# Common intervals of genomes

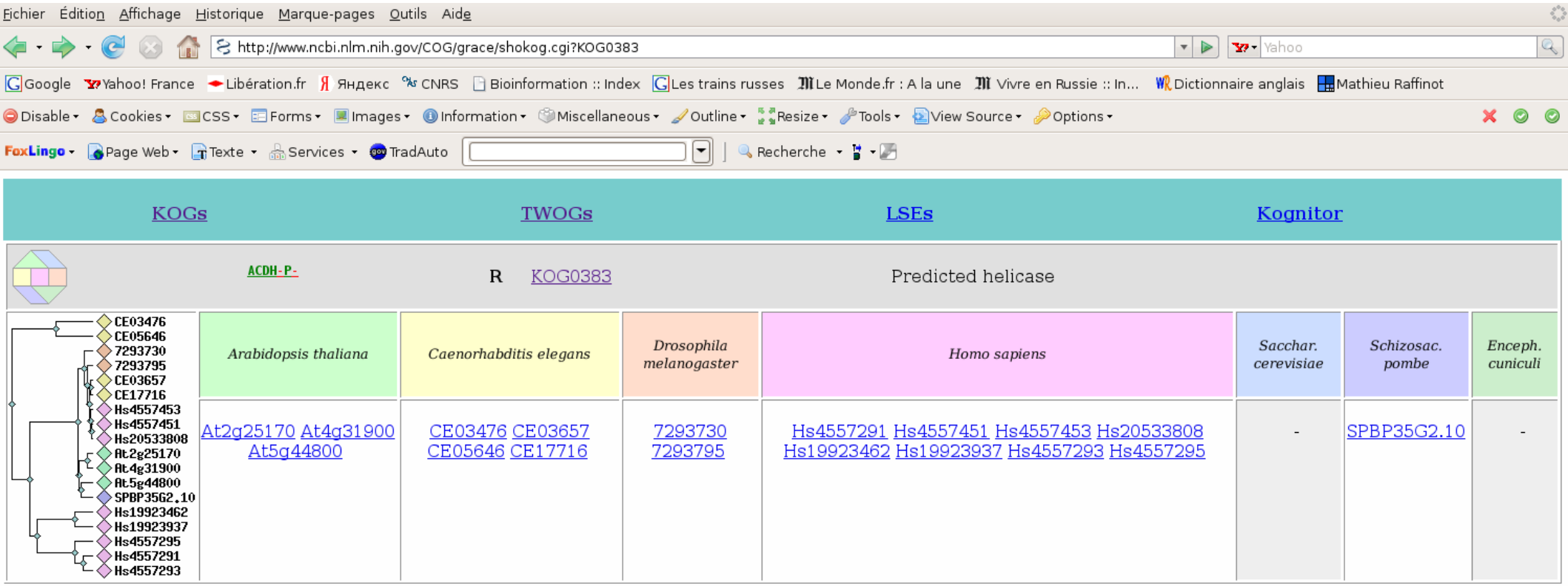
Mathieu Raffinot
CNRS - LIAFA

## Context:

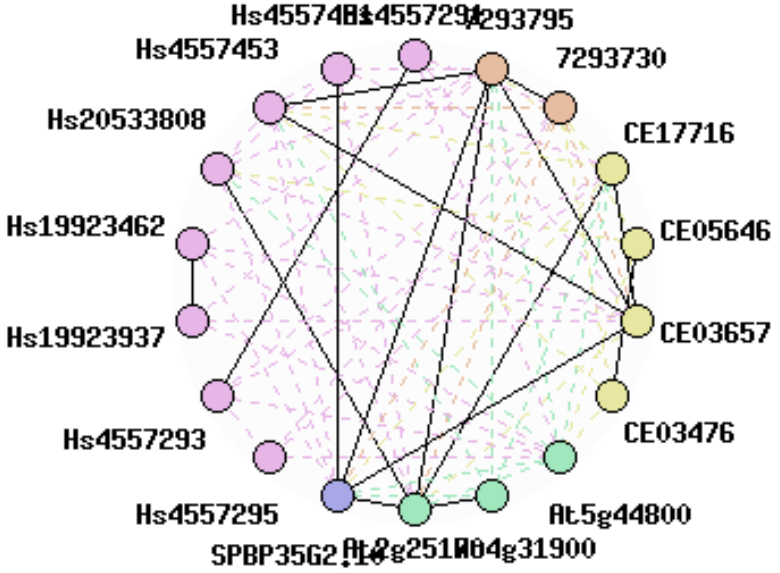
- comparative genomics.
- set of genomes partially/totally annotated

→ Informative group of genes or domains ?

Ex: COG database



# BeTs for KOG0383:




| | | | | | | |
|---|---|---|---|---|---|---|
| [ At2g25170] | CE17716 | 7293795 | Hs20533808 | YER164w | SPAC3G6.01 | ECU01g0350 |
| [ At4g31900] | CE17716 | 7293795 | Hs4557453 | YER164w | SPAC3G6.01 | ECU10g1320 |
| [ At5g44800] | CE17716 | 7293795 | Hs20533808 | YER164w | SPAC3G6.01 | ECU01g0350 |
| At2g36720 | [ CE03476] | 7293618 | Hs4557453 | YMR075w | SPAC16C9.05 | - |
| At2g25170 | [ CE03657] | 7293795 | Hs4557453 | YER164w | SPAC3G6.01 | ECU01g0350 |
| At3g14980 | [ CE05646] | 7293730 | Hs20533808 | YMR075w | SPAC2F7.07c | - |
| At2g25170 | [ CE17716] | 7293795 | Hs4557453 | YER164w | SPAC3G6.01 | ECU01g0350 |
| At2g25170 | CE03657 | [ 7293730] | Hs4557453 | YER164w | SPAC1783.05 | ECU01g0350 |
| At2g25170 | CE03657 | [ 7293795] | Hs4557453 | YER164w | SPAC3G6.01 | ECU01g0350 |
| At5g44800 | CE03657 | 7293730 | [ Hs4557291] | YMR075w | SPAC16C9.05 | - |
| At2g25170 | CE17716 | 7293795 | [ Hs4557451] | YER164w | SPAC3G6.01 | ECU01g0350 |
| At2g25170 | CE03657 | 7293795 | [ Hs4557453] | YER164w | SPAC3G6.01 | ECU01g0350 |
| At2g25170 | CE03657 | 7293795 | [ Hs20533808] | YER164w | SPAC3G6.01 | ECU01g0350 |
| At5g44800 | CE03657 | 7293795 | [ Hs19923462] | YMR075w | SPAC2F7.07c | ECU08g0060 |
| At5g44800 | CE03657 | 7293795 | [ Hs19923937] | YMR075w | SPAC2F7.07c | - |
| At5g44800 | CE17716 | 7293730 | [ Hs4557293] | YMR075w | SPAC16C9.05 | - |
| At5g44800 | CE17716 | 7293730 | [ Hs4557295] | YMR075w | SPAC16C9.05 | - |
| At2g25170 | CE03657 | 7293795 | Hs4557451 | YER164w | [ SPBP35G2.10] | ECU01g0350 |

# Many difficulties !

Biology

What are two similar genes ? What about alternative splicing ?

When are two genes close (notion of distance) ?

What is an interesting cluster ?

basis: pressure selection -> keep genes working together close

How to model clusters ? Graphs / strings ?

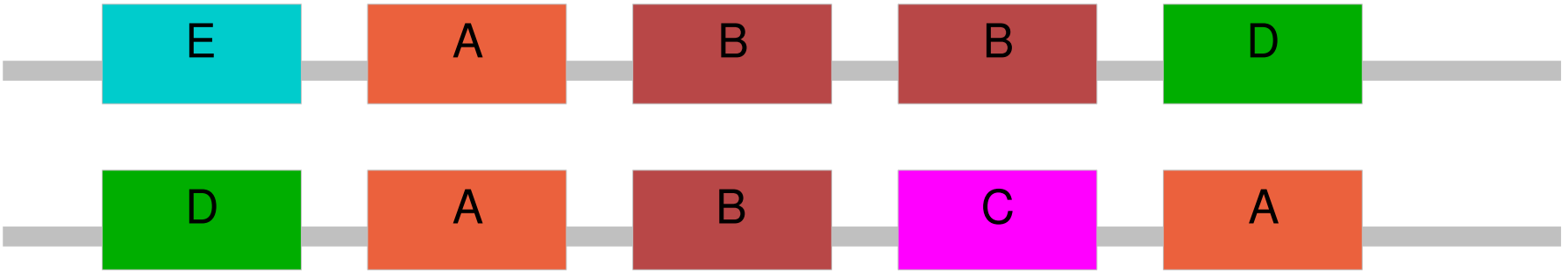
How to compute those clusters ?

How to manage the sets of clusters and extract useful information ?

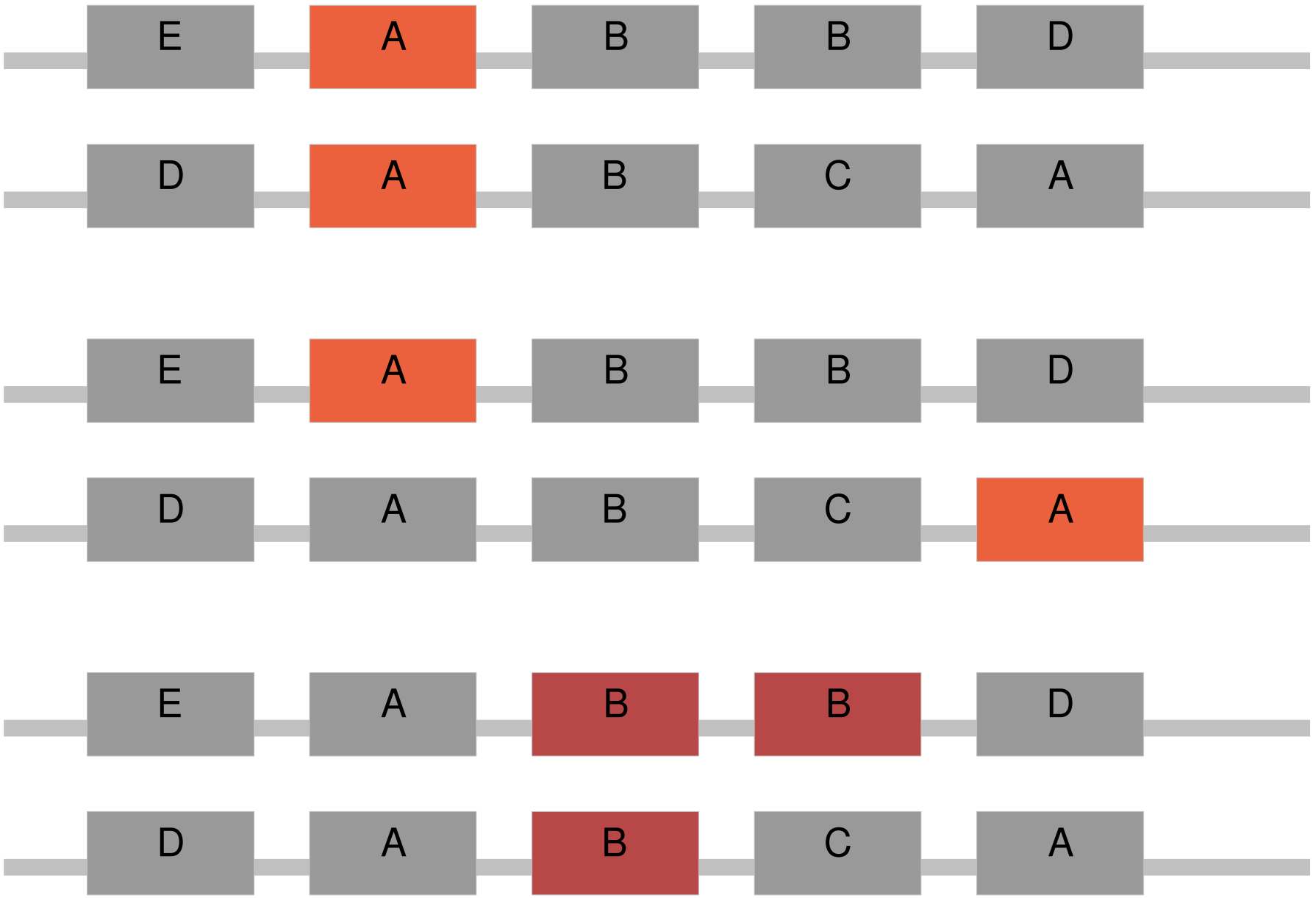
Computer science

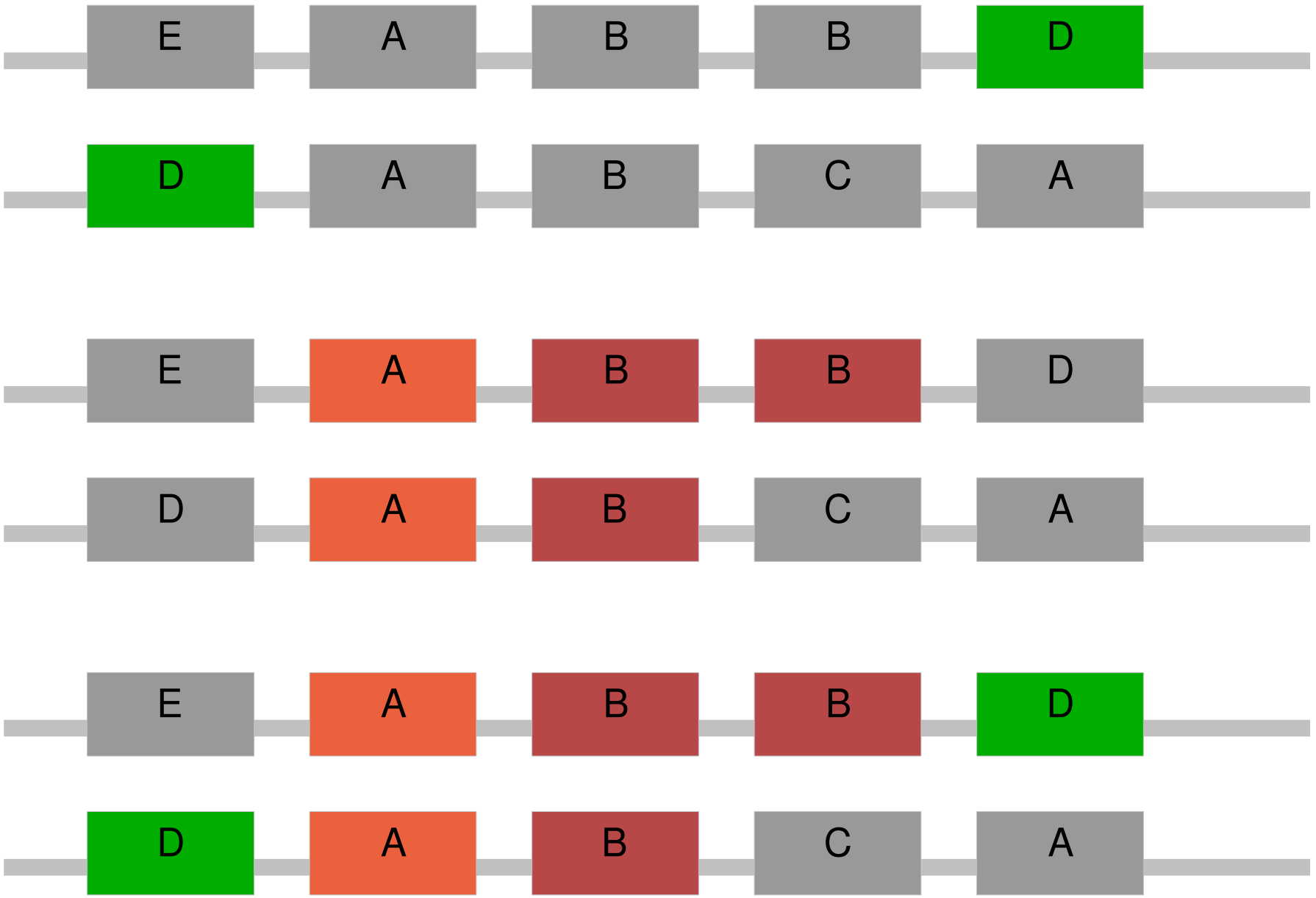
One of the simplest model :
- genomes as strings of units
- common intervals

Simplest case in this model: 2 genomes !




Common interval:
- one interval on each chromosome
- same set of gene in each interval
- externals bounds not in the set of gene

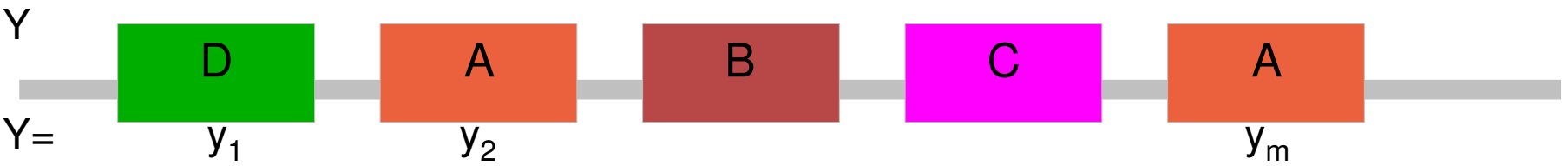

E A B B D
D A B C A
E A B B D
D A B C A
E A B B D
D A B C A


E A B B D
D A B C A
E A B B D
D A B C A
E A B B D
D A B C A

# How many common intervals ?

- X first chromosome, $X = x_1\ x_2\ ..\ x_n$
- Y second chromosome, $Y = y_1\ y_2\ ..\ y_m$

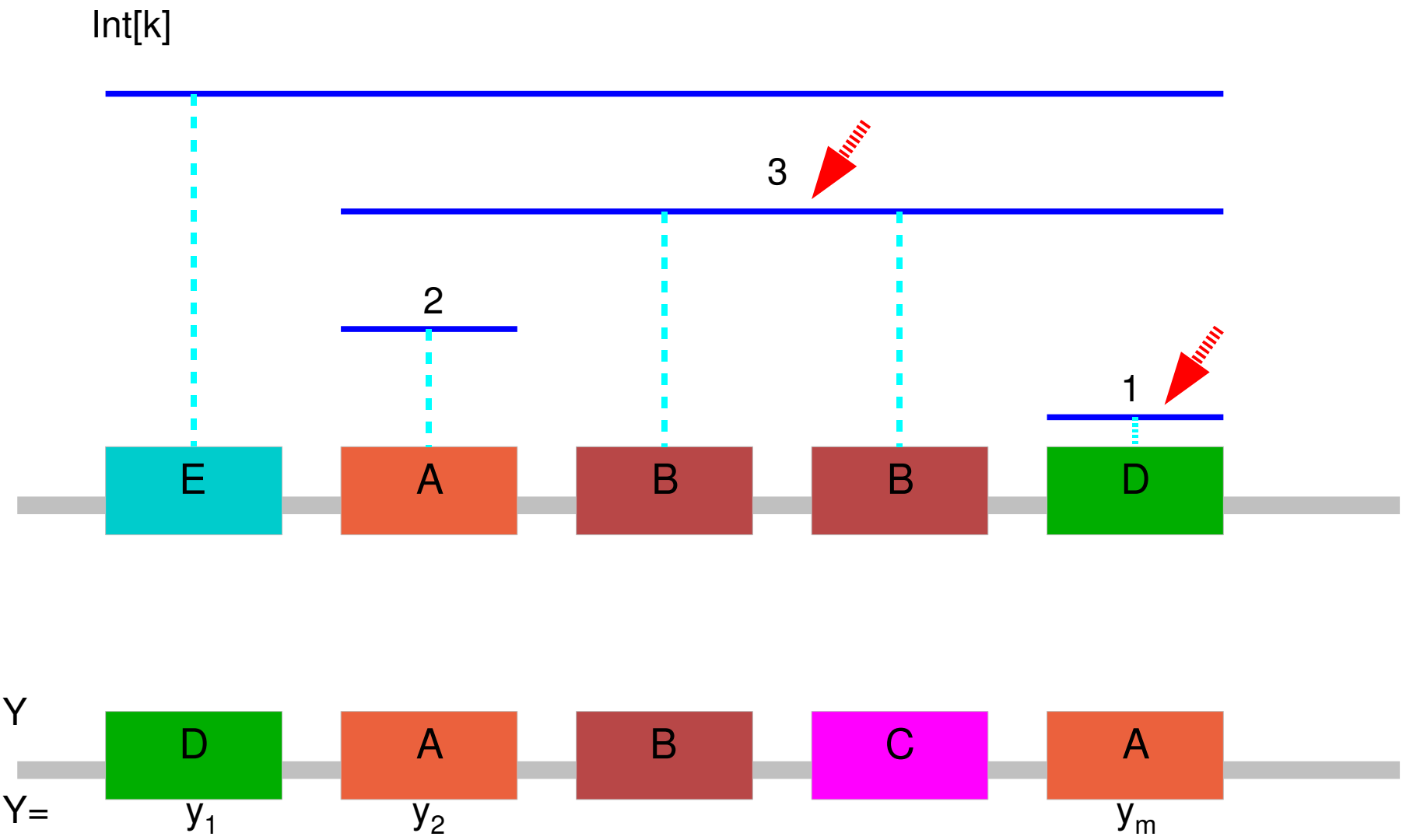
Common alphabet $\Sigma$, $|\Sigma| <= \max(|X|,|Y|)$




fo(Y,1)= D A B C
fo(Y,2) = A B C
fo(Y,3) = B C A
fo (Y,4) = C A
fo (Y,5) = A

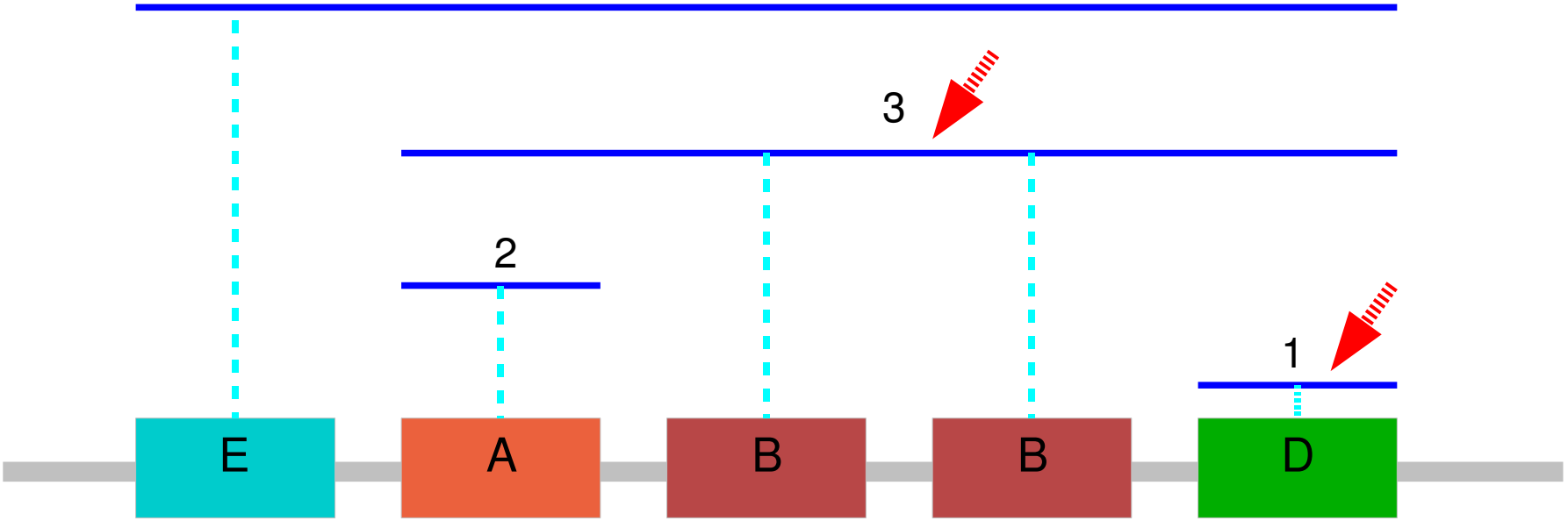
D= 1 A = 2 B = 3 C = 4
A = 1 B = 2 C = 3
B = 1 C= 2 A = 3
C = 1 A = 2
A =1

$Rank_{(Y,1)}[B]=3$




fo(Y,1) = D A B C  B = 1 A =2 C = 3  $Rank_{(Y,2)}[A]=2$

Int[k] are nested !  They form a tree. !




2 n valid Int[k] at max !

2 nm common intervals at maximum

The bound is reached !!

## How to identify all them ?

Two approaches

Direct computation (Didier)

O(nm) but

+ Lowest common ancestor (otherwise O(n m logn)
+ No structure in the output !
+ Complexity does not depend of the input
+ No index

Fingerprint computation on a single string + index+ merge after

+ $O(n+|L_1|\log n + m\,|L_2|\,\log m)$ (can be worst than Didier)
+ Structure in the output and possibility of search of fingerprint
+ Complexity does depend of the input
+ Keep the index for further computations

- $S = s_1..s_N$ string of length n
- alphabet $\Sigma$ of size $|\Sigma|$, not fixed (possibly O(n))

A fingerprint f : set of character(s) of a substring $s_i.. s_j$

General problem:

Compute and represent the set of all fingerprints of S

Examples:

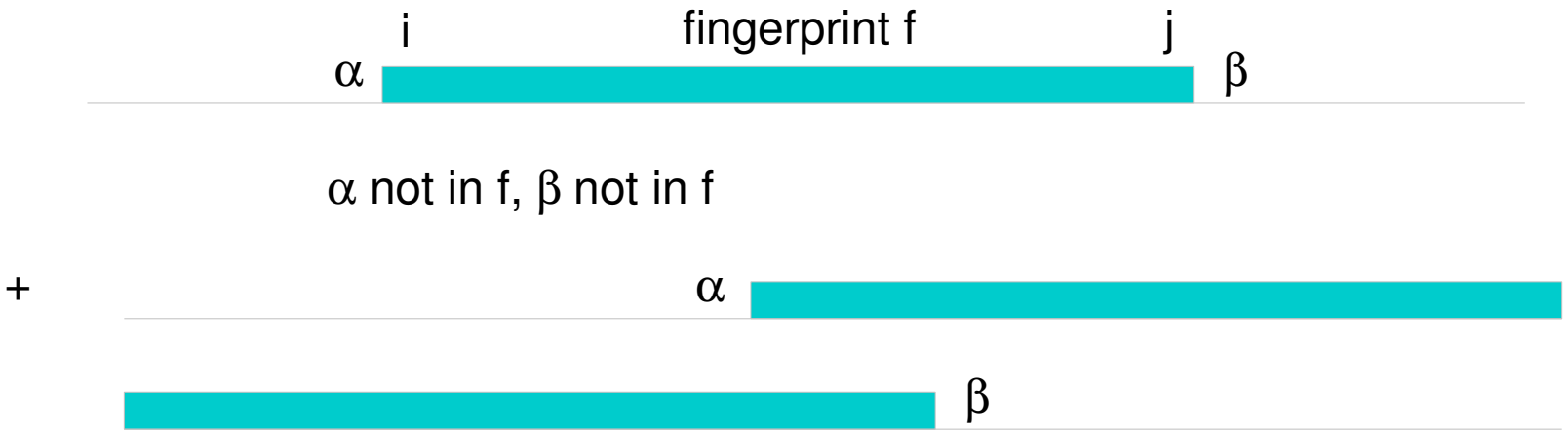
dccbcbabbbc

{a} {b} {c} {d} {c,d} {b,c} {a,b} {b,c,d} {a,b,c} {a,b,c,d}

acbdcadad

{a} {b} {c} {d} {a,c} {a,d} {b,c} {b,d} {c,d} {a,b,c} {a,c,d} {b,c,d} {a,b,c,d}

Maximal location <i,j> of f




Number of maximal locations: $L <= n|\Sigma|$ Complexity of the bound easily reached

But is usually much less

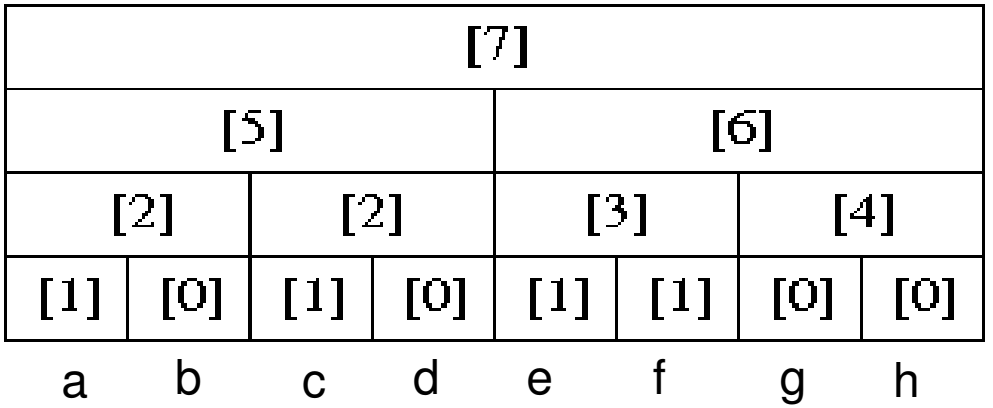
$\Sigma_k = \{a_1,a_2,..,a_k\}$ $w_1 = a_1$, $w_k = w_{k-1}\, a_k\, w_{k-1}$

$w_2=a_1(a_2)a_1$, $w_3=(a_1a_2a_1)a_3(a_1a_2a_3)$, ...

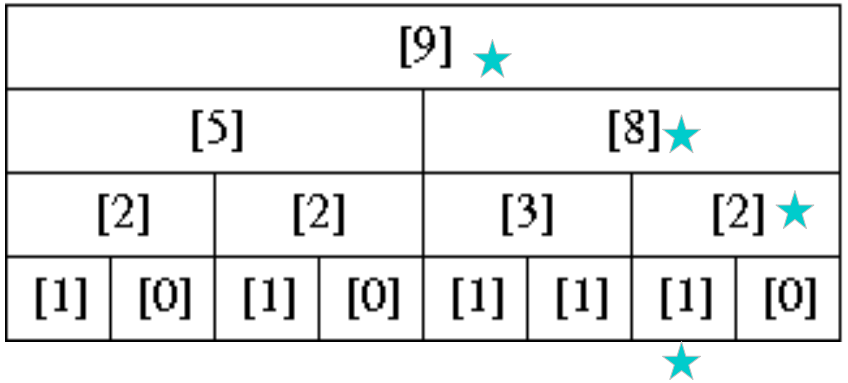
$|w_k| \,.\, |L_k| = k \,.\, (2^k - 1)$ $|L|_k = 2^{k+1}-(k+2)$ → $|L|_k=o(|w_k| \,.\, |L_k|)$

## Naming technique

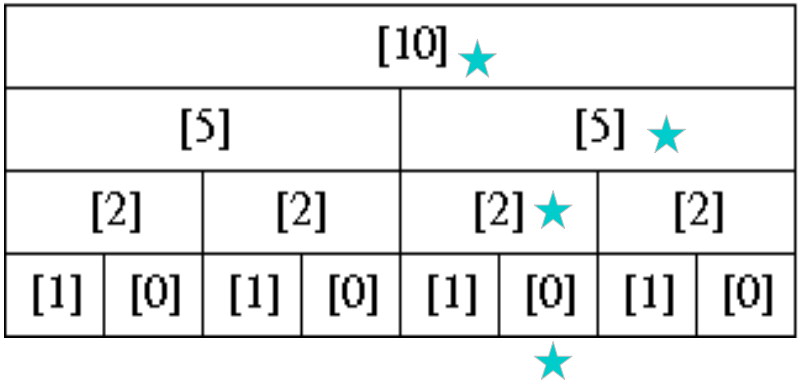
{a,c,e,f}  $\Sigma = \{a,b,c,d,e,f,g,h\}$




$\log |\Sigma| + 1$




{a,c,e,f,g}

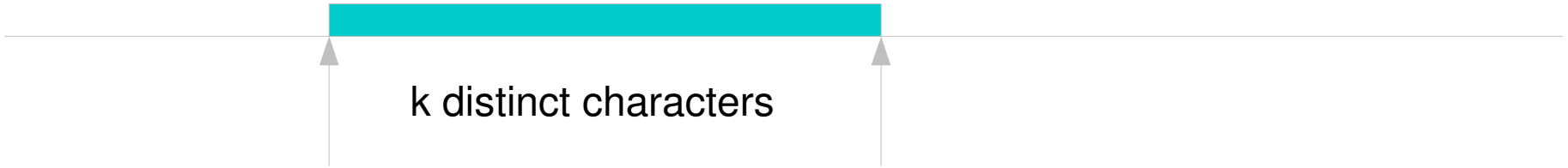



{a,c,e,g}

Names = {[1],[2],[3],[4],[5],[6],[7],[8],[9],[10]}

Fingerprints ={[7],[9],[10]}

Amir, Apostolico, Landau, Satta 2003

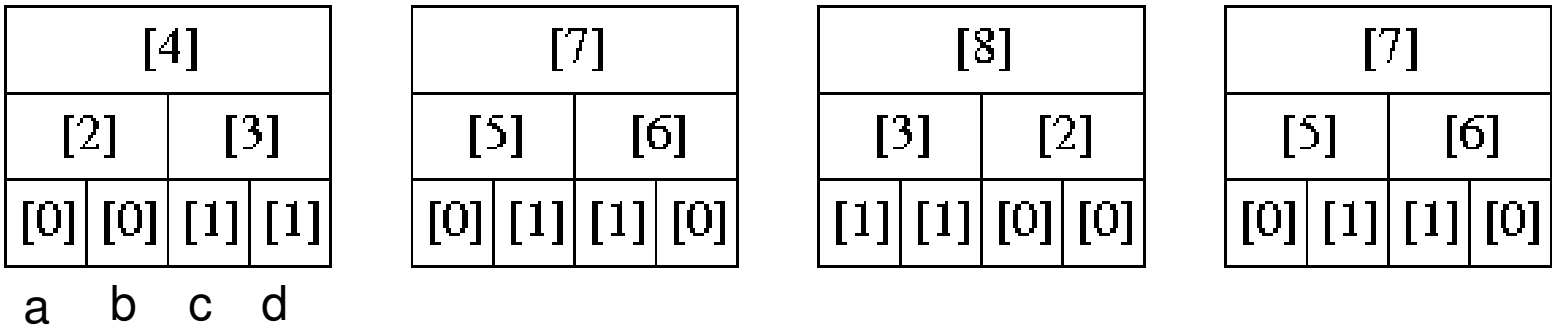



Changing a character: O(log |Σ| log n)  (n new names maximum by level)

One iteration: n log |Σ| log n

|Σ| iterations:  |Σ| n log |Σ| log n

Important: different set of names for each iteration

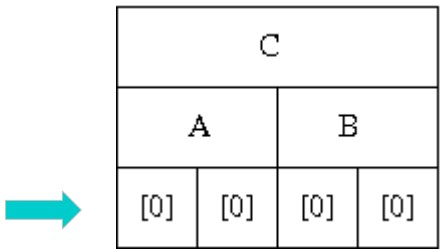



k=2 d c c b c b a b b b c

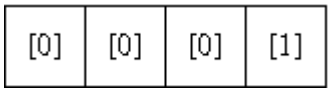
d c c b c b a b b b c

Tsur 2005

List of fingerprints: d c a $d^{-1}$ b  {d}, {c,d}, {a,c,d}, {a,c}, {a,b,c}

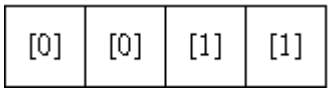



d

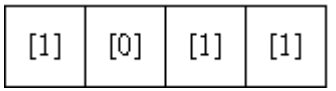



{([0],[1]), B}

d c

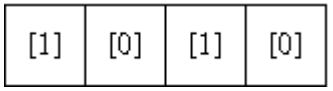



{([1],[1]), B}



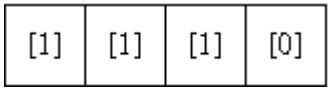

{([1],[0]), A}

d c a $d^{-1}$




{([1],[0]), B}

d c a $d^{-1}$ b




{([1],[1]), A}

List of changes:

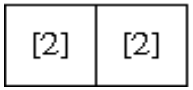
{([0],[0]), A} {([0,0]), B} | {([0],[1]), B} {([1],[1], B} {([1],[0]), A} {([1],[0]), B} {([1],[1]), A}

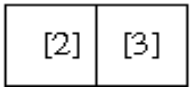
Radix sort on the pairs + unique -> new names

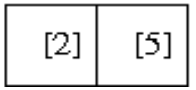
List of changes:
{([0],[0]), A} {([0],[0]), B} | {([0],[1]), B} {([1],[1], B} {([1],[0]), A} {([1],[0]), B} {([1],[1]), A}

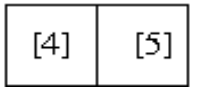
[2] -> ([0],[0])
[3] -> ([0],[1])
[4] -> ([1],[0])
[5] -> ([1],[1])

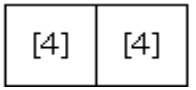
New list:
{[2], A} {[2], B} | {[3], B} {[5], B} {[4], A} {[4], B} {[5], A}

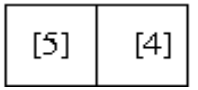



{([2],[2]), C}  {([2],[3]),C}

New list: {([2],[2]),C} | {([2],[3]),C} {([2],[5]),C} {([4],[5]),C} {([4],[4]),C} {([5],[4]),C}

Radix sort, ...

Tsur 2005

Radix sort: O(n)  (bounded integers)

One iteration : n log $|\Sigma|$  No more name search !

$|\Sigma|$ iterations:  $|\Sigma|$ n log $|\Sigma|$

Problems
- does not depend of L
- distinct names at each iteration

Our approach (2006)  Simple sequence: no repeated character

lfo(i)  a b a c e a b a c d  a b a c e a b a c d

lfo(4)=ceab  lfo(2) = bace

Concatenate # to the sequence

Bijection  L / proper prefixes of lfo(i)

cea  a b a c e a b a c d #

bac  a b a c e a b a c d #

Compute all lfo(i) of S#

Our approach (2006)

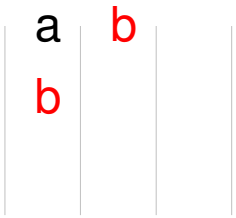
How to calculate all lfo(i) ?

abcbadca

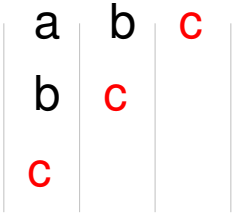
a | bcbadca#



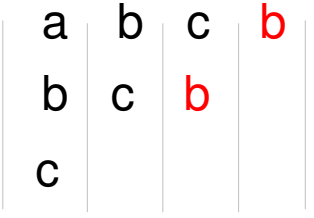

ab | cbadca#



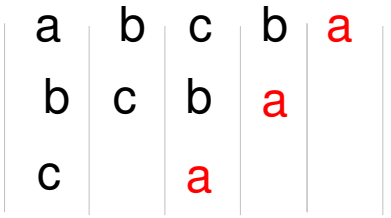

abc | badca#



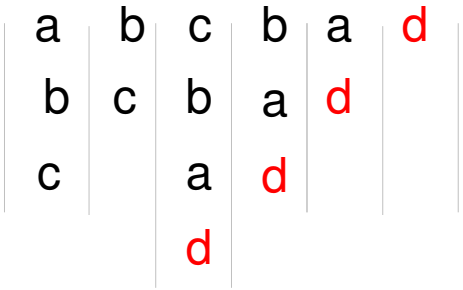

abcb | adca#



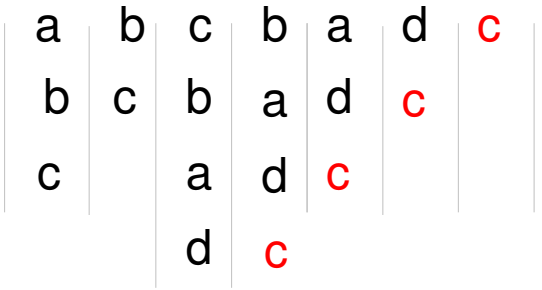

abcba | dca#



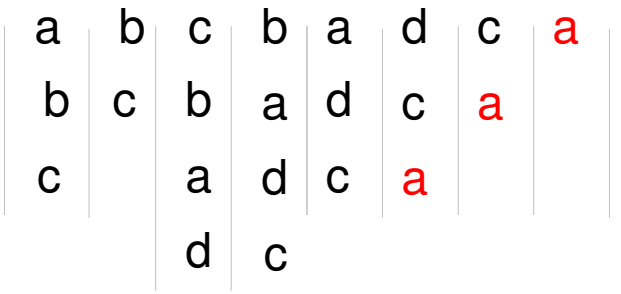

abcbad | ca#




abcbadc | a#

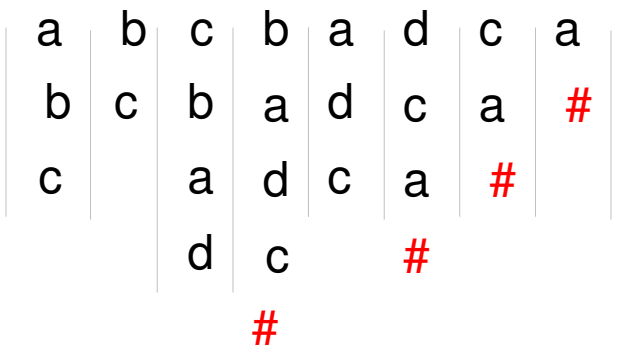


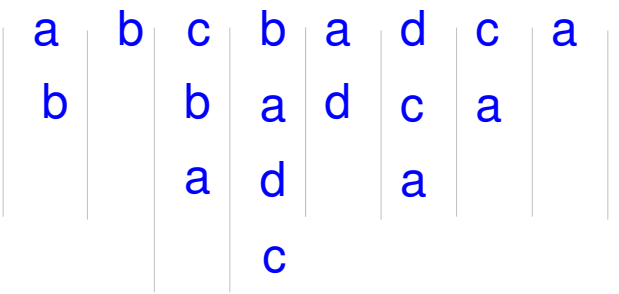

abcbadca | #



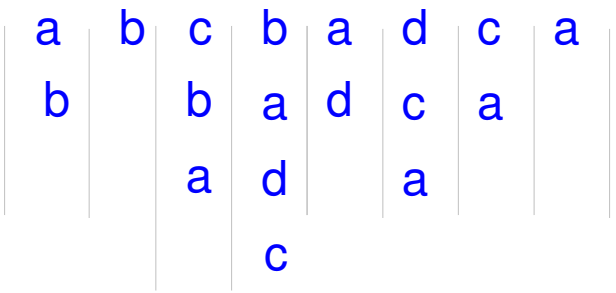

abcbadca#

lfo(i)



Our approach (2006)  Naming all proper prefixes of lfo(i)




n lists:
- Tsur algorithm
- Common names

Simple sequence: O(|L| log |Σ|)
General sequence: O(n+|L| log |Σ|)

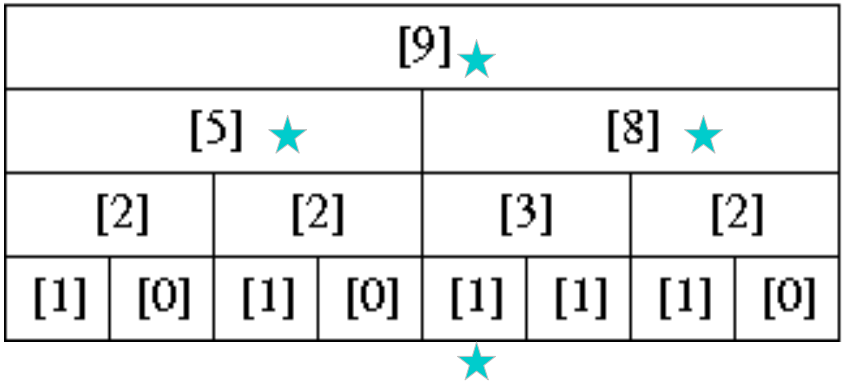
|L|<= n |Σ|  ➡  Faster or as fast as that of Tsur

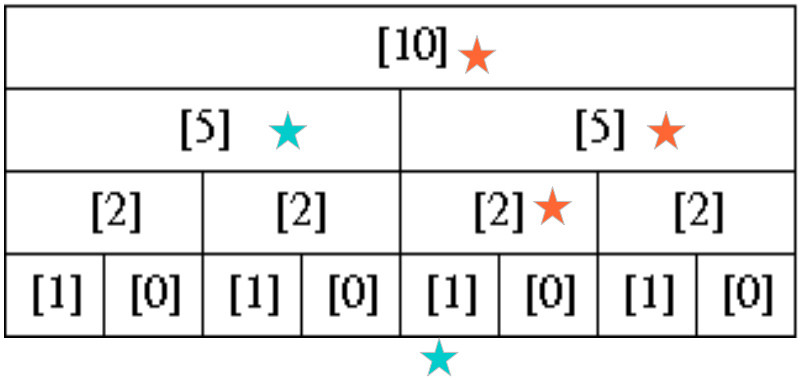
Our approach (2006)

Properties and operations on our names

- a unique set of names

→ Compute the LCP of two fingerprints in log |Σ|



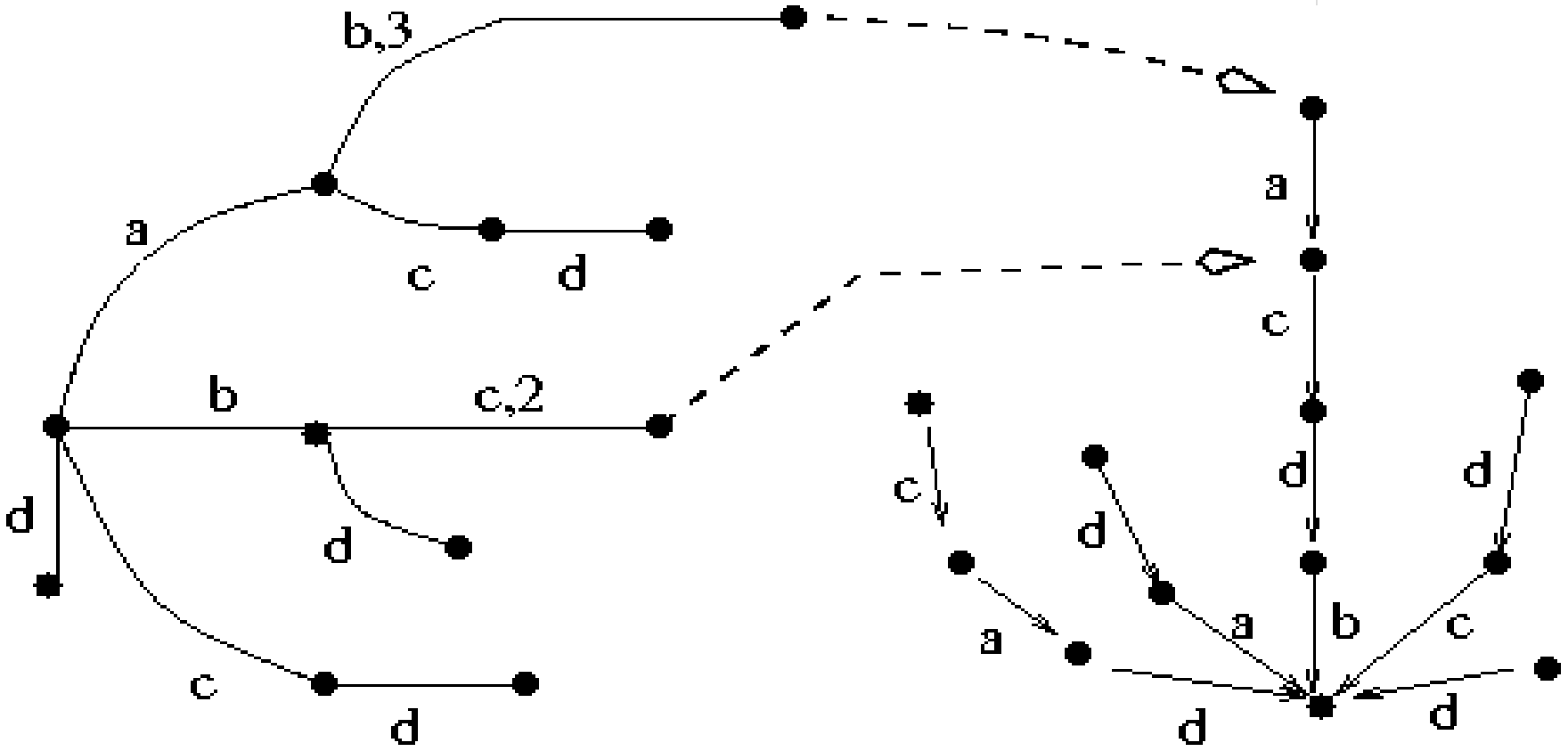

- names sorted by lexicographic order of fingerprints

Fingerprint trie

Chan *et al*, ESA 2007

bdcad

| b | d | c | a | d |
|---|---|---|---|---|
| d | c | a | d | |
| c | | d | | |
| a | | | | |




O(|F|) space

O(|F|log|Σ|) time

Search in O(|f|log(|f|/|Σ|))

## Back to common intervals:

1) Build the tree for the first sequence: $O(n+|L_1| \log |\Sigma|)$
2) Build the tree for the second sequence: $O(m+|L_2| \log |\Sigma|)$

3) Merge the two trees !

Complexity: $O((n+m)+(|L_1|+|L_2|) \log |\Sigma|)$ time.

## Open problems

- Memory space reduction
- Order ?
- Approximate fingerprint
- Distance by fingerprints
- 2D fingerprints