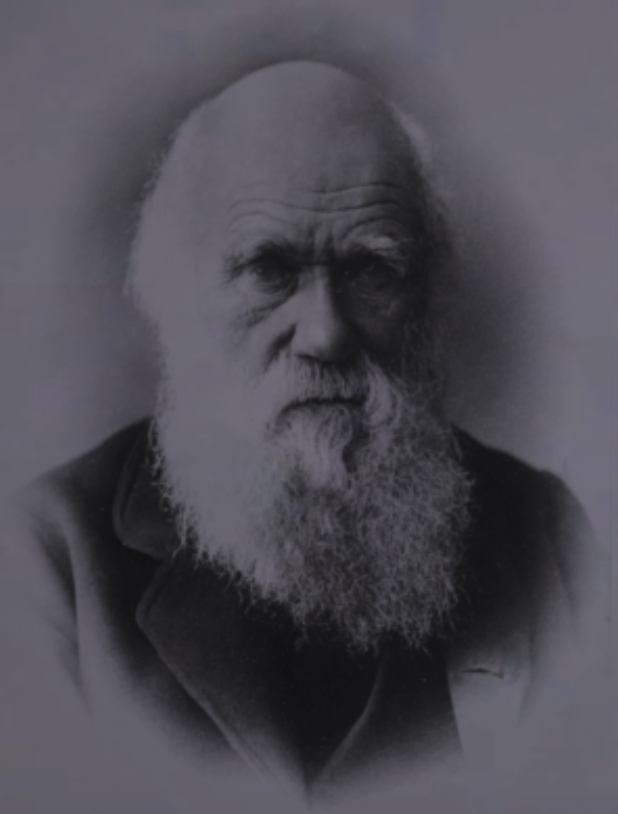
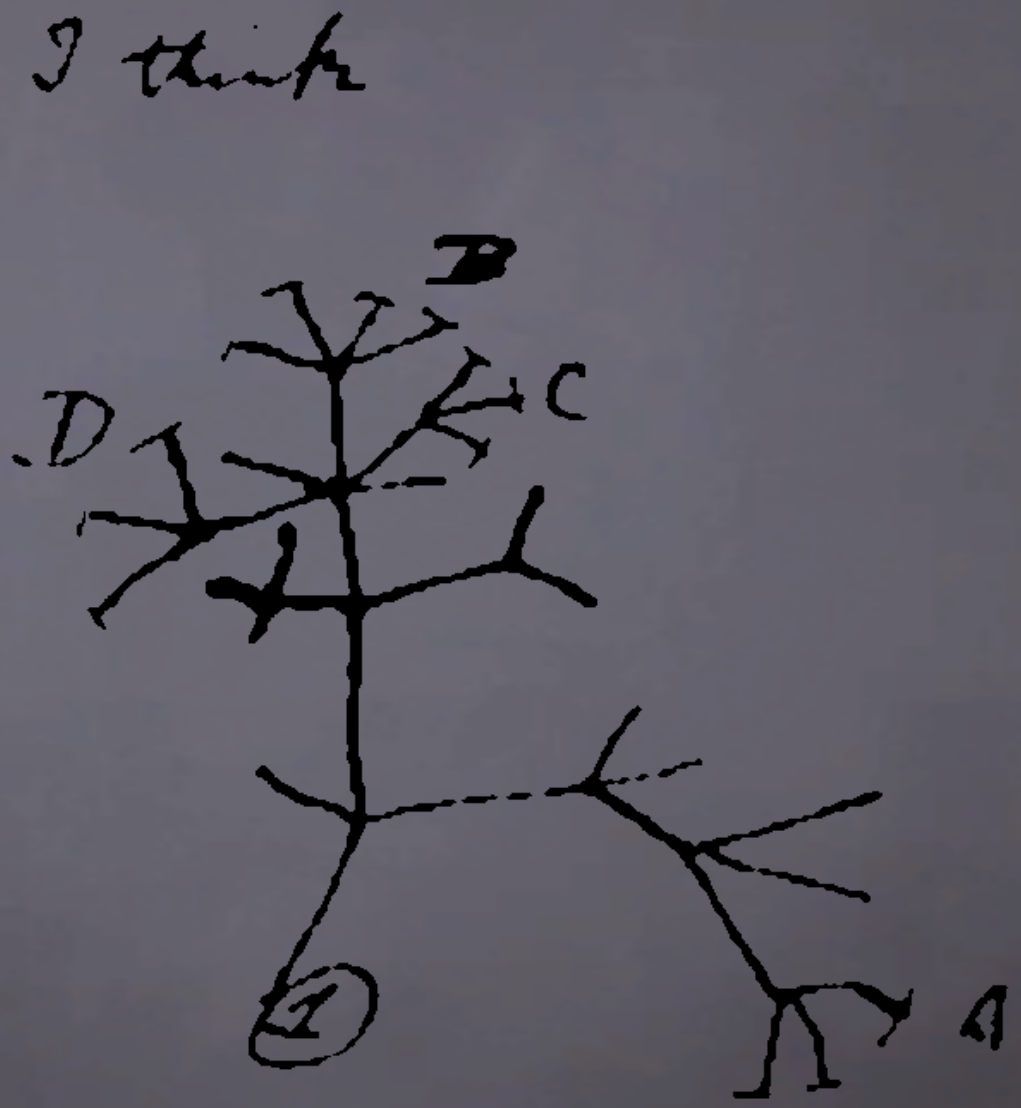
Lecture: Algorithmic Bioinformatics

Doctoral School, Université Dauphine, 2022

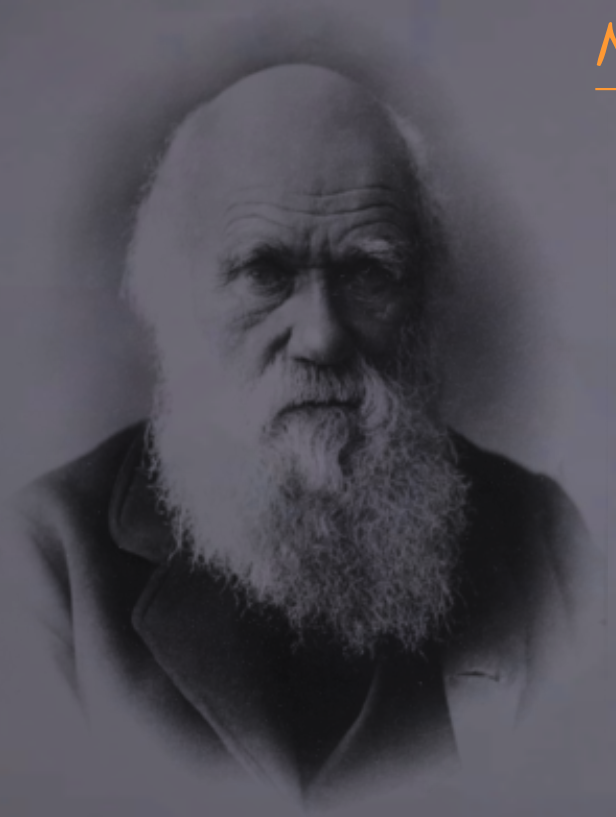
Some slides graciously provided by Daniel Huson & Celine Scornavacca

Phylogenetic Trees - Motivation





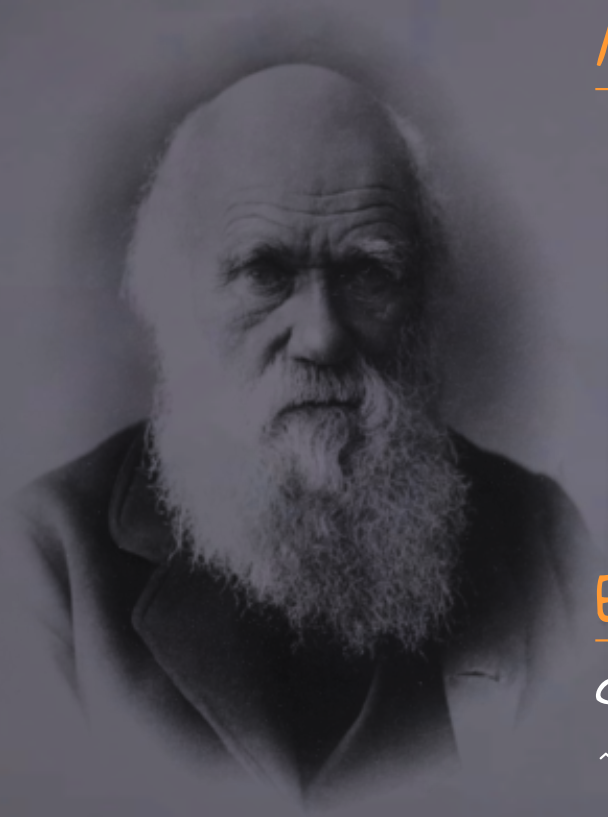
Phylogenetic Trees - Motivation



Motivation

- study relation between species
- evolution of characteristics
- co-evolution (host-parasite)
- geological migration
- genetic development of viruses/diseases

Phylogenetic Trees - Motivation



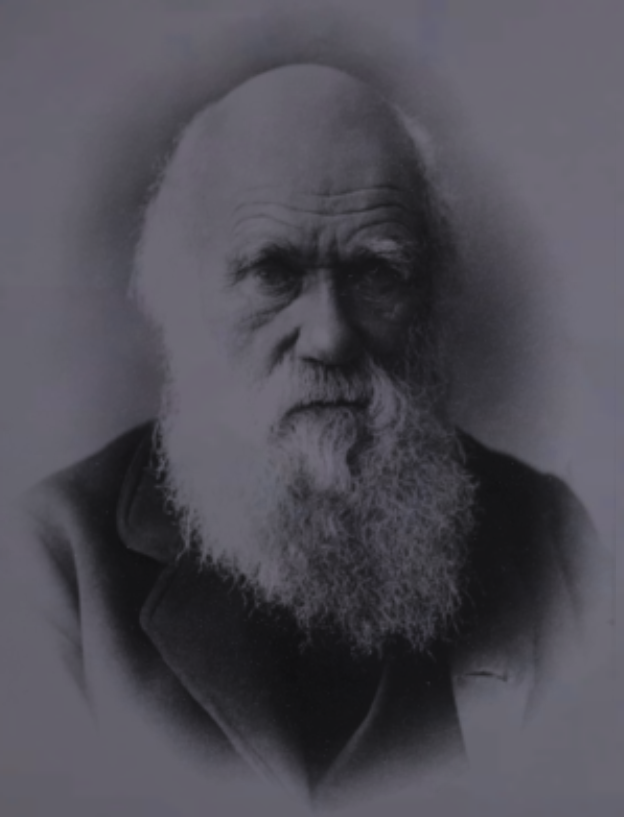
Motivation

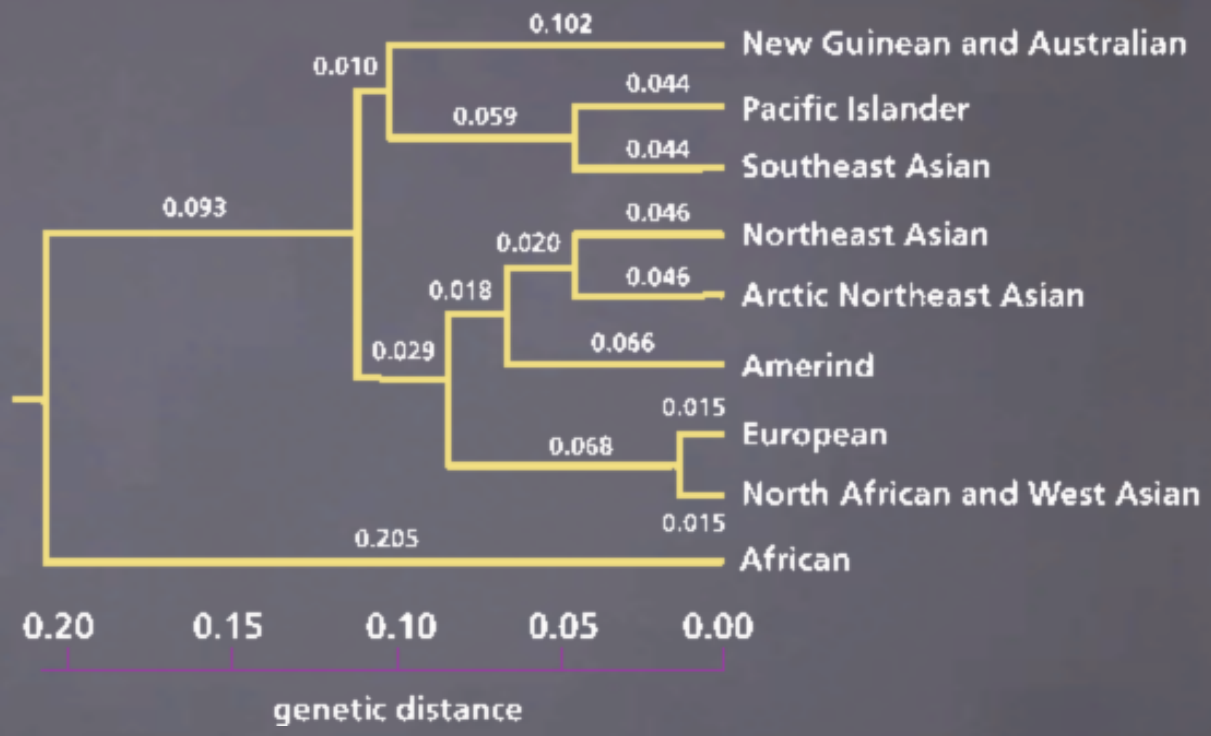
- study relation between species
- evolution of characteristics
- co-evolution (host-parasite)
- geological migration
- genetic development of viruses/diseases

Evolution

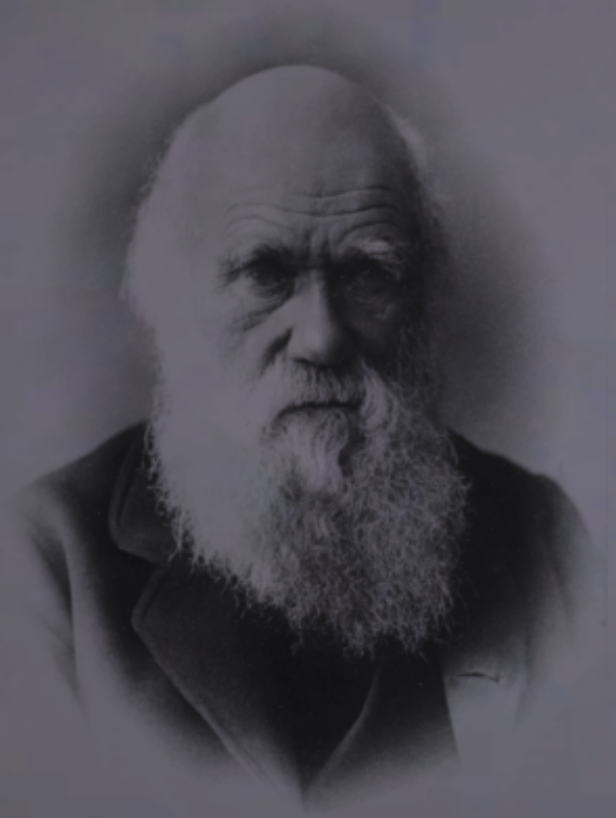
genetic material changes over time
$\rightsquigarrow$ new species "branch off"
$\rightsquigarrow$ "tree of life"

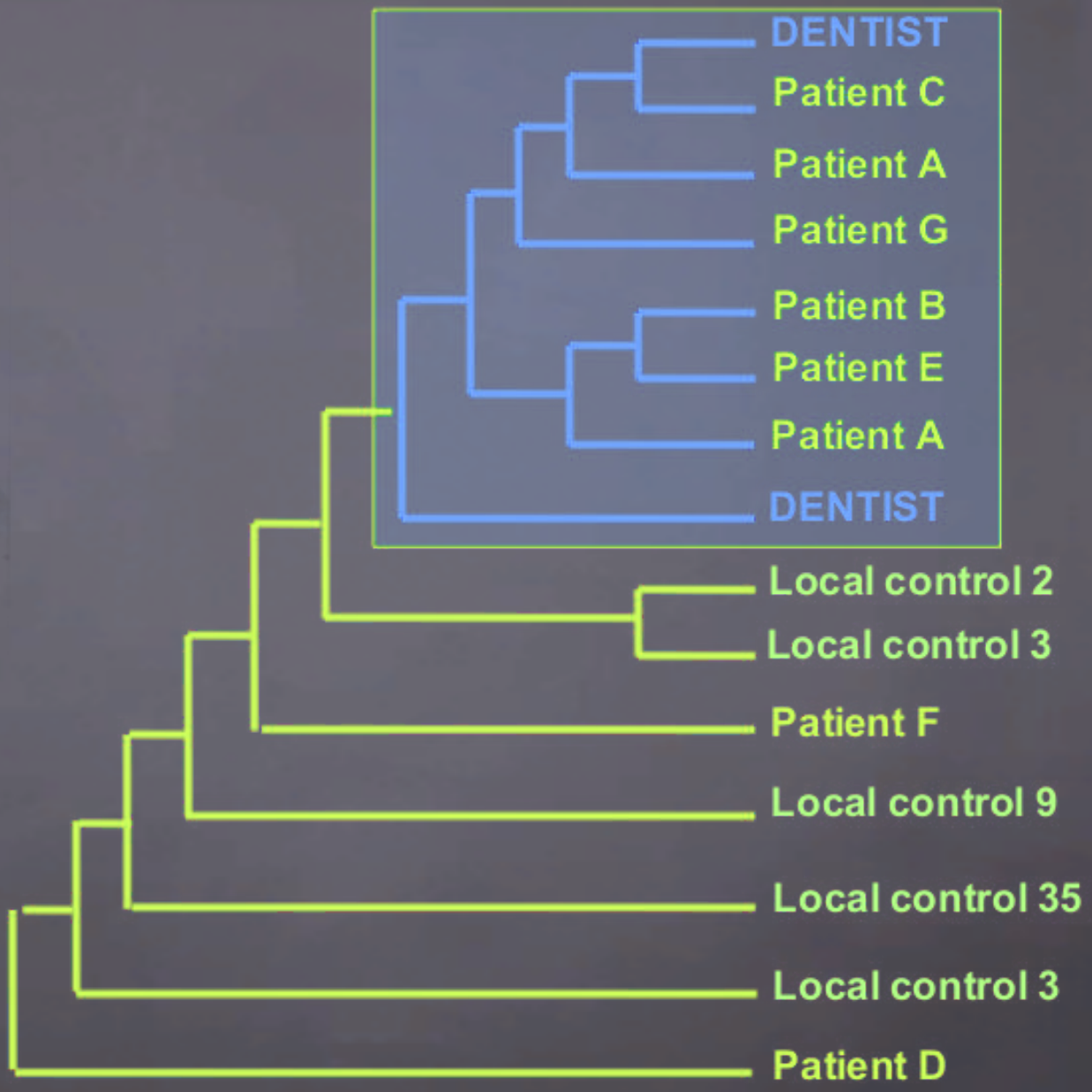
Phylogenetic Trees - Motivation





Phylogenetic Trees - Motivation

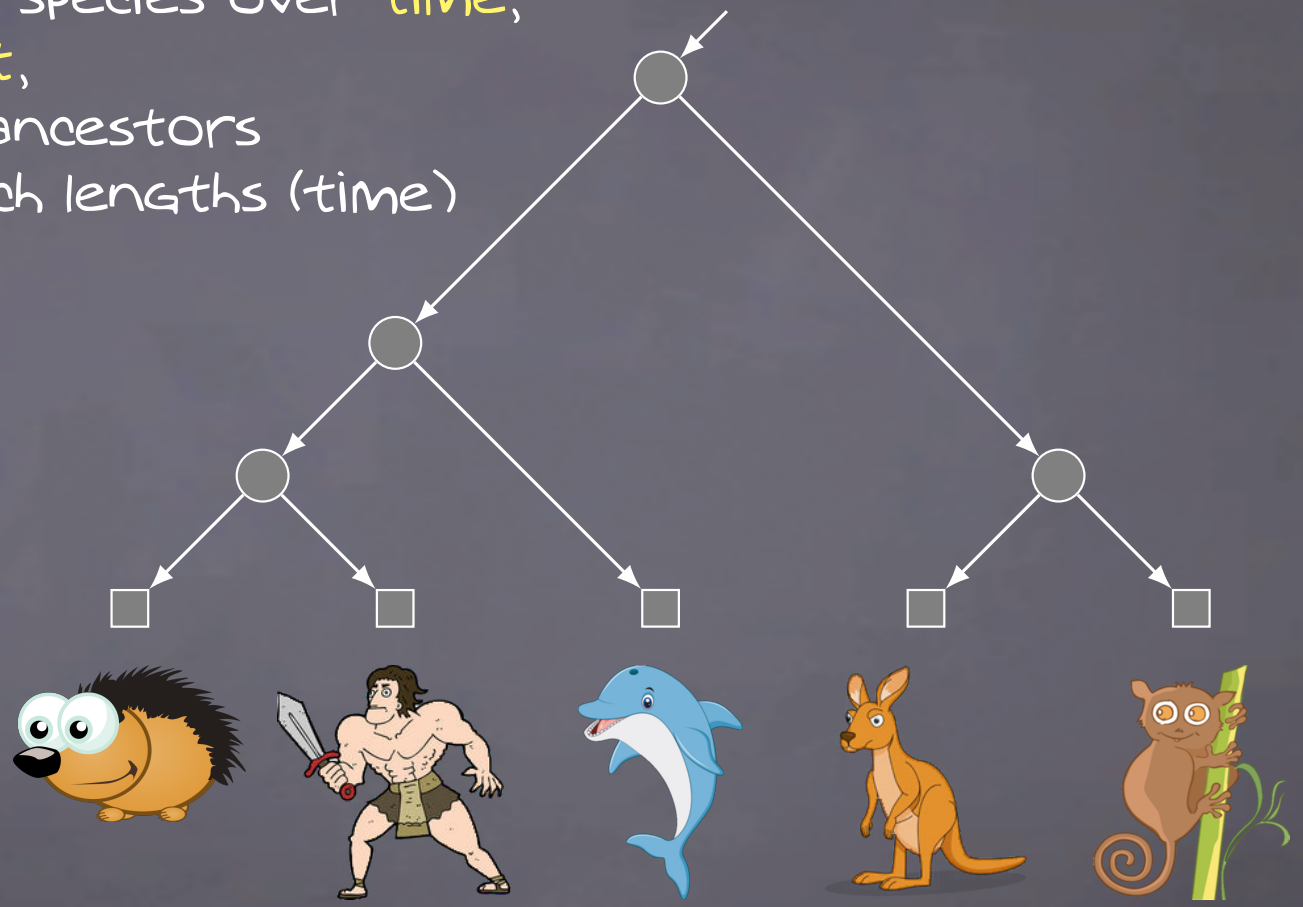




Ou et al. (1992), Page et Holmes (1998)

Rooted Phylogenetic Trees

evolution of species over time,
leaves extant,
hypothetical ancestors
possibly branch lengths (time)



Notation

taxon, cluster, triplet

Rooted Phylogenetic Trees

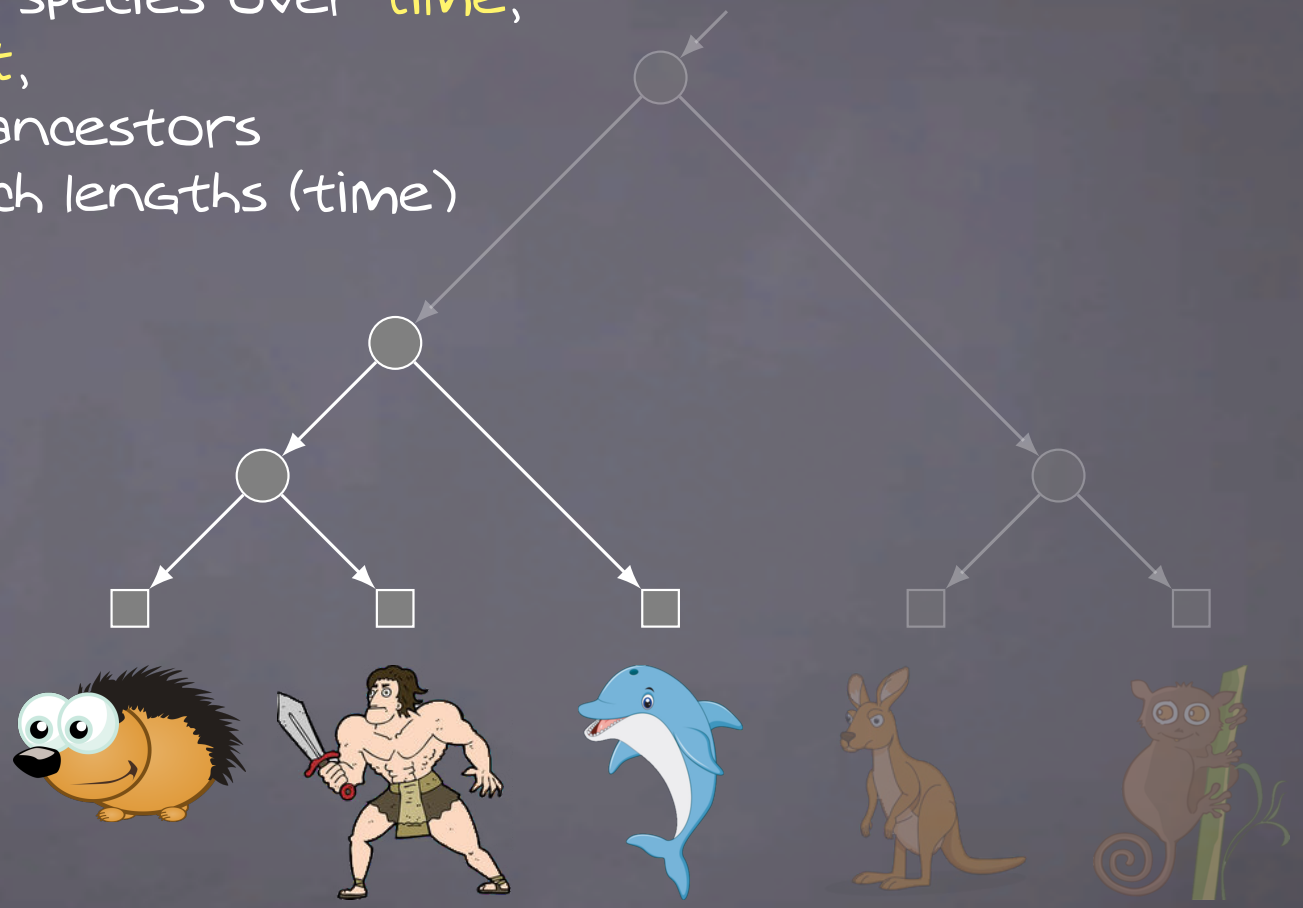
evolution of species over time,
leaves extant,
hypothetical ancestors
possibly branch lengths (time)

Notation

taxon, cluster, triplet

Rooted Phylogenetic Trees

evolution of species over time,
leaves extant,
hypothetical ancestors
possibly branch lengths (time)



Notation

taxon, cluster, triplet

Rooted Phylogenetic Trees

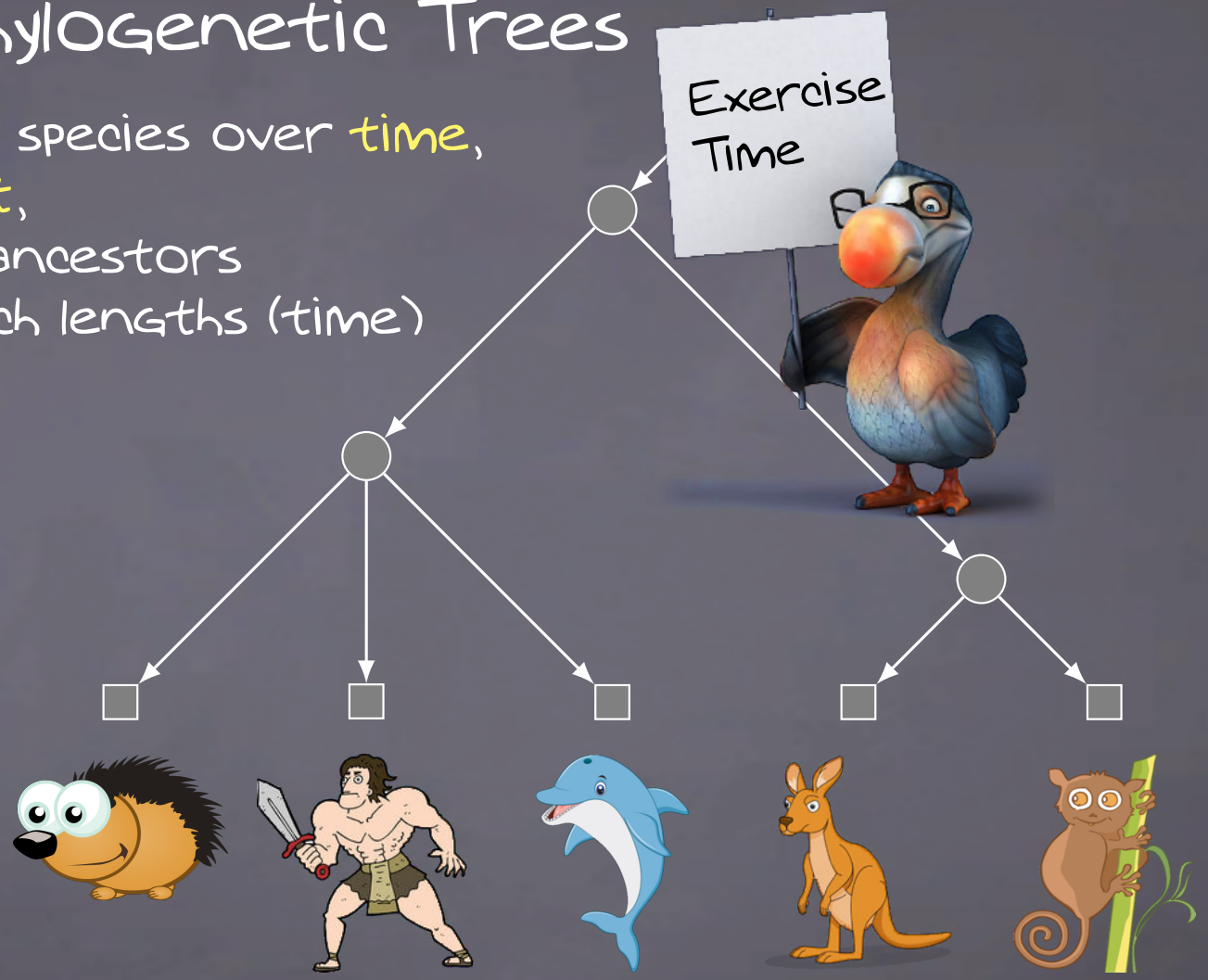
evolution of species over time,
leaves extant,
hypothetical ancestors
possibly branch lengths (time)

Notation

taxon, cluster, triplet

Rooted Phylogenetic Trees

evolution of species over time,
leaves extant,
hypothetical ancestors
possibly branch lengths (time)



Notation

taxon, cluster, triplet

"Polytomies"

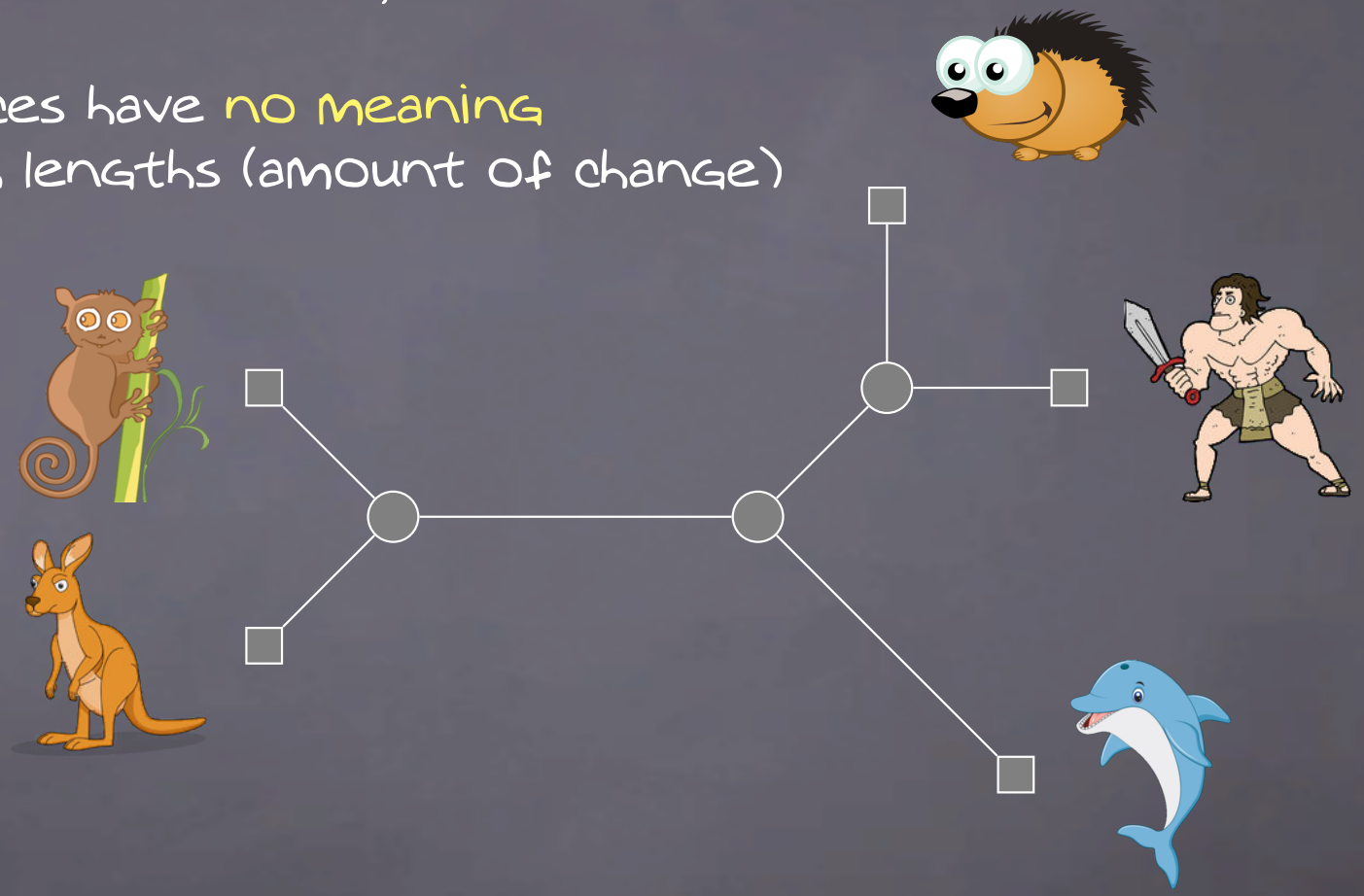
history not clear ⇝ "soft"
known "fan out" ⇝ "hard"

Exercise:

use $xy|z \leftrightarrow LCA(xy)<LCA(xz)=LCA(xyz)$ to prove $ab|c + bc|d \rightarrow ac|d$

Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)

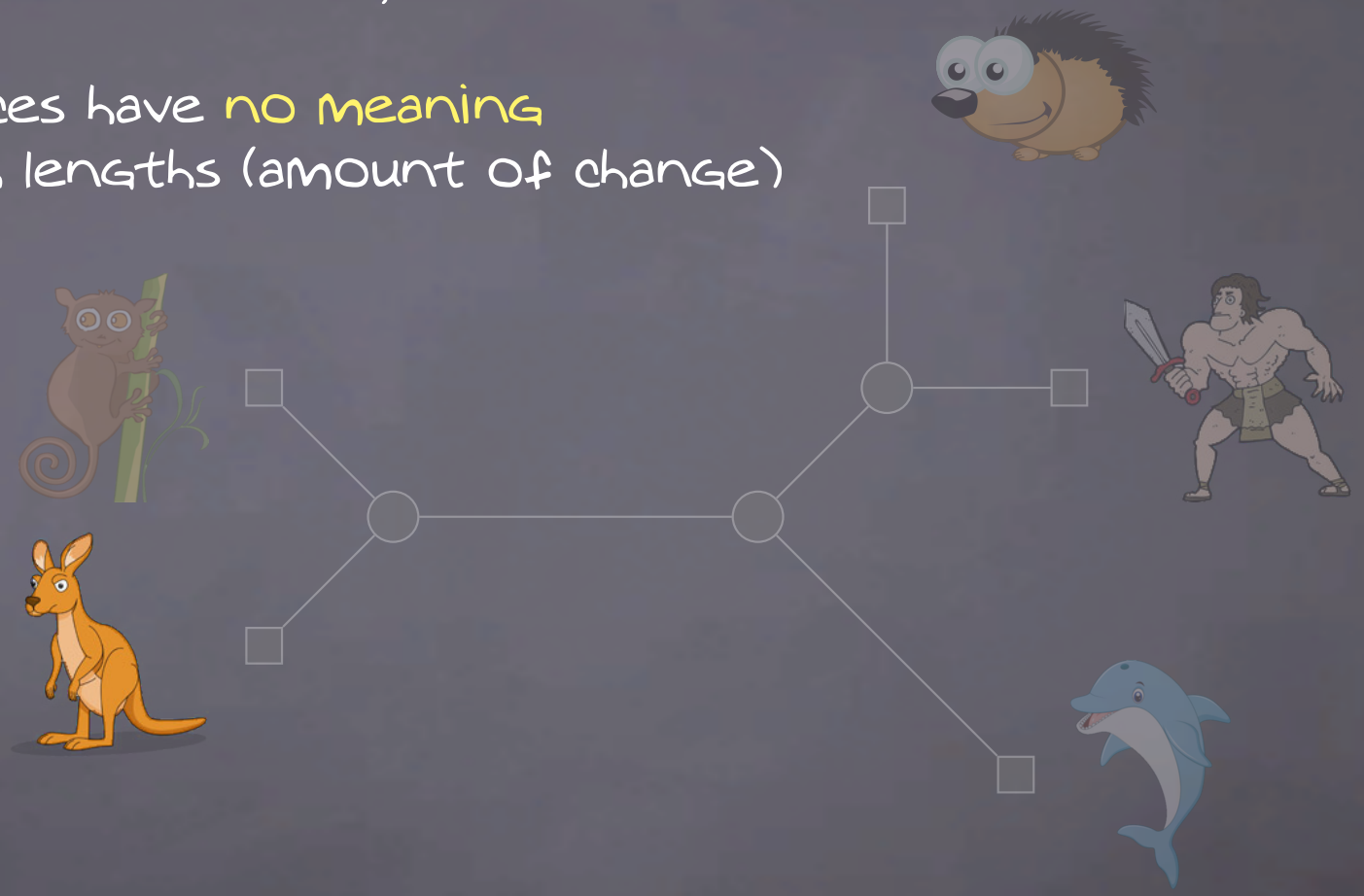


Notation

taxon, split, quartet

Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)

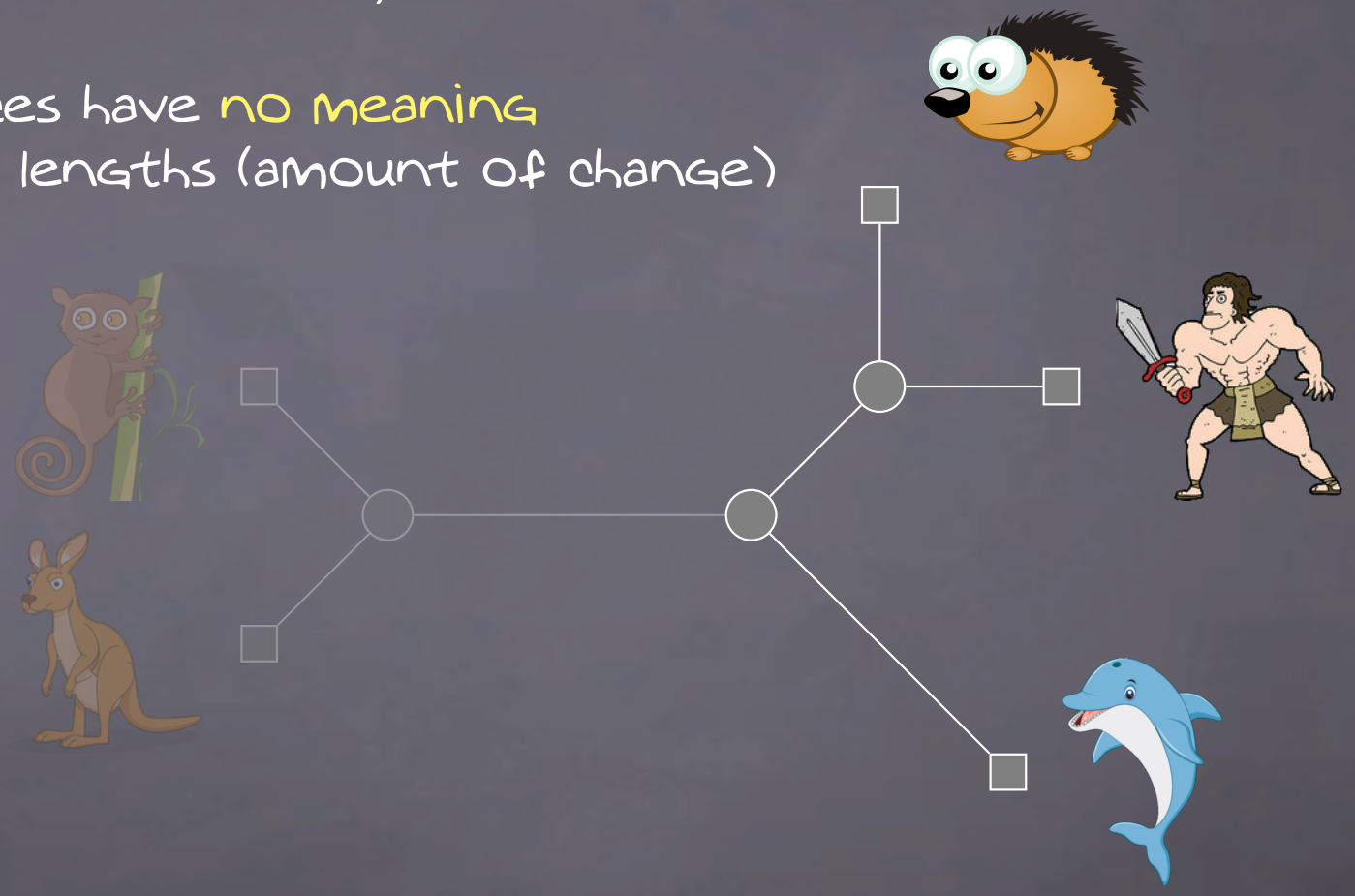


Notation

taxon, split, quartet

Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)

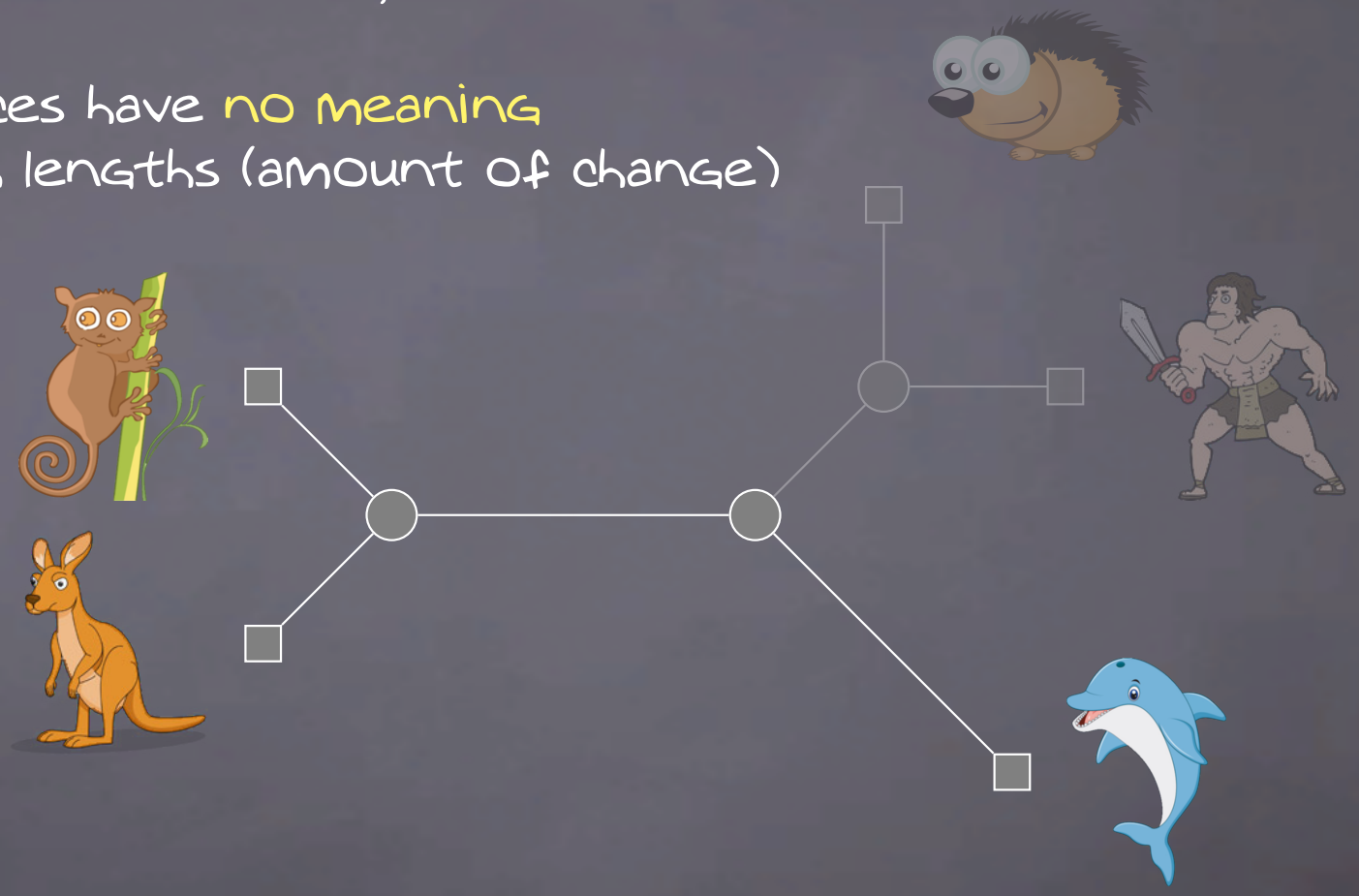


Notation

taxon, split, quartet

Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)

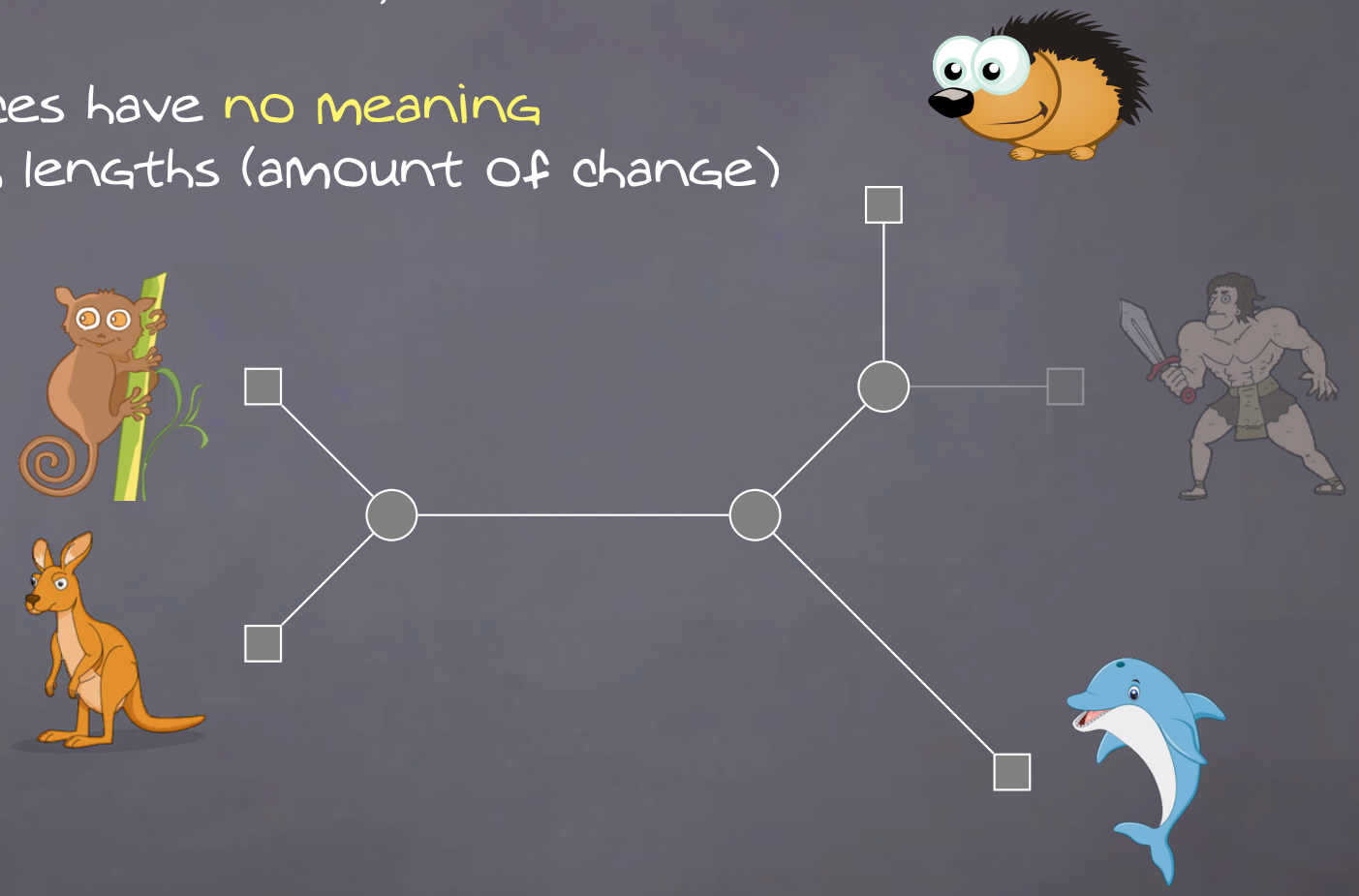


Notation

taxon, split, quartet

Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)



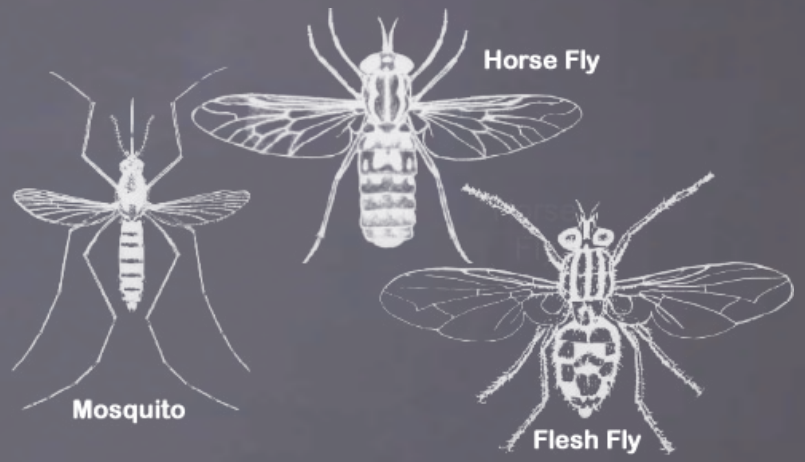
Notation

taxon, split, quartet

Reconstructing Phylogenetic Trees

Group Species By...

- morphology
- behavior
- geography

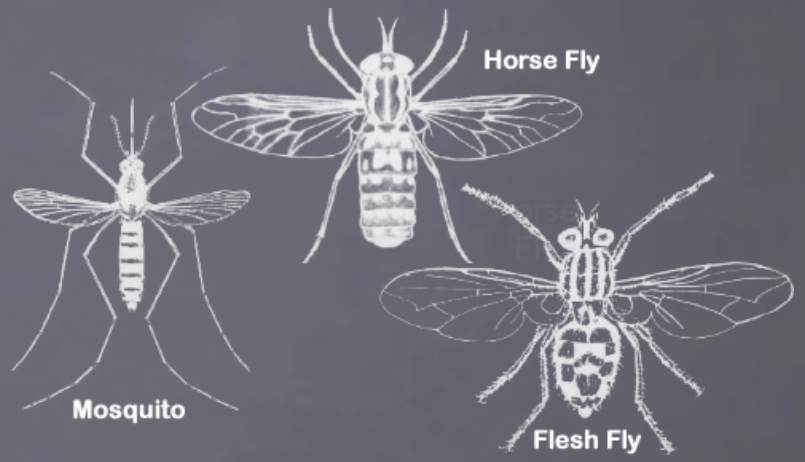


Diptera = two wings

Reconstructing Phylogenetic Trees

Group Species By...

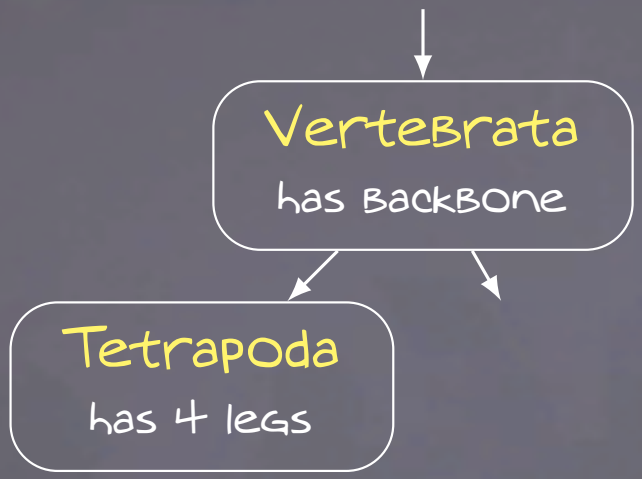
- morphology
- behavior
- geography
- distance of sequences
- "genetic distance"
- etc.



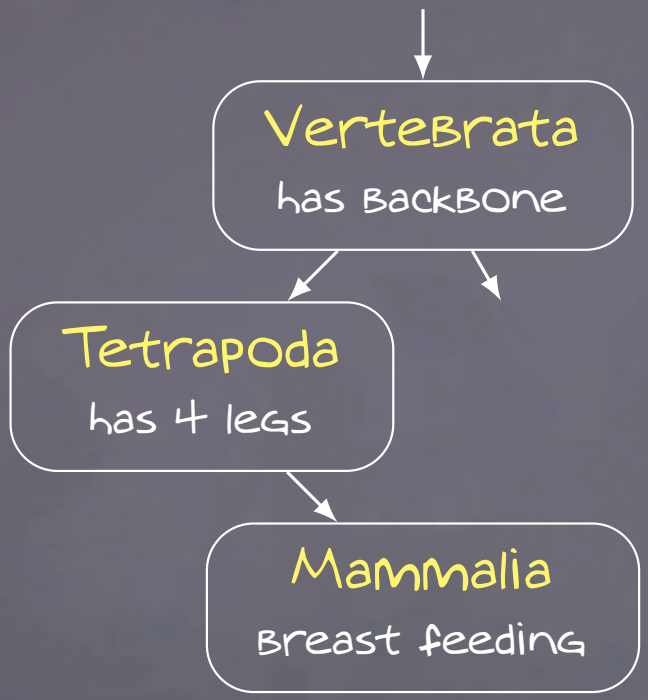
Diptera = two wings

Reconstructing Phylogenetic Trees

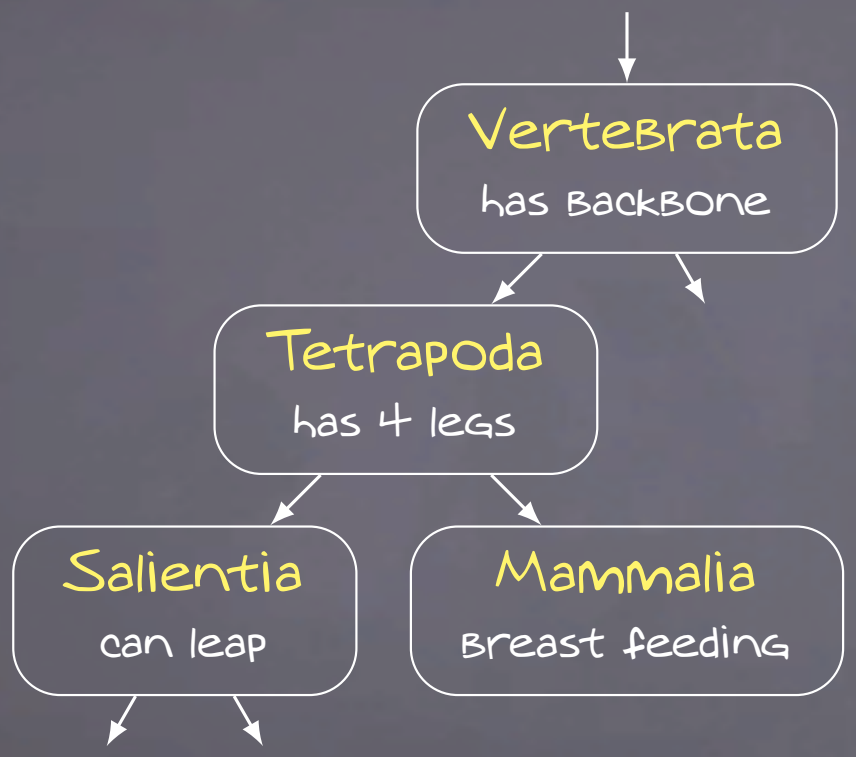
Reconstructing Phylogenetic Trees



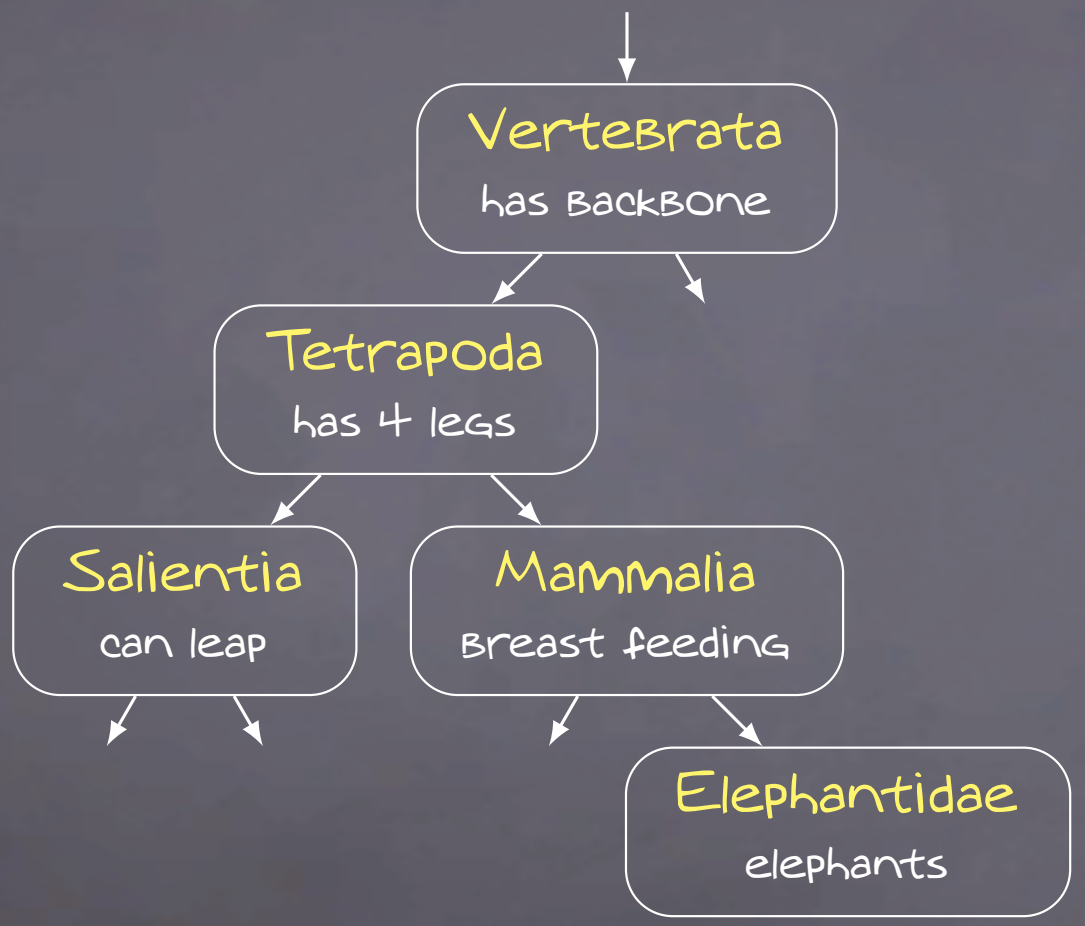
Reconstructing Phylogenetic Trees



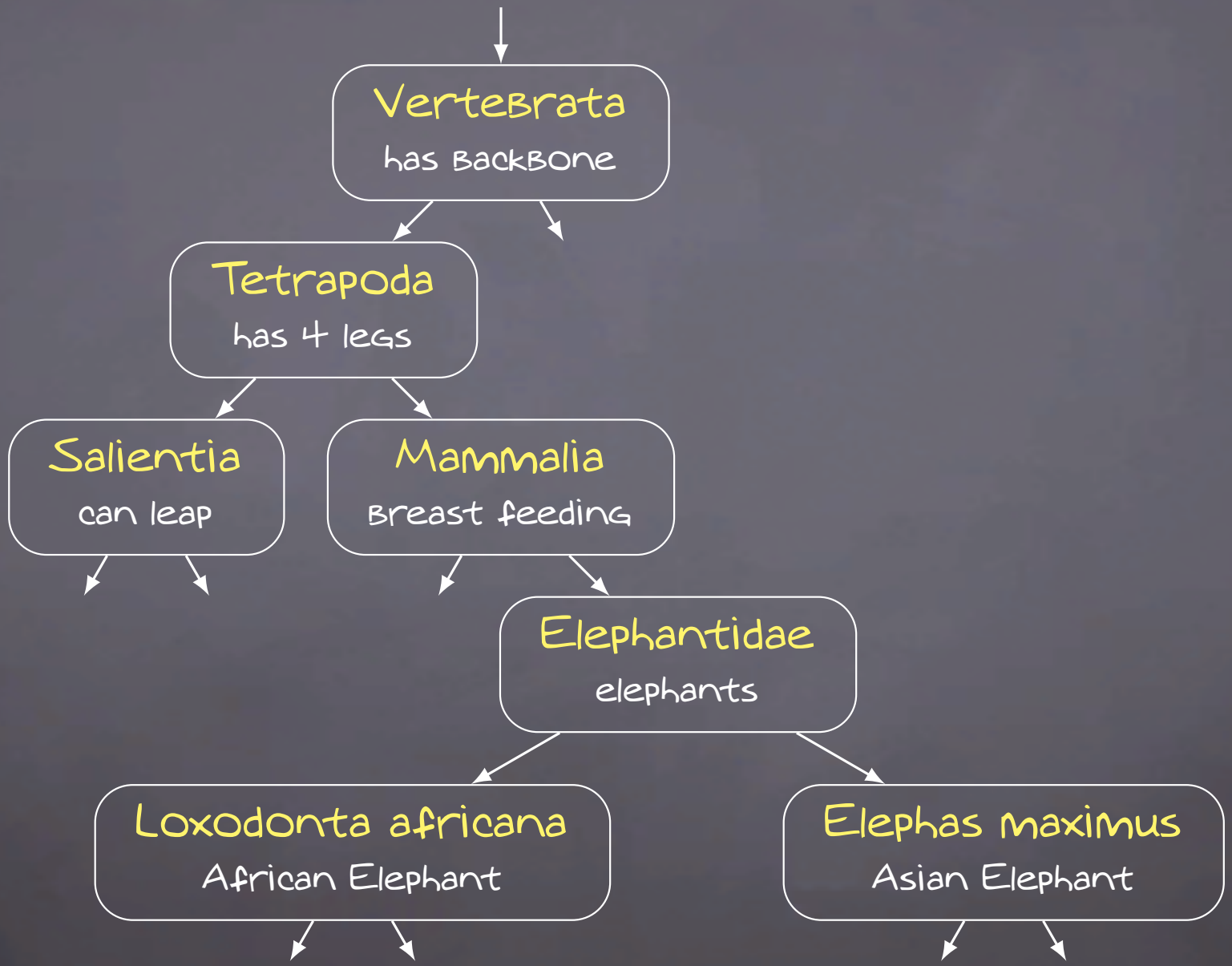
Reconstructing Phylogenetic Trees



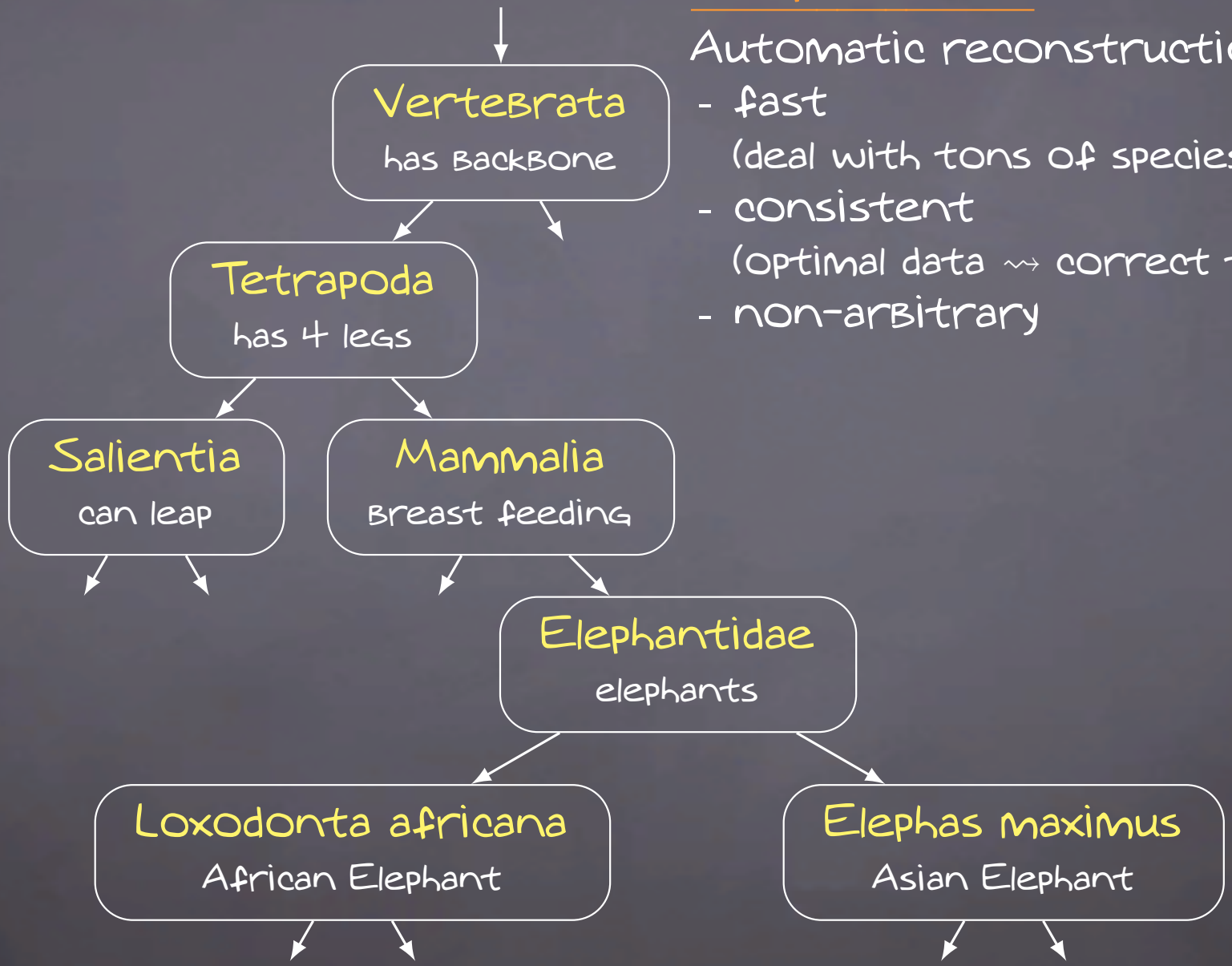
Reconstructing Phylogenetic Trees



Reconstructing Phylogenetic Trees



Reconstructing Phylogenetic Trees



Keep in Mind

Automatic reconstruction should be

- fast
  (deal with tons of species & genes)
- consistent
  (optimal data $\rightsquigarrow$ correct tree)
- non-arbitrary

Distance-Based Reconstruction

Idea: cluster hierarchically

Distance-Based Reconstruction

Idea: cluster hierarchically

Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

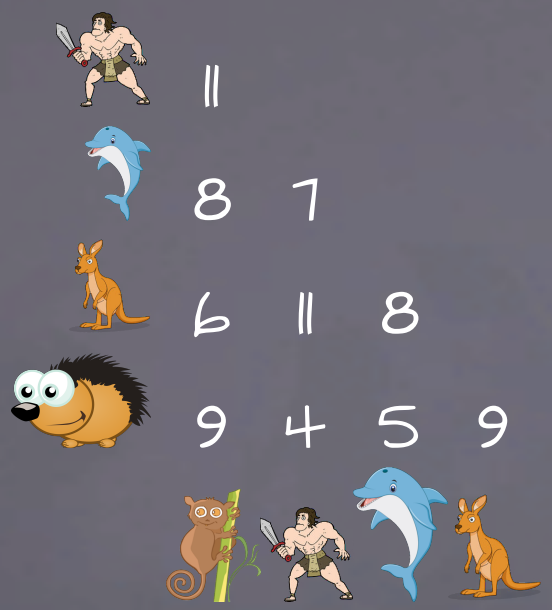
Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

Distance-Based Reconstruction

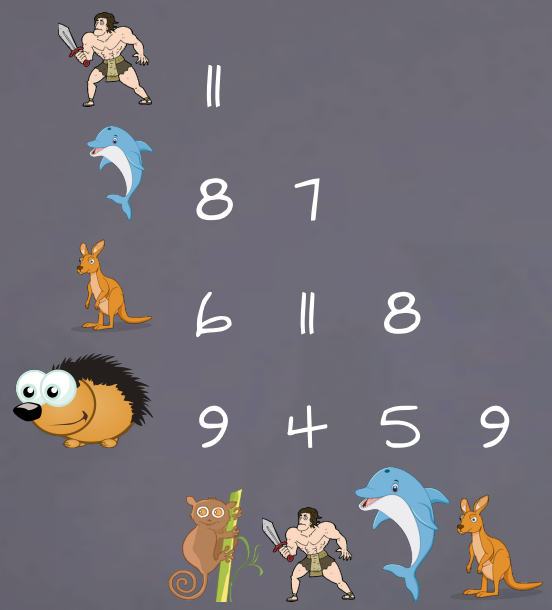
Idea: cluster hierarchically



Idea: merge closest clusters

Distance-Based Reconstruction

Idea: cluster hierarchically



Idea: merge closest clusters

Branch lengths ??

assume molecular clock

⇝ ultrametric

Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric

Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric

Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

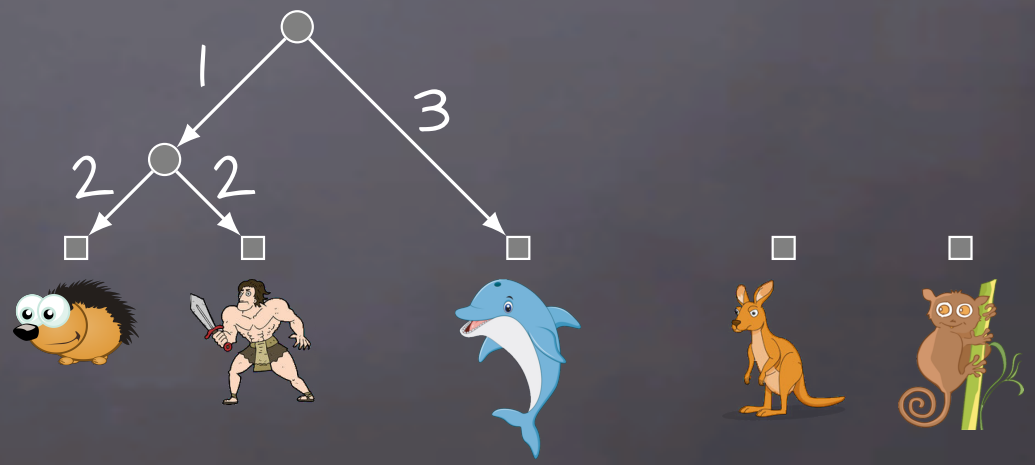
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

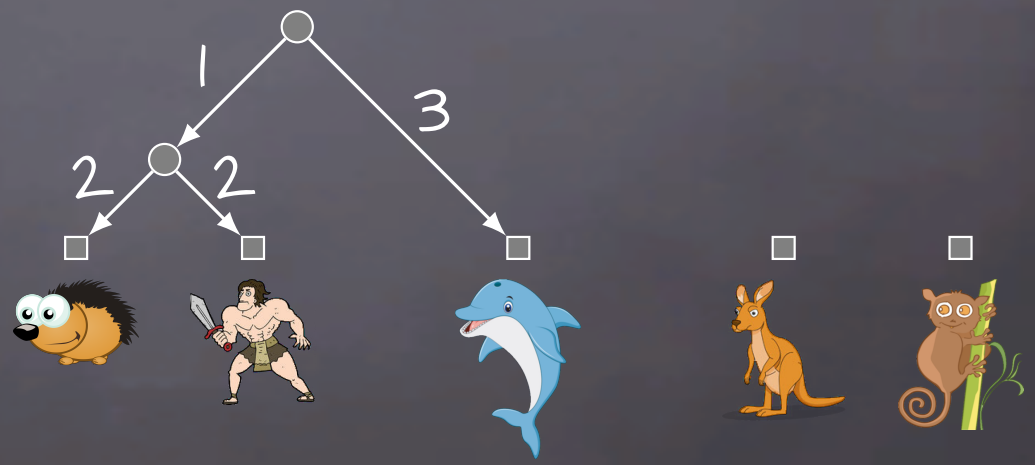
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

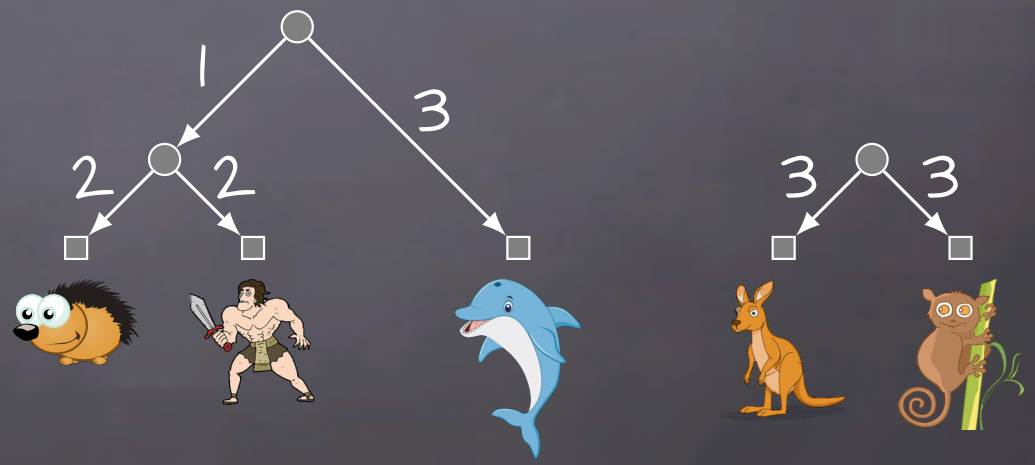
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

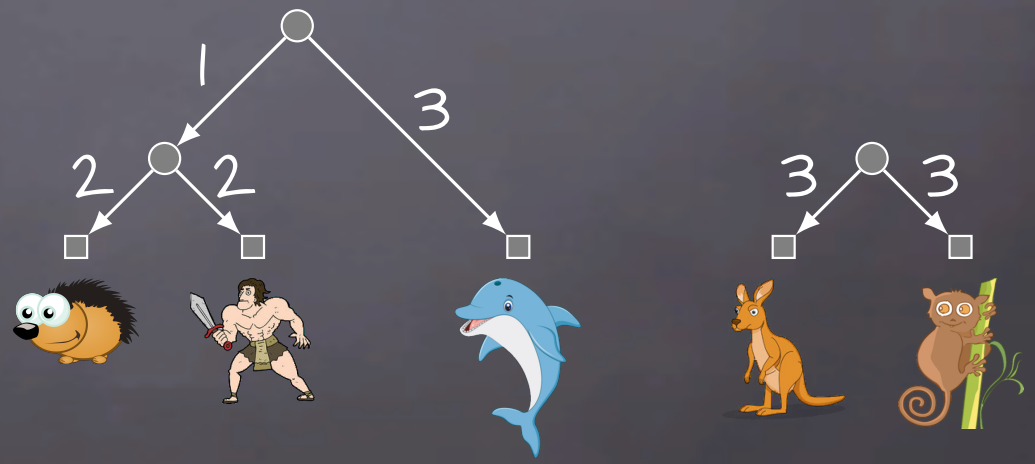
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically

28/3

Idea: merge closest clusters

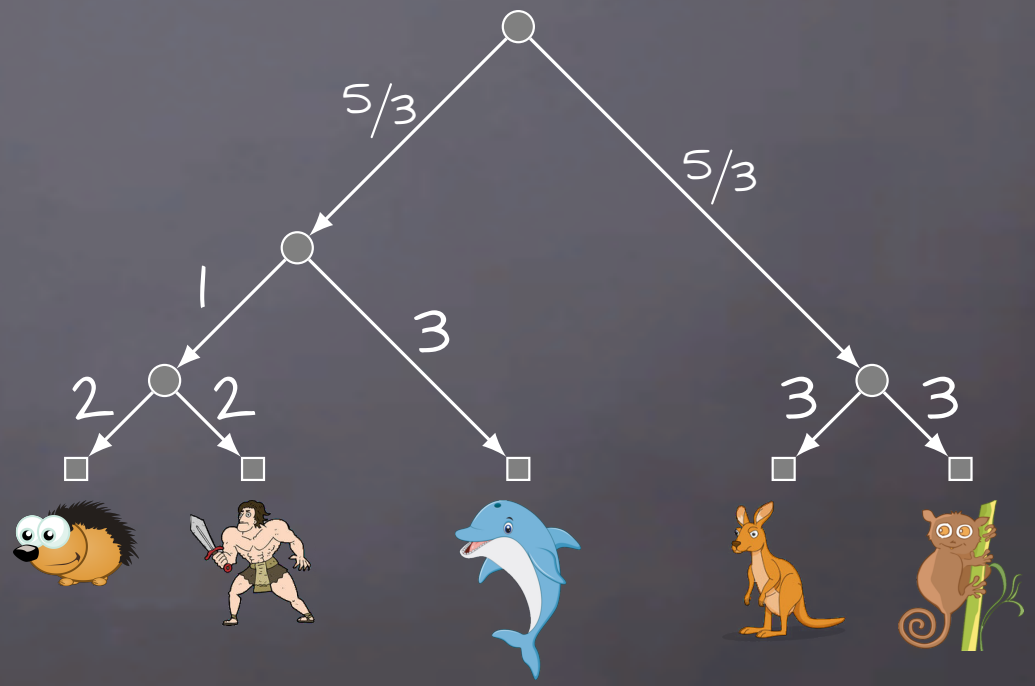
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically

Unweighted Pair Group Method w/ Avg.

- find "closest pair"
- "join" them
- update distances & recurse

28/3

Idea: merge closest clusters

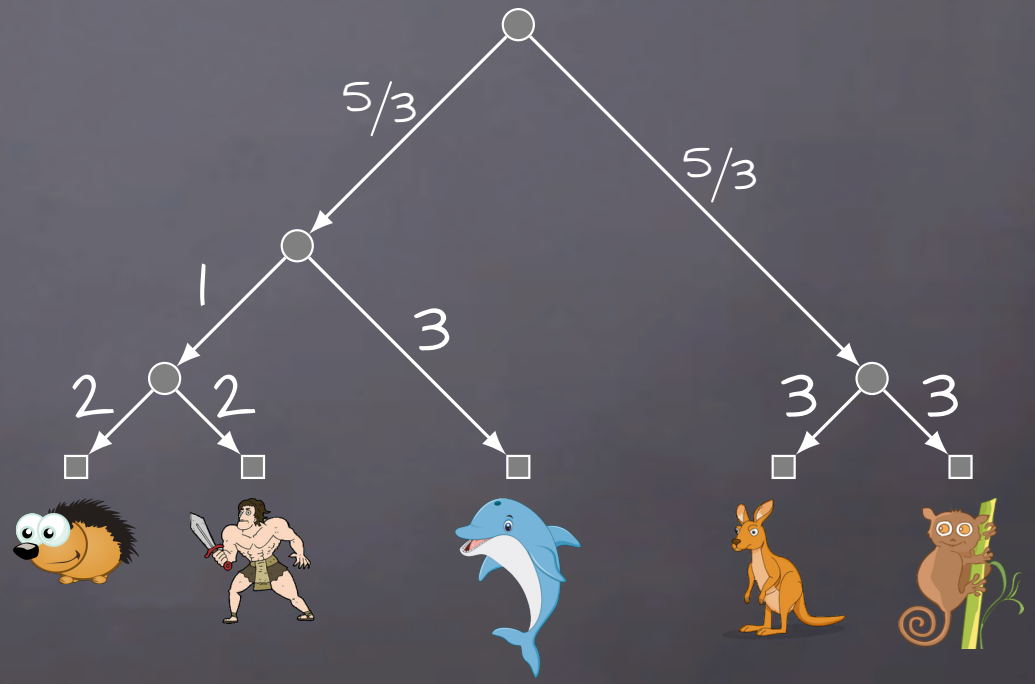
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

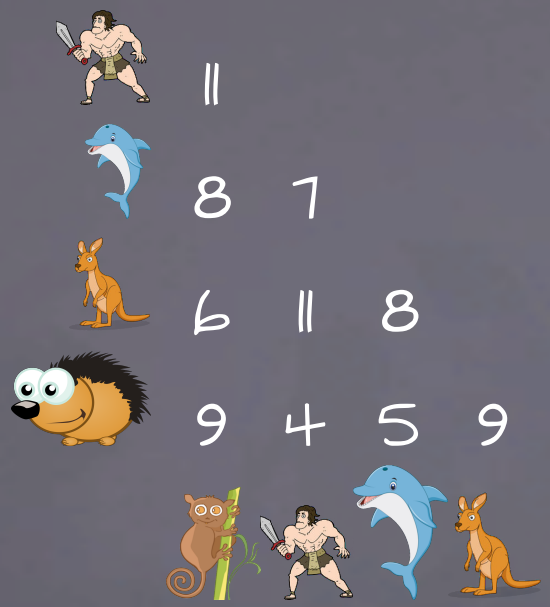
assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically



Unweighted Pair Group Method w/ Avg.

- find "closest pair"
- "join" them
- update distances & recurse

Idea: merge closest clusters

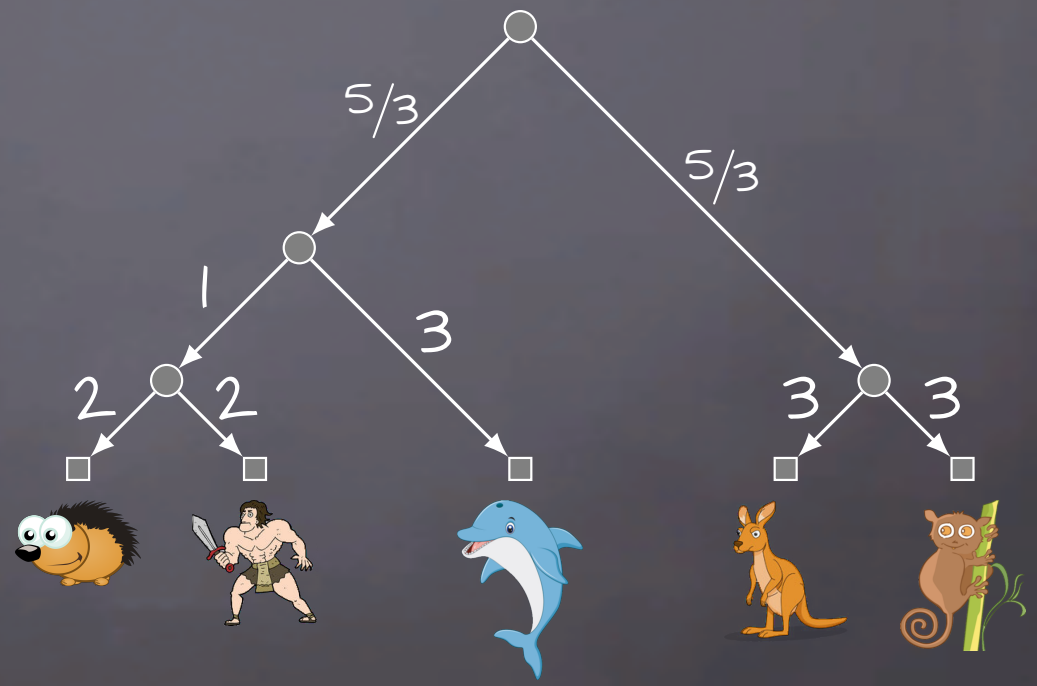
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

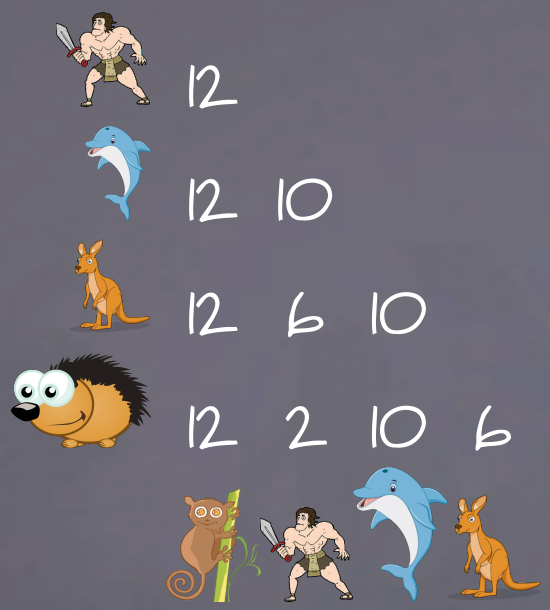
assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically



Unweighted Pair Group Method w/ Avg.

- find "closest pair"
- "join" them
- update distances & recurse

Idea: merge closest clusters

update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

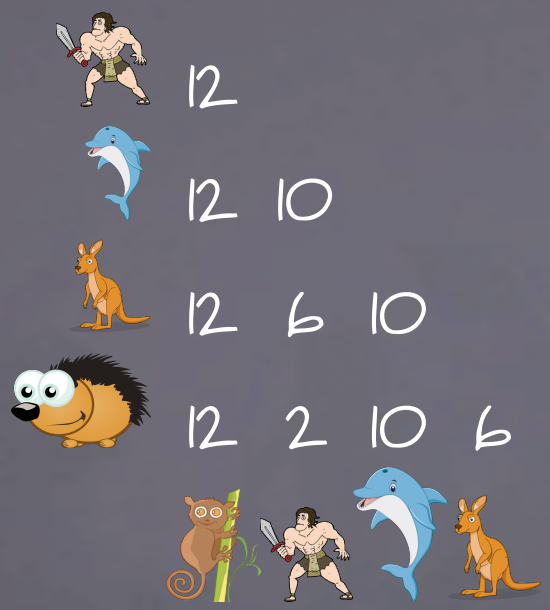
Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric

Distance-Based Reconstruction

Idea: cluster hierarchically



Unweighted Pair Group Method w/ Avg.

- find "closest pair"
- "join" them
- update distances & recurse

Idea: merge closest clusters

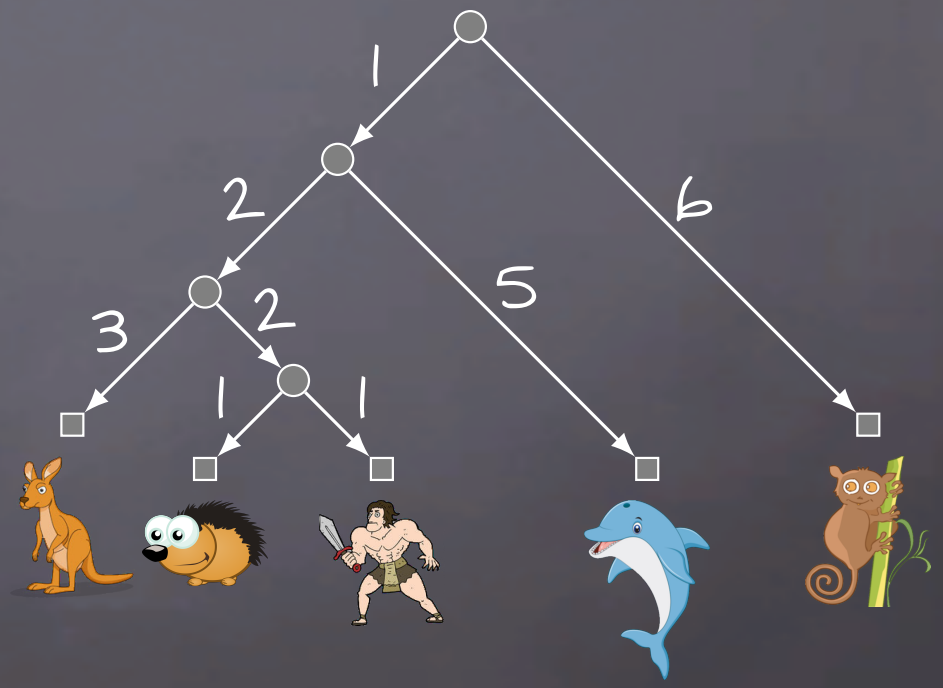
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

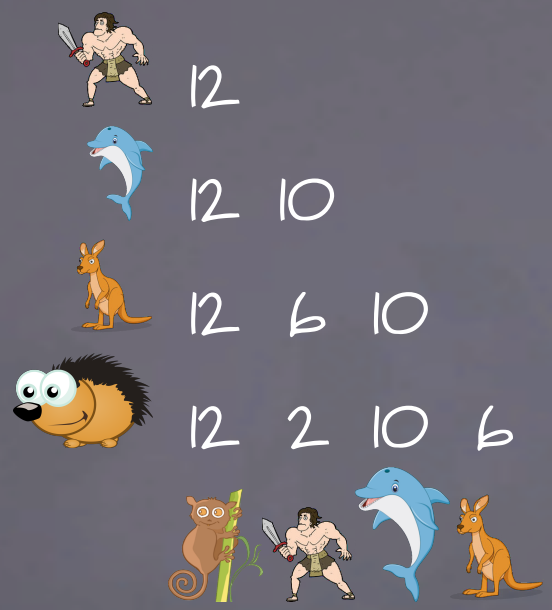
assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

Idea: cluster hierarchically



Unweighted Pair Group Method w/ Avg.

- find "closest pair"
- "join" them
- update distances & recurse
- only accurate if ultrametric

Idea: merge closest clusters

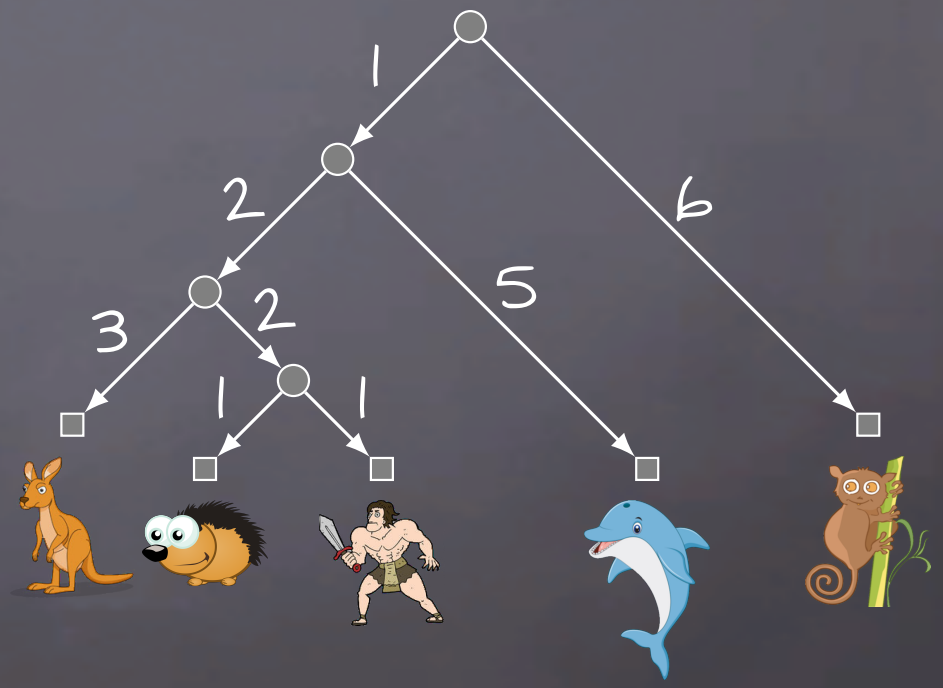
update matrix

$$d_{X \cup Y, Z} = \frac{|X| d_{X,Z} + |Y| d_{Y,Z}}{|X| + |Y|}$$

Branch lengths ??

assume molecular clock

$\rightsquigarrow$ ultrametric



Distance-Based Reconstruction

What about unrooted trees? No root $\rightsquigarrow$ no molecular clock...

Distance-Based Reconstruction

B	9		
C	11	8	
D	9	12	10
	A	B	C

What about unrooted trees? No root $\rightsquigarrow$ no molecular clock...

Distance-Based Reconstruction

Problem: correct pairs may not be closest

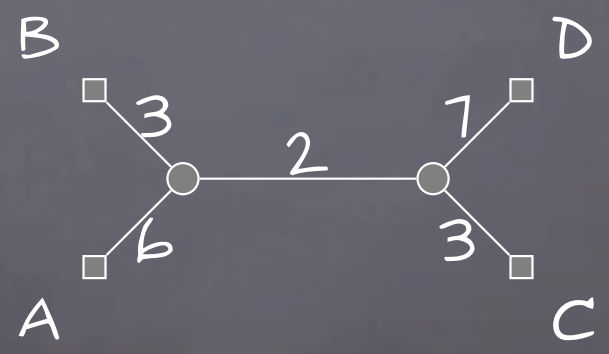
B	9		
C	11	8	
D	9	12	10
	A	B	C

What about unrooted trees? No root $\rightsquigarrow$ no molecular clock...

Distance-Based Reconstruction

Problem: correct pairs may not be closest

B	9		
C	11	8	
D	9	12	10
	A	B	C



Distance-Based Reconstruction

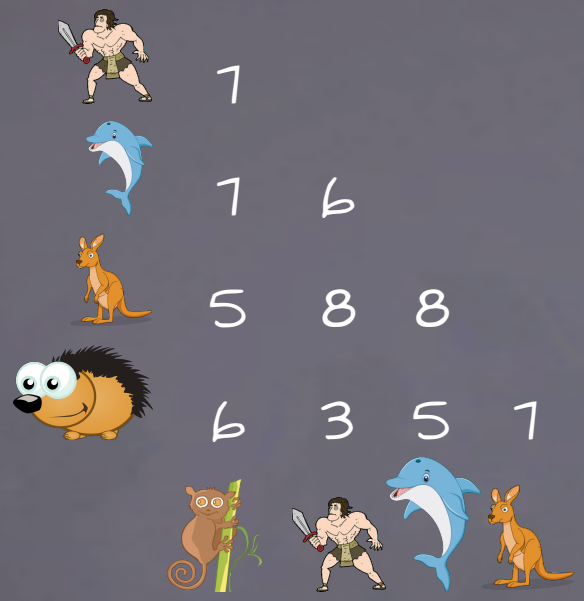
Problem: correct pairs may not be closest

Neighbor Joining (unrooted)

- build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse

Distance-Based Reconstruction

Problem: correct pairs may not be closest

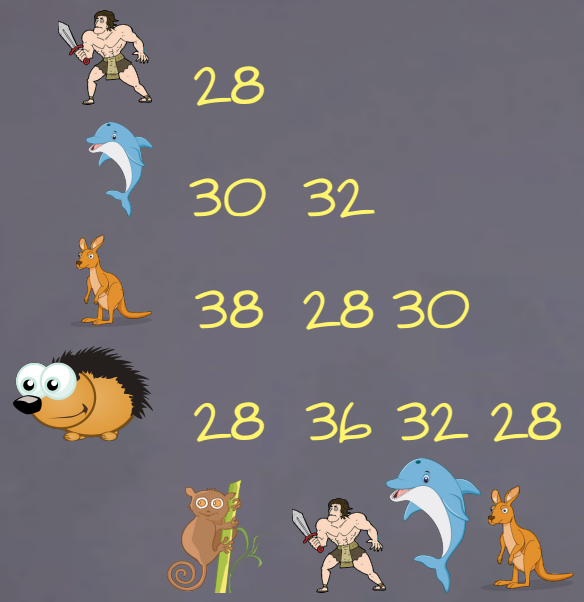


Neighbor Joining (unrooted)

- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse

Distance-Based Reconstruction

Problem: correct pairs may not be closest

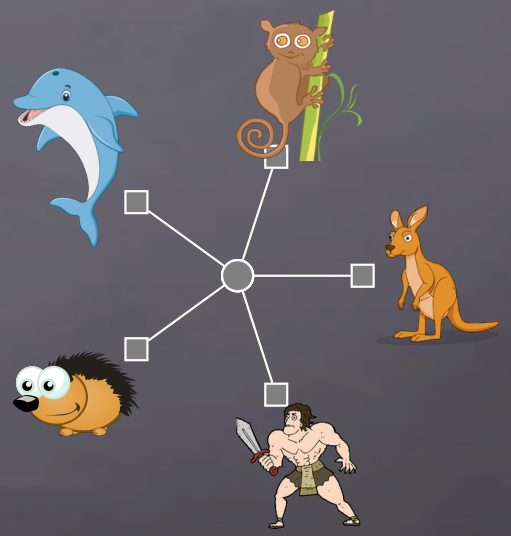


Neighbor Joining (unrooted)

- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse

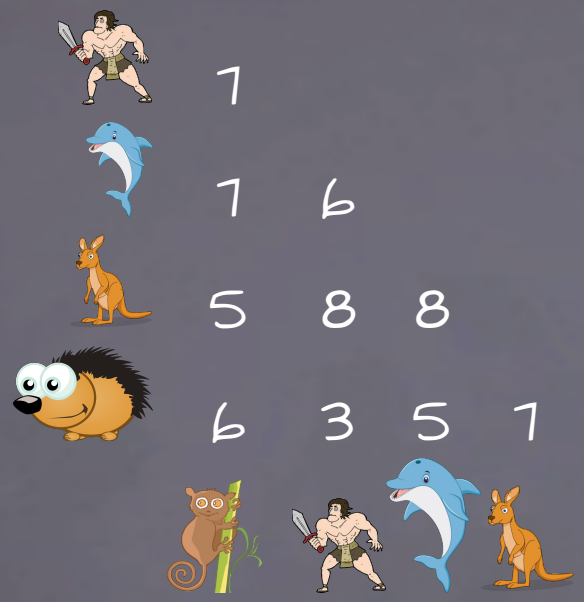
Theorem

$Q_{X,Y}$ max $\Leftrightarrow$ any tree T yielding Q has "cherry" (X, Y)



Distance-Based Reconstruction

Problem: correct pairs may not be closest



Neighbor Joining (unrooted)

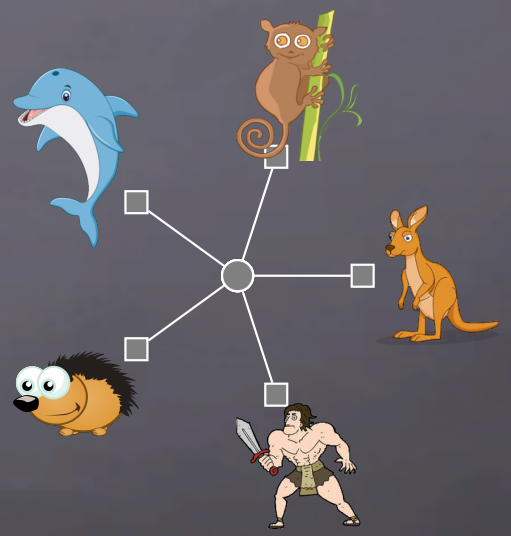
- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse

update distances

$$d_{X \cup Y,Z} = {}^1\!/_2 \left(d_{X,Z} + d_{Y,Z} - d_{X,Y}\right)$$

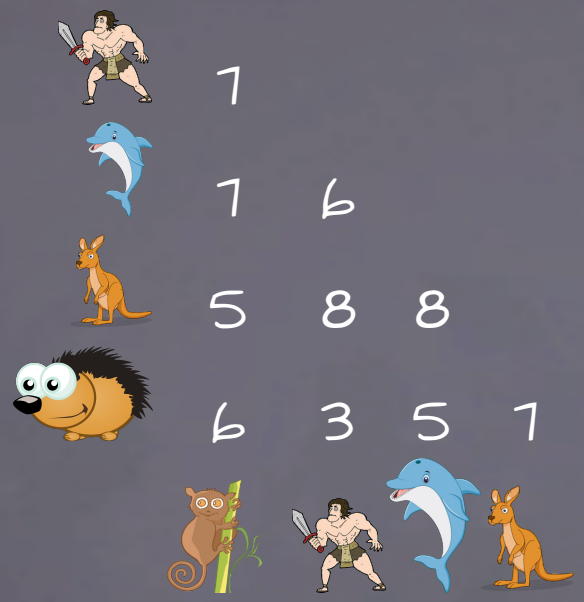
Branch lengths

$$2b(X) = \frac{\sum_Z (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2}$$



Distance-Based Reconstruction

Problem: correct pairs may not be closest



Neighbor Joining (unrooted)

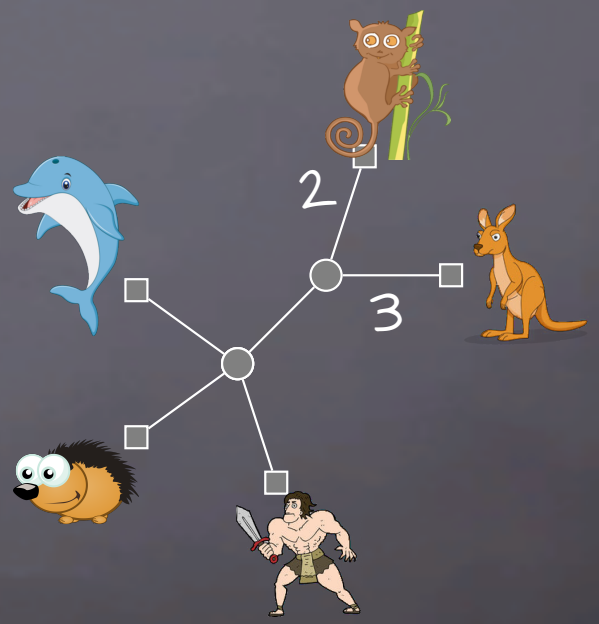
- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse

update distances

$$d_{X \cup Y,Z} = {}^1\!/_2 (d_{X,Z} + d_{Y,Z} - d_{X,Y})$$

Branch lengths

$$2b(X) = \frac{\sum_Z (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2}$$



Distance-Based Reconstruction

Problem: correct pairs may not be closest

Neighbor Joining (unrooted)

- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse

update distances

$$d_{X \cup Y,Z} = {}^1\!/_2 (d_{X,Z} + d_{Y,Z} - d_{X,Y})$$

Branch lengths

$$2b(X) = \frac{\sum_Z (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2}$$

Distance-Based Reconstruction

Problem: correct pairs may not be closest

Neighbor Joining (unrooted)

- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse

update distances

$$d_{X \cup Y,Z} = {}^1\!/_2 \left(d_{X,Z} + d_{Y,Z} - d_{X,Y}\right)$$

Branch lengths

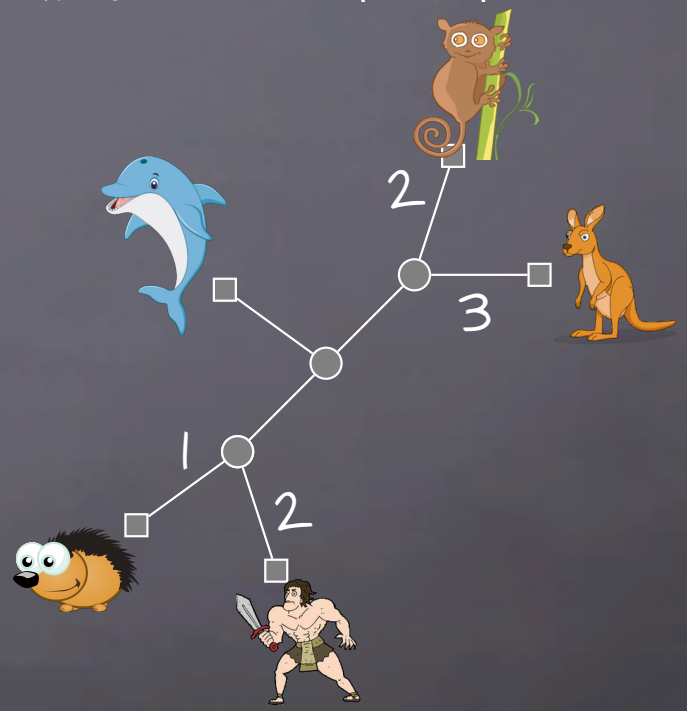
$$2b(X) = \frac{\sum_Z (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2}$$

Distance-Based Reconstruction

Problem: correct pairs may not be closest

Neighbor Joining (unrooted)

- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse



update distances

$$d_{X \cup Y,Z} = {}^1\!/_2 (d_{X,Z} + d_{Y,Z} - d_{X,Y})$$

Branch lengths

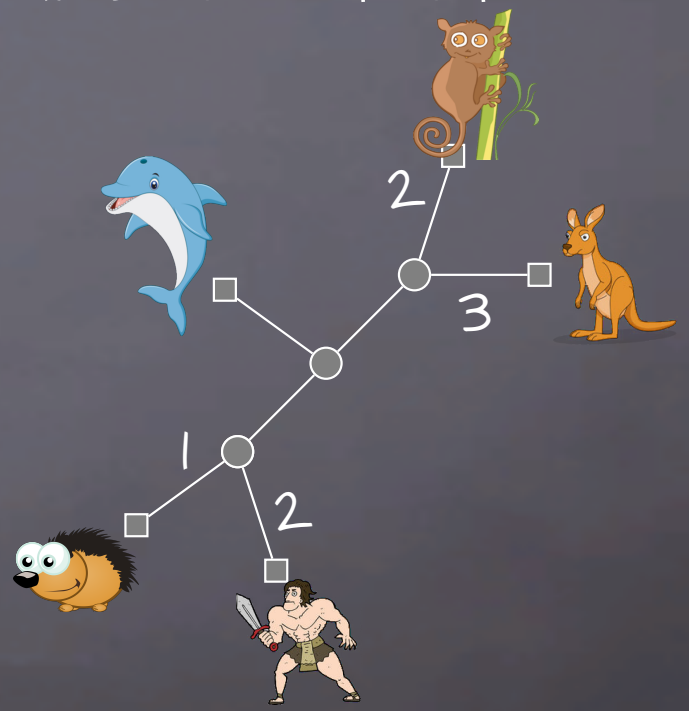
$$2b(X) = \frac{\sum_Z (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2}$$

Distance-Based Reconstruction

Problem: correct pairs may not be closest

Neighbor Joining (unrooted)

- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse



update distances

$$d_{X \cup Y,Z} = {}^1\!/_2 \left(d_{X,Z} + d_{Y,Z} - d_{X,Y}\right)$$

Branch lengths

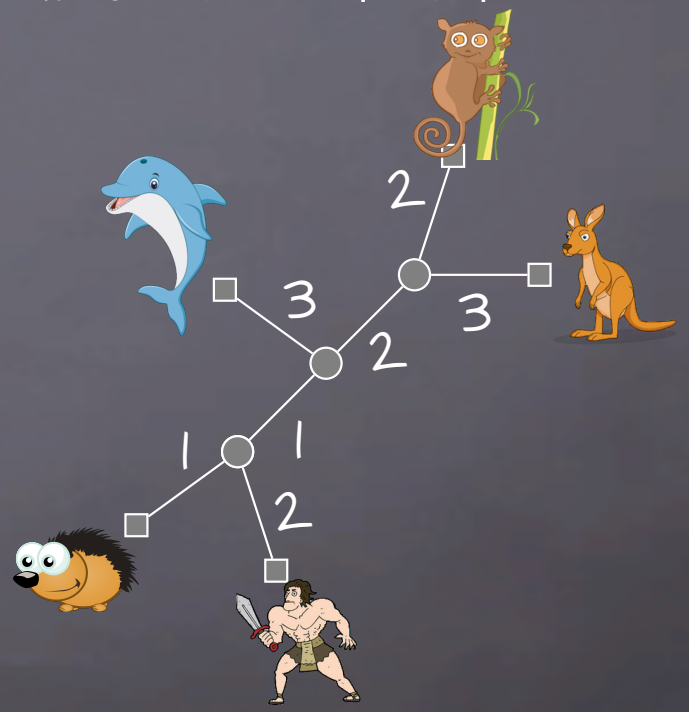
$$2b(X) = \frac{\sum_Z (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2}$$

Distance-Based Reconstruction

Problem: correct pairs may not be closest

Neighbor Joining (unrooted)

- Build eccentricity matrix:
  $Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y}$
- find max in Q
- join them
- update distances & recurse



update distances

$$d_{X \cup Y,Z} = {}^1\!/_2 (d_{X,Z} + d_{Y,Z} - d_{X,Y})$$

Branch lengths

$$2b(X) = \frac{\sum_Z (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2}$$

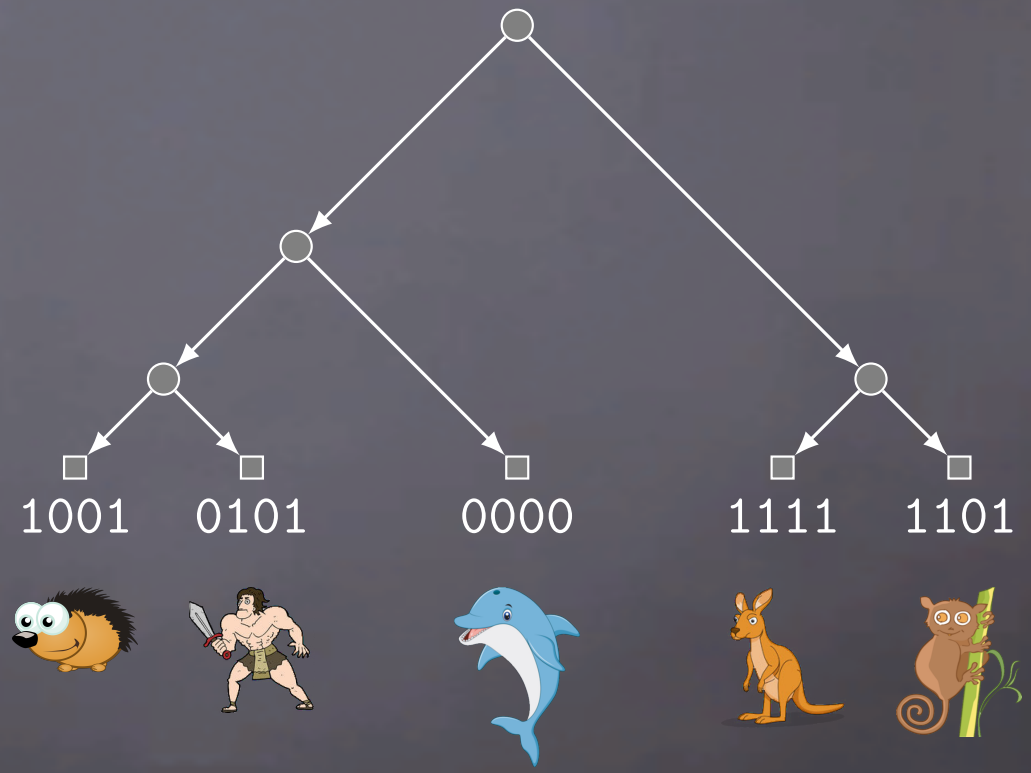
Parsimony Reconstructing

Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓

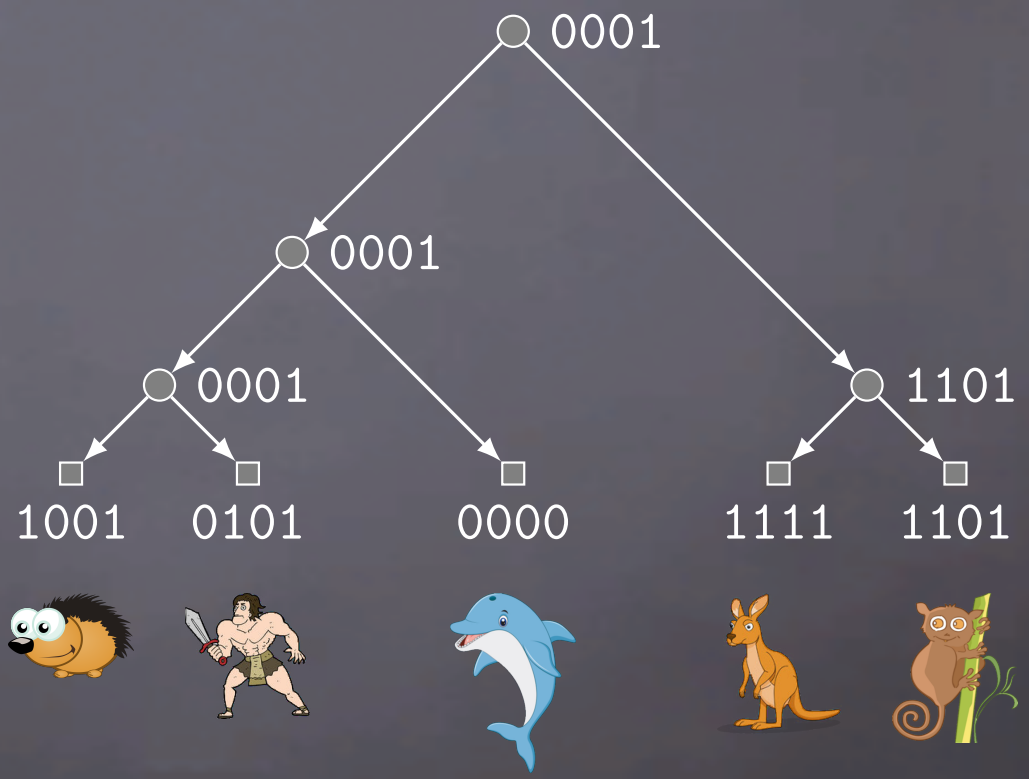
Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓



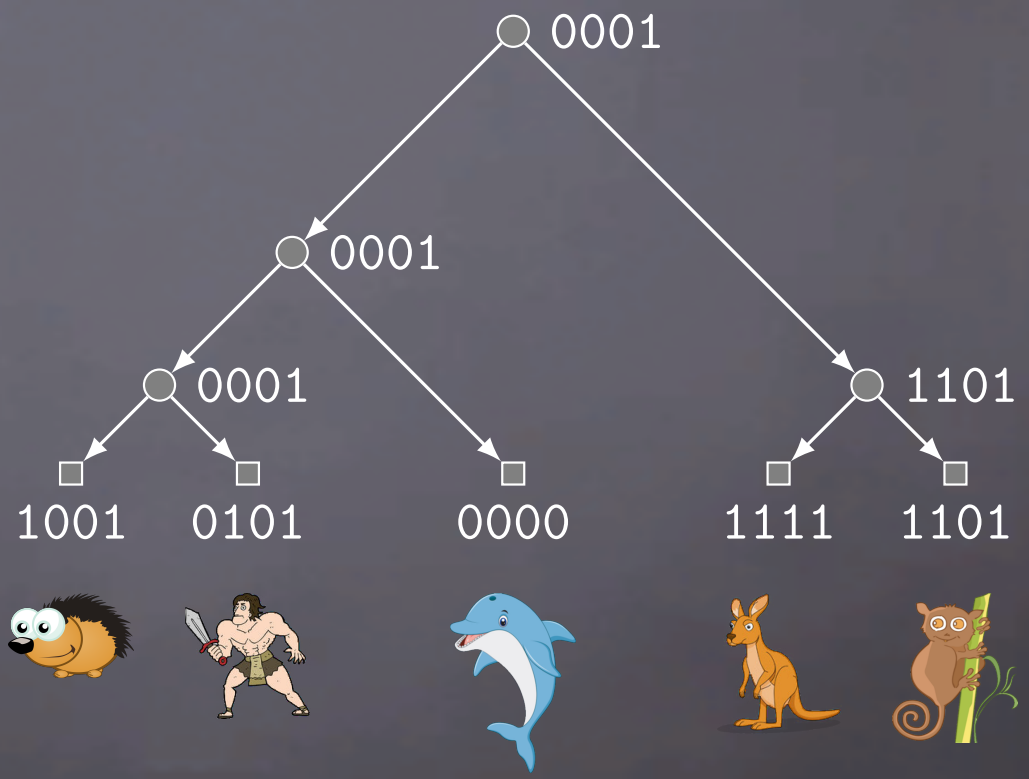
Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓



Parsimony Reconstructing

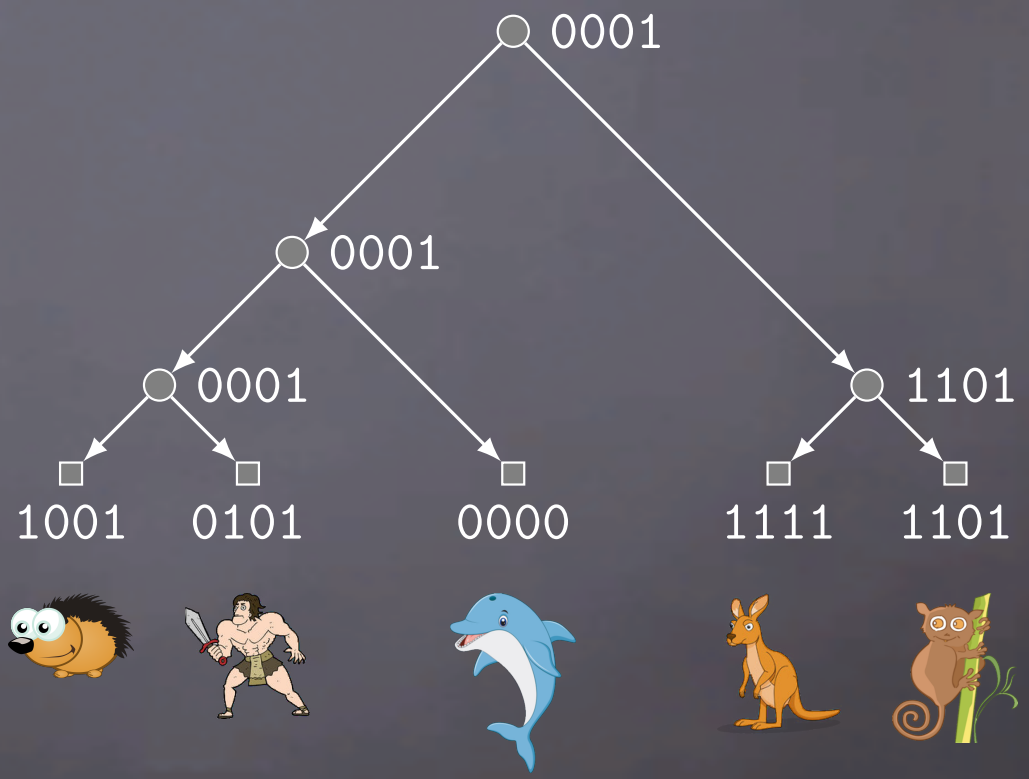
	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓



sum "distance" of endpoints of each edge

Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓



sum "distance" of endpoints of each edge ⇝ cost 6 (Hamming)

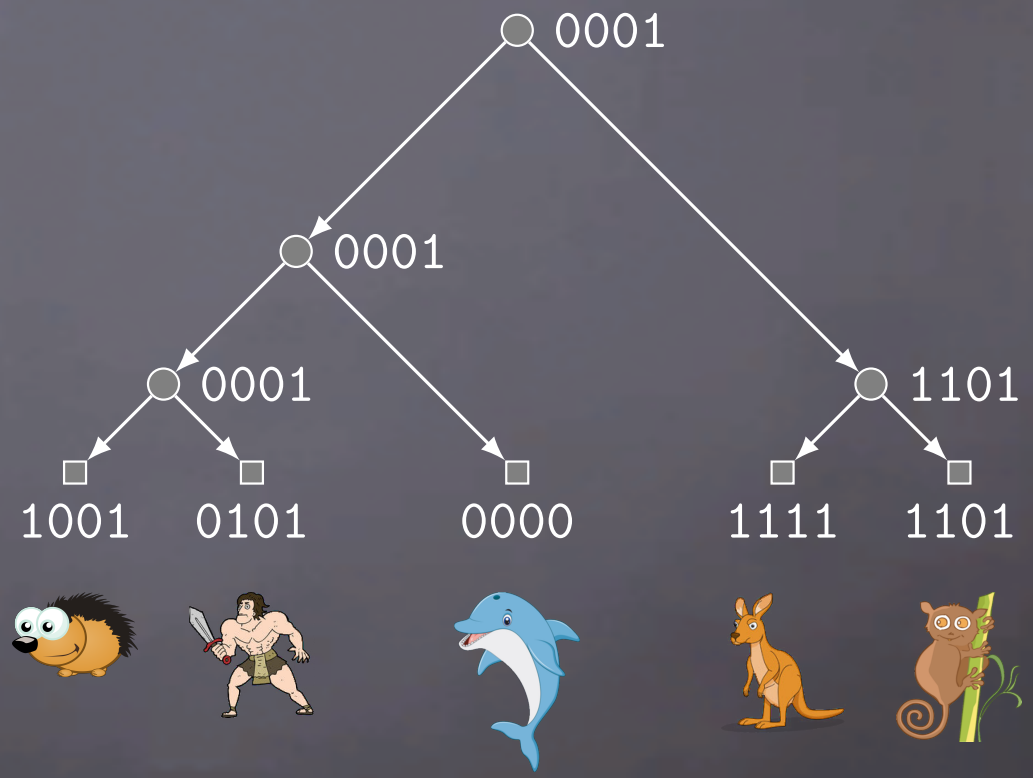
Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓

Small Parsimony

Input: character state matrix M, rooted tree T

Task: assign characters to internal nodes minimizing total cost



Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓

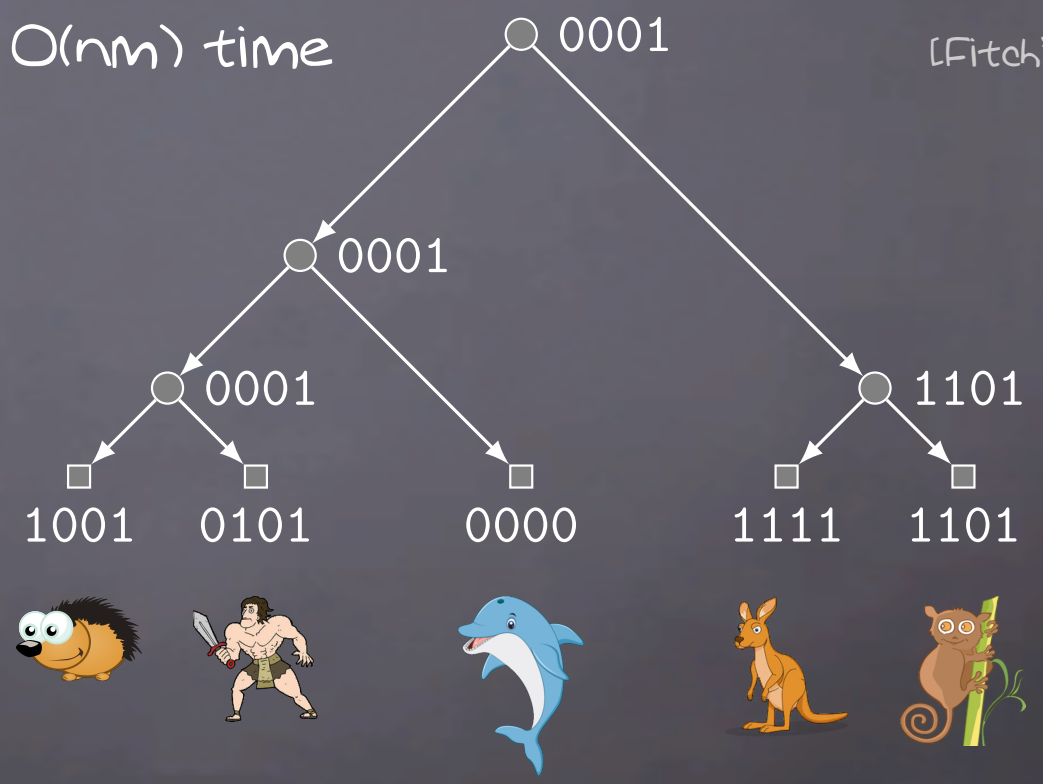
Small Parsimony

Input: character state matrix M, rooted tree T

Task: assign characters to internal nodes minimizing total cost

$\rightsquigarrow$ O(nm) time

[Fitch'71]



Algo (simple DFS):
Backtrack from v:
if v is leaf:
$\rightsquigarrow R(v) = \text{label}(v)$
else if $R(u) \cap R(w) = \emptyset$:
$\rightsquigarrow R(v) = R(u) \cup R(w)$
else:
$\rightsquigarrow R(v) = R(u) \cap R(w)$

Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓

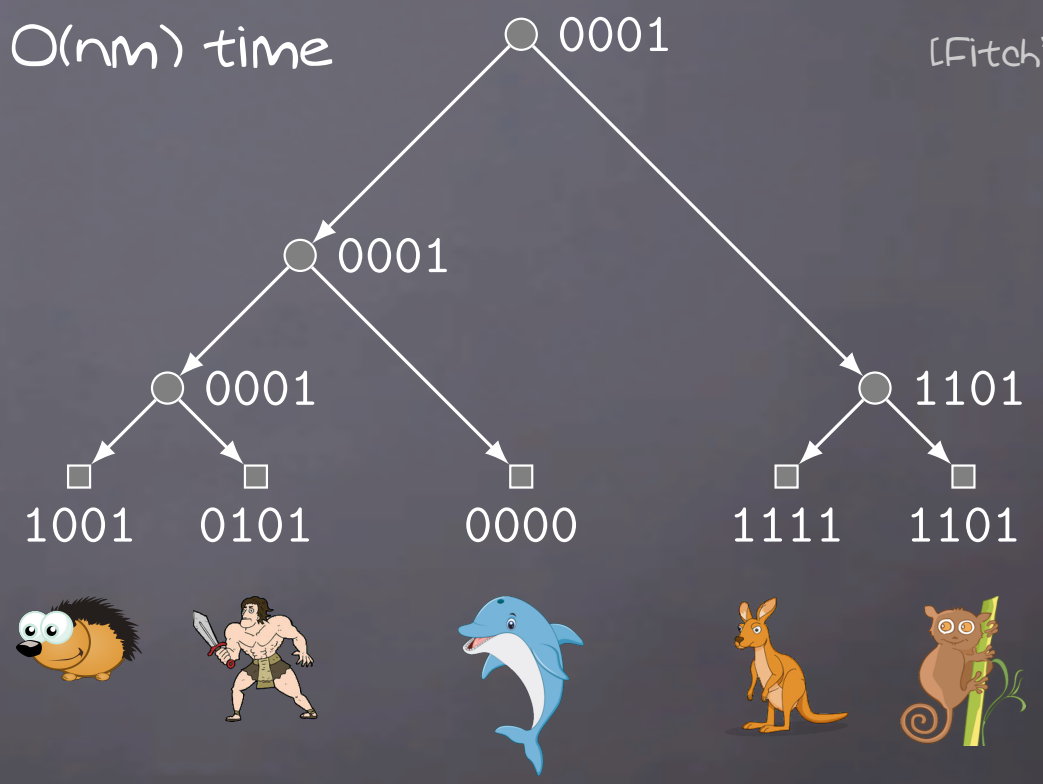
Small Parsimony

Input: character state matrix M, rooted tree T

Task: assign characters to internal nodes minimizing total cost

$\rightsquigarrow$ O(nm) time

[Fitch'71]



Algo (simple DFS):
Backtrack from v:
if v is leaf:
$\rightsquigarrow R(v) = \text{label}(v)$
else if $R(u) \cap R(w) = \emptyset$:
$\rightsquigarrow R(v) = R(u) \cup R(w)$
else:
$\rightsquigarrow R(v) = R(u) \cap R(w)$

Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓

Small Parsimony

Input: character state matrix M, rooted tree T

Task: assign characters to internal nodes minimizing total cost

$\rightsquigarrow$ $O(nm)$ time [Fitch'71]

Large Parsimony

Input: character state matrix M

Task: find tree T & assign characters to internal nodes minimizing total cost

Algo (simple DFS):
Backtrack from v:
if v is leaf:
$\rightsquigarrow$ $R(v)=\text{label}(v)$
else if $R(u) \cap R(w) = \emptyset$:
$\rightsquigarrow$ $R(v) = R(u) \cup R(w)$
else:
$\rightsquigarrow$ $R(v) = R(u) \cap R(w)$

Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓

Algo (simple DFS):
Backtrack from v:
if v is leaf:
$\rightsquigarrow R(v)=\text{label}(v)$
else if $R(u) \cap R(w) = \emptyset$:
$\rightsquigarrow R(v) = R(u) \cup R(w)$
else:
$\rightsquigarrow R(v) = R(u) \cap R(w)$

Small Parsimony

Input: character state matrix M, rooted tree T

Task: assign characters to internal nodes minimizing total cost

$\rightsquigarrow O(nm)$ time [Fitch'71]

Large Parsimony

Input: character state matrix M

Task: find tree T & assign characters to internal nodes minimizing total cost

$\rightsquigarrow$ NP-hard

Parsimony Reconstructing

	fur	AUS	pouch	land
	✓	✓	✗	✓
	✗	✓	✗	✓
	✗	✗	✗	✗
	✓	✓	✓	✓
	✓	✗	✗	✓

Small Parsimony

Input: character state matrix M, rooted tree T

Task: assign characters to internal nodes minimizing total cost

$\rightsquigarrow$ $O(nm)$ time [Fitch'71]

Large Parsimony

Input: character state matrix M

Task: find tree T & assign characters to internal nodes minimizing total cost

$\rightsquigarrow$ NP-hard

Note: alignment is crucial!

Algo (simple DFS):
Backtrack from v:
if v is leaf:
$\rightsquigarrow$ $R(v)=\text{label}(v)$
else if $R(u) \cap R(w) = \emptyset$:
$\rightsquigarrow$ $R(v) = R(u) \cup R(w)$
else:
$\rightsquigarrow$ $R(v) = R(u) \cap R(w)$

Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes

Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes
$\rightsquigarrow$ need model of evolution

Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes
$\rightsquigarrow$ need model of evolution

Jukes & Cantor Model

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate μ of mutation
- each base is equally likely to be result of mutation

Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes
$\rightsquigarrow$ need model of evolution

Jukes & Cantor Model

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate μ of mutation
- each base is equally likely to be result of mutation

Generalized Time Reversible Model

- each base evolves individually
- each base X has a frequency π_X to occur in the genome
- each base-substitution has its own rate of occurance

Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes
$\rightsquigarrow$ need model of evolution

Jukes & Cantor Model

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate μ of mutation
- each base is equally likely to be result of mutation

Generalized Time Reversible Model

- each base evolves individually
- each base X has a frequency π_X to occur in the genome
- each base-substitution has its own rate of occurance

compute likelihood, given tree & parameters $\rightsquigarrow$ $O(mn)$ time

Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes
$\rightsquigarrow$ need model of evolution

Jukes & Cantor Model

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate μ of mutation
- each base is equally likely to be result of mutation

Generalized Time Reversible Model

- each base evolves individually
- each base X has a frequency π_X to occur in the genome
- each base-substitution has its own rate of occurance

compute likelihood, given tree & parameters $\rightsquigarrow$ $O(mn)$ time
find best tree & parameters $\rightsquigarrow$ NP-hard

Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes
$\rightsquigarrow$ need model of evolution

Jukes & Cantor Model

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate μ of mutation
- each base is equally likely to be result of mutation

Generalized Time Reversible Model

- each base evolves individually
- each base X has a frequency π_X to occur in the genome
- each base-substitution has its own rate of occurance

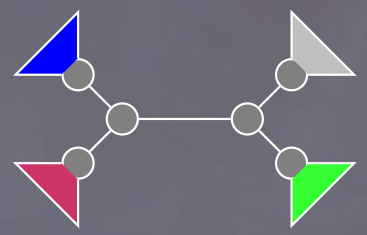
compute likelihood, given tree & parameters $\rightsquigarrow$ $O(mn)$ time
find best tree & parameters $\rightsquigarrow$ NP-hard
$\rightsquigarrow$ local search in the tree space

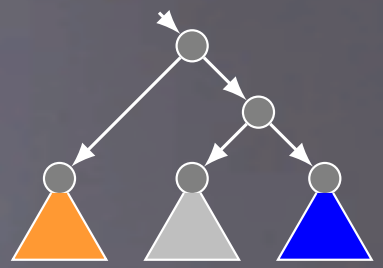
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration
of 4 (3) "neighboring"
subtrees into another



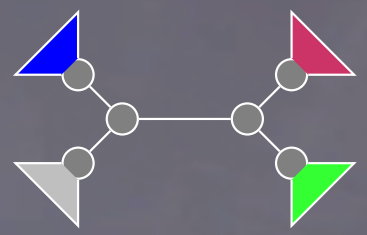


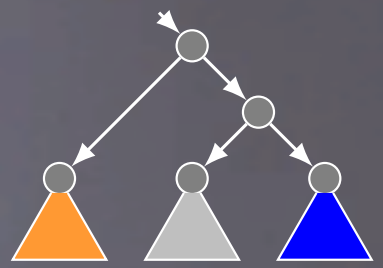
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration
of 4 (3) "neighboring"
subtrees into another



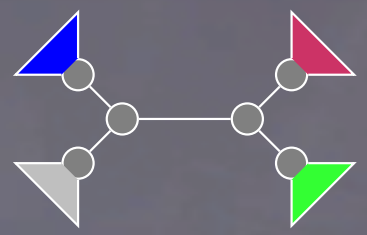


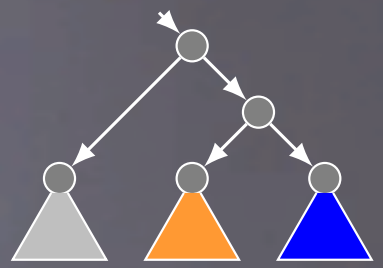
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration
of 4 (3) "neighboring"
subtrees into another



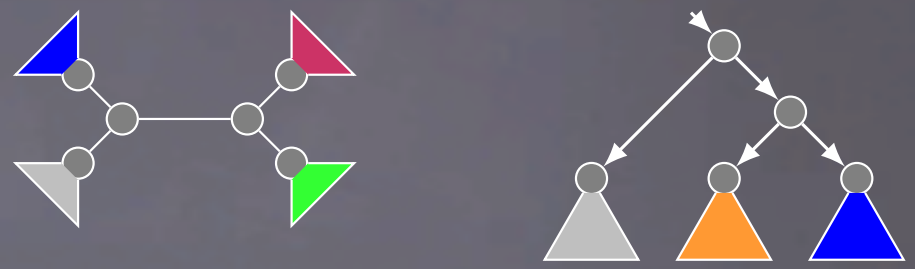


ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another



Subtree Prune & Regraft

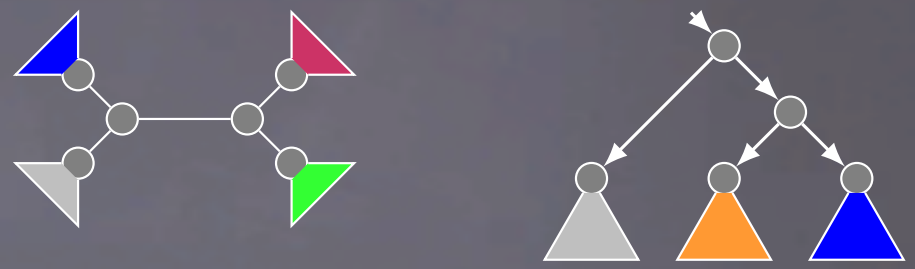
Break any edge uv & connect v to any edge of the component of u

ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another



Subtree Prune & Regraft

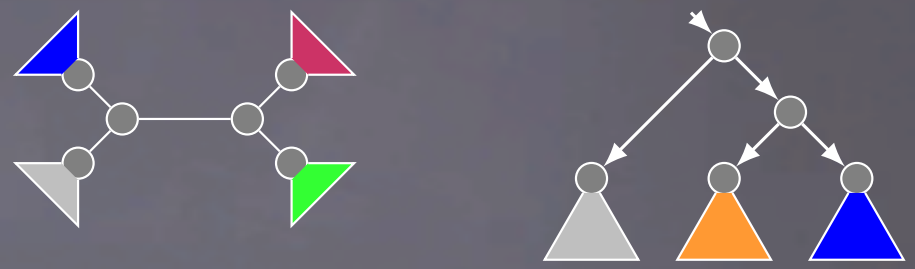
Break any edge uv & connect v to any edge of the component of u

ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another



Subtree Prune & Regraft

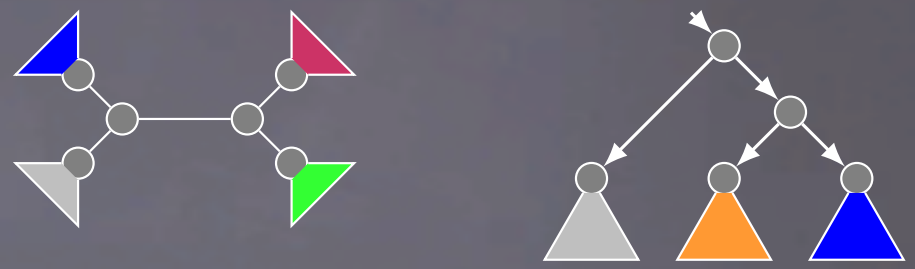
Break any edge uv & connect v to any edge of the component of u

ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another



Subtree Prune & Regraft

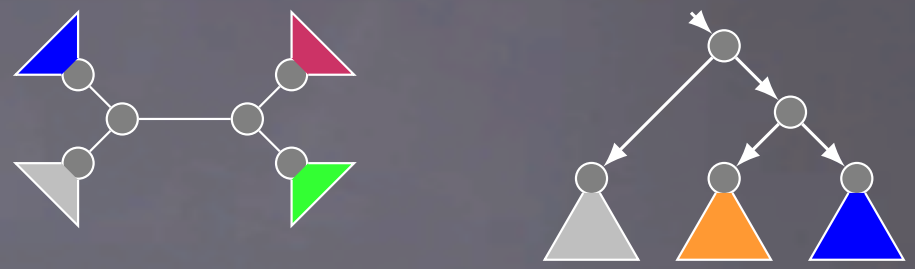
Break any edge uv & connect v to any edge of the component of u

ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another



Subtree Prune & Regraft

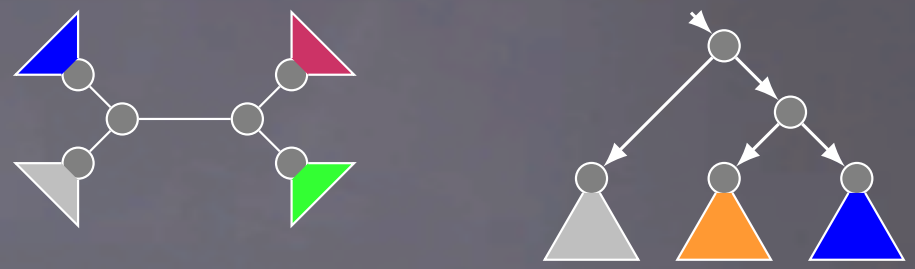
Break any edge uv & connect v to any edge of the component of u

ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another

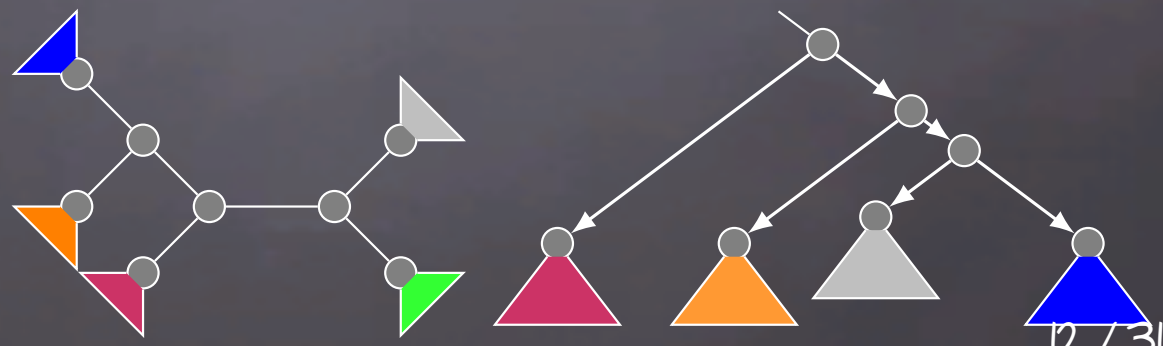


Subtree Prune & Regraft

Break any edge uv & connect v to any edge of the component of u

Tree Bisection & Reconnection

Break any edge & insert a new reconnecting edge "between" any 2 edges

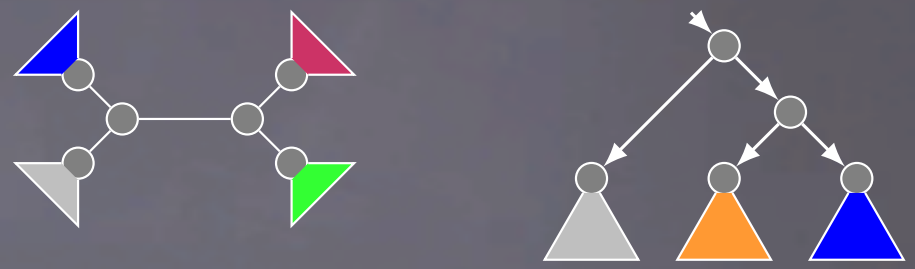


ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another

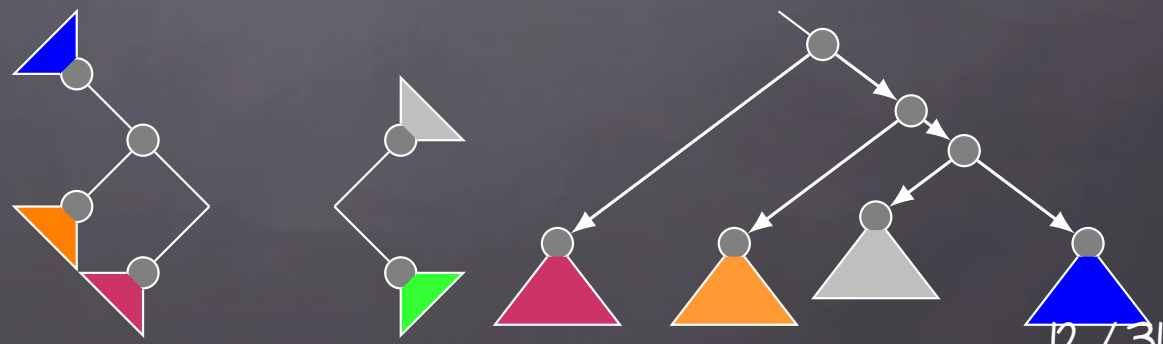


Subtree Prune & Regraft

Break any edge uv & connect v to any edge of the component of u

Tree Bisection & Reconnection

Break any edge & insert a new reconnecting edge "between" any 2 edges

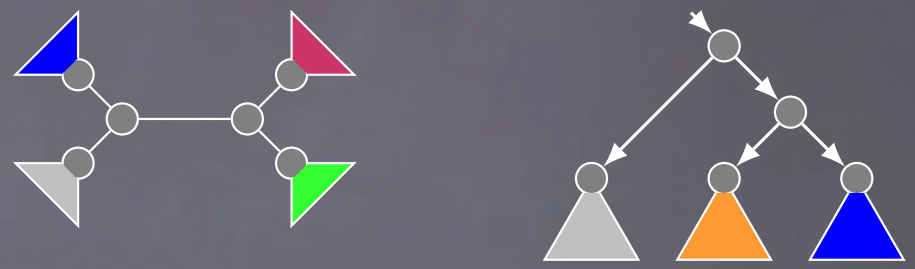


ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another

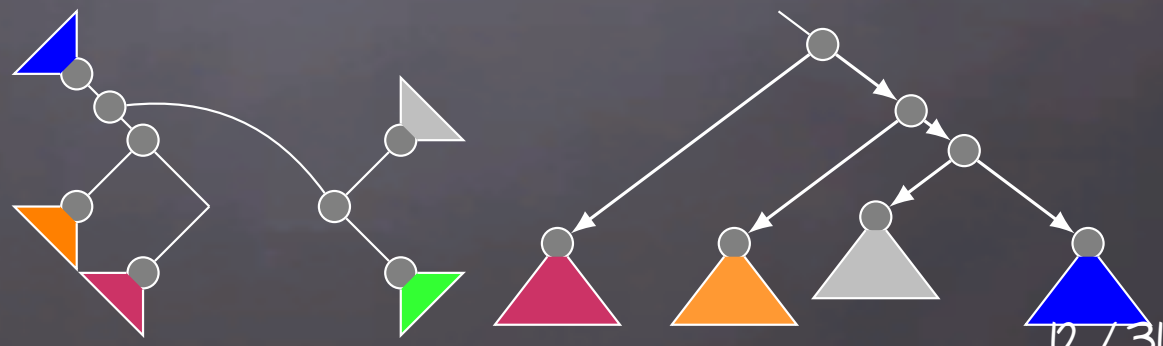


Subtree Prune & Regraft

Break any edge uv & connect v to any edge of the component of u

Tree Bisection & Reconnection

Break any edge & insert a new reconnecting edge "between" any 2 edges

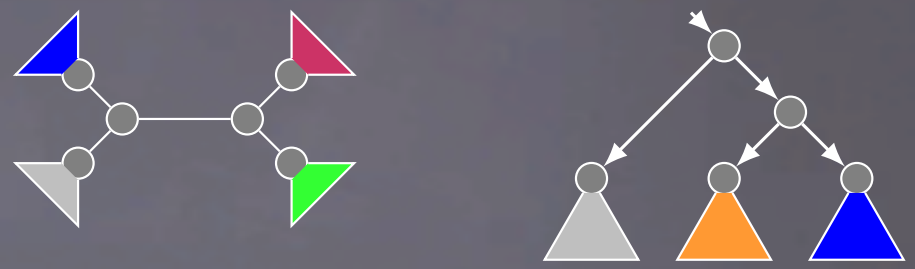


ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another

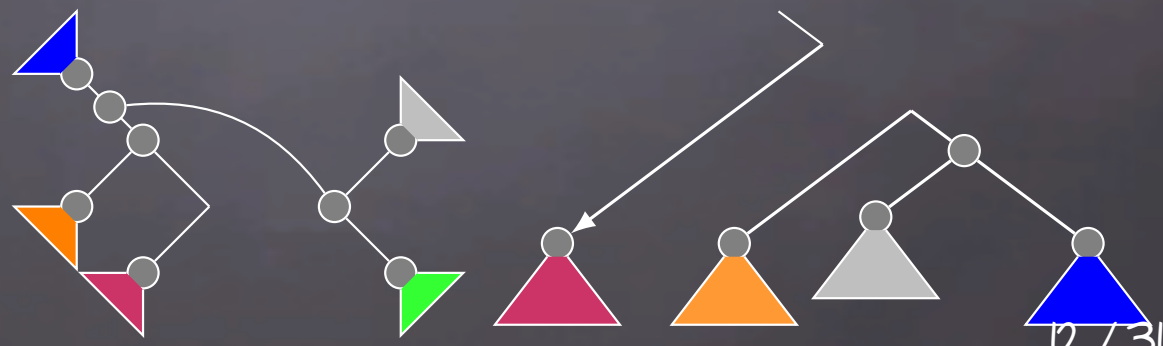


Subtree Prune & Regraft

Break any edge uv & connect v to any edge of the component of u

Tree Bisection & Reconnection

Break any edge & insert a new reconnecting edge "between" any 2 edges

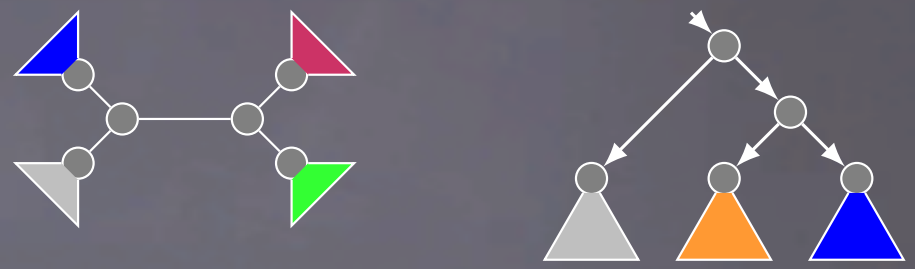


ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another

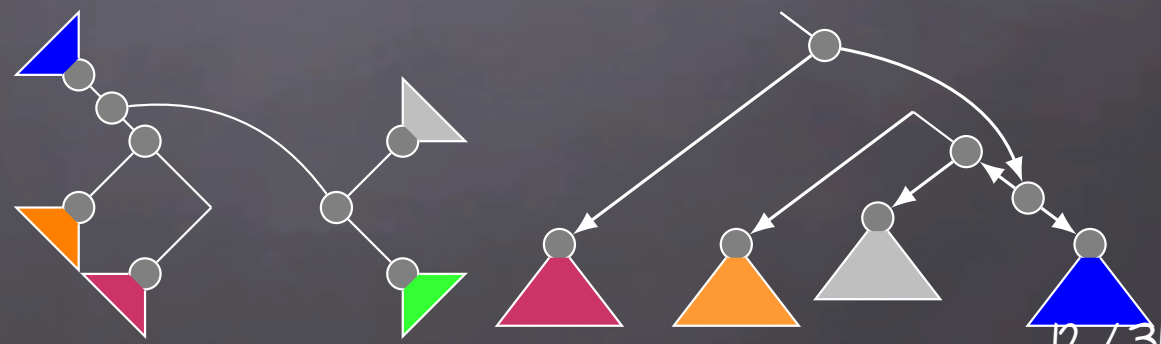


Subtree Prune & Regraft

Break any edge uv & connect v to any edge of the component of u

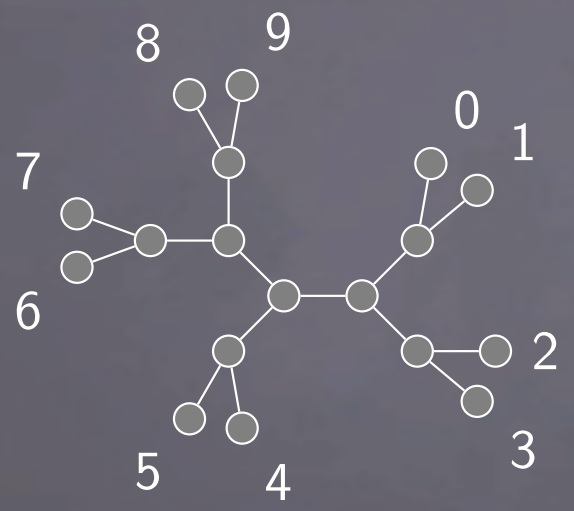
Tree Bisection & Reconnection

Break any edge & insert a new reconnecting edge "between" any 2 edges



ML Reconstruction - Tree Spaces

Exercise: turn into (any) caterpillar:



Exercise: how are the distances related?

Checking Robustness – Bootstrap Method

suppose: method $\mathcal{X}$ yields tree T from $n \times m$ character-state matrix M
repeat k times the following experiment:

1. draw m columns from M (with repetition)
2. use $\mathcal{X}$ to compute T_i

Finally, for each branch of T, check how often it occurs in the T_i
$\rightsquigarrow$ "bootstrap value" measures robustness ("support") of each branch

Reconstruction by Gene Trees

Reconstruction By Gene Trees

A Common Method For Reconstructing Trees

1. get genomes of multiple species
2. extract "genes" using START & STOP codons
3. cluster genes in "families" of similar genes
4. within each family, infer a "gene tree" using dissimilarities
5. Build a consensus among the gene trees $\rightsquigarrow$ "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes

Reconstruction By Gene Trees

A Common Method For Reconstructing Trees

1. ~~get genomes of multiple species~~
2. ~~extract "genes" using START & STOP codons~~
3. cluster genes in "families" of similar genes
4. within each family, infer a "gene tree" using dissimilarities
5. Build a consensus among the gene trees $\rightsquigarrow$ "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes

Reconstruction By Gene Trees

A Common Method For Reconstructing Trees

1. ~~get genomes of multiple species~~
2. ~~extract "genes" using START & STOP codons~~
3. ~~cluster genes in "families" of similar genes~~
4. within each family, infer a "gene tree" using dissimilarities
5. Build a consensus among the gene trees $\rightsquigarrow$ "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes

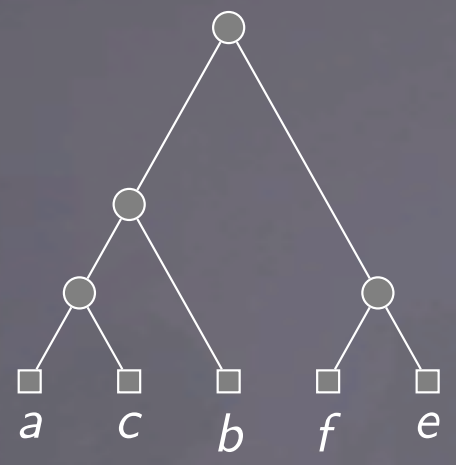
Reconstruction By Gene Trees

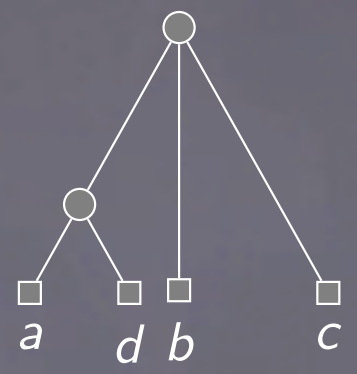
A Common Method For Reconstructing Trees

1. ~~get genomes of multiple species~~
2. ~~extract "genes" using START & STOP codons~~
3. ~~cluster genes in "families" of similar genes~~
4. ~~within each family, infer a "gene tree" using dissimilarities~~
5. Build a consensus among the gene trees $\rightsquigarrow$ "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes

Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)



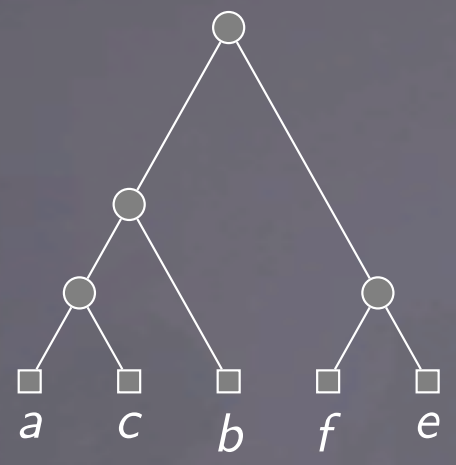


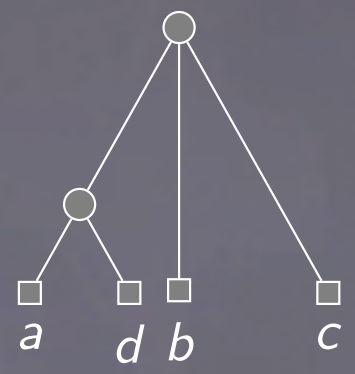
Algo [Aho et al'81]

1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)





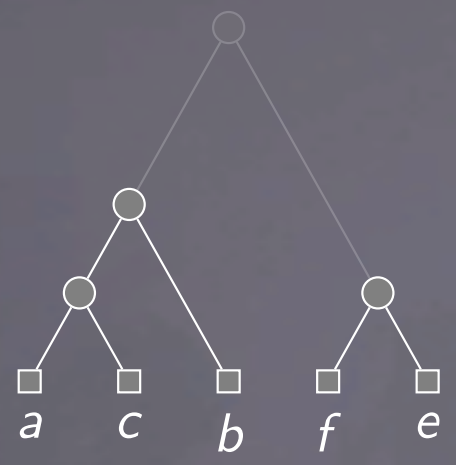
b	c
d	a
e	f

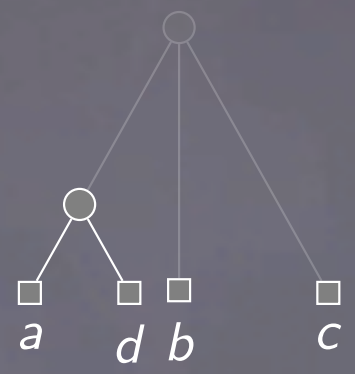
Algo [Aho et al'81]

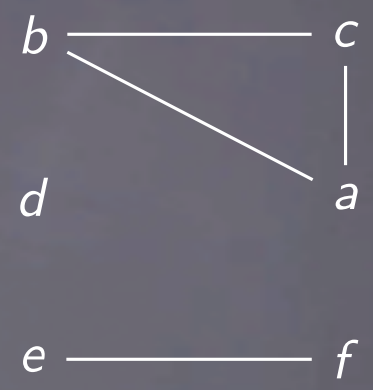
1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)





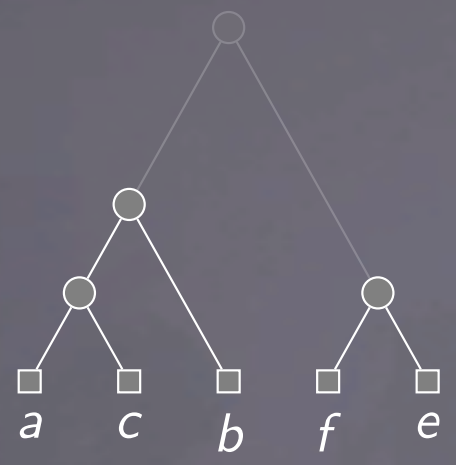


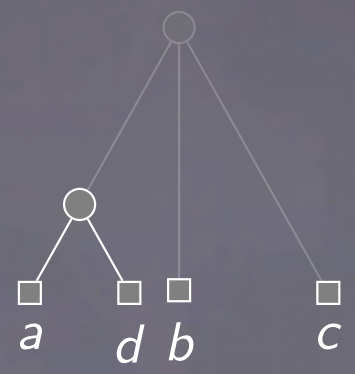
Algo [Aho et al'81]

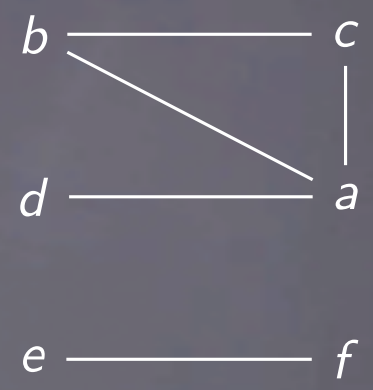
1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)





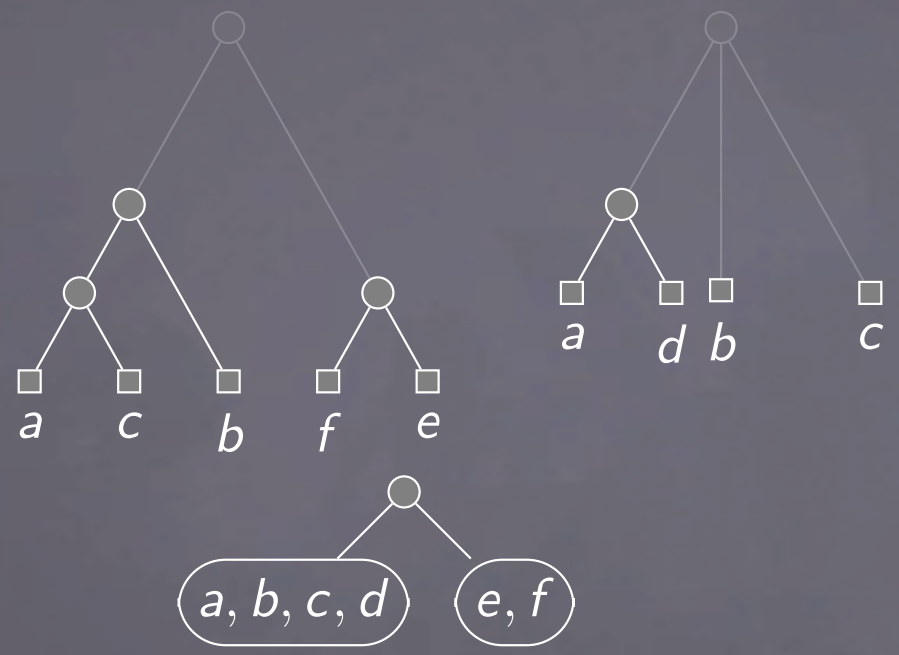


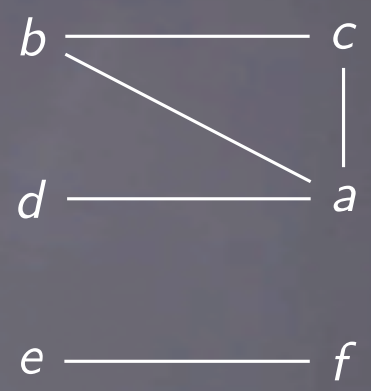
Algo [Aho et al'81]

1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)



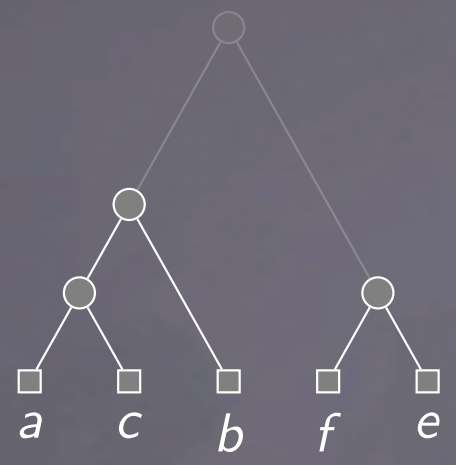


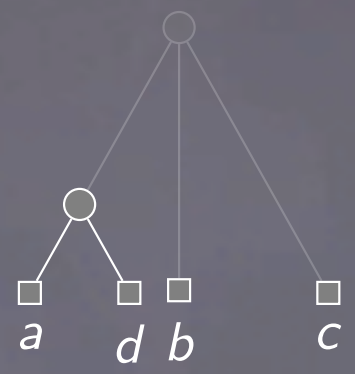
Algo [Aho et al'81]

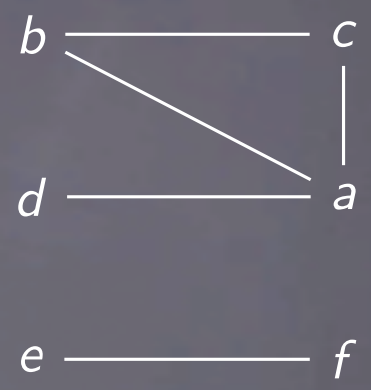
1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

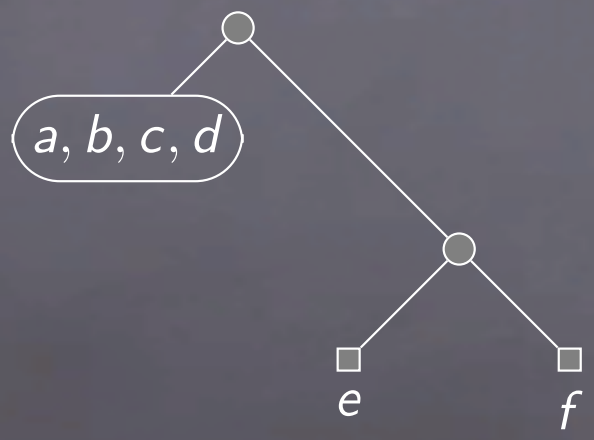
Supertrees – "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)







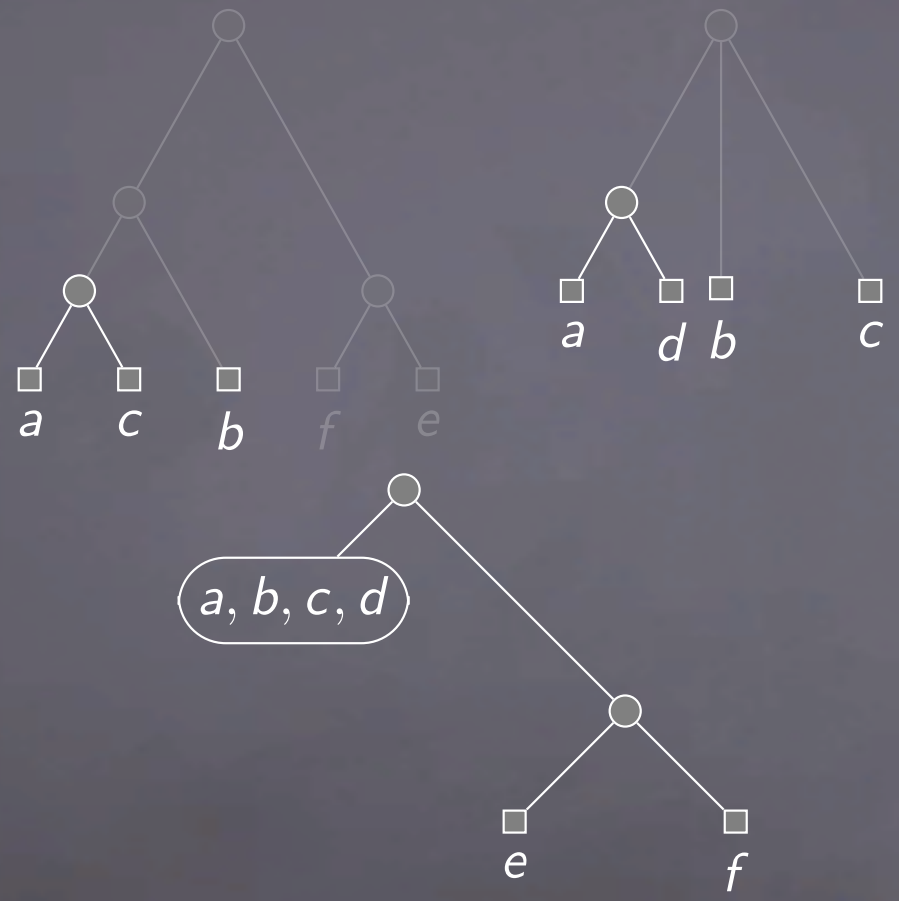


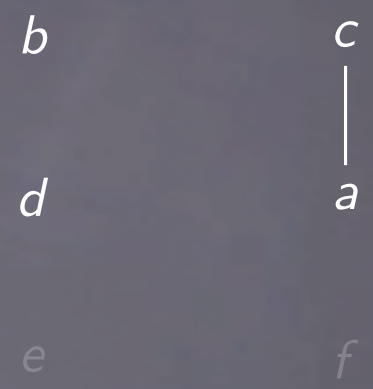
Algo [Aho et al'81]

1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

Supertrees – "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)



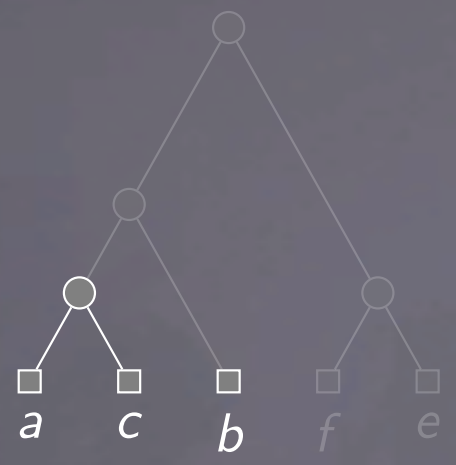


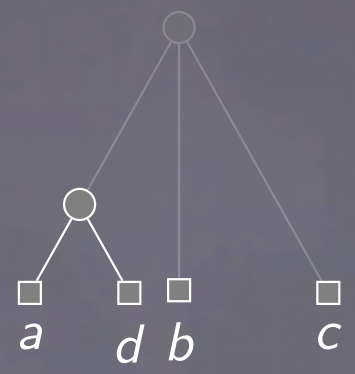
Algo [Aho et al'81]

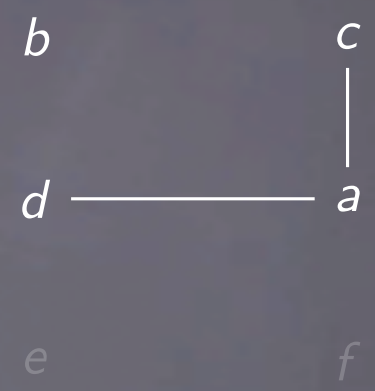
1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

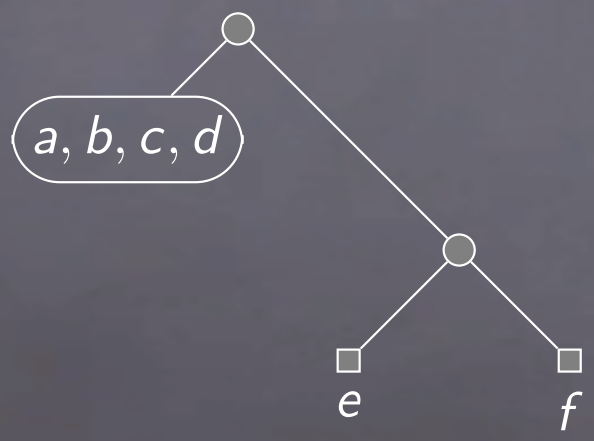
Supertrees – "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)







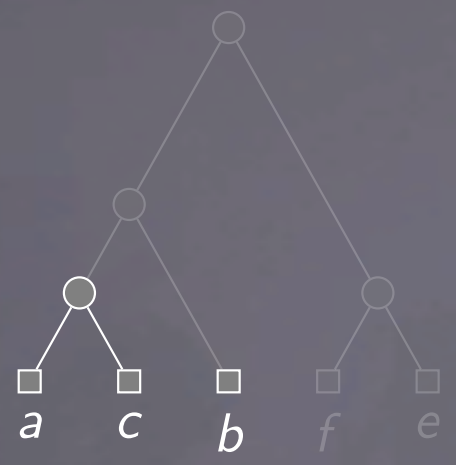


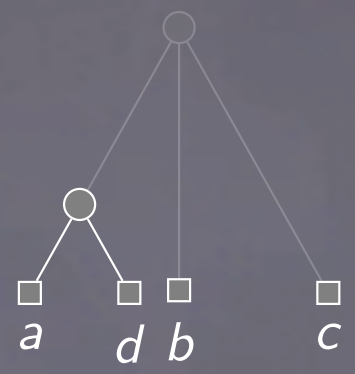
Algo [Aho et al'81]

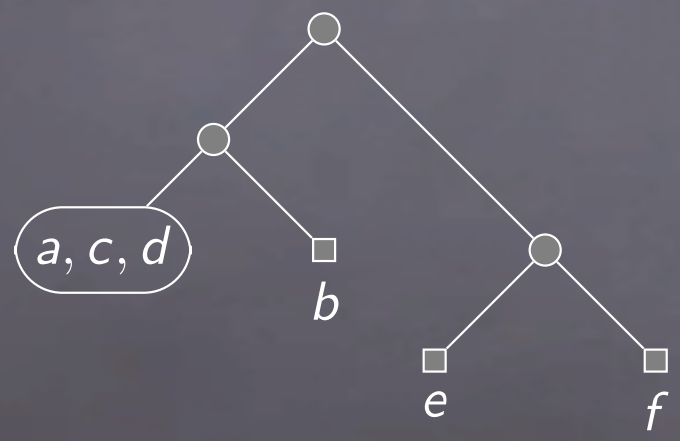
1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

Supertrees – "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)







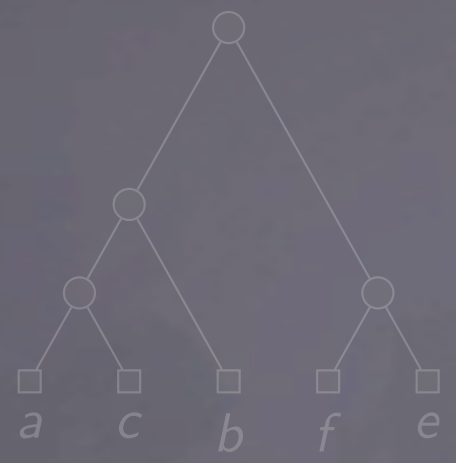
b	c
d	a
e	f

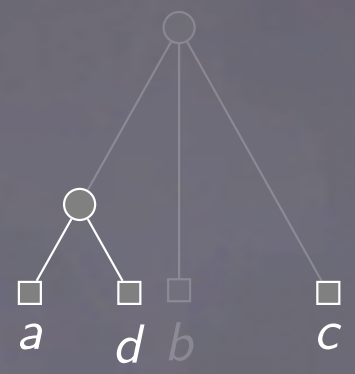
Algo [Aho et al'81]

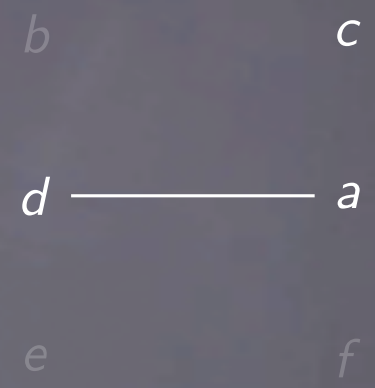
1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

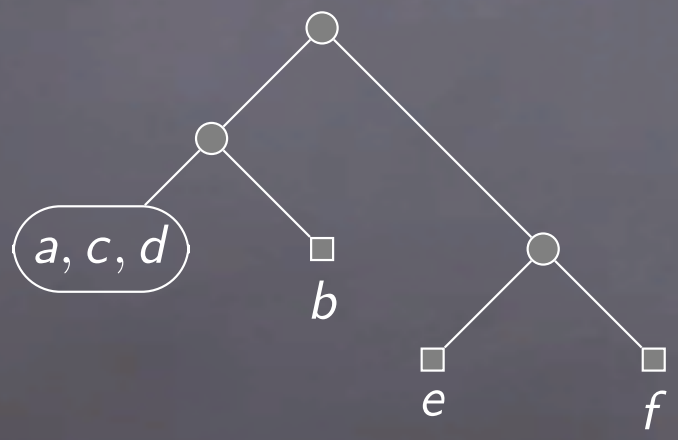
Supertrees – "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)







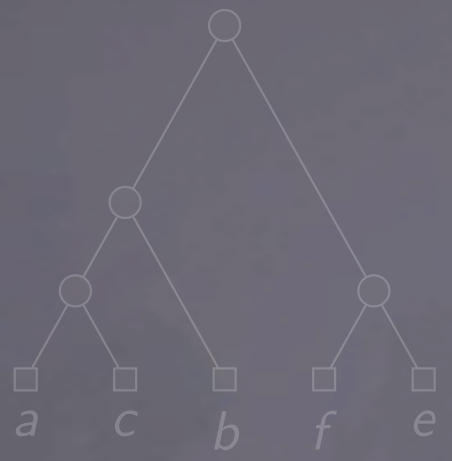


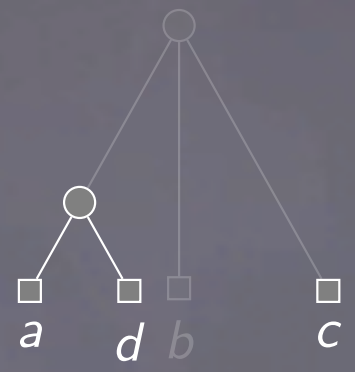
Algo [Aho et al'81]

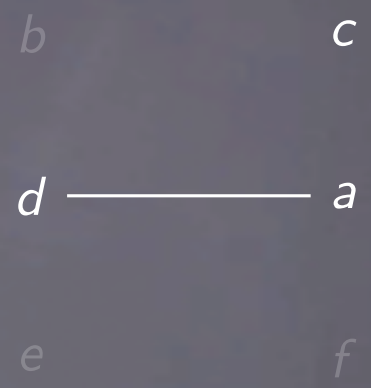
1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

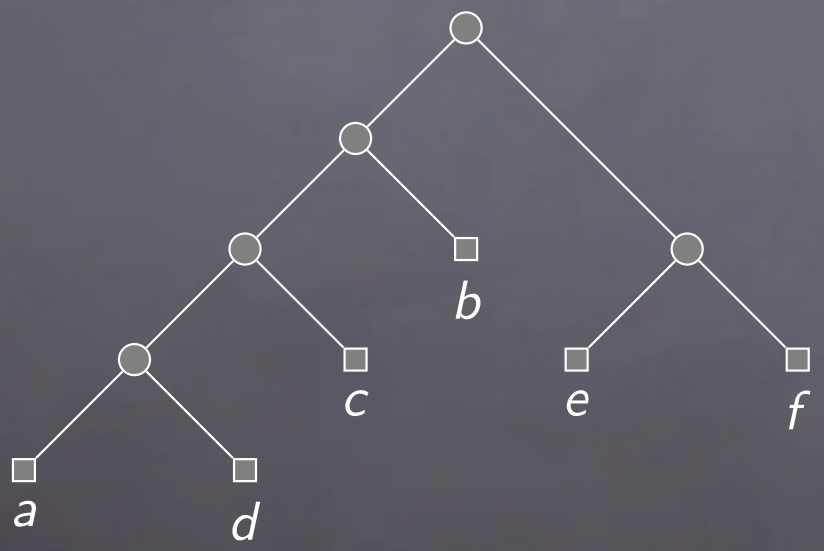
Supertrees – "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)







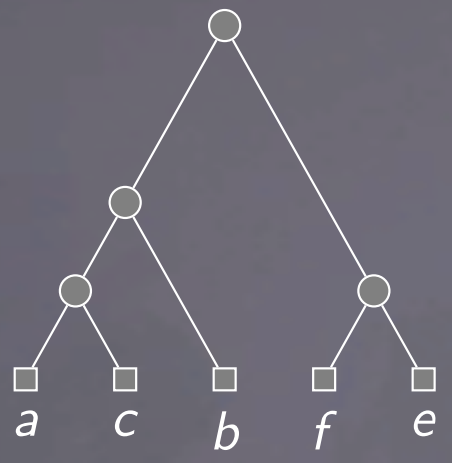


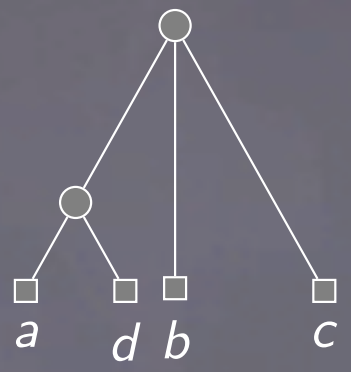
Algo [Aho et al'81]

1. Build graph G with edge $uv \Leftrightarrow \exists uv|x$
2. recurse for each component of G
3. plug subtrees to root

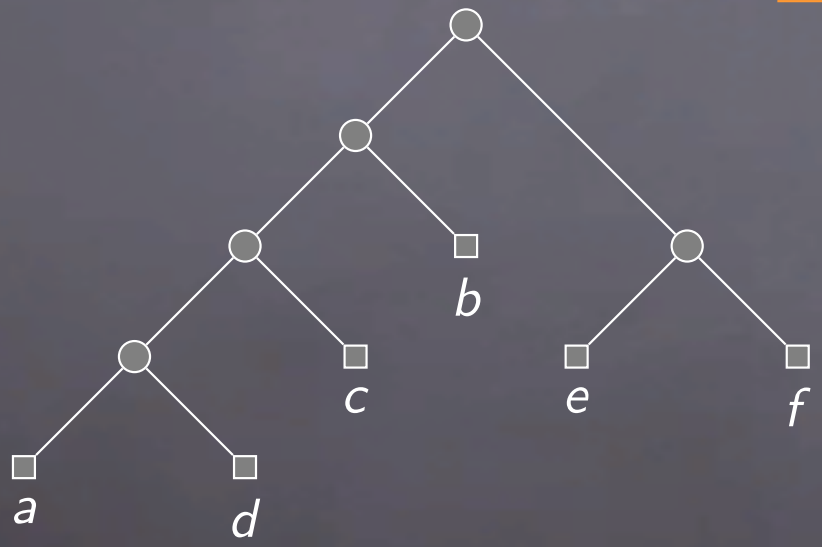
Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)



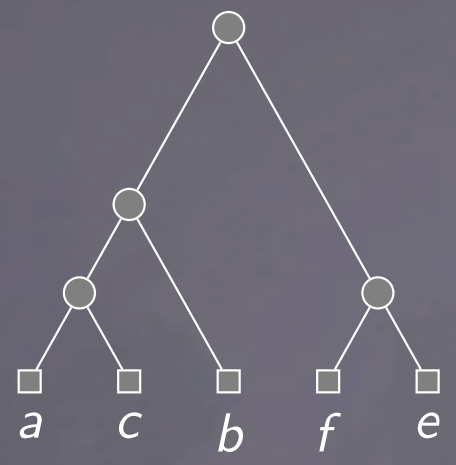


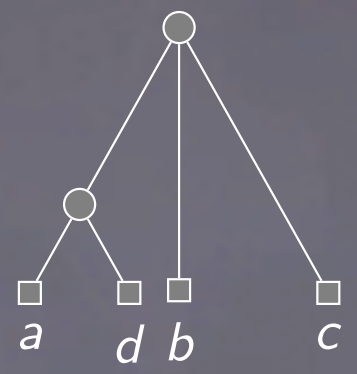
Note: always works if trees are compatible



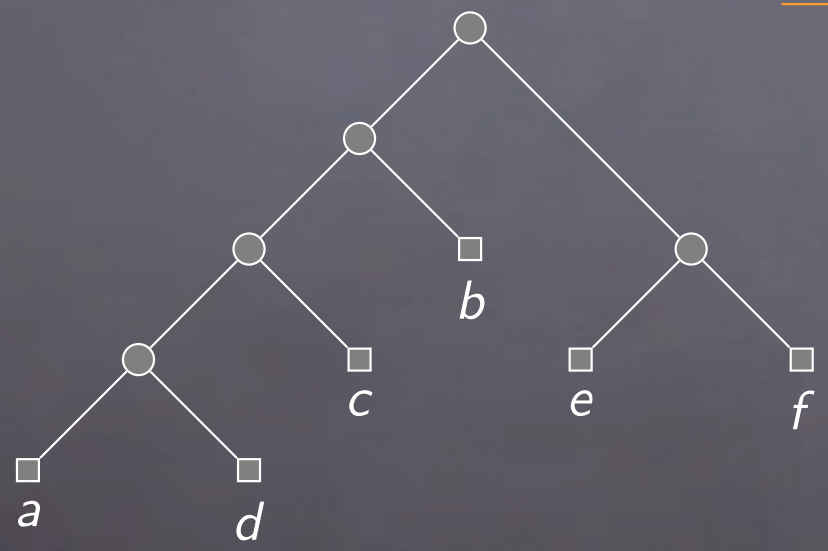
Supertrees – "Build" Algorithm

Idea: find root partition & recurse (as long as there are ≥ 3 taxa)





Note: always works if trees are compatible



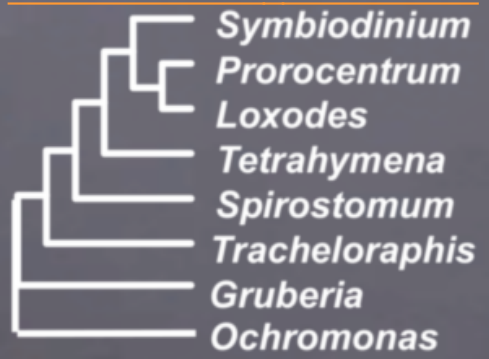
incompatible?

- largest compatible subset
  $\rightsquigarrow$ NP-hard (even for triplets)
- voting schemes
  (each tree votes for their clades)
- reinterpret clades as characters,
  combine into matrix & reconstruct

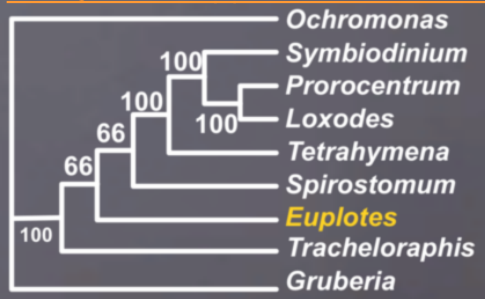
Consensi of Non-Agreeing Trees

strict consensus

consensus subtree



majority consensus



Reconstruction By Gene Trees

A Common Method For Reconstructing Trees

1. ~~get genomes of multiple species~~
2. ~~extract "genes" using START & STOP codons~~
3. ~~cluster genes in "families" of similar genes~~
4. ~~within each family, infer a "gene tree" using dissimilarities~~
5. Build a consensus among the gene trees $\rightsquigarrow$ "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes

Reconstruction By Gene Trees

A Common Method For Reconstructing Trees

1. ~~get genomes of multiple species~~
2. ~~extract "genes" using START & STOP codons~~
3. ~~cluster genes in "families" of similar genes~~
4. ~~within each family, infer a "gene tree" using dissimilarities~~
5. ~~Build a consensus among the gene trees $\rightsquigarrow$ "species tree"~~
   ~~(Note: species tree may differ significantly from individual gene trees)~~
6. reconcile all gene trees with the species tree to learn the evolution of those genes

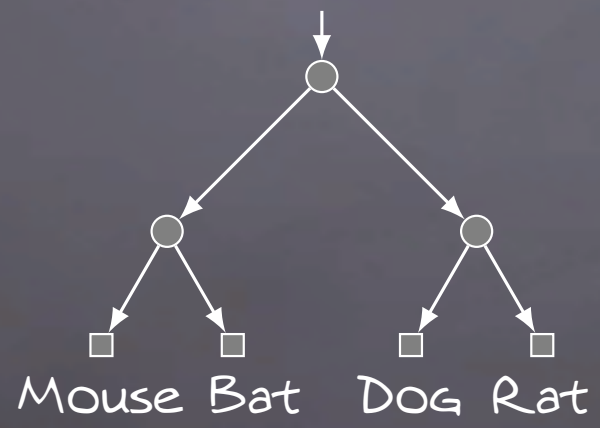
The History of a Gene Family

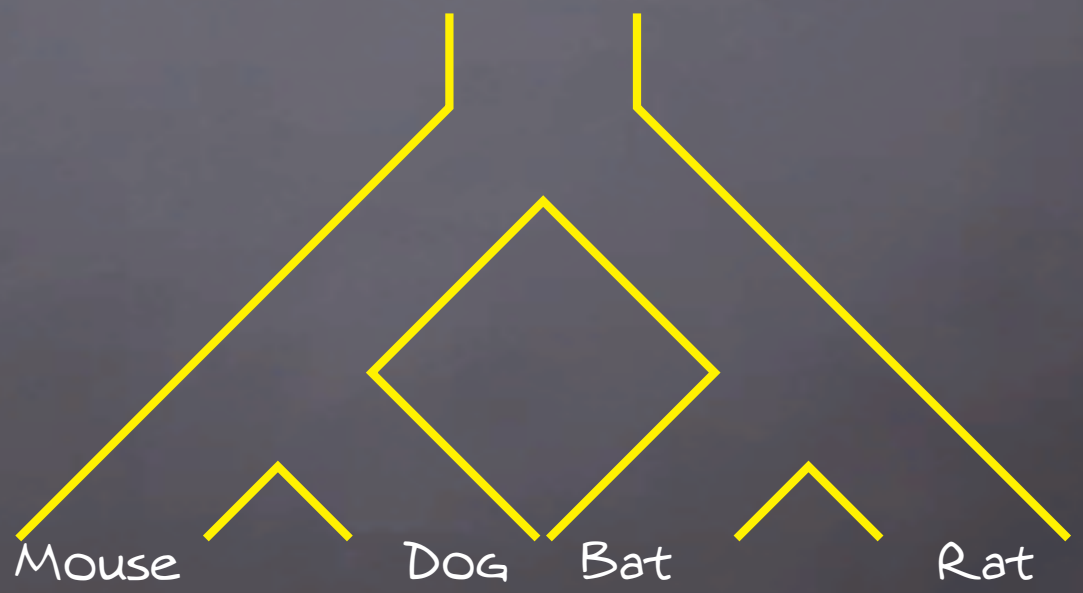
Recall

gene = "functional element" of DNA, clustered into gene-families

```
Mouse  GGAGCTTGAGCCGGAATAGTAGGAACATCTTTAAGAATTTTAATTCGAGC
Dog    GGAATCTGAACAGGCTTAGTAGCCACTAGAATAAGACTTTTAATTCGAGC
Bat    GGAATTTGAACAGGTTTAGTAGCCACTAGAATAAGACTCTTAATTCGAGC
Rat    GGAATTTGAACCGGCCTCGTAGCAACAAGAATAAGCTTATTAATCCGTGC
```

each family yields a tree depicting its history ⇝ "gene tree"
consensus of the gene trees yields "species tree"
But: what did really happen???





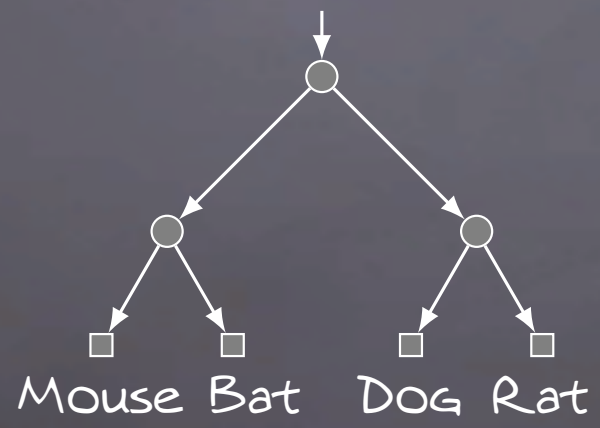
The History of a Gene Family

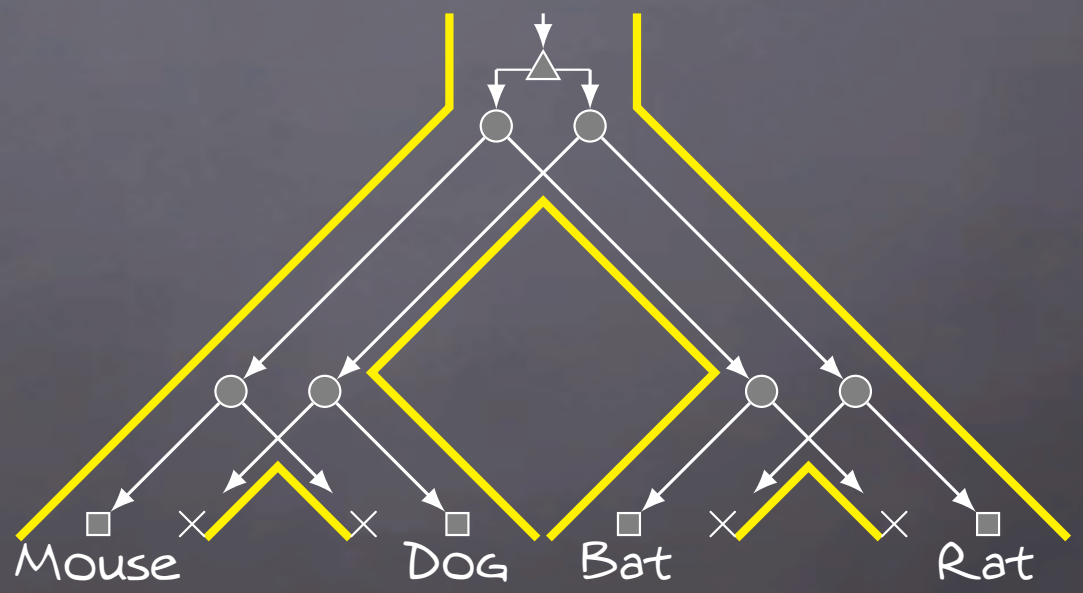
Recall

gene = "functional element" of DNA, clustered into gene-families

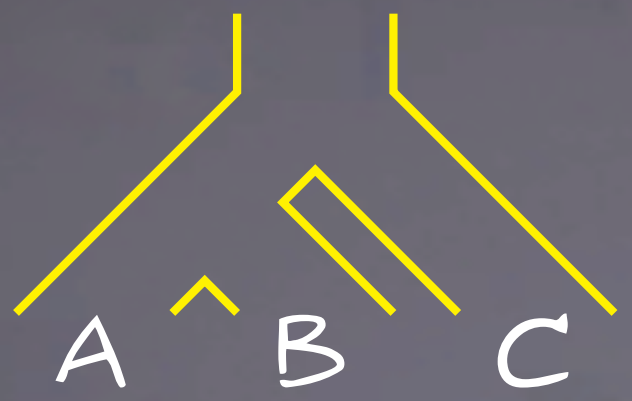
```
Mouse  GGAGCTTGAGCCGGAATAGTAGGAACATCTTTAAGAATTTTAATTCGAGC
Dog    GGAATCTGAACAGGCTTAGTAGCCACTAGAATAAGACTTTTAATTCGAGC
Bat    GGAATTTGAACAGGTTTAGTAGCCACTAGAATAAGACTCTTAATTCGAGC
Rat    GGAATTTGAACCGGCCTCGTAGCAACAAGAATAAGCTTATTAATCCGTGC
```

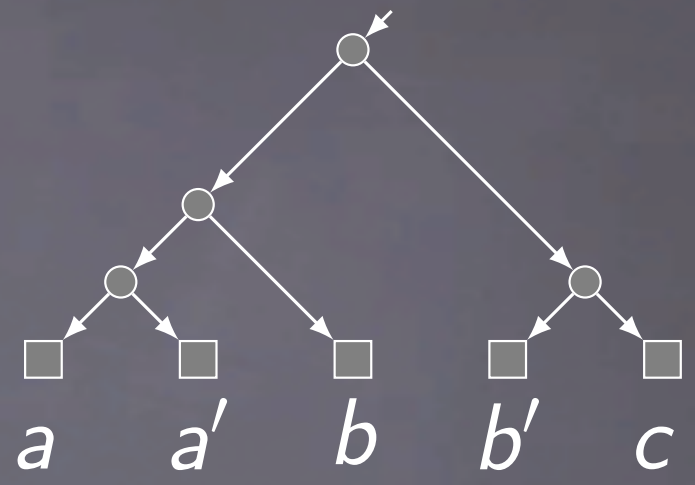
each family yields a tree depicting its history ⇝ "gene tree"
consensus of the gene trees yields "species tree"
But: what did really happen???





Reconciliation



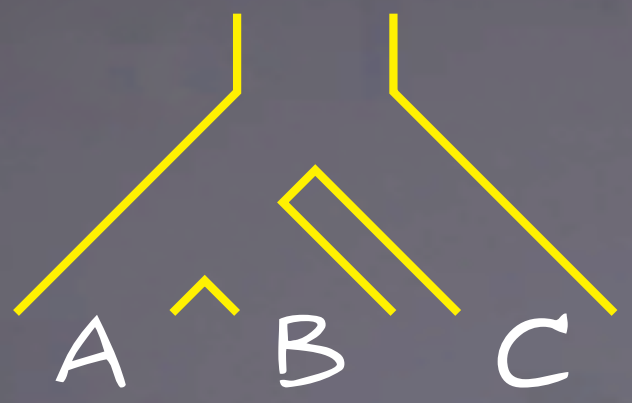


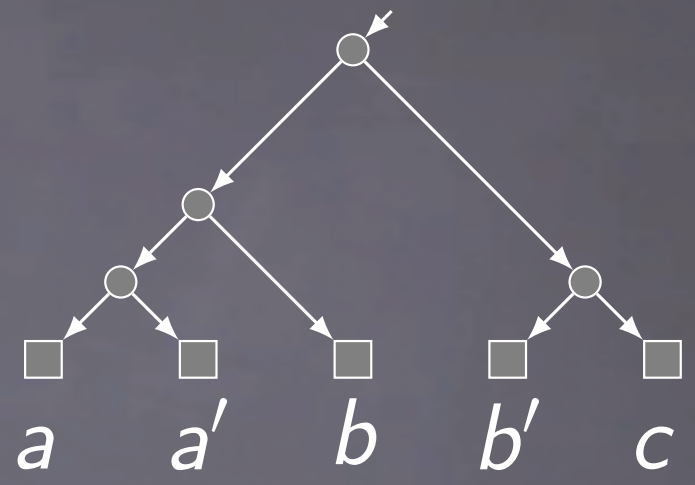
Embedding Rules

gene tree G, species tree S

- mapping $\rho : V(G) \to V(S)$
- ℓ is leaf in $G \rightsquigarrow \rho(\ell)$ "corresponds" to ℓ ($a \to A$, $a' \to A$, etc.)
- $u \in V(G)$ is called duplication if $\rho(u) = \rho(c)$ for any child c of u in G
- all non-leaves of G that not duplications are called speciations
- each edge uv of G incurs a loss-cost equal to the number of edges in the $\rho(u)$-$\rho(v)$-path in S minus 1 if v is a speciation or 0 if v is a duplication

Reconciliation



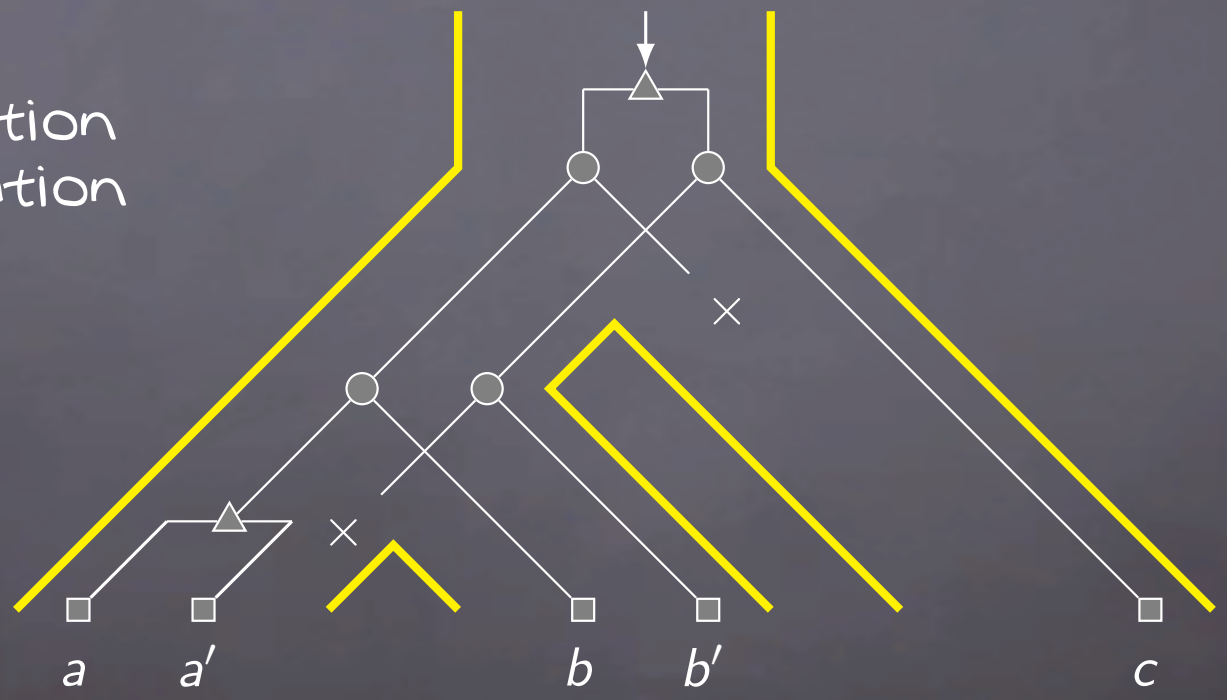


DL-model

○ = speciation

△ = duplication

× = loss



Reconciliation

Goal: embed gene tree into species tree
(extant genes must map to their species)

Max. Likelihood

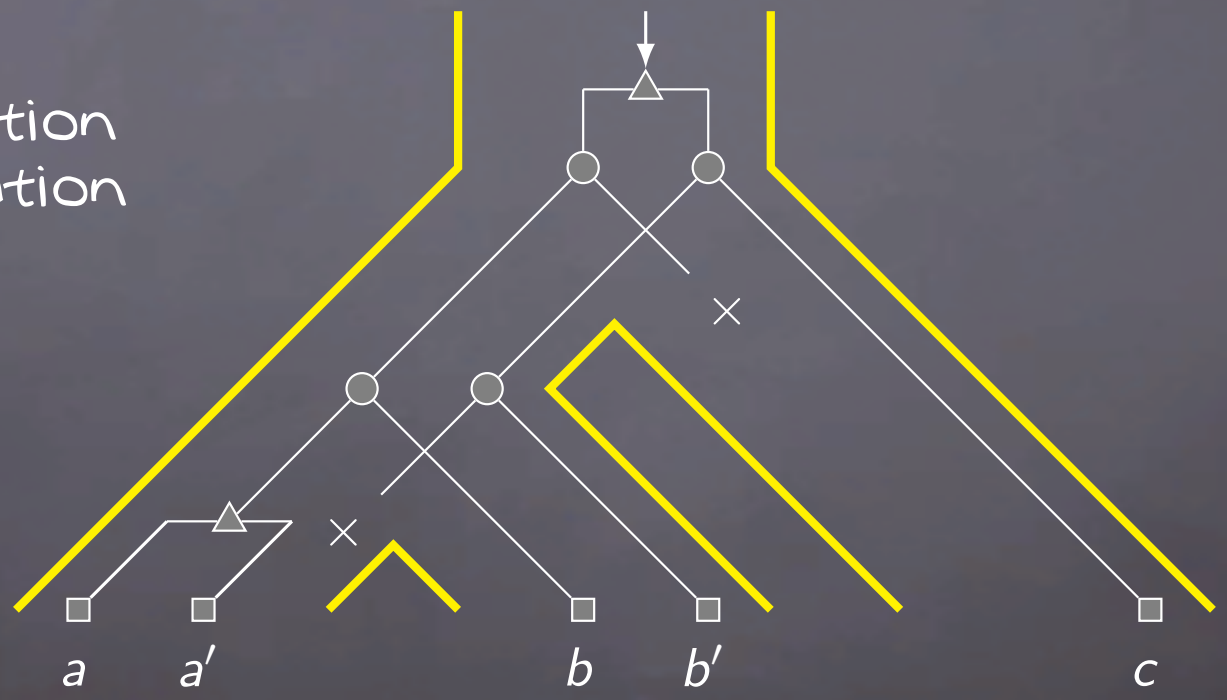
find most probable embedding
(computationally expensive)

Parsimony

find embedding minimizing
#events (possibly weighted)

DL-model

- ○ = speciation
- △ = duplication
- × = loss



Reconciliation

Parsimonious Reconciliation

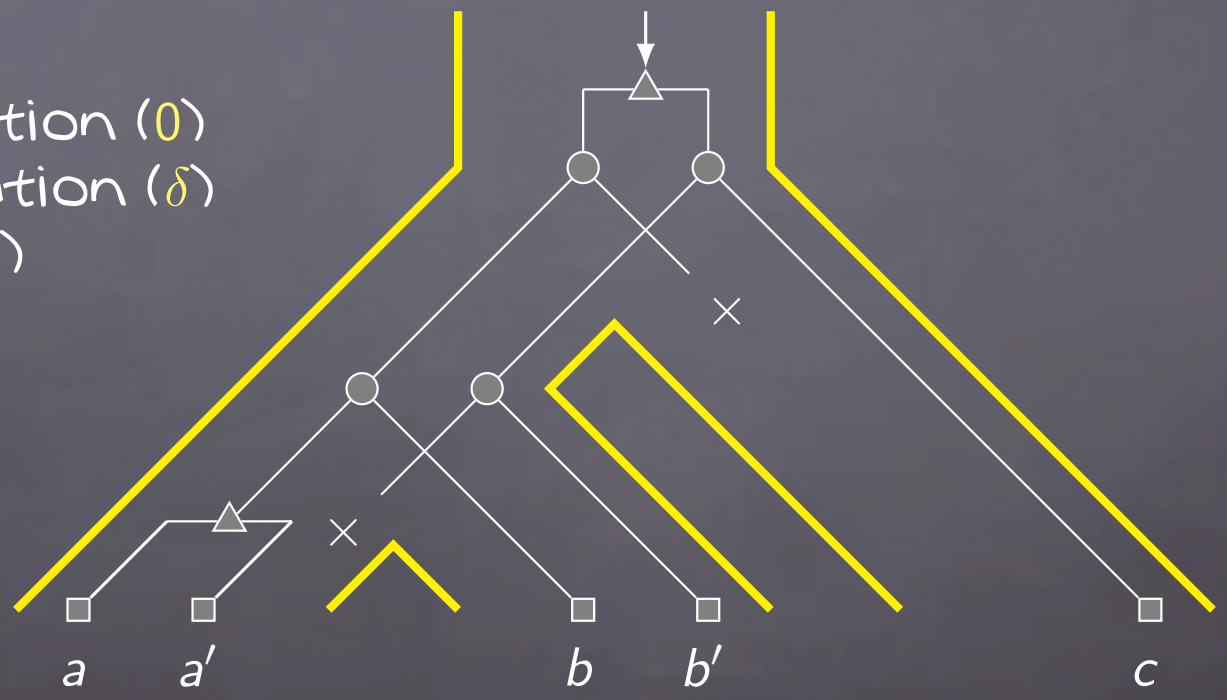
Input: species tree S, gene tree G, $\delta, \lambda \in \mathbb{N}$

Task: embed G in S, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$

DL-model

- ○ = speciation (0)
- △ = duplication (δ)
- × = loss (λ)



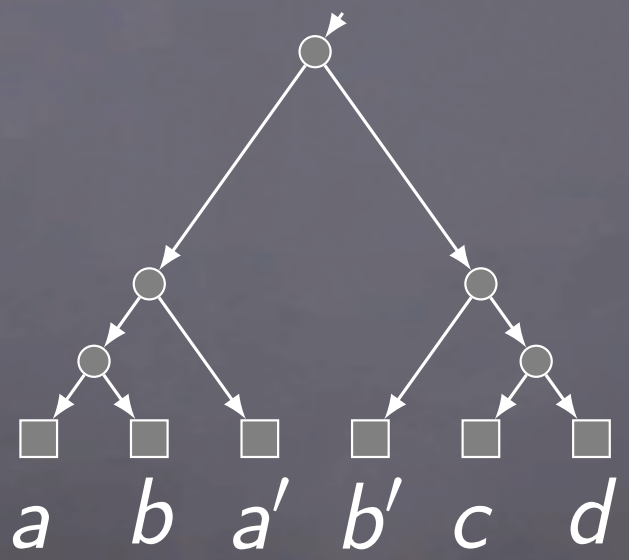
Reconciliation

Parsimonious Reconciliation

Input: species tree S, gene tree G, $\delta, \lambda, \tau \in \mathbb{N}$

Task: embed G in S, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$



Reconciliation

Parsimonious Reconciliation

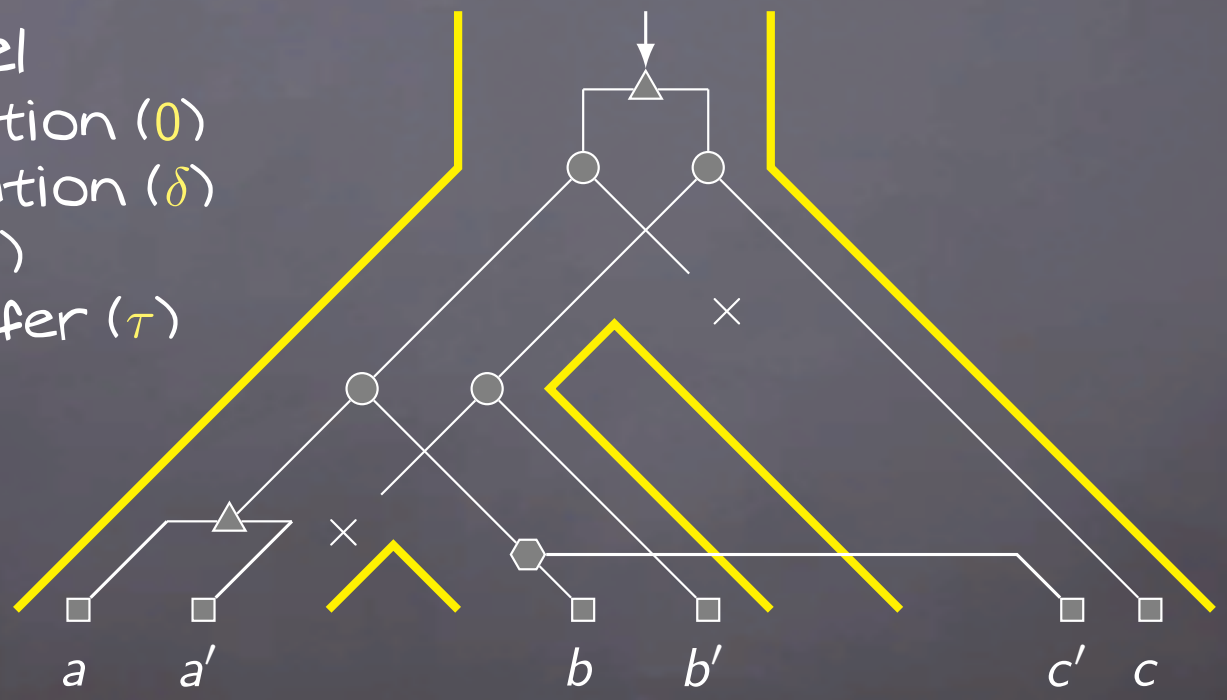
Input: species tree S, gene tree G, $\delta, \lambda, \tau \in \mathbb{N}$

Task: embed G in S, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$

DTL-model

- ○ = speciation (0)
- △ = duplication (δ)
- × = loss (λ)
- ⬡ = transfer (τ)



Reconciliation

Parsimonious Reconciliation

Input: species tree S, gene tree G, $\delta, \lambda, \tau \in \mathbb{N}$

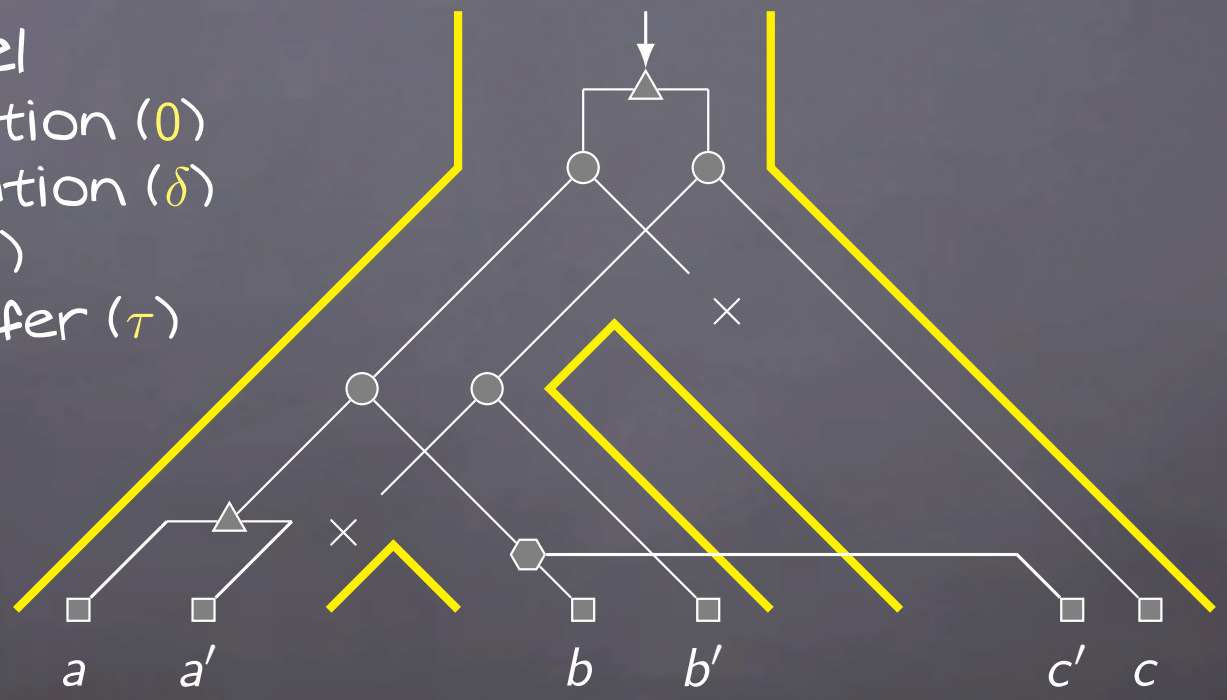
Task: embed G in S, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$

T events only between co-existing species $\rightsquigarrow$ time constraints $\rightsquigarrow$ NP-hard

DTL-model

- ○ = speciation (0)
- △ = duplication (δ)
- × = loss (λ)
- ⬡ = transfer (τ)



Reconciliation

Parsimonious Reconciliation

Input: species tree S, gene tree G, $\delta, \lambda, \tau \in \mathbb{N}$

Task: embed G in S, minimizing the weighted sum of events

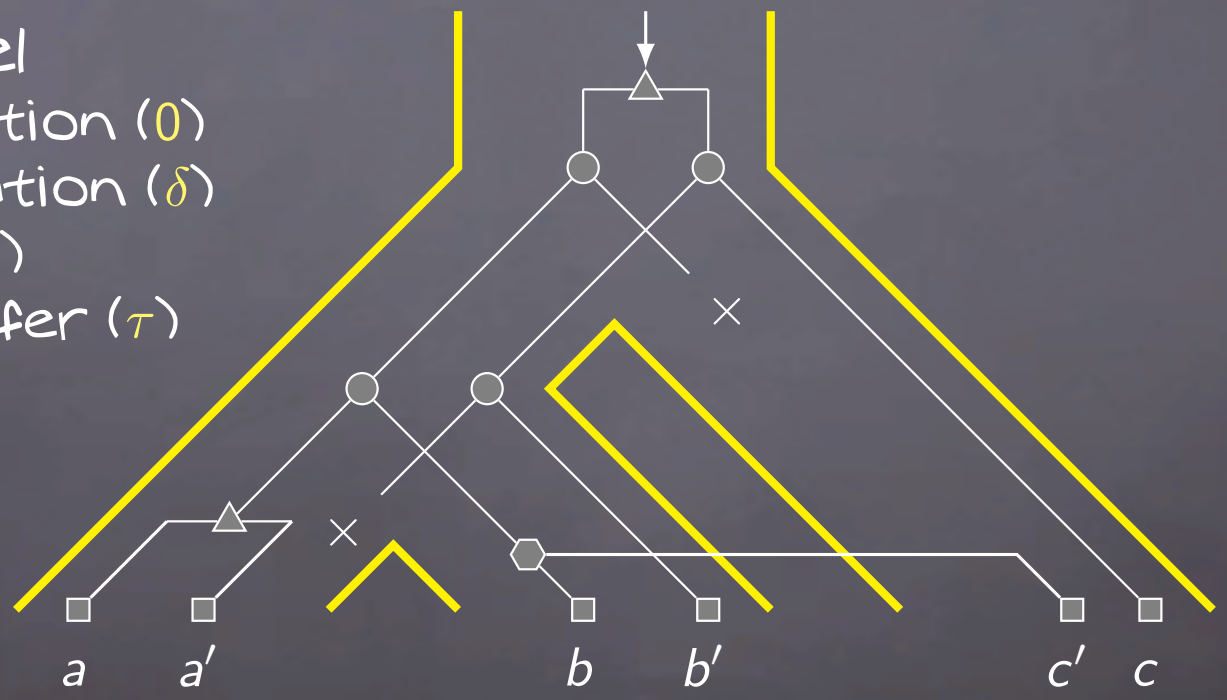
Result: LCA-assignment solves this optimally in $O(|S|+|G|)$

T events only between co-existing species $\rightsquigarrow$ time constraints $\rightsquigarrow$ NP-hard

Idea: take dated species tree $\rightsquigarrow$ $O(|S|^2|G|)$ time [Doyon et al.'10]

DTL-model

- ○ = speciation (0)
- △ = duplication (δ)
- × = loss (λ)
- ⬡ = transfer (τ)



Comparing Phylogenetic Trees

Distance Measures

- Nearest Neighbor Interchange
- Subtree Prune & Regraft
- Tree Bisection & Reconnection

Comparing Phylogenetic Trees

Distance Measures

- Nearest Neighbor Interchange
- Subtree Prune & Regraft
- Tree Bisection & Reconnection
- now: via agreement-forests

Comparing Phylogenetic Trees

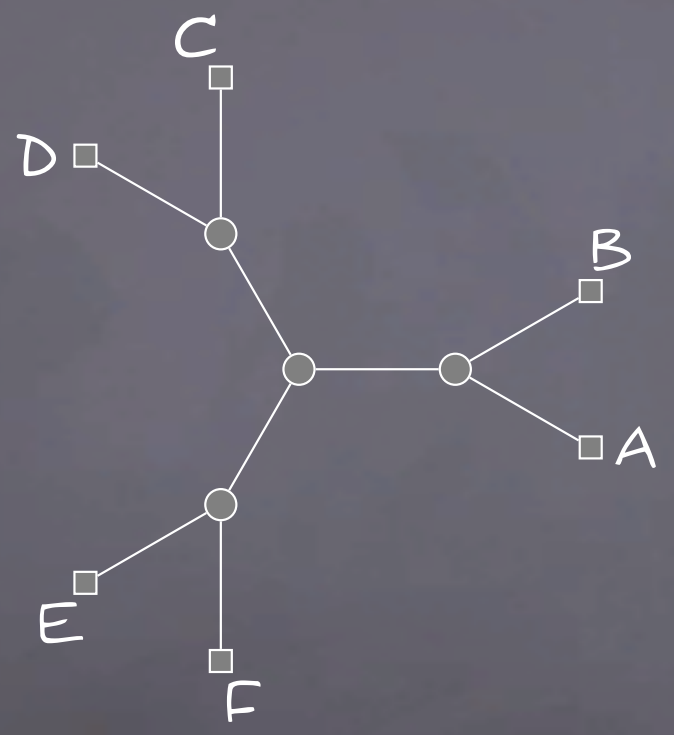
Distance Measures

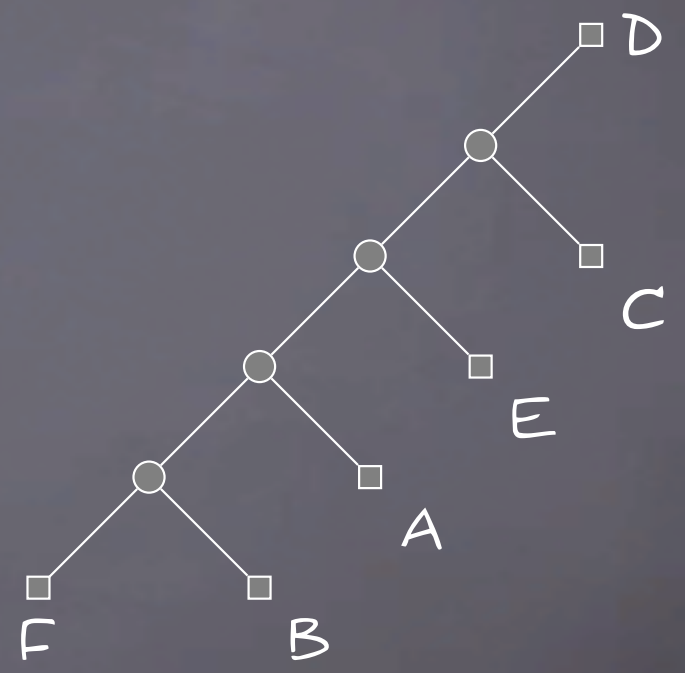
- Nearest Neighbor Interchange
- Subtree Prune & Regraft
- Tree Bisection & Reconnection
- now: via agreement-forests
- Robinson-Foulds distance
- quartet/triplet distance

Agreement Forests

Definition

A forest F is called **agreement forest** of trees T_1 and T_2 if F can be obtained from T_1 and T_2 by removing edges.

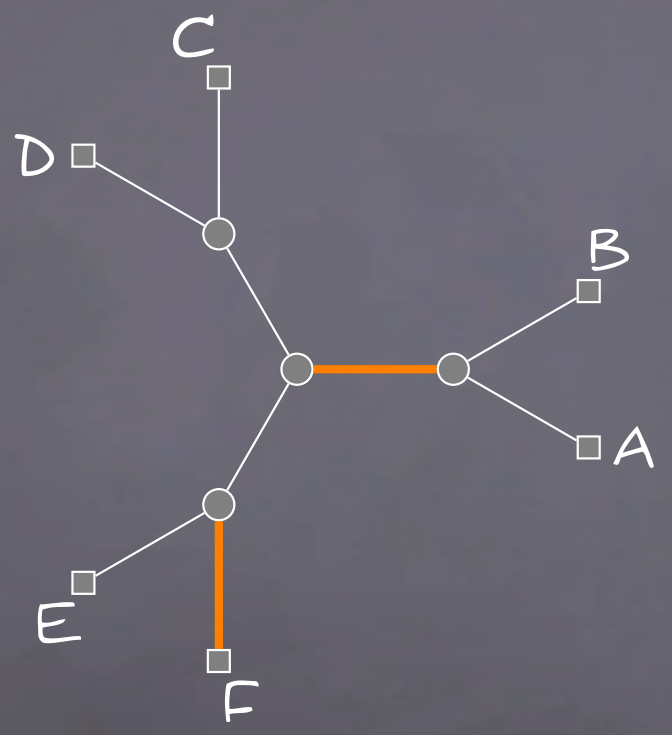


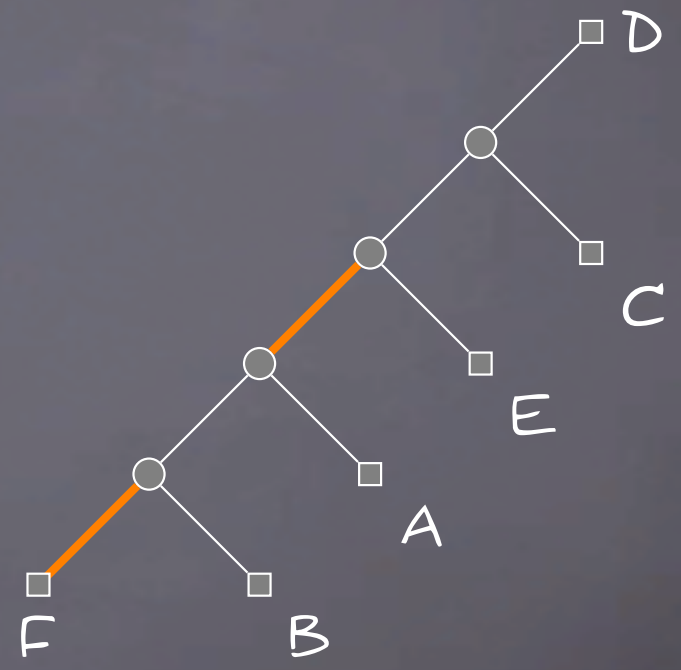


Agreement Forests

Definition

A forest F is called agreement forest of trees T_1 and T_2 if F can be obtained from T_1 and T_2 by removing edges.

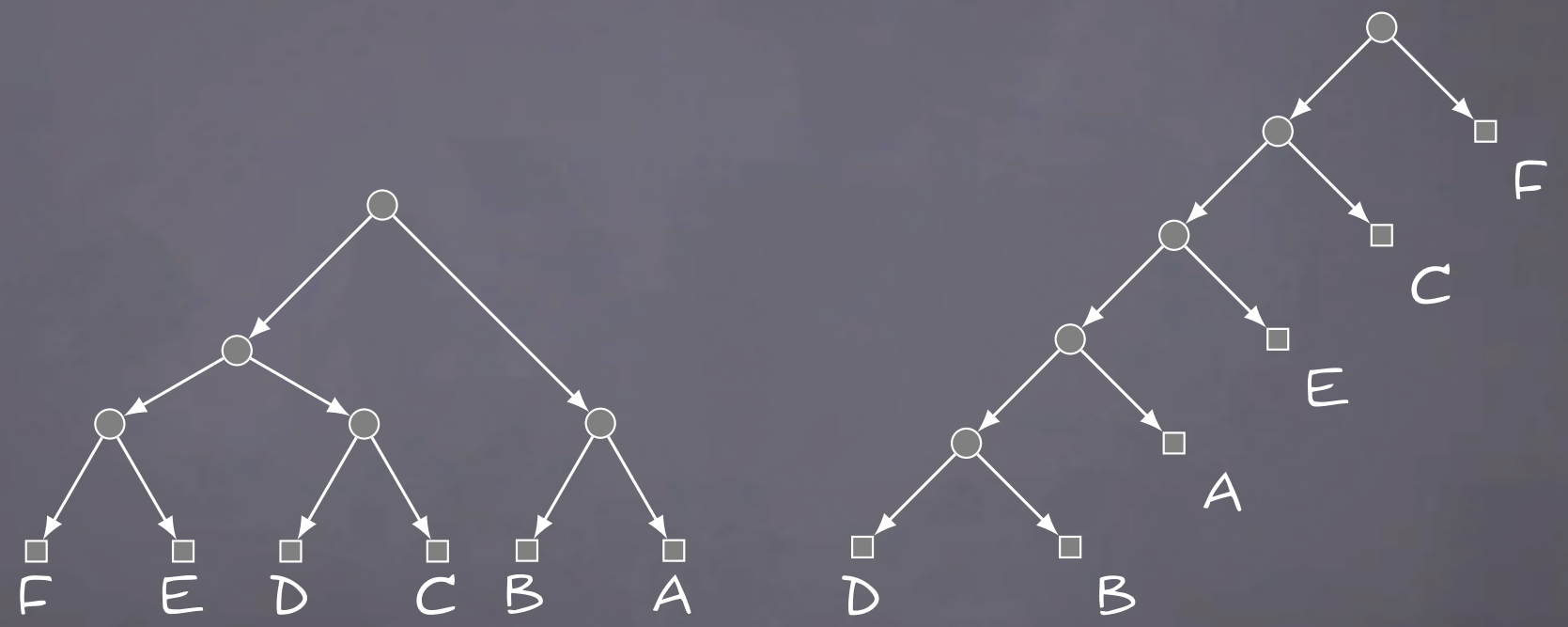




Agreement Forests

Definition

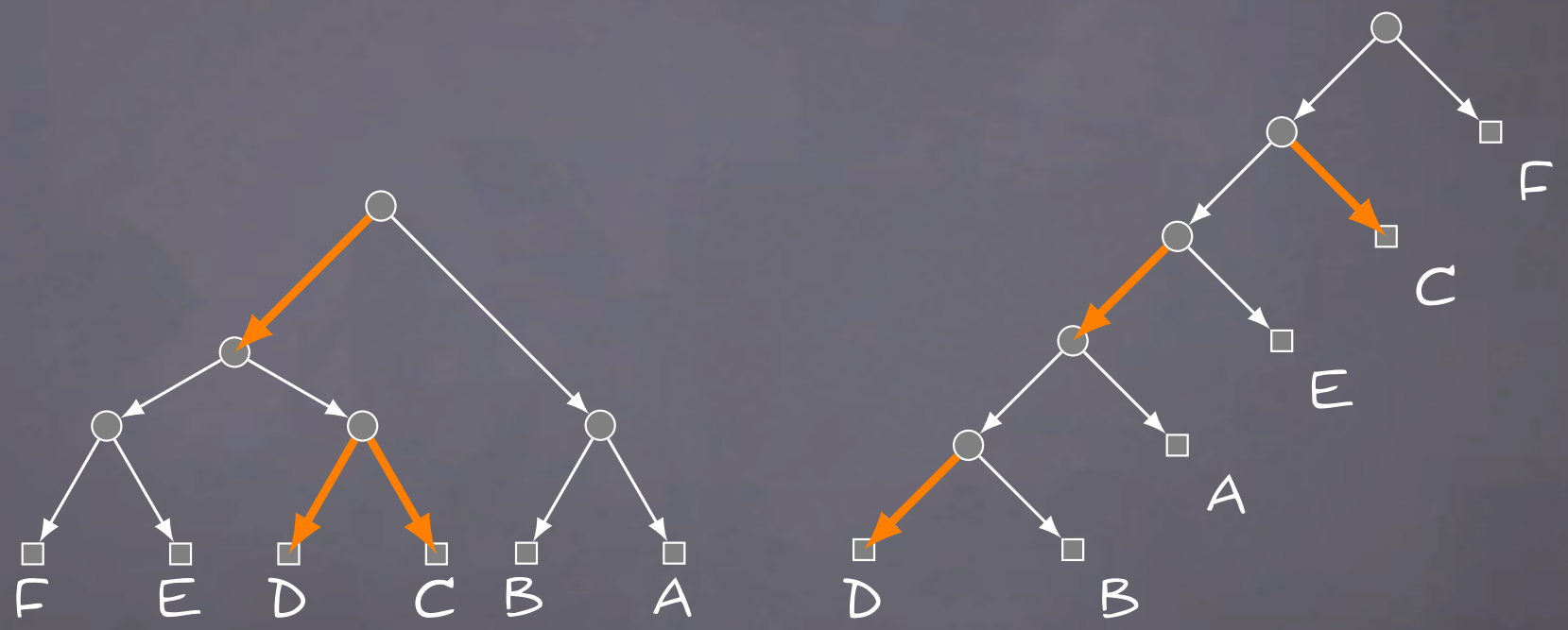
A forest F is called **agreement forest** of trees T_1 and T_2 if F can be obtained from T_1 and T_2 by removing edges.



Agreement Forests

Definition

A forest F is called **agreement forest** of trees T_1 and T_2 if F can be obtained from T_1 and T_2 by removing edges.



Agreement Forests

Definition

A forest F is called **agreement forest** of trees T_1 and T_2 if F can be obtained from T_1 and T_2 by removing edges.

Theorem [Allen & Steel, '01]

TBR-distance(T_1, T_2) = #trees in smallest agreement forest - 1
NP-hard to compute

Theorem [Bordewich & Semple, '04]

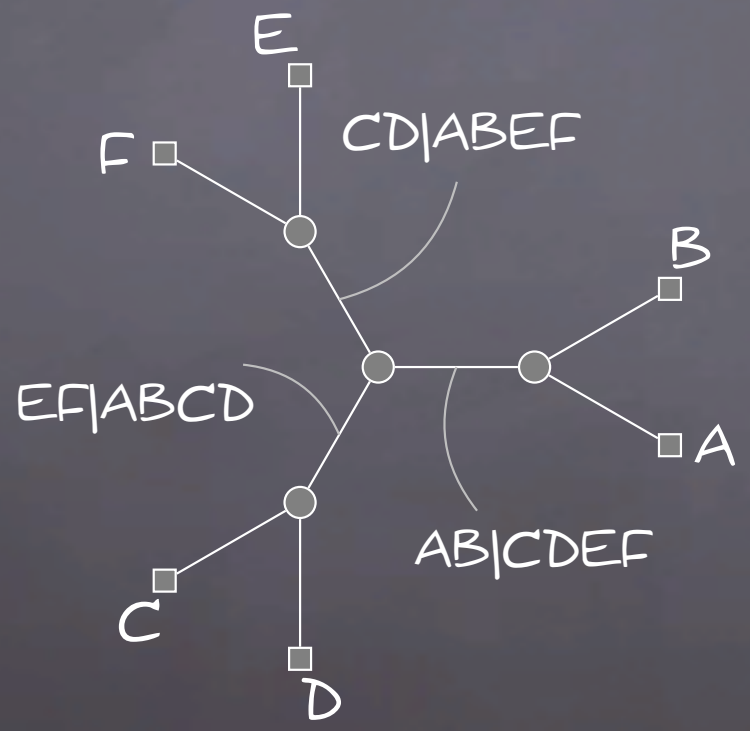
rSPR-distance(T_1, T_2) = #trees in smallest rooted agreement forest - 1
NP-hard to compute

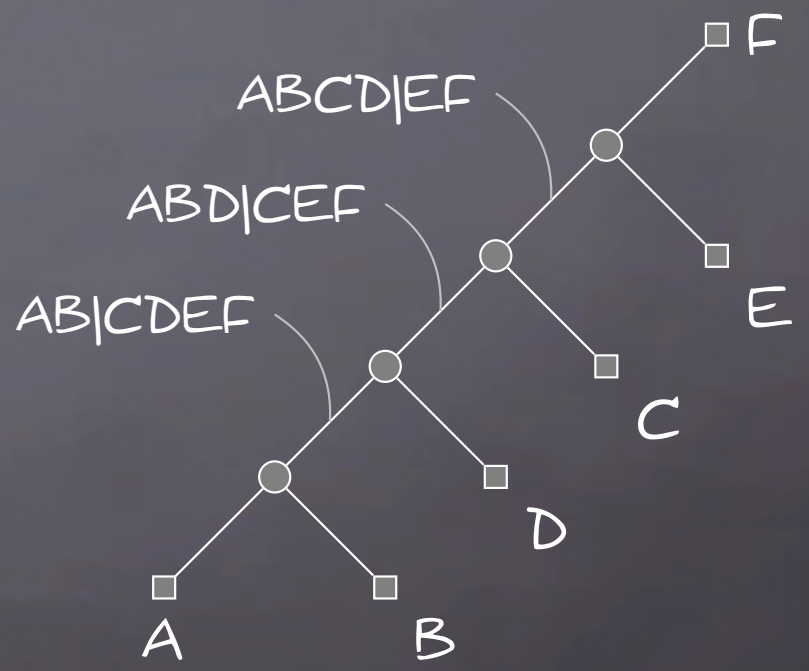
Robinson-Foulds Distance

Definition

$RF(T_1, T_2)$ = #splits/clusters occuring in exactly one of T_1 and T_2
= edge-contraction distance a common tree

Note: observe relation to NNI: $RF(T_1, T_2) \leq 2\ NNI(T_1, T_2)$



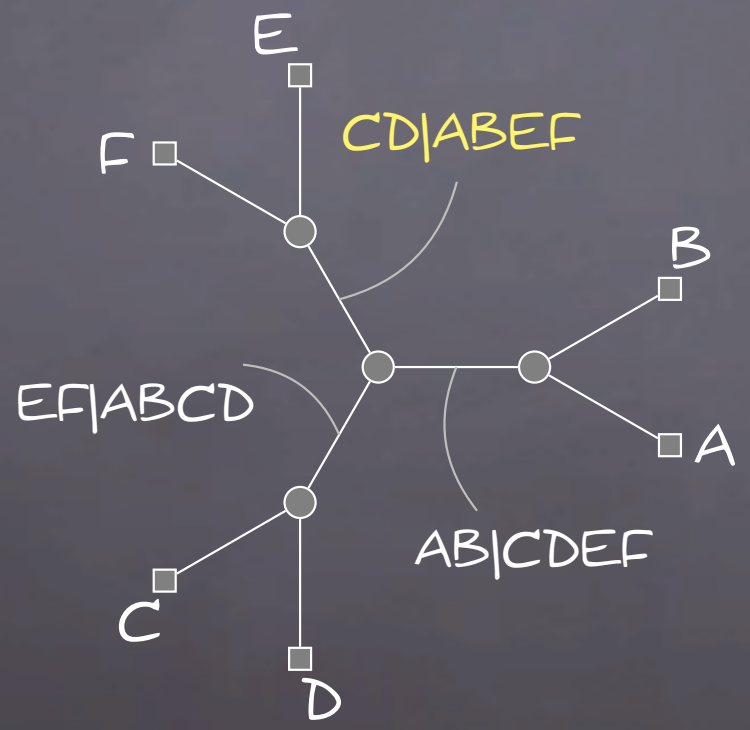


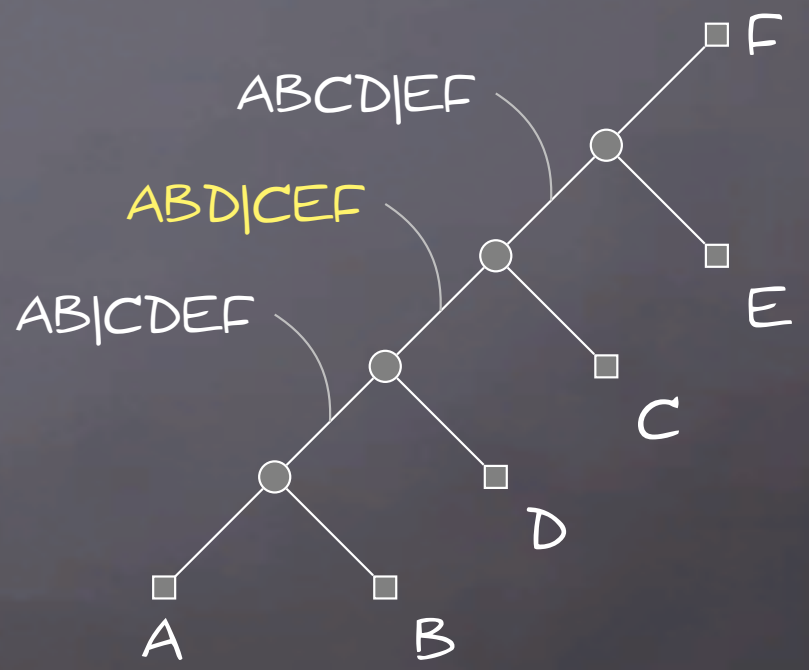
Robinson-Foulds Distance

Definition

$RF(T_1, T_2)$ = #splits/clusters occuring in exactly one of T_1 and T_2
= edge-contraction distance a common tree

Note: observe relation to NNI: $RF(T_1, T_2) \leq 2\ NNI(T_1, T_2)$



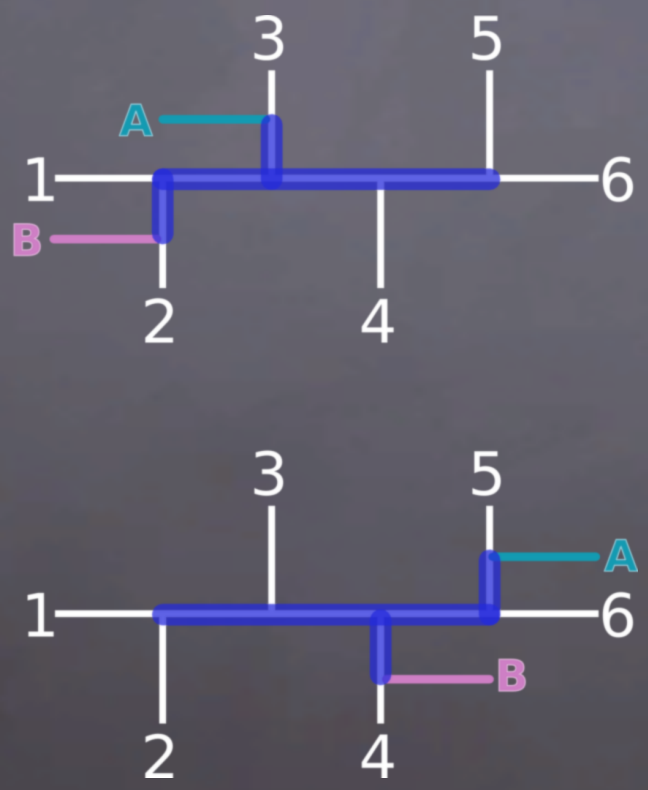


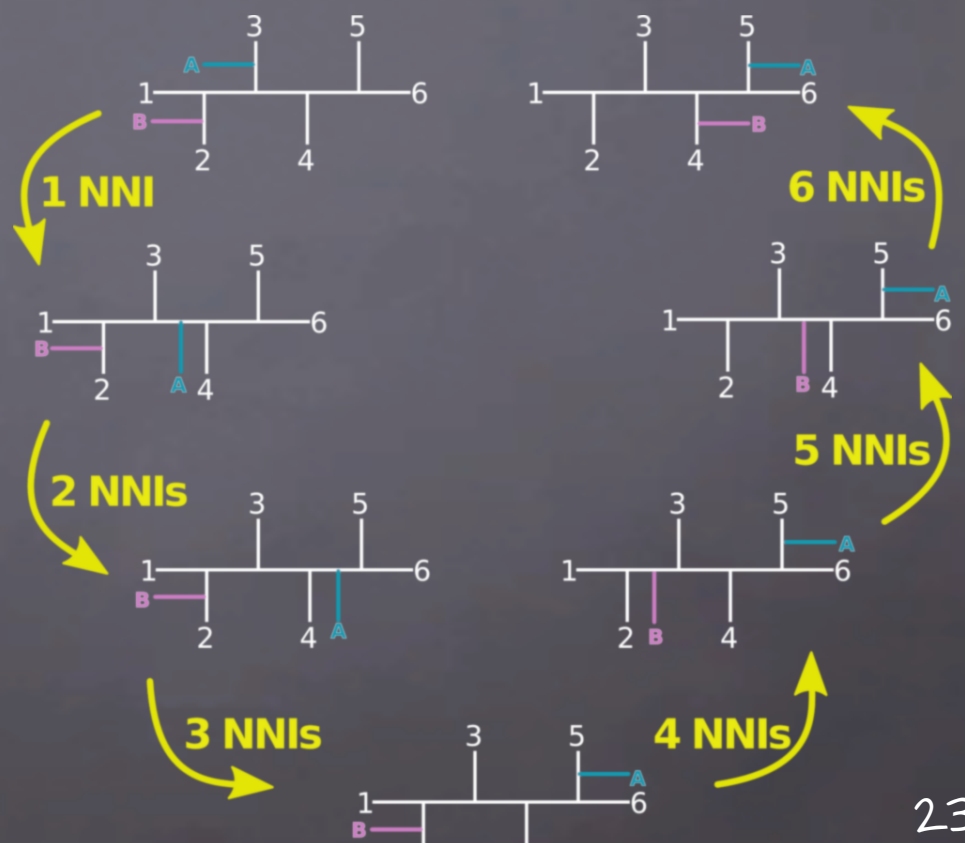
Robinson-Foulds Distance

Definition

$RF(T_1, T_2)$ = #splits/clusters occuring in exactly one of T_1 and T_2
= edge-contraction distance a common tree

Note: observe relation to NNI: $RF(T_1, T_2) \leq 2\ NNI(T_1, T_2)$





Robinson-Foulds Distance

Definition

$RF(T_1, T_2)$ = #splits/clusters occuring in exactly one of T_1 and T_2
= edge-contraction distance a common tree

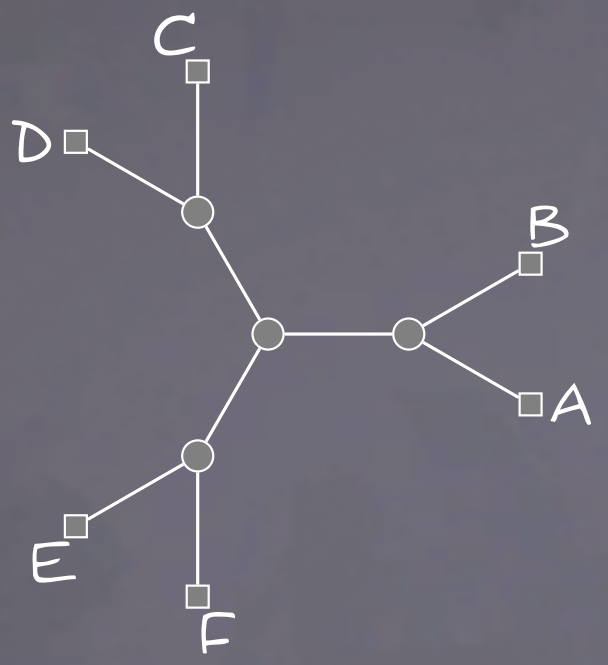
Note: observe relation to NNI: $RF(T_1, T_2) \leq 2\ NNI(T_1, T_2)$
Note: splits correspond to clusters when rooted at last leaf

Day's Algorithm (common clusters in $O(n)$)

[Day'85]

1. relabel all leaves such that leaves continuous in T_1
2. each node in T_1 knows:
   L = smallest leaf in cluster & R = largest leaf in cluster
   $\rightsquigarrow$ note T_1's clusters in hash-set
3. each node in T_2 knows: L, R, and size N of its cluster
4. each node in T_2 checks $[L, R]$ in table only if $R - L = N - 1$
   (lookup in T_1's cluster-set in $O(1)$ (average) time)

Robinson-Foulds Distance

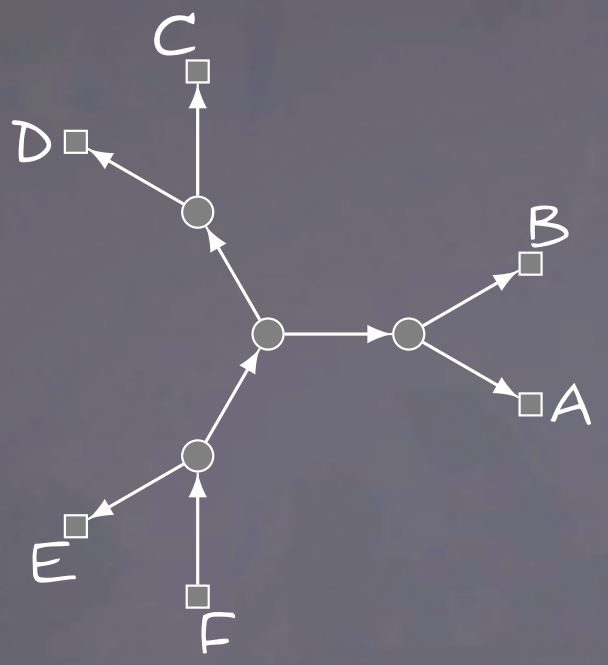


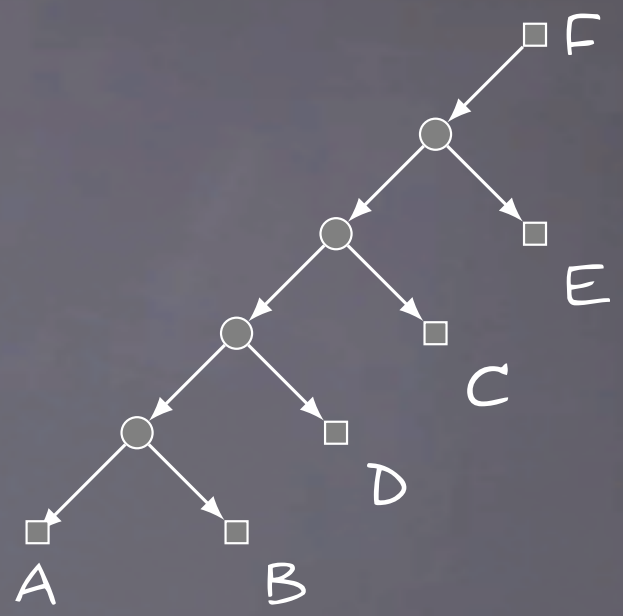
Day's Algorithm (common clusters in $O(n)$)

[Day'85]

1. relabel all leaves such that leaves continuous in T_1
2. each node in T_1 knows:
   L = smallest leaf in cluster & R = largest leaf in cluster
   $\rightsquigarrow$ note T_1's clusters in hash-set
3. each node in T_2 knows: L, R, and size N of its cluster
4. each node in T_2 checks $[L, R]$ in table only if $R - L = N - 1$
   (lookup in T_1's cluster-set in $O(1)$ (average) time)

Robinson-Foulds Distance



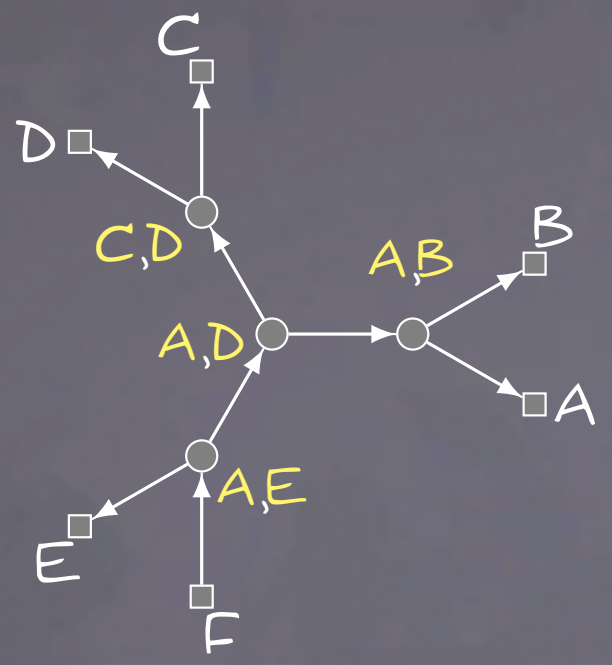


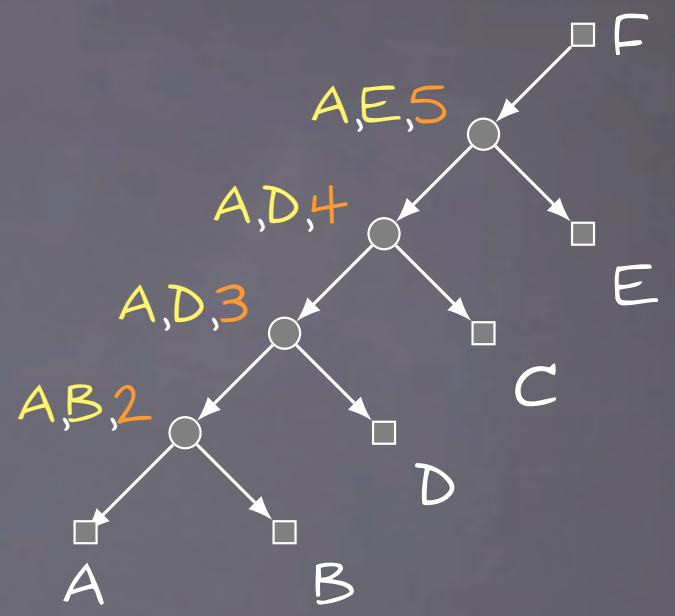
Day's Algorithm (common clusters in $O(n)$)

[Day'85]

1. relabel all leaves such that leaves continuous in T_1
2. each node in T_1 knows:
   L = smallest leaf in cluster & R = largest leaf in cluster
   $\rightsquigarrow$ note T_1's clusters in hash-set
3. each node in T_2 knows: L, R, and size N of its cluster
4. each node in T_2 checks $[L, R]$ in table only if $R - L = N - 1$
   (lookup in T_1's cluster-set in $O(1)$ (average) time)

Robinson-Foulds Distance



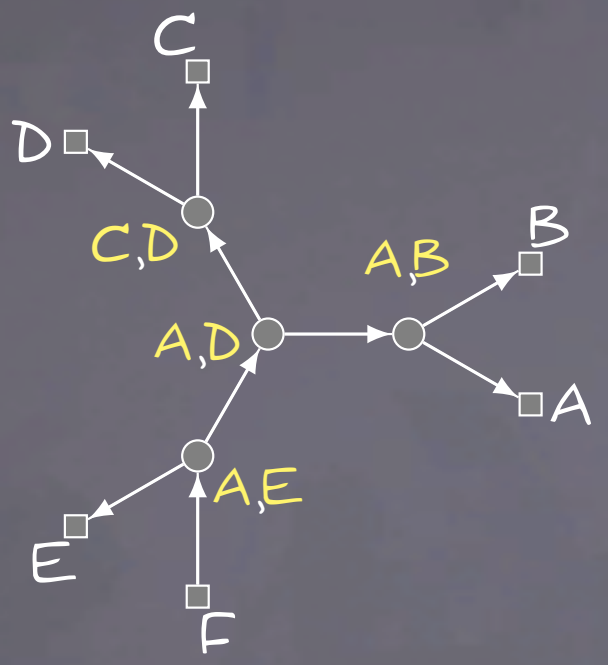


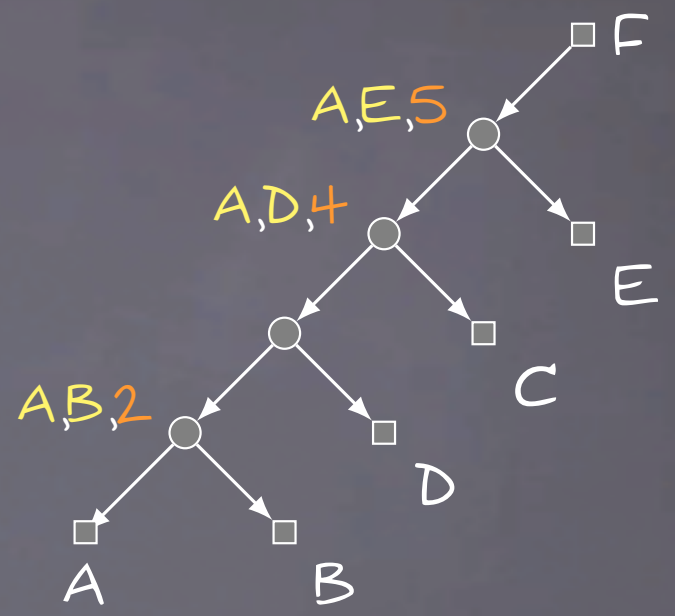
Day's Algorithm (common clusters in $O(n)$)

[Day'85]

1. relabel all leaves such that leaves continuous in T_1
2. each node in T_1 knows:
   L = smallest leaf in cluster & R = largest leaf in cluster
   $\rightsquigarrow$ note T_1's clusters in hash-set
3. each node in T_2 knows: L, R, and size N of its cluster
4. each node in T_2 checks $[L, R]$ in table only if $R - L = N - 1$
   (lookup in T_1's cluster-set in $O(1)$ (average) time)

Robinson-Foulds Distance





Day's Algorithm (common clusters in $O(n)$)

[Day'85]

1. relabel all leaves such that leaves continuous in T_1
2. each node in T_1 knows:
   L = smallest leaf in cluster & R = largest leaf in cluster
   $\rightsquigarrow$ note T_1's clusters in hash-set
3. each node in T_2 knows: L, R, and size N of its cluster
4. each node in T_2 checks $[L, R]$ in table only if $R - L = N - 1$
   (lookup in T_1's cluster-set in $O(1)$ (average) time)

Quartet/Triplet Distance

Definition

$Q/T(T_1, T_2)$ = #quartets/triplets occur. in exactly one of T_1 and T_2

Quartet/Triplet Distance

Definition

$Q/T(T_1, T_2)$ = #quartets/triplets occur. in exactly one of T_1 and T_2

computing Q-distance (Binary trees) [Bryant et al.'00]

1. each edge uv has 4 sets (2 clusters for each of u & v)
2. quartet AB|CD "belongs" to edge e if e splits AB|CD and e touches AB-path $\rightsquigarrow$ each quartet is owned exactly once
3. $\forall\ uv \in T_1$ & $qr \in T_2$: intersect 4 sets of uv with split of qr in T_2
4. sizes of intersections can be precomputed bottom-up in $O(n^2)$ time

Quartet/Triplet Distance

Definition

$Q/T(T_1, T_2)$ = #quartets/triplets occur. in exactly one of T_1 and T_2

computing Q-distance (Binary trees) [Bryant et al.'00]

1. each edge uv has 4 sets (2 clusters for each of u & v)
2. quartet AB|CD "belongs" to edge e if e splits AB|CD and e touches AB-path $\rightsquigarrow$ each quartet is owned exactly once
3. $\forall\ uv \in T_1$ & $qr \in T_2$: intersect 4 sets of uv with split of qr in T_2
4. sizes of intersections can be precomputed bottom-up in $O(n^2)$ time

State of the Art

count conflict quartets/triplets $\rightsquigarrow$ $O(n \log n)$ time [Brodal et al.'13]
enumerate conflict quartets $\rightsquigarrow$ $O(n^2 + d)$ time [Bryant et al.'00]
enumerate conflict triplets $\rightsquigarrow$ $O(n + d)$ time [Weller'17]

Phylogenetic Networks

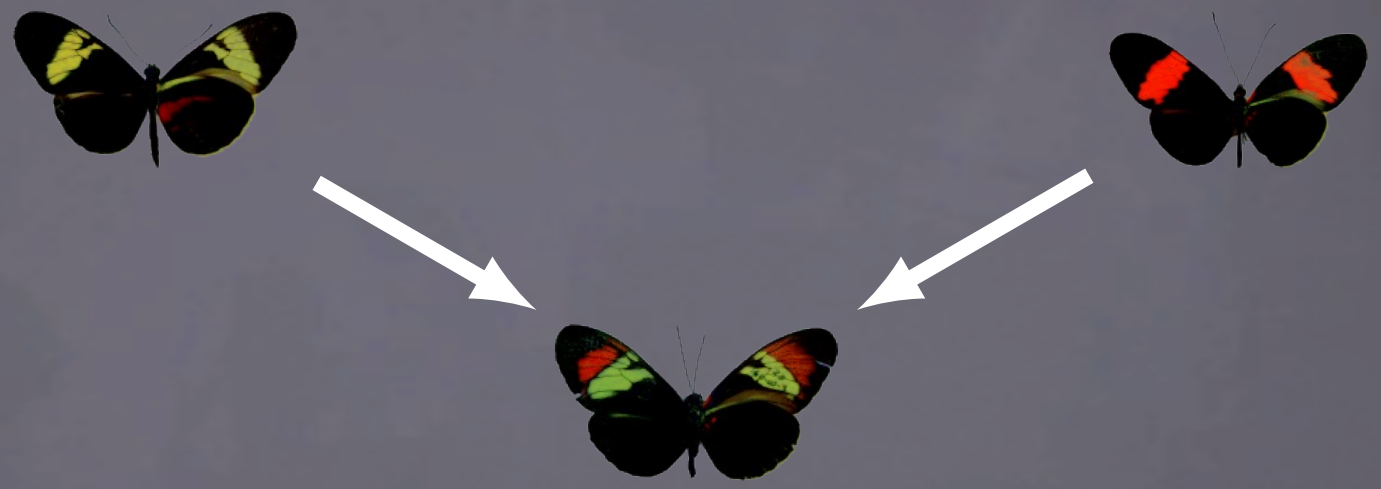
Observation

Trees cannot capture hybridization

Phylogenetic Networks

Observation

Trees cannot capture hybridization $\rightsquigarrow$ phylogenetic network



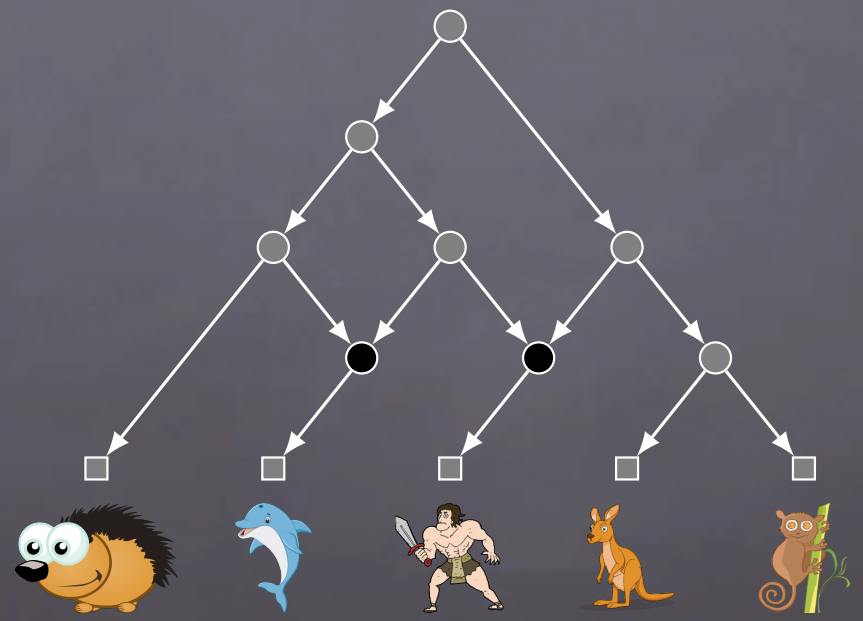
Phylogenetic Networks

Observation

Trees cannot capture hybridization $\rightsquigarrow$ phylogenetic network

Definition

evolutionary network N = rooted DAG, leaves labeled (taxa)
reticulations R = vertices of in-degree ≥ 2
binary = all inner vertices degree 3
block = maximal biconnected component
display T = subdivision of T is a subgraph



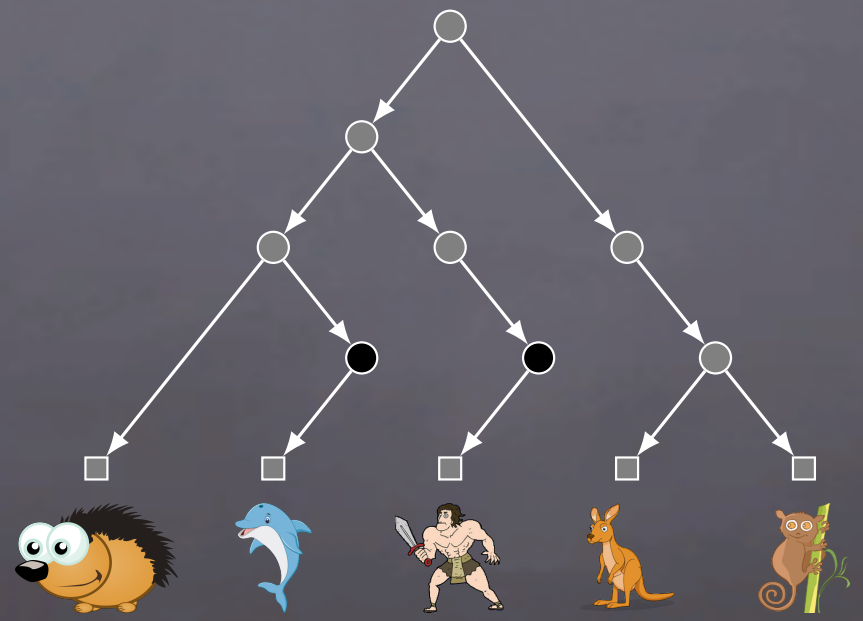
Phylogenetic Networks

Observation

Trees cannot capture hybridization $\rightsquigarrow$ phylogenetic network

Definition

evolutionary network N = rooted DAG, leaves labeled (taxa)
reticulations R = vertices of in-degree ≥ 2
binary = all inner vertices degree 3
block = maximal biconnected component
display T = subdivision of T is a subgraph

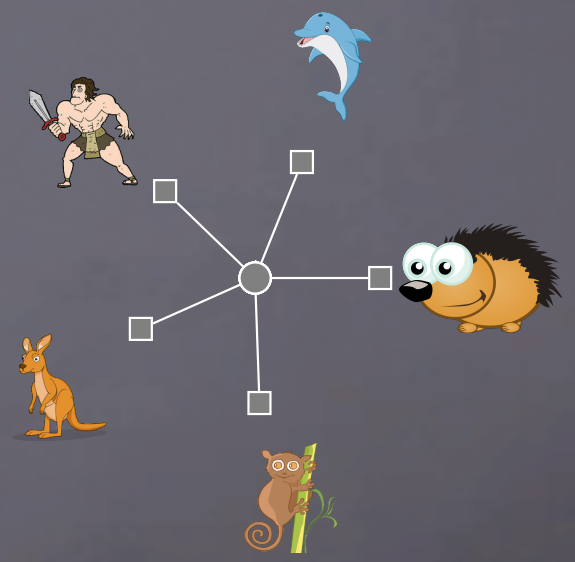


Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]



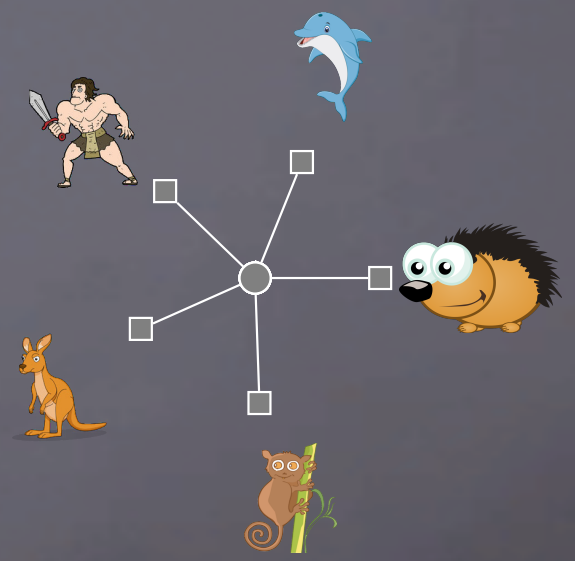
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]



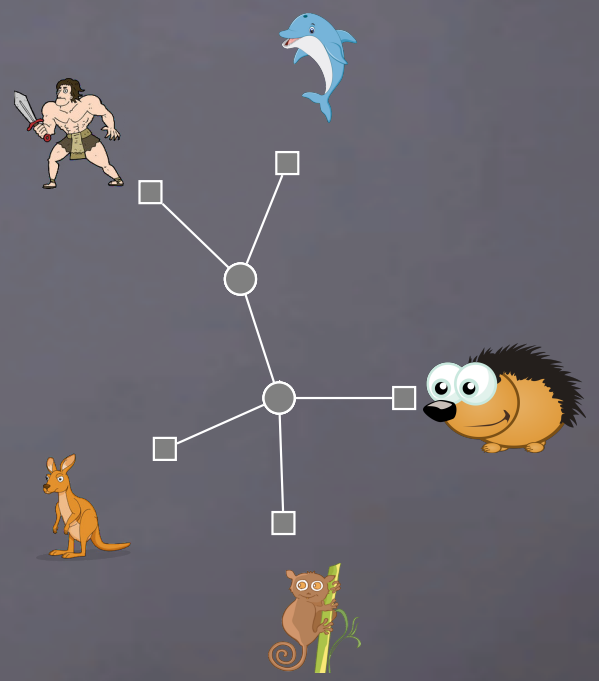
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]



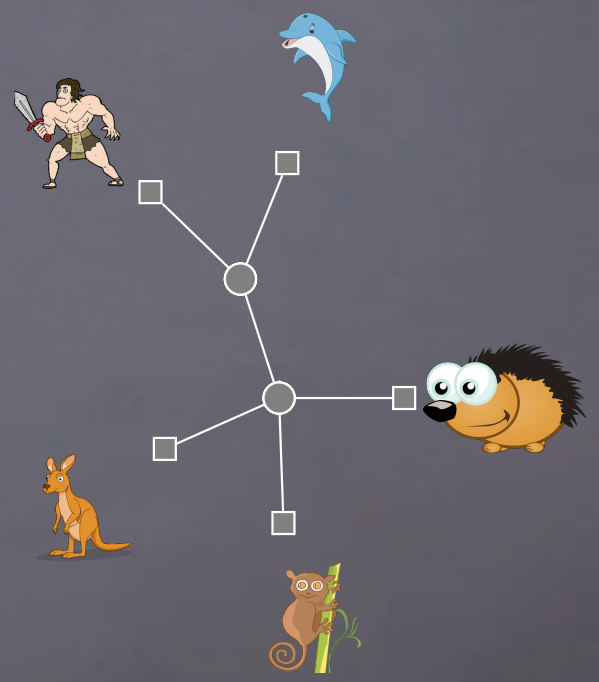
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]



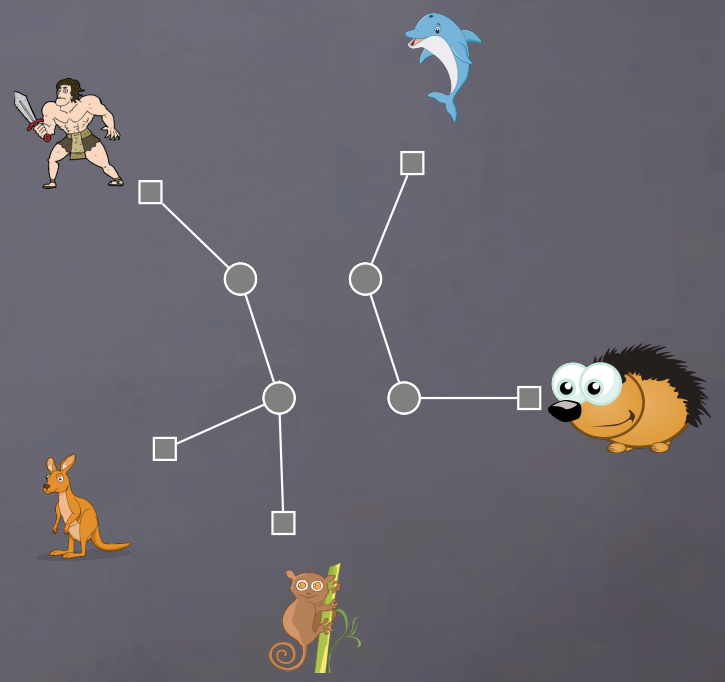
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]



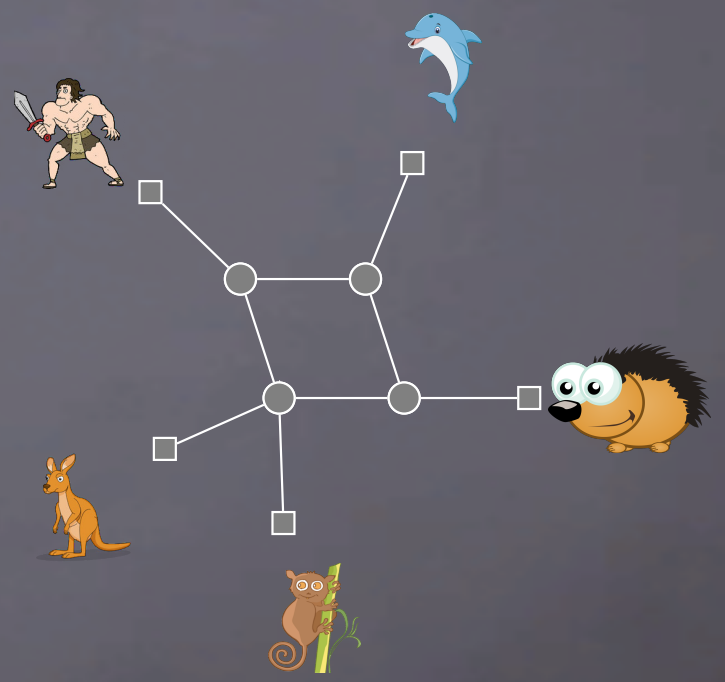
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]



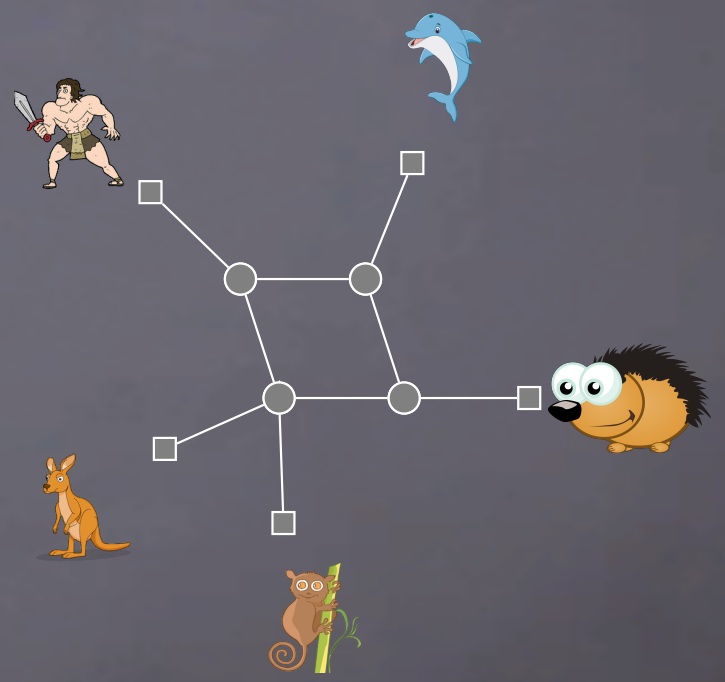
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]



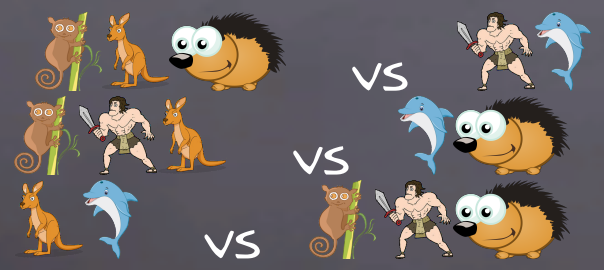
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

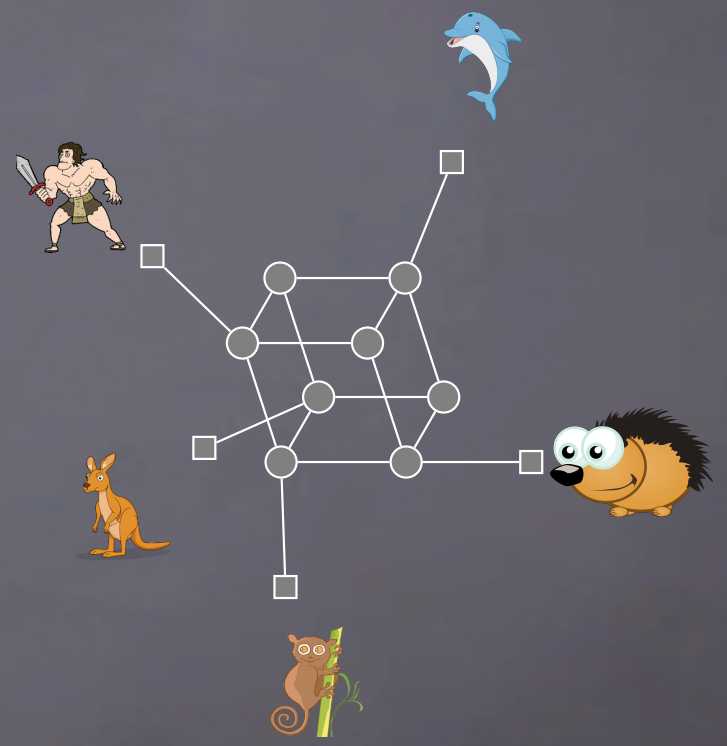
Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]





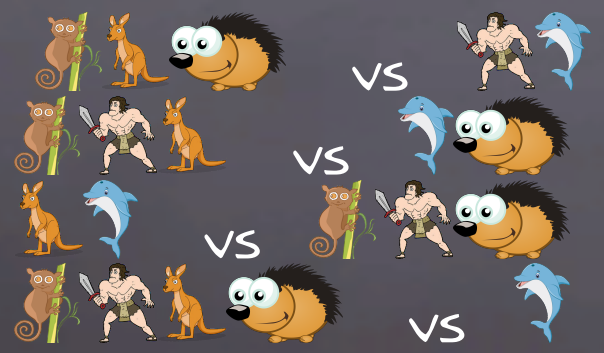
c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

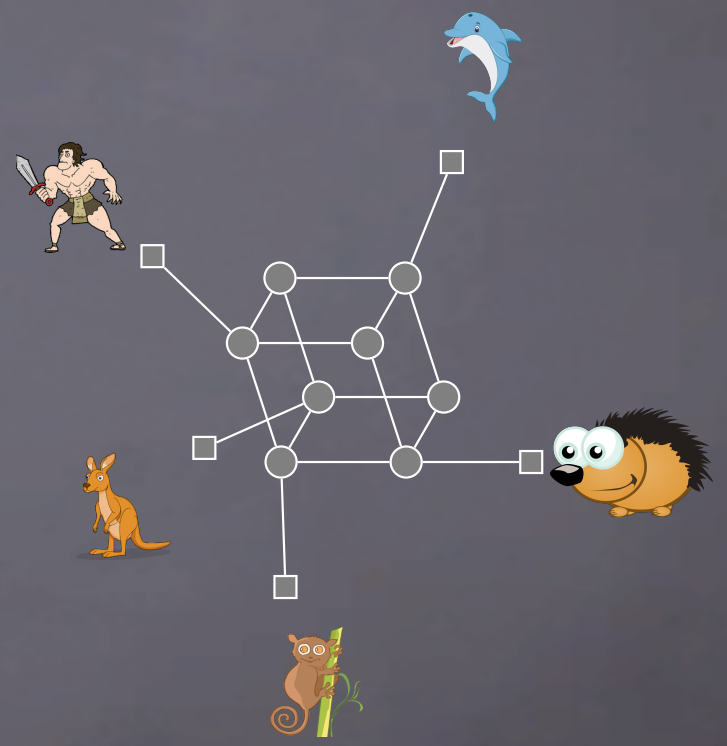
Split Networks

split = bipartition of set of taxa

splits A|B & X|Y incompatible if both A & B intersect both X & Y

Convex Hull Algorithm [Holland et al.,'04]





c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)

Consensus Split Networks

Strategy

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. Build consensus

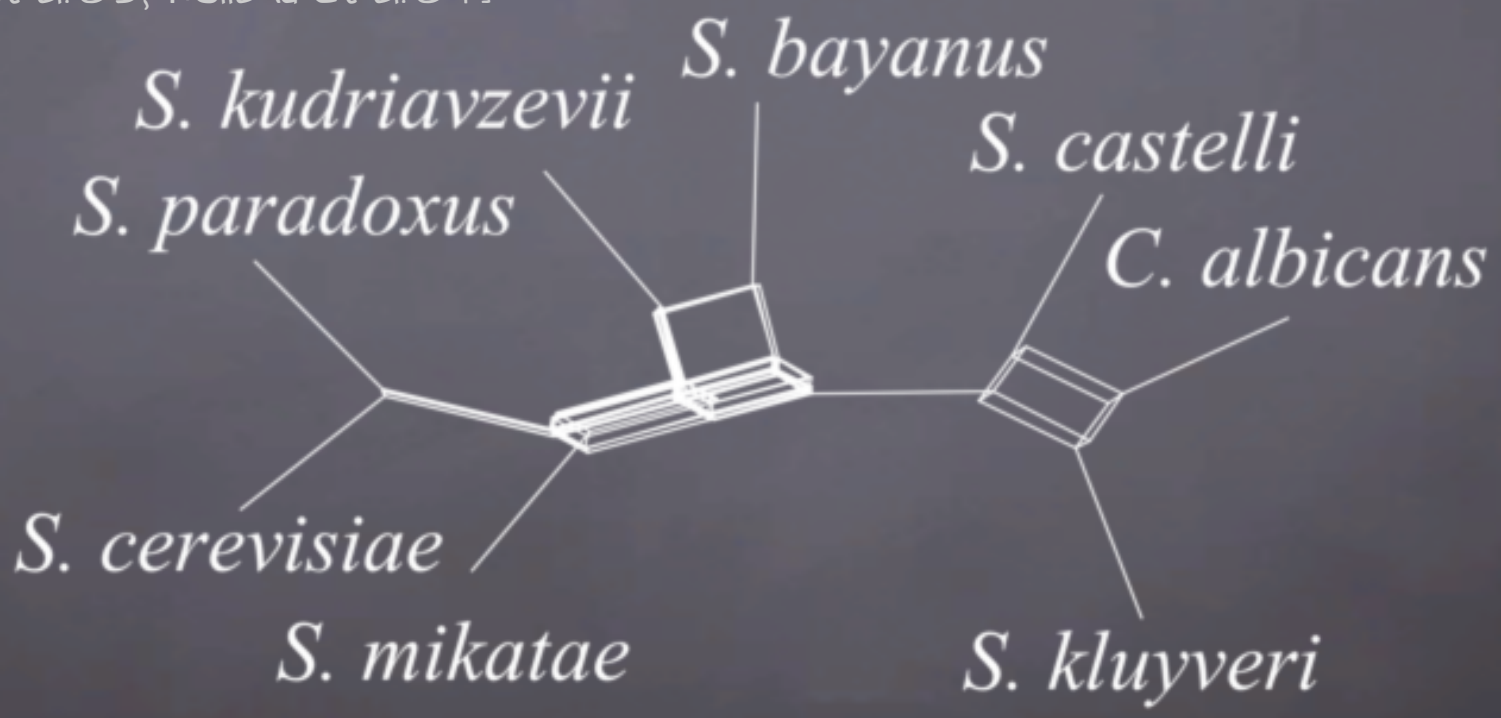
Consensus Split Networks

Strategy

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. Build consensus

Experimental Study – 106 gene trees (yeast)

[Rokas et al.'03, Holland et al.'04]



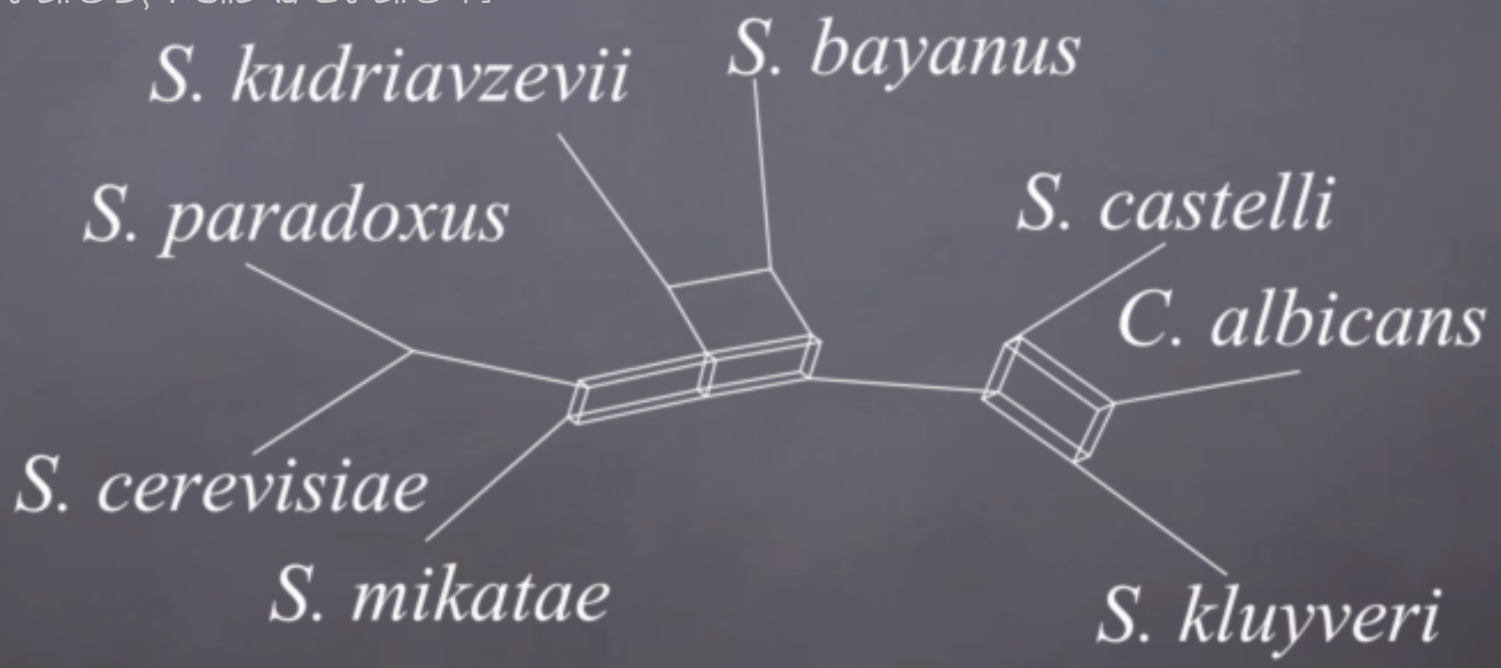
Consensus Split Networks

Strategy

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. Build consensus

Experimental Study – 106 gene trees (yeast)

[Rokas et al.'03, Holland et al.'04]



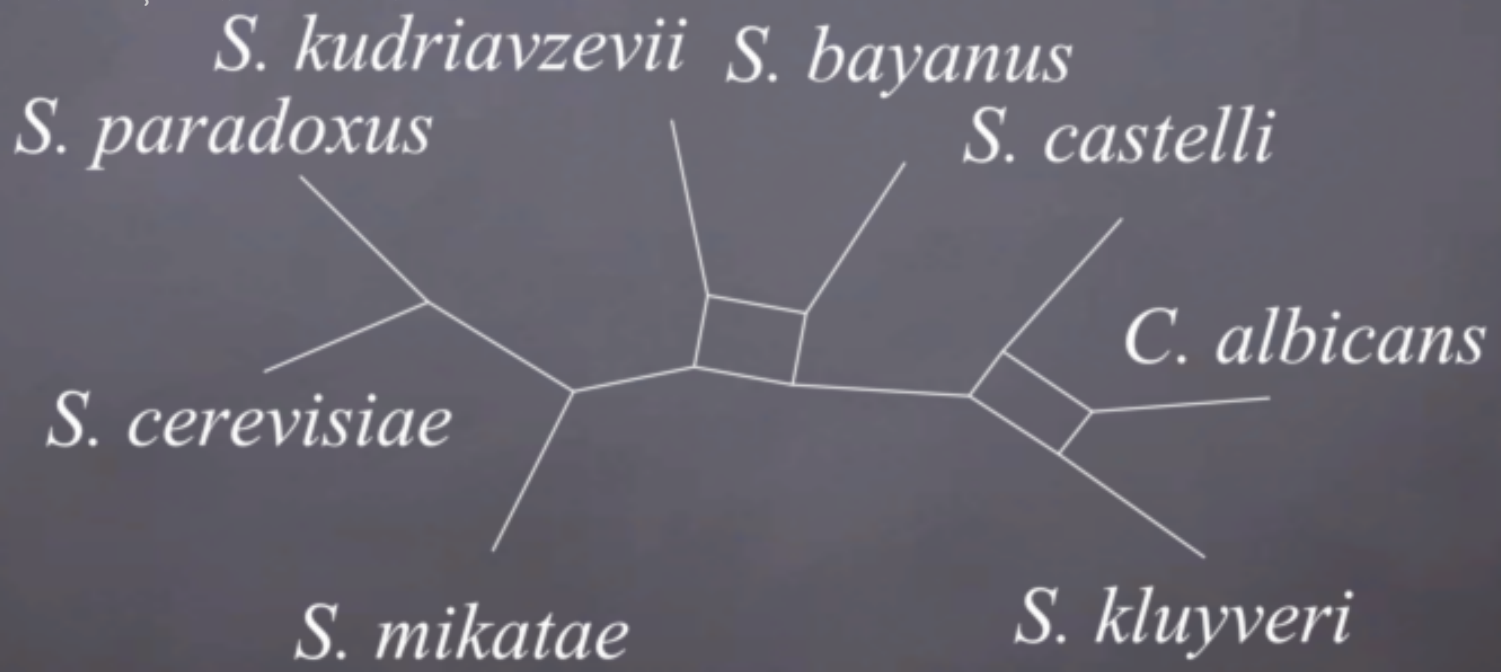
Consensus Split Networks

Strategy

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. Build consensus

Experimental Study – 106 gene trees (yeast)

[Rokas et al.'03, Holland et al.'04]



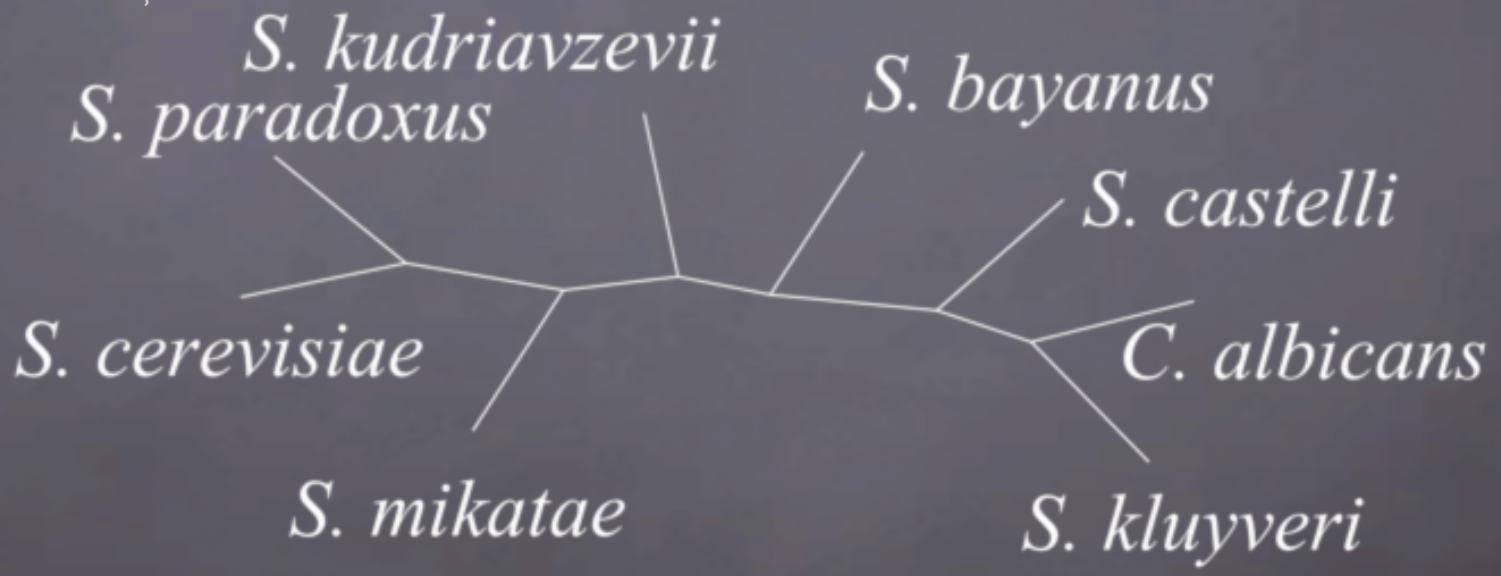
Consensus Split Networks

Strategy

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. Build consensus

Experimental Study – 106 gene trees (yeast)

[Rokas et al.'03, Holland et al.'04]



Rooted Network Reconstruction

Observation

rooted network: cluster of $u \subseteq$ cluster of $v \Leftrightarrow u \leq v$

$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Rooted Network Reconstruction

Observation

rooted network: cluster of u $\subseteq$ cluster of v $\Leftrightarrow$ u $\leq$ v

$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Example

{a,b,c,d},{c,d,e,f,g,h}, {c,d,e,f,g}, {e,f,g,h}, {c,d,e}, {e,f,g}, {a,b}, {c,d}, {f,g}

Rooted Network Reconstruction

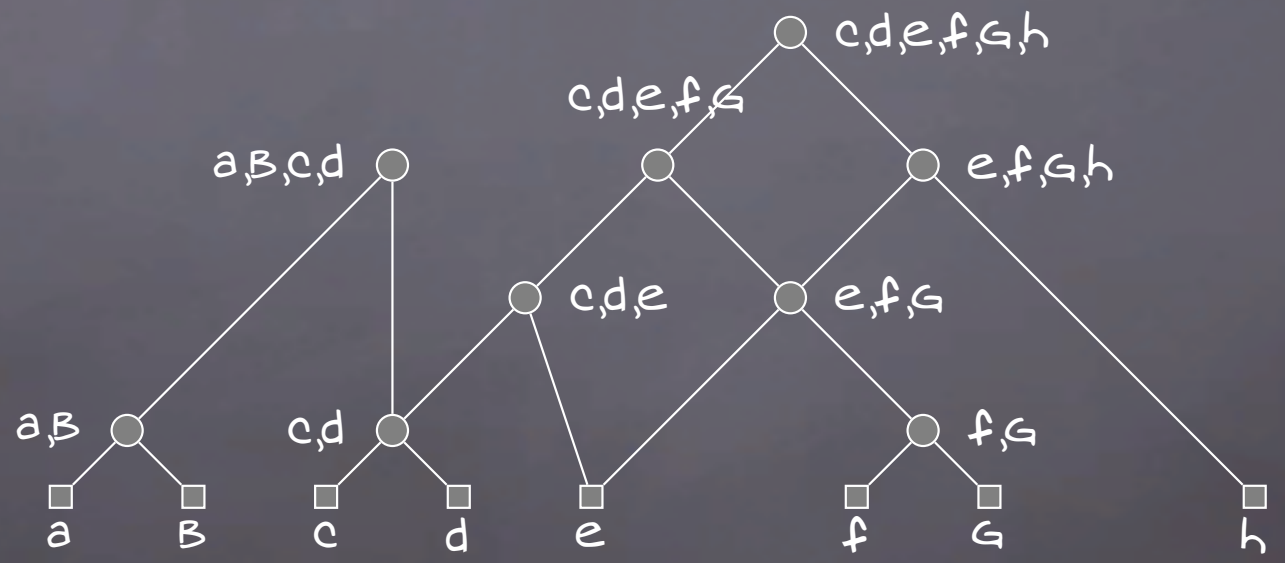
Observation

rooted network: cluster of $u \subseteq$ cluster of $v \Leftrightarrow u \leq v$

$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Example

{a,B,c,d},{c,d,e,f,G,h}, {c,d,e,f,G}, {e,f,G,h}, {c,d,e}, {e,f,G}, {a,B}, {c,d}, {f,G}



Rooted Network Reconstruction

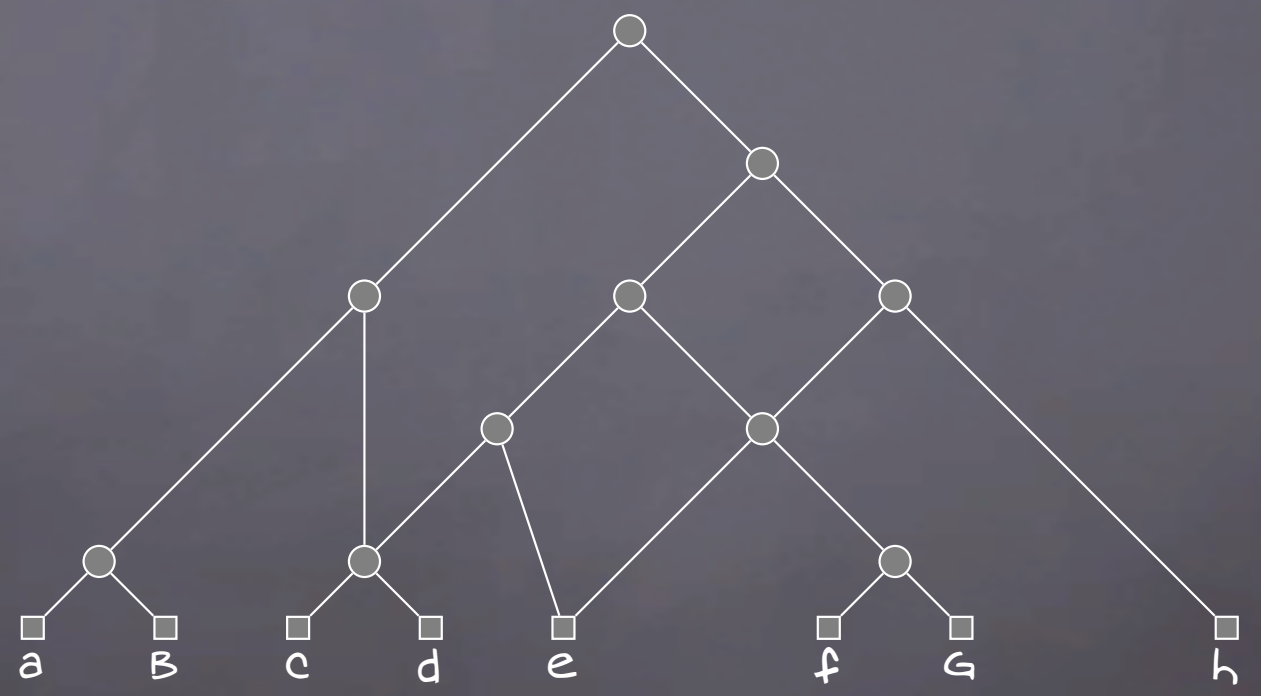
Observation

rooted network: cluster of u $\subseteq$ cluster of v $\Leftrightarrow$ u $\leq$ v

$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Example

{a,B,c,d},{c,d,e,f,G,h}, {c,d,e,f,G}, {e,f,G,h}, {c,d,e}, {e,f,G}, {a,B}, {c,d}, {f,G}



Rooted Network Reconstruction

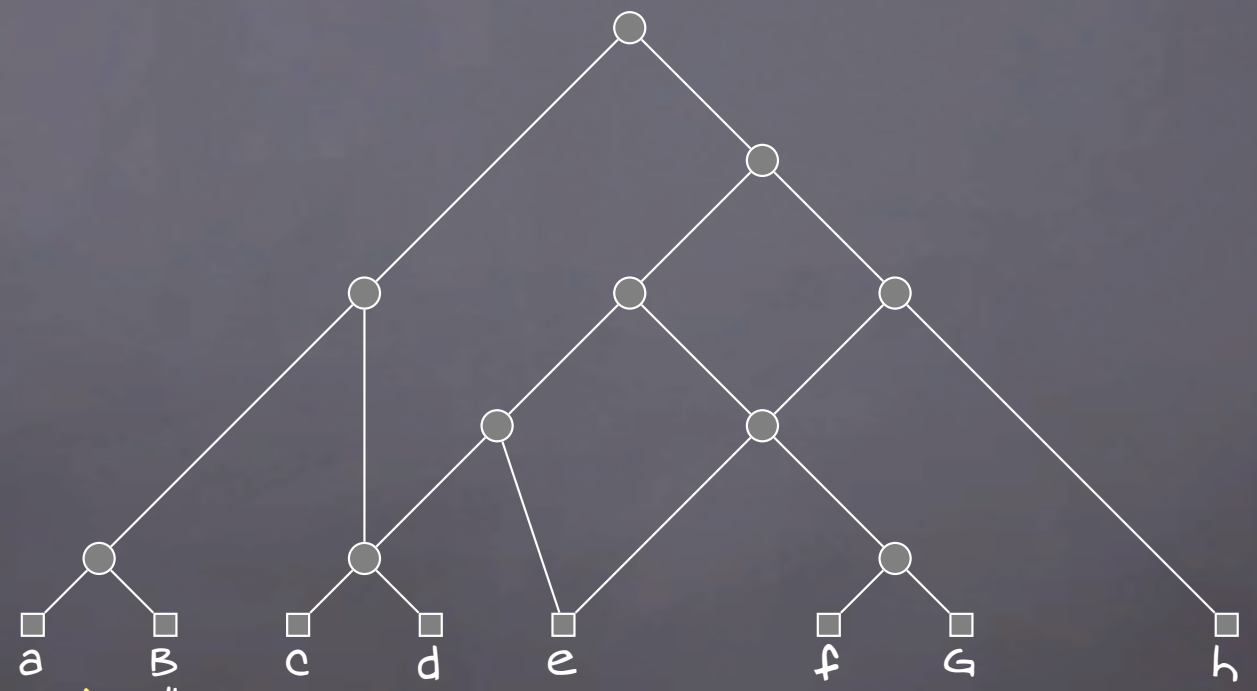
Observation

rooted network: cluster of u $\subseteq$ cluster of v $\Leftrightarrow$ u $\leq$ v

$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Example

{a,B,c,d},{c,d,e,f,G,h}, {c,d,e,f,G}, {e,f,G,h}, {c,d,e}, {e,f,G}, {a,B}, {c,d}, {f,G}



c.f. "cluster popping" [Huson & Rupp,'08]

Rooted Network Reconstruction

Observation

rooted network: cluster of $u \subseteq$ cluster of $v \Leftrightarrow u \leq v$
$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Problem

may produce more reticulations than necessary to explain the data

Rooted Network Reconstruction

Observation

rooted network: cluster of u $\subseteq$ cluster of v $\Leftrightarrow$ u $\leq$ v
$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Problem

may produce more reticulations than necessary to explain the data

Hybridization Number

Input: set of trees $\mathcal{T}$, int k

Question: Is there a network with $\leq k$ reticulations displaying all trees in $\mathcal{T}$?

Rooted Network Reconstruction

Observation

rooted network: cluster of $u \subseteq$ cluster of $v \Leftrightarrow u \leq v$

$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Problem

may produce more reticulations than necessary to explain the data

Hybridization Number

Input: set of trees $\mathcal{T}$, int k

Question: Is there a network with $\leq k$ reticulations displaying all trees in $\mathcal{T}$?

$\rightsquigarrow$ NP-hard for 2 trees [Bordewich & Semple,'07]

Rooted Network Reconstruction

Observation

rooted network: cluster of u $\subseteq$ cluster of v $\Leftrightarrow$ u $\leq$ v

$\rightsquigarrow$ rooted network is hasse diagram of its clusters

Problem

may produce more reticulations than necessary to explain the data

Hybridization Number

Input: set of trees $\mathcal{T}$, int k

Question: Is there a network with $\leq k$ reticulations displaying all trees in $\mathcal{T}$?

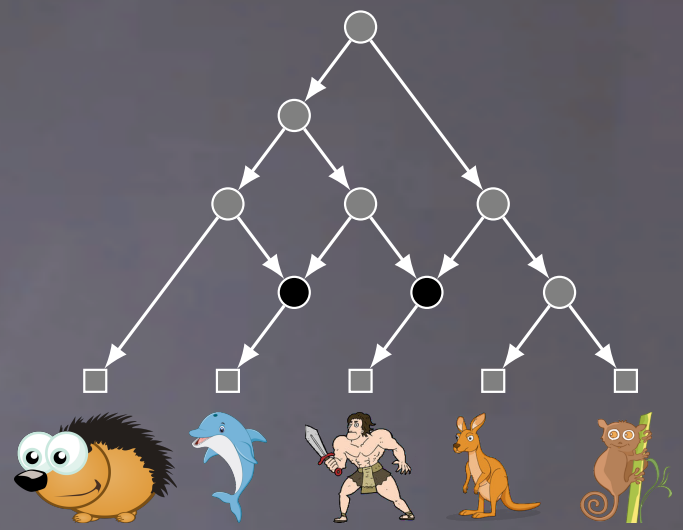
$\rightsquigarrow$ NP-hard for 2 trees [Bordewich & Semple,'07]

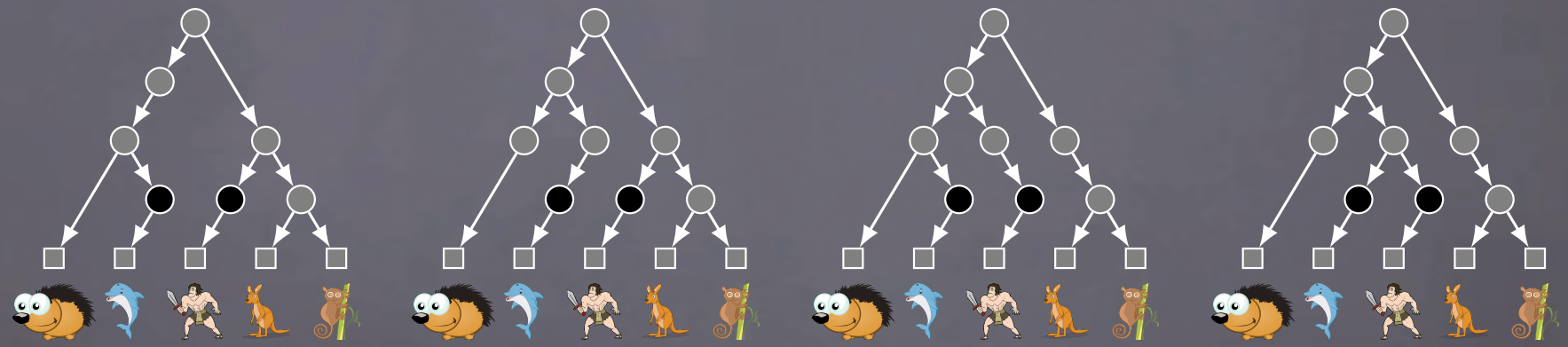
Note: $HN(T_1, T_2)$ = max. acyclic agreement forest − 1 [Baroni et al.,'05]

Networks Display Trees

Observation

A network may display up to $2^{|R|}$ trees.

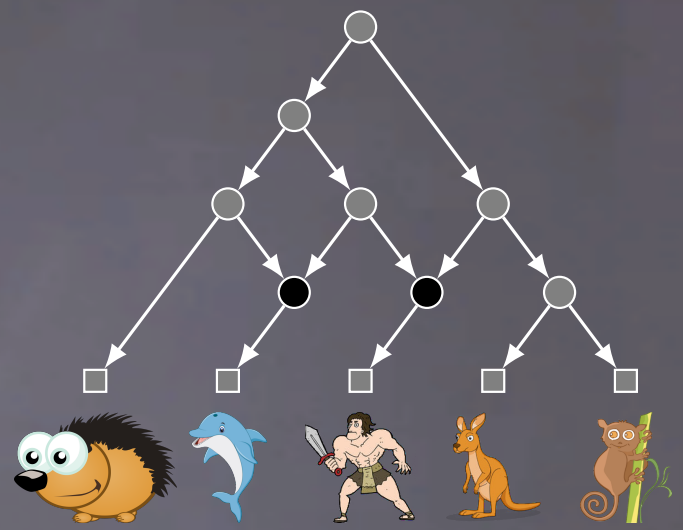


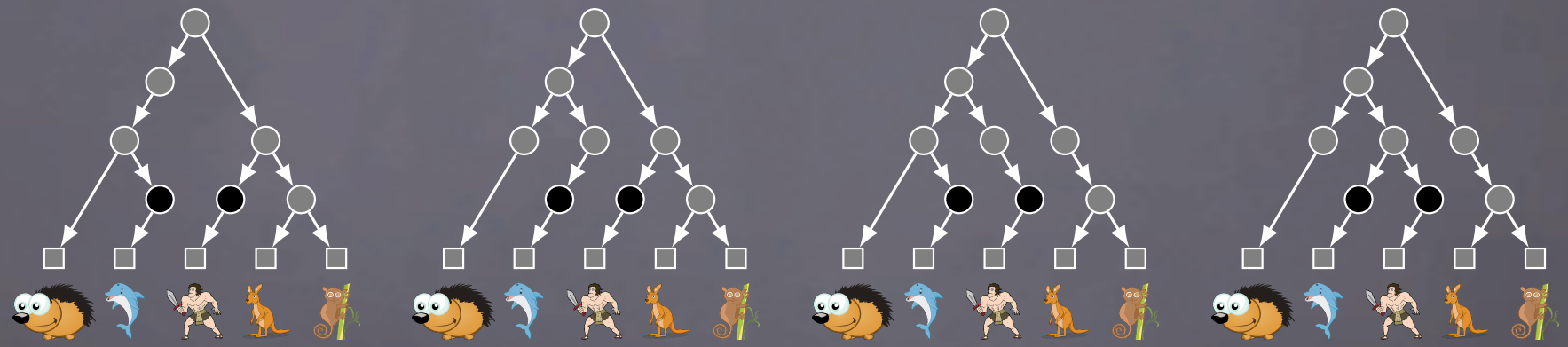


Networks Display Trees

Observation

A network may display up to $2^{|R|}$ trees.



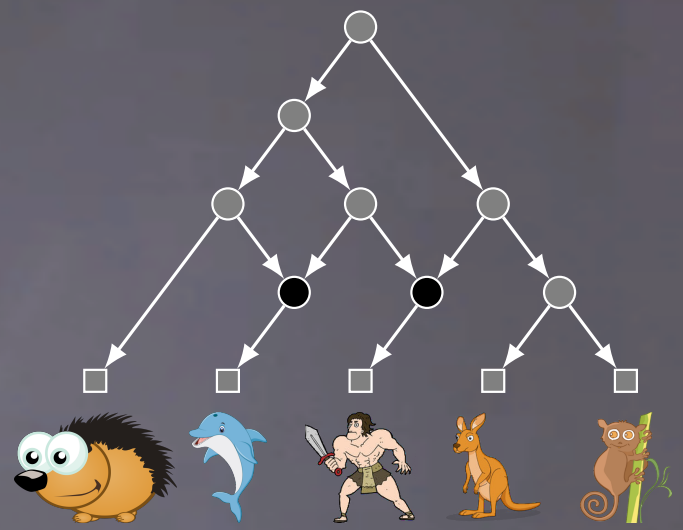


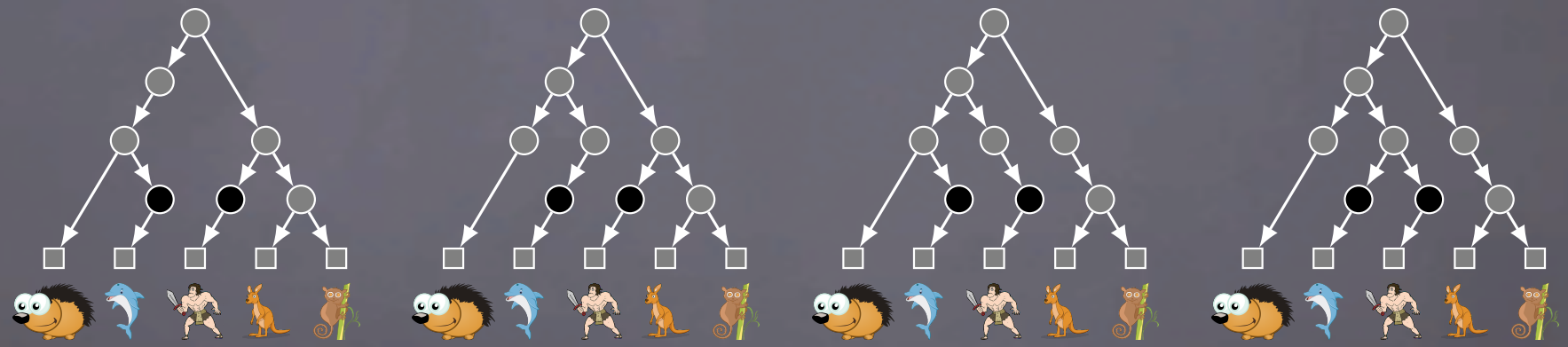
Networks Display Trees

Observation

A network may display up to $2^{|R|}$ trees.

But: how to decide if a given tree is displayed?





Networks Display Trees

Tree Containment

Input: a network N, a tree T

Question: Does N display T?

Networks Display Trees

Tree Containment

Input: a network N, a tree T

Question: Does N display T?

$\rightsquigarrow$ NP-hard (from Disjoint Paths) [Kanj et al.'08]

Networks Display Trees

Tree Containment

Input: a network N, a tree T

Question: Does N display T?

$\rightsquigarrow$ NP-hard (from Disjoint Paths) [Kanj et al.'08]

Note: linear time on reticulation visible N [Gunawan,'18][Weller,'18]

Networks Display Trees

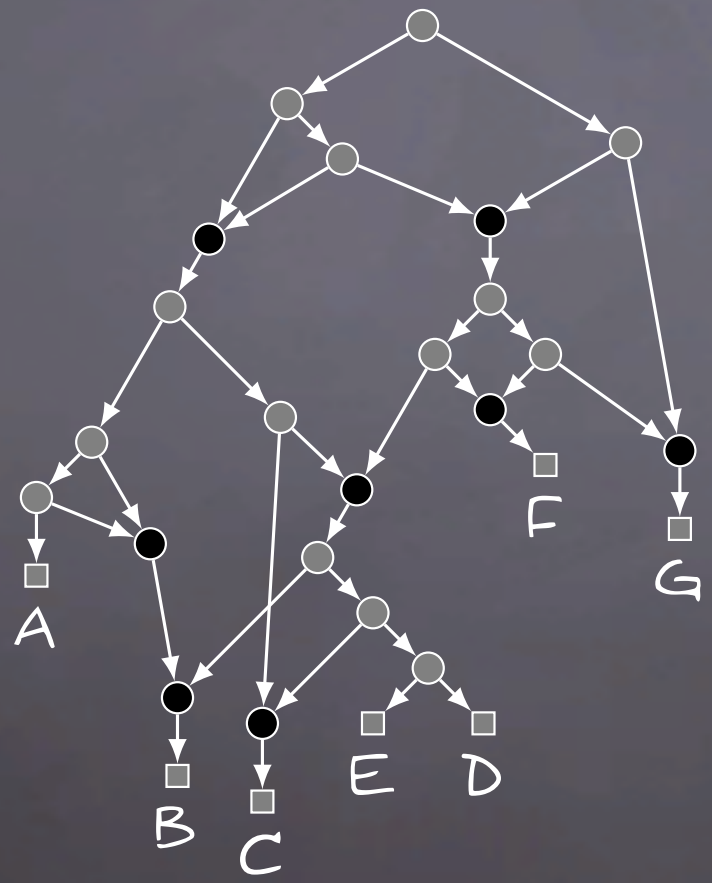
Tree Containment

Input: a network N, a tree T

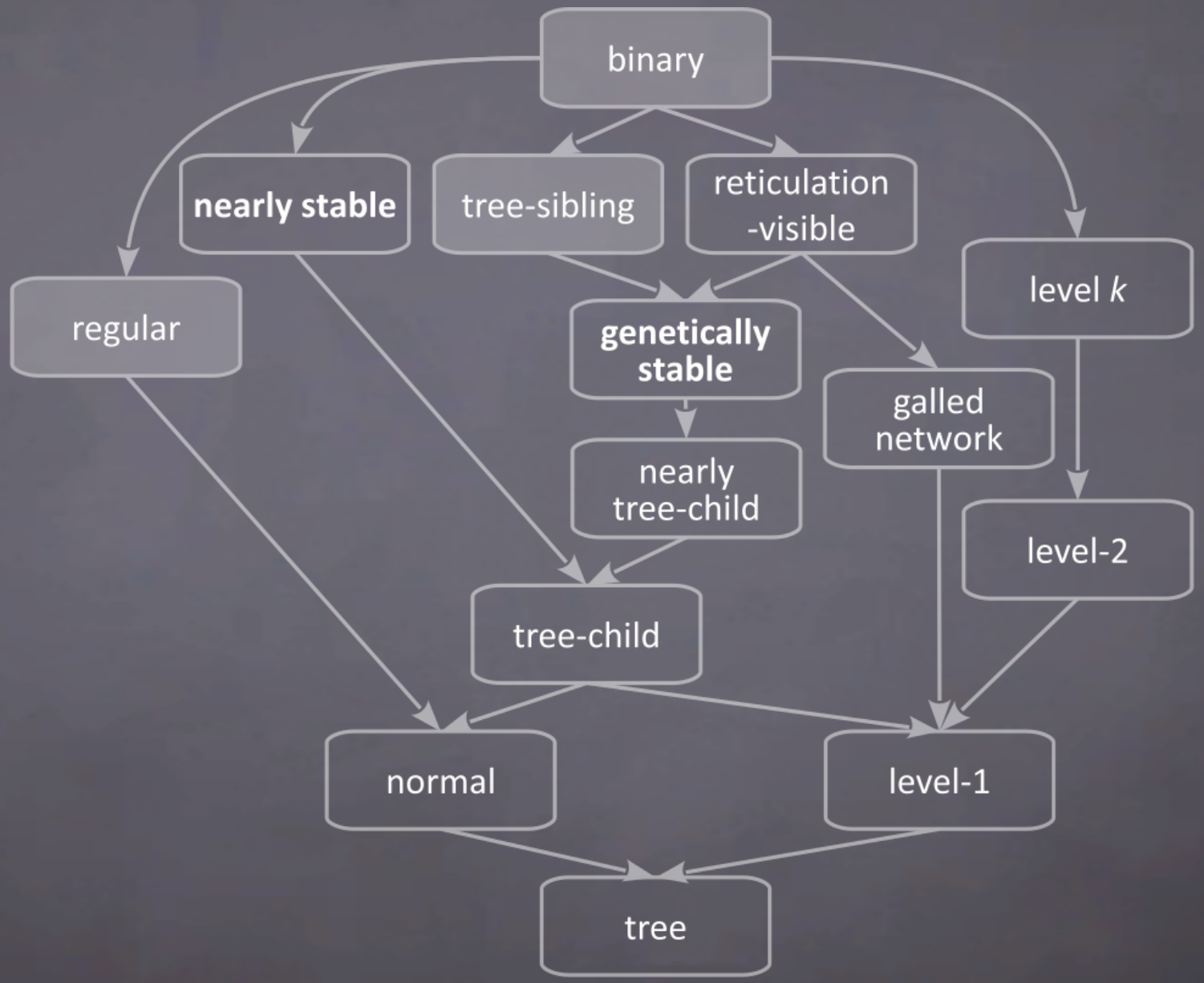
Question: Does N display T?

⇝ NP-hard (from Disjoint Paths) [Kanj et al.'08]

Note: linear time on reticulation visible N [Gunawan,'18][Weller,'18]



Small Taxonomy of Network Classes



c.f. "Who is Who in Phylogenetic Networks" (http://phylnet.univ-mlv.fr/)


Thanks
& Enjoy
Part III