Algorithms for Genome Rearrangements

Pedro Feijão

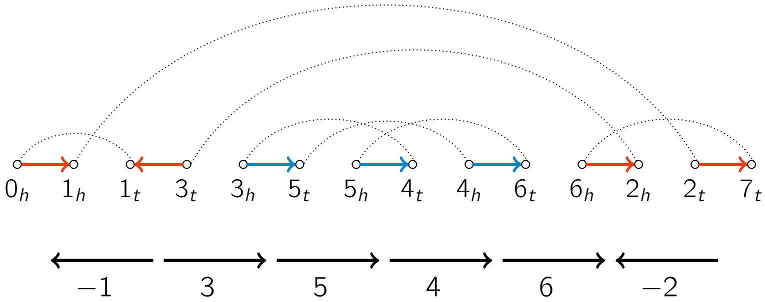
Lecture 4 – Sorting by Signed Reversals II

Summer 2014

pfeijao@cebitec.uni-bielefeld.de

Quick Recap

BP Graph, **oriented** and **unoriented** components:



Quick Recap

- Sorting is equivalent of increasing # of cycles in BP graph
  - In *oriented (good) components – at least 1 oriented edge* – this is always possible (safe reversals).
  - In *unoriented (bad) components*, we will need **extra operations**.

- If there are only oriented components in the BP graph:

$$d = N - C$$

- If there are also unoriented components:

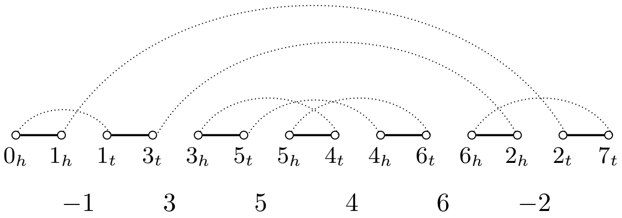
$$d = N - C + k,$$

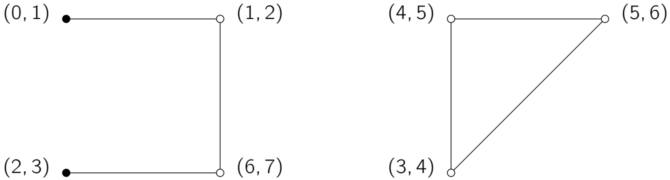
where k is the minimum cost of these *extra* operations.

Overlap Graph $O(\pi)$

- Each vertex in $O(\pi)$ corresponds to a gray edge in $BP(\pi)$.
  - Oriented edges $\rightarrow$ Black vertices (odd degree).
  - Unoriented edges $\rightarrow$ White vertices (even degree).
- If two edges in $BP(\pi)$ overlap, there is an edge in $O(\pi)$ between the corresponding vertices.

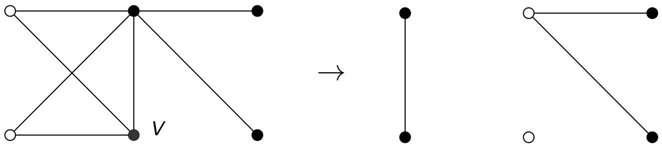
Overlap Graph $O(\pi)$





Effect of Reversal in the Overlap Graph

1. The subgraph induced by v and its neighbours is **complemented**.
2. All neighbours of v have their orientation inverted.

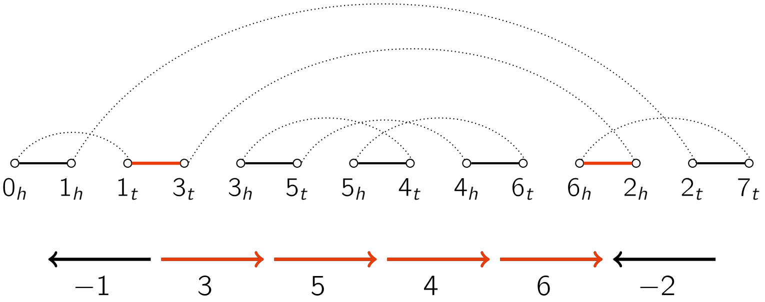


- The **score** of a reversal induced by v is

$$s(v) = T + U - O - 1$$

Sorting Unoriented Components

- Let' s analyse the effect that reversals have on cycles of $BP(\pi)$.
- Reversals change # of cycles by -1, 0, or $+1$.
- What happens exactly when we apply a reversal **defined** by two black edges?



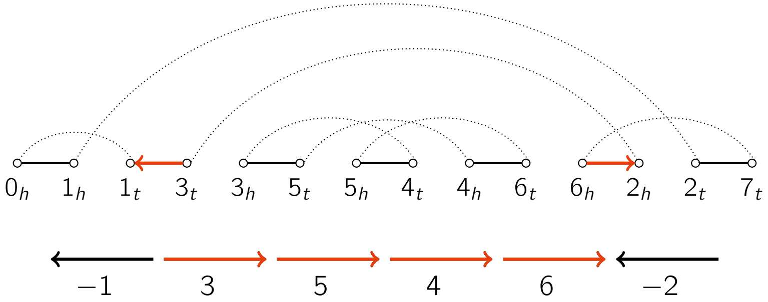

```
java InversionVisualisation L2/recap1.txt
```

Reversals and effect on cycles

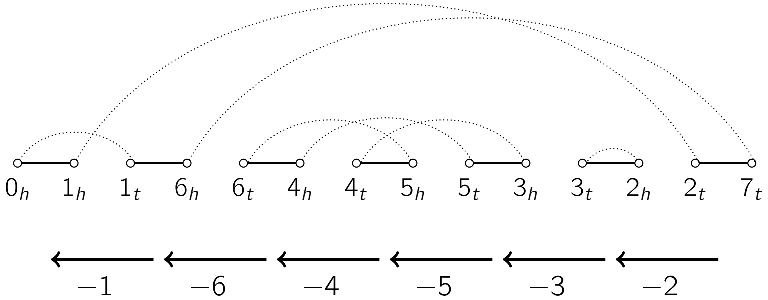
1. Edges are on the **same cycle**:
   - **Type I**: Divergent edges: breaks the cycle. $\Delta C = +1$.
   - **Type II**: Convergent edges: $\Delta C = 0$, may change cycle orientation.
2. Edges on **different cycles**:
   - **Type III**: Merges the two cycles. $\Delta C = -1$.

So far, we only used **Type I** operations, to sort oriented components.

Type I - Same Cycle, divergent

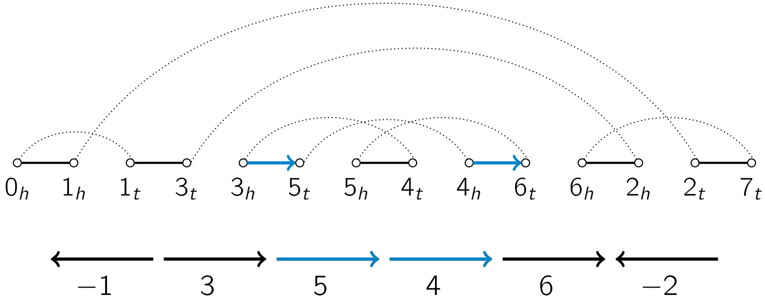


Type I - Same Cycle, divergent

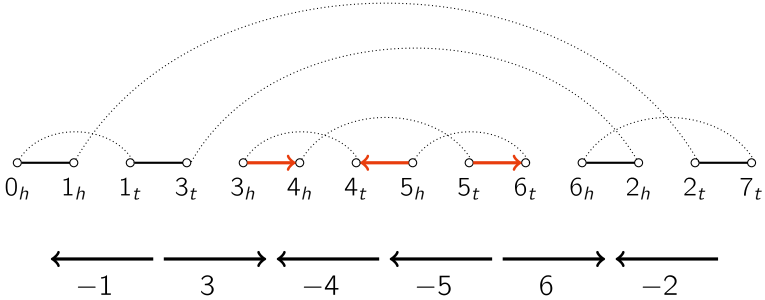


This reversal increases the number of cycles by one, $\Delta C = +1$.

Type II - Same Cycle, convergent

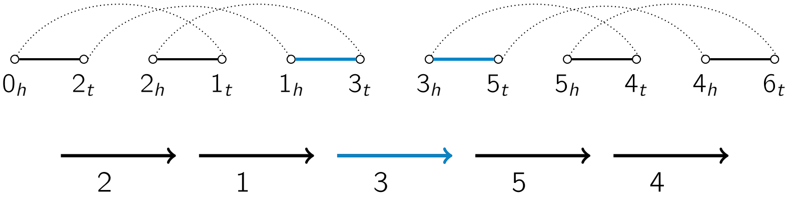


Type II - Same Cycle, convergent

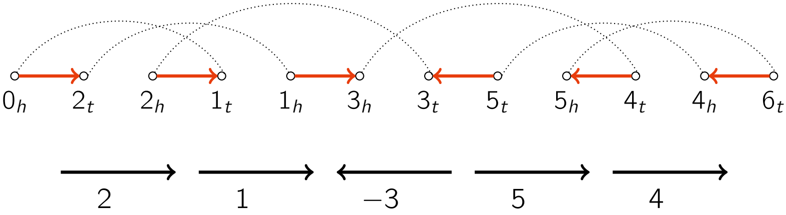


Does not change number of cycles ($\Delta C = 0$), but the cycle is **oriented**.

Type III - Different Cycles



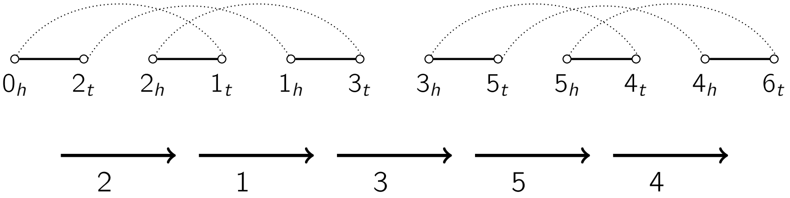
Type III - Different Cycles



Merges the two cycles, decreasing the number of cycles by one ($\Delta C = -1$), but the new cycle is **oriented**.

Extra Operations

- How many extra operations do we need to sort unoriented components?



`java InversionVisualisation L2/2unoriented.txt`

Extra Operations

- Applying one reversal in each cycle, orients both cycles, with 2 extra operations:

$$d = N - C + 2$$

- Applying one reversal merging both cycles, creates one new **oriented** cycle. Only one operation, but also one less cycle:

$$d = N - (C - 1) + 1 = N - C + 2$$

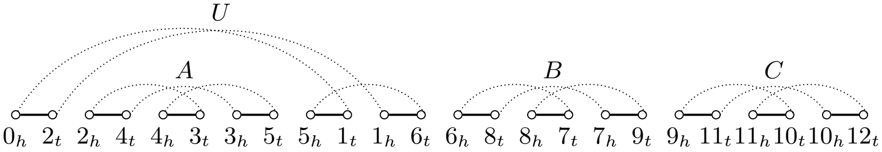
- In both cases, 2 extra operations. Does this mean that

$$d = N - C + K$$

  where K is the number of unoriented components? **Almost...**

Definitions

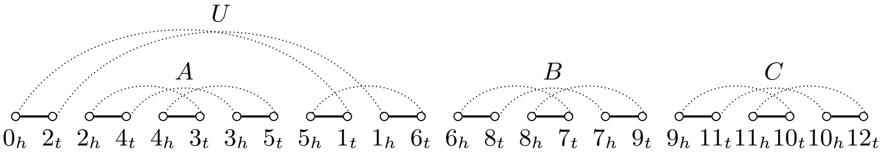
- A Component U **separates** two other components A and B if any edge from a vertex from A to B would cross an edge of U.



- U separates A and B. (Also A and C).

Definitions

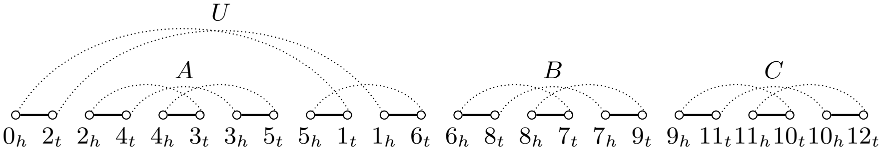
- A **hurdle** is an unoriented component that does **not** separate other two unoriented components.



- A,B and C are hurdles.

Definitions

- A **super-hurdle** is a hurdle that, when removed, causes the creation of a new hurdle.



- *A* is a super-hurdle. *B* and *C* are called *simple* hurdles.

- Why are these definitions important? Because except for one very rare special case, we have

$$d = N - C + H$$

where H is the number of hurdles.

BP Graph – Component Types

- Component
  - Oriented
  - Unoriented
    - Non-Hurdle
    - Hurdle
      - Simple
      - Super-Hurdle

Reversal Types

- **Type I**: **Oriented Reversal**: $\Delta C = +1$.
  - Edges on same cycle, divergent.
- **Type II**: **Hurdle Cutting**: $\Delta C = 0$, $\Delta H = -1$.
  - Edges on same cycle (hurdle), convergent.
- **Type III**: **Hurdle Merging**: $\Delta C = -1$, $\Delta H = -2$.
  - Edges on different cycles (hurdles).

Separating component

- Why a separating component is not a Hurdle?
- Because it can be oriented by a **Hurdle Merging** of two hurdles that is separates.

```
java InversionVisualisation L2/sep-hur-example.txt
```

Super-hurdles: Problems might occur

- Cutting a super-hurdle is bad.
- Merging hurdles that are separated from a super-hurdle can cause the separating component to become a hurdle.

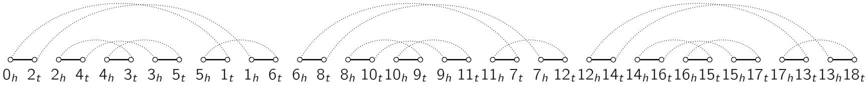
```
java InversionVisualisation L2/sep-hur-example.txt
```

Super-hurdles: Problems might occur

- How to avoid those problems?
- When there is an odd # of hurdles, cut a simple hurdle.
- When there is an even # of hurdles, merge opposite hurdles.
- Can we always do that? No... meet the **fortress**!

Fortresses

- A **fortress** is a permutation that has an odd number of hurdles, and all are super-hurdles.



In this kind of permutation, there is no way to avoid an **extra operation**, a hurdle cut that creates a new hurdle.

```
java InversionVisualisation L2/fortress.txt
```

Reversal Distance - Complete equation

Theorem (Reversal Distance, HP 95)

The reversal distance of a permutation π is given by

$$d(\pi) = N - C + H + F$$

where:

- *N is the number of genes*
- *C is the number of cycles in $BP(\pi)$*
- *H the number of hurdles in $BP(\pi)$*
- $F = \begin{cases} 1, & \pi \text{ is a fortress} \\ 0, & \text{otherwise} \end{cases}$

Reversal Distance - Complete Algorithm

procedure ReversalSort(π)
    while $\pi \neq$ identity **do**
        if $\exists$ oriented component in $BP(\pi)$ **then**
            $\rightarrow$ *Apply a max score oriented reversal – Type I*
        else if even # of hurdles **then**
            $\rightarrow$ *Apply a Hurdle Merging on opposite hurdles – Type III*
        else if $\exists$ simple hurdle **then**
            $\rightarrow$ *Apply a Hurdle Cutting on a simple hurdle – Type II*
        else
            $\rightarrow$ *Merge any two super hurdles* (Fortress)
        end if
    end while
end procedure