Introduction to Bioinformatics for Computer Scientists

Lecture 7

Plan for next lectures

- Today: Phylogenetic search algorithms
- Lecture 8 (Alexis): Statistical Models of Evolution 1
- Lecture 9 (Benoit): Discrete Operations on Trees

Phylogenetic Inference so far

- Distinction between
  - **distance-based versus character-based methods**
- Alignment-free versus alignment-based methods
- Ultrametric versus non-ultrametric trees
- Some file formats
- Some terminology
- Molecular versus morphological data
- The number of unrooted binary trees
- Scoring trees and NP-hardness
- What can we do with phylogenies?
- The Neighbor-Joining algorithm

The UPGMA algorithm

- Usually introduced before Neighbor Joining *NJ* → it is simpler and older
- UPGMA is practically not used any more today for phylogeny reconstruction, but it is used for progressive multiple sequence alignment (see MUSCLE algorithm)
- In contrast to NJ it produces *ultrametric* trees!
- It produces rooted trees
- UPGMA stands for: *Unweighted Pair Group Method with Arithmetic Mean*
- Like NJ it uses a distance matrix *D* for clustering/joining nodes
- UPGMA can be used if we know that we have an ultrametric tree!

  → this is usually not the case!

UPGMA example

	A	B	C	D
A		0.4	0.6	0.6
B			0.6	0.6
C				0.2
D				

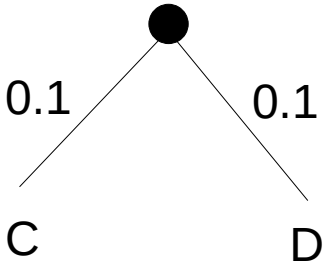
We will first walk through the algorithm and then look at the formal description!

UPGMA example

	A	B	C	D
A		0.4	0.6	0.6
B			0.6	0.6
C				0.2
D				

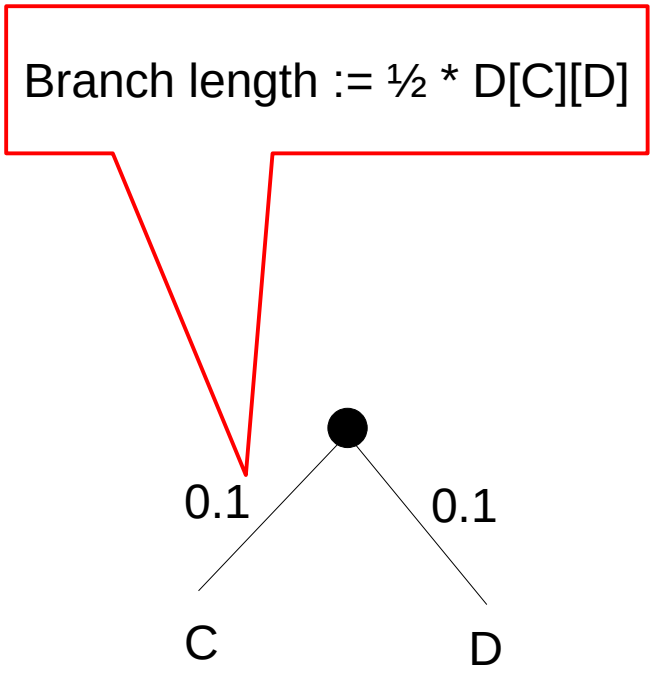
UPGMA example

	A	B	C	D
A		0.4	0.6	0.6
B			0.6	0.6
C				0.2
D				



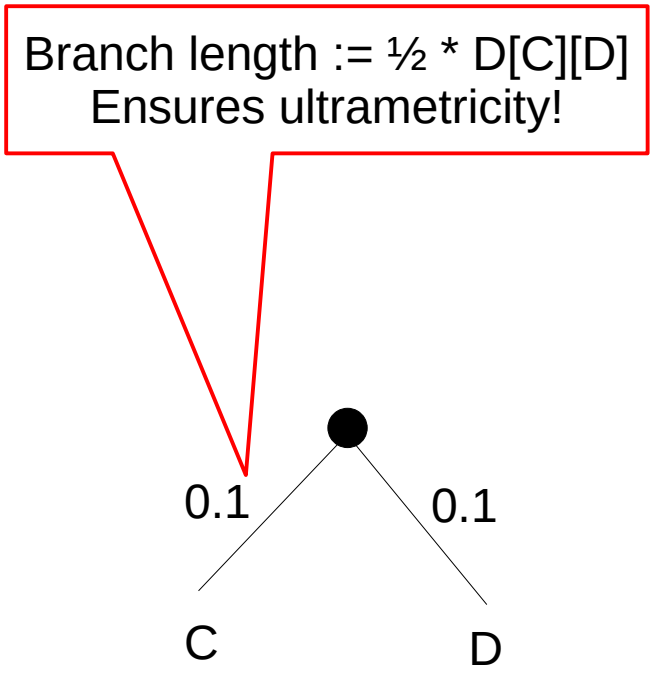
UPGMA example

	A	B	C	D
A		0.4	0.6	0.6
B			0.6	0.6
C				0.2
D				



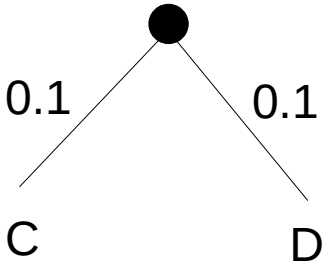
UPGMA example

	A	B	C	D
A		0.4	0.6	0.6
B			0.6	0.6
C				0.2
D				



UPGMA example

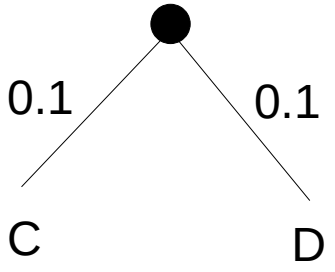
	A	B	C	D
A		0.4	0.6	0.6
B			0.6	0.6
C				0.2
D				



D [A][(C,D)] = ½ * 0.6 + ½ * 0.6
D [B][(C,D)] = ½ * 0.6 + ½ * 0.6

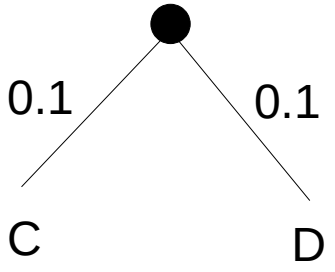
UPGMA example

	A	B	(C,D)
A		0.4	0.6
B			0.6
(C, D)			



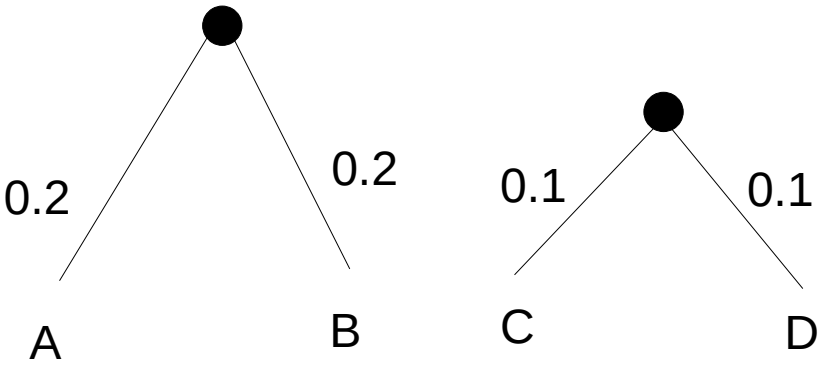
UPGMA example

	A	B	(C,D)
A		0.4	0.6
B			0.6
(C, D)			



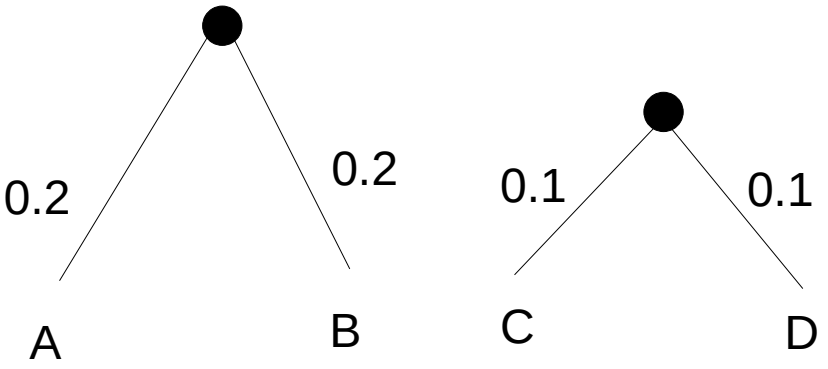
UPGMA example

	A	B	(C,D)
A		0.4	0.6
B			0.6
(C, D)			



UPGMA example

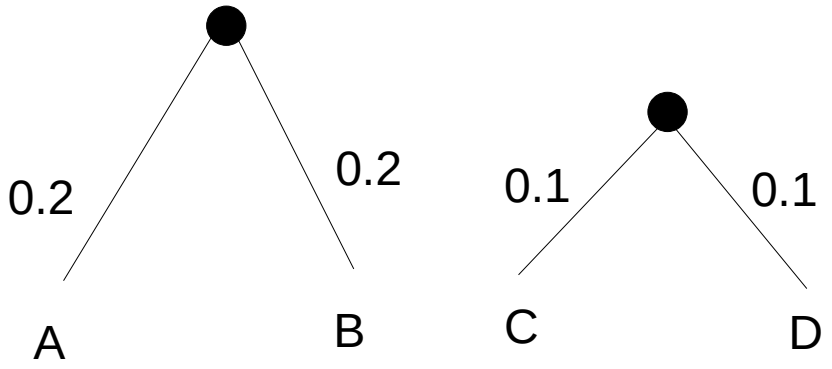
	A	B	(C,D)
A		0.4	0.6
B			0.6
(C, D)			



D[A,B][C,D] = ½ * 0.6 + ½ * 0.6

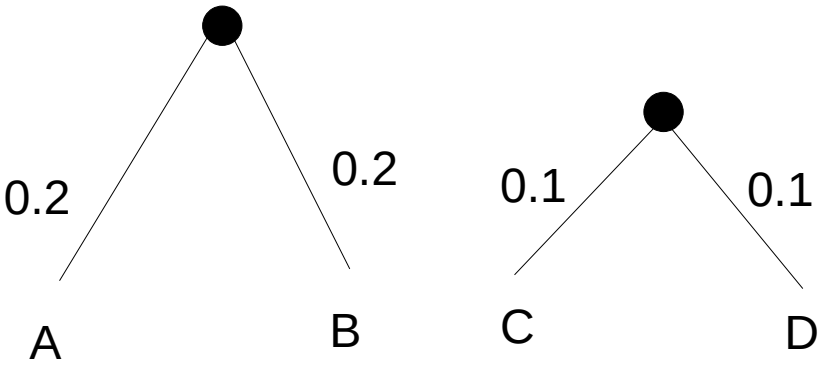
UPGMA example

	(A,B)	(C,D)
(A,B)		0.6
(C,D)		



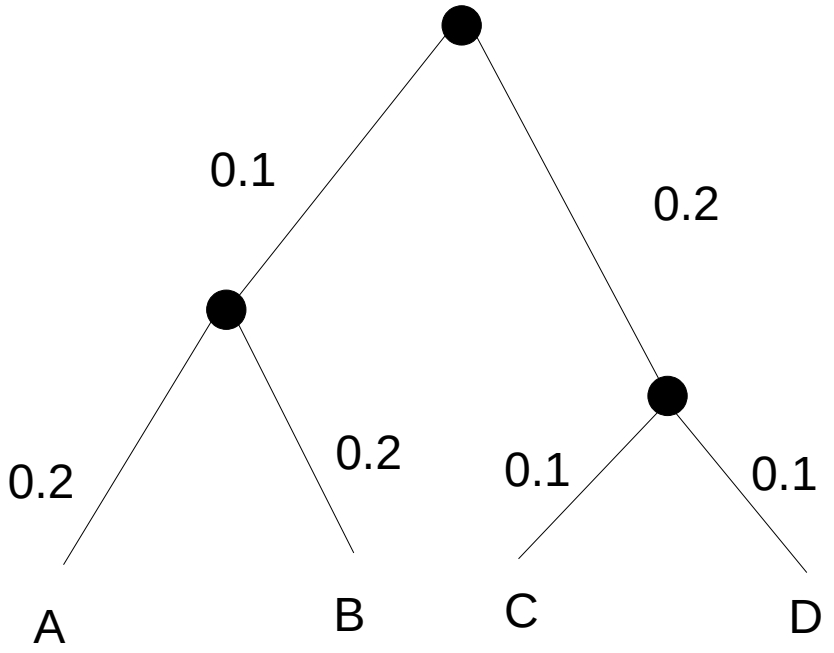
UPGMA example

	(A,B)	(C,D)
(A,B)		0.6
(C,D)		



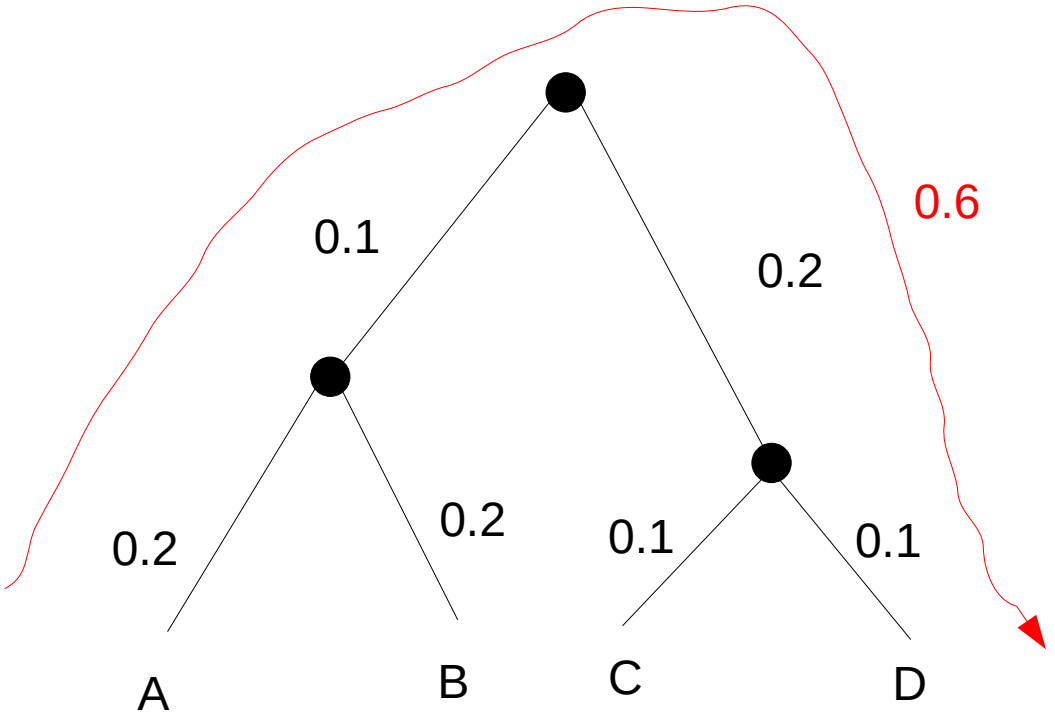
UPGMA example

	(A,B)	(C,D)
(A,B)		0.6
(C,D)		



UPGMA example

	(A,B)	(C,D)
(A,B)		0.6
(C,D)		

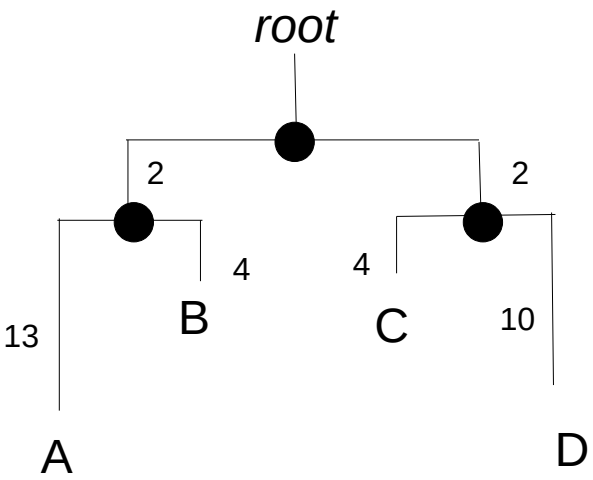


UPGMA Formal description

- Find the minimum *D[i][j]*
- Merge *i* and *j* → *(i,j)*
- This new group has $n_{(i,j)}$ members, where $n_{(i,j)} := n_i + n_j$
- Connect *i* and *j* to form a new node *(i,j)*
- Assign the two branches connecting *i* → *(i,j)* and *j* → *(i,j)* the length *D[i][j]/2*
- Update the distances between *(i,j)* and all *k*, where $k \neq i$ and $k \neq j$ via $D[(i,j)][k] = (n_i/(n_i+n_j)) * D[i][k] + (n_j/(n_i+n_j)) * D[j][k]$
- Naive implementation: $O(n^3)$ → search for minimum in each instance of matrix *D*
- Maintain a list of per-column (or per-row) minima
  - → update list $O(n)$
  - → look for minimum $O(n)$
  - → $O(n^2)$
- In contrast to NJ we don't need to update the entire matrix each time, thus only $O(n^2)$

UPGMA on non-ultrametric trees

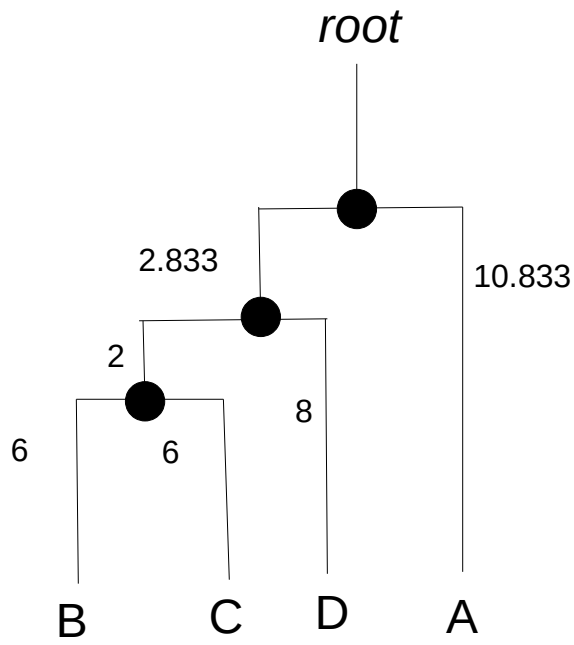
- Can yield misleading results
- Most trees are not ultrametric → do not have equal evolutionary rates among all lineages



True tree

	A	B	C	D
A	0	17	21	27
B		0	12	18
C			0	14
D				0

Patristic distance matrix

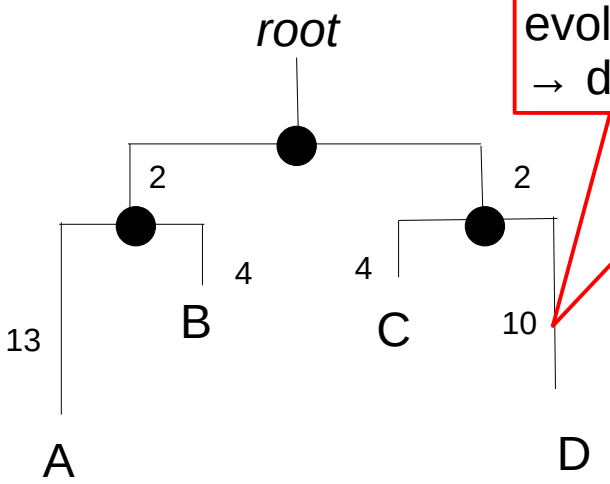


UPGMA tree

UPGMA on non-ultrametric trees

- Can yield misleading results
- Most trees are not ultrametric → do not have equal evolutionary r ages

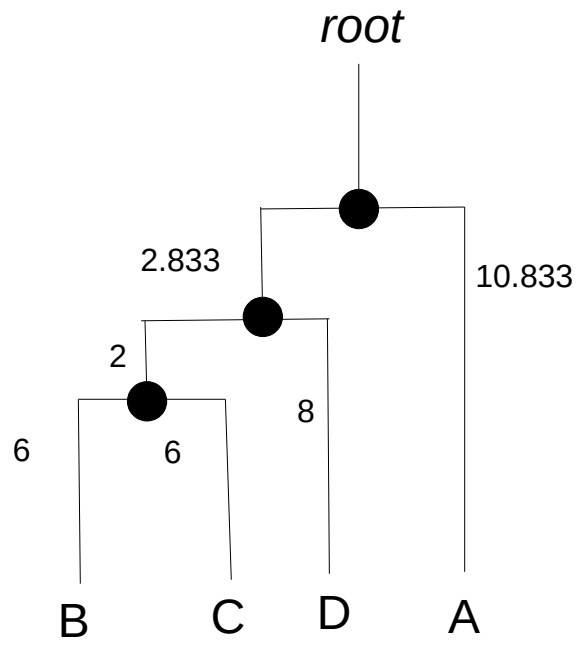
Imagine a higher evolutionary pressure!
→ difficult life conditions!



True tree

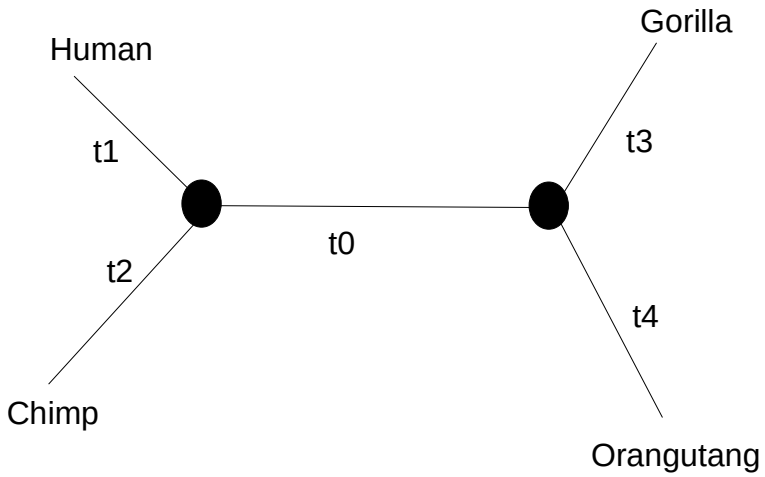
	A	B	C	D
A	0	17	21	27
B		0	12	18
C			0	14
D				0

Patristic distance matrix

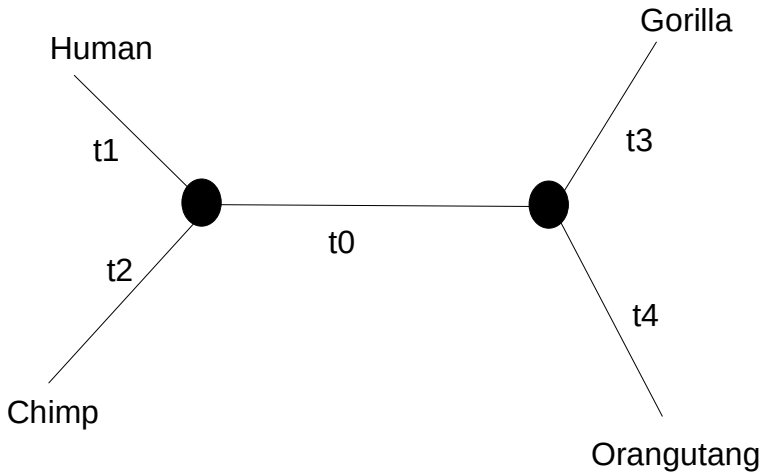


UPGMA tree

Least Squares



Least Squares



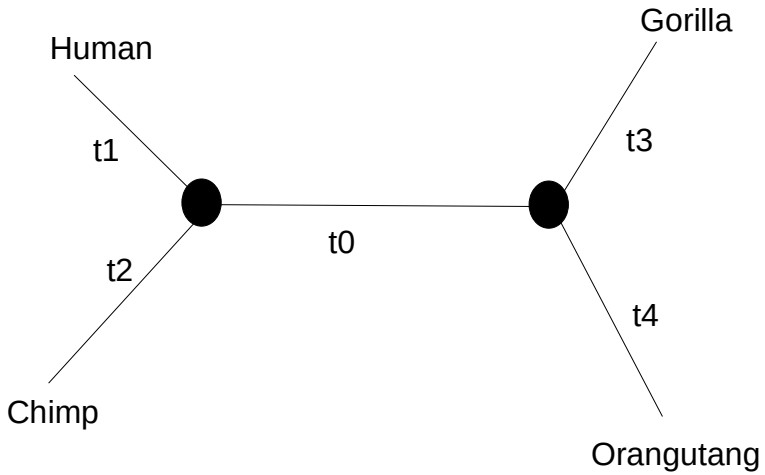
Patristic distances

d[H][C] = t1 + t2
d[H][G] = t1 + t0 + t3
d[H][O] = t1 + t0 + t4
d[C][G] = t2 + t0 + t3
d[C][O] = t2 + t0 + t4
d[G][O] = t3 + t4

Least Squares

Given distance matrix D

	H	C	G	O
H		0.0965	0.1140	0.1849
C			0.1180	0.2009
G				0.1947
O				

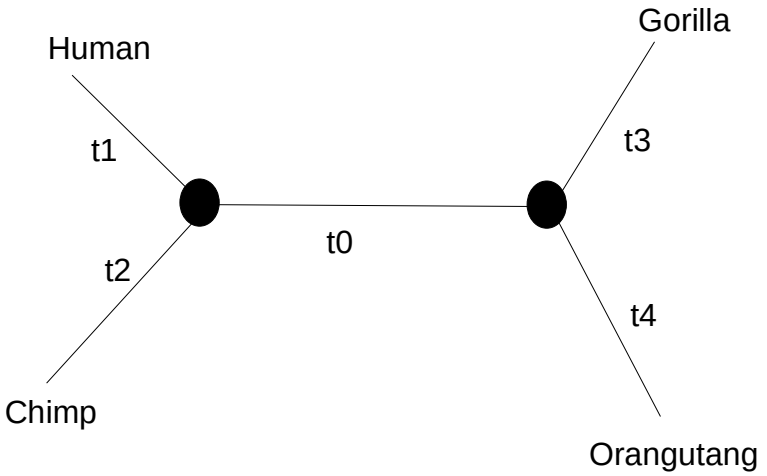


d[H][C] = t1 + t2
d[H][G] = t1 + t0 + t3
d[H][O] = t1 + t0 + t4
d[C][G] = t2 + t0 + t3
d[C][O] = t2 + t0 + t4
d[G][O] = t3 + t4

Least Squares

Given distance matrix D

	H	C	G	O
H		0.0965	0.1140	0.1849
C			0.1180	0.2009
G				0.1947
O				



Find *t0, t1, ..., t4* such that deviation of *d[i][j]* from *D[i][j]* is minimized!

$Q := (d[H][C] - D[H][C])^2 + (d[H][G] - D[H][G])^2 + (d[H][O] - D[H][O])^2 + (d[C][G] - D[C][G])^2 + (d[C][O] - D[C][O])^2 + (d[G][O] - D[G][O])^2$

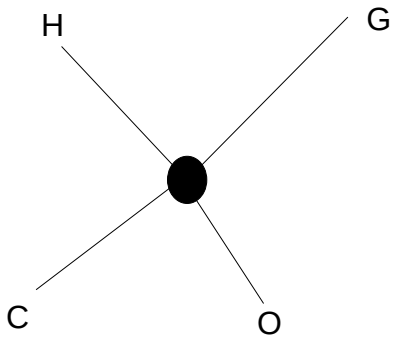
$d[H][C] = t1 + t2$
$d[H][G] = t1 + t0 + t3$
$d[H][O] = t1 + t0 + t4$
$d[C][G] = t2 + t0 + t3$
$d[C][O] = t2 + t0 + t4$
$d[G][O] = t3 + t4$

Least Squares Example

tree	t0	t1	t2	t3	t4	Q
((H,C),G,O)	0.008840	0.043266	0.053280	0.058908	0.135795	0.000035
((H,G),C,O)	0.000000	0.046212	0.056227	0.061854	0.138742	0.000140
((H,O),C,G)	As above	-	-	-	-	-

Least Squares Example

Star tree



tree	t0	t1	t2	t3	t4	Q
((H,C),G,O)	0.008840	0.043266	0.053280	0.058908	0.135795	0.000035
((H,G),C,O)	0.000000	0.046212	0.056227	0.061854	0.138742	0.000140
((H,O),C,G)	As above	-	-	-	-	-

Least Squares Optimization

- Given a fixed, fully binary, unrooted tree T with n taxa
- Given a pair-wise distance matrix D
- Assign branch lengths t_1 , ..., t_{2n-3} to the tree such that:

  the sum of the squared differences between the pair-wise *patristic* (tree-based!) distances d_{ij} and the *plain* pair-wise distances D_{ij} is minimized

- In other words:

  $Q = \Sigma_{i<j} (D_{ij} - d_{ij})^2$ → find an assignment t_1 ,..., t_{2n-3} to the tree such that Q is minimized

  Q can be minimized by taking the derivative and solving a system of linear equations in $O(n^3)$

  Minimization methods for Q that take into account the tree-like structure run in $O(n^2)$ or even $O(n)$

- Then, also find that tree topology T that minimizes Q
- Finding the minimal least squares tree is NP-hard

W.H.E. Day “Computational Complexity of Inferring Phylogenies from dissimilarity matrices”, *Bulletin of Mathematical Biology* 49: 461-467, 1986.

Least Squares

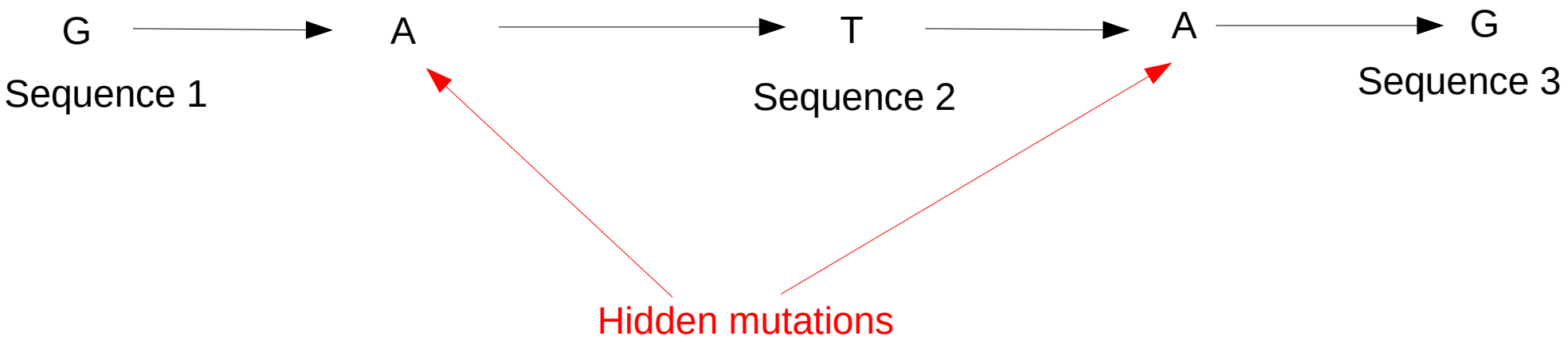
- *NP-hard* because of tree search problem
- Scoring a single tree takes time between $O(n)$ to $O(n^3)$
- There also exist weighted versions:

  $Q = \Sigma_{i<j}\, w_{ij}(D_{ij} - d_{ij})^2$

  where $w_{ij} := 1/D_{ij}$ *or* $w_{ij} := 1/D_{ij}^2$
- We will see how to search for trees a bit later-on
- Make sure you understand the difference between
  - Scoring a single tree
  - Searching for the tree with the best score

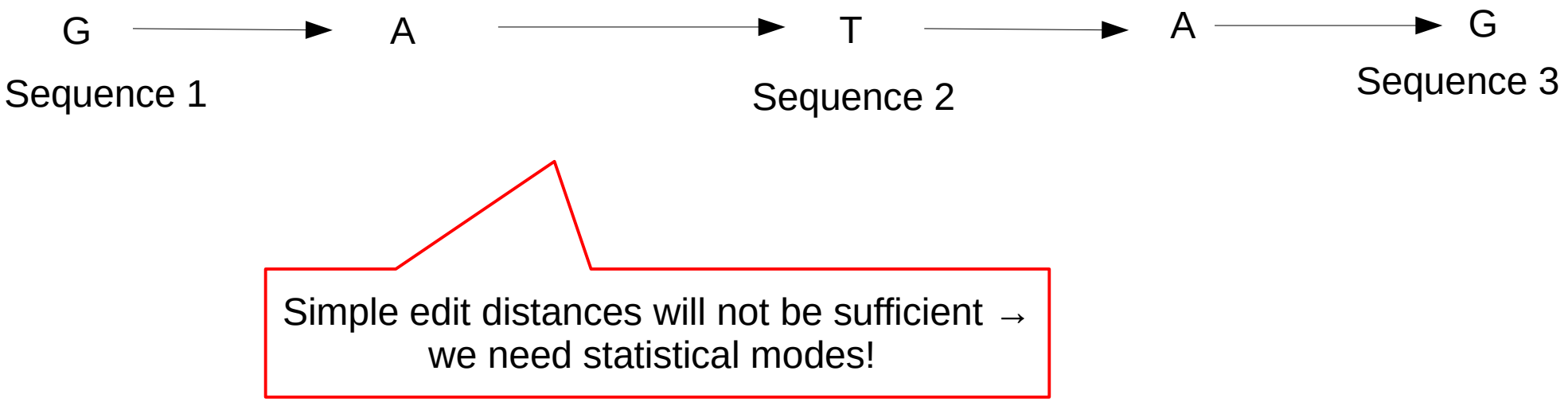
Distances

- A preview of the next lecture
- We need to accommodate multiple substitutions in the evolutionary history of sequences



Distances

- A preview of the next lecture
- We need to accommodate multiple substitutions in the evolutionary history of sequences



Minimum Evolution Method

- Similar to least squares
- Explicit Criterion → minimize total branch length (tree length) of the reconstructed tree
- Branch lengths are obtained using least-squares method → same time complexity
- Instead of searching for the tree that minimizes the squared difference between *D[i][j]* and *d[i][j]* that is denoted by *Q* we search for the tree where *t0* + *t1* + *t2* + *t3* + *t4* is minimized

tree	t0	t1	t2	t3	t4	Q	Tree length
((H,C),G,O)	*0.008840*	*0.043266*	*0.053280*	*0.058908*	*0.135795*	0.000035	***0.240741***
((H,G),C,O)	*0.000000*	*0.046212*	*0.056227*	*0.061854*	*0.138742*	0.000140	*0.303035*
((H,O),C,G)	As above	-	-	-	-	-	

Distance-based Methods

- Clustering Algorithms/Heuristics
  - Neighbor Joining
    → Heuristic for Minimum Evolution Method
  - UPGMA
- Explicit criteria
  - least squares
  - minimum evolution
- All depend on the accuracy of the pair-wise distance matrix *D*
- The distance matrix needs to be an exact reflection of the tree

Character-based Methods

- Parsimony
- Maximum Likelihood
- Bayesian Inference

The Parsimony Criterion

- Directly operates on the MSA
- Find the tree that explains the data with the least amount of mutations
- Questions:
  - How do we count the least amount of mutations on a given tree?

    → dynamic programming algorithm
  - How do we find the tree topology that requires the least amount of mutations

    → requires a tree search!

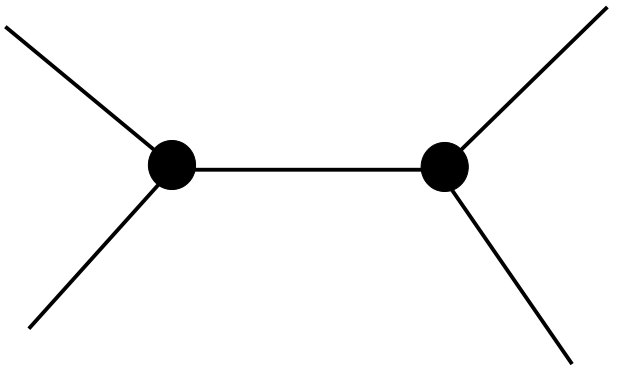
    → remember the number of trees!

    → this is also NP-hard!

Parsimony

MSA

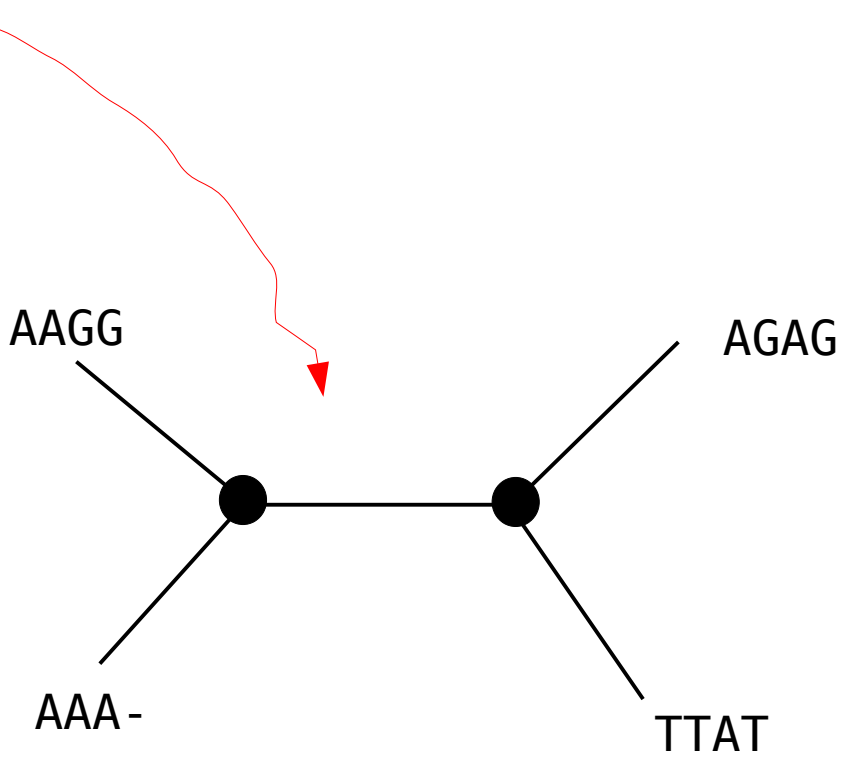
```
S1: AAGG
S2: AAA-
S3: AGAG
S4: TTAT
```



Parsimony

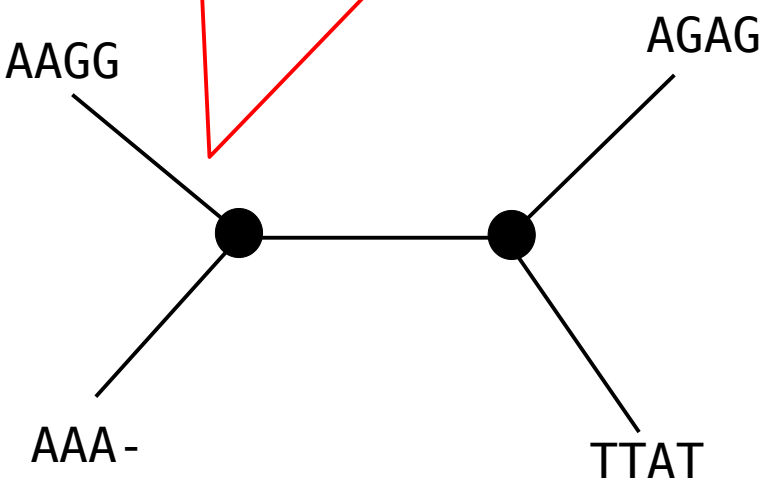
MSA

S1: AAGG
S2: AAA-
S3: AGAG
S4: TTAT



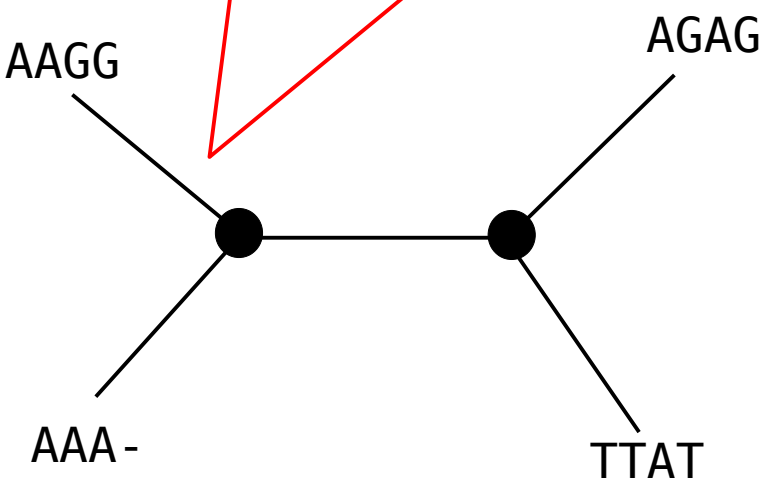
Parsimony

Find an assignment of sequences
to inner nodes such that the number of mutations
on the tree is minimized

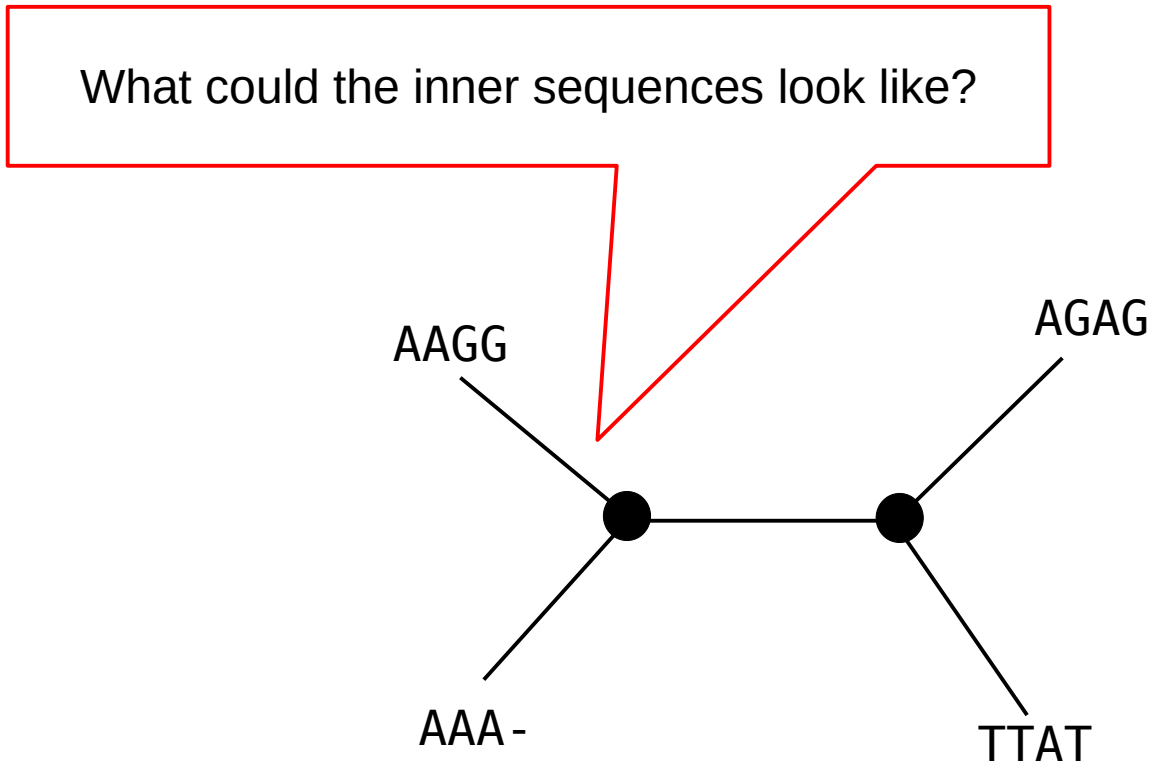


Parsimony

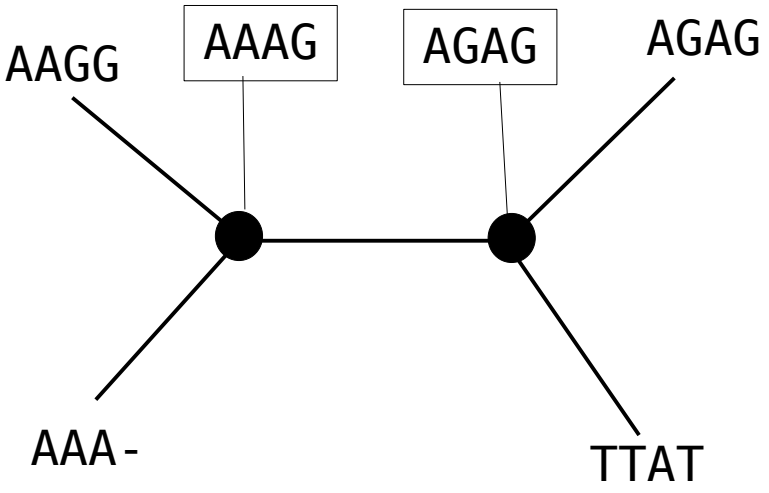
This is somewhat similar to the tree alignment problem, but here, we are given an alignment!



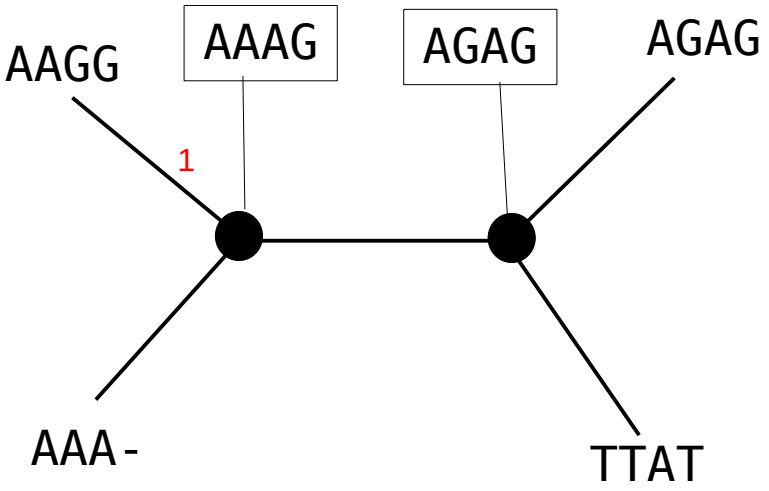
Parsimony



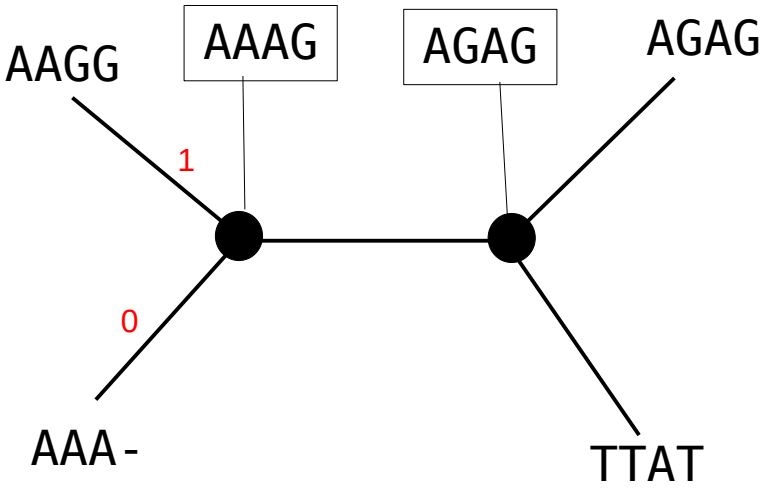
Parsimony



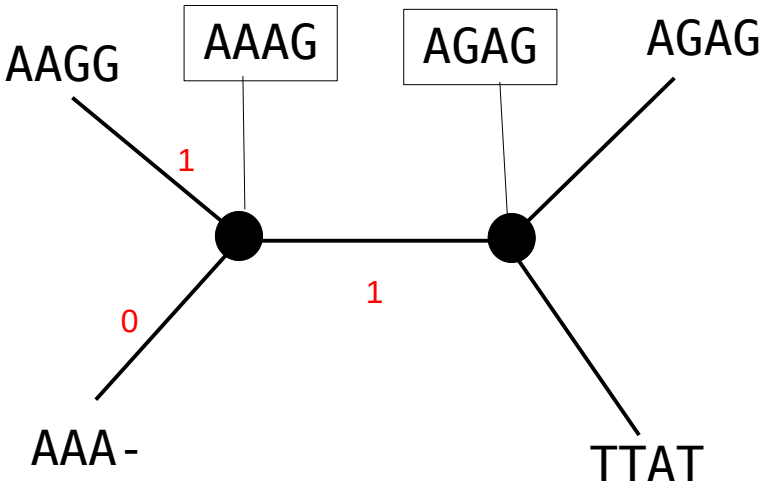
Parsimony



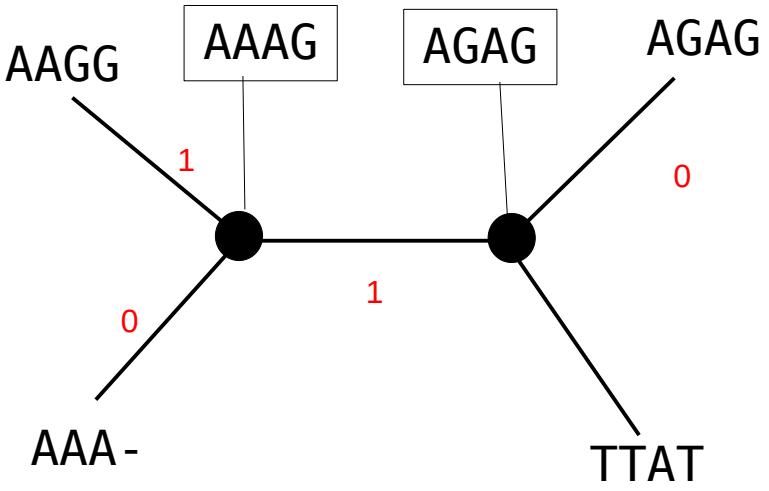
Parsimony



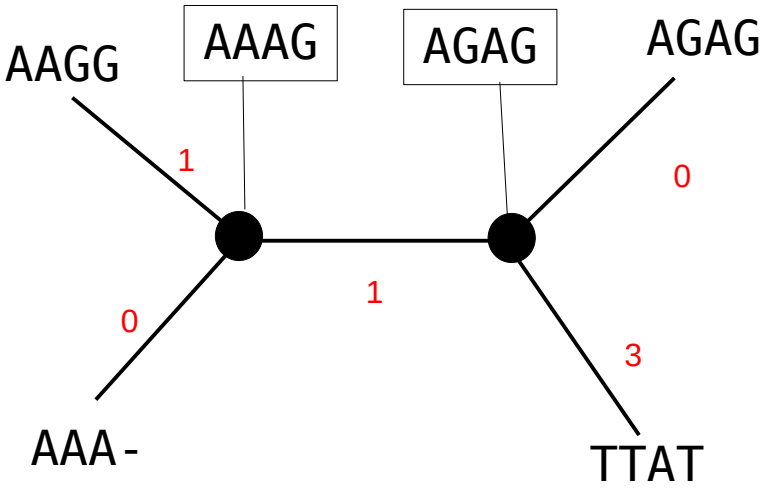
Parsimony



Parsimony

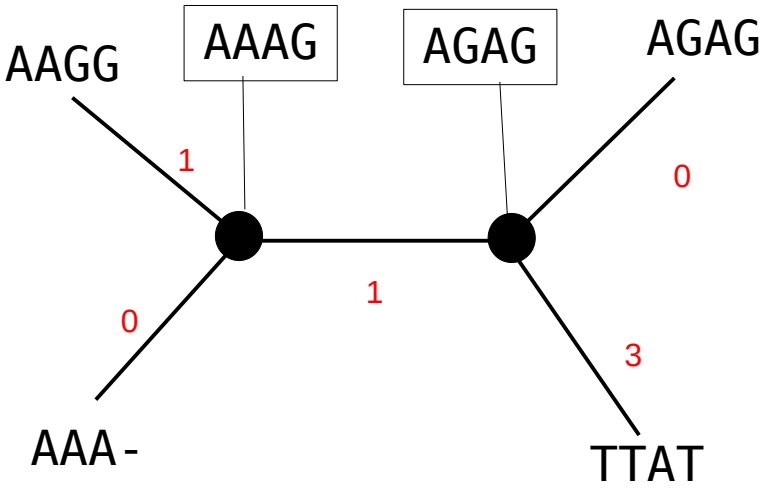


Parsimony



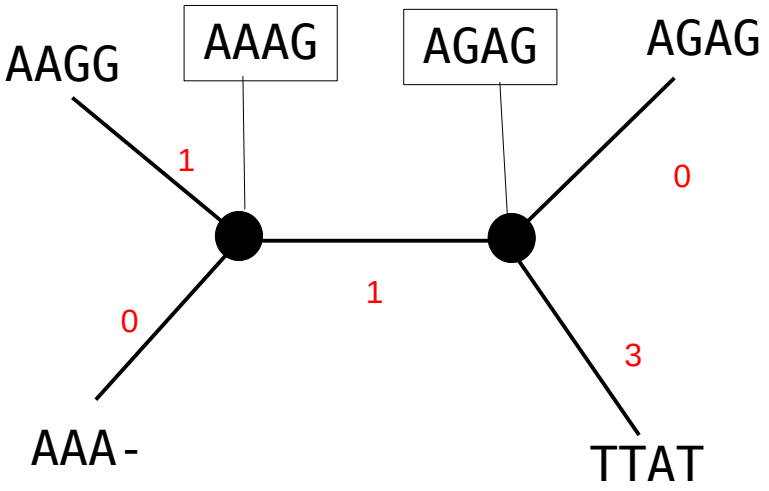
Parsimony

Parsimony Score of this tree = 5



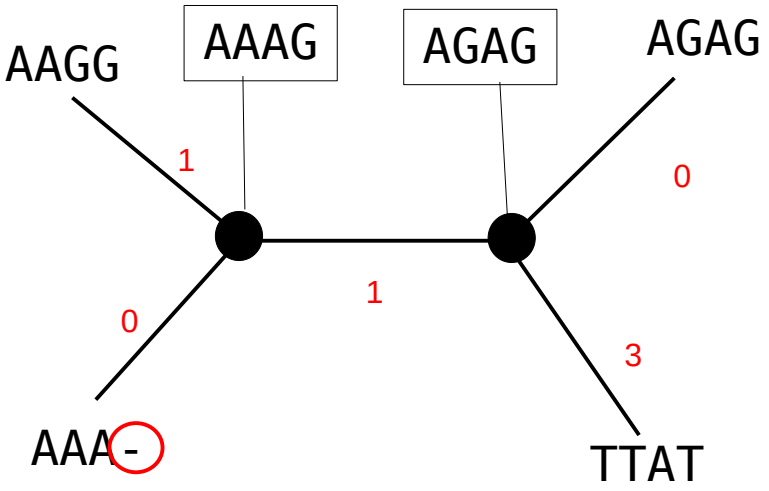
Parsimony

Parsimony Score of this tree = 5
This is also the minimum score for this tree.



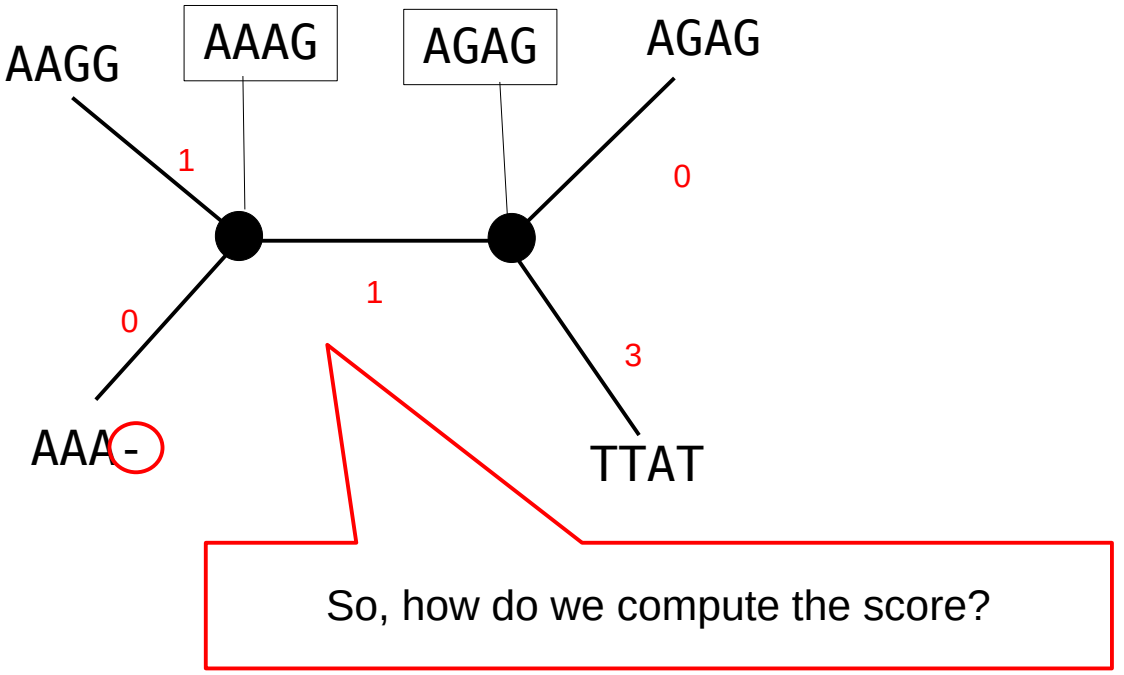
Parsimony

Parsimony Score of this tree = 5

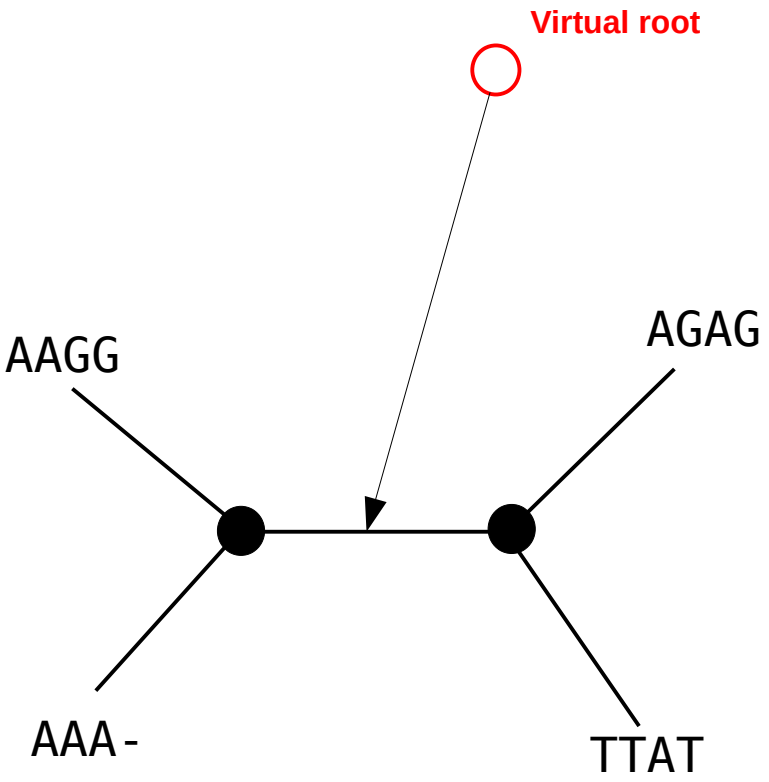


Gaps (also called Indels → Insertions or Deletions) are treated as so-called undetermined characters, also frequently denoted as ***N*.**
The interpretation is that *N* could be either *A, C, G*, or *T*.

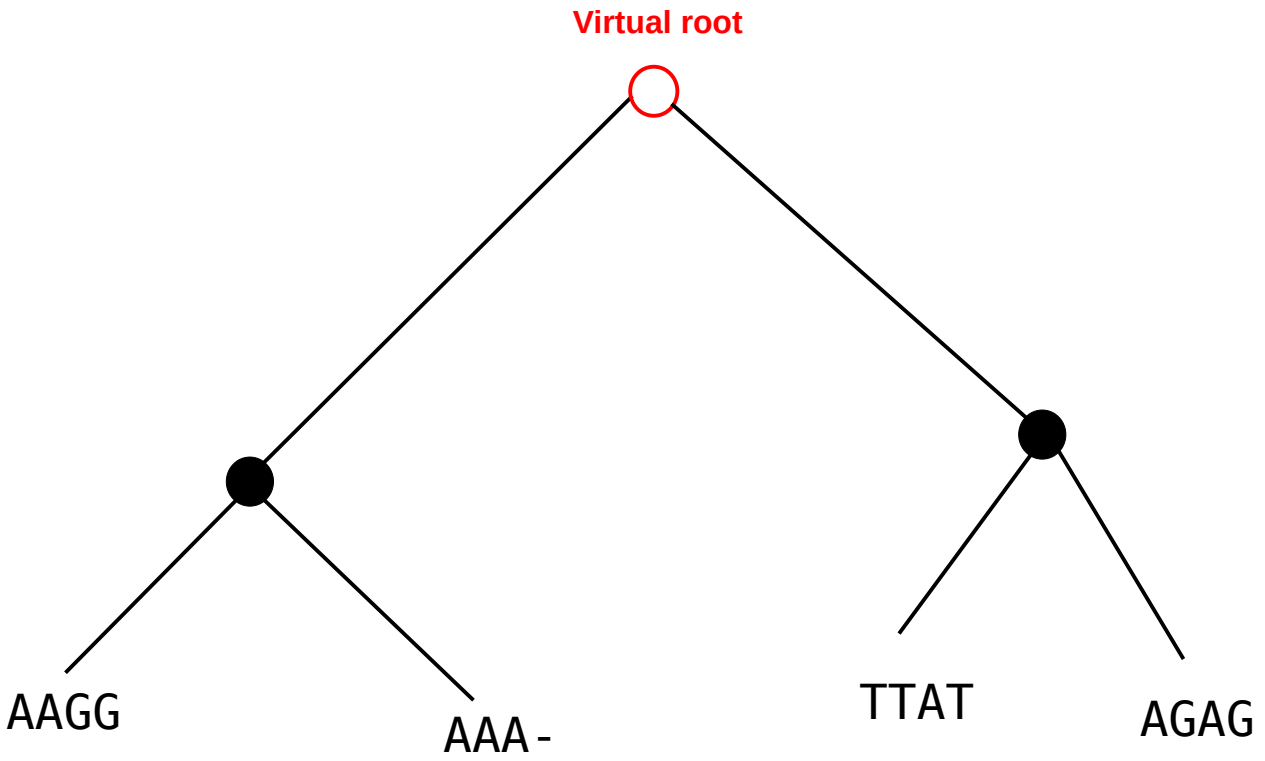
Parsimony


AAGG
AAAG
AGAG
AGAG
1
0
1
0
3
AAA-
TTAT
So, how do we compute the score?

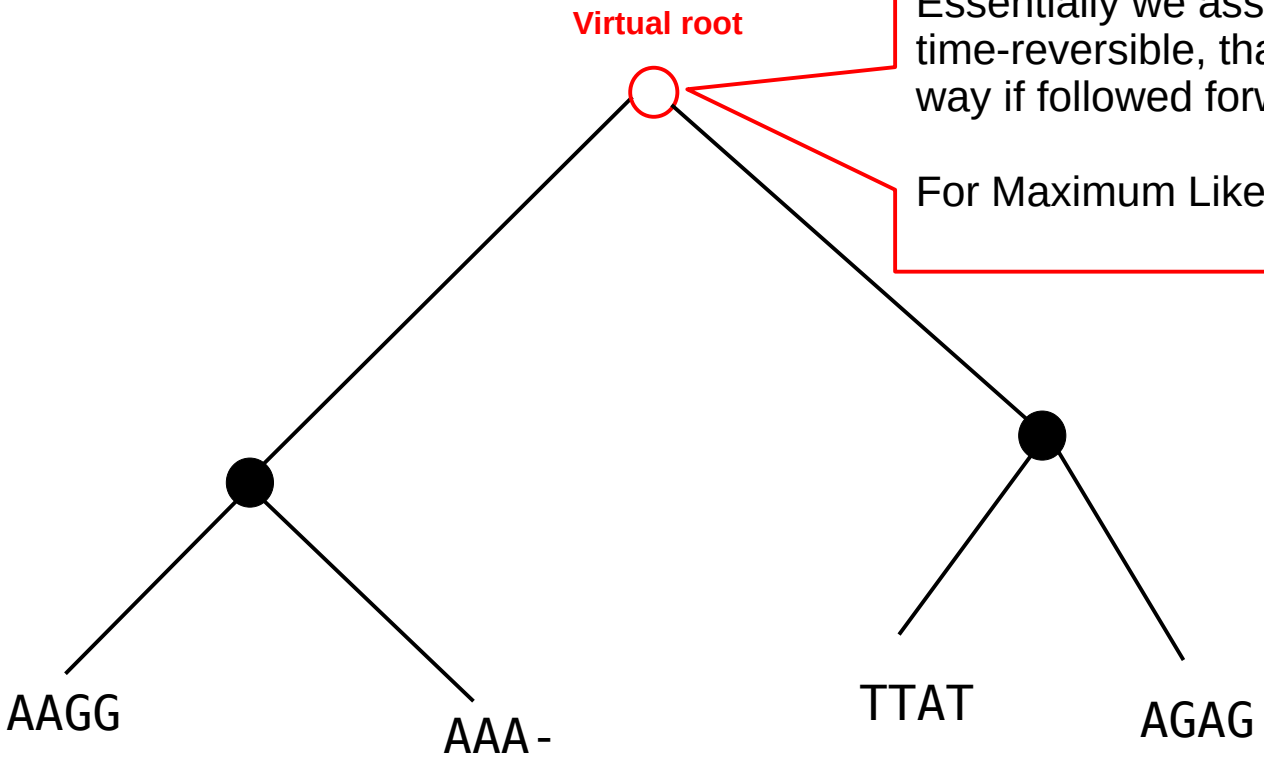
Parsimony



Parsimony



Parsimony



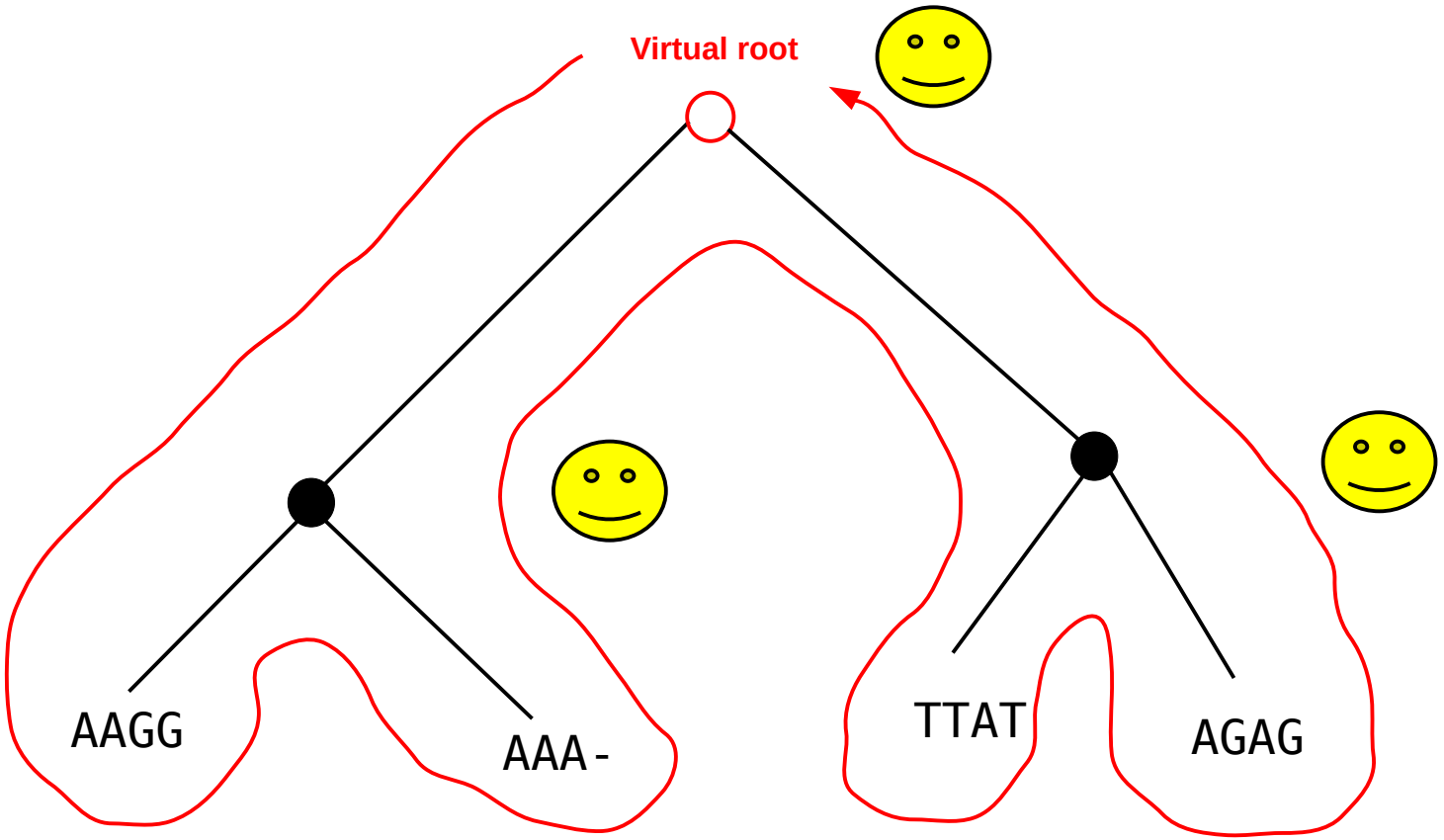
The score will be the same, regardless of where we place the root
→ makes computations much easier

Essentially we assume that evolution is time-reversible, that is, it occurred in the same way if followed forward or backward in time

For Maximum Likelihood we use a similar concept!

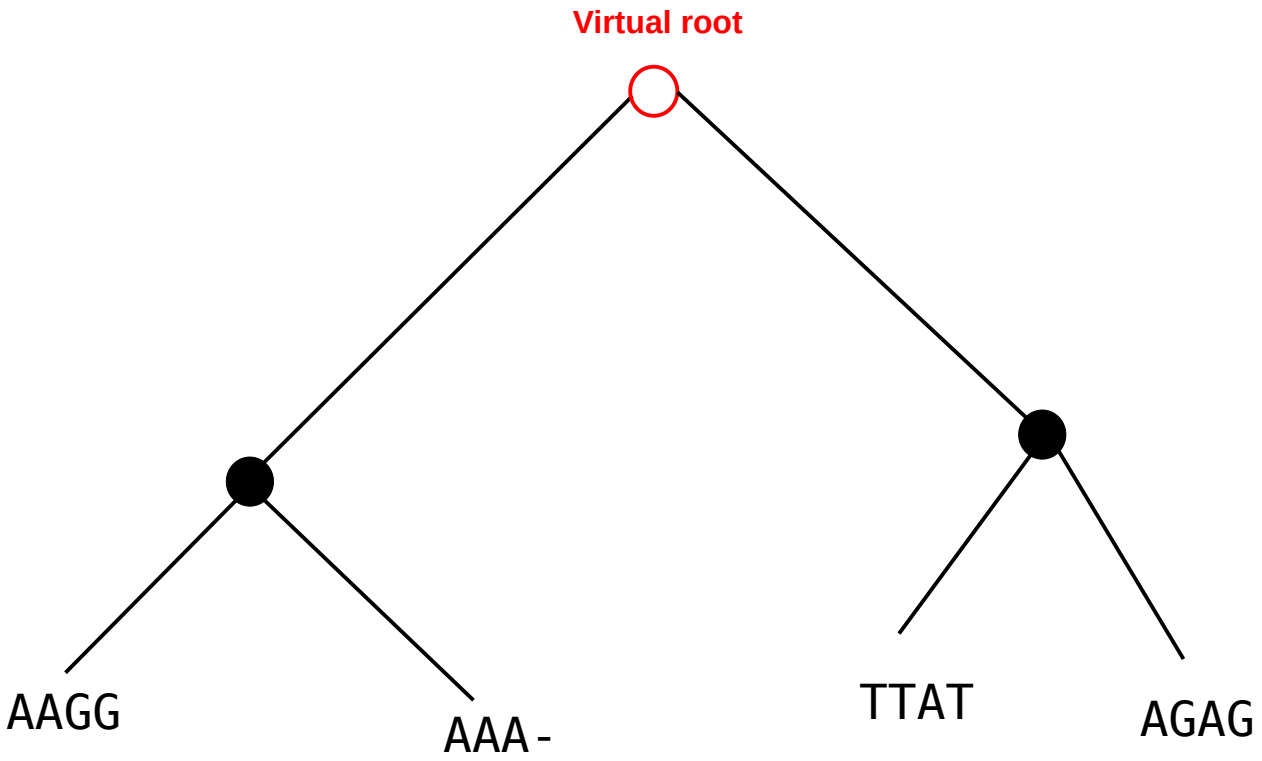
Parsimony

Post-order traversal to compute inner states

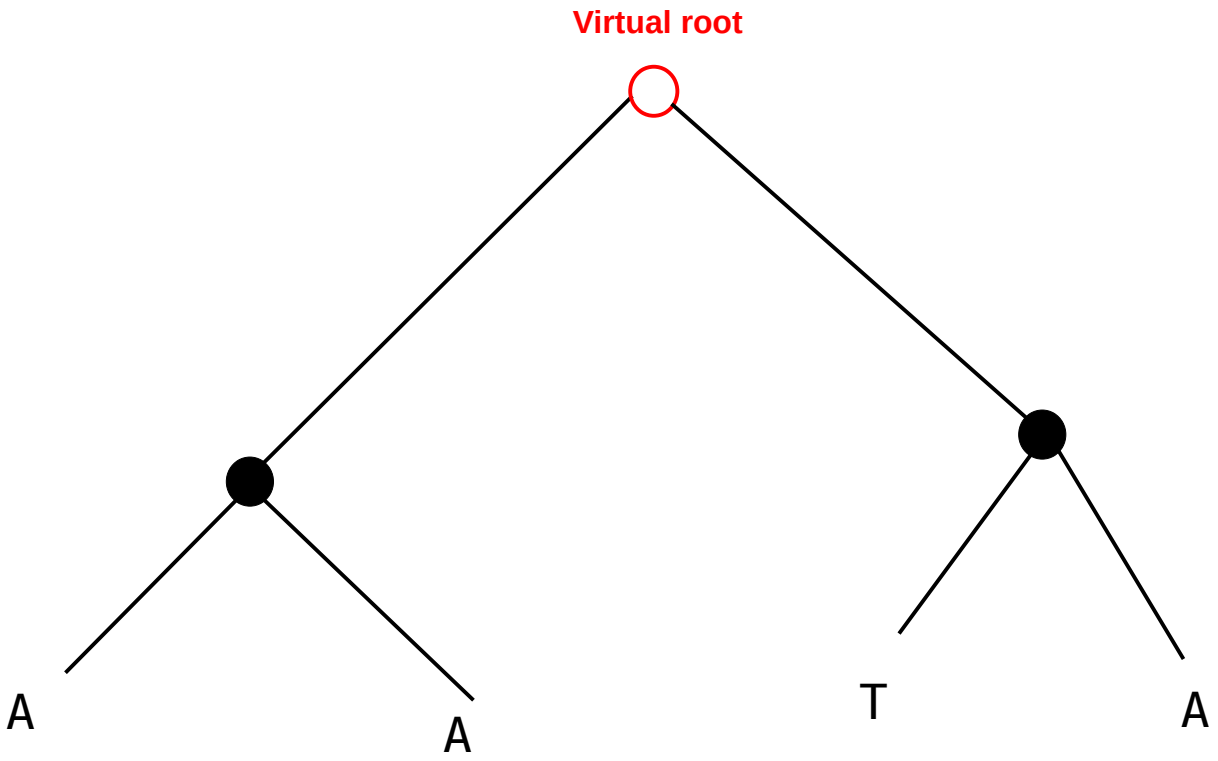


Parsimony

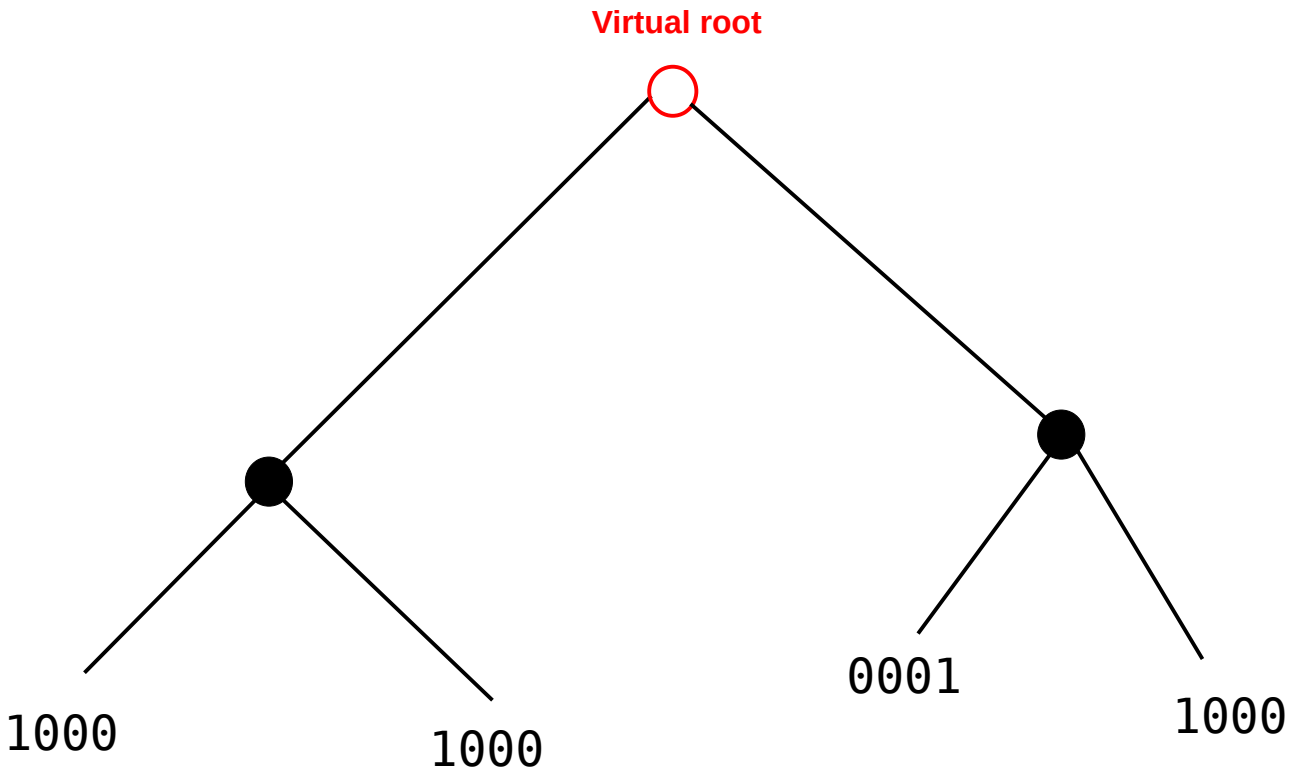
Compute scores on a site-per-site basis
→ we assume that sites evolve independently!



Parsimony

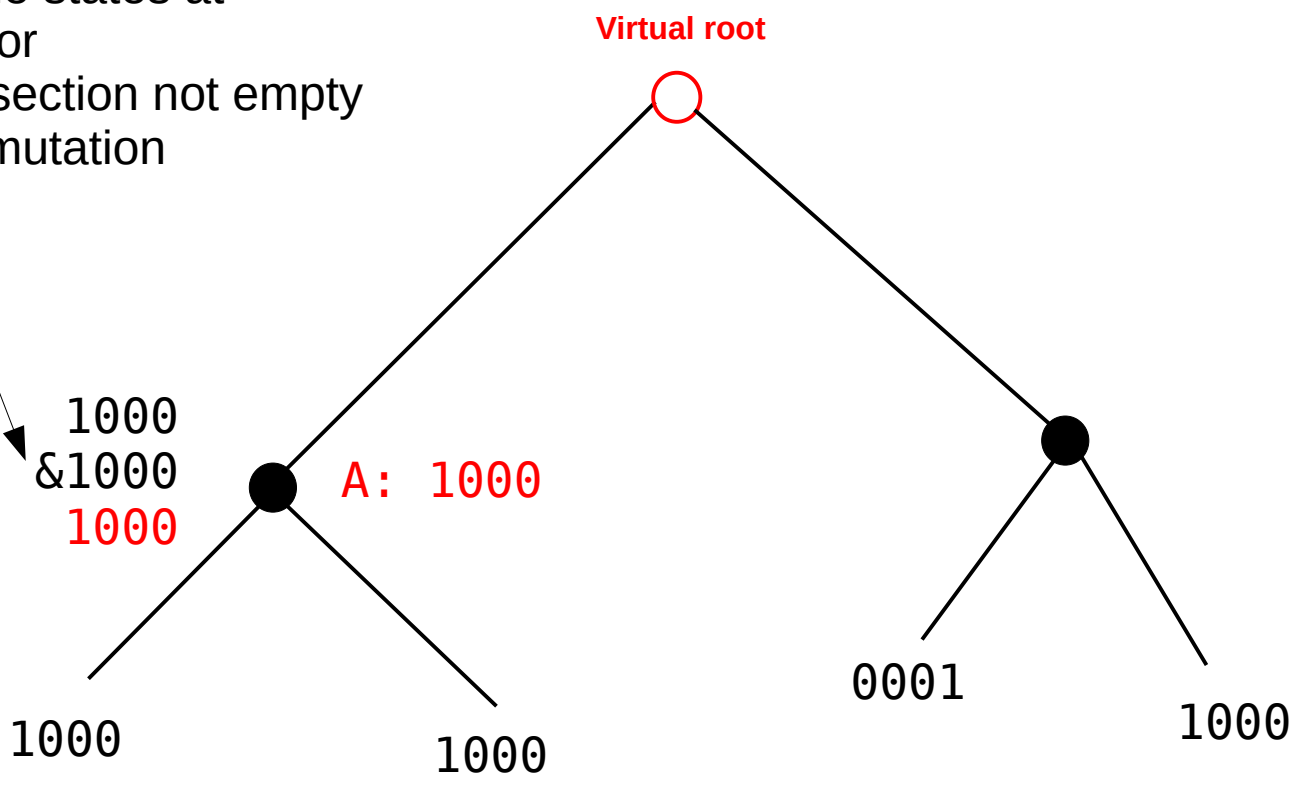


Parsimony

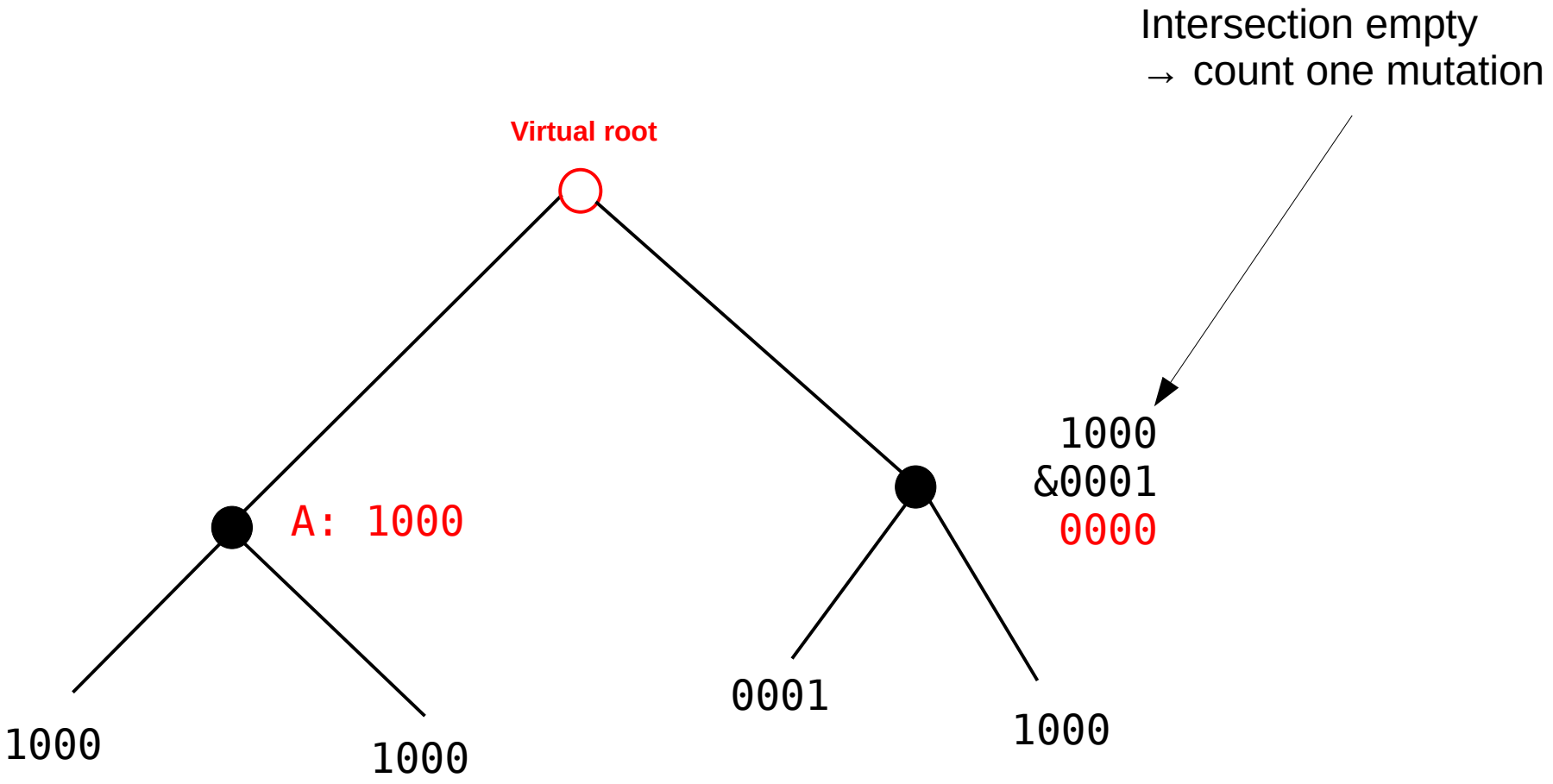


Parsimony

Intersection of sets of possible states at ancestor
If intersection not empty
→ no mutation

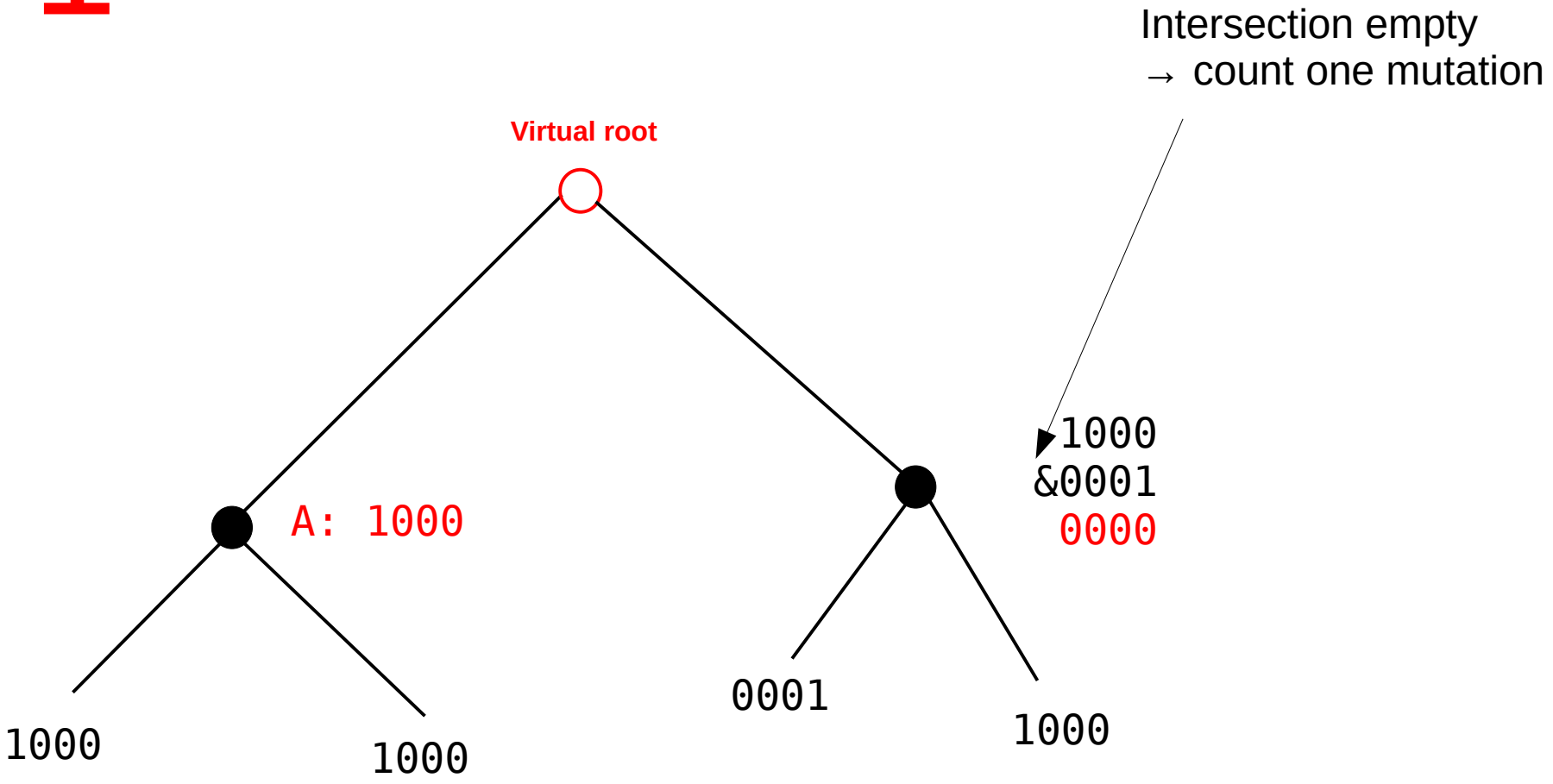


Parsimony


Intersection empty
→ count one mutation
Virtual root
A: 1000
1000
1000
0001
1000
1000
&0001
0000

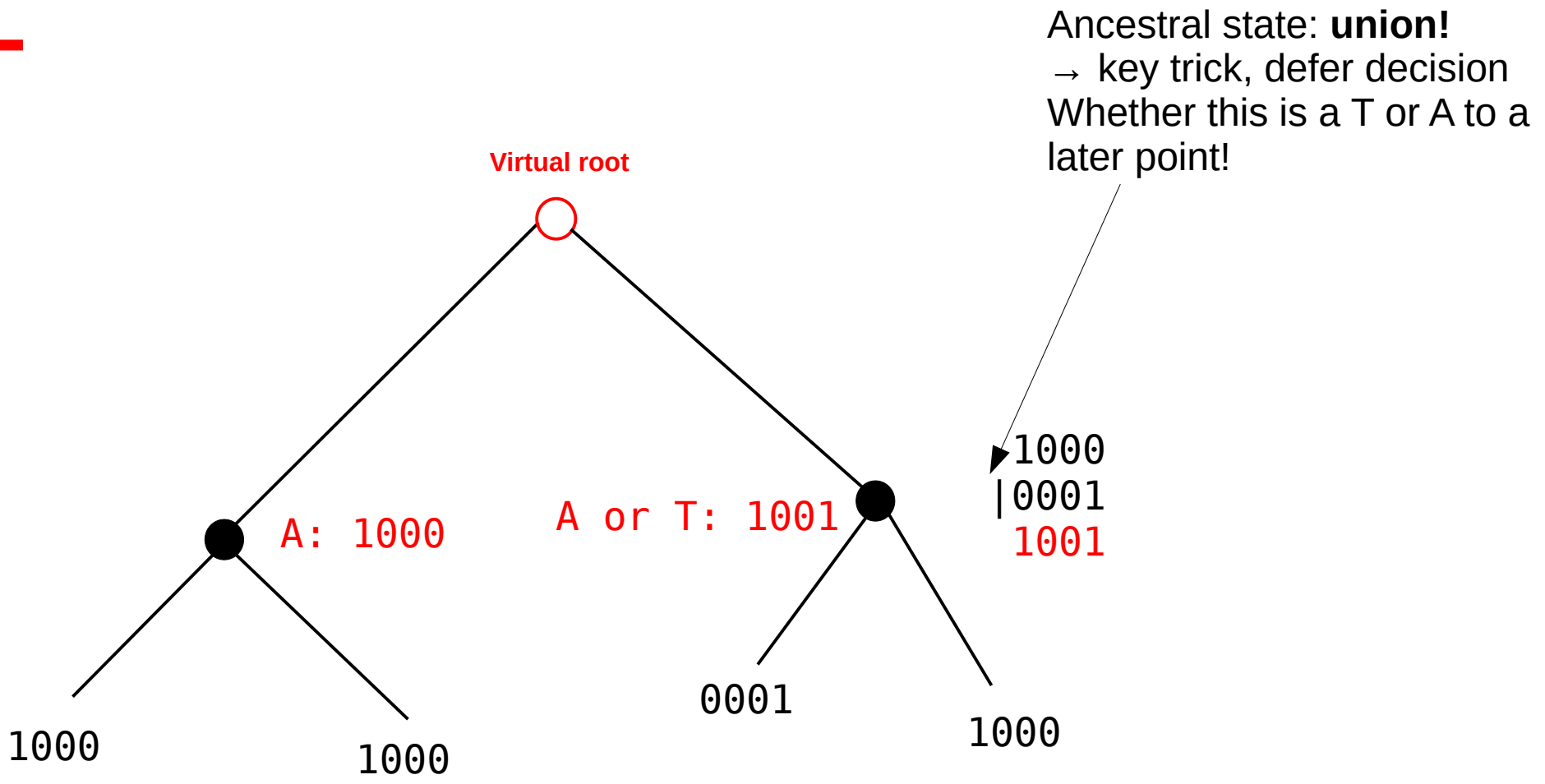
Parsimony

+1


Intersection empty
→ count one mutation
Virtual root
A: 1000
1000
&0001
0000
1000
1000
0001
1000

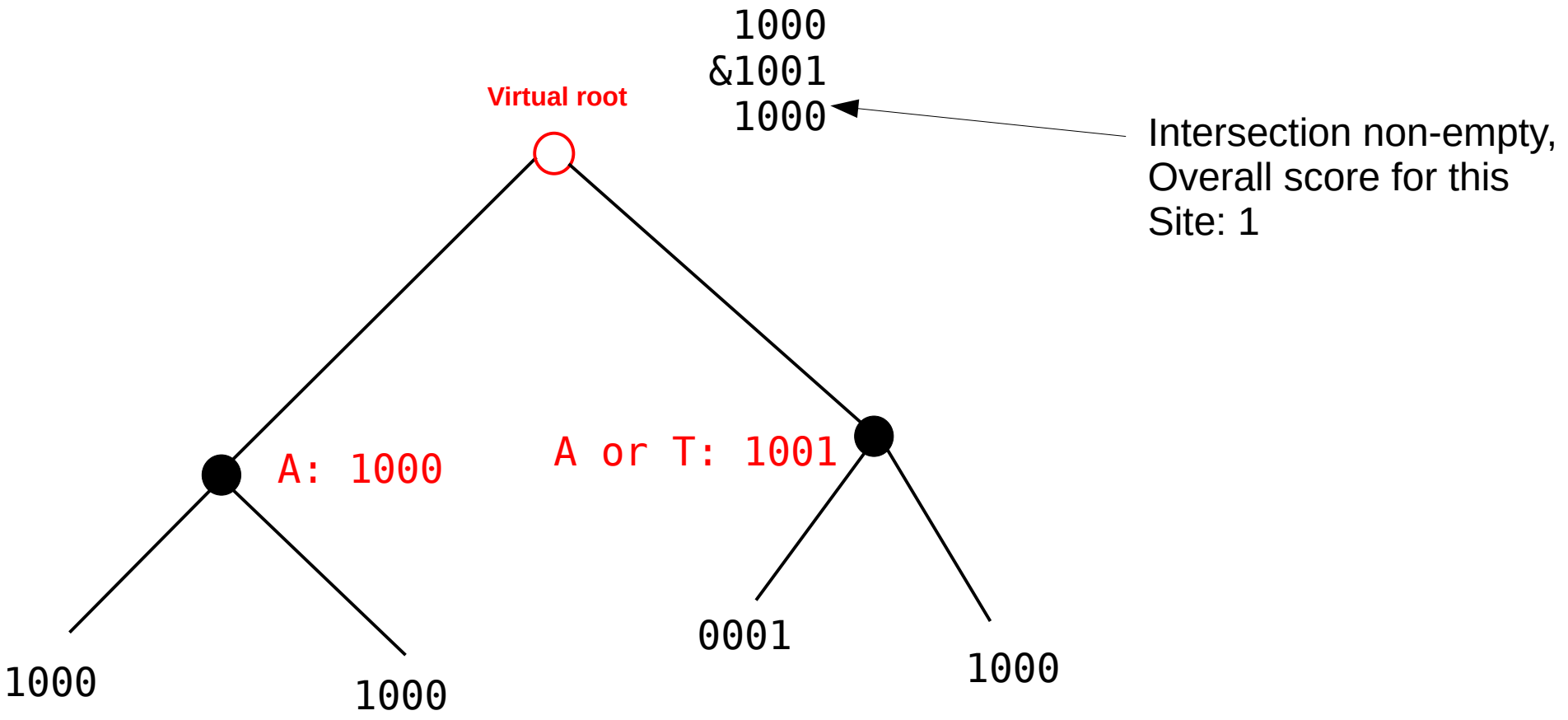
Parsimony

1


Ancestral state: **union!**
→ key trick, defer decision
Whether this is a T or A to a later point!
Virtual root
A: 1000
A or T: 1001
1000
|0001
1001
1000
1000
0001
1000

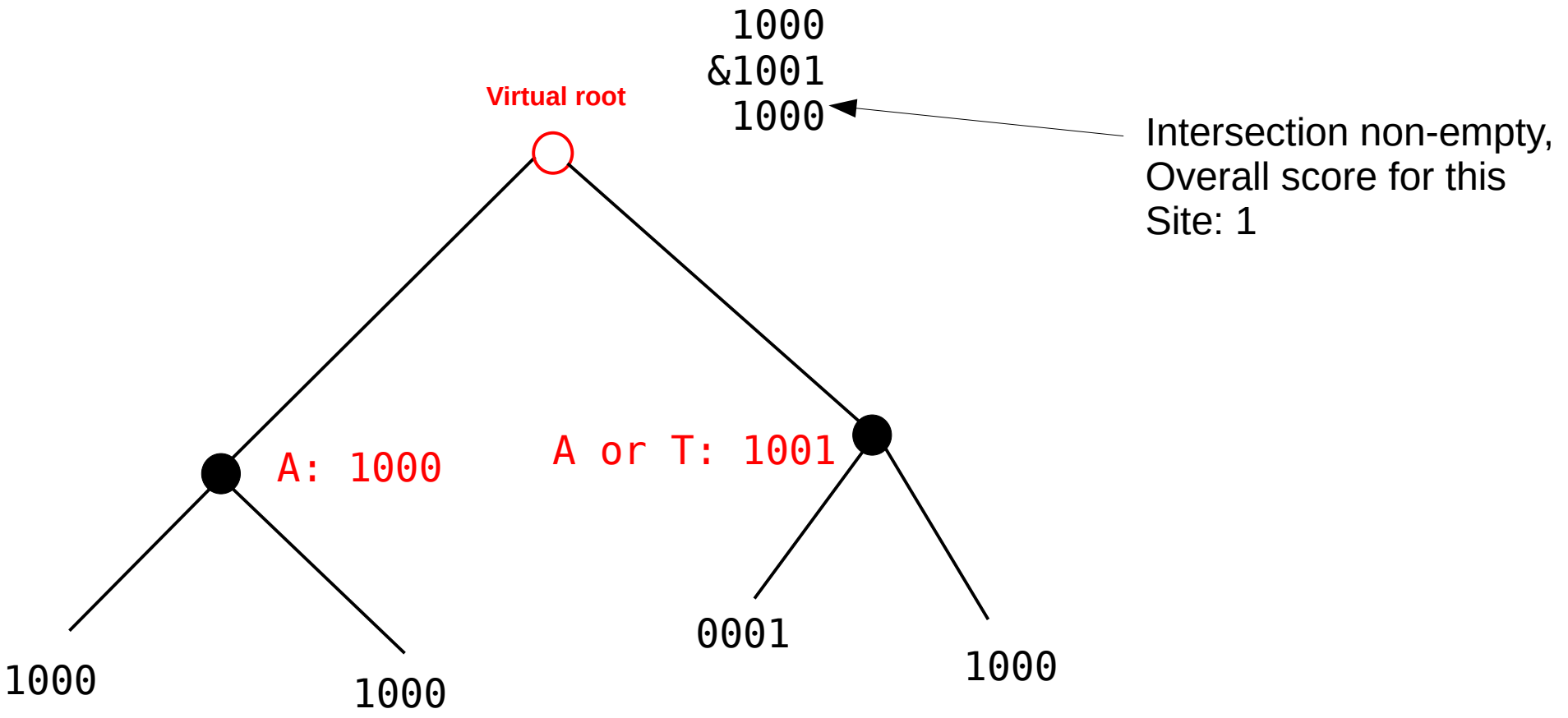
Parsimony

1


1000
&1001
1000
Virtual root
Intersection non-empty,
Overall score for this
Site: 1
A: 1000
A or T: 1001
1000
1000
0001
1000

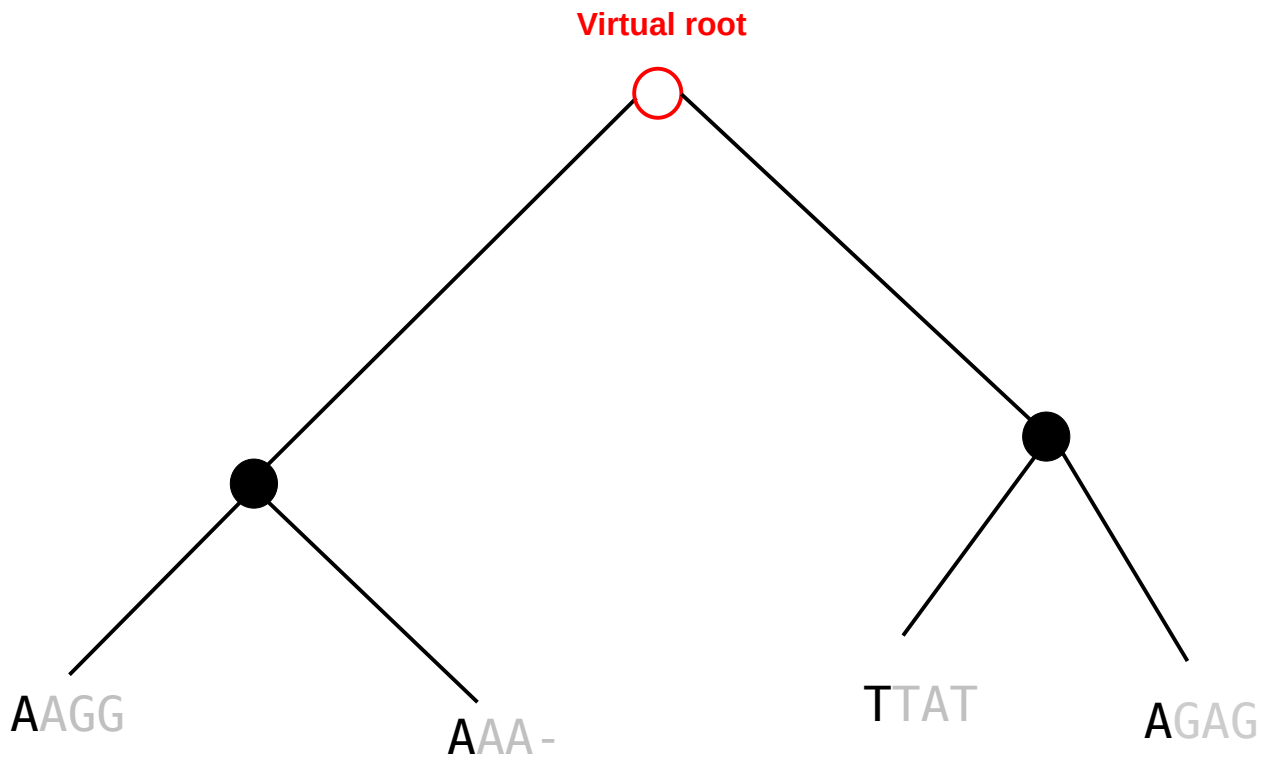
Parsimony

1


1000
&1001
1000
Virtual root
Intersection non-empty,
Overall score for this
Site: 1
A: 1000
A or T: 1001
1000
1000
0001
1000

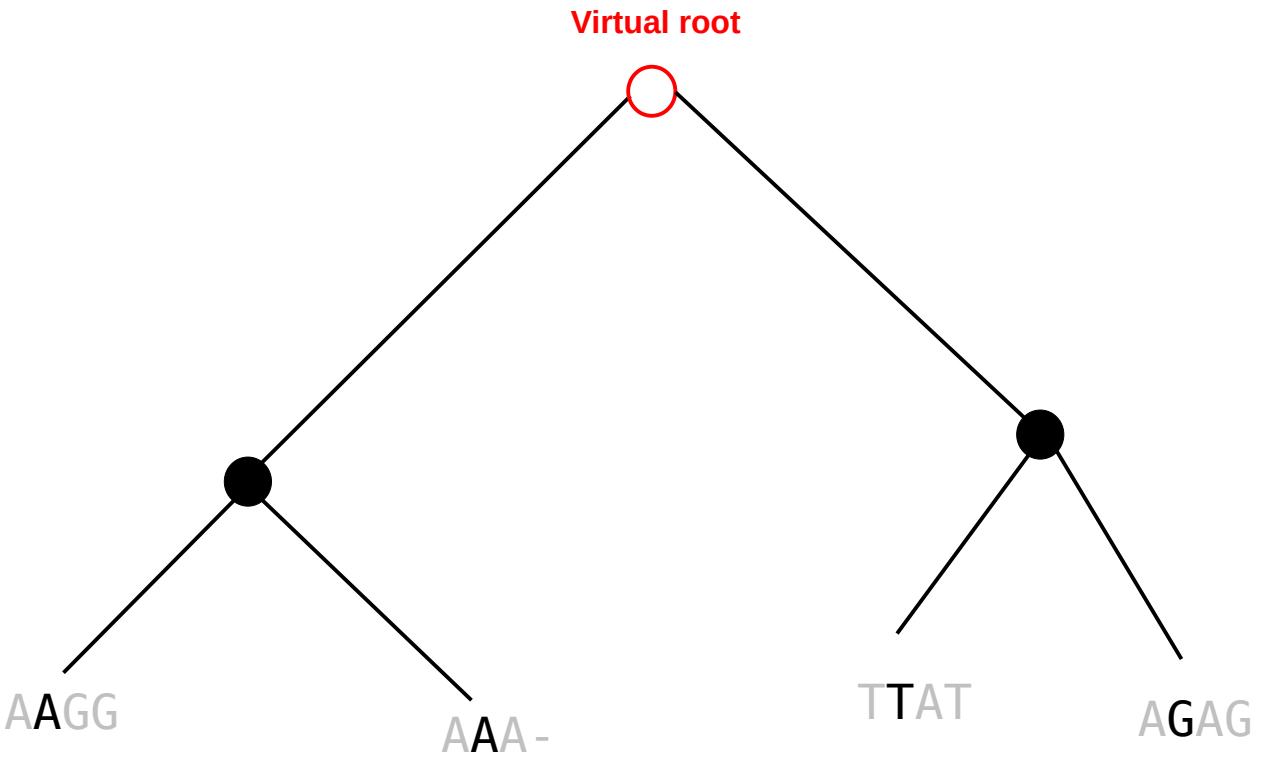
Parsimony

1



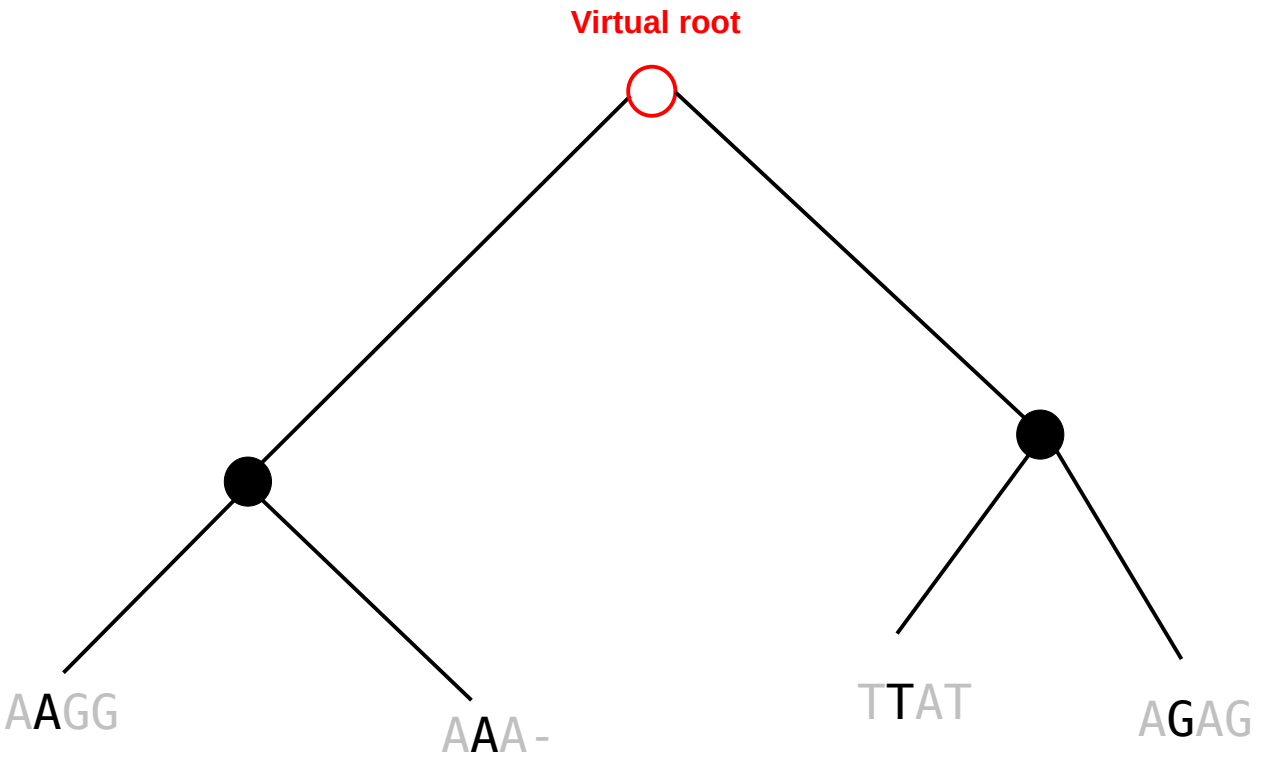
Parsimony

1+?



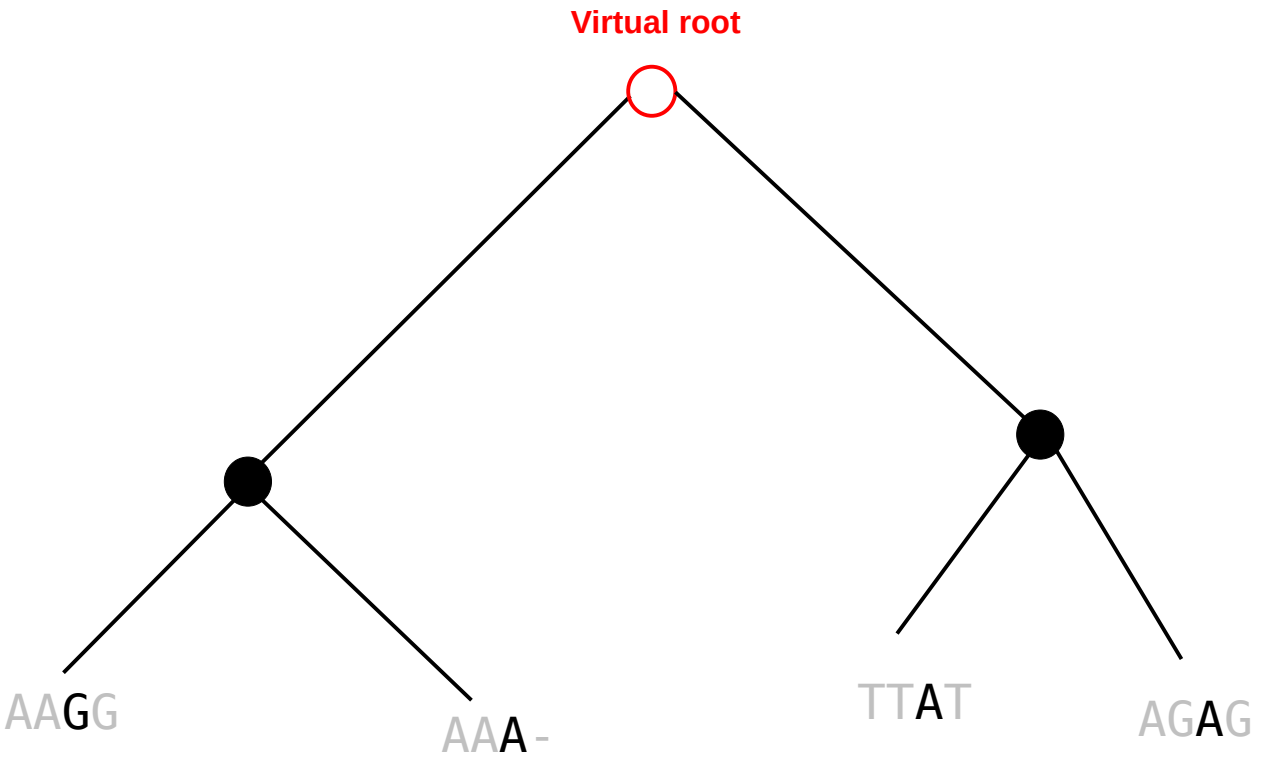
Parsimony

1+2



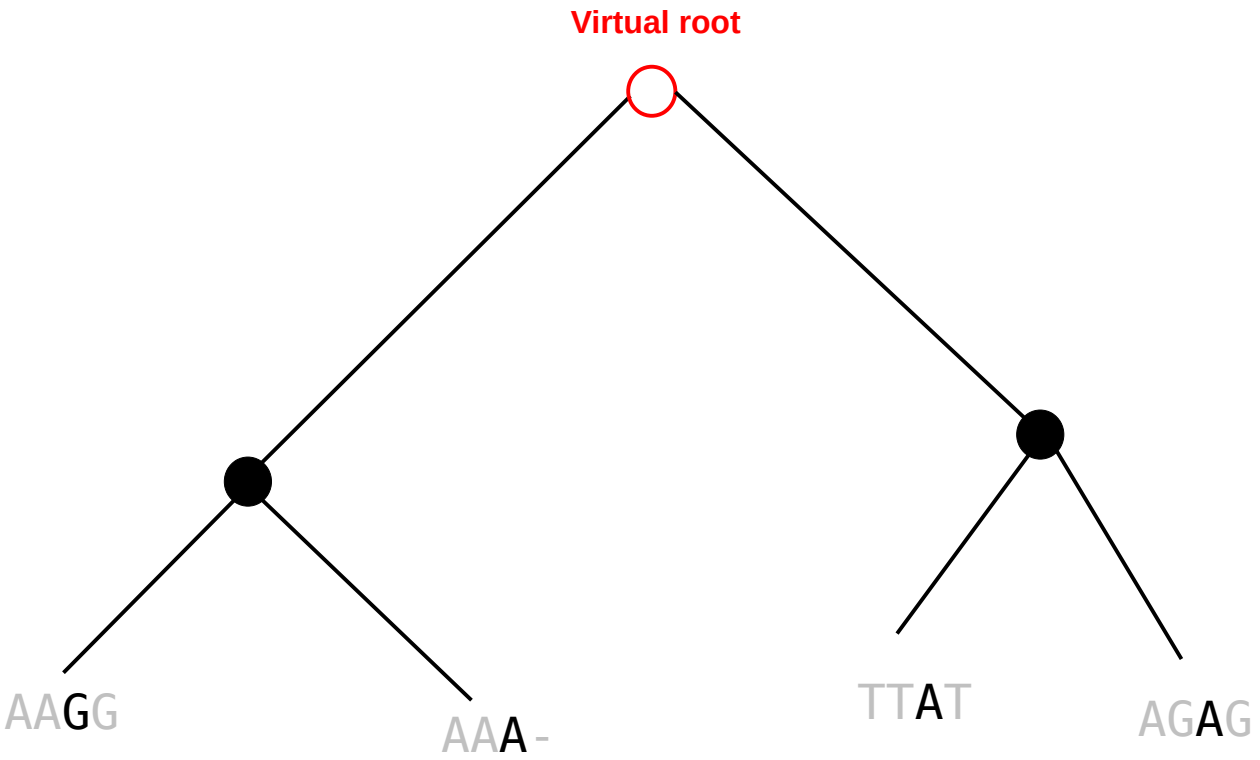
Parsimony

1+2+?



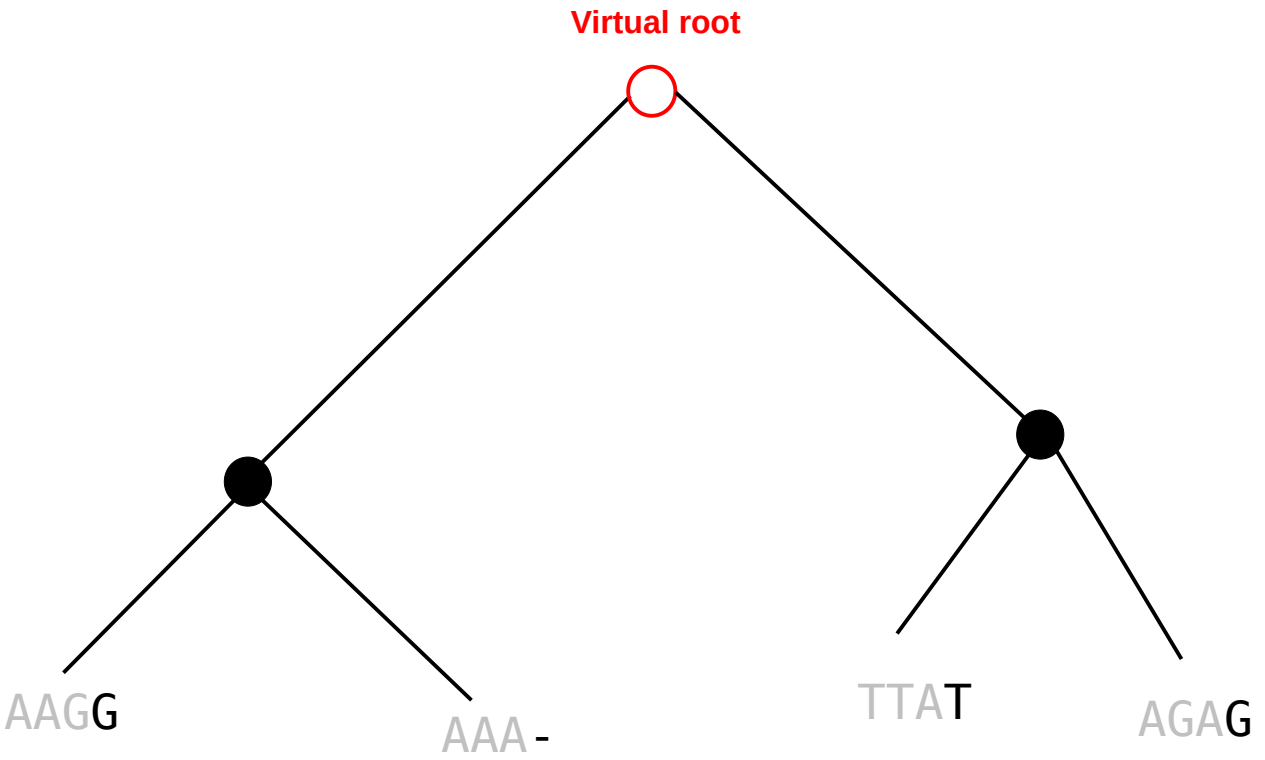
Parsimony

1+2+1



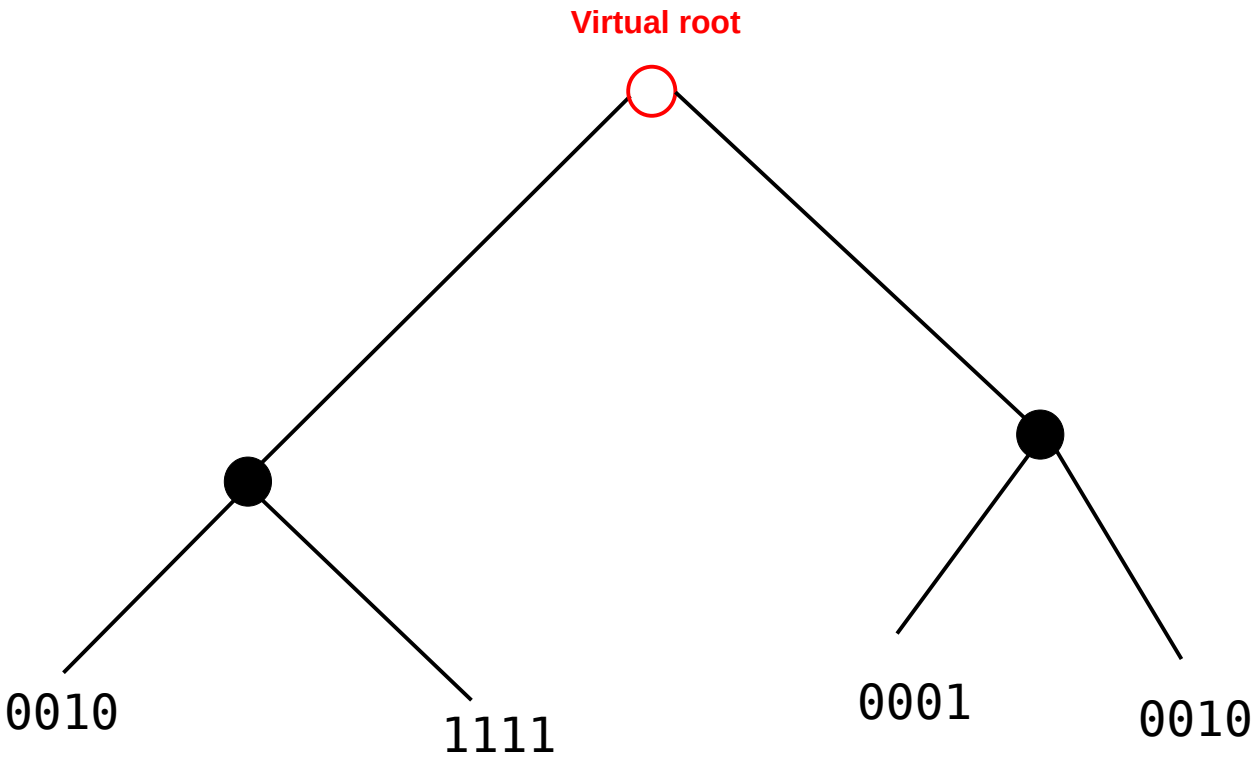
Parsimony

1+2+1



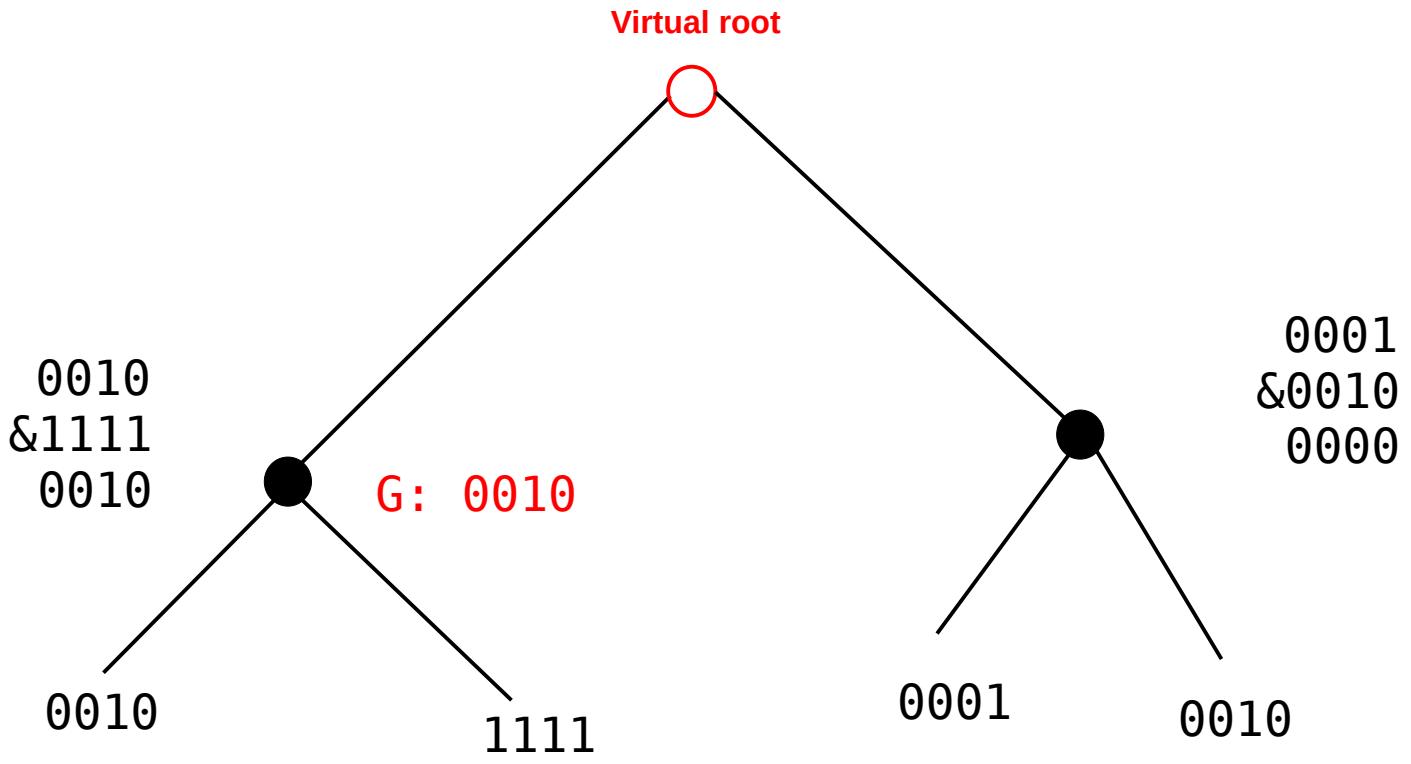
Parsimony

1+2+1



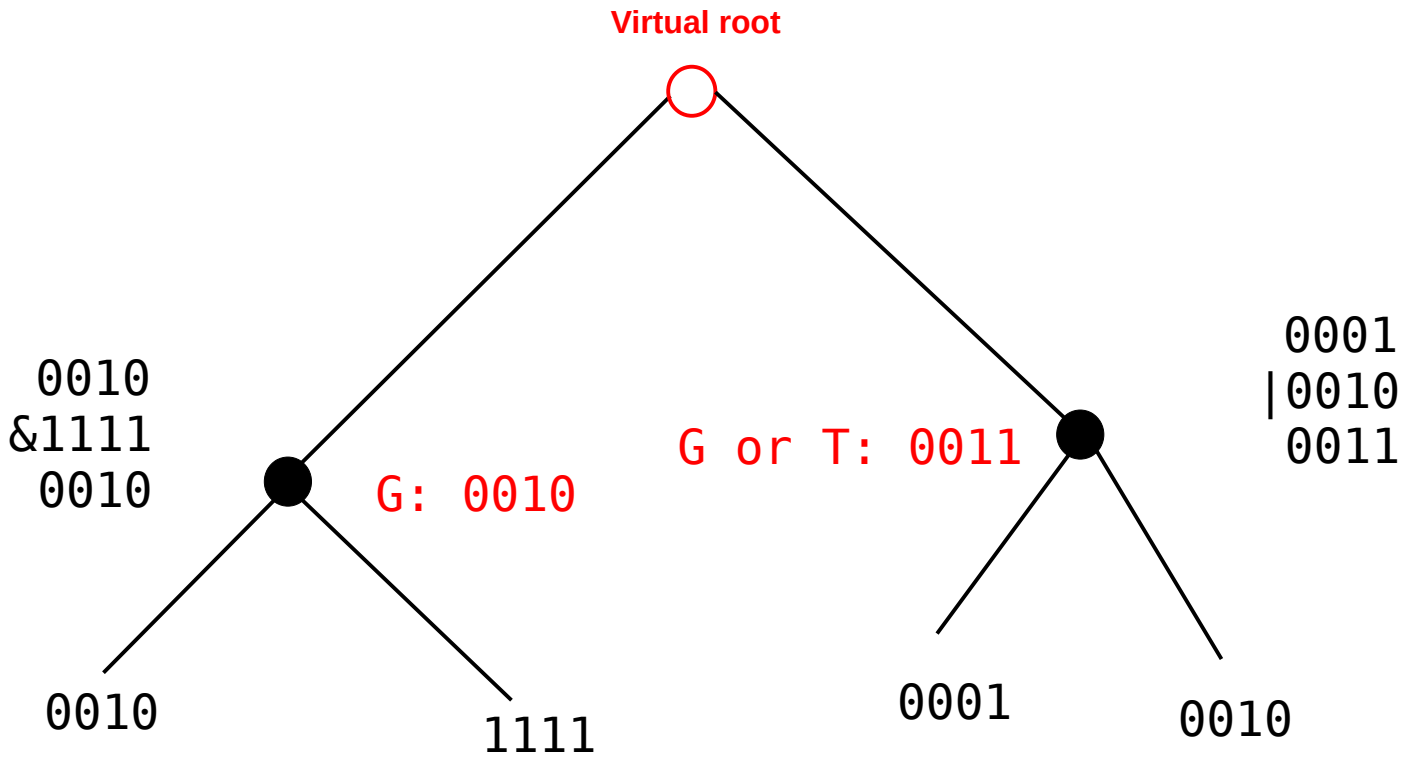
Parsimony

1+2+1



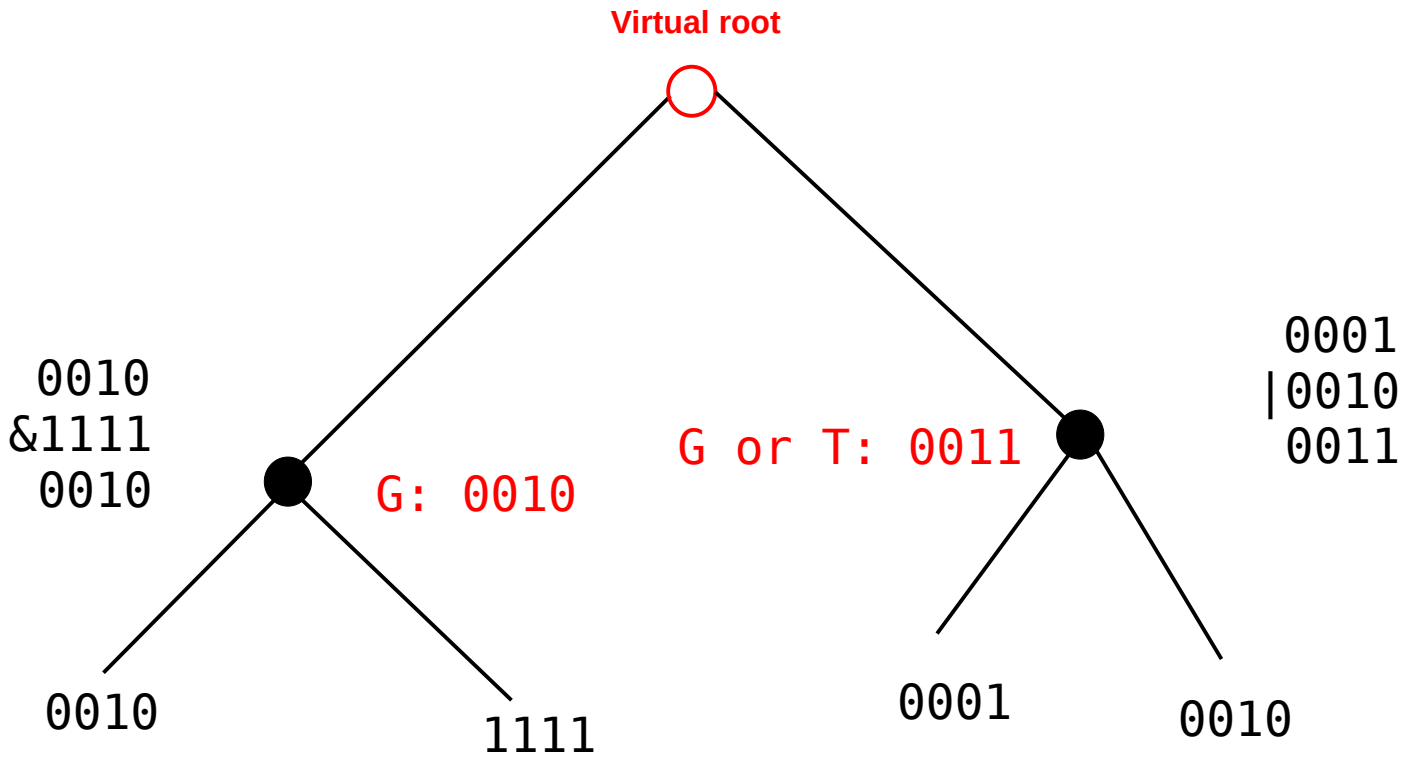
Parsimony

1+2+1+?



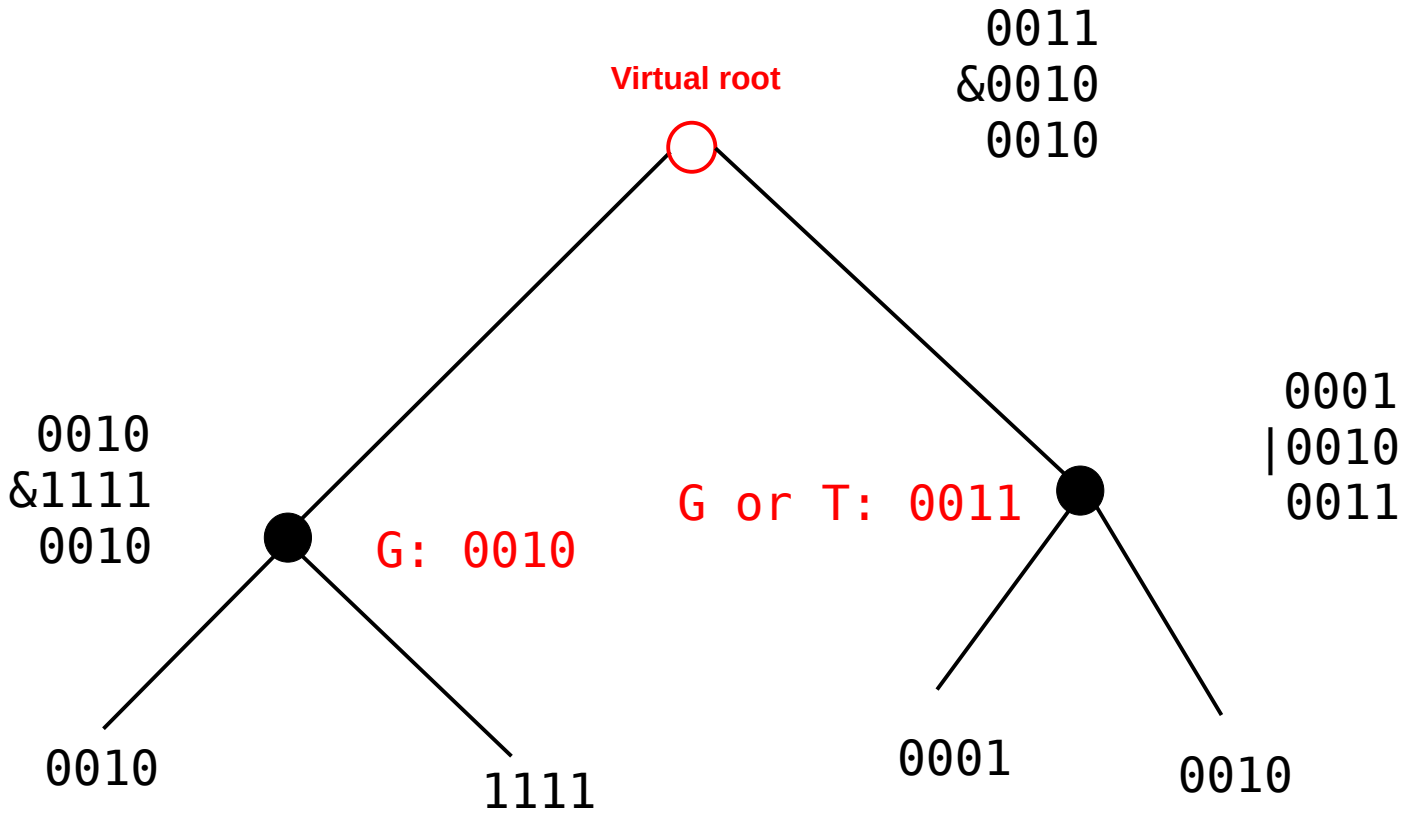
Parsimony

1+2+1+1



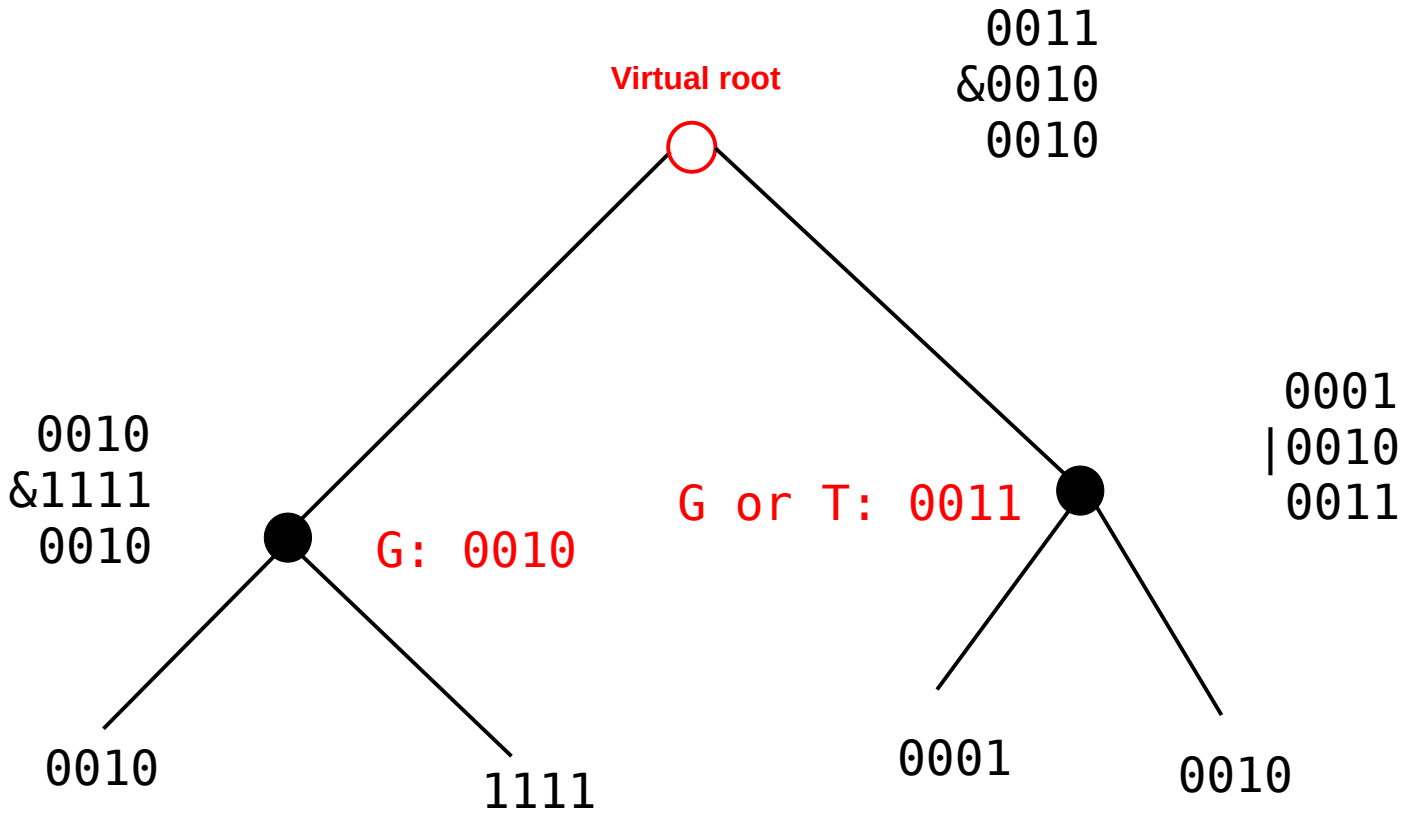
Parsimony

1+2+1+1

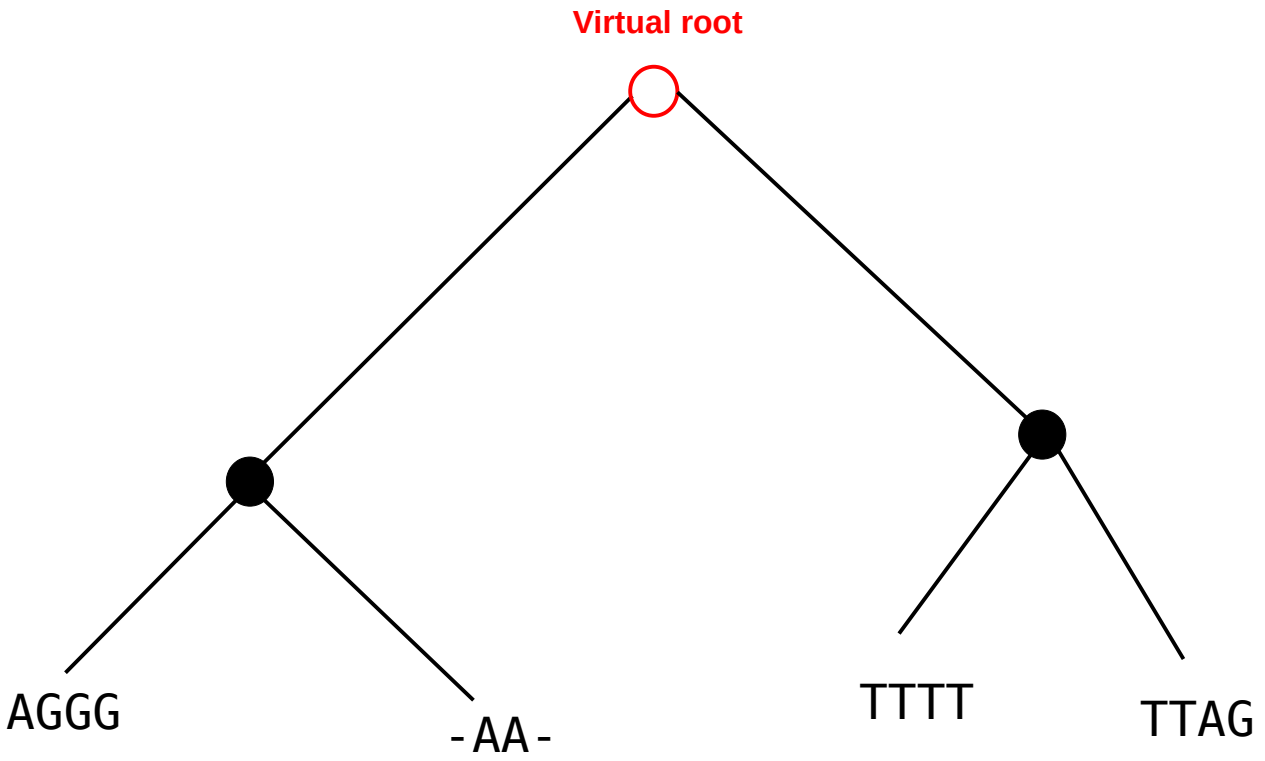


Parsimony

1+2+1+1= 5

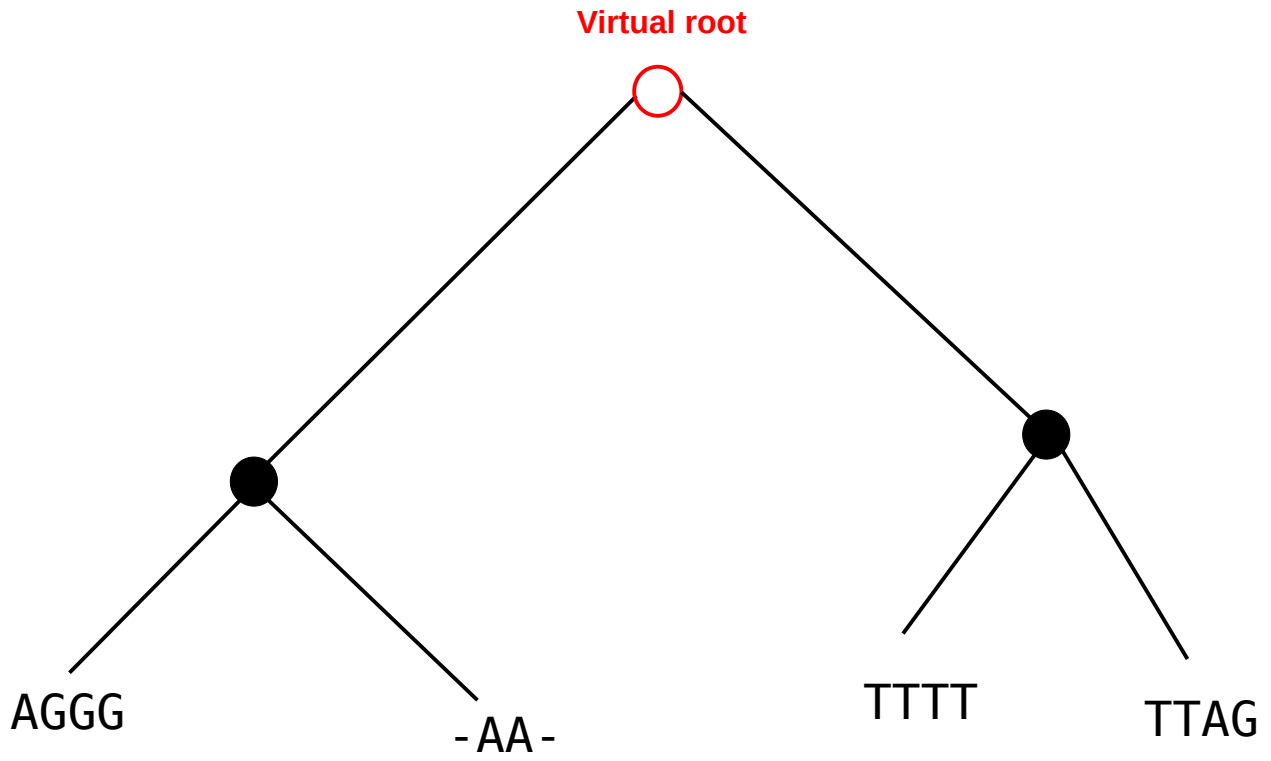


Exercise: What's the parsimony score of this tree?



Exercise: What's the parsimony score of this tree?

1+2+2+1=6



Parsimony

- Time complexity to score one tree

  MSA with *n* taxa and *m* sites

  → *(n-2) * m* calculations; *n-2* is the number of inner nodes of a tree with *n* taxa

  → *O(nm)*, but the constant hidden in *O()* is very small

- Space complexity *DNA* data

  → alignment: *n * m* * 4 bits

  → ancestral nodes: *(n-2) * m* * 4 bits

  → score counter: *(n-2)* * 32 bits

  → space complexity *O(nm)*, but the constant hidden in *O()* is very small

- **Maximum Likelihood**: same time & space complexity, but constants much, much larger!

Parsimony Implementation Notes

- Intersections and Unions can be implemented efficiently at the bit-level
- 4 bits for one DNA character (remember, ambiguous character encoding)
- Plain implementation: 32 bits
- `SSE3` vector intrinsics: 128 bits
- `AVX` vector intrinsics: 256 bits
- `Parsimonator` program (www.exelixis-lab.org/software.html)
  - → uses SSE3 and AVX intrinsics
  - → I will show a demo now
  - → Implements simple search algorithm
  - → probably fastest available open-source parsimony implementation

Parsimony Implementation Notes

- Without going into the details:
- In the `parsimonator` implementation we need to compute a so-called population count (`popcount`) that computes the number of bits (# mutations) that are set to `1` in a 32-, 128-, or 256-bit word
- `popcount` is a very important operation
- There are various fancy bit-twisting implementations for fast `popcounts`
- In fact, this operation is so important that modern `x86` architectures have a dedicated HW-based `popcount`
- You can use it in C code via `__builtin_popcount(x)`

Parsimony Implementation Notes

- Why did we write `parsimonator`?
- A paper was published that claimed to have achieved a FPGA-based acceleration of the parsimony function of up to factor 10,000
- Remember: the speedup is defined as *T(1)/T(N)*, where *T(1)* is the fastest available sequential implementation/algorithm!
- Compared to Parsimonator (AVX version), the corresponding FPGA design achieved a speedup of up to 10, only!
- N. Alachiotis, A. Stamatakis: "FPGA Acceleration of the Phylogenetic Parsimony Kernel**?**", *FPL 2011*.

Tree Search Algorithms

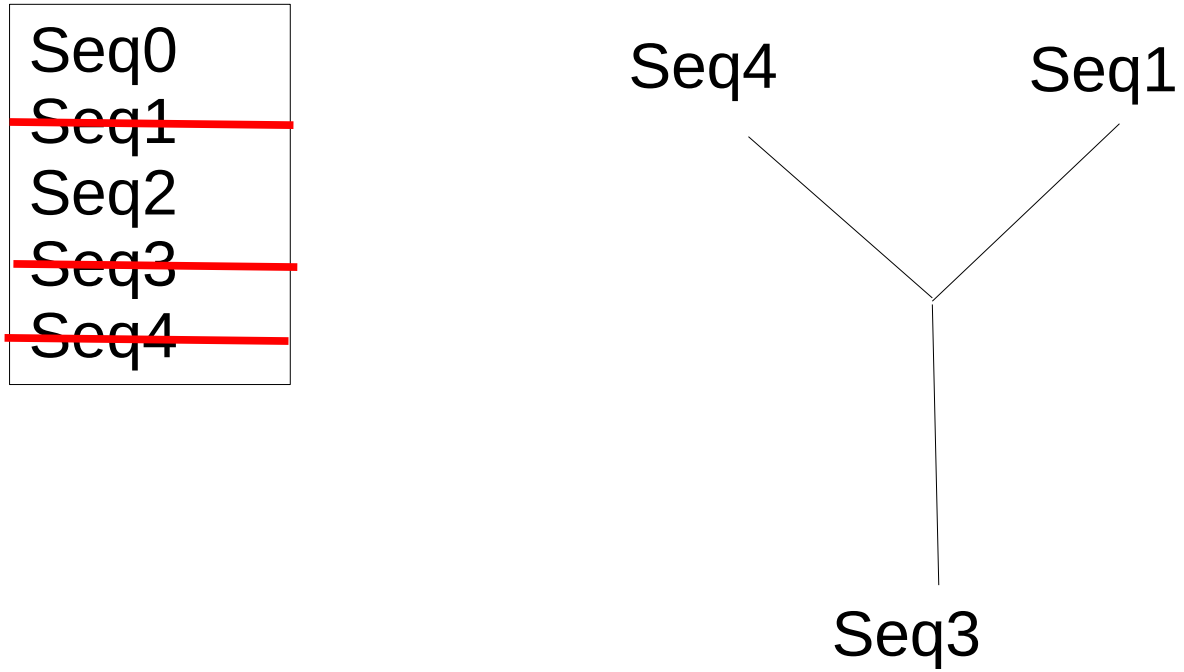
- How do we obtain an initial starting tree with *n* taxa → comprehensive tree
  - → NJ or UPGMA tree
  - → random tree
  - → stepwise addition algorithm
- How do we change such a comprehensive tree to improve its score?

  Scores can be improved with optimality criteria: least squares, minimum evolution, parsimony, maximum likelihood

Building a Random Tree

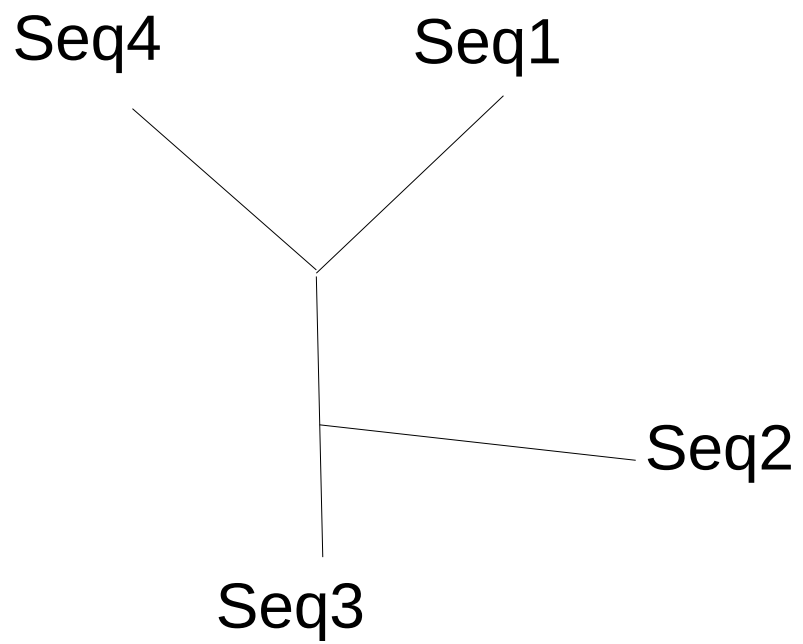
Seq0
Seq1
Seq2
Seq3
Seq4

Building a Random Tree

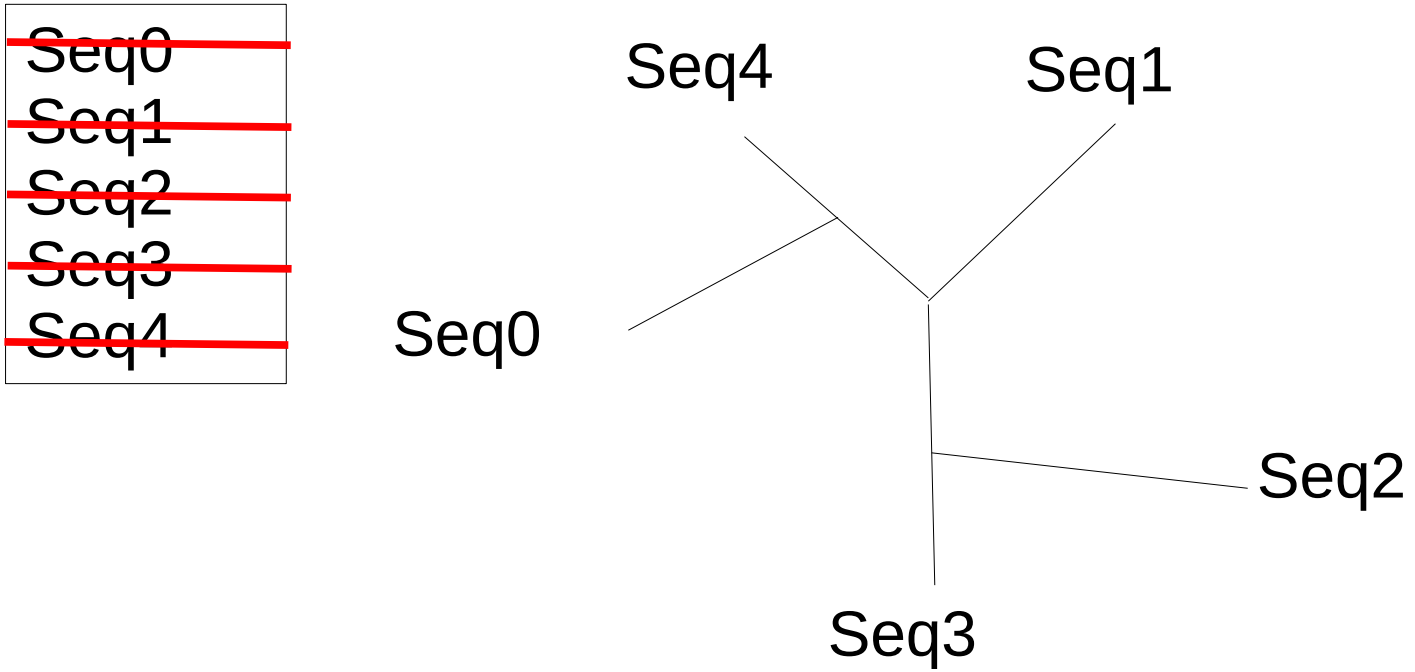


Building a Random Tree

Seq0
~~Seq1~~
~~Seq2~~
~~Seq3~~
~~Seq4~~



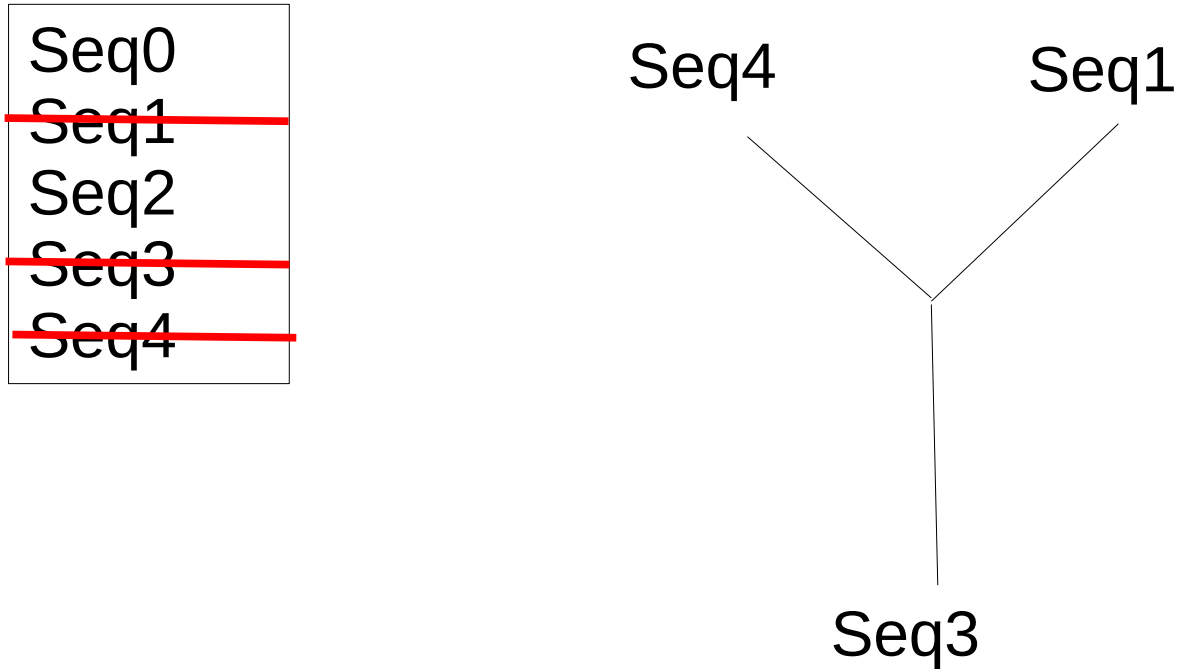
Building a Random Tree



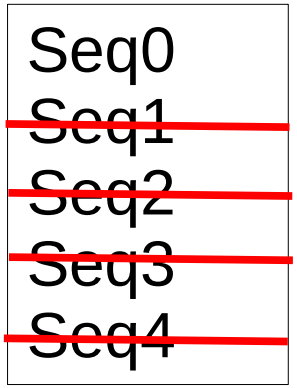
Randomized Stepwise Addition Order Algorithm

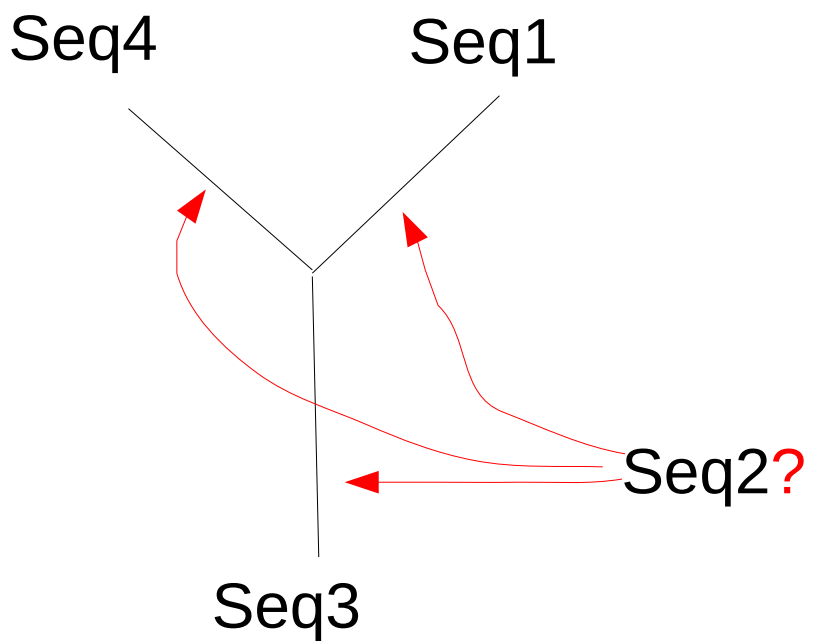
Seq0
Seq1
Seq2
Seq3
Seq4

Randmoized Stepwise Addition Order Algorithm

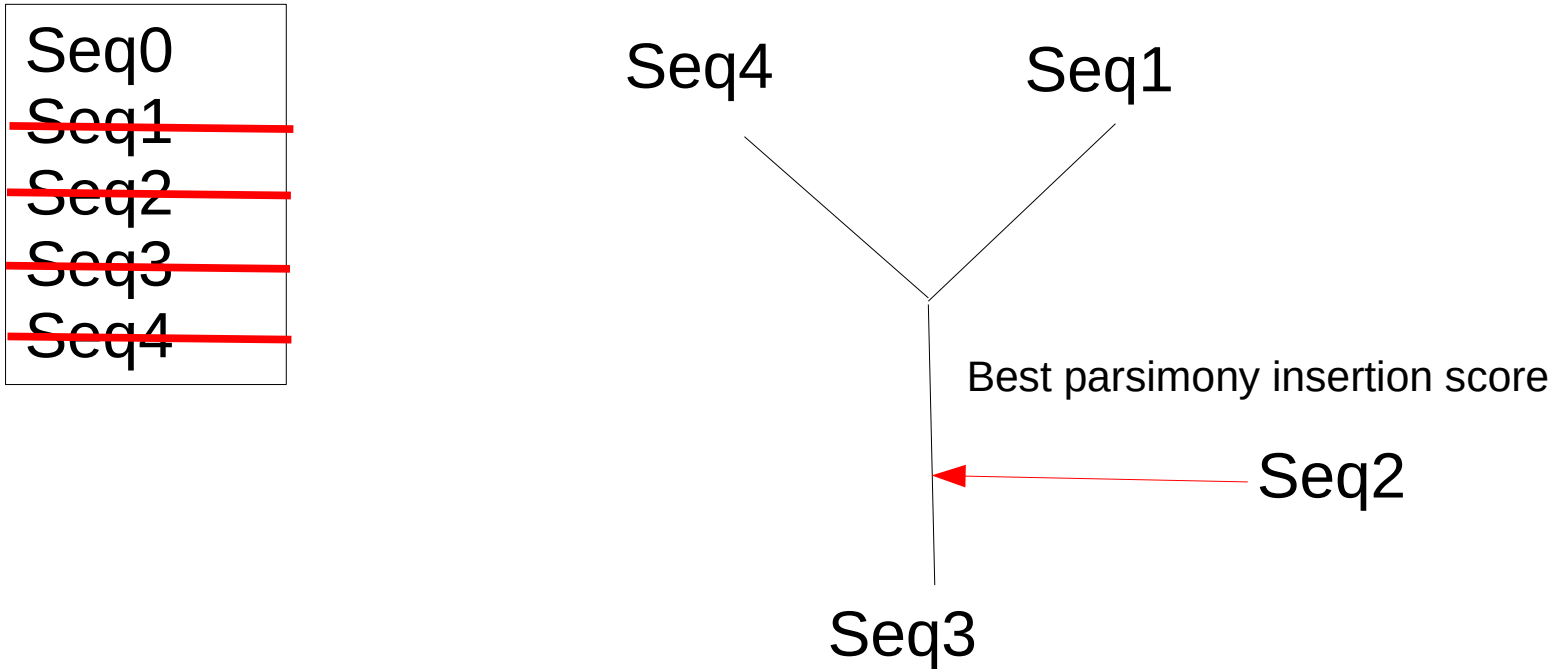


Randomized Stepwise Addition Order Algorithm



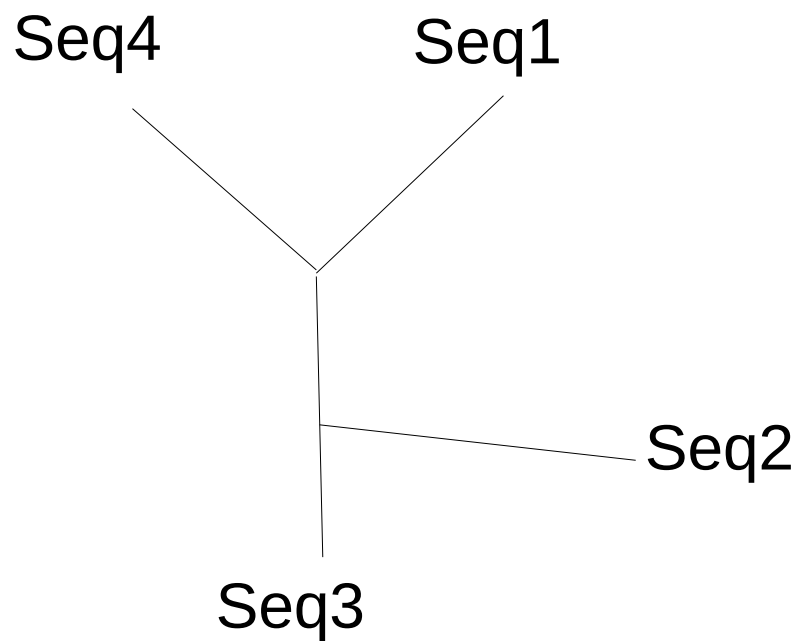


Randomized Stepwise Addition Order Algorithm

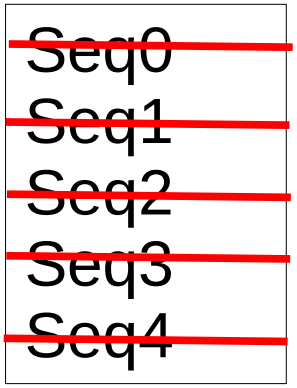


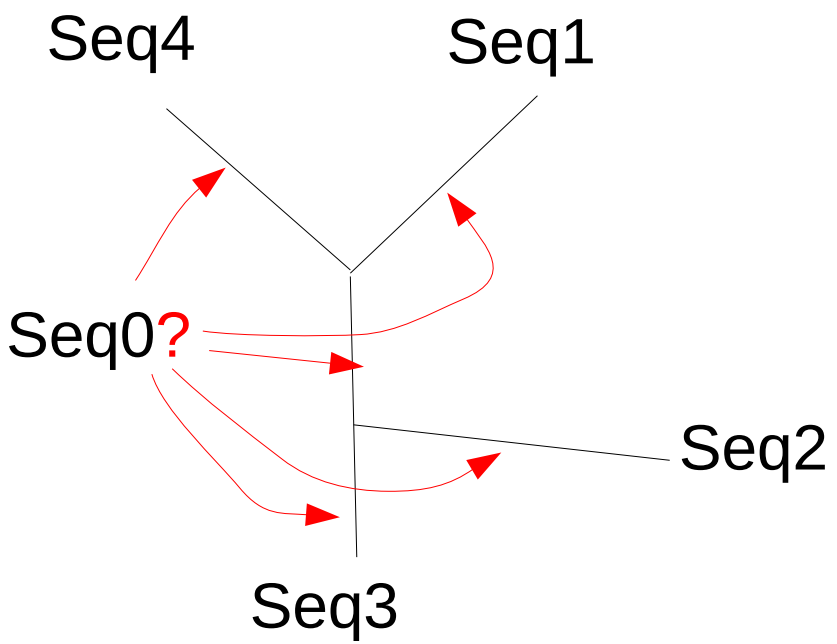
Randomized Stepwise Addition Order Algorithm

Seq0
~~Seq1~~
~~Seq2~~
~~Seq3~~
~~Seq4~~

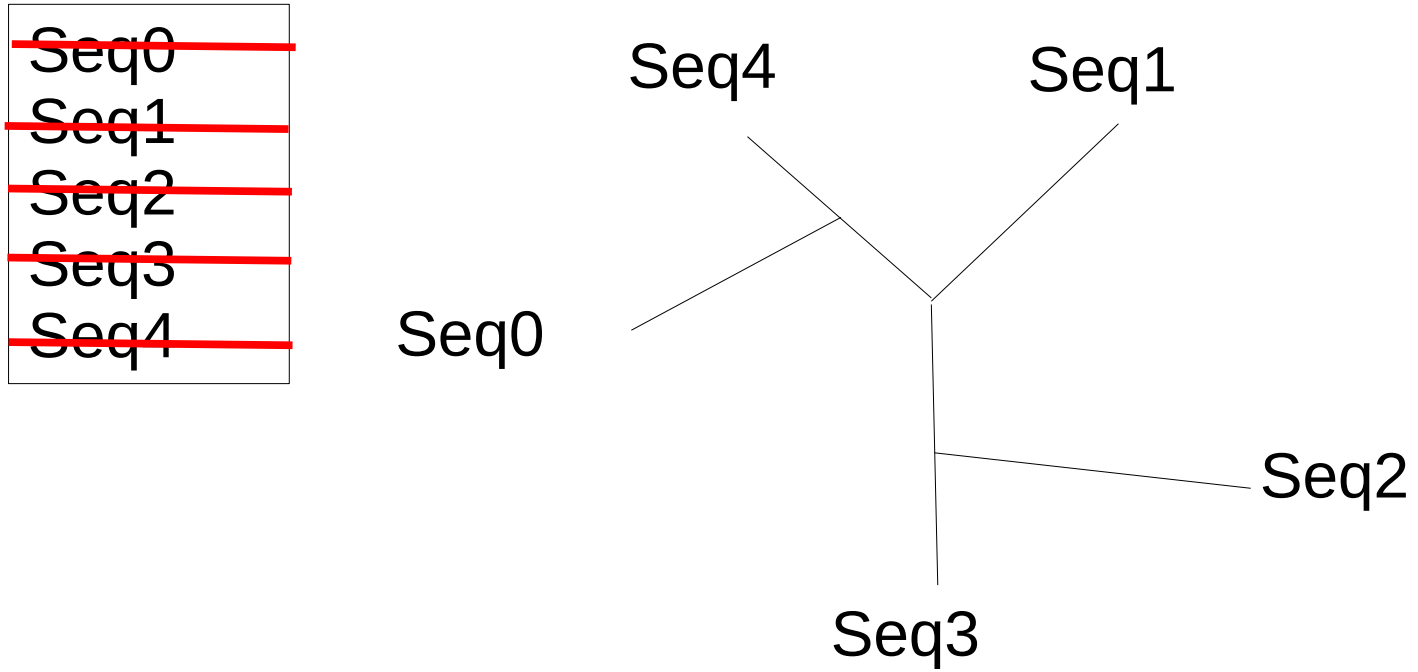


Randomized Stepwise Addition Order Algorithm

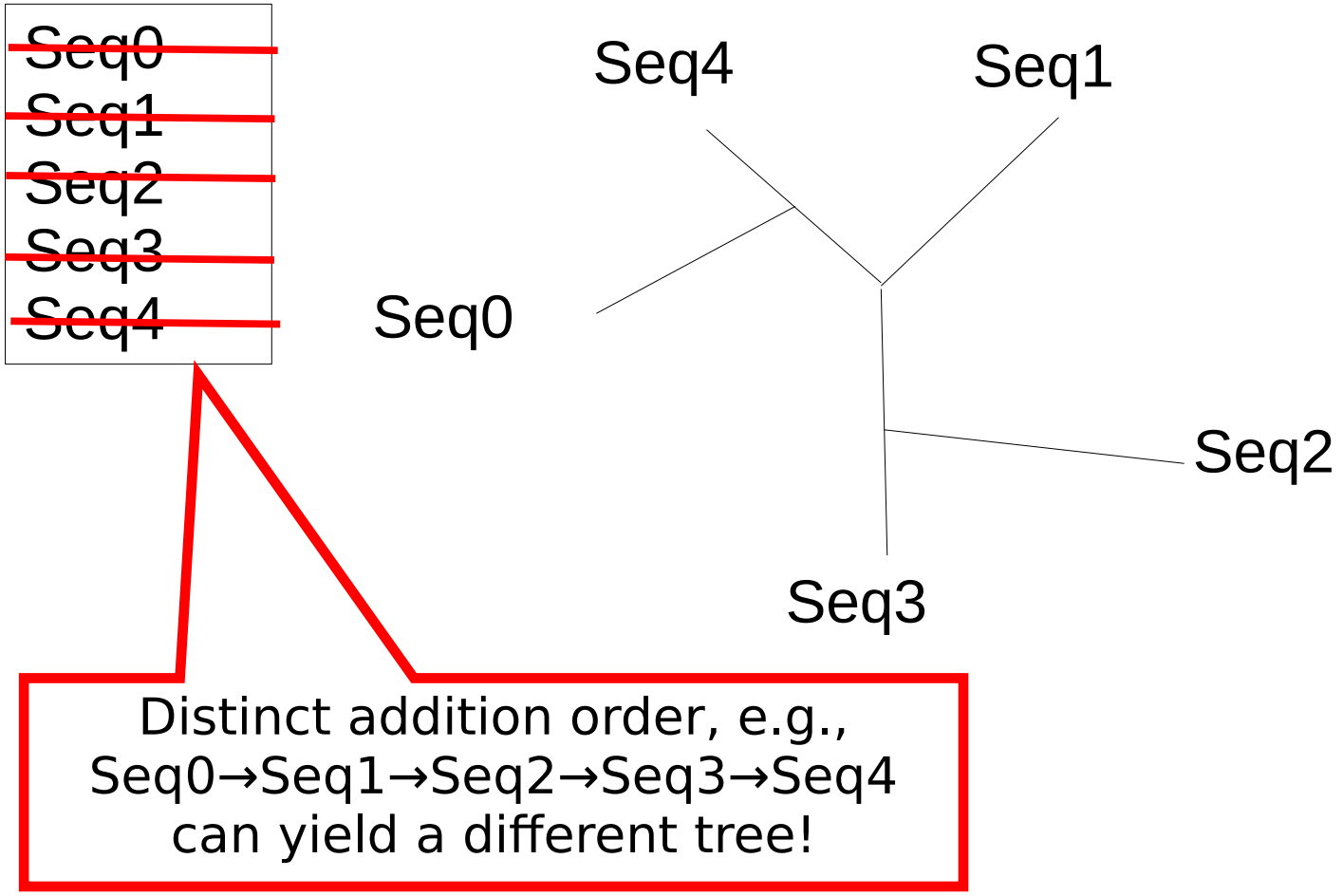




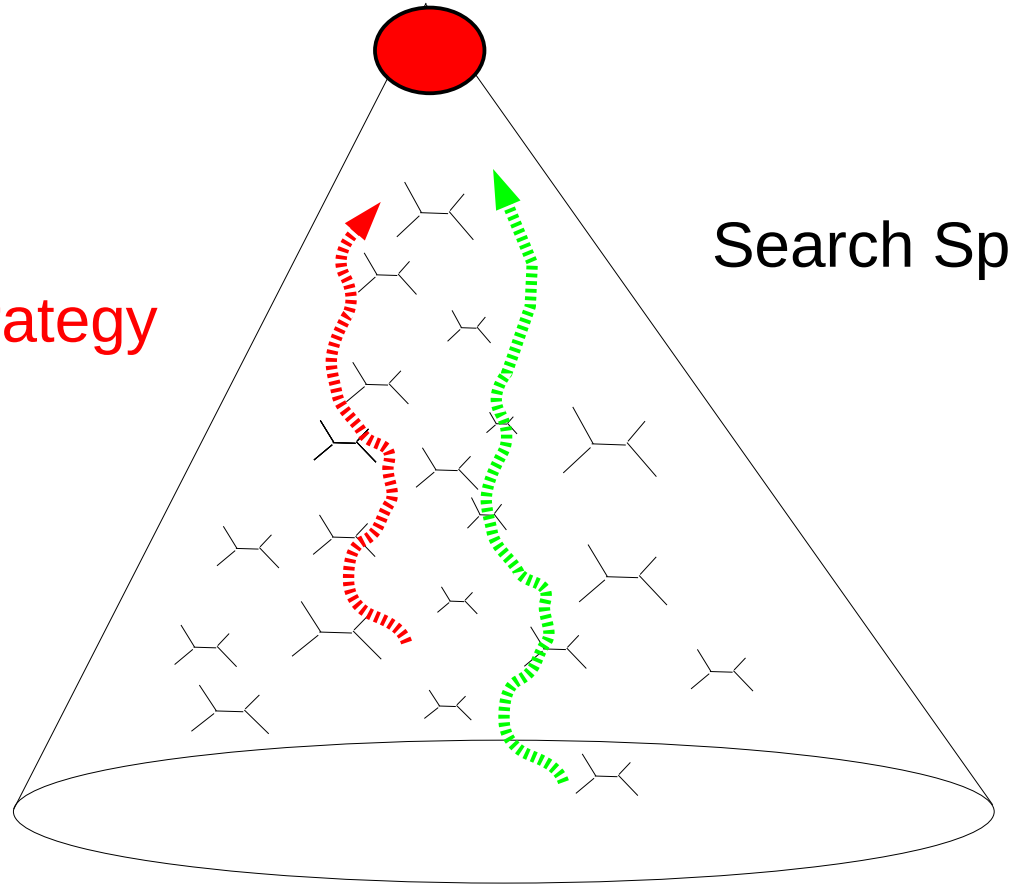
Randomized Stepwise Addition Order Algorithm



Randomized Stepwise Addition Order Algorithm



Why are distinct Starting Trees useful?



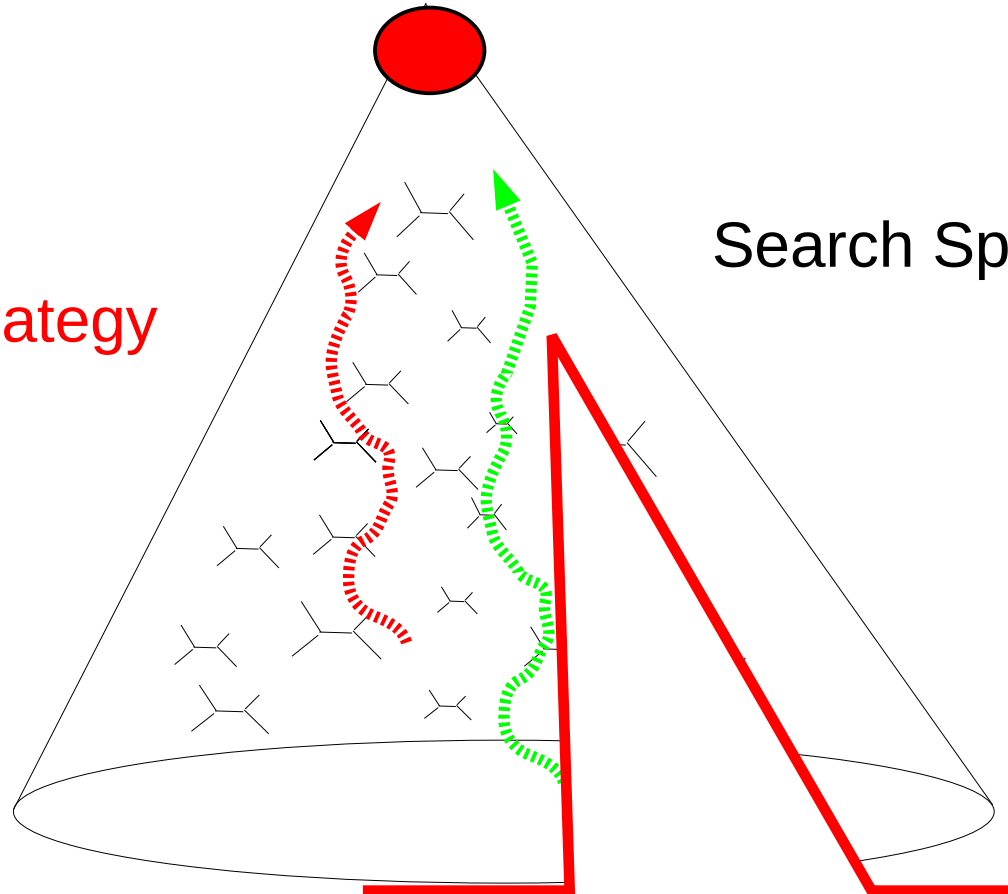
Search Strategy

Search Space

good

bad

Why are distinct Starting Trees useful?



Search Strategy

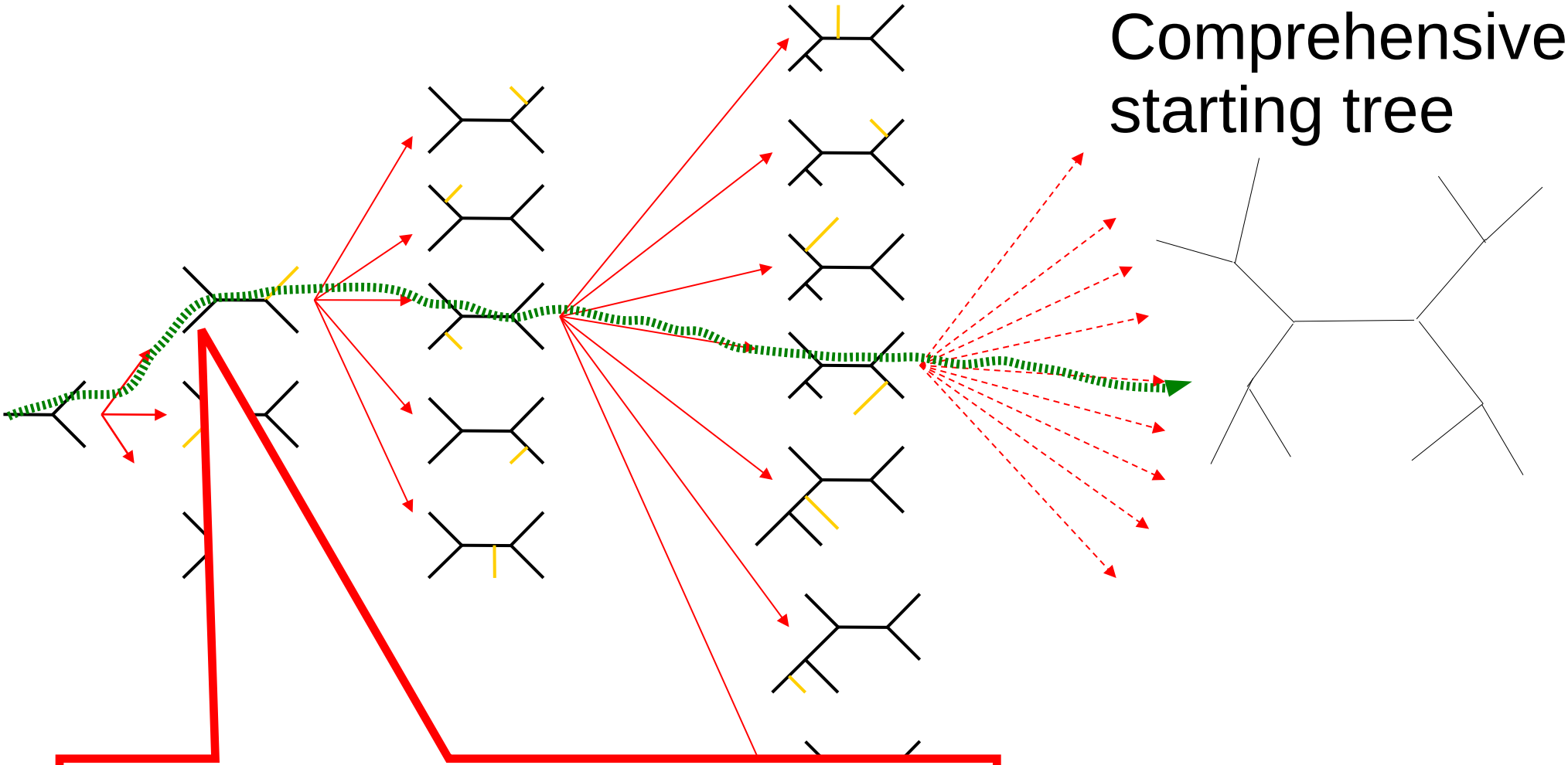
Search Space

good

bad

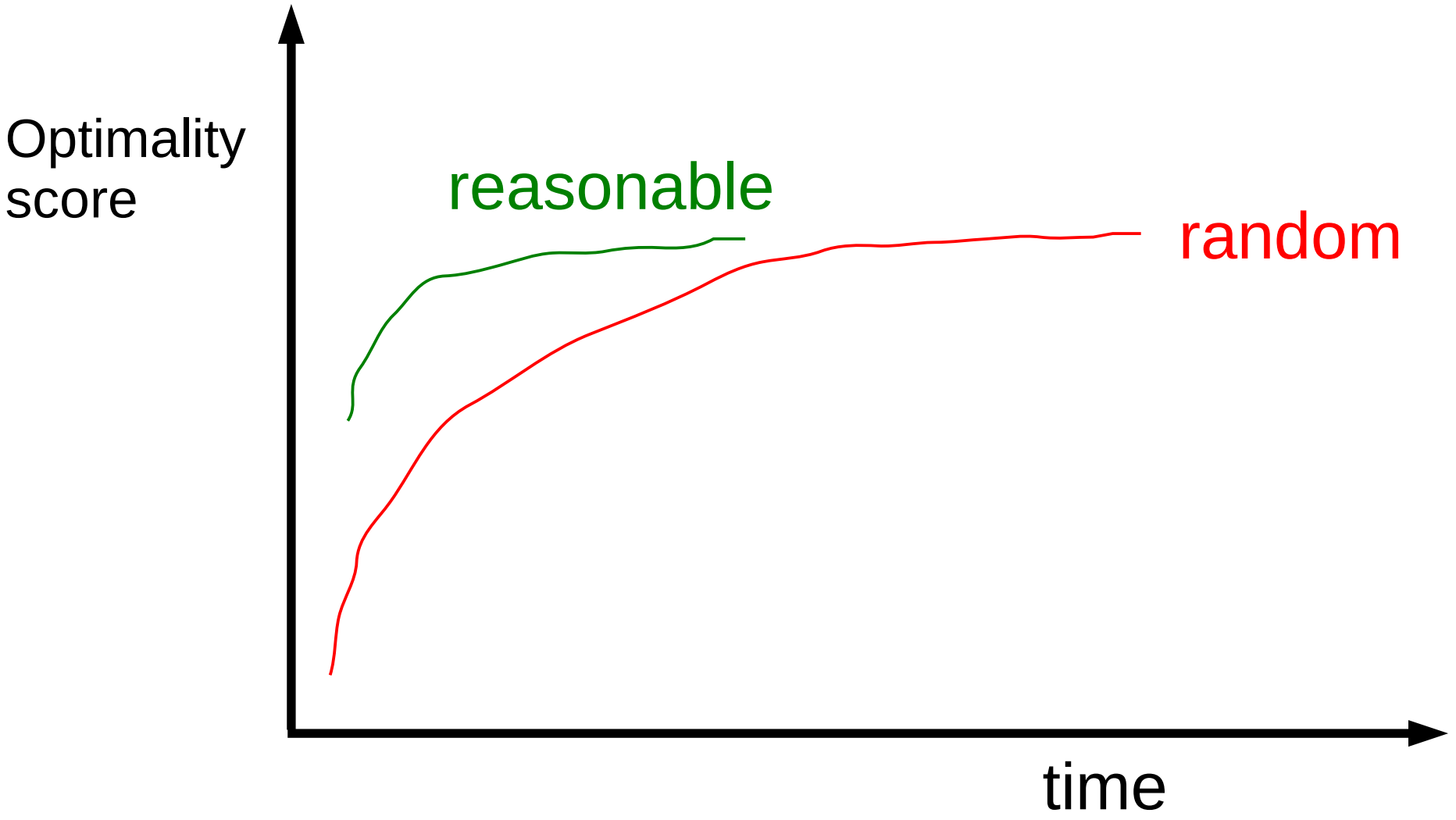
This is the drawback of using NJ or UPGMA trees as starting trees → only one tree

The number of trees



Stepwise addition is like following a single path through this!

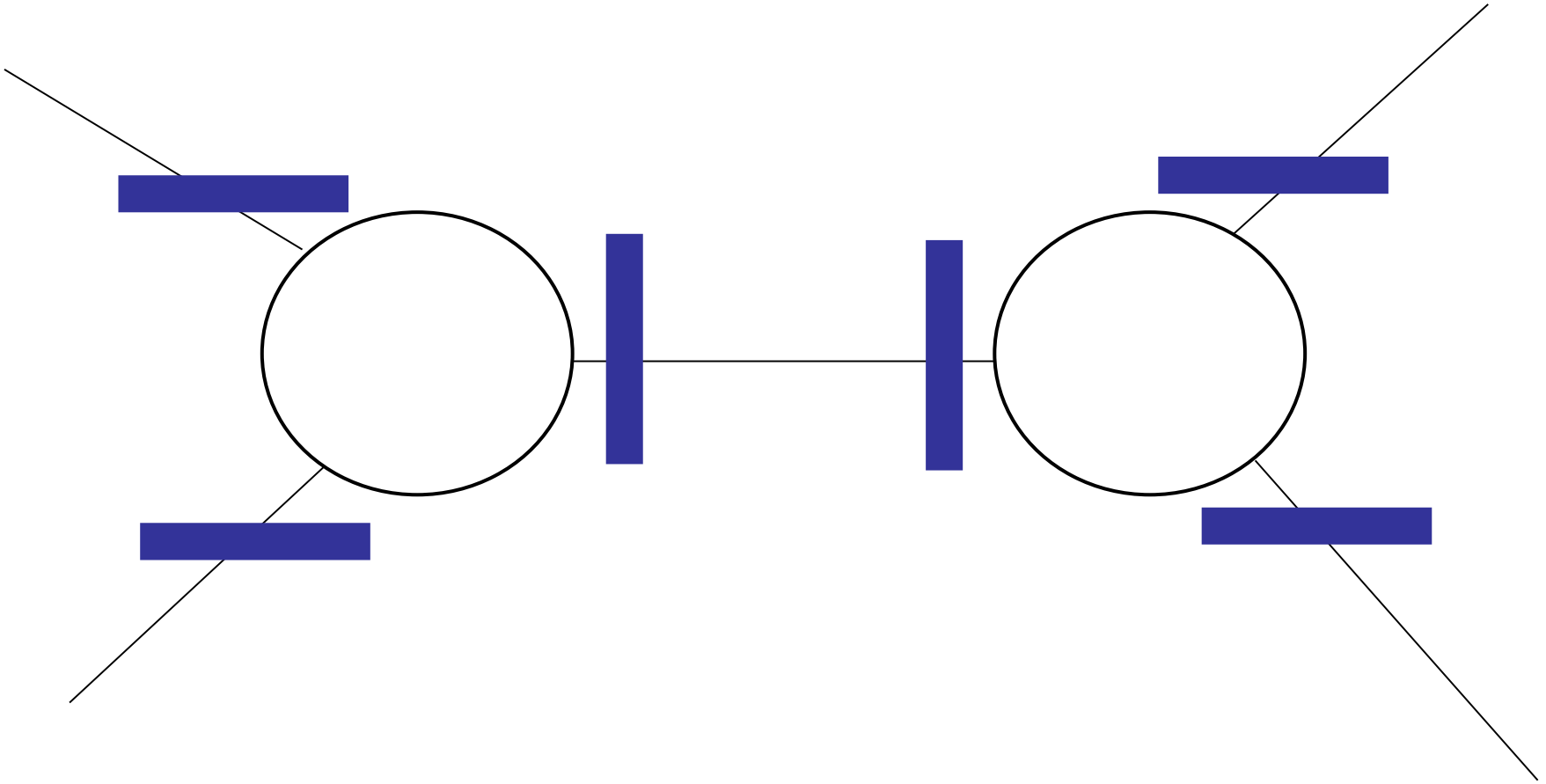
Random versus Reasonable Starting trees



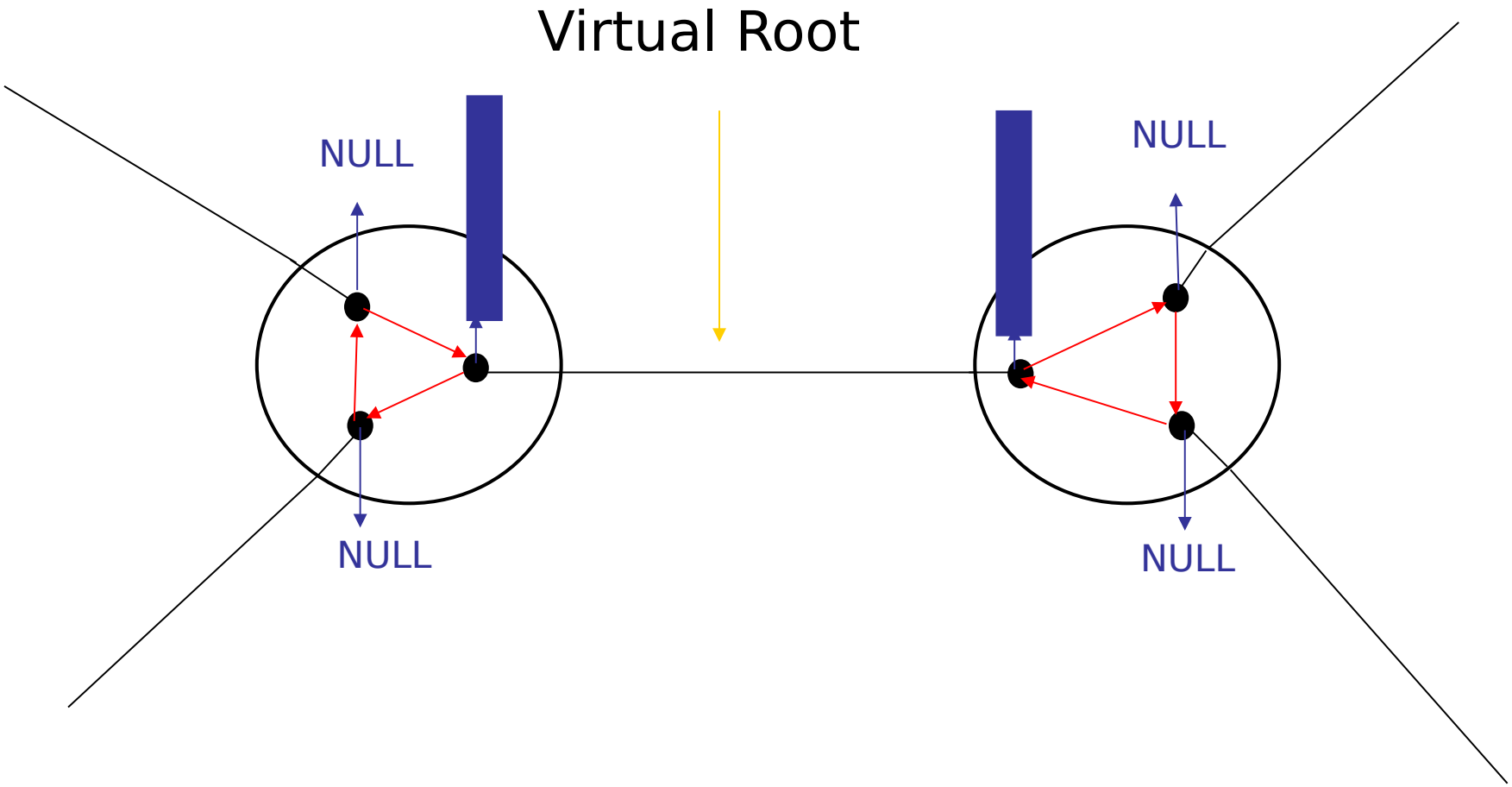
Data Structures for unrooted Trees

- Unrooted trees with dynamically changing virtual roots need a dedicated tree data structure

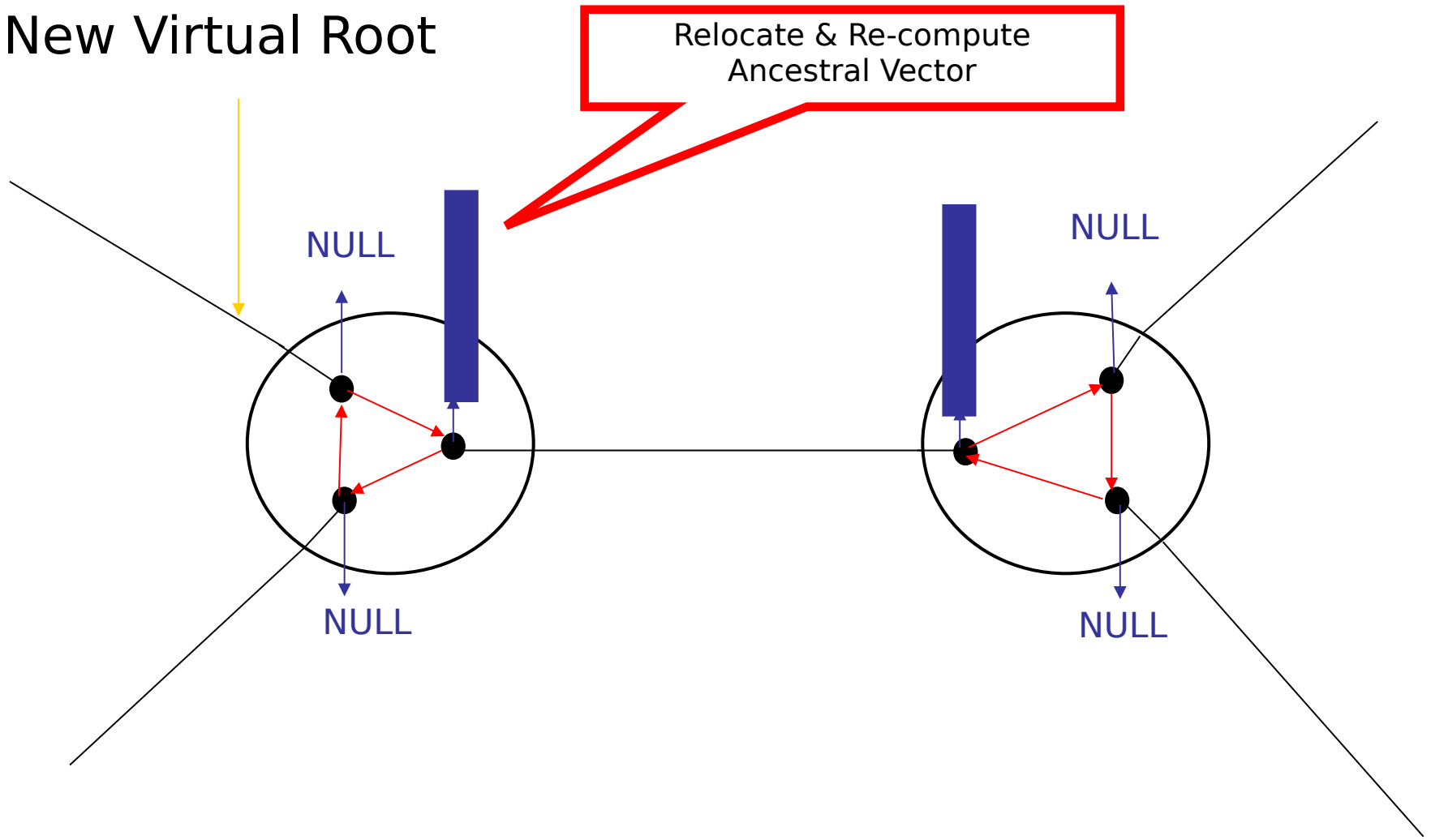
Memory Organization: Ancestral Vectors with an Unrooted View



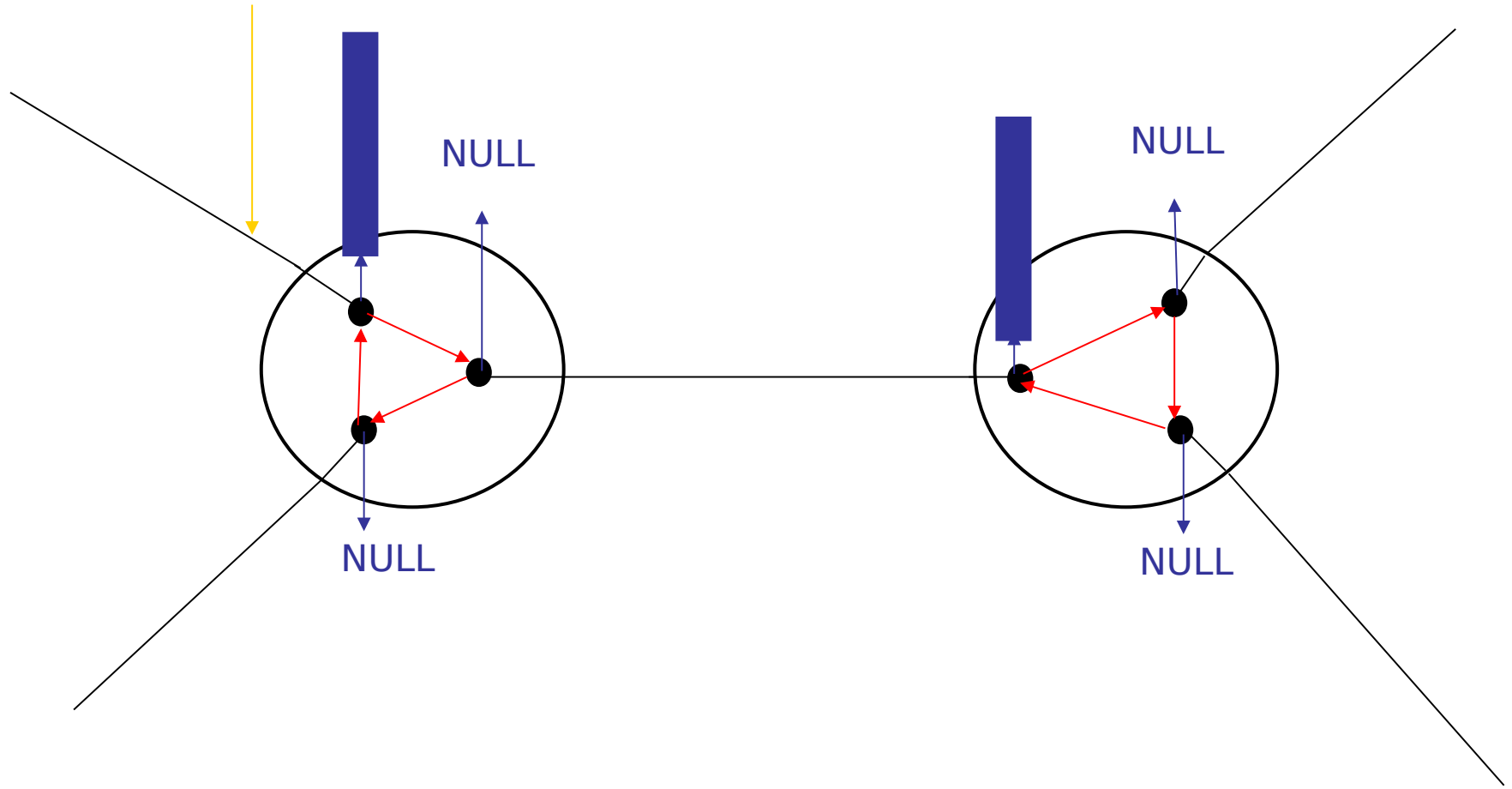
Memory Organization: Ancestral Vectors with a Rooted View



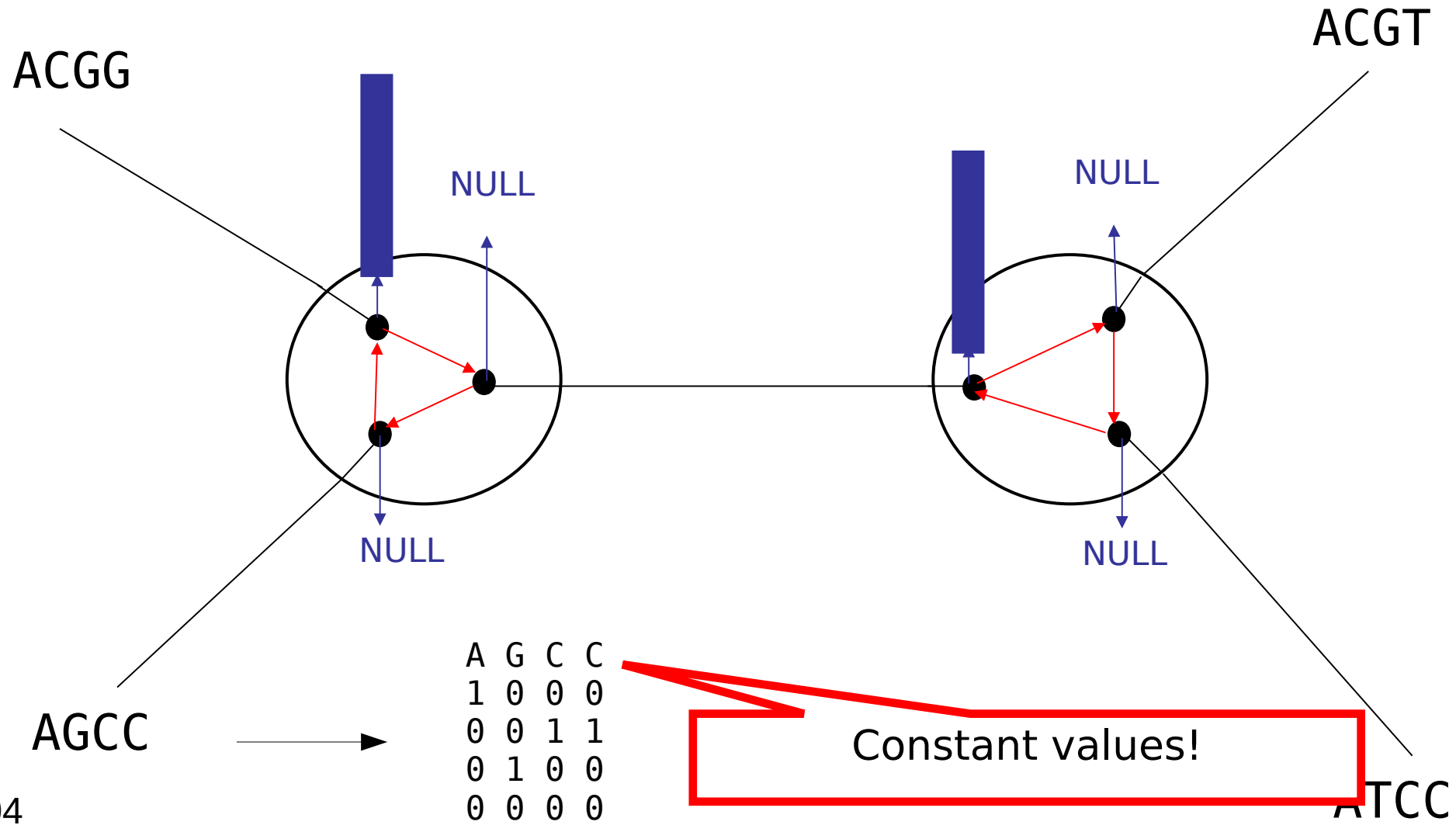
Memory Organization: Ancestral Vectors with a Rooted View



Memory Organization: Ancestral Vectors with a Rooted View



Memory Organization: Tip Vectors

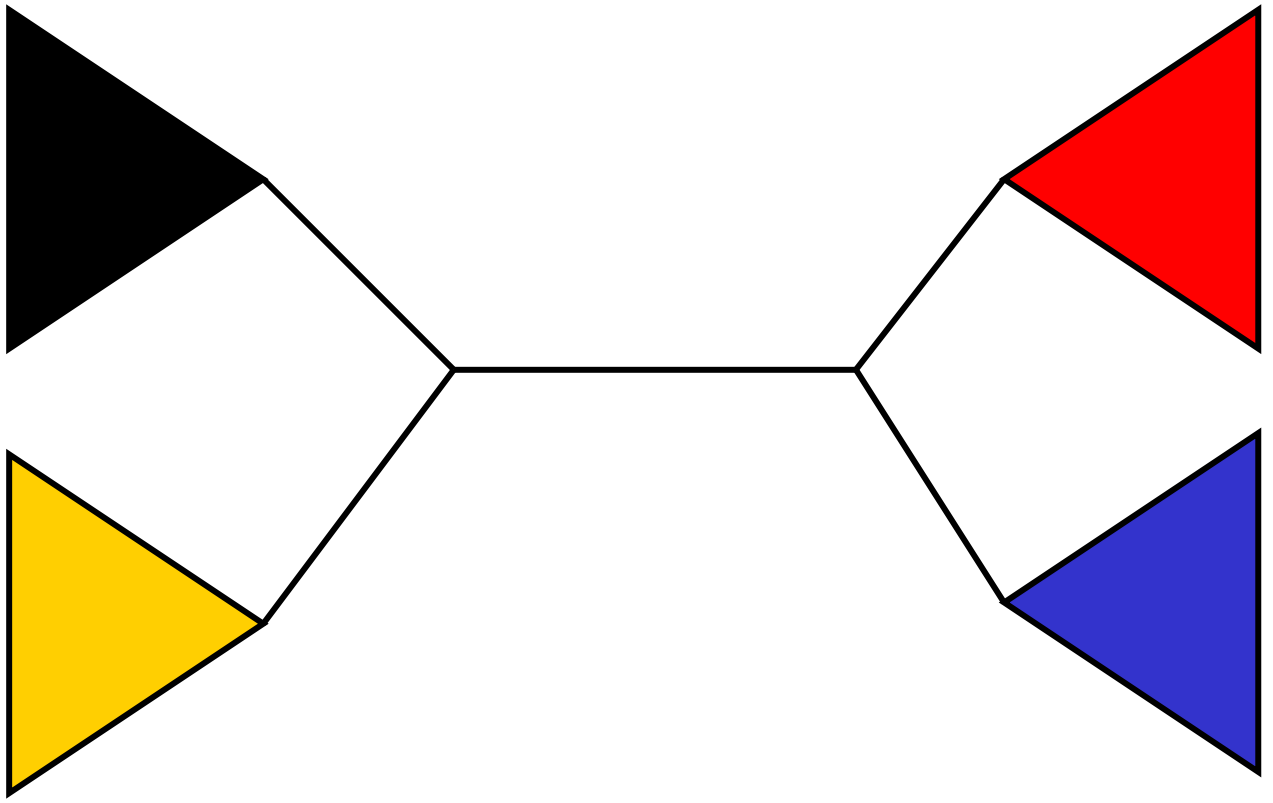


Search Strategies

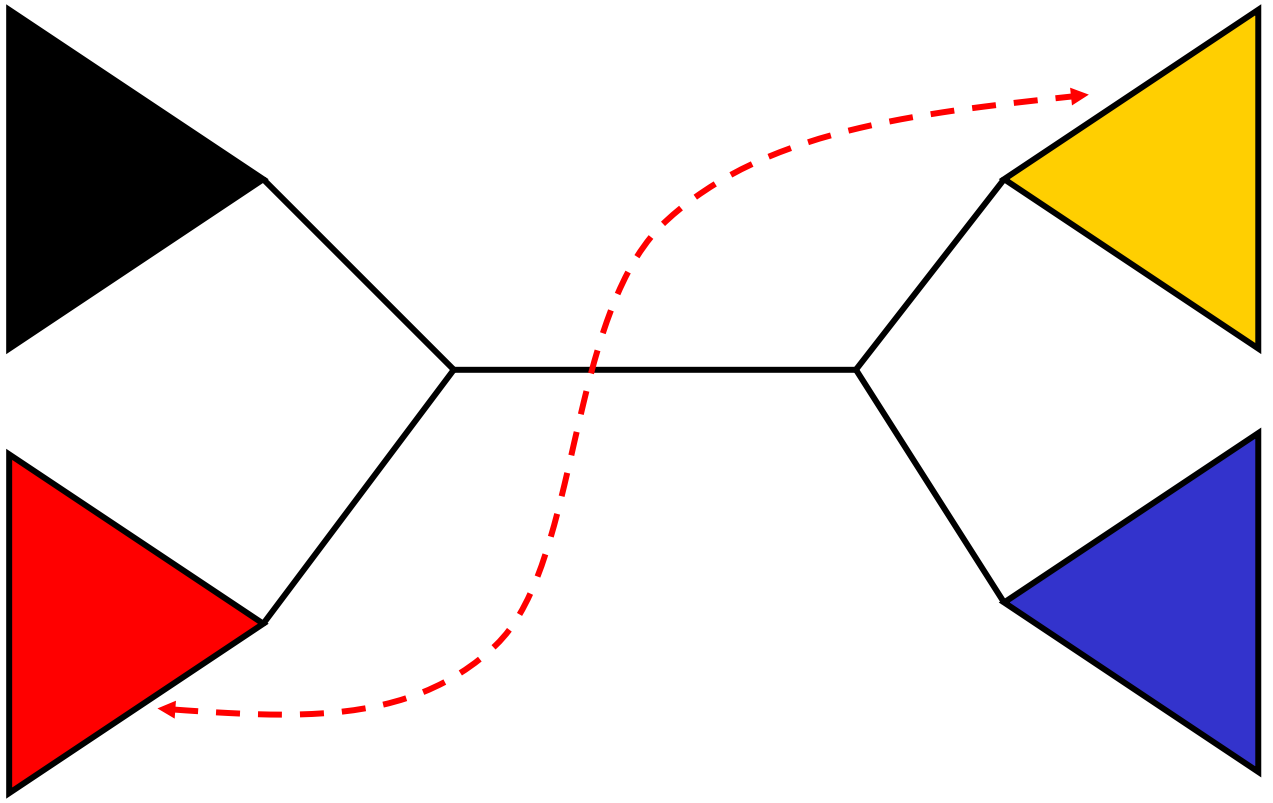
- Given a comprehensive tree
- Apply topological alteration mechanisms in some order to improve the score, for instance, via
  - Hill-climbing
  - Simulated annealing
  - Some other technique

    → design of ad hoc heuristics
- The three basic moves are:
  - **NNI**: Nearest Neighbor Interchange
  - **SPR**: Subtree Pruning and Re-Grafting
  - **TBR**: Tree Bisection and Reconnection

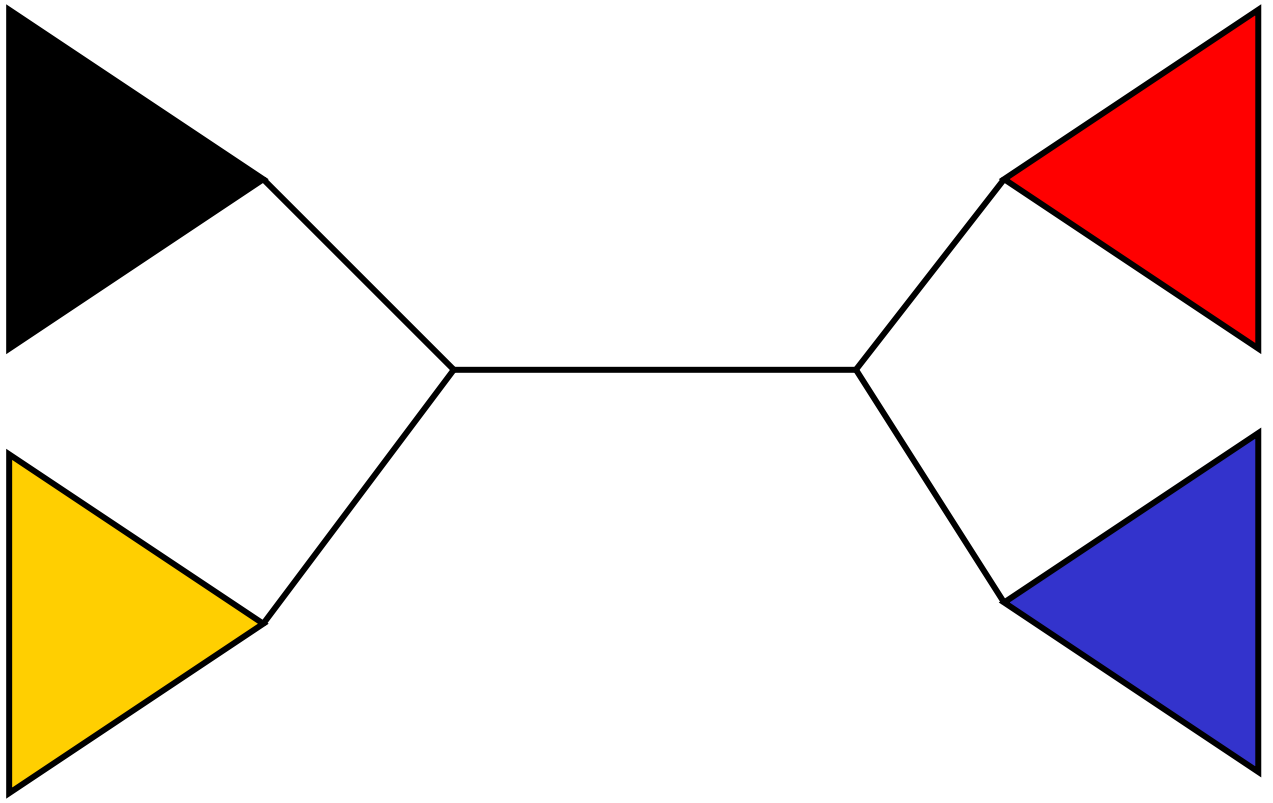
NNI



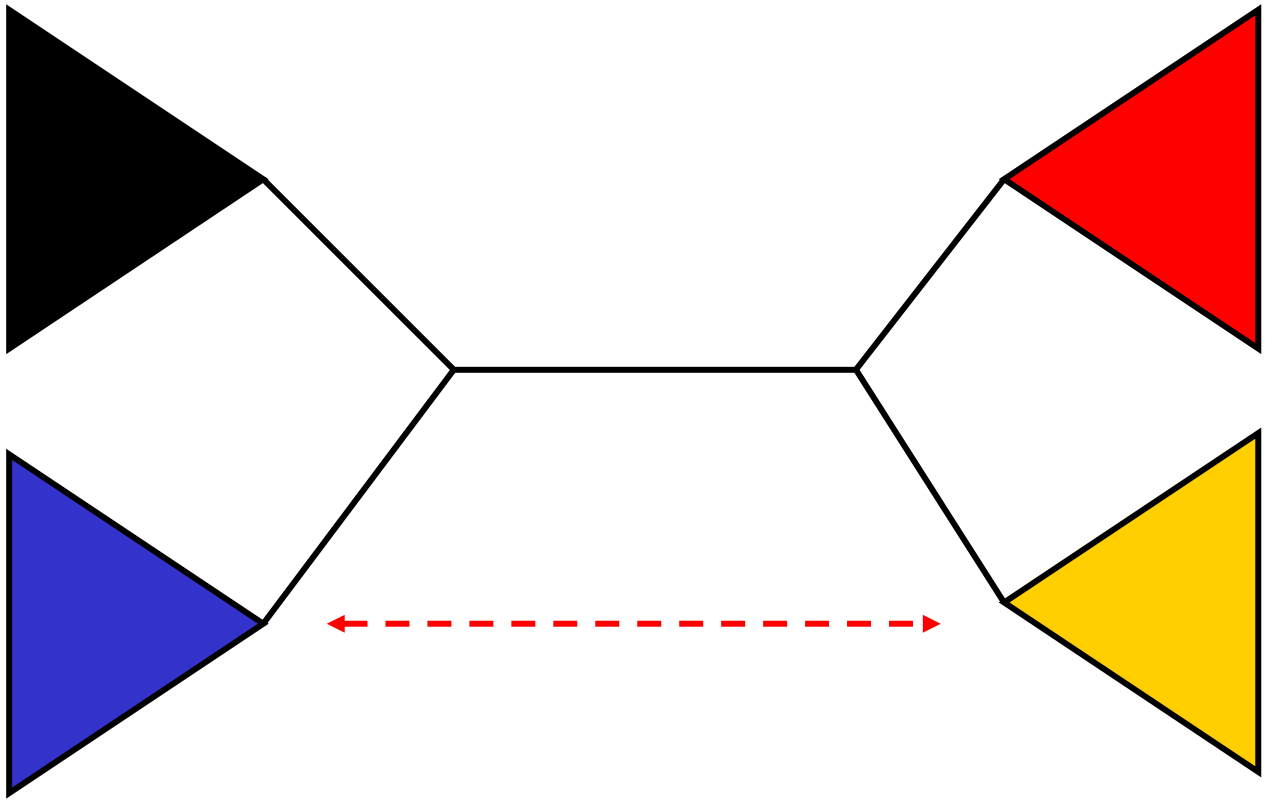
NNI



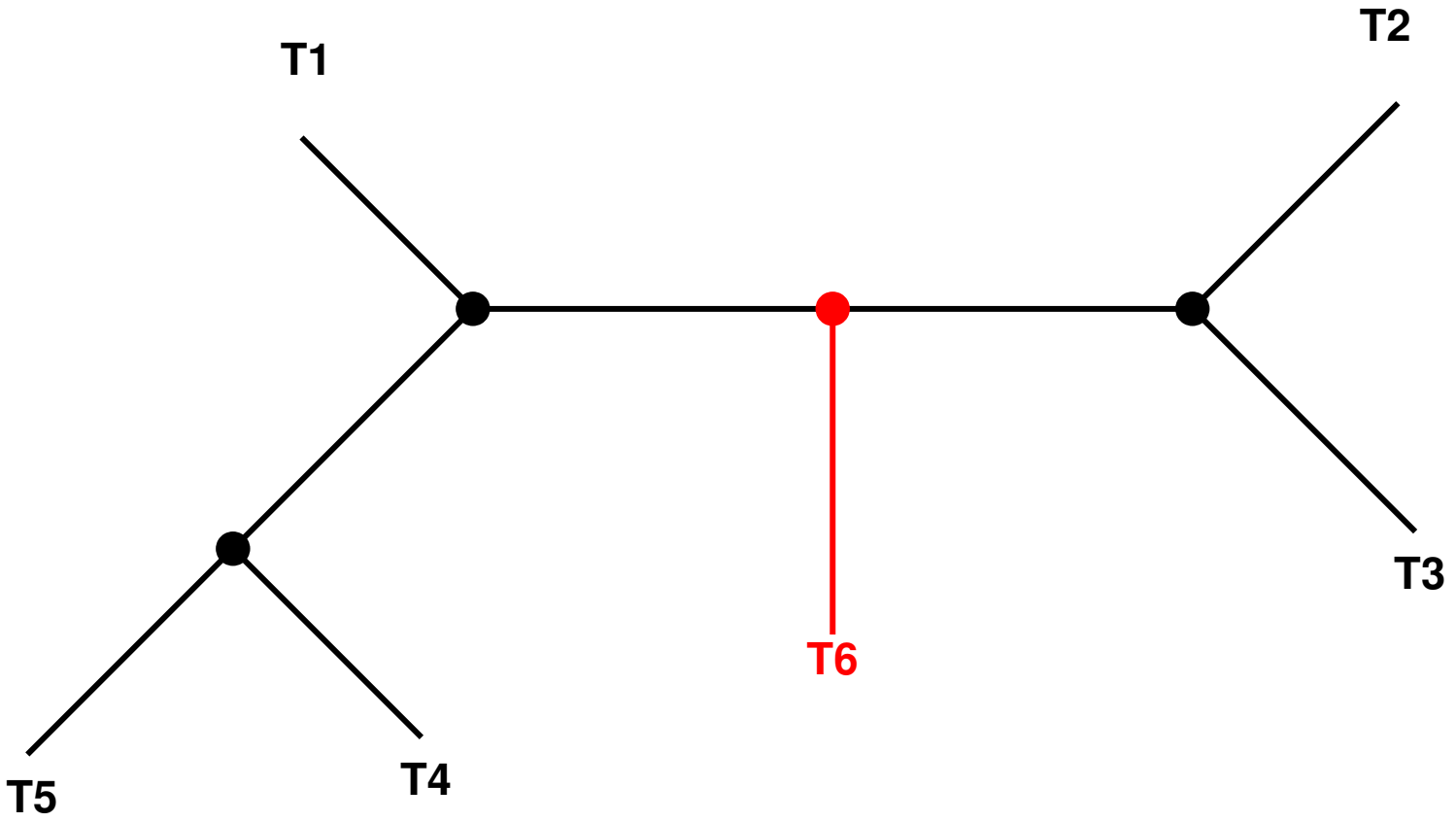
NNI



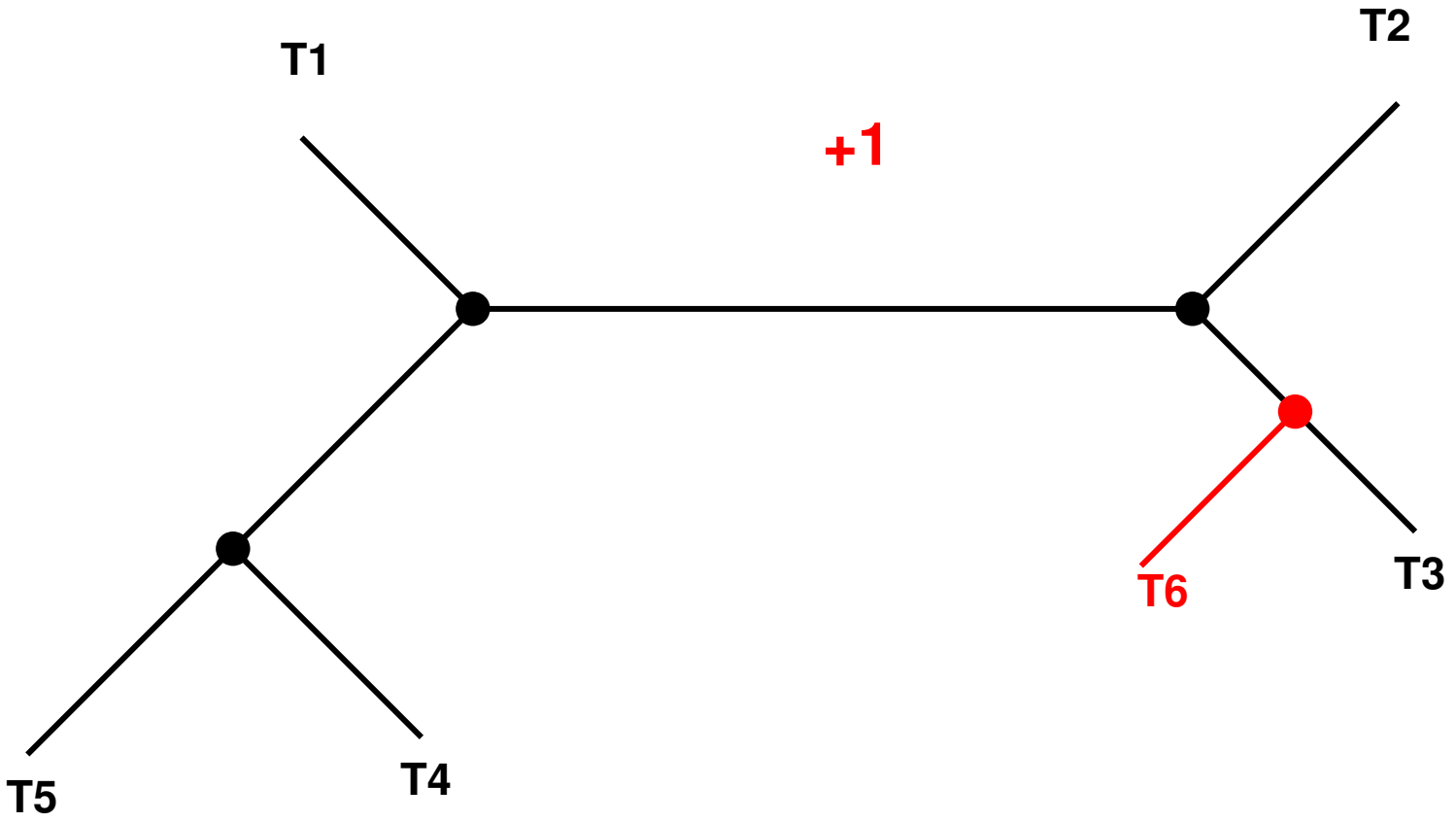
NNI



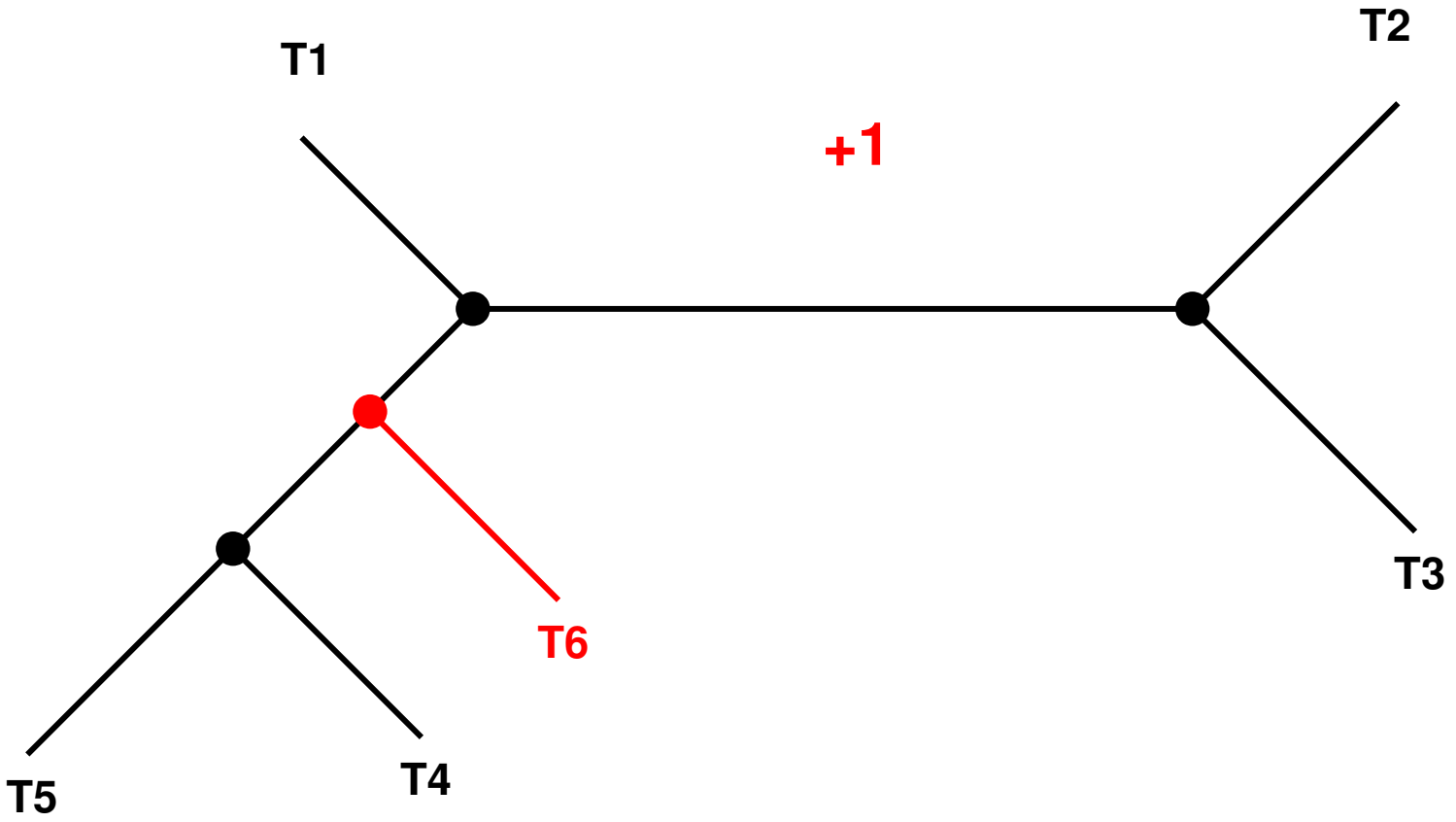
SPR



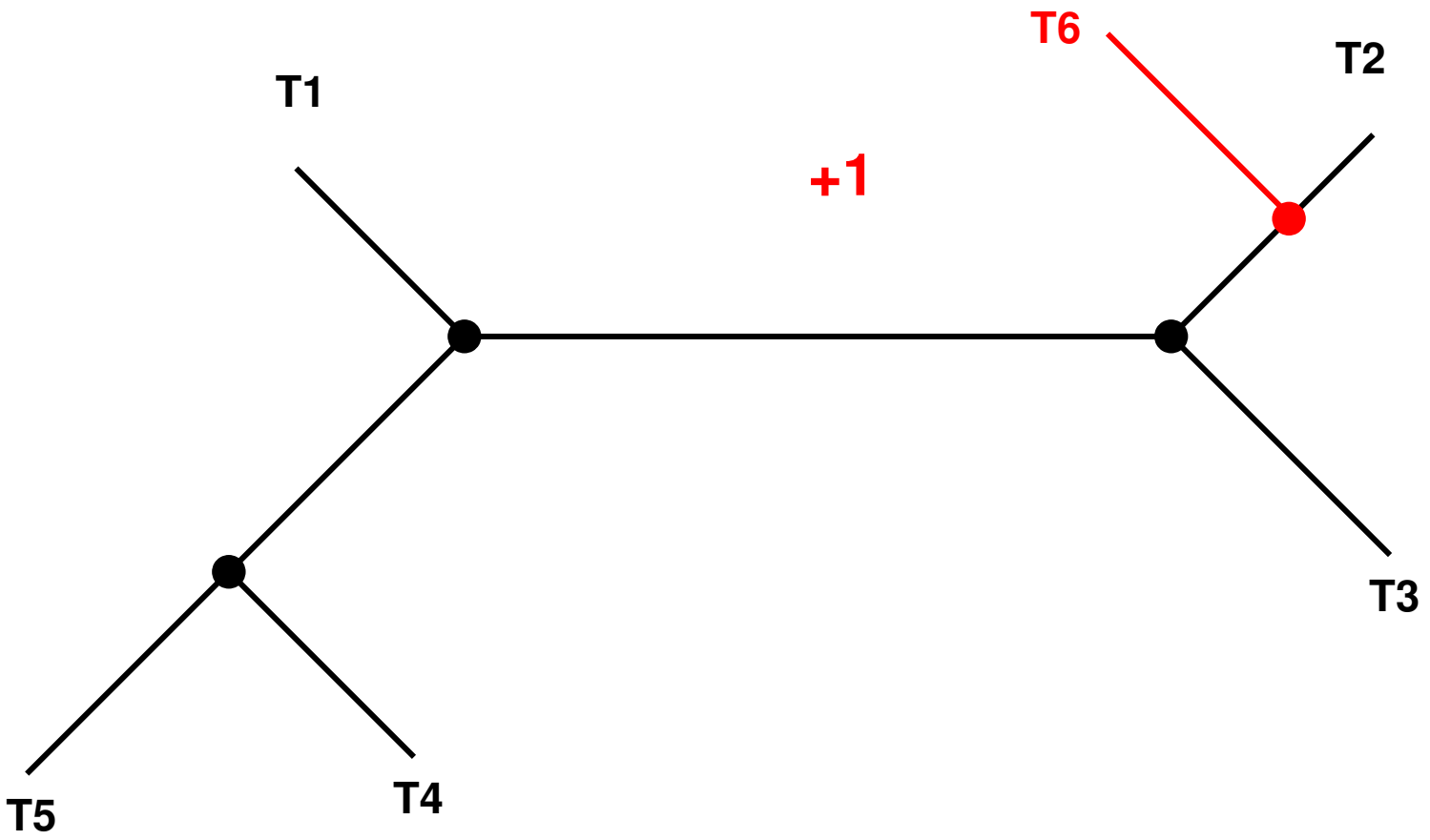
SPR



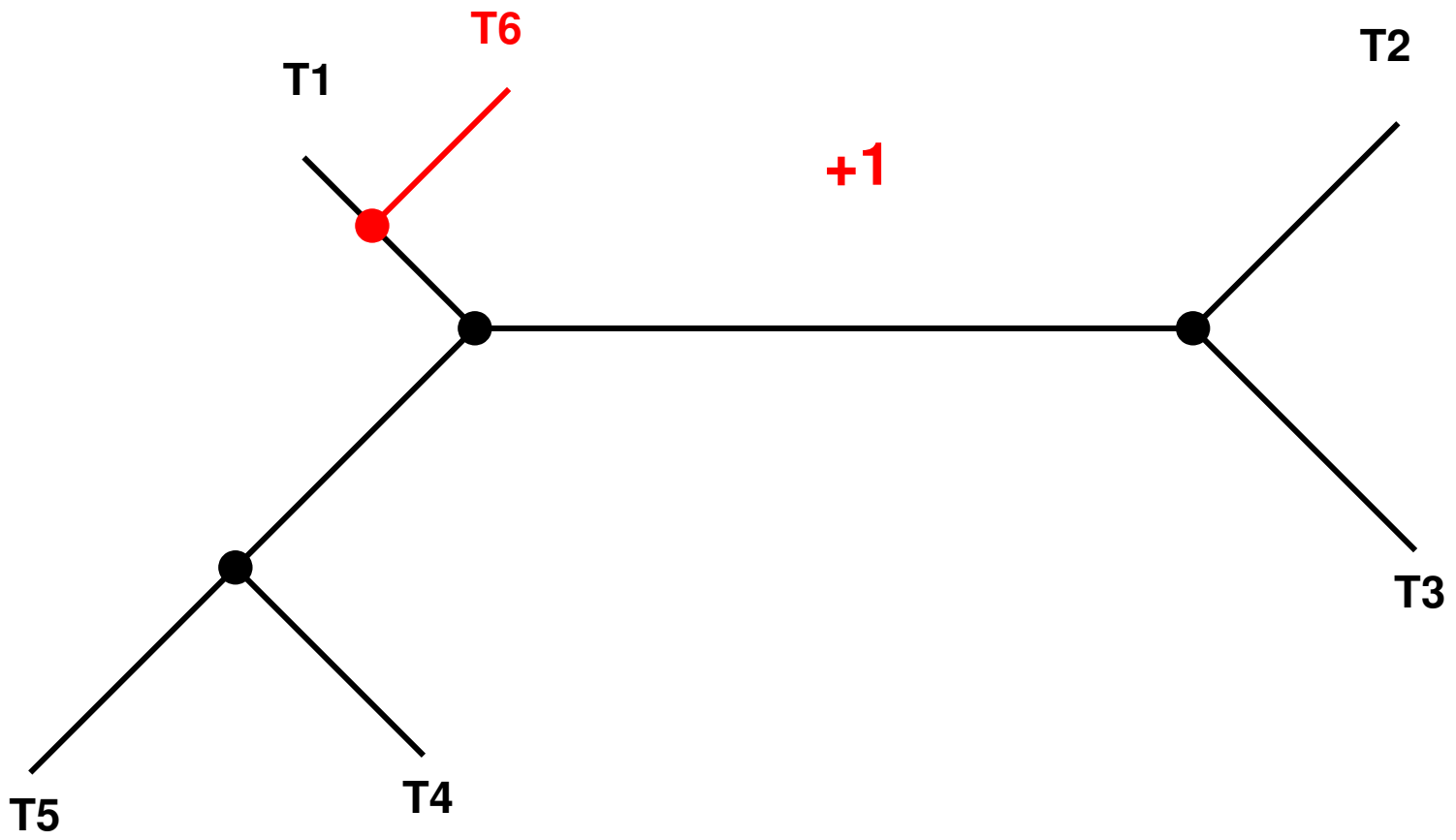
SPR



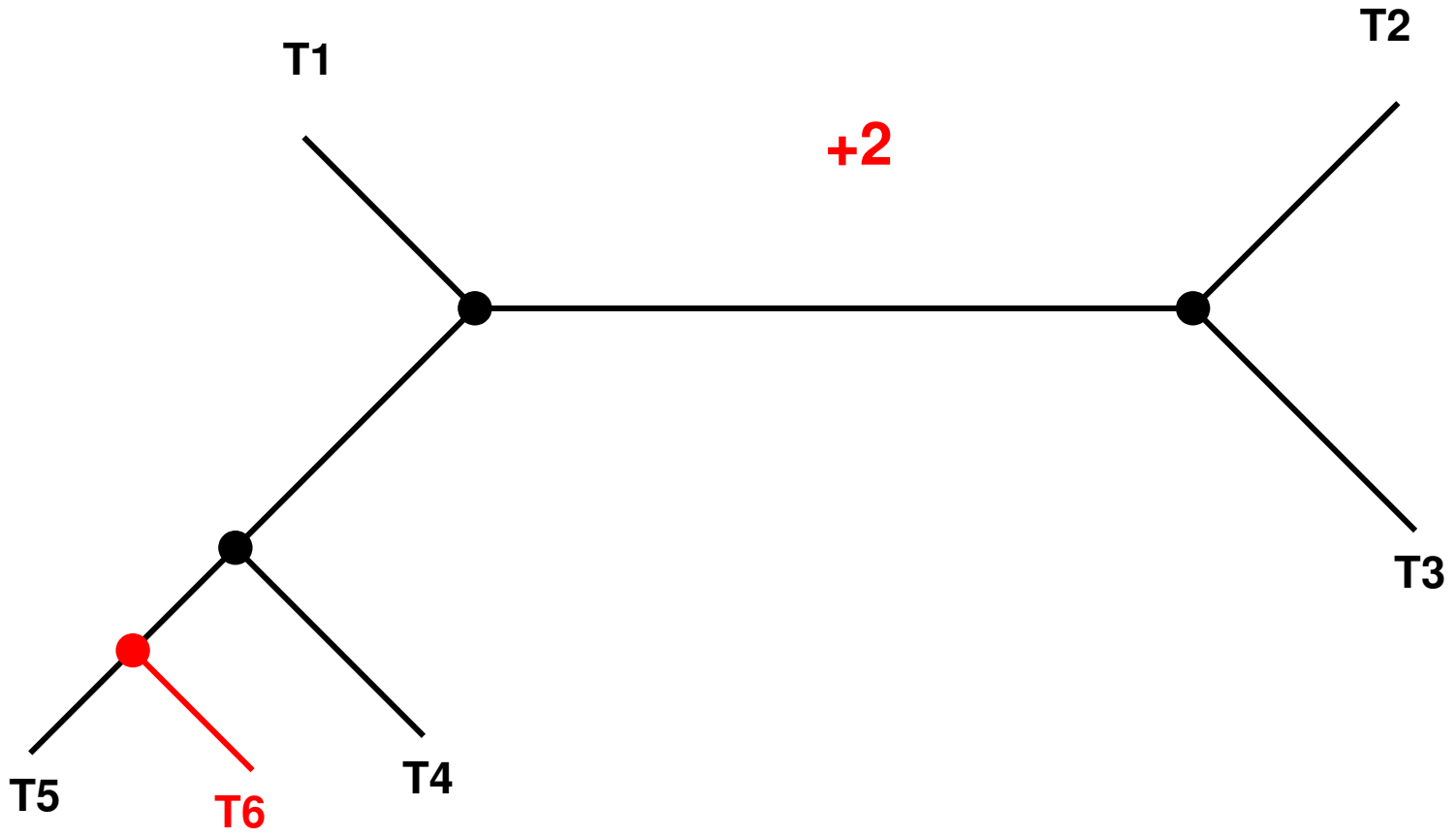
SPR



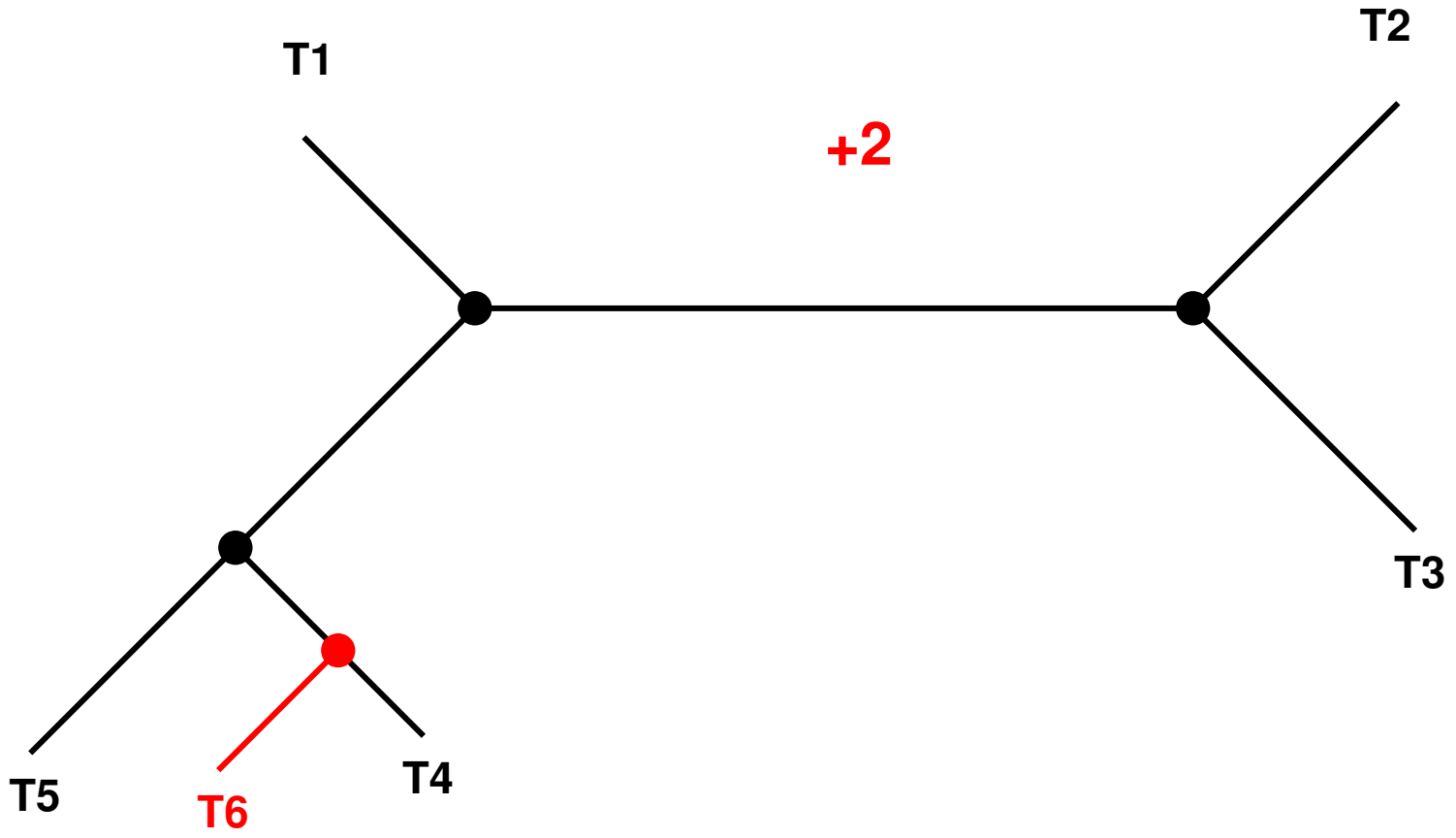
SPR



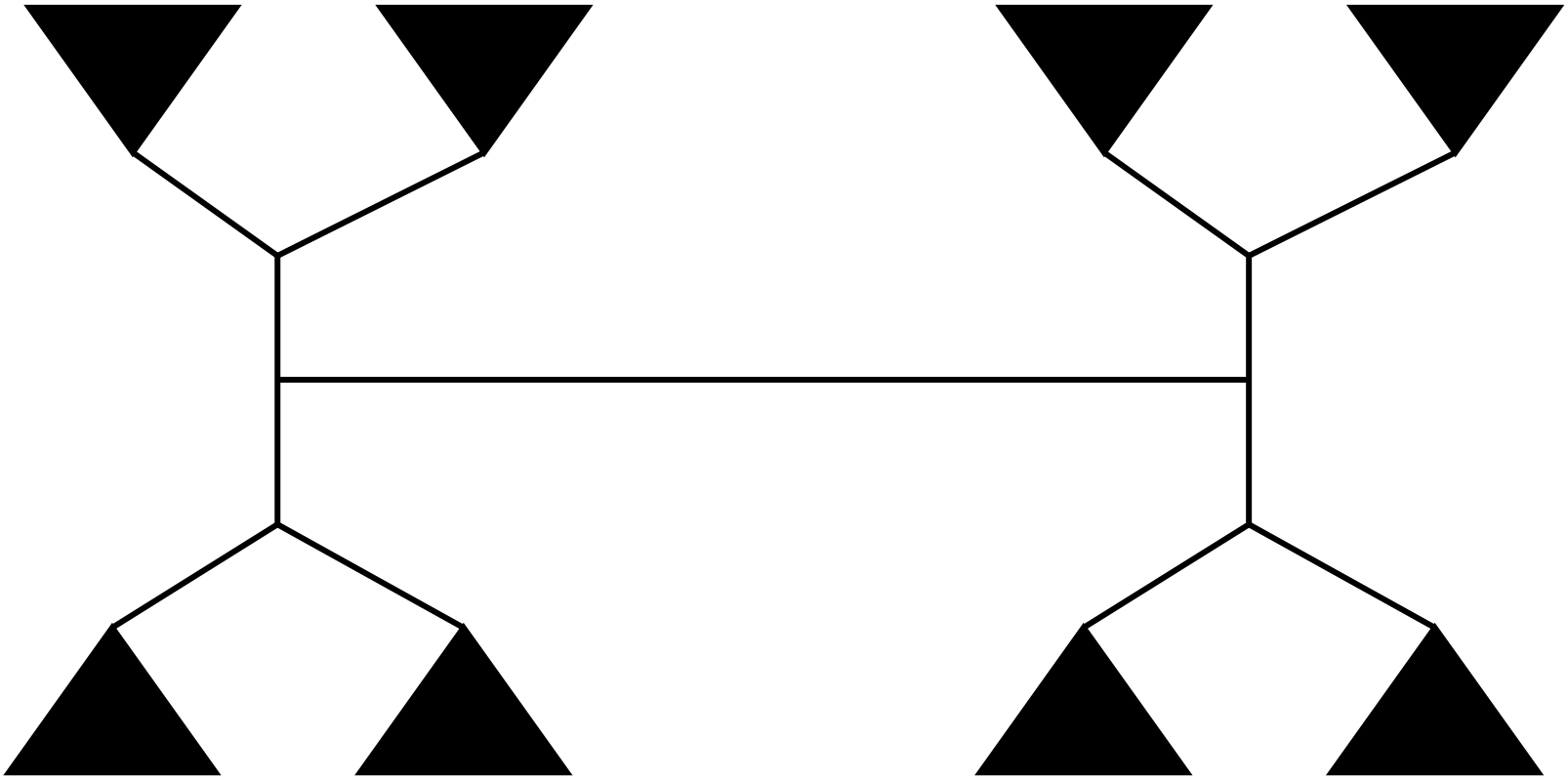
SPR



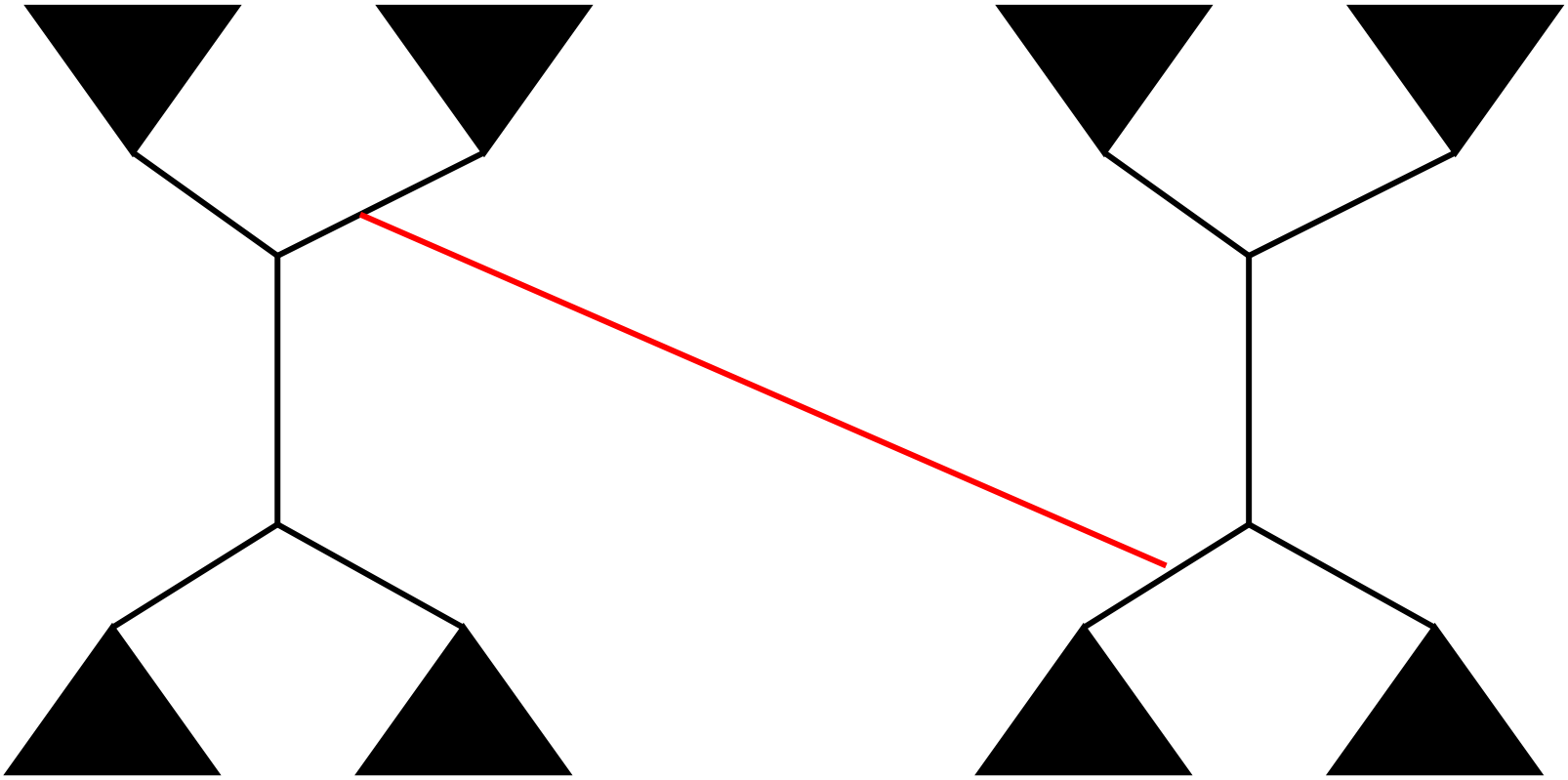
SPR



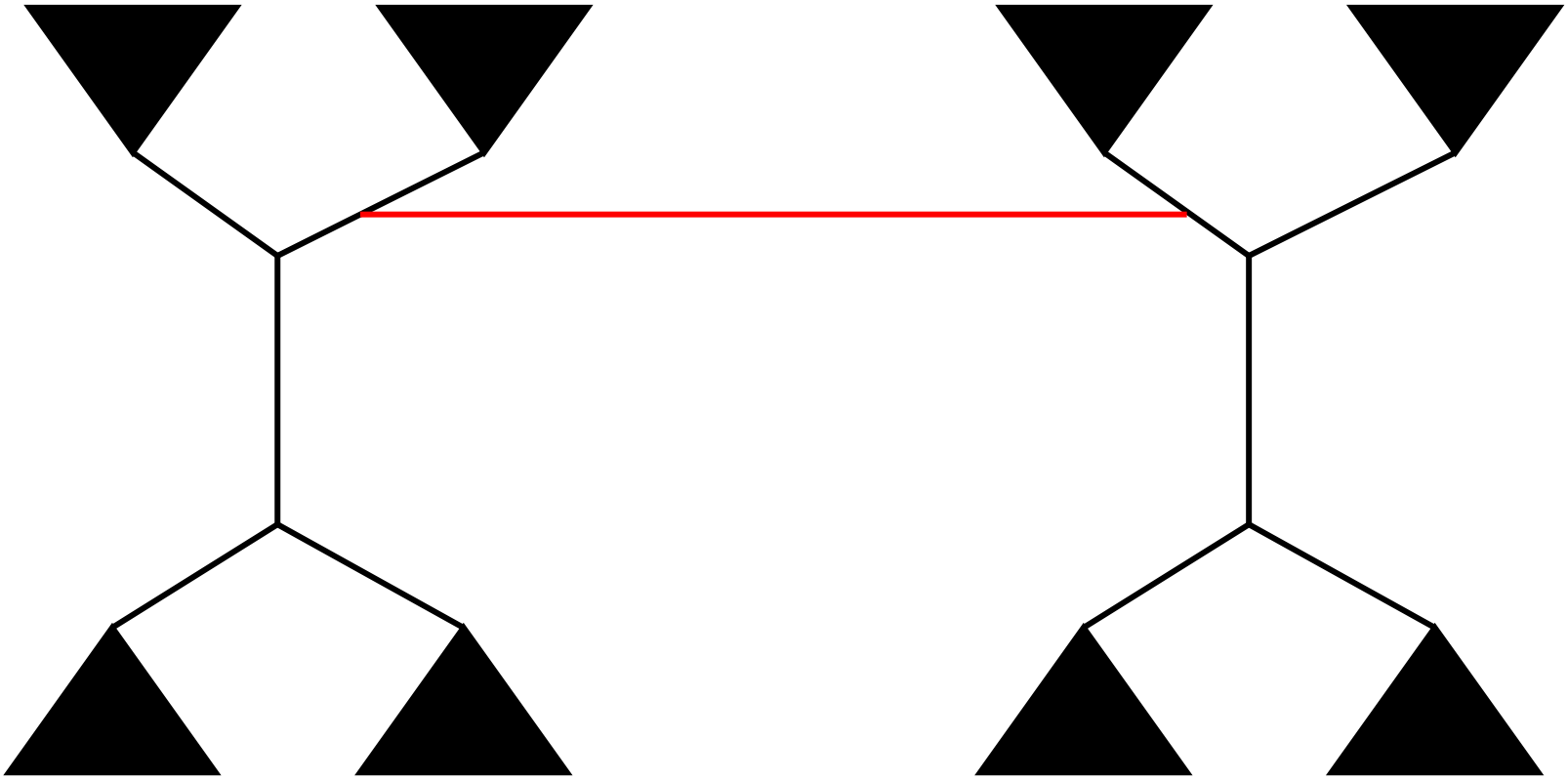
TBR



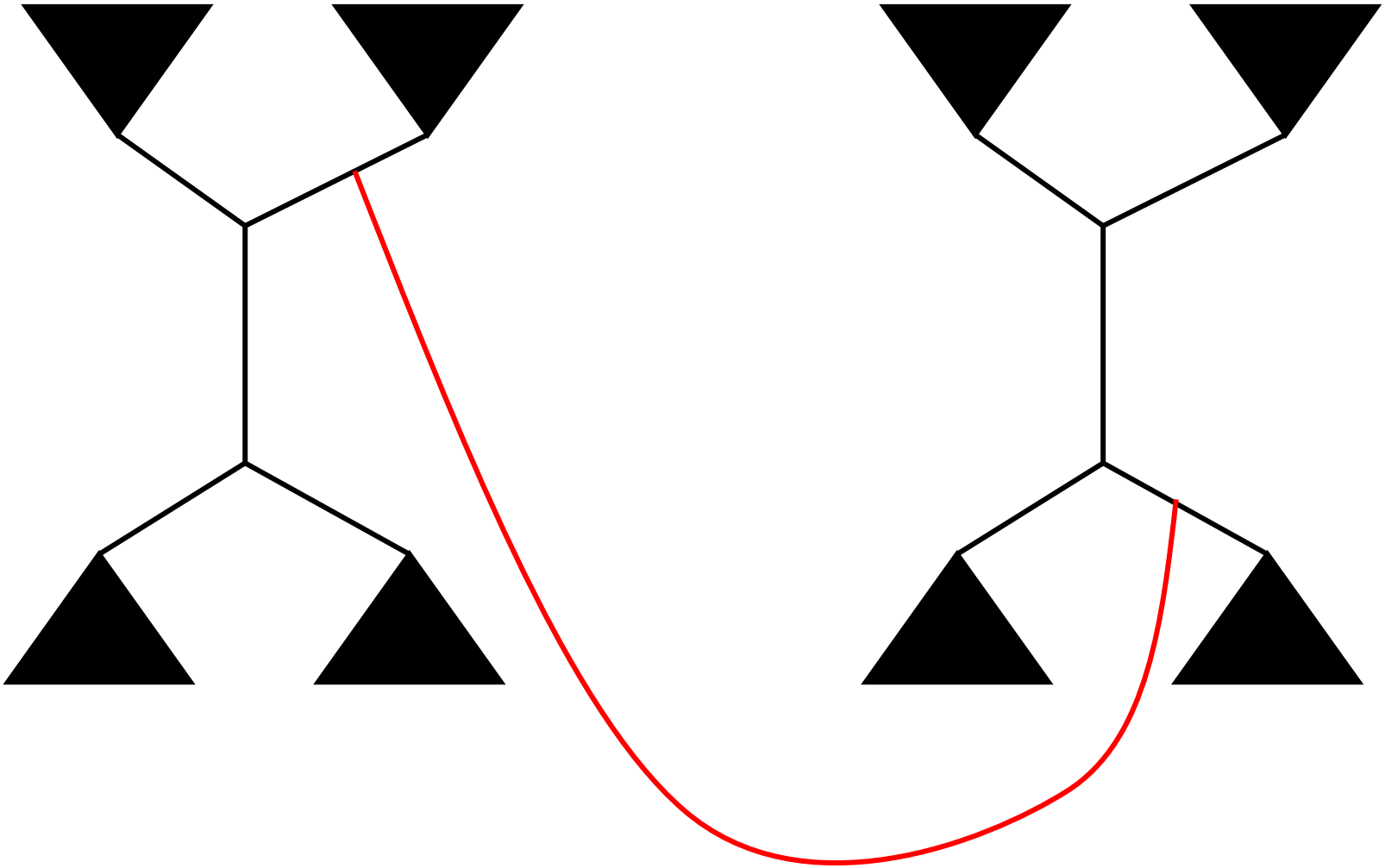
TBR



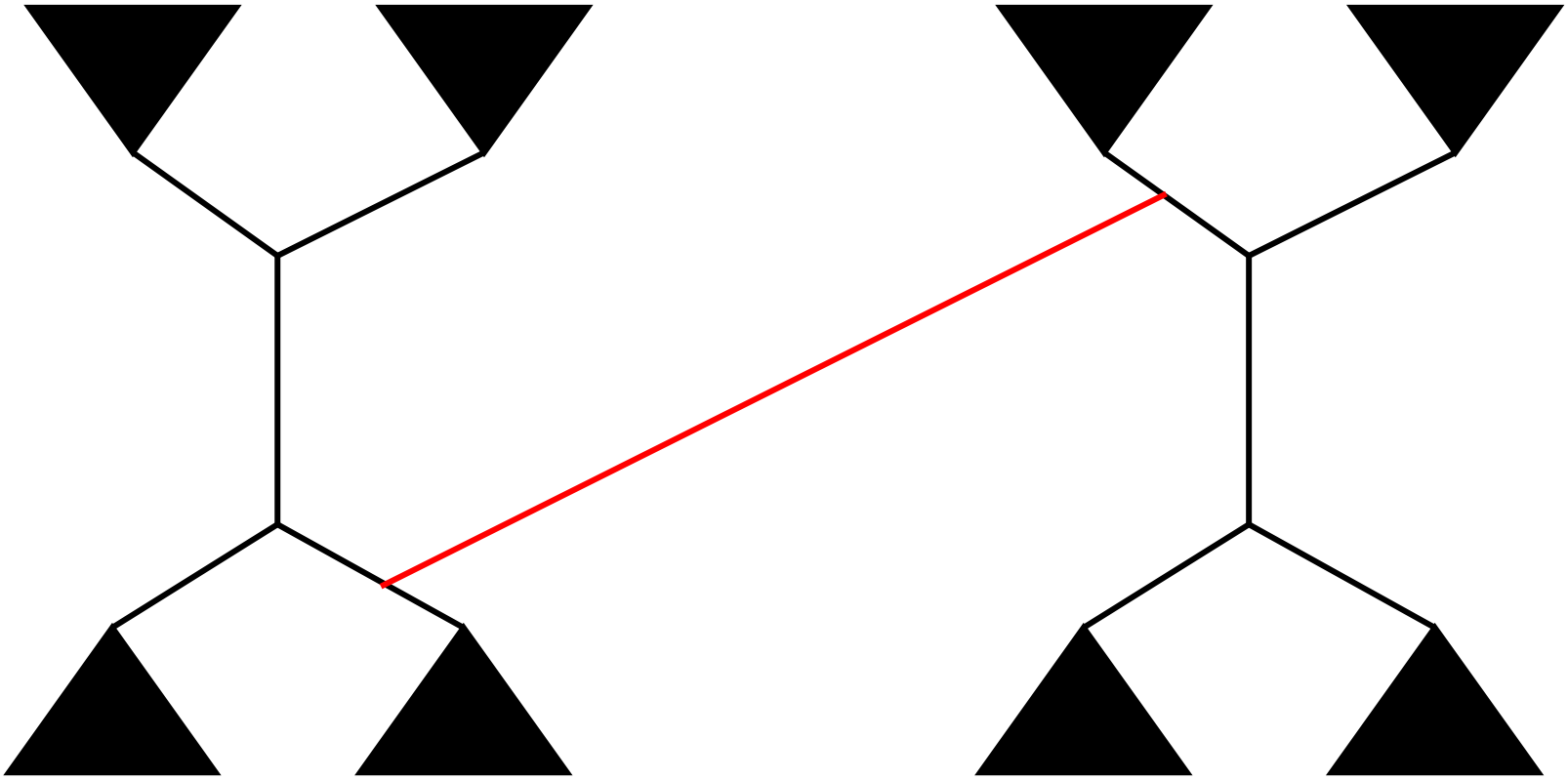
TBR



TBR



TBR



Question

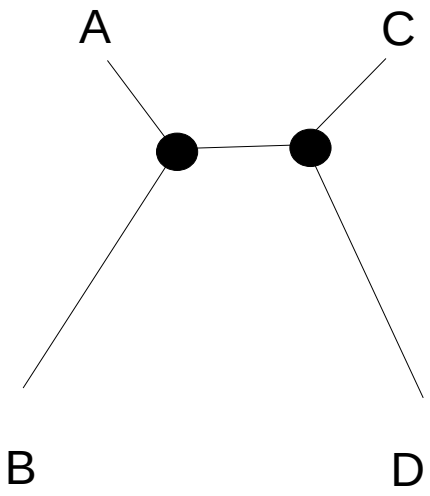
- How could one design a search algorithm for the least squares criterion given a function *f()* and a distance matrix *D* to compute the least squares score on a given tree?

The Parsimonator Algorithm

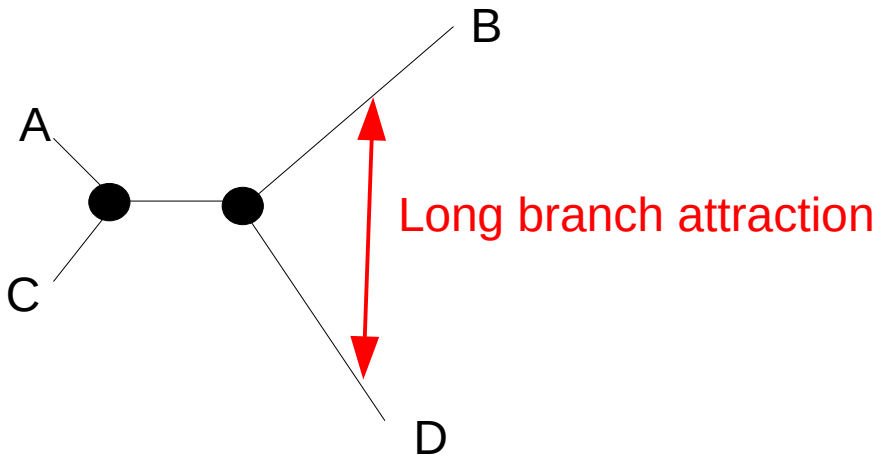
- Build a randomized stepwise addition order parsimony tree
- Apply SPR moves to all subtrees of the current (comprehensive) tree with a rearrangement radius of 20
- If the rearrangement of a subtree yields an improved parsimony score, keep it immediately
  - → this is somewhat greedy as opposed to a steepest ascent hill climbing algorithm
- Continue applying SPR moves with a radius of 20 to all subtrees until no tree with a better parsimony score can be found
- There are much more sophisticated algorithms available
  - → TNT tool by Pablo Goloboff
- Keep in mind that parsimony returns discrete scores, that is, there may be many equally parsimonious trees among which we can not distinguish!

Parsimony & Long Branch Attraction

- Because parsimony tries to minimize the number of mutations it faces some problems on trees with long branches



Correct tree



Wrong tree inferred by parsimony

Parsimony & Long Branch Attraction

- Settings under which parsimony recovers the wrong tree are also called “the Felsenstein Zone” after *Joe Felsenstein* who has made numerous very important contributions to the field, e.g.
  - The Maximum Likelihood model
  - The Bootstrapping procedure
- If you are interested in statistics, there are some on-line courses by Joe at http://evolution.gs.washington.edu/courses.html