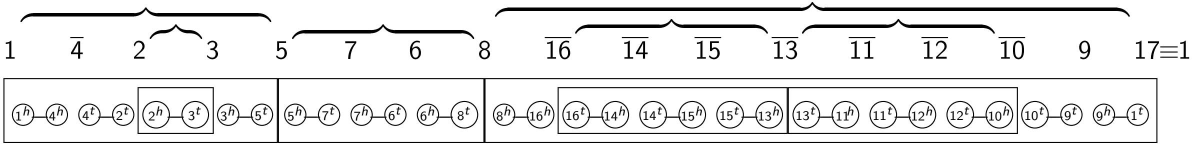
Component: framed conserved interval that is not a union of framed conserved intervals

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

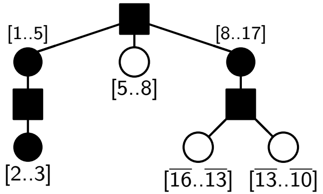


1
4̄
2
3
5
7
6
8
16̄
14̄
15̄
13̄
11̄
12̄
10̄
9
17≡1
1h 4h 4t 2t 2h 3t 3h 5t 5h 7t 7h 6t 6h 8t 8h 16h 16t 14h 14t 15h 15t 13h 13t 11h 11t 12h 12t 10h 10t 9t 9h 1t

Components are framed conserved intervals



1 $\overline{4}$ 2 3 5 7 6 8 $\overline{16}$ $\overline{14}$ $\overline{15}$ $\overline{13}$ $\overline{11}$ $\overline{12}$ $\overline{10}$ 9 17≡1
1^h 4^h 4^t 2^t 2^h 3^t 3^h 5^t 5^h 7^t 7^h 6^t 6^h 8^t 8^h 16^h 16^t 14^h 14^t 15^h 15^t 13^h 13^t 11^h 11^t 12^h 12^t 10^h 10^t 9^t 9^h 1^t



[1..5]
[8..17]
[5..8]
[2..3]
[$\overline{16}$..$\overline{13}$]
[$\overline{13}$..$\overline{10}$]

Complexity of inversion distance and sorting

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 $\mathbb{B}$:

$\mathbb{B} = (0\ 1\ 2\ 3 \ldots\ n)$

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

$$\mathrm{d}_{\mathrm{INV}}(\mathbb{A}, \mathbb{B}) = \min \begin{cases} \mathrm{d}_{\mathrm{INV}}((0\ \mathbb{A}), \mathbb{B}') \\ \mathrm{d}_{\mathrm{INV}}((\bar{0}\ \mathbb{A}), \mathbb{B}') \end{cases}$$

Example:

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

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

$$\mathrm{d}_{\mathrm{INV}}((0\ \bar{5}\ 1\ 2\ \bar{3}\ 4), \mathbb{B}') = 3$$

$$\mathrm{d}_{\mathrm{INV}}((\bar{0}\ \bar{5}\ 1\ 2\ \bar{3}\ 4), \mathbb{B}') = 2$$

$$\mathrm{d}_{\mathrm{INV}}(\mathbb{A}, \mathbb{B}) = 2$$

Quiz 3

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.

References

Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals

Sridhar Hannenhalli and Pavel A. Pevzner

Journal of the ACM, vol 46, issue 1, pages 1–27 (1999)

Reversal Distance without Hurdles and Fortresses

(Anne Bergeron, Julia Mixtacki and Jens Stoye)

LNCS, volume 3109, pages 388-399 (2004)

The Inversion Distance Problem

(Anne Bergeron, Julia Mixtacki and Jens Stoye)

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