RECONSTRUCTING A **SUPERGENETREE** MINIMIZING RECONCILIATION

Manuel Lafond[1], Aïda Ouangraoua[2], Nadia El-Mabrouk[1]

[1]Université de Montréal
[2]Université de Sherbrooke

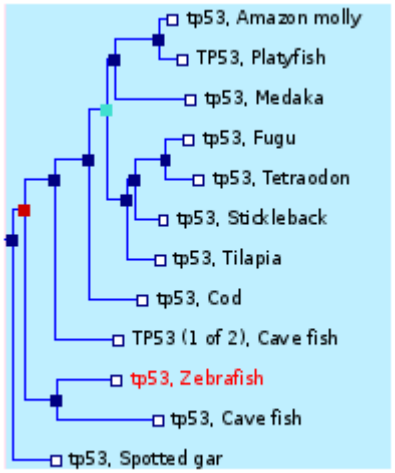
The plan

In this talk I…

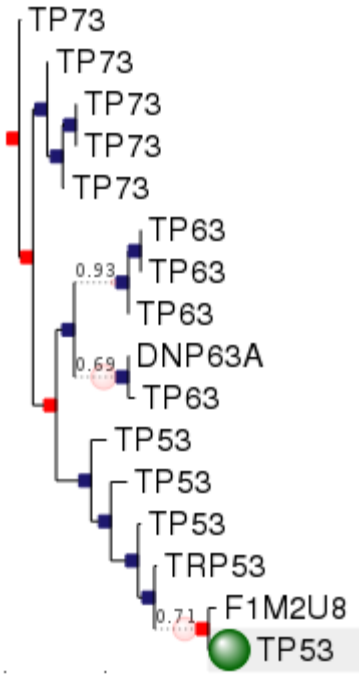
- …come up with supertree problems
  - Finding a supergenetree that minimizes duplications
- …convince you that they're hard
- …try to do something about it
  - Exact, brute-force algorithm
  - A greedy heuristic

TP53 gene tree(s)

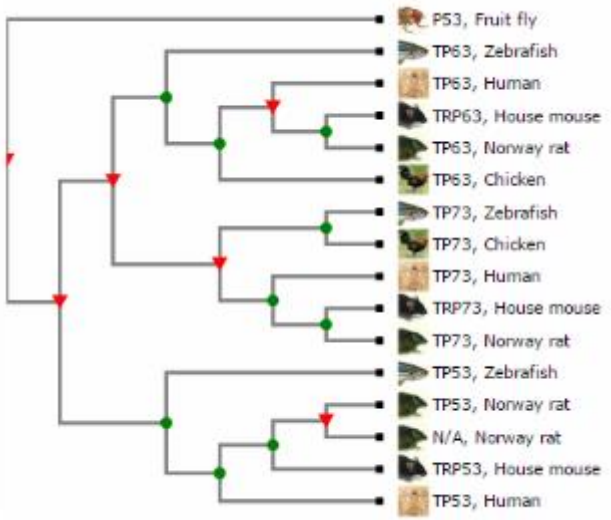
Ensembl

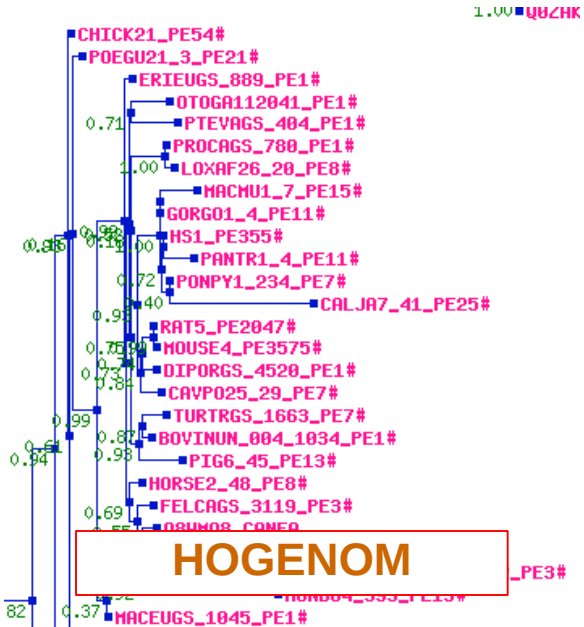


PhylomeDB



TreeFam





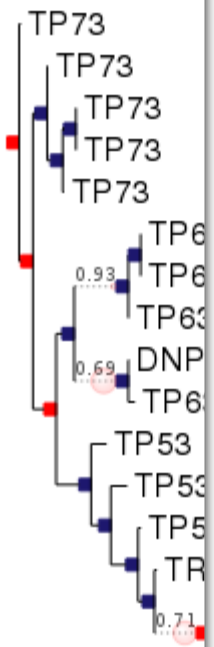
HOGENOM

TP53 gene tree(s)

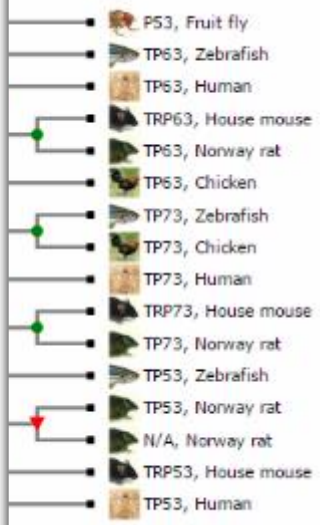
Ensembl

Phylome

Ensembl + PhylomeDB + TreeFam + HOGENOM + …

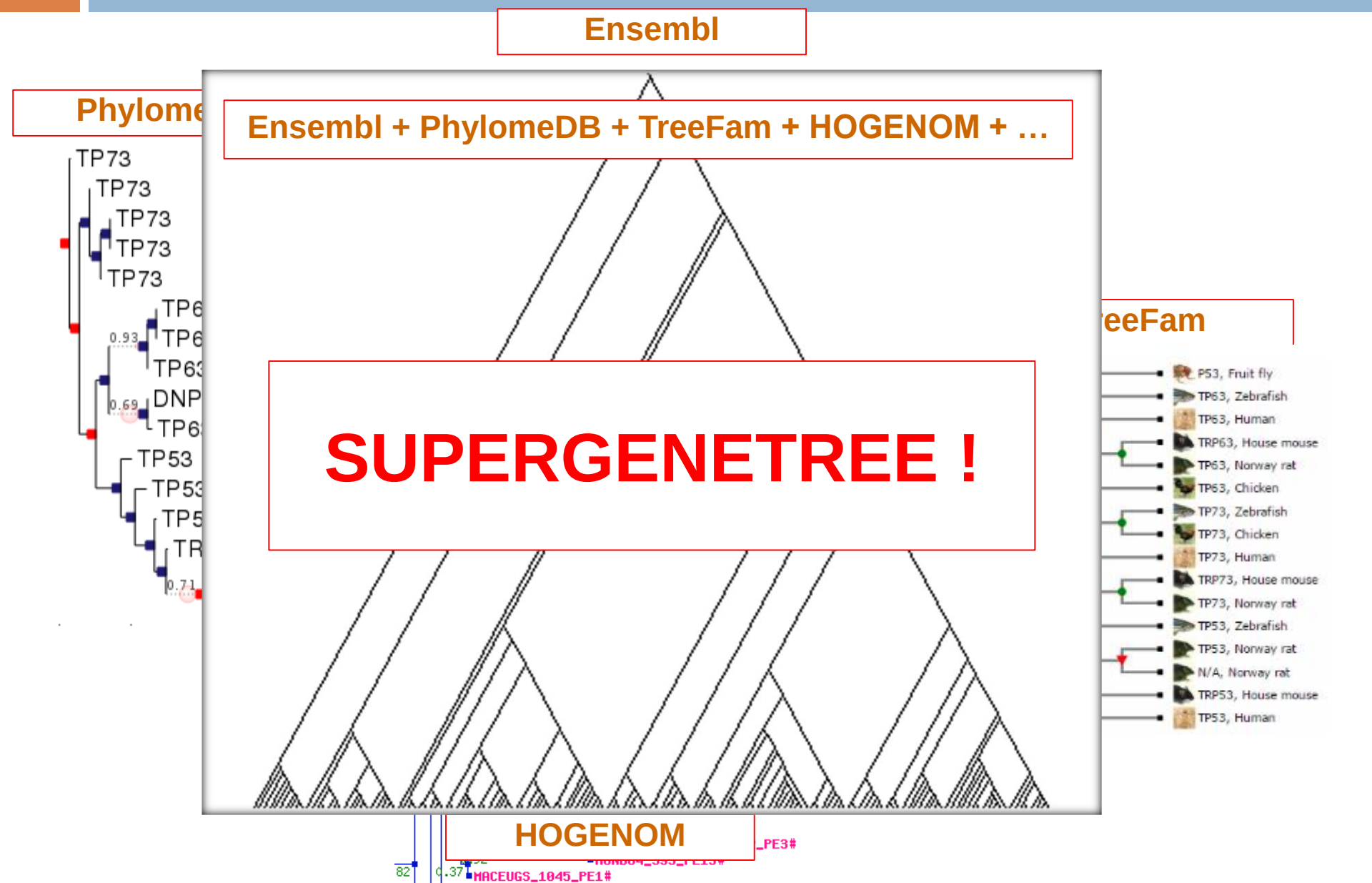


eeFam

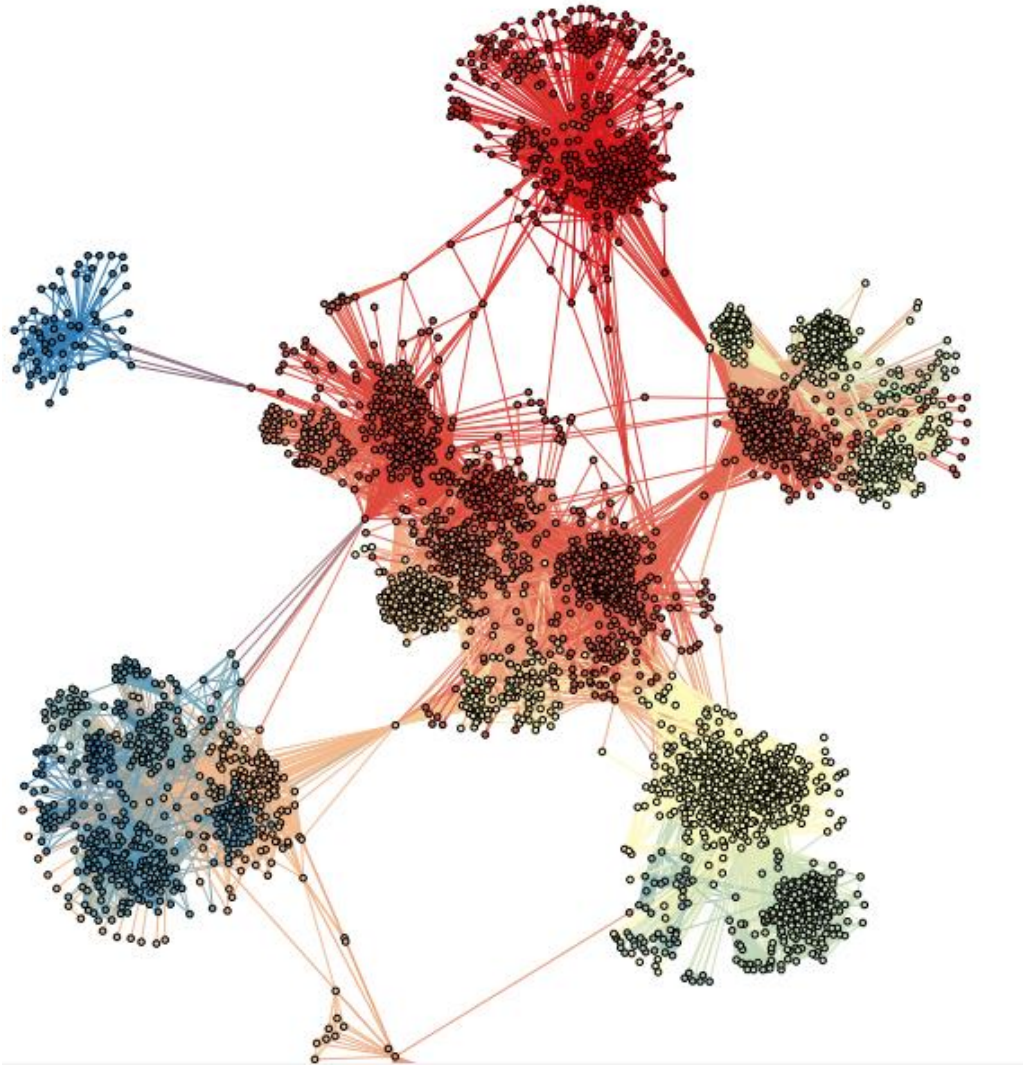


HOGENOM

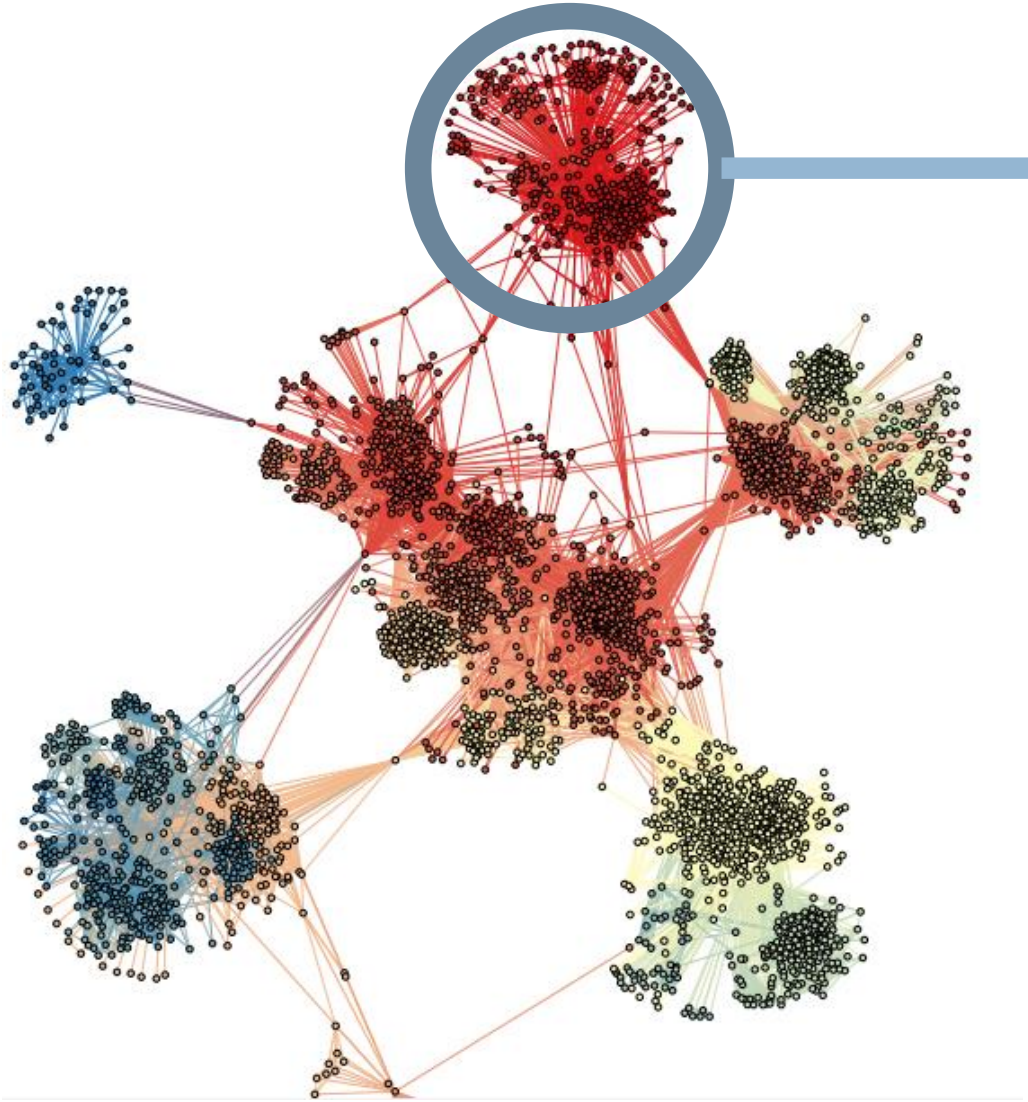
TP53 gene tree(s)

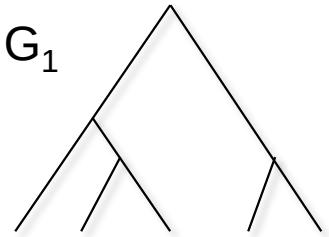


Clusters of orthologous groups

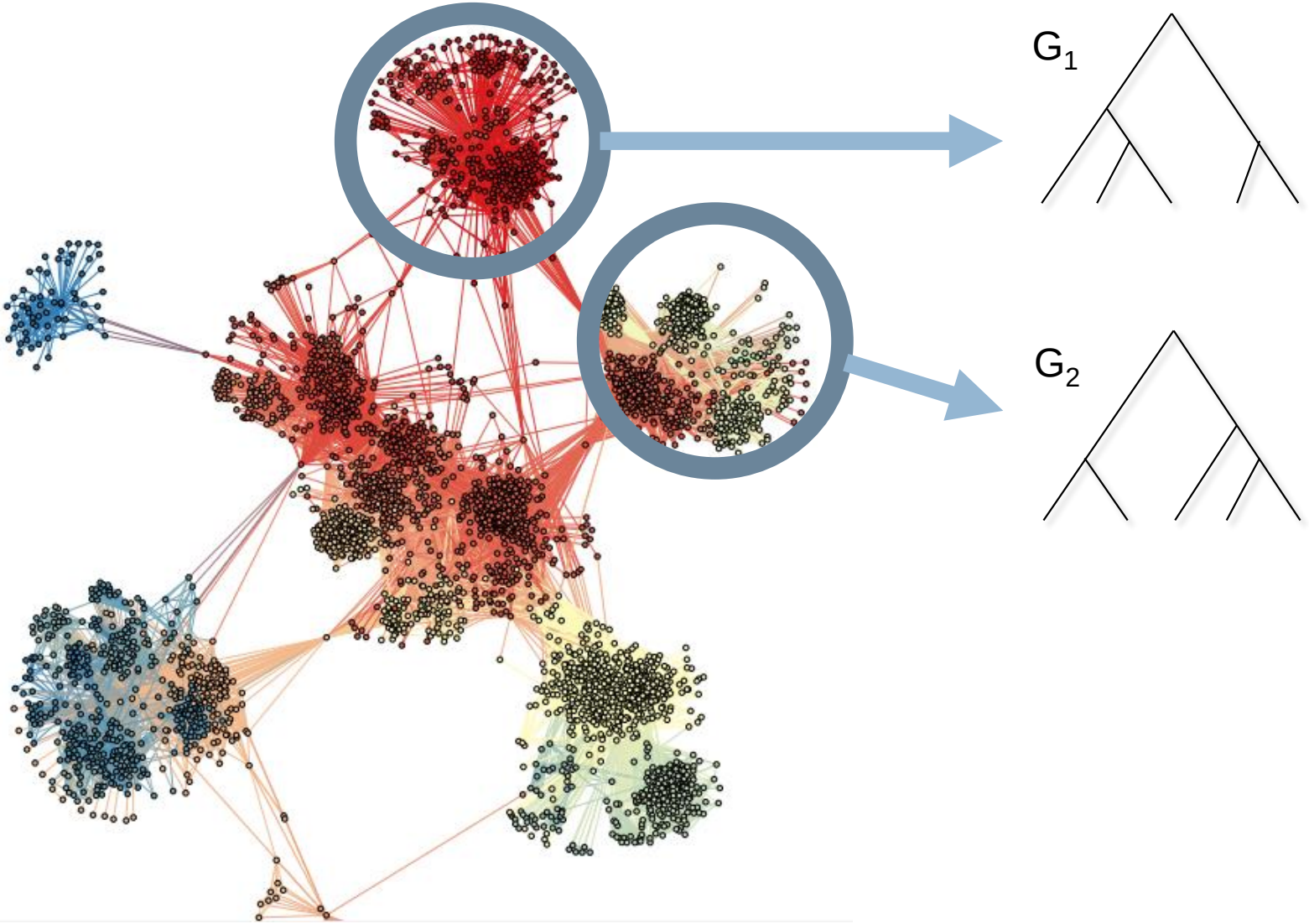


Clusters of orthologous groups

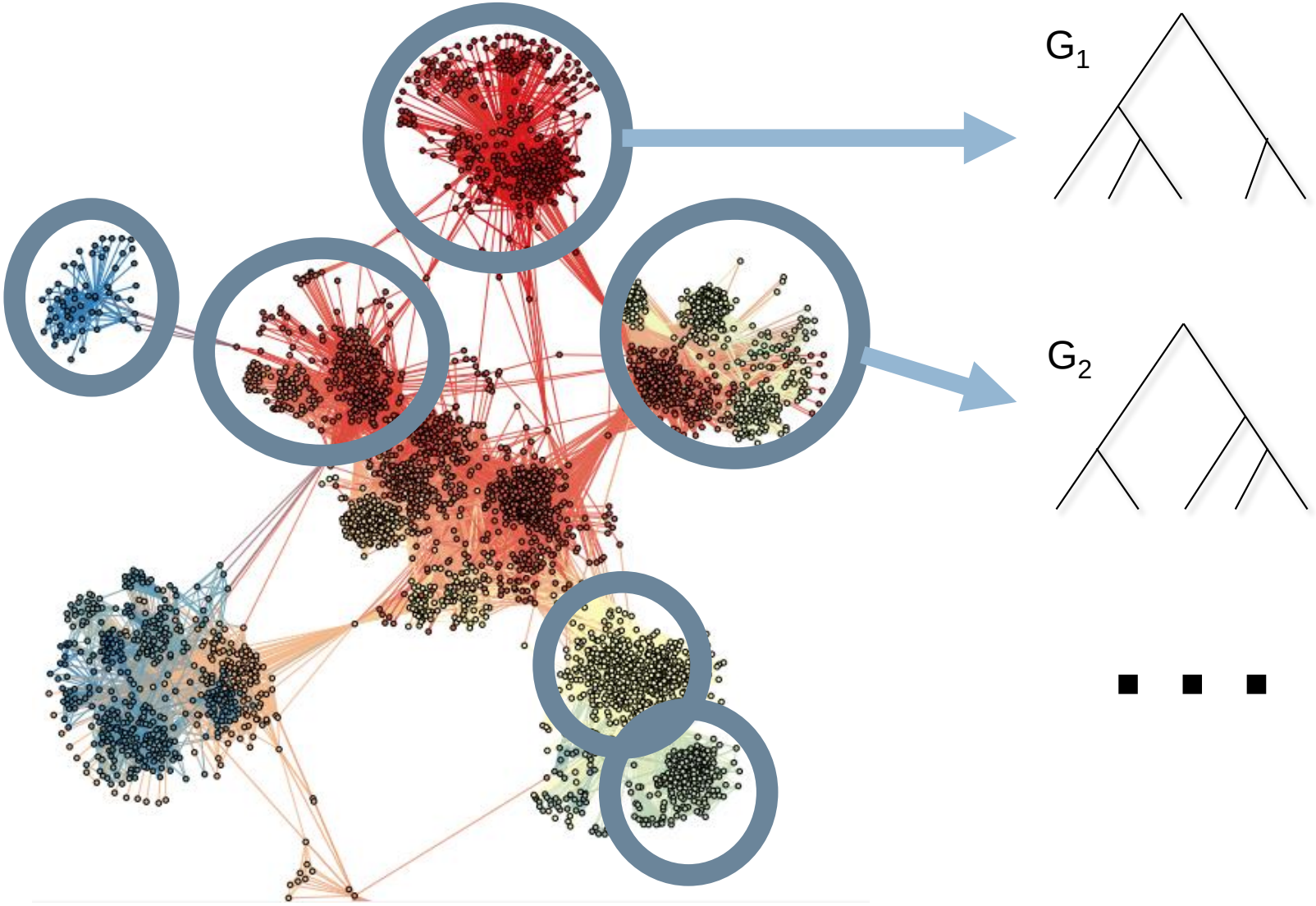




Clusters of orthologous groups



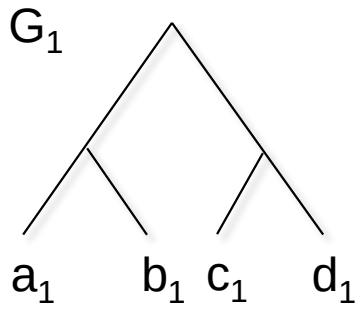
Clusters of orthologous groups

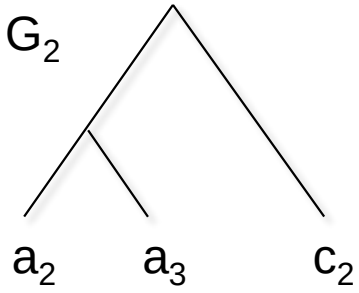


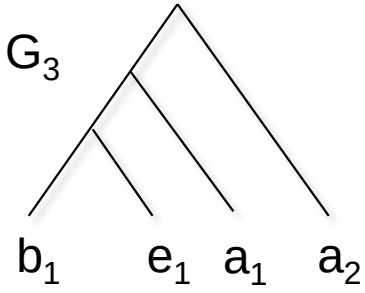
Clusters of orthologous groups

The Supergenetree problem

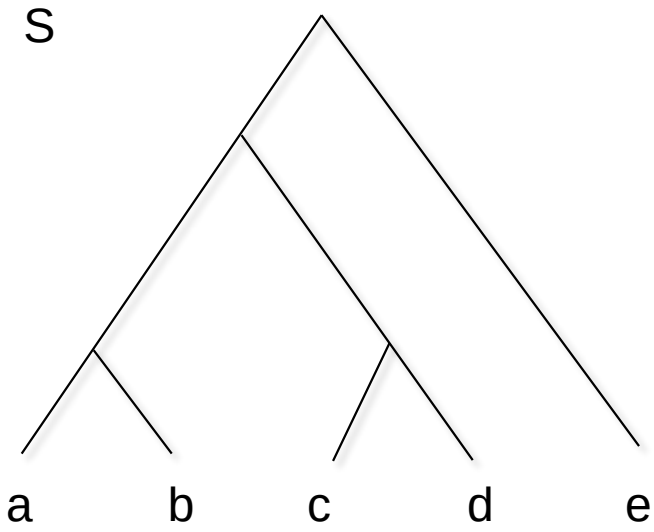
Multiple gene trees







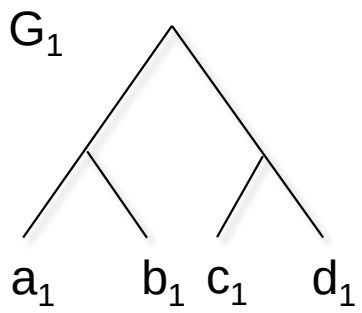
Species tree

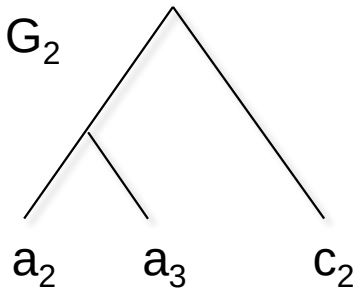


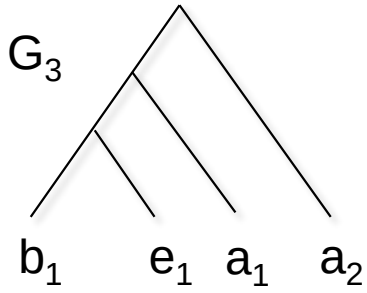
- **Gene tree label = species**
- **Multiple copies (paralogs)**
  - **e.g. a_1, a_2, a_3**
- **Gene trees may be partial + discordant with S (e.g. G_3)**

The Supergenetree problem

Multiple gene trees



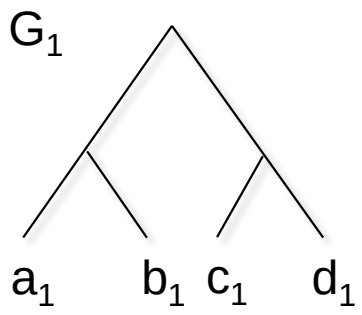


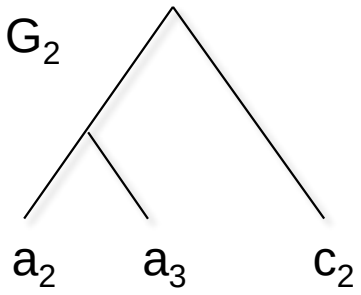


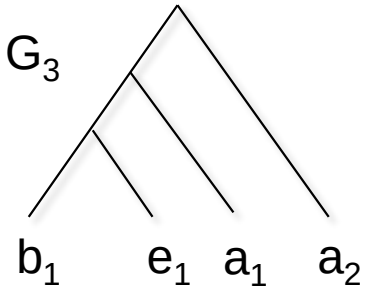
- **Our goal : find a gene tree that displays them all**

The Supergenetree problem

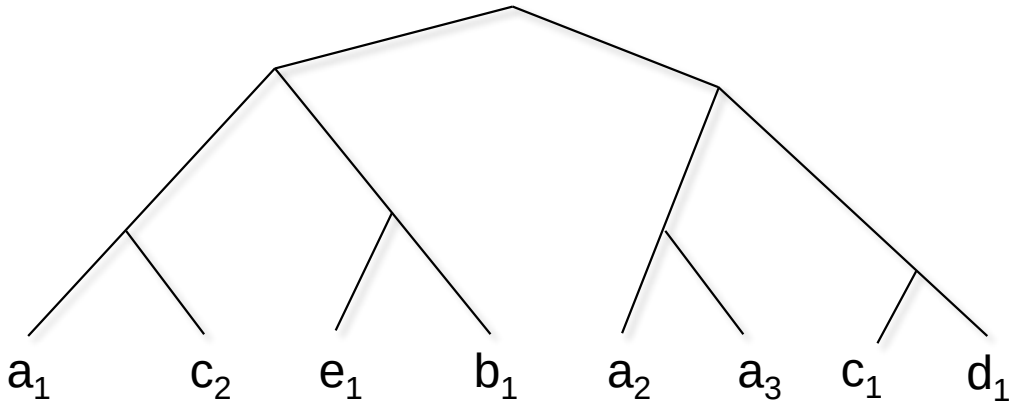
Multiple gene trees







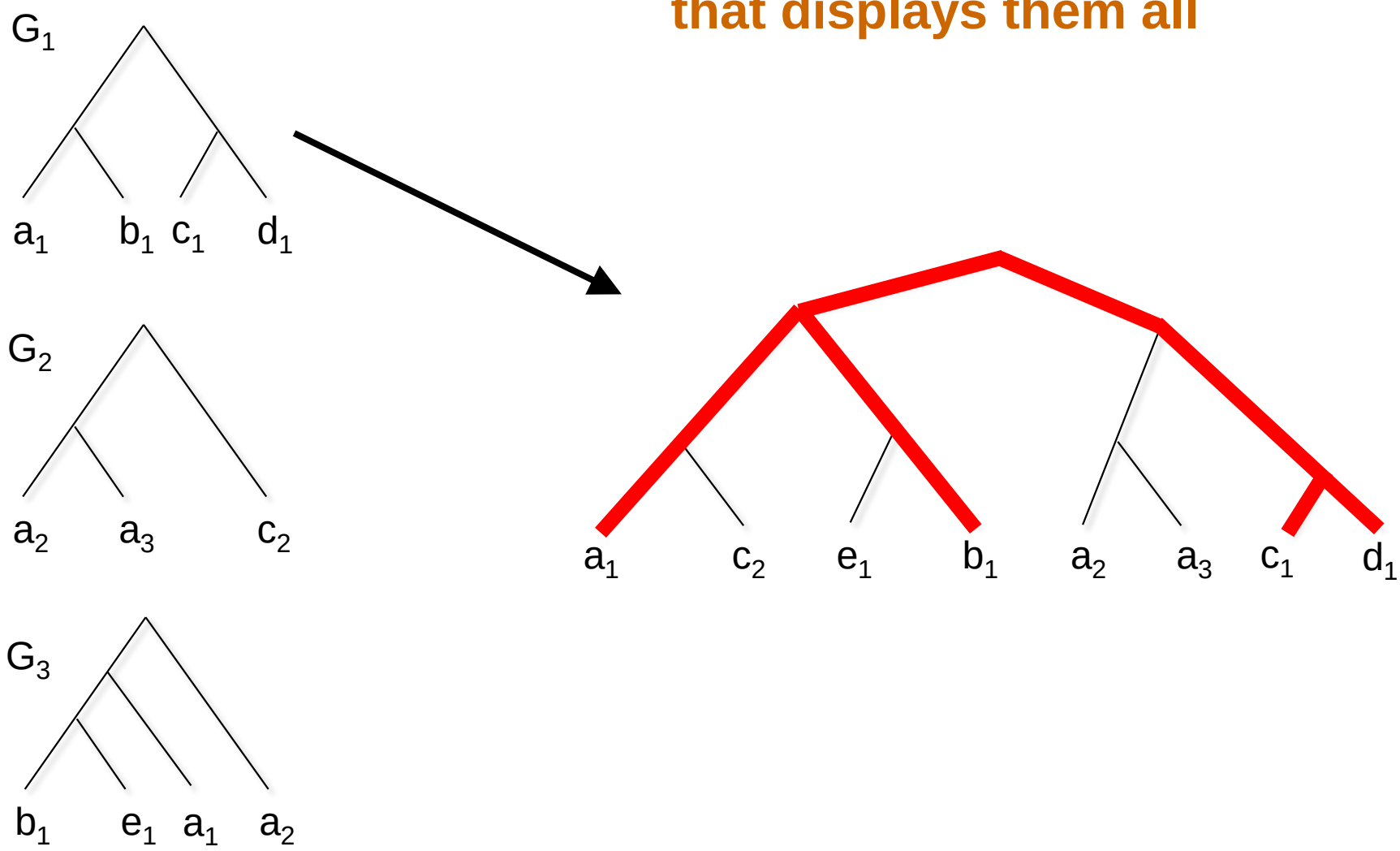
- **Our goal : find a gene tree that displays them all**



The Supergenetree problem

Multiple gene trees

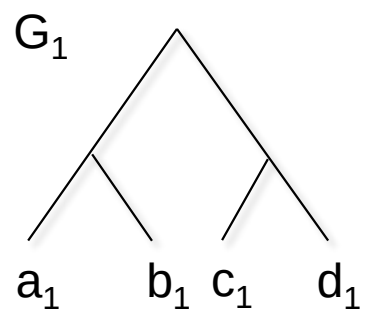
- **Our goal : find a gene tree that displays them all**

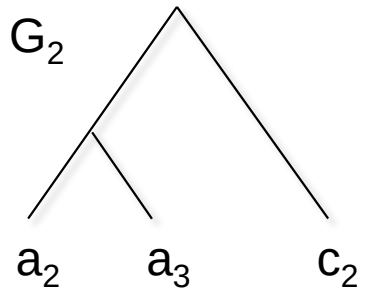


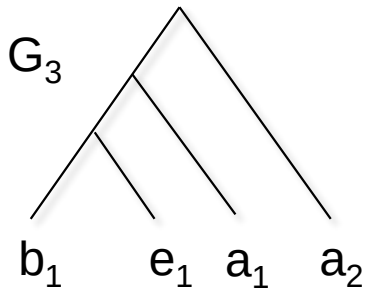
The Supergenetree problem

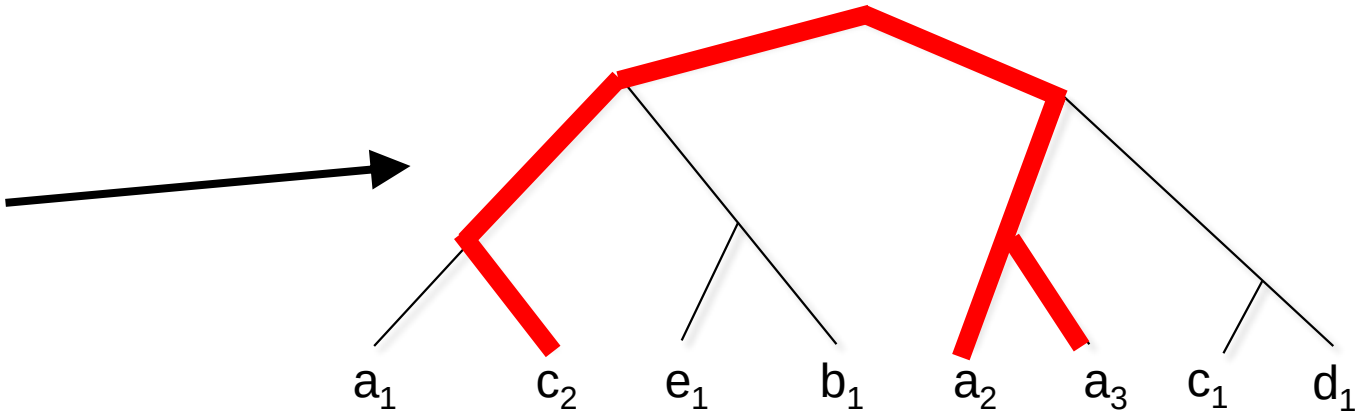
Multiple gene trees

- **Our goal : find a gene tree that displays them all**





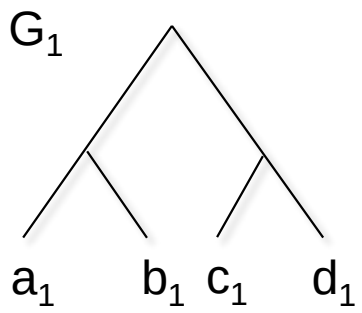


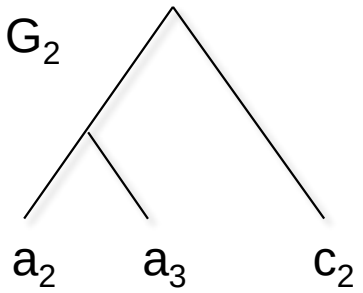


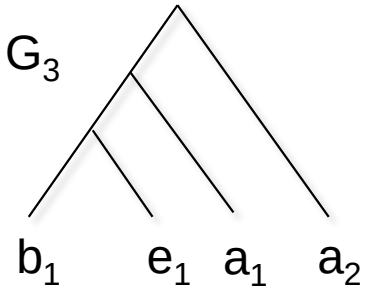
The Supergenetree problem

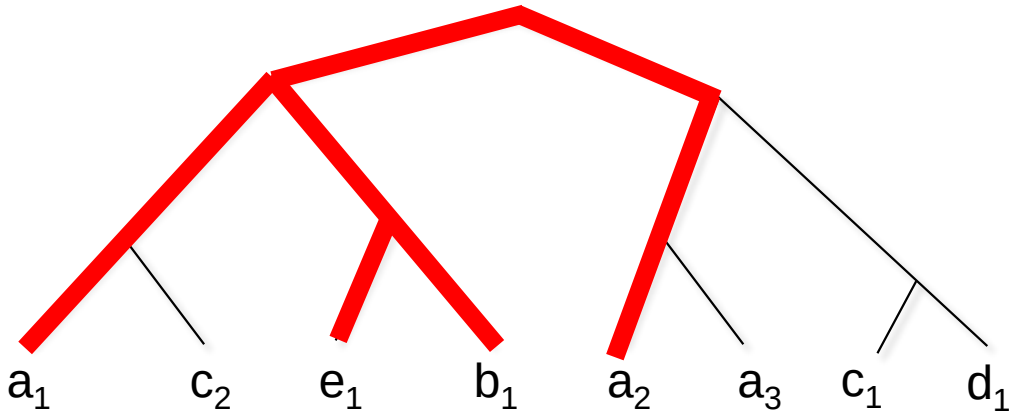
Multiple gene trees

- **Our goal : find a gene tree that displays them all**









SuperGeneTree

- Our trees are said **compatible** if there is a supertree displaying them all
- Finding a supertree (or determining incompatibility) is an old problem
  - The BUILD algorithm does that *(Aho & al., 1981)*
- What's different about super**gene**trees ?

SuperGeneTree

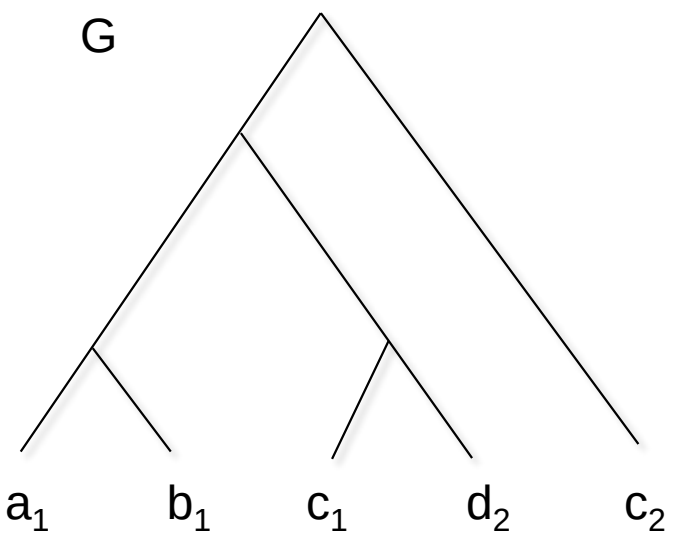
- Our trees are said **compatible** if there is a supertree displaying them all
- Finding a supertree (or determining incompatibility) is an old problem
  - The BUILD algorithm does that *(Aho & al., 1981)*
- What's different about super**gene**trees ?
- We have the **species tree**

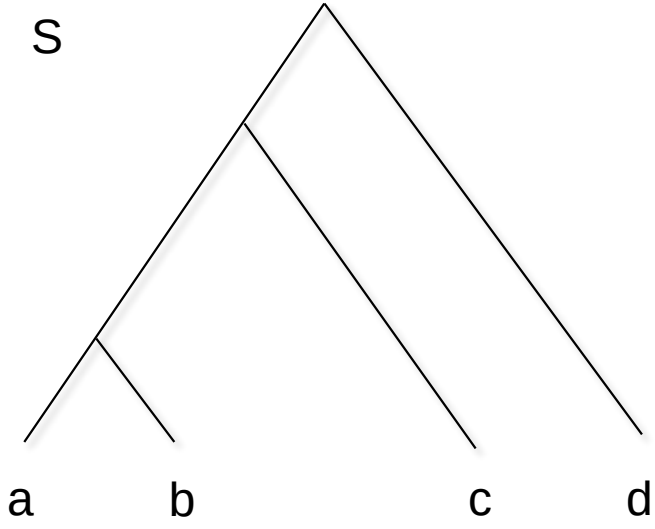
SuperGeneTree

- Often, many supergenetrees exist
- Which one is the best ?
- We **explore ways to choose** using information from the species tree S
- More specifically, we **explore ways to use reconciliation** with S to pick the best supergenetree

Reconciliation

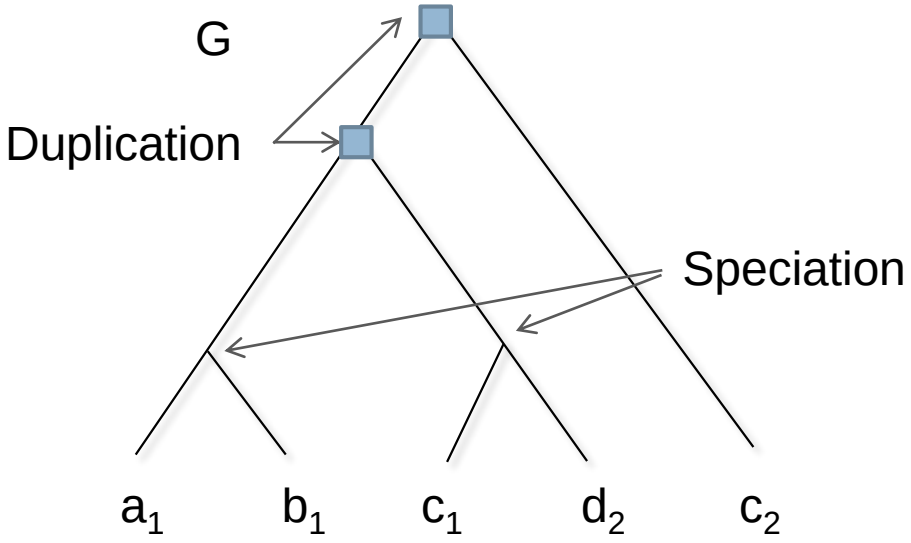
Reconciliation identifies **duplication**, **speciation** and **loss** events in G.

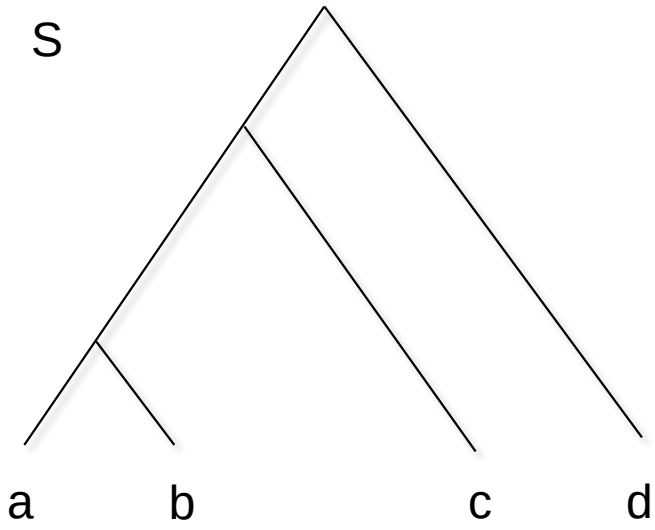




Reconciliation

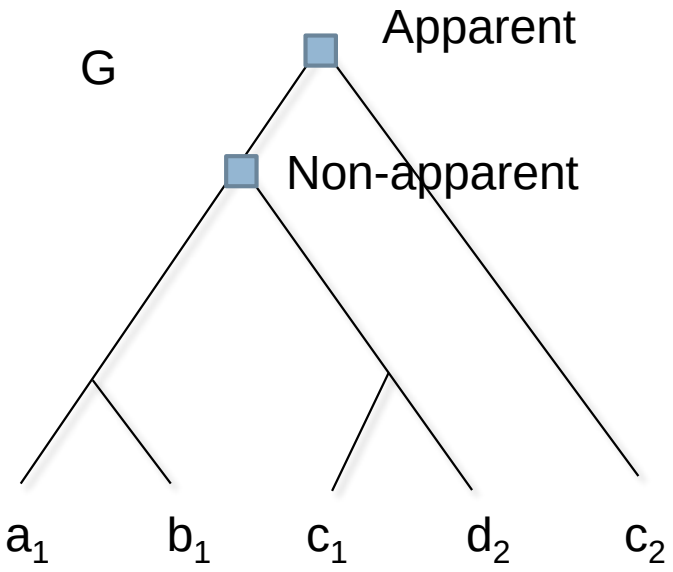
Reconciliation identifies **duplication**, **speciation** and **loss** events in G.

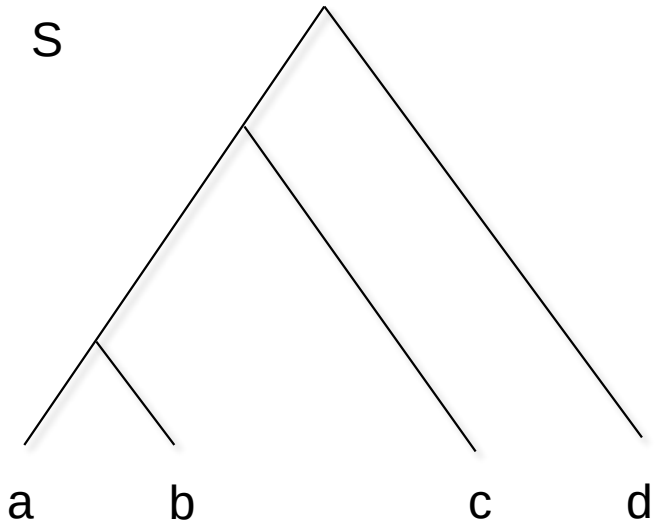




Reconciliation

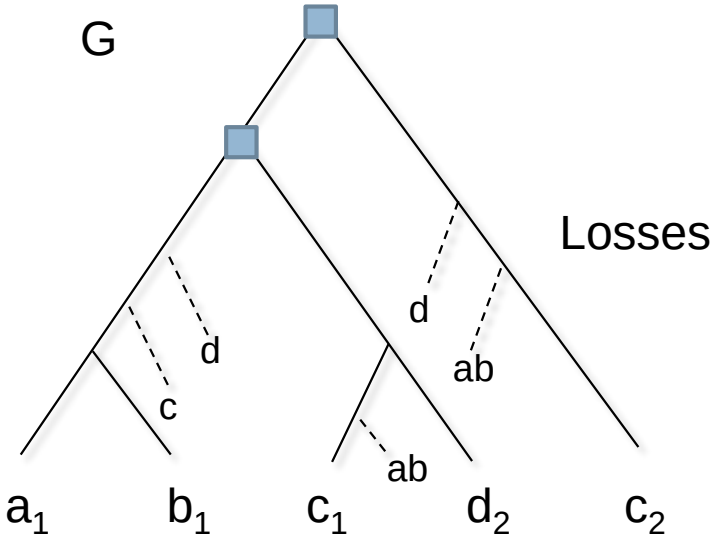
Reconciliation identifies **duplication**, **speciation** and **loss** events in G.

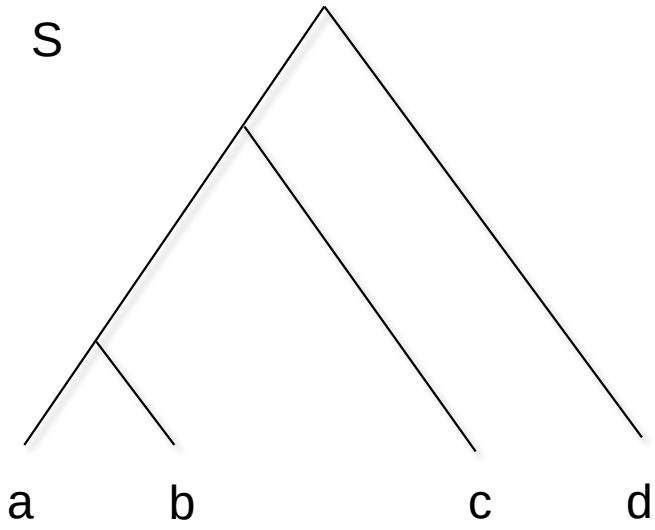




Reconciliation

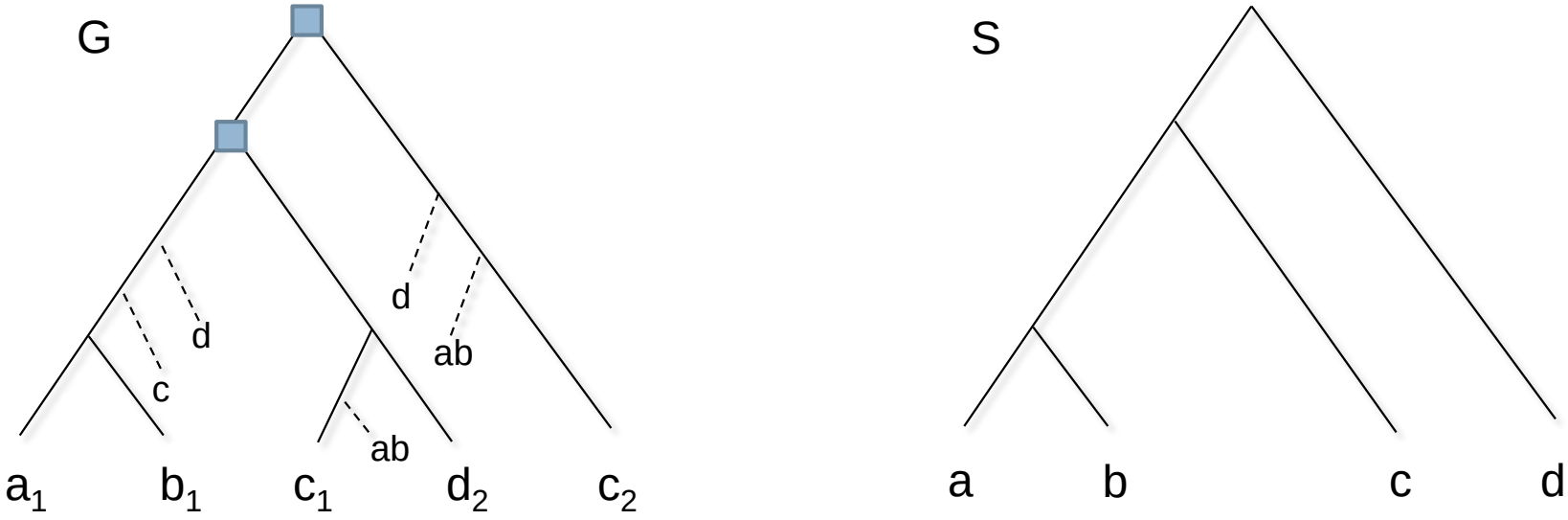
Reconciliation identifies **duplication**, **speciation** and **loss** events in G.





Reconciliation

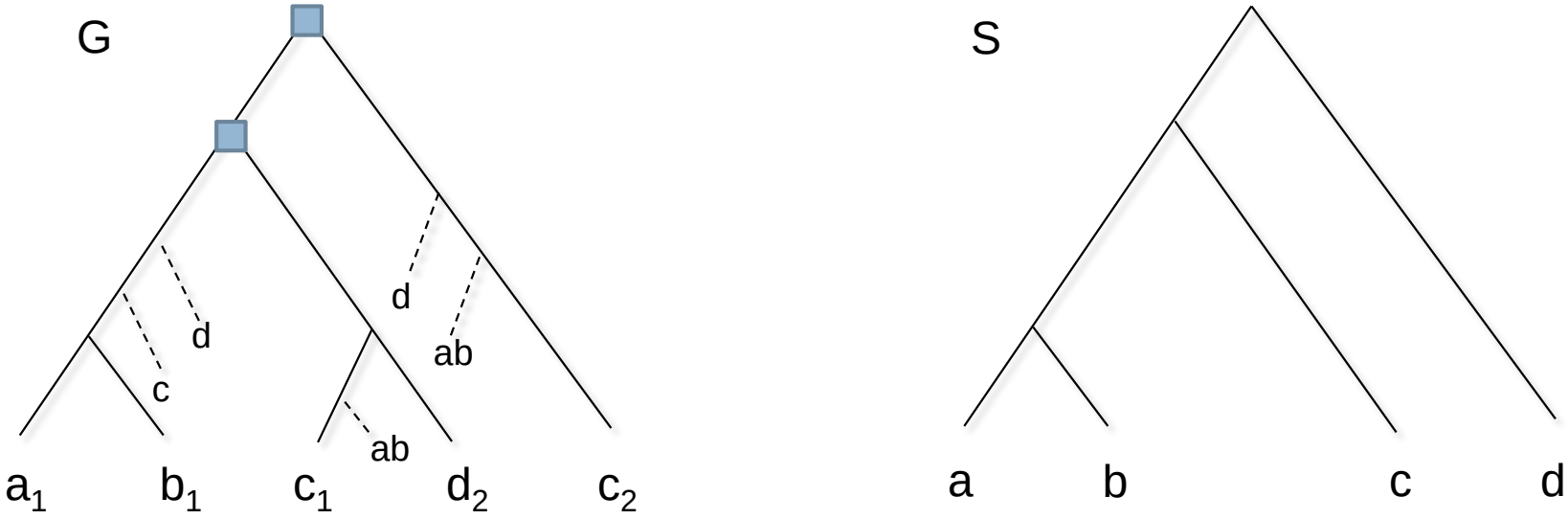
Reconciliation identifies **duplication**, **speciation** and **loss** events in G.



Possible reconciliation costs : #dups, #dups + #losses

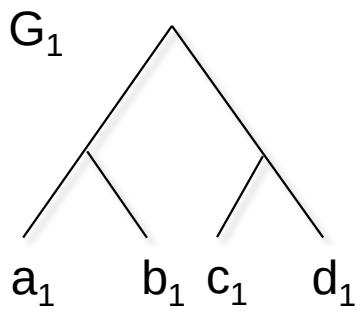
Reconciliation

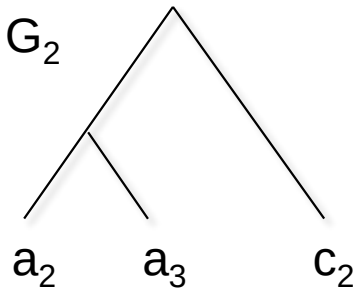
Reconciliation identifies **duplication**, **speciation** and **loss** events in G.

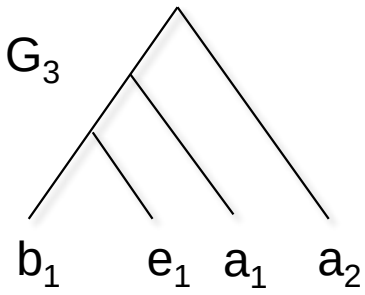


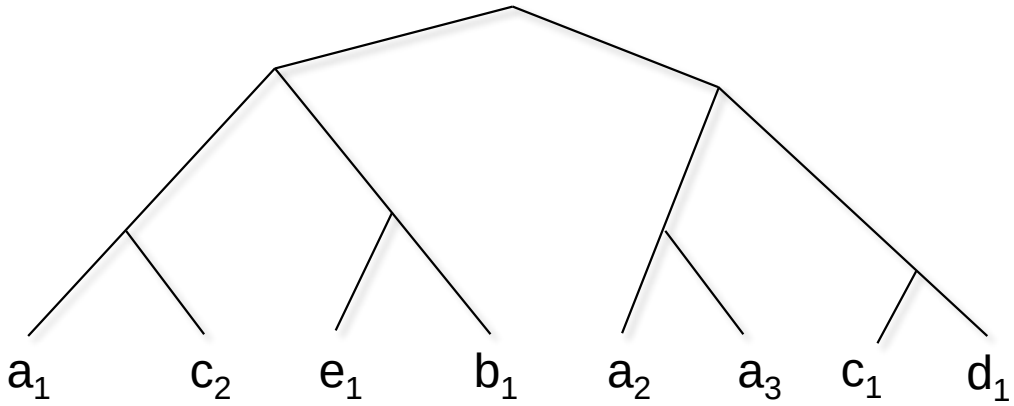
Possible reconciliation costs : **#dups**, #dups + #losses

The Supergenetree problem

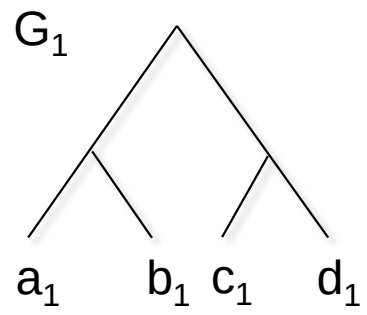


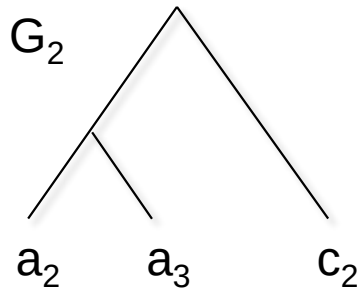


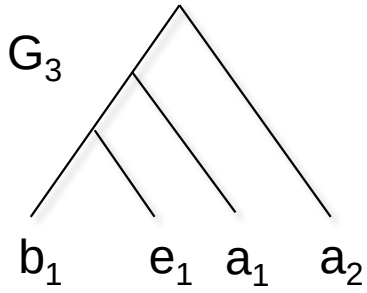


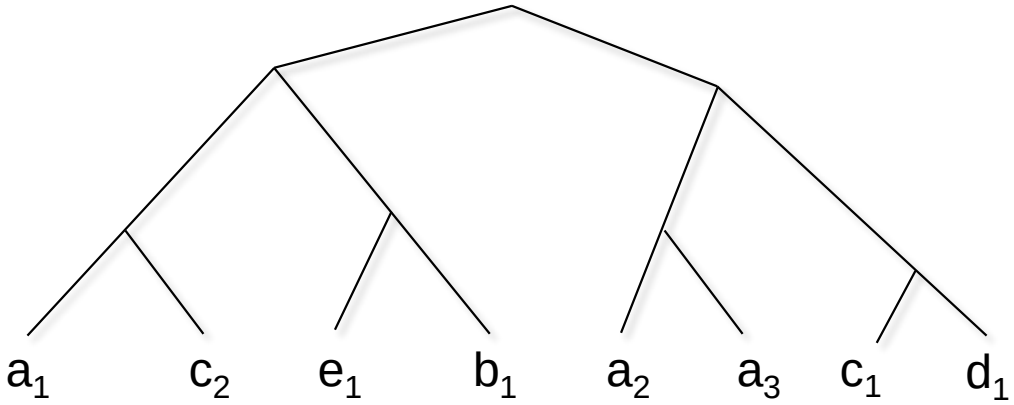


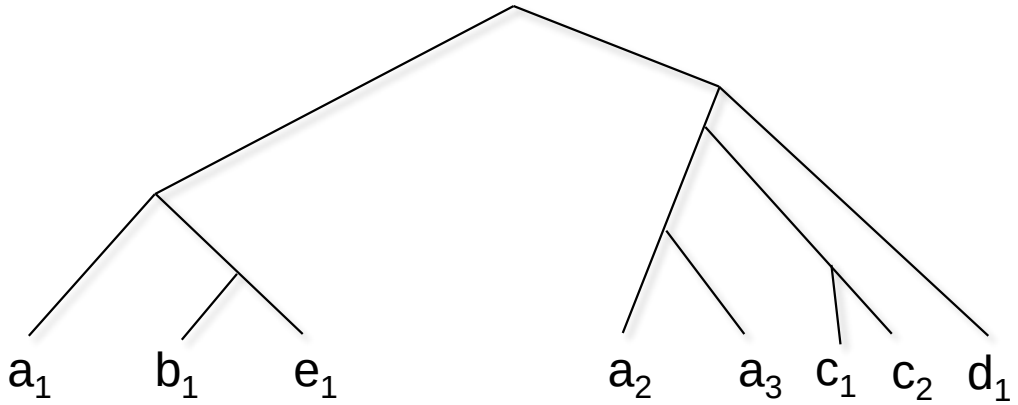
The Supergenetree problem



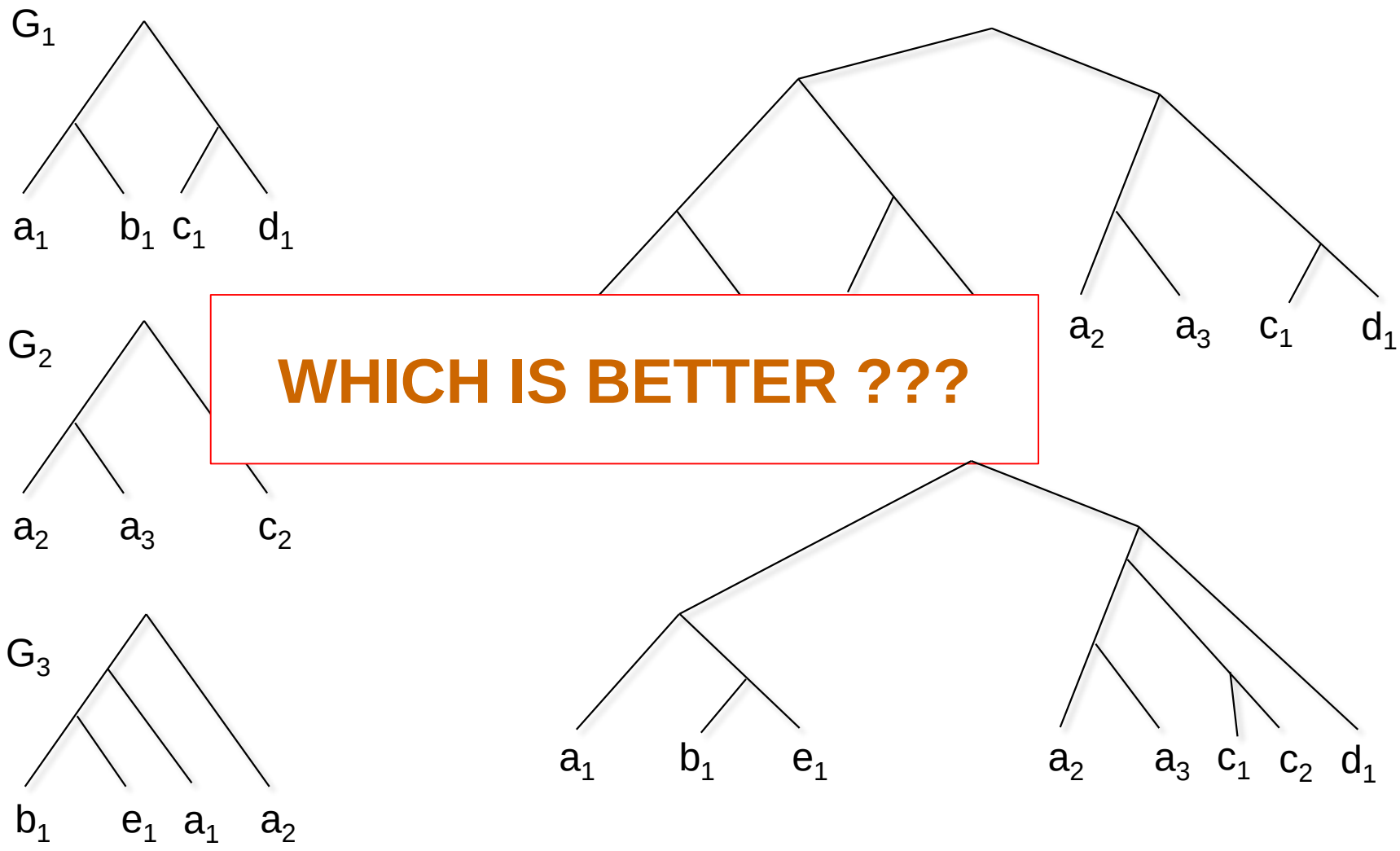




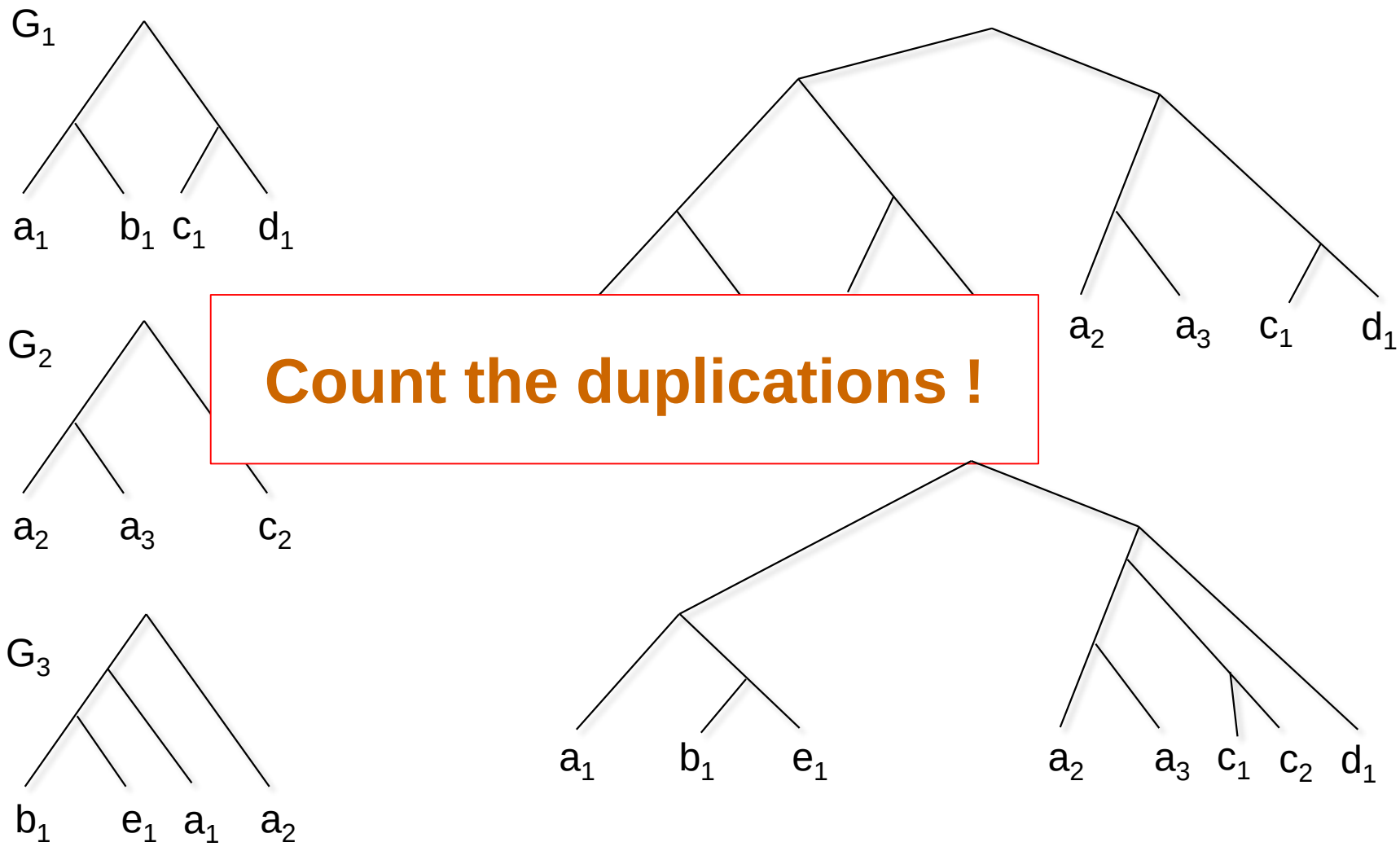




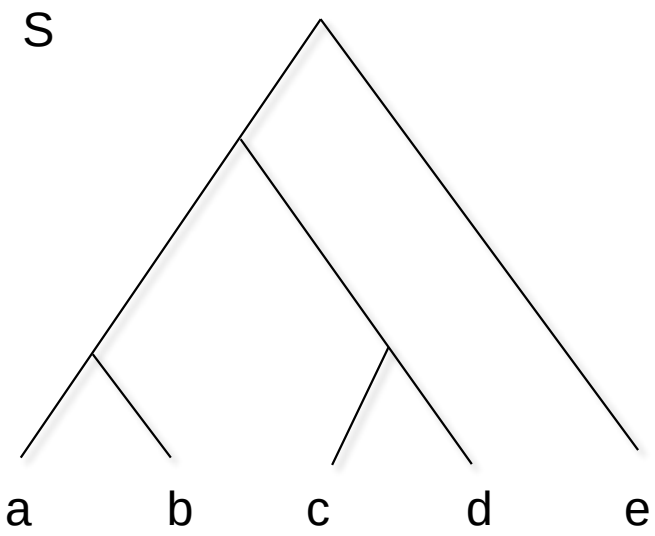
The Supergenetree problem

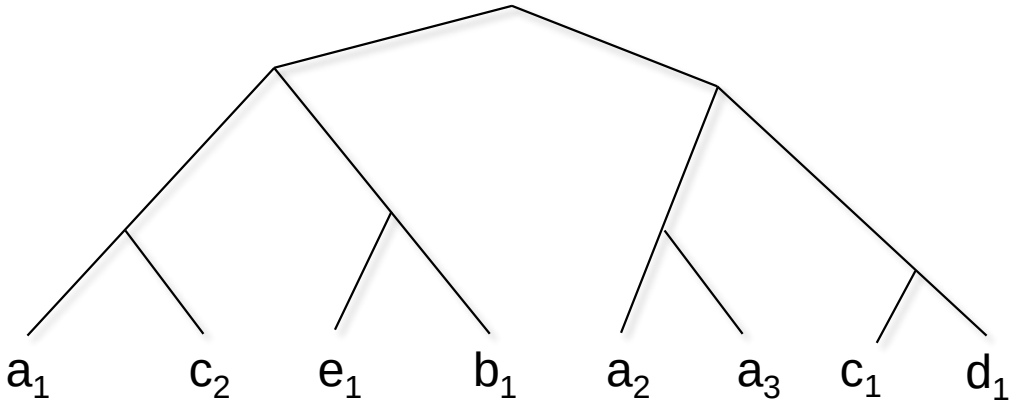


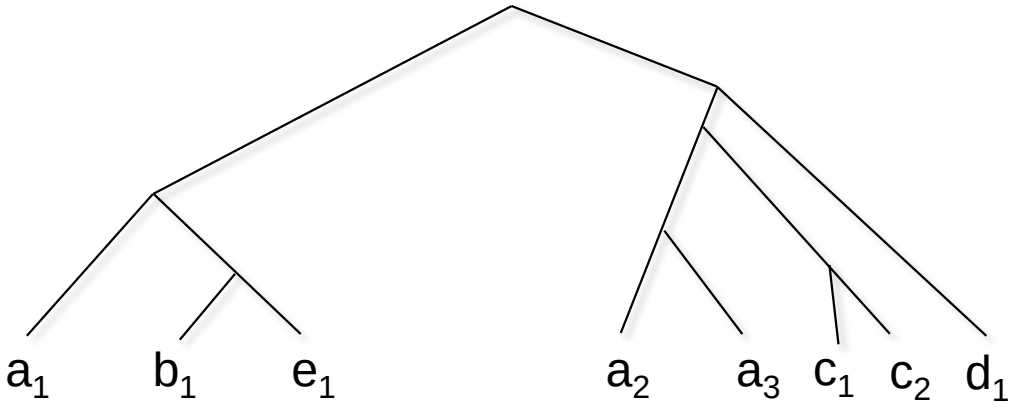
The Supergenetree problem



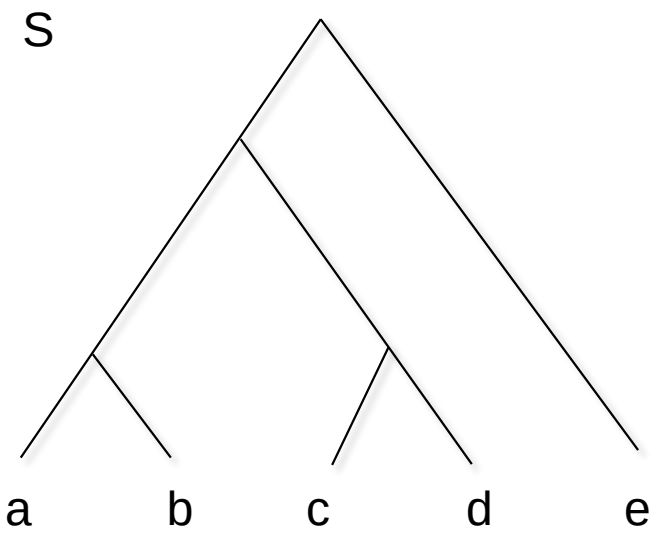
The Supergenetree problem

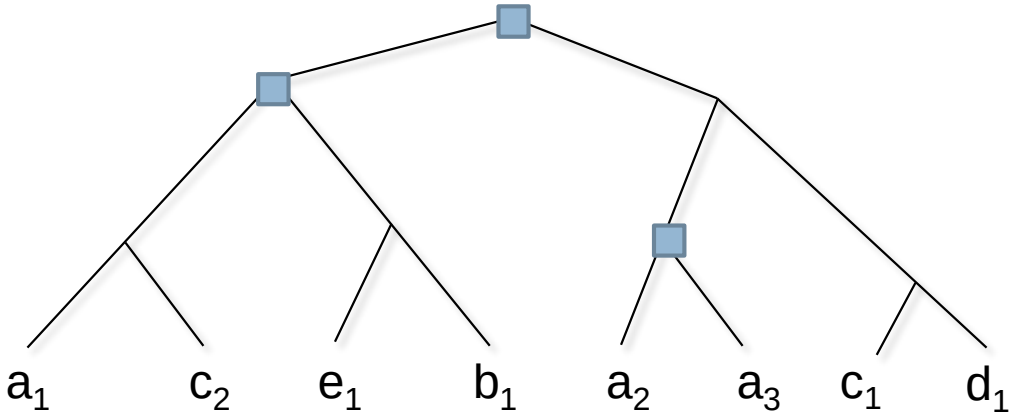


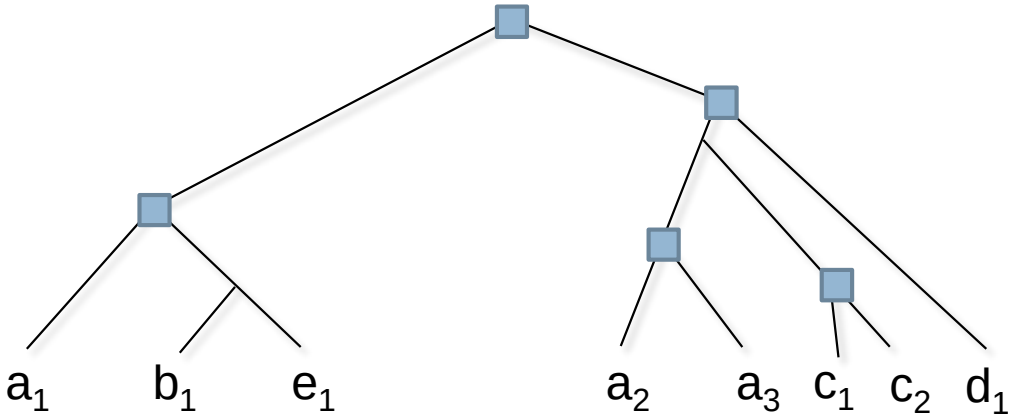




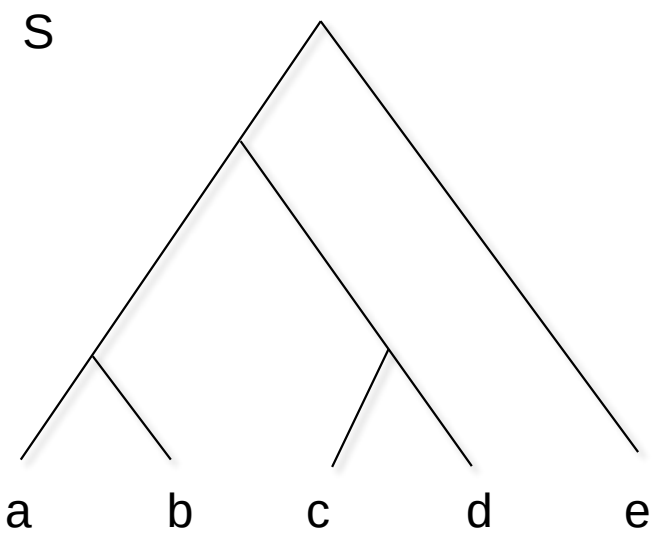
The Supergenetree problem

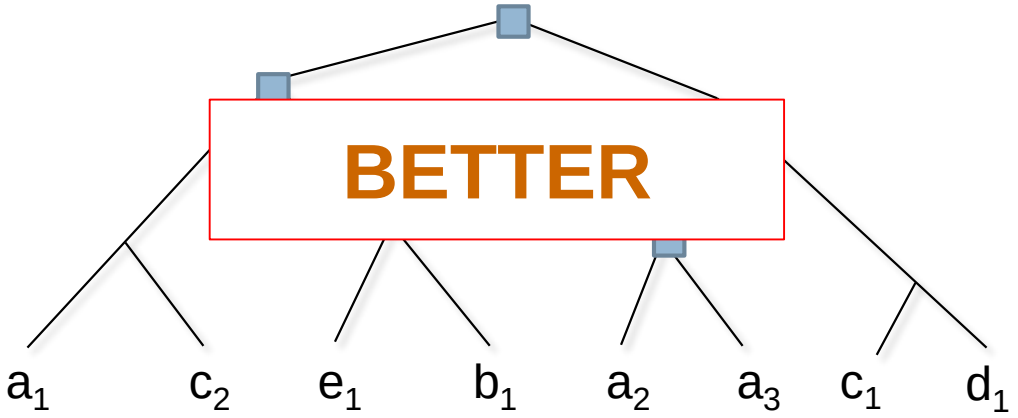


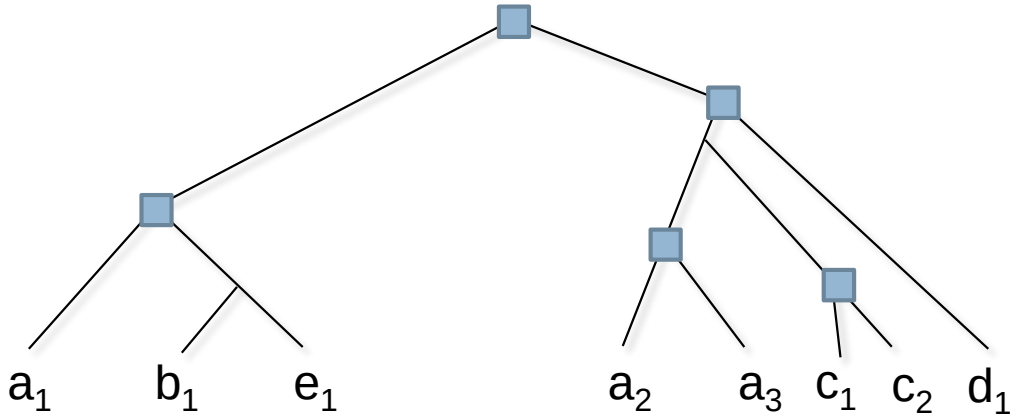




The Supergenetree problem







The plan

In this talk I…

- …come up with supertree problems
  - Finding a supergenetree that minimizes duplications
- …convince you that they're hard
- …try to do something about it
  - Exact, brute-force algorithm
  - A greedy heuristic

SuperGeneTree Problem 1

- **Given:** a set of compatible gene trees $G = \{G_1, \ldots, G_k\}$ and a species tree S
- **Find:** a SuperGeneTree G* that
  - displays every tree of G
  - minimizes #dups(G*, S)

SuperGeneTree Problem 1

- **Given:** a set of compatible gene trees $G = \{G_1, \ldots, G_k\}$ and a species tree S
- **Find:** a SuperGeneTree G* that
  - displays every tree of G
  - minimizes #dups(G*, S)

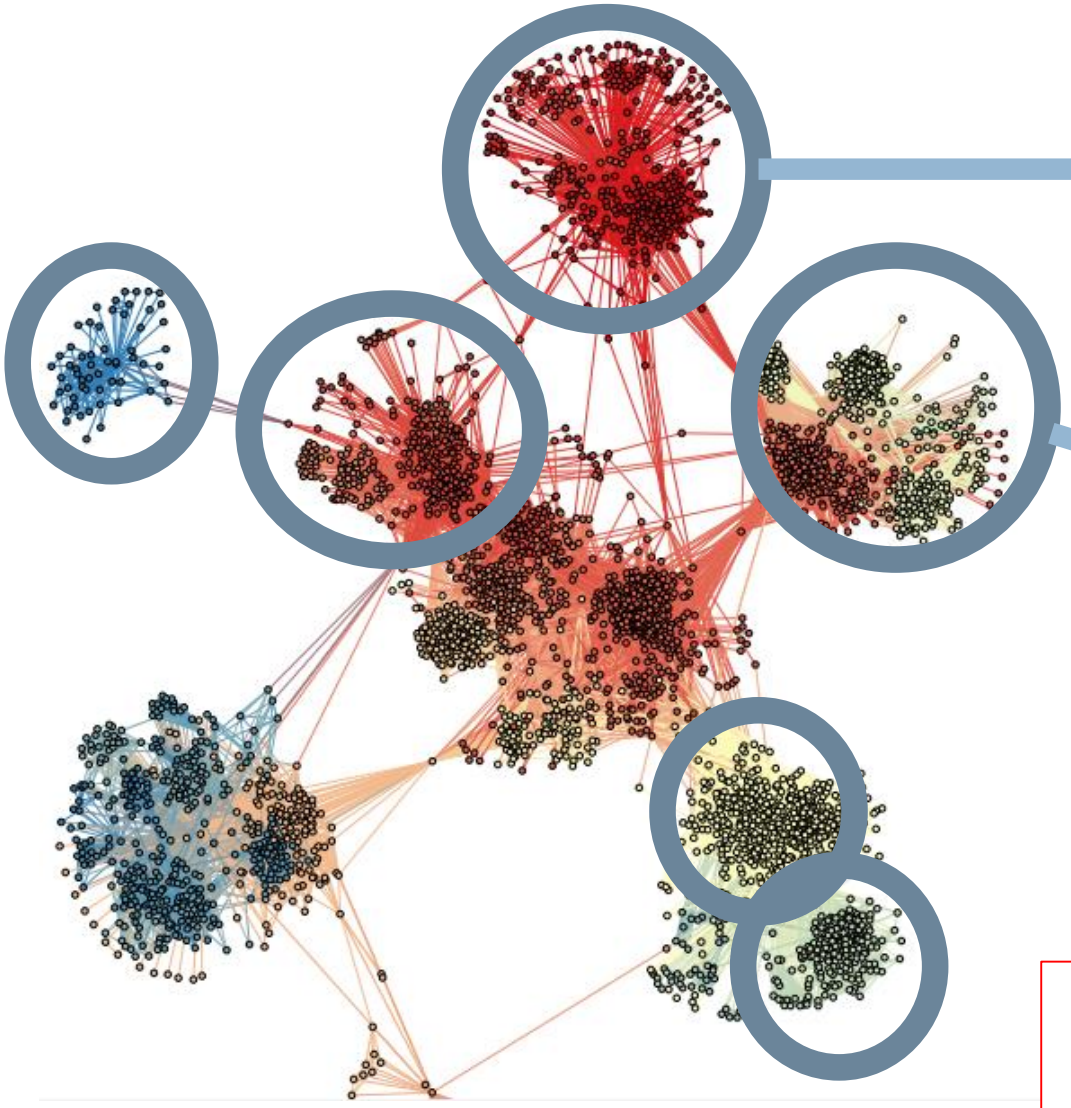
- NP-Complete

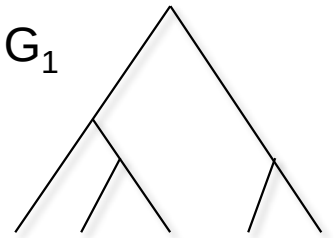
SuperGeneTree Problem 1

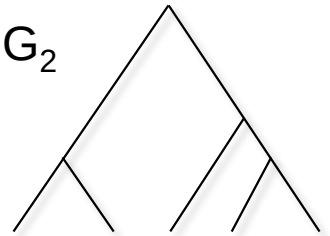
- **Given:** a set of compatible gene trees $G = \{G_1, \ldots, G_k\}$ and a species tree S
- **Find:** a SuperGeneTree G* that
  - displays every tree of G
  - minimizes #dups(G*, S)

- NP-Complete
- NP-Hard to approximate within a $n^{1-\varepsilon}$ factor

Independent speciation trees



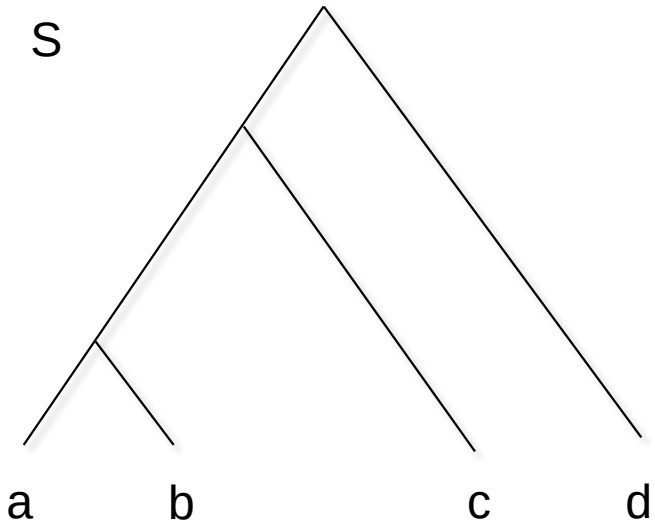


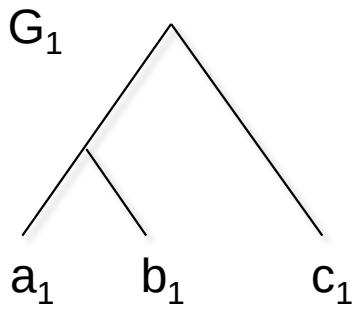


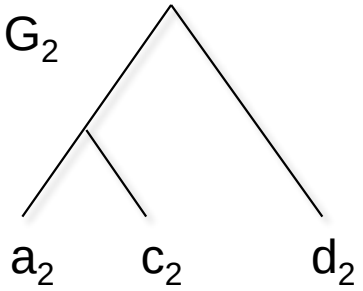
■ ■ ■

No two trees share a common gene + all trees of orthologous groups

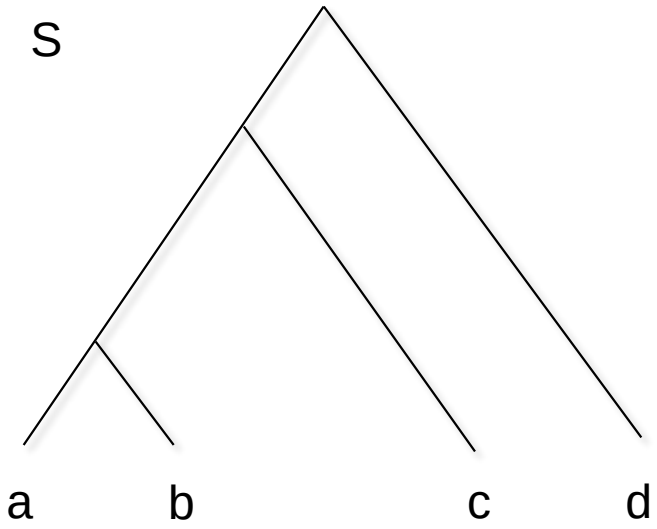
Independent speciation trees

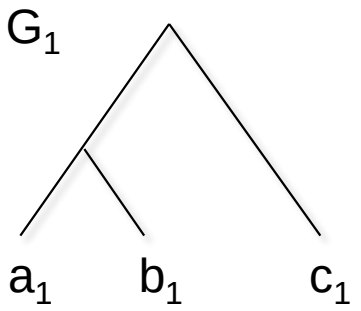


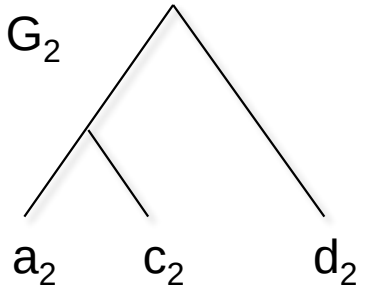


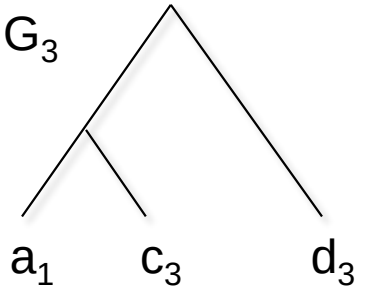


Independent speciation trees

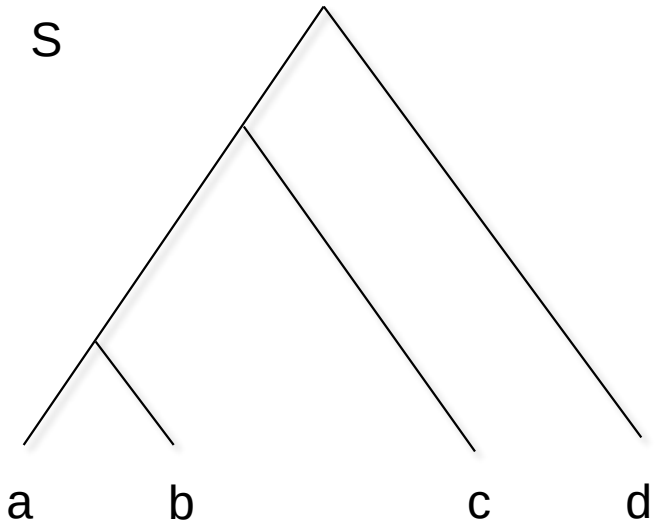


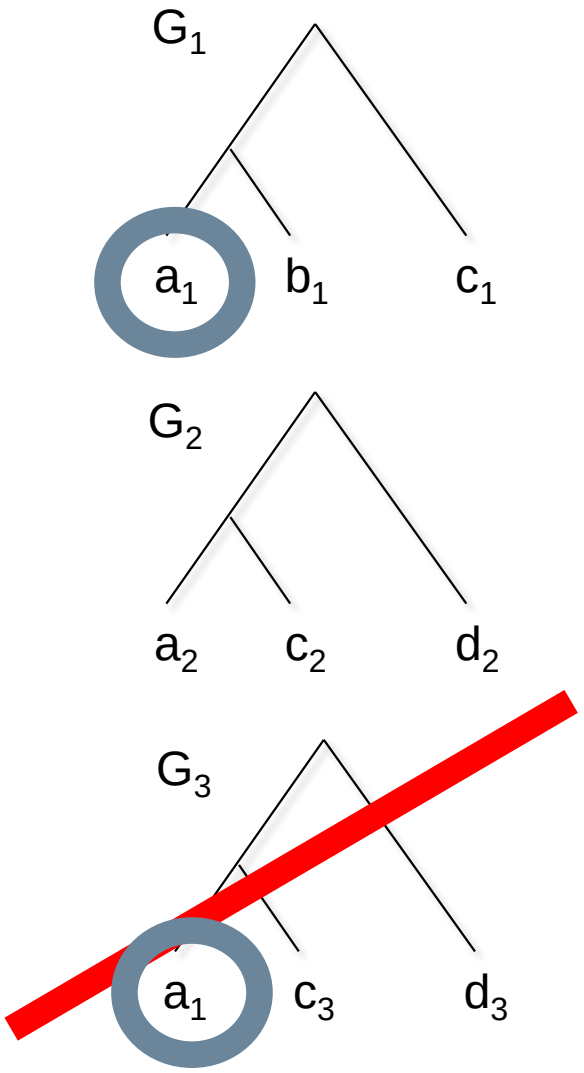






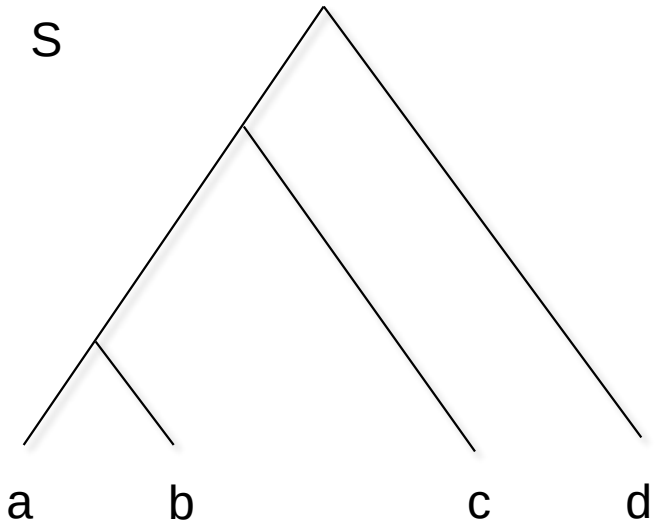
Independent speciation trees

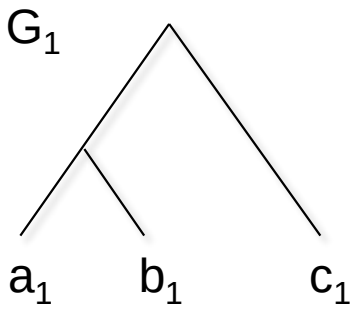


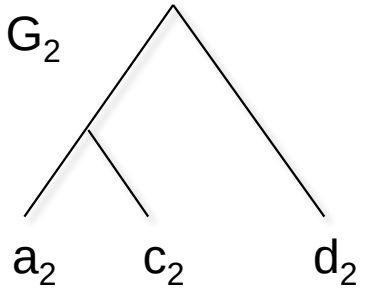


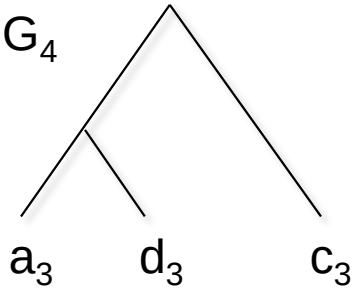
Independent = each gene appears only once

Independent speciation trees

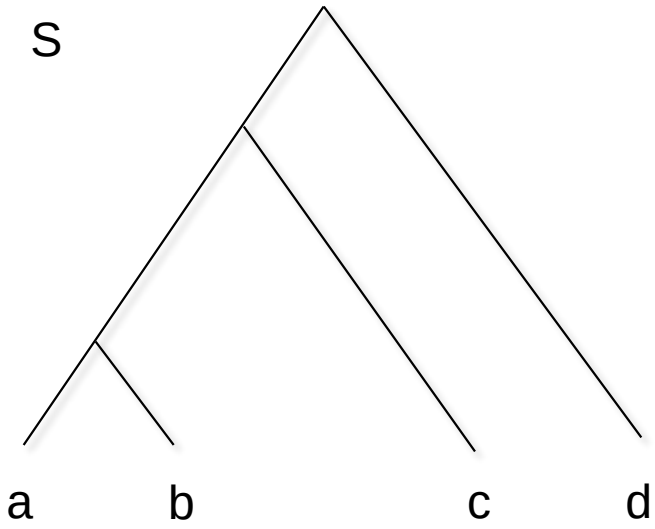


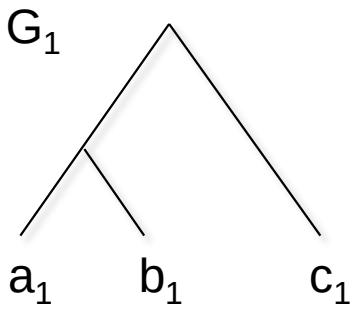


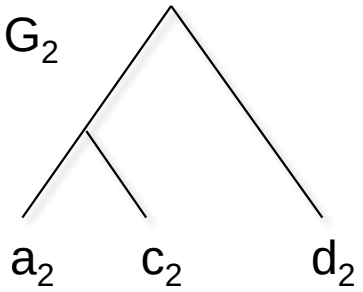


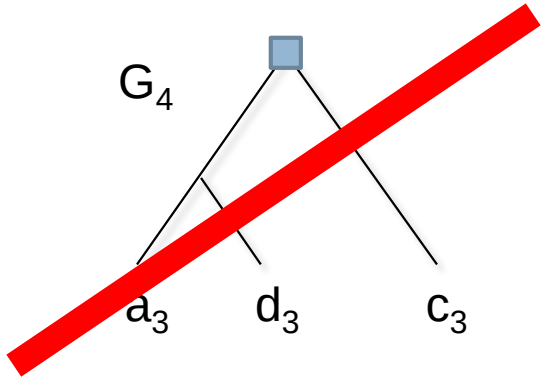


Independent speciation trees









Speciation trees = all speciation (all agree with S)

SuperGeneTree Problem 2

- **Given:** a set of **independent speciation** gene trees $G = \{G_1, \ldots, G_k\}$ and a species tree S
- **Find:** a SuperGeneTree G* that
  - displays every tree of G
  - minimizes #dups(G*, S)

SuperGeneTree Problem 2

- **Given:** a set of **independent speciation** gene trees $G = \{G_1, \ldots, G_k\}$ and a species tree S
- **Find:** a SuperGeneTree G* that
  - displays every tree of G
  - minimizes #dups(G*, S)

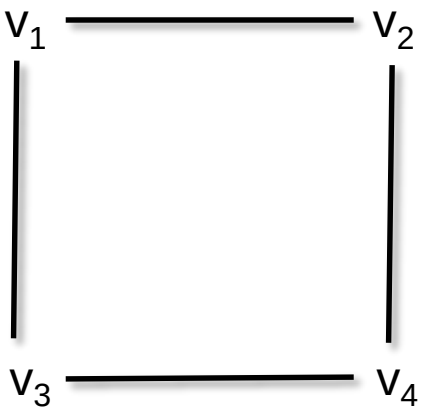
- NP-Complete

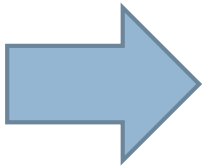
The plan

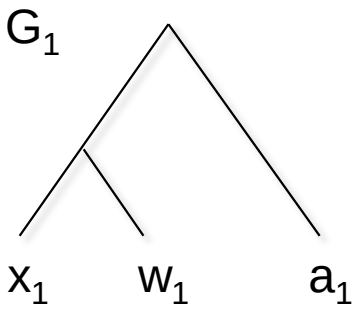
In this talk I…

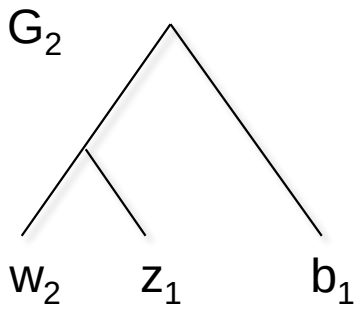
- …come up with supertree problems
  - Finding a supergenetree that minimizes duplications
- …convince you that they're hard
- …try to do something about it
  - Exact, brute-force algorithm
  - A greedy heuristic

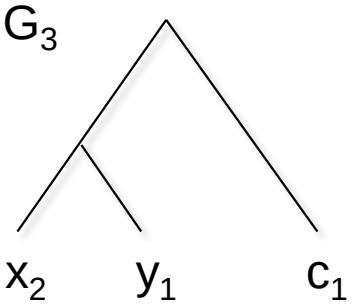
What is so hard about it ?

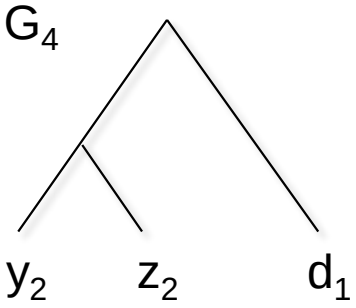








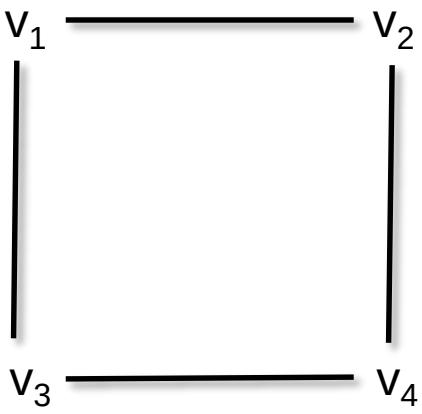


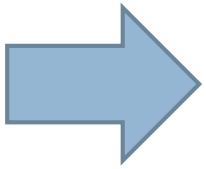


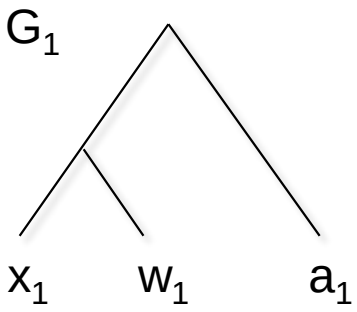
We will find a vertex-coloring of our graph

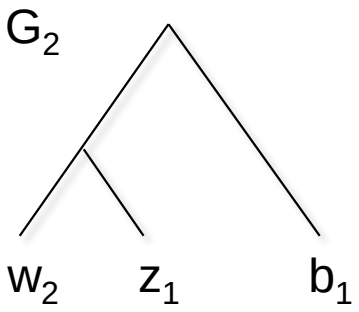
(a partition into independent sets)

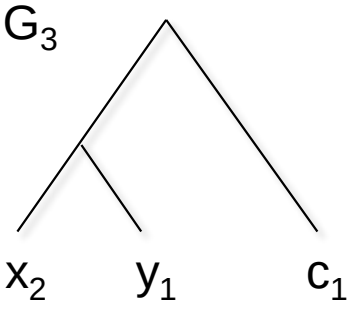
What is so hard about it ?

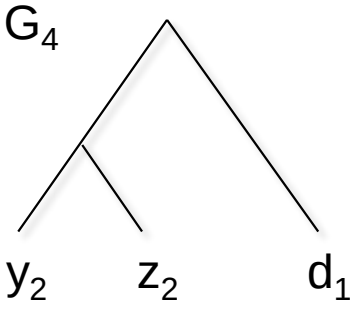










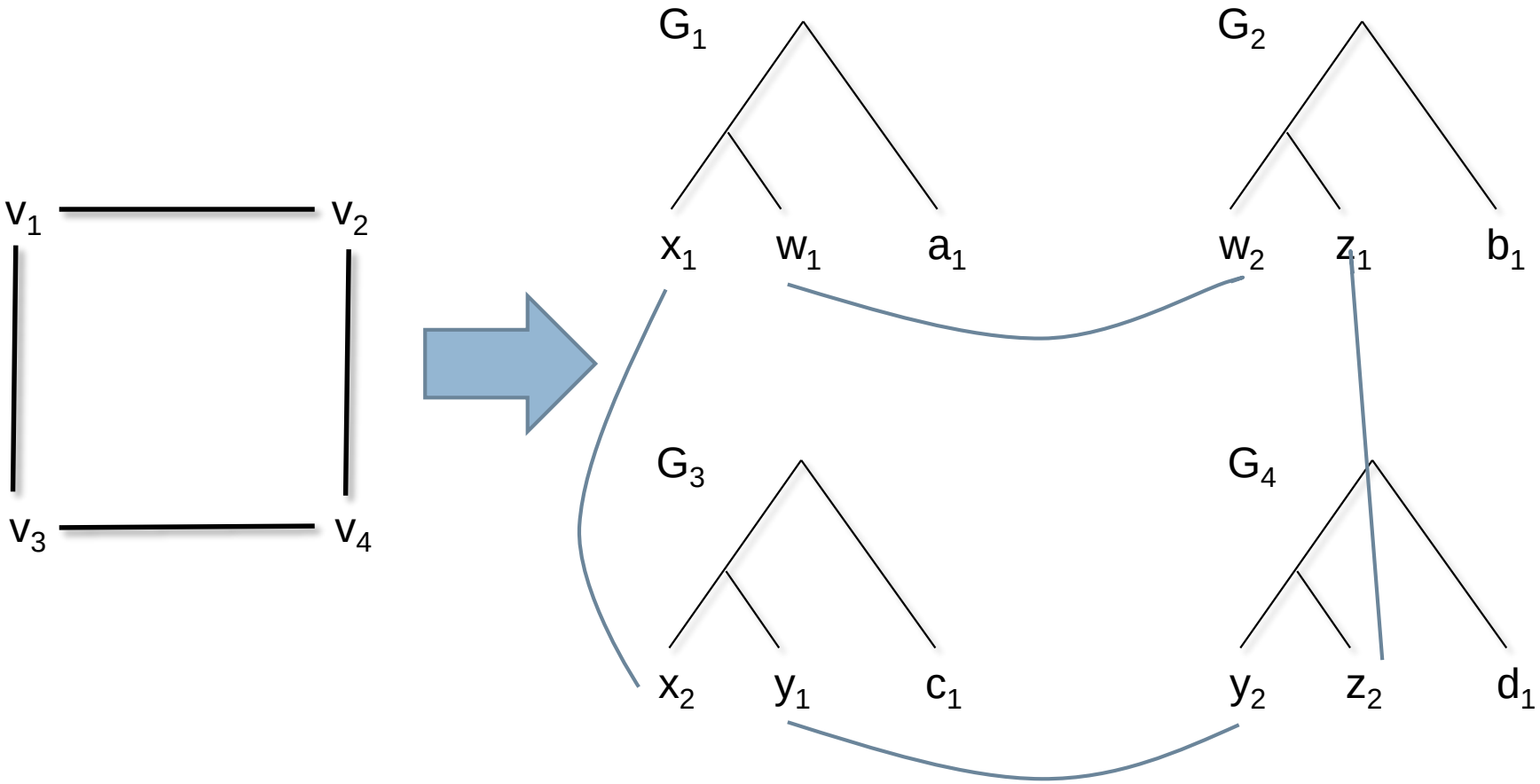


G_i, G_j **share a gene from the same species (i.e. a label)** iff v_i, v_j **share an edge**

⇔

G_i, G_j can be merged into a supertree **without duplications** iff v_i, v_j **share no edge**

What is so hard about it ?

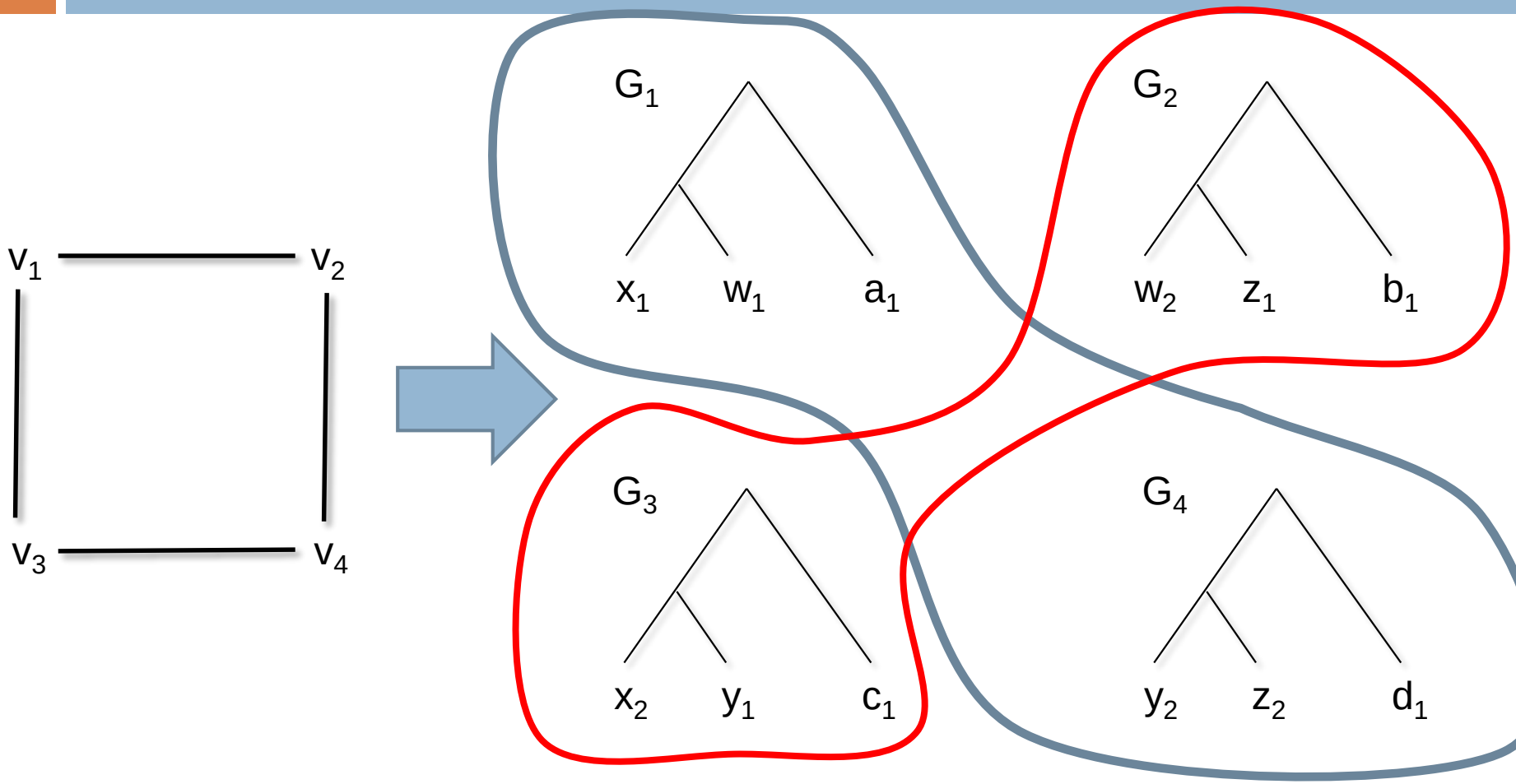


G_i, G_j **share a gene from the same species (i.e. a label)** iff v_i, v_j **share an edge**

⇔

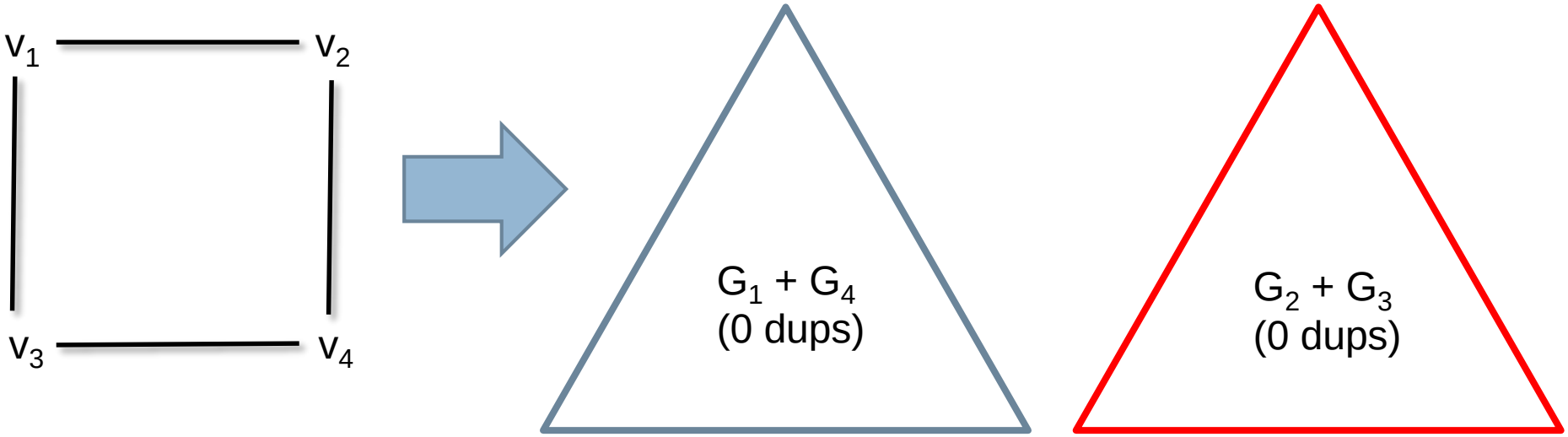
G_i, G_j can be merged into a supertree **without duplications** iff v_i, v_j **share no edge**

What is so hard about it ?



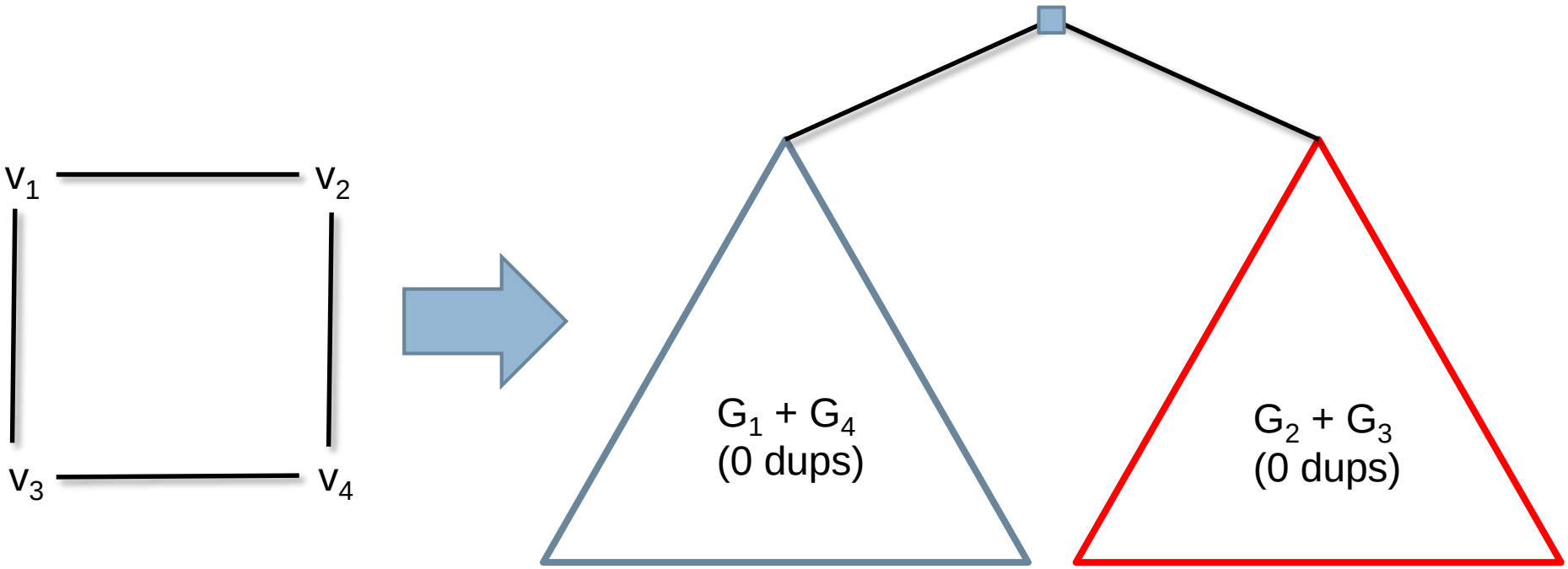
A best solution partitions the trees into **k** sets of trees that **all share no "label"**

What is so hard about it ?



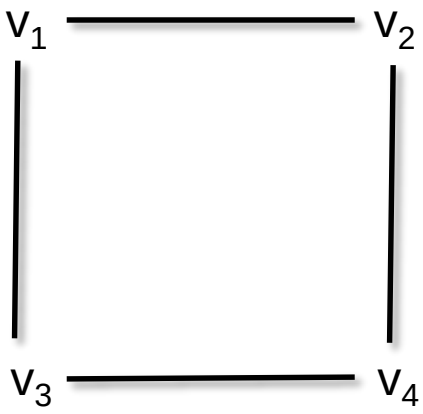
A best solution partitions the trees into **k** sets of trees that **all share no "label"**
Makes one zero-duplication tree for each part.

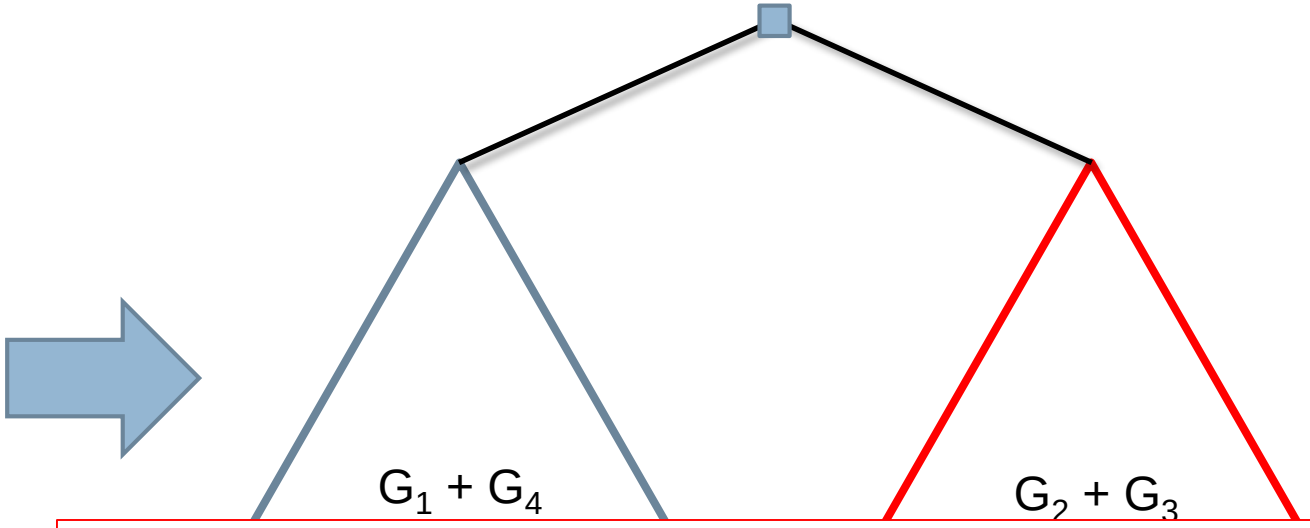
What is so hard about it ?



A best solution partitions the trees into **k** sets of trees that **all share no "label"**
Makes one zero-duplication tree for each part.
Connects these k subtrees with at most k – 1 duplications.

What is so hard about it ?

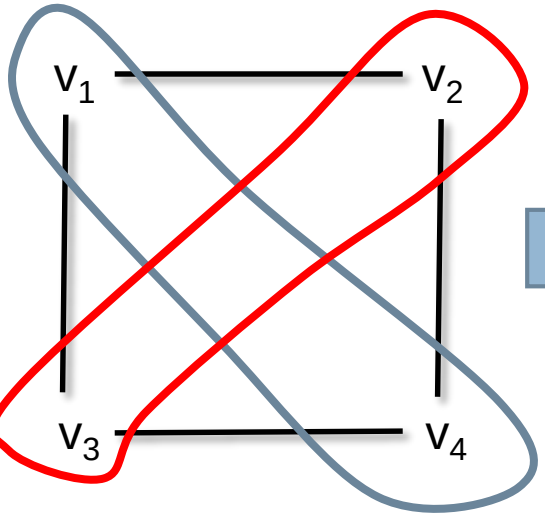


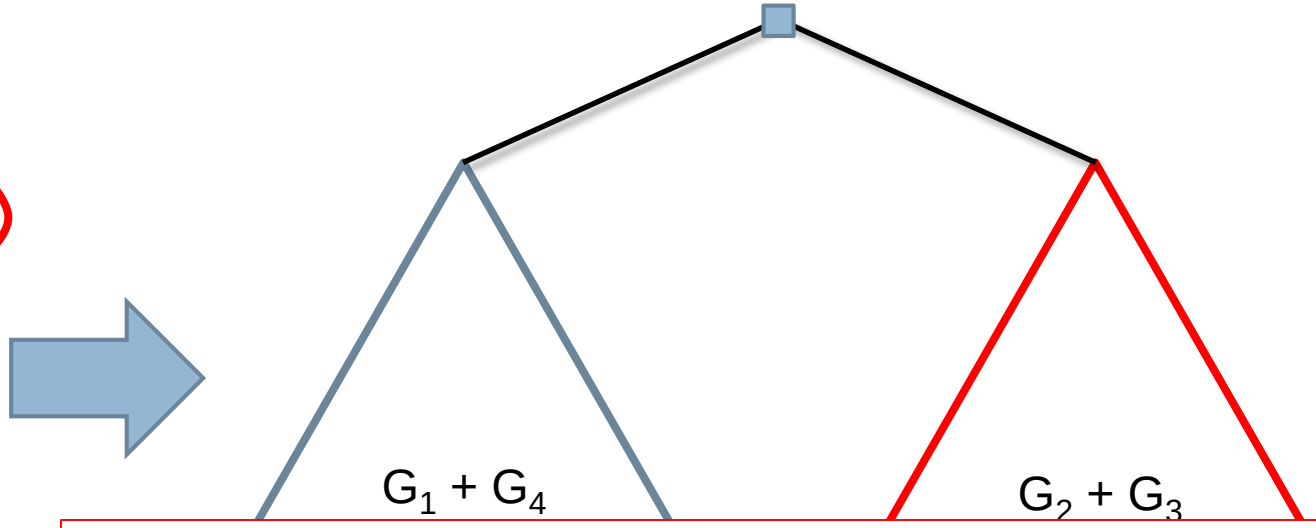


This is a partition of the vertices of our graph into independent sets, i.e. a vertex-coloring !

A best solution partitions the trees into **k** sets of trees that **all share no "label"**
Makes one zero-duplication tree for each part.
Connects these k subtrees with at most k – 1 duplications.

What is so hard about it ?





This is a partition of the vertices of our graph into independent sets, i.e. a vertex-coloring !

A best solution partitions the trees into **k** sets of trees that **all share no "label"**
Makes one zero-duplication tree for each part.
Connects these k subtrees with at most k – 1 duplications.

The plan

In this talk I…

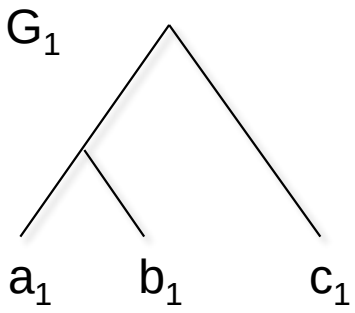
- …come up with supertree problems
  - Finding a supergenetree that minimizes duplications
- …convince you that they're hard
- …try to do something about it
  - Exact, brute-force algorithm
  - A greedy heuristic

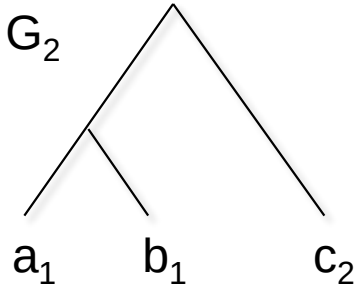
Extending the BUILD algorithm

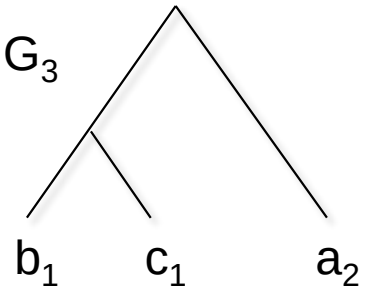
- Given a set of trees G, the BUILD algorithm outputs, if it exists, a supertree T displaying every tree of G
  - T might be partially resolved (non-binary)
  - Every binary resolution of T displays G

- BUILD can be extended to output **every** supertree displaying G + every minimally resolved *(Constantinescu & Sankoff, 1995, Ng & Wormald, 1996, Semple, 2003)*

Extending the BUILD algorithm



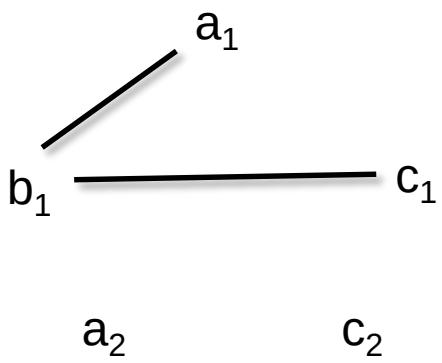




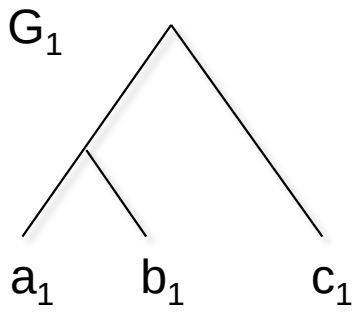
BUILD graph

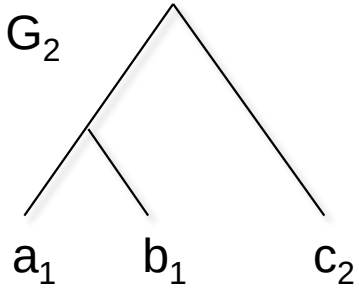
vertices = genes

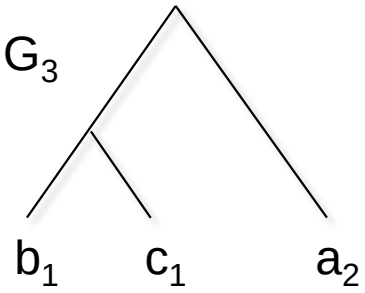
edges = genes together in some triplet



Extending the BUILD algorithm



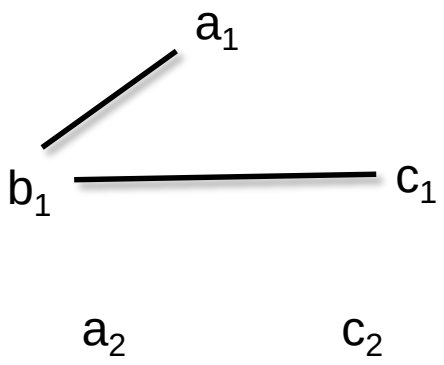




BUILD graph

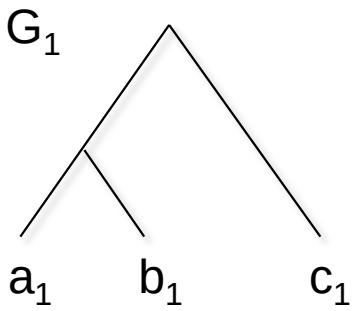
vertices = genes

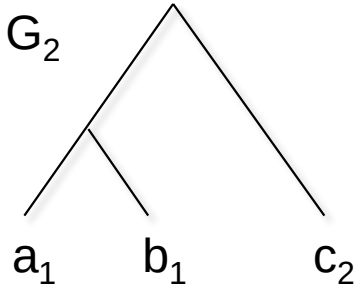
edges = genes together in some triplet

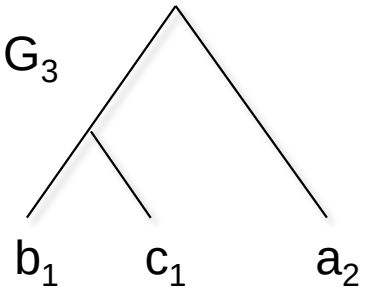


Partition of connected components = possible splits at the root

Extending the BUILD algorithm



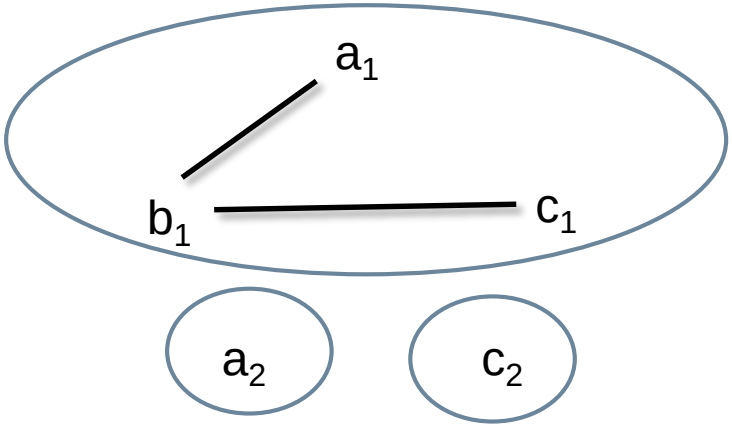




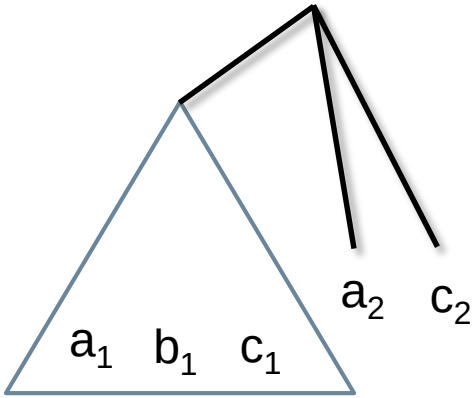
BUILD graph

vertices = genes

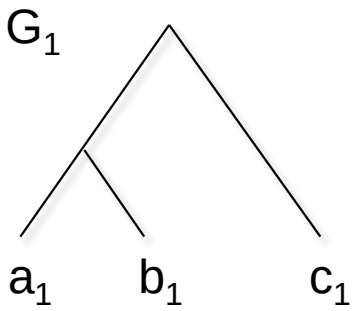
edges = genes together in some triplet

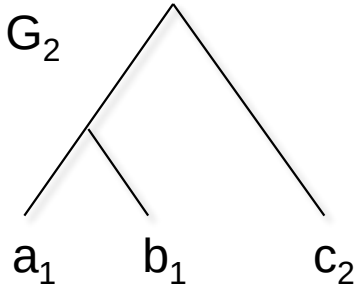


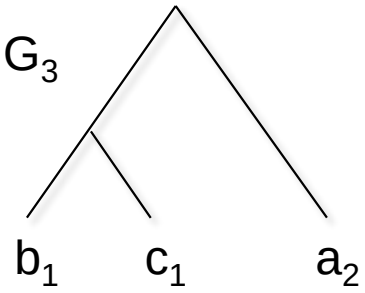
Partition of connected components = possible splits at the root



Extending the BUILD algorithm



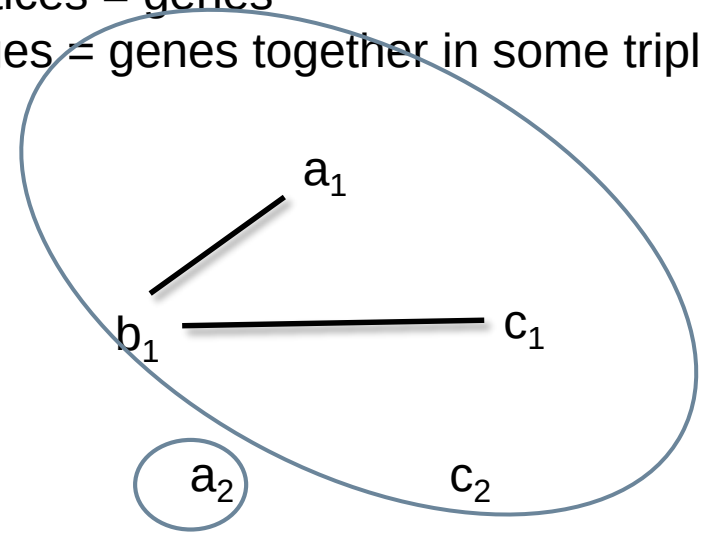




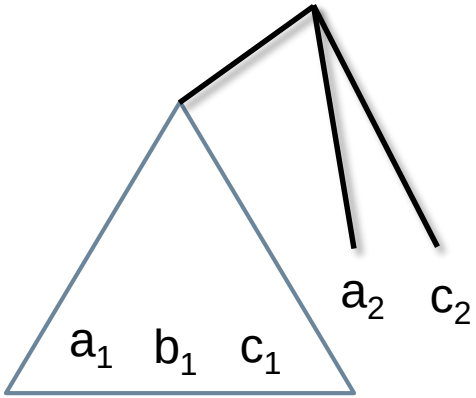
BUILD graph

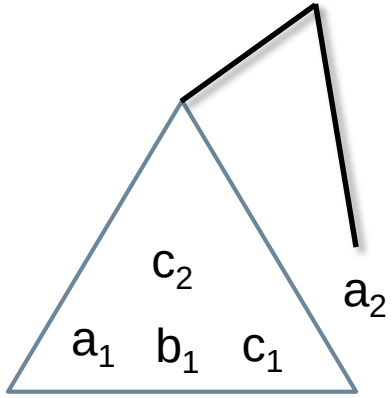
vertices = genes

edges = genes together in some triplet

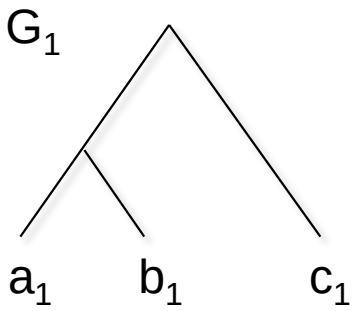


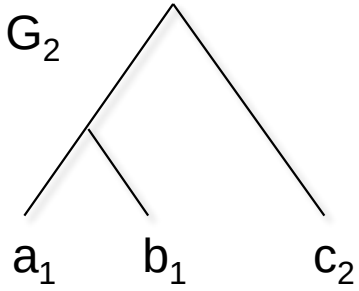
Partition of connected components = possible splits at the root

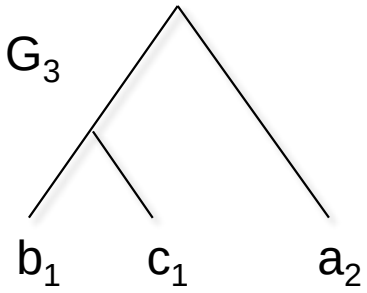




Extending the BUILD algorithm



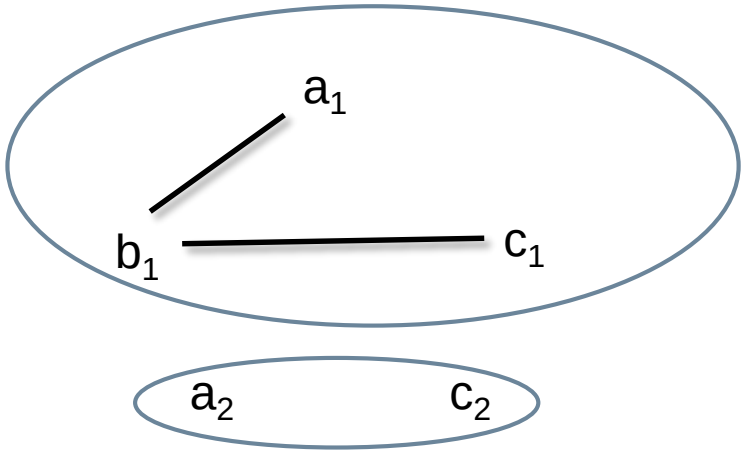




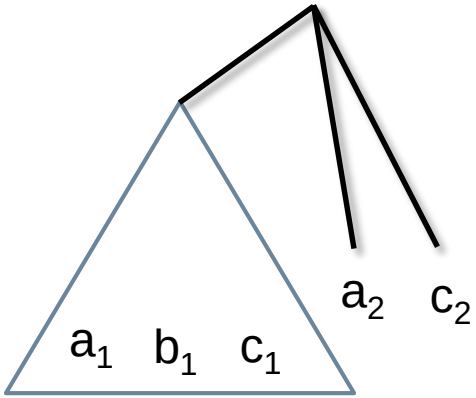
BUILD graph

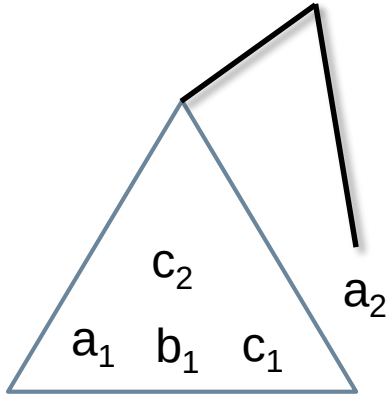
vertices = genes

edges = genes together in some triplet



Partition of connected components = possible splits at the root





...

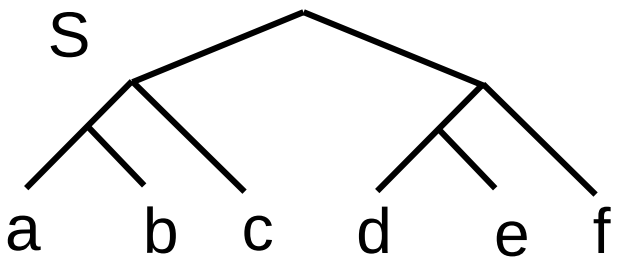
Extending the BUILD algorithm

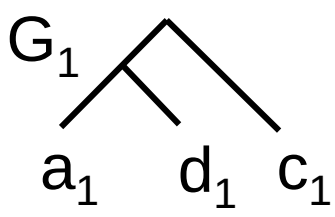
- For every partially unresolved tree T obtained in this fashion :
  - Find a resolution that minimizes the number of duplications *(linear time, Lafond & al. 2012)*
- In the worst case, there are $\Omega(n^{n/2})$ trees to resolve *(Jansson, Lemence, Lingas, 2012)*.
  - Total time : $\Omega(n * n^{n/2})$
- Worst case in practice : ?

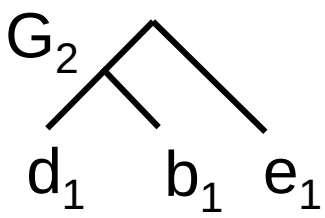
Extending the BUILD algorithm

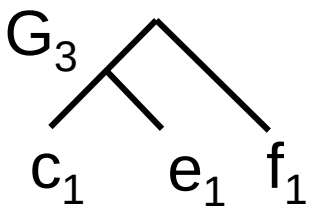
- Trying every partition of the components can take some time.
- Instead, let’s find a way to choose a partition that "looks good".

A greedy approach

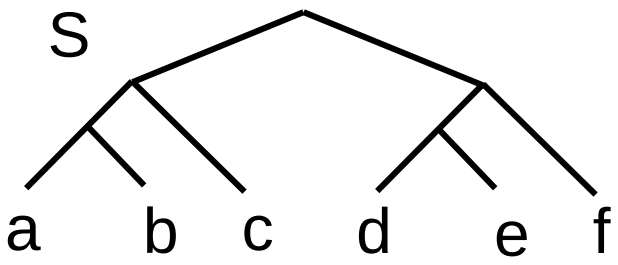


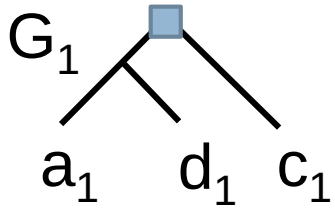


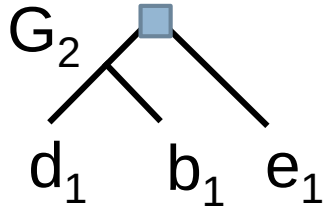


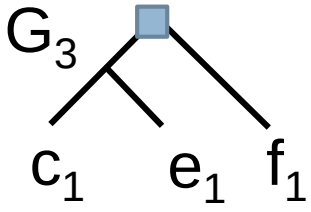


A greedy approach



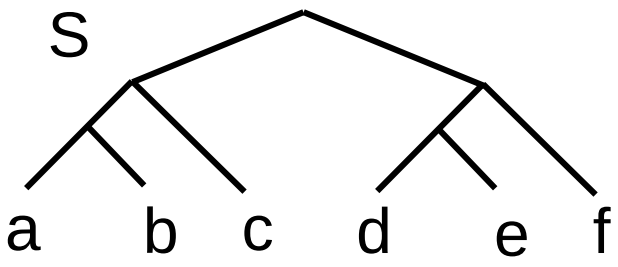


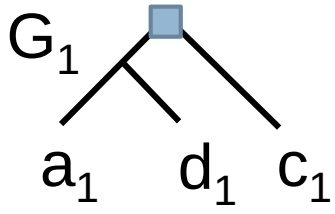


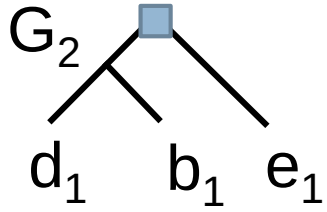


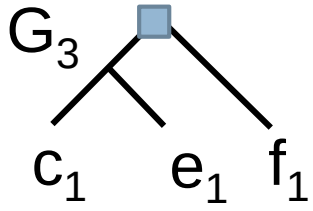
We already know that some duplications will be required.

A greedy approach



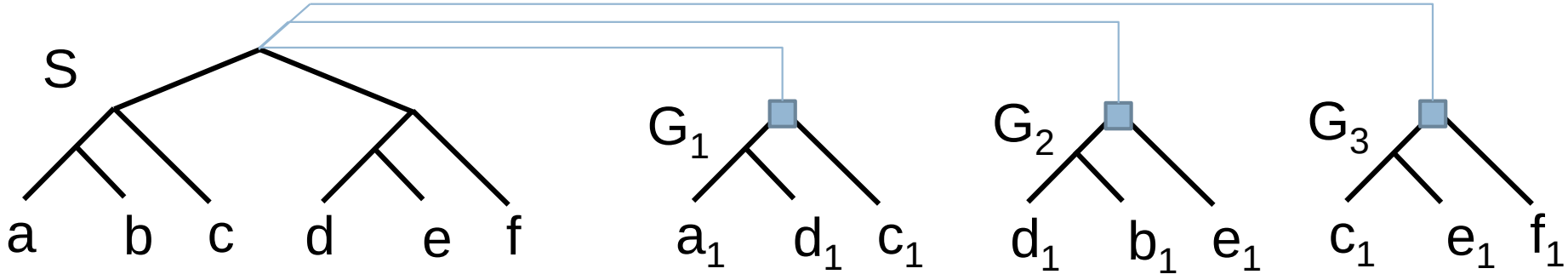






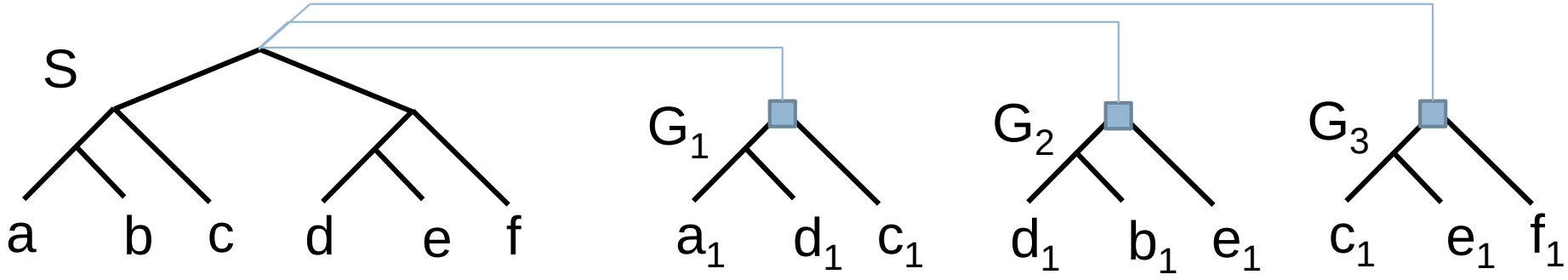
We already know that some duplications will be required.
Focus on the "highest" ones, i.e. those that occur before the first speciation in S.

A greedy approach



We already know that some duplications will be required.
Focus on the "highest" ones, i.e. those that occur before the first speciation in S.

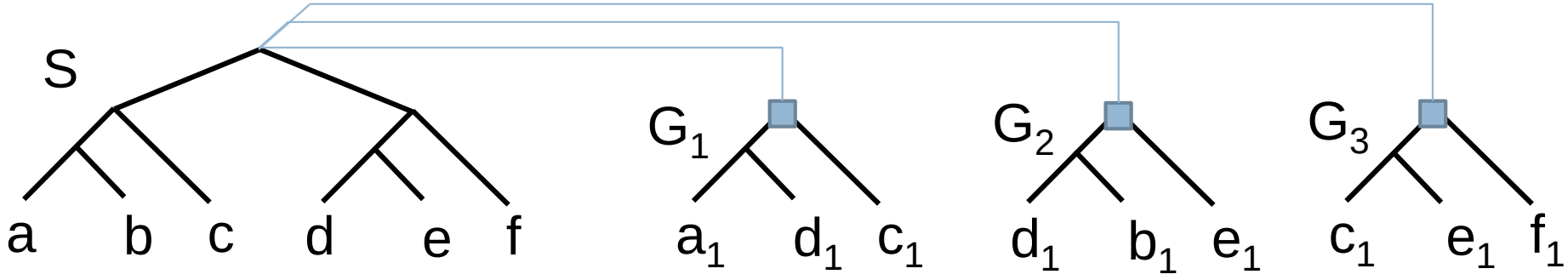
A greedy approach



We already know that some duplications will be required.
Focus on the "highest" ones, i.e. those that occur before the first speciation in S.

We call those duplication Pre Speciation Duplications (PreSpecDups).

A greedy approach

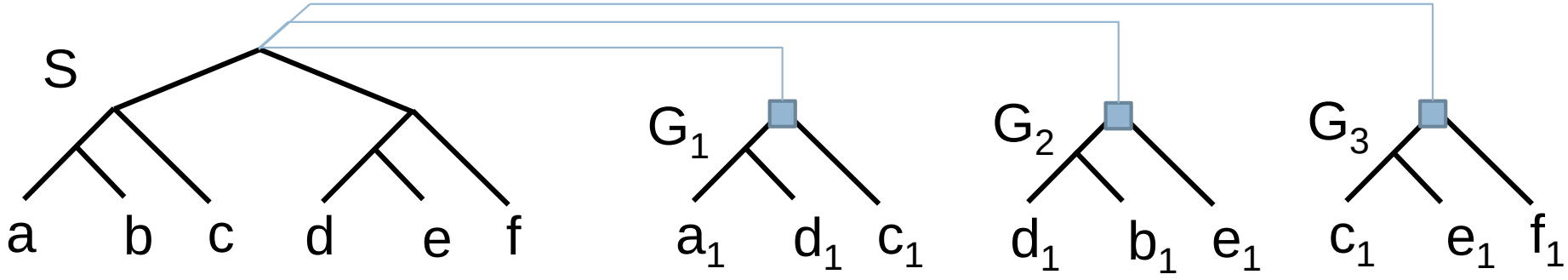


We already know that some duplications will be required.
Focus on the "highest" ones, i.e. those that occur before the first speciation in S.

We call those duplication Pre Speciation Duplications (PreSpecDups).

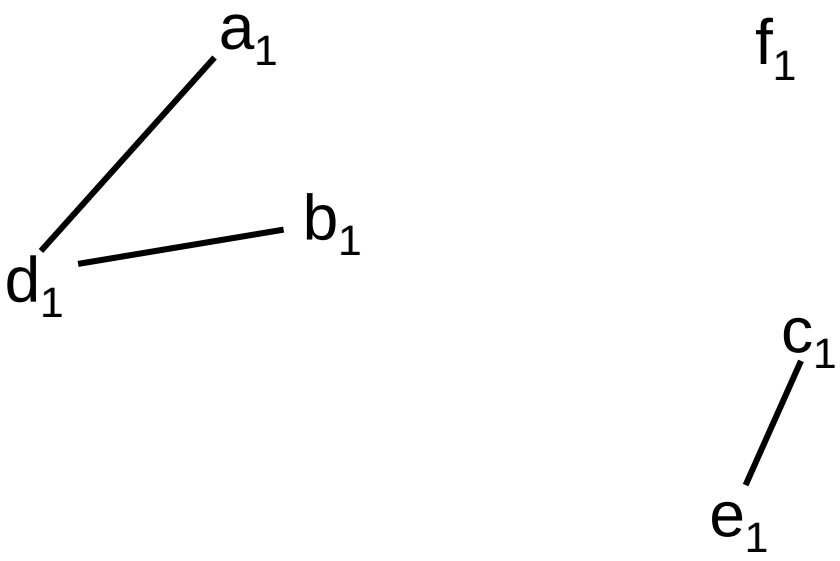
New subproblem : minimize only these PreSpecDups

A greedy approach

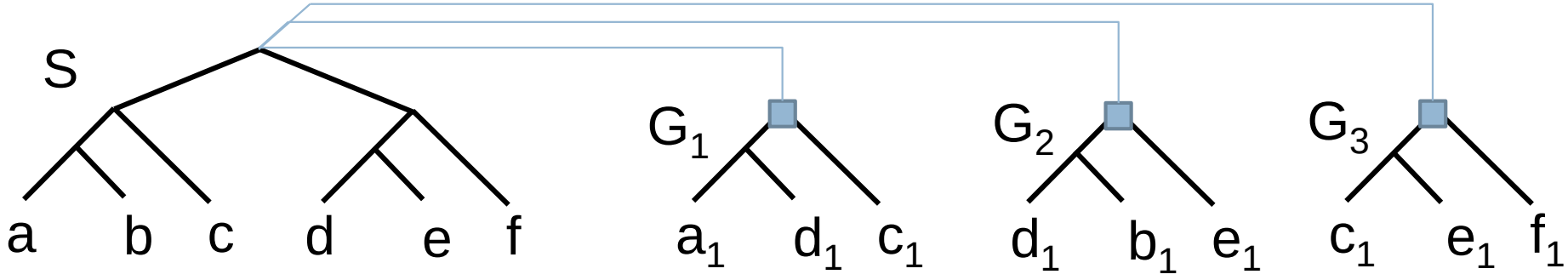


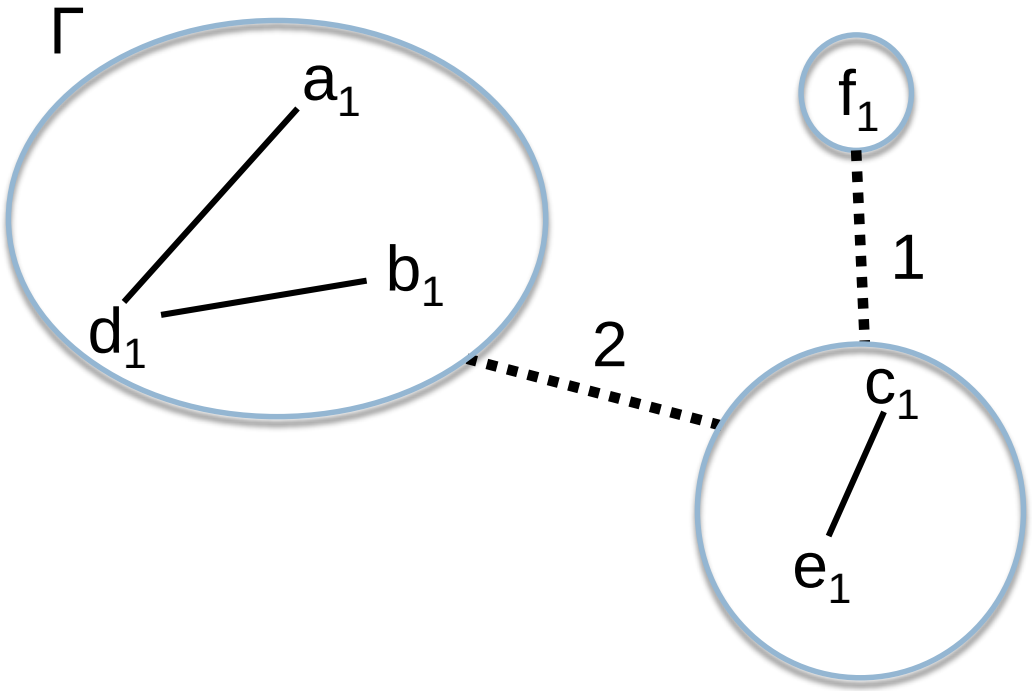
- Make the BUILD graph and identify the components.

Γ



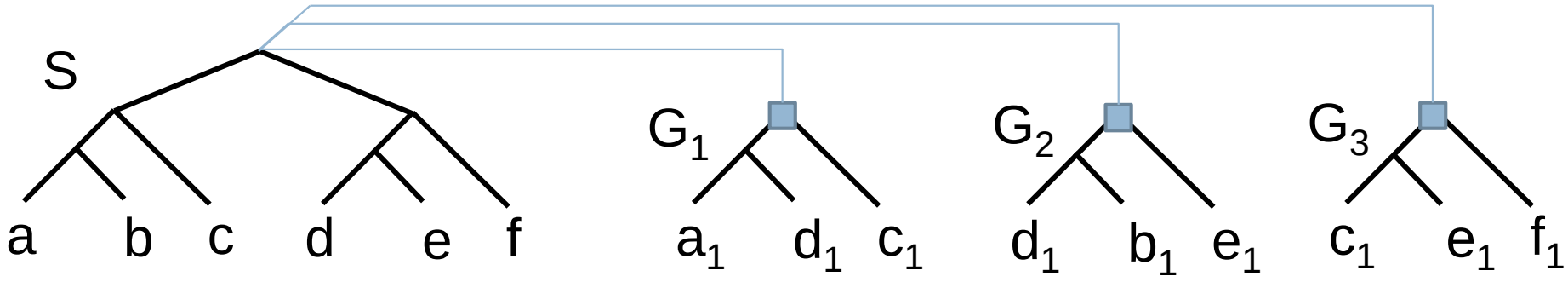
A greedy approach

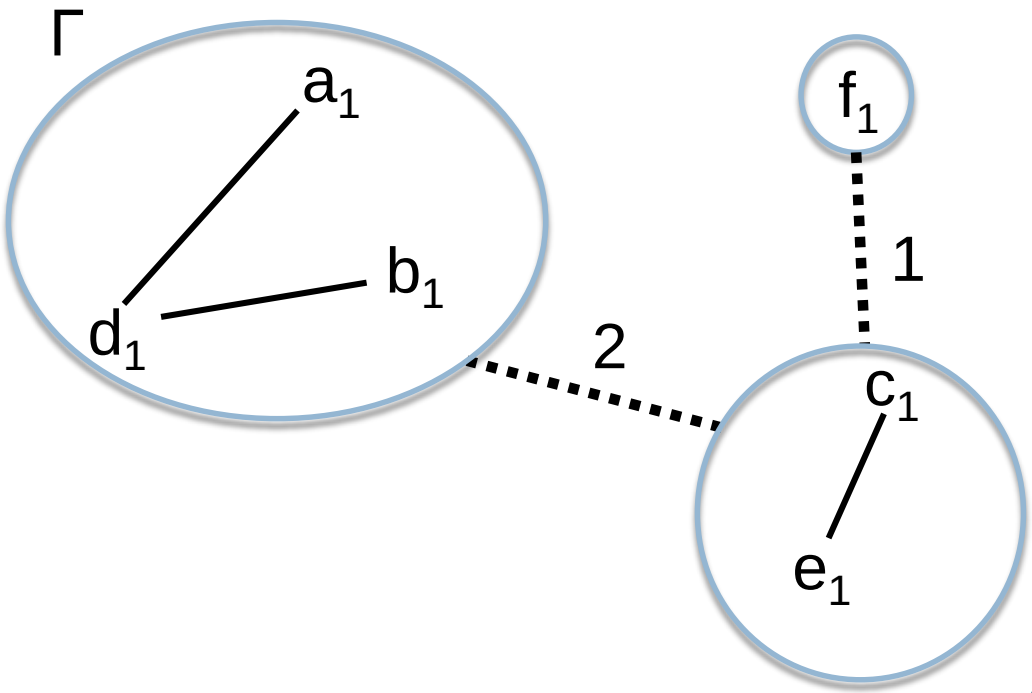




- Make the BUILD graph and identify the components.
- Add a special edge between components that requires a PreSpecDup when split.

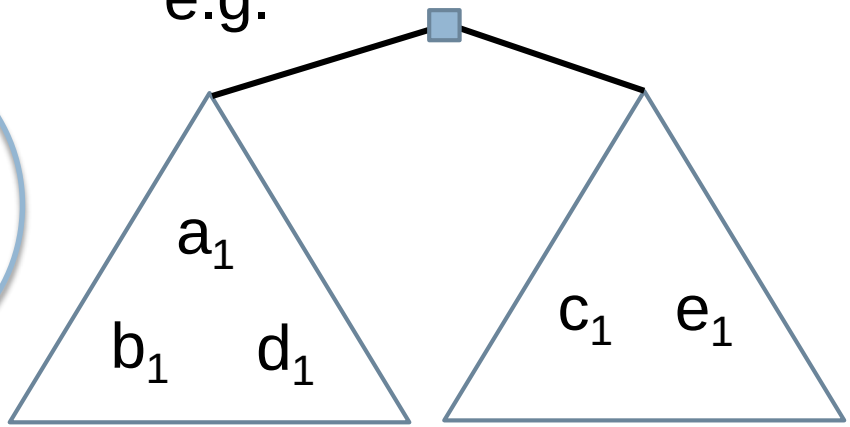
A greedy approach



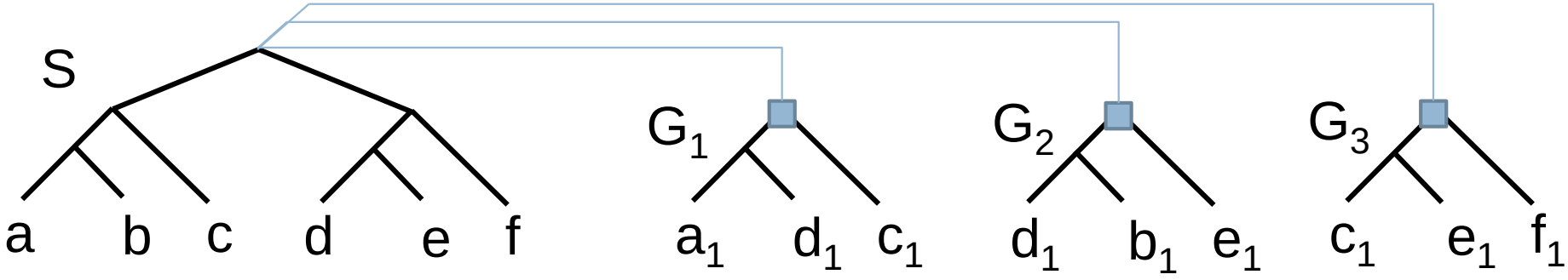


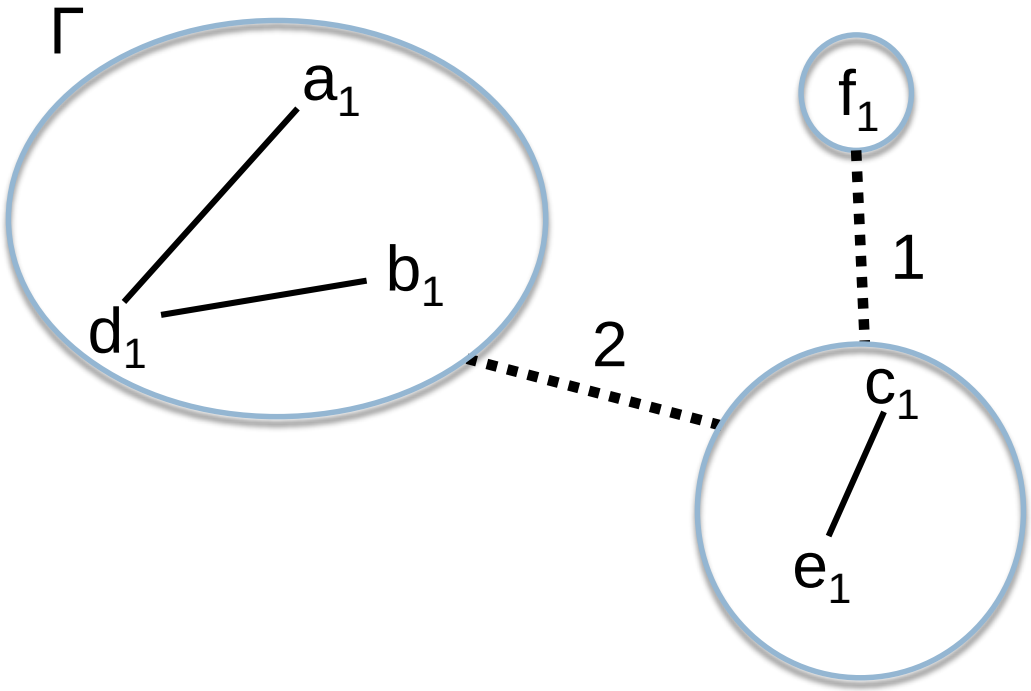
- Make the BUILD graph and identify the components.
- Add a special edge between components that requires a PreSpecDup when split.

e.g.



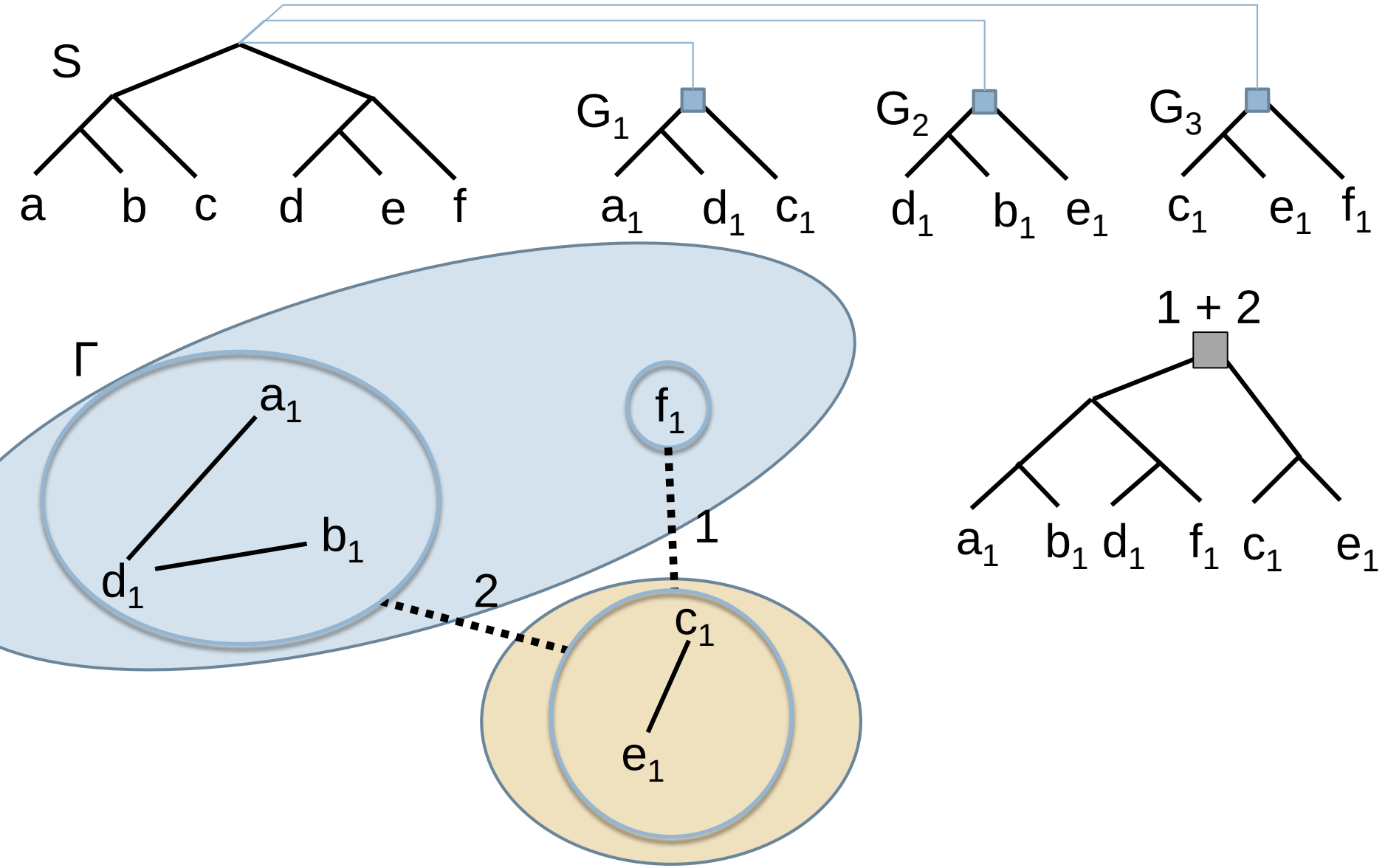
A greedy approach



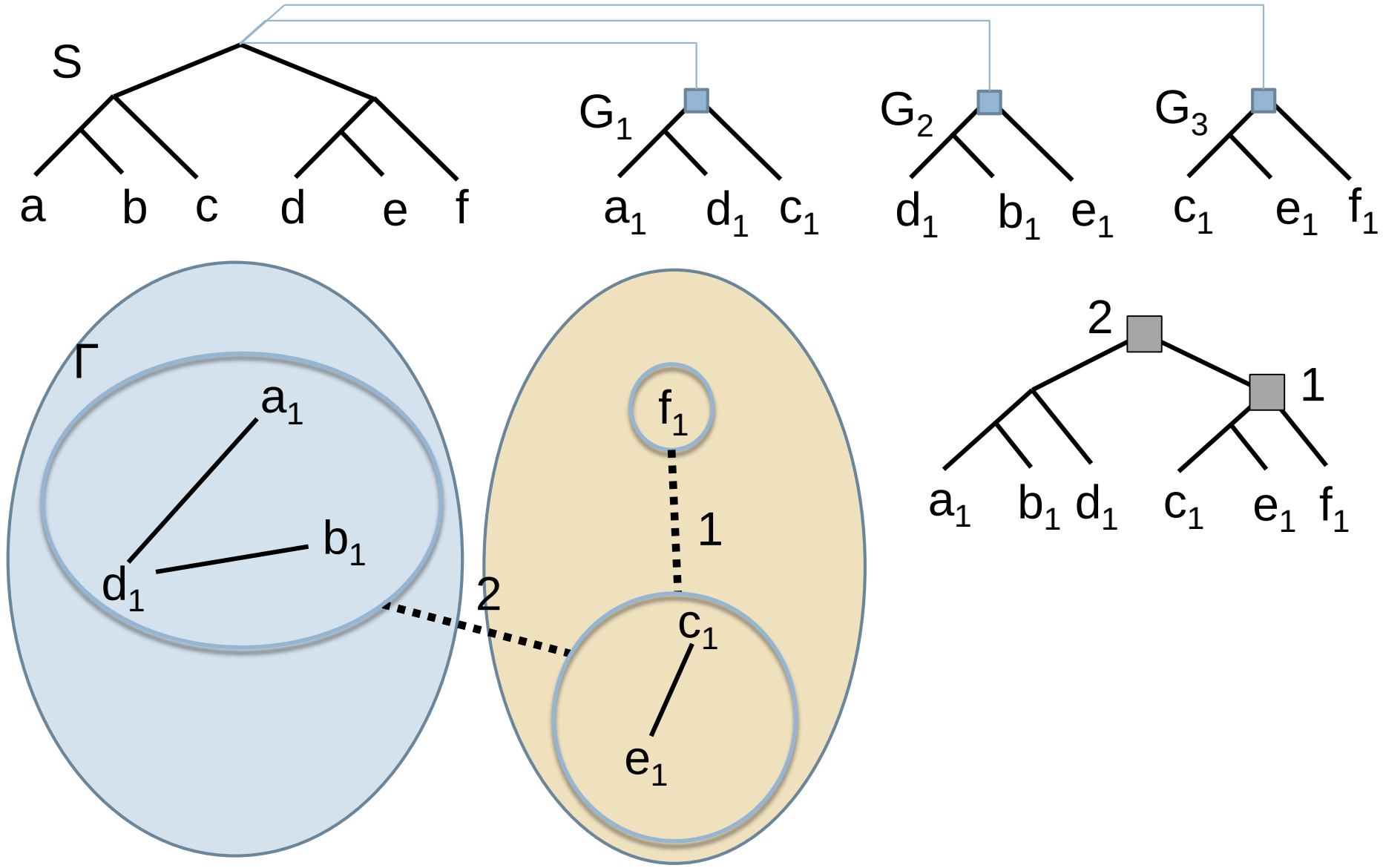


- Make the BUILD graph and identify the components.
- Add a special edge between components that requires a PreSpecDup when split.
- Find the partition that merges a maximum of duplications.

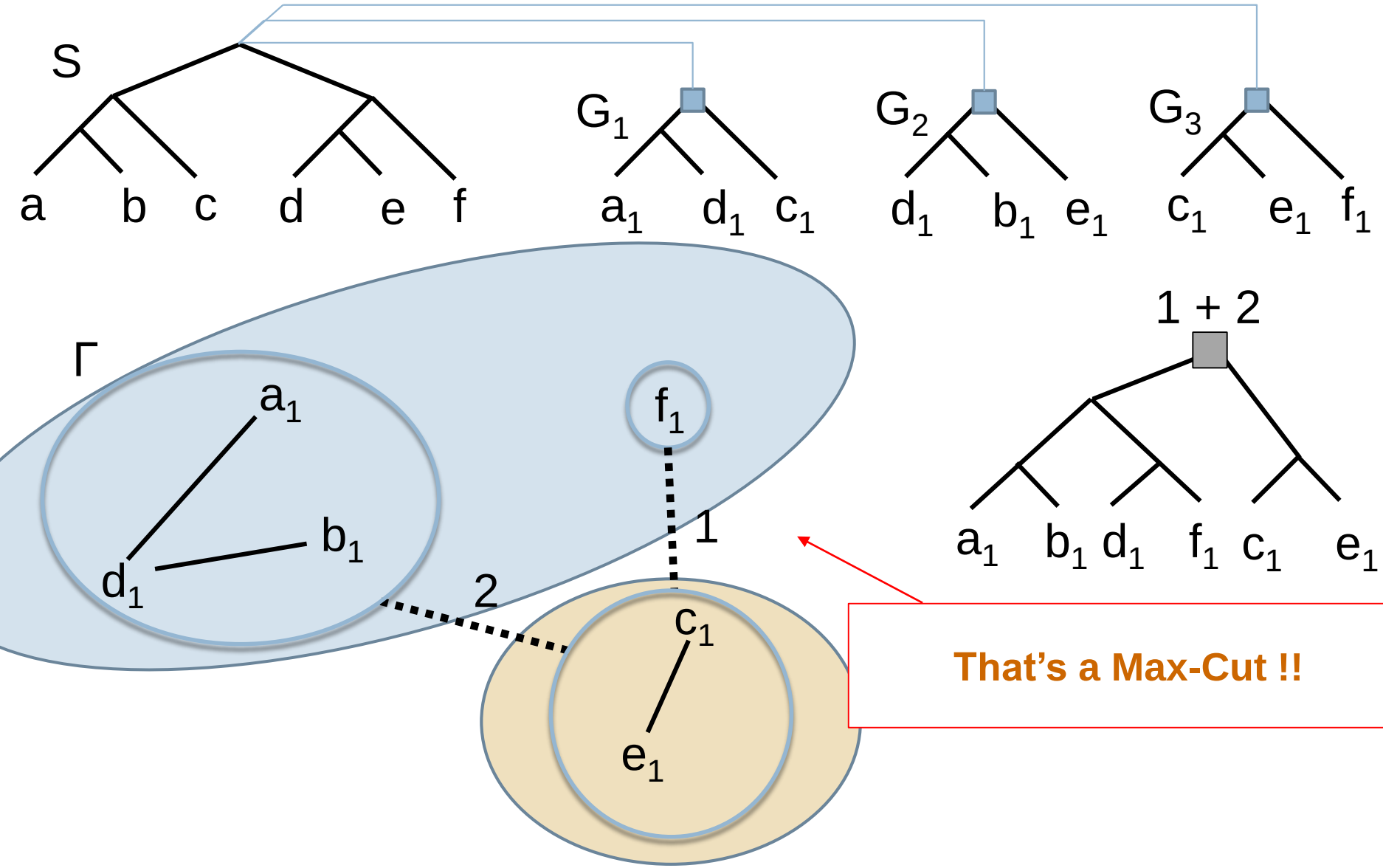
A greedy approach



A greedy approach



A greedy approach



Extending the BUILD algorithm

To minimize the number of PreSpecDups :

- Make the BUILD graph
- Add the PreSpecDup edges
- Find a Max-Cut partition of the components
- Repeat recursively on the parts

Extending the BUILD algorithm

To minimize the number of PreSpecDups :

- Make the BUILD graph
- Add the PreSpecDup edges
- Find a Max-Cut partition of the components
- Repeat recursively on the parts

That's NP-Hard ! And we have to repeat it recursively !!

Extending the BUILD algorithm

To minimize the number of PreSpecDups :

- Make the BUILD graph
- Add the PreSpecDup edges
- Find a Max-Cut partition of the components
- Repeat recursively on the parts

That's NP-Hard ! And we have to repeat it recursively !!

The result : even this problem is hard to approximate !

Conclusion

- Fixed Parameter Tractability ?
- Criteria other than duplications ?
  - e.g. gene losses
- What to do if the input gene trees are incompatible ?

Acknowledgements

Aïda Ouangraoua

Nadia El-Mabrouk

NSERC
CRSNG

Fonds de recherche
Nature et
technologies
Québec

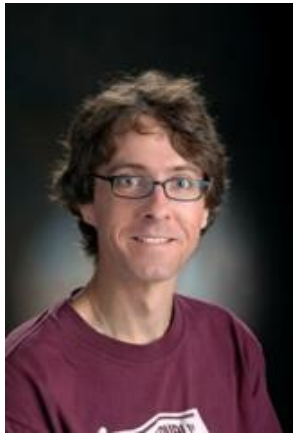
Canada Research Chairs
www.chairs-chaires.gc.ca

DIRO

The 14th RECOMB-CG

October 2016 in MONTRÉAL ☺

Probably from Monday 10 to Wednesday 12



SFU

Simon Fraser University

DIRO

CENTRE DE RECHERCHES MATHÉMATIQUES

Université de Montréal