582606 Introduction to bioinformatics

Autumn 2006

Esa Pitkänen
Master's Degree Programme in Bioinformatics (MBI)
Department of Computer Science, University of Helsinki
http://www.cs.helsinki.fi/mbi/courses/06-07/itb/

Administrative issues

- Master level course
- Obligatory course in Master's Degree Programme in Bioinformatics
- 4 credits
- Prerequisites: basic mathematical skills
- Lectures: Tuesdays and Fridays 14-16 in Exactum C222
- Exercises: Fridays 12-14 in Exactum C222
  - **Note: exercises start on Friday 22.9.2006!**

Teachers

- Esa Pitkänen, Department of Computer Science, University of Helsinki
- Prof. Elja Arjas, Department of Mathematics and Statistics, University of Helsinki
- Prof. Samuel Kaski, Helsinki University of Technology

How to enrol for the course?

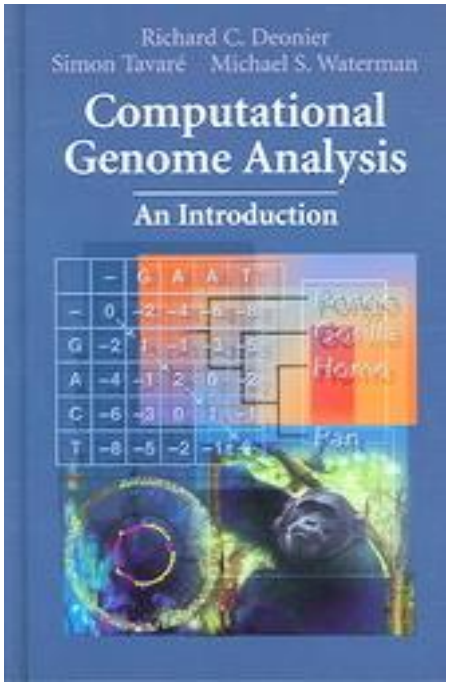
- Use the registration system of the Computer Science department: https://ilmo.cs.helsinki.fi

How to successfully pass the course?

- You can get a maximum of 60 points
  - Course exam: maximum of 50 points
  - Exercises: maximum of 10 points
    - 0% completed assignments gives you 0 points, 80% gives 10 points
- Course will be graded on the scale 0-5
  - To get the lowest passing grade 1/5, you need to have at least 30 points
- Course exam: Monday 16.10. at 16.00-19.00

Course material

- Course book: Richard C. Deonier, Simon Tavare & Michael S. Waterman: Computational Genome Analysis – an Introduction, Springer 2005
- Available at Kumpula and Viikki science libraries; Yliopistokirjakauppa 69€, Akateeminen 75€, Suomalainen 87€, amazon.com $66.57, amazon.co.uk £47.50 (6.9.2006)
- It is recommended that you have access to the course book!
- Slides for some lectures will be available on the course web page



Course contents

- Biological background (book chapter 1)
- Probability calculus (chapters 2 and 3)
- Sequence alignment (chapter 6)
- Phylogenetics (chapter 12)
- Expression data analysis (chapter 11)

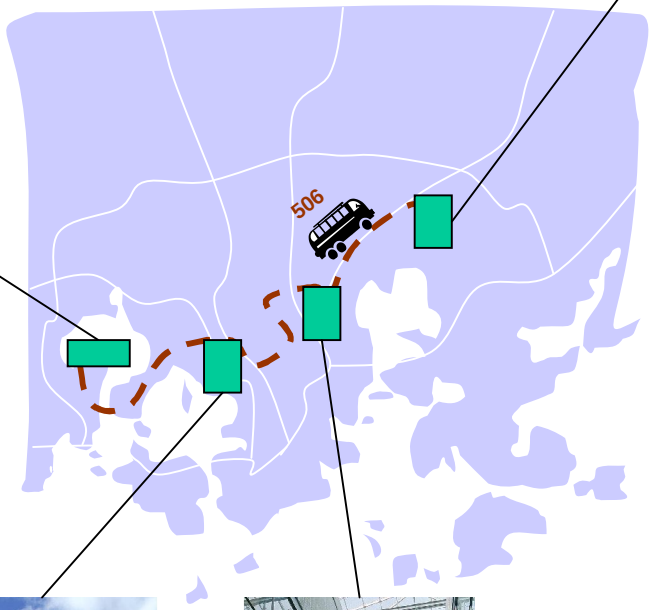
Master's Degree Programme in Bioinformatics (MBI)

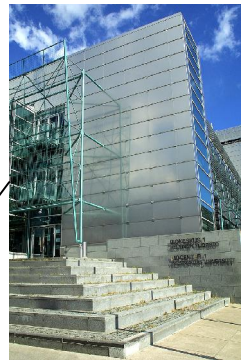
- Two-year MSc programme
- Offered jointly by the University of Helsinki and Helsinki University of Technology
- Admission for 2007-2008 in January 2007

Institutions participating in MBI

Helsinki University of Technology

- **Laboratory of Computer and Information Science**





University of Helsinki
Faculty of Biosciences
Faculty of Agriculture and Forestry

University of Helsinki
Faculty of Medicine

University of Helsinki
Faculty of Science

- **Department of Computer Science**
- **Department of Mathematics and Statistics**

Bioinformatics courses at the University of Helsinki

- Department of Computer Science
  - Practical course in biodatabases (II period): techniques for accessing and integrating data in biology databases.
  - Computational neuroscience (II period): mathematical modeling of information processing taking place in the brain.
  - Biological sequence analysis (III period): basic probabilistic methods for modelling and analysis of biological sequences.
  - Modeling of vision (III period): mechanisms and modeling of human perception.
  - Metabolic modeling (IV period): metabolic networks, fluxes and regulation of metabolism.

Bioinformatics courses at the University of Helsinki

- Department of Mathematics and Statistics
  - Modelling fluctuating populations (I and II periods): systems driven by fluctuating parameters
  - Evolution and the theory of games (III period): introduction to game theory with emphasis on applications in evolutionary and behavioural biology

Bioinformatics courses at Helsinki University of Technology

- Laboratory of Computer and Information Science
  - Special course in bioinformatics II (I and II periods): data integration and fusion in bioinformatics.
  - Signal processing in neuroinformatics (I and II periods): overview of some of the main biomedical signal processing techniques
  - High-throughput bioinformatics (III and IV periods): computational and statistical methods for analyzing modern high-throughput biological data
  - Image analysis in neuroinformatics (III and IV periods): biomedical image processing techniques

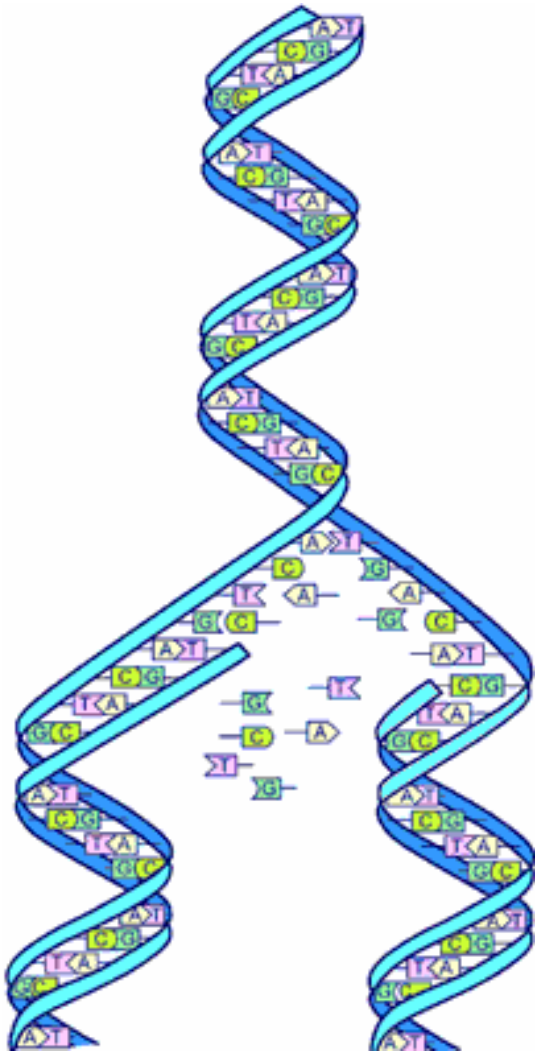
Biology for methodological scientists (8 cr)

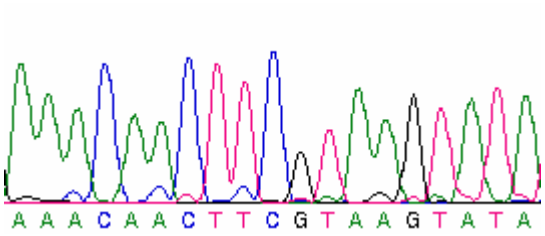
- Course organized by the Faculties of Bioscience and Medicine for the MBI programme
- Introduction to basic concepts of microarrays, genetics, molecular medicine and developmental biology
- Organized in four modules, 2 cr each
- Each module has an individual registration so you can participate even if you missed the first module
- www.cs.helsinki.fi/bioinformatiikka/mbi/courses/06-07/bfms/

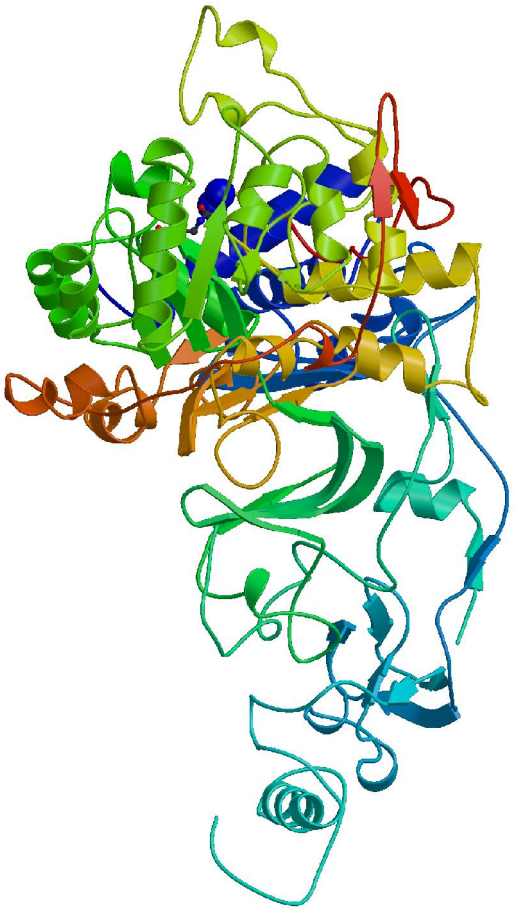
Bioinformatics courses

- Visit the website of Master's Degree Programme in Bioinformatics for up-to-date course lists: http://www.cs.helsinki.fi/mbi

An introduction to bioinformatics







What is bioinformatics?

- Solving biological problems with computation?
- Collecting, storing and analysing biological data?
- Informatics - library science?

What is bioinformatics?

- Bioinformatics, *n.* The science of information and information flow in biological systems, esp. of the use of computational methods in genetics and genomics. (Oxford English Dictionary)
- "The mathematical, statistical and computing methods that aim to solve biological problems using DNA and amino acid sequences and related information."
  -- Fredj Tekaia
- "I do not think all biological computing is bioinformatics, e.g. mathematical modelling is not bioinformatics, even when connected with biology-related problems. In my opinion, bioinformatics has to do with management and the subsequent use of biological information, particular genetic information."
  -- Richard Durbin

What is not bioinformatics?

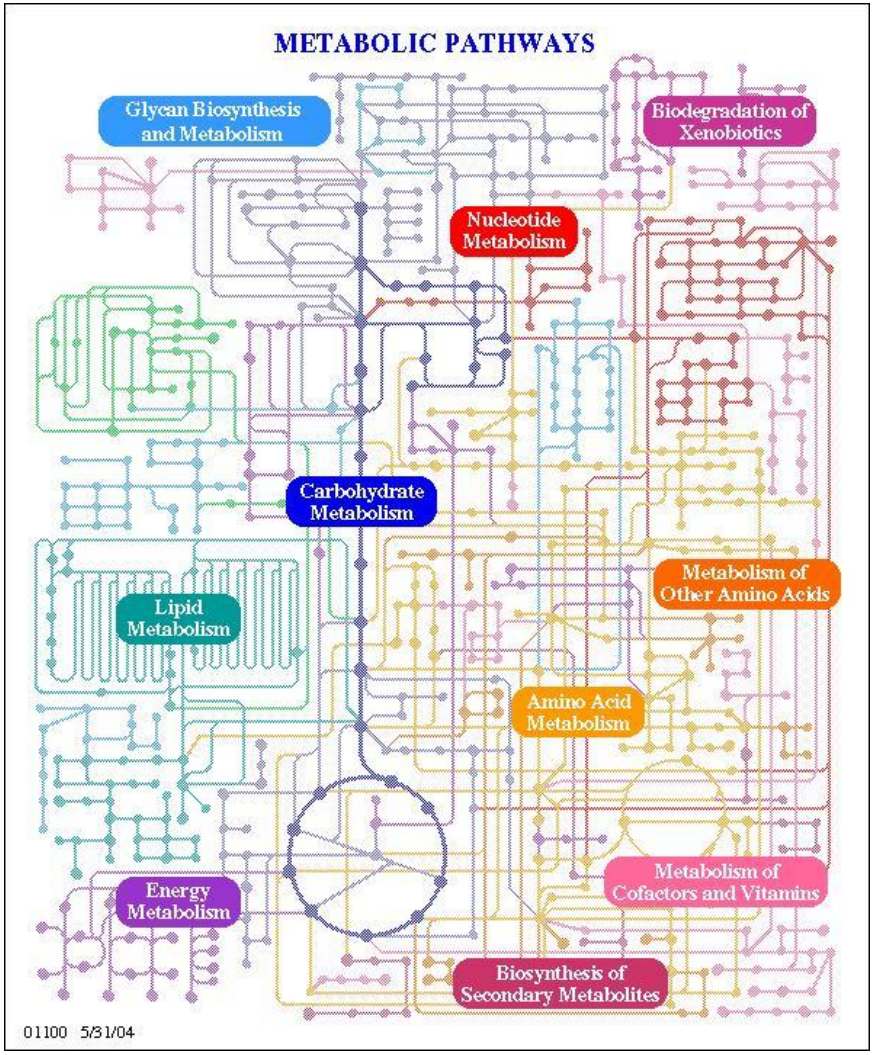
- Biologically-inspired computation, e.g., genetic algorithms and neural networks
- However, application of neural networks to solve some biological problem, could be called bioinformatics
- What about DNA computing?

Related concepts

- Computational biology
  - Application of computing to biology (broad definition)
  - Often used interchangeably with bioinformatics
- Biometry: the statistical analysis of biological data
- Biophysics: "an interdisciplinary field which applies techniques from the physical sciences to understanding biological structure and function" -- British Biophysical Society
- Mathematical biology “tackles biological problems, but the methods it uses to tackle them need not be numerical and need not be implemented in software or hardware.” -- Damian Counsell

Related concepts

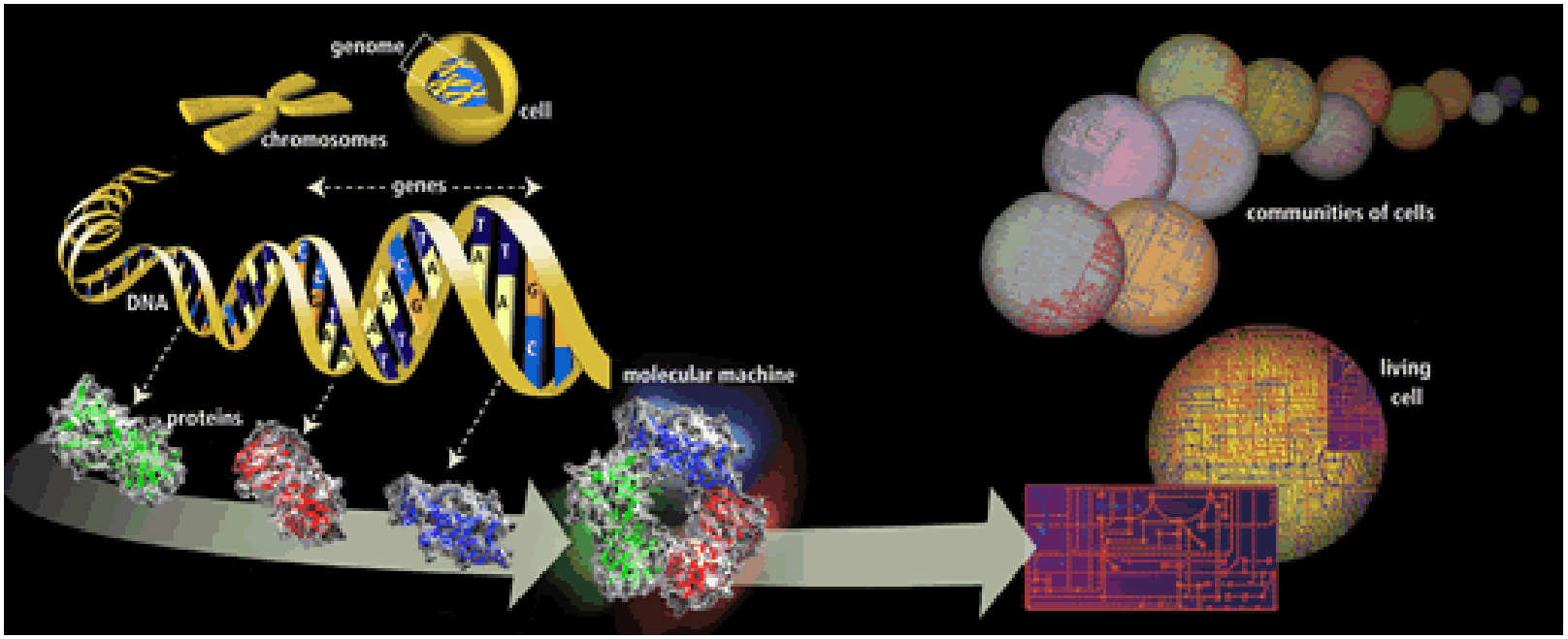
- Systems biology
  - “biology of networks”
  - integrating different levels of information to understand how biological systems work



Overview of metabolic pathways in KEGG database, www.genome.jp/kegg/

Biological background

- Molecular Biology Primer: www.bioalgorithms.info



Sequence Alignment (chapter 6)

- *The biological problem*
- Global alignment
- Local alignment
- Multiple alignment

Background: comparative genomics

- Basic question in biology: *what properties are shared among organisms?*
- Genome sequencing allows comparison of organisms at DNA and protein levels
- Comparisons can be used to
  - Find evolutionary relationships between organisms
  - Identify functionally conserved sequences
  - Identify corresponding genes in human and model organisms: develop models for human diseases

Homologs

- Two genes or characters g_B and g_C evolved from the same ancestor g_A are called *homologs*
- Homologs usually exhibit conserved functions
- Close evolutionary relationship => expect a high number of homologs

g_A = agtgtccgttaagtgcgttc

g_B = agtgccgttaaagttgtacgtc

g_C = ctgactgtttgtggttc

Sequence similarity

- Intuitively, similarity of two sequences refers to the degree of match between corresponding positions in sequence

```
agtgccgttaaagttgtacgtc
  |   |  ||      |   |
ctgactgtttgtggttc
```

- What about sequences that differ in length?

Similarity vs homology

- Sequence similarity is not sequence homology
  - If the two sequences g_B and g_C have accumulated enough mutations, the similarity between them is likely to be low

#mutations	
0	agtgtccgttaagtgcgttc
1	agtgtccgttatagtgcgttc
2	agtgtccgcttatagtgcgttc
4	agtgtccgcttaagggcgttc
8	agtgtccgcttcaaggggcgt
16	gggccgttcatgggggt
32	gcagggcgtcactgagggct

#mutations	
64	acagtccgttcgggctattg
128	cagagcactaccgc
256	cacgagtaagatatagct
512	taatcgtgata
1024	acccttatctacttcctggagtt
2048	agcgacctgcccaa
4096	caaac

Homology is more difficult to detect over greater evolutionary distances.

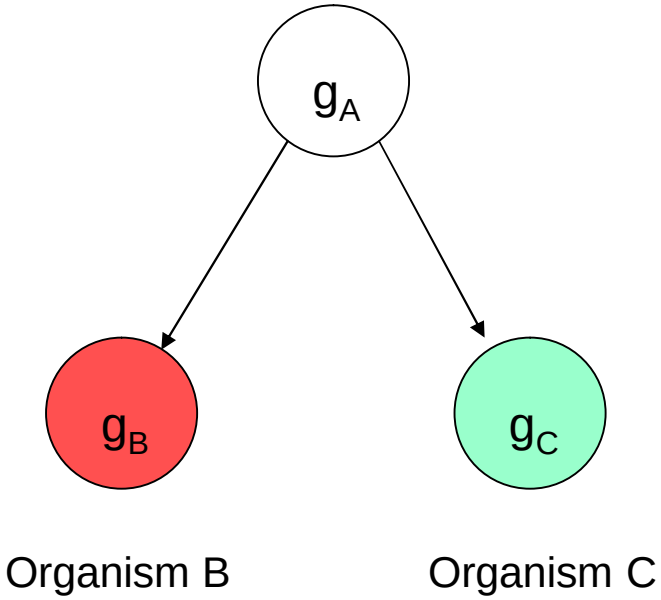
Similarity vs homology (2)

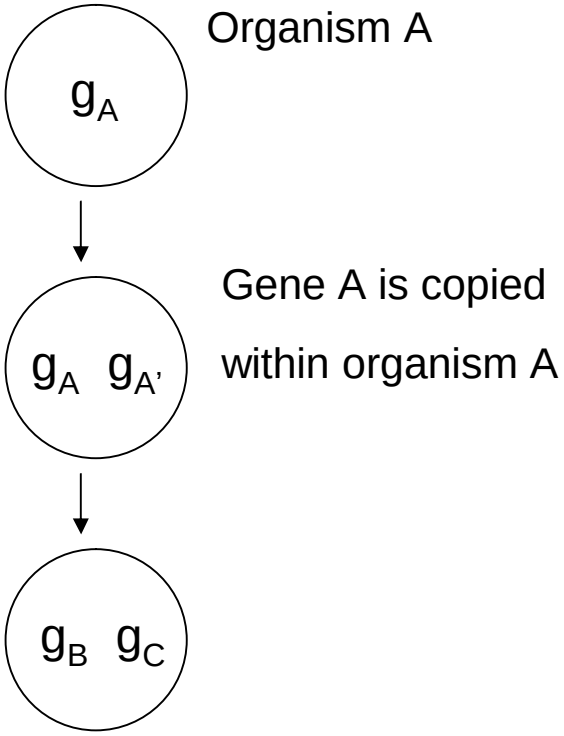
- Sequence similarity can occur by chance
  - *Similarity does not imply homology*

- Similarity is an expected consequence of homology

Orthologs and paralogs

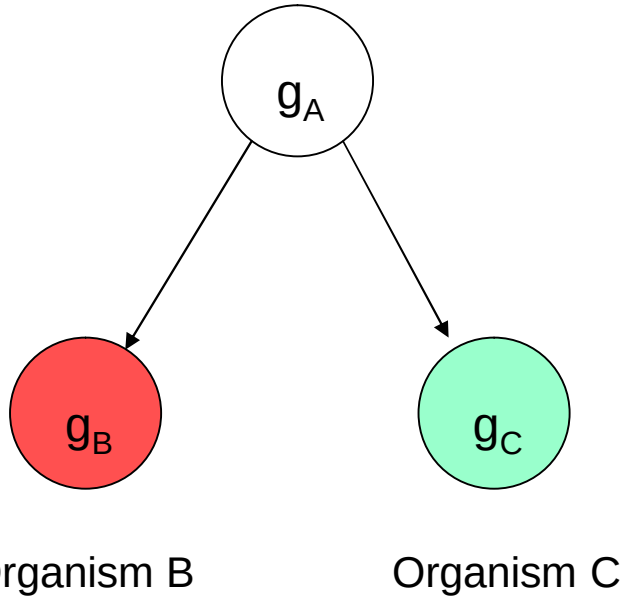
- We distinguish between two types of homology
  - Orthologs: homologs from two different species
  - Paralogs: homologs within a species

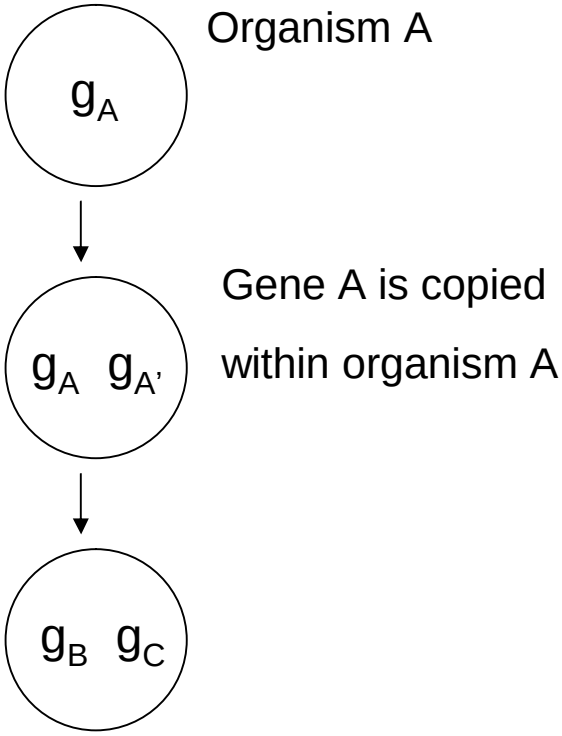




Orthologs and paralogs (2)

- Orthologs typically retain the original function
- In paralogs, one copy is free to mutate and acquire new function (no selective pressure)





Sequence alignment

- Alignment specifies which positions in two sequences match

```
acgtctag   acgtctag    acgtctag
||            |||||    || |||||
actctag-   -actctag    ac-tctag
```

2 matches	5 matches	7 matches
5 mismatches	2 mismatches	0 mismatches
1 not aligned	1 not aligned	1 not aligned

Mutations: Insertions, deletions and substitutions

Indel: insertion or deletion of a base with respect to the ancestor sequence

```
acgtctag
   |||||
-actctag
```

Mismatch: substitution (point mutation) of a single base

- Insertions and/or deletions are called *indels*
  - *We can't tell whether the ancestor sequence had a base or not at indel position*

Problems

- What sorts of alignments should be considered?
- How to score alignments?
- How to find optimal or good scoring alignments?
- How to evaluate the statistical significance of scores?

In this course, we discuss the first three problems.

Course *Biological sequence analysis* tackles all four in-depth.

Sequence Alignment (chapter 6)

- The biological problem
- *Global alignment*
- Local alignment
- Multiple alignment

Global alignment

- Problem: find optimal scoring alignment between two sequences (Needleman & Wunsch 1970)
- We give score for each position in alignment
  - Identity (match) +1
  - Substitution (mismatch) -μ
  - Indel -δ

```
WHAT
||
WH-Y
```

$$S(\text{WHAT/WH-Y}) = 1 + 1 - \delta - \mu$$

Representing alignments and scores

```
WHAT
||
WH-Y
```

	-	W	H	A	T
-					
W		X			
H			X	X	
Y					X

Representing alignments and scores

WHAT

||

WH-Y

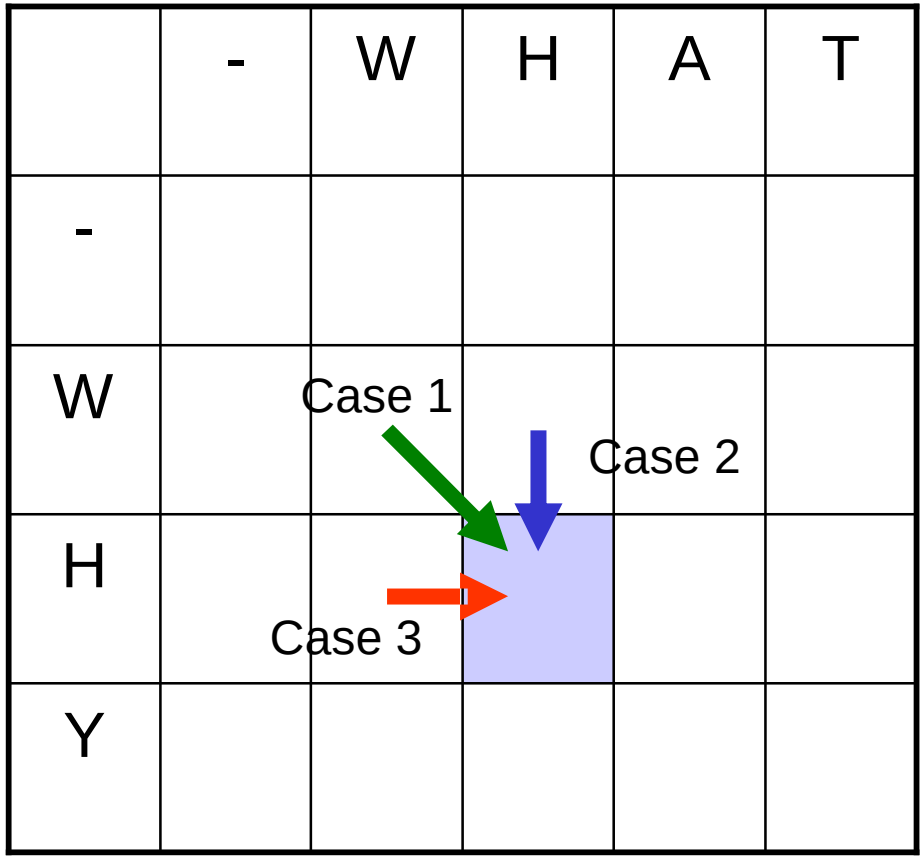
Global alignment score $S_{3,4} = 2-\delta-\mu$

	-	W	H	A	T
-	0				
W		1			
H			2	2-δ	
Y					2-δ-μ

Dynamic programming

- How to find the optimal alignment?
- We use previous solutions for optimal alignments of smaller subsequences
- This general approach is known as dynamic programming

Filling the alignment matrix



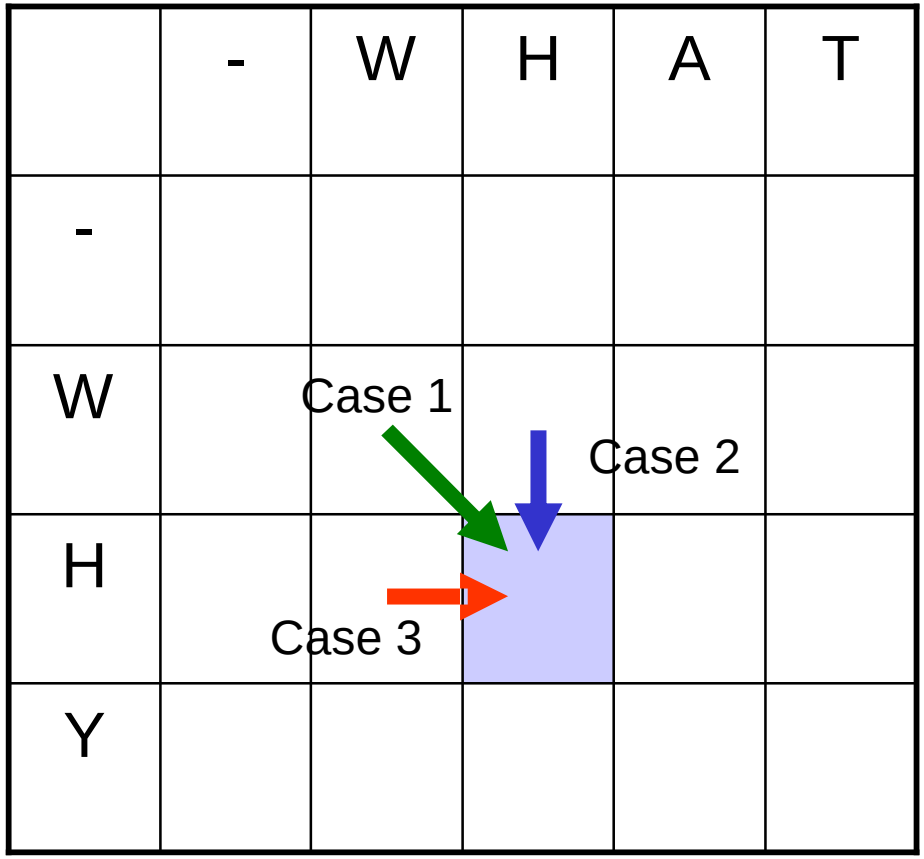
Consider the alignment process at shaded square.

Case 1. Align H against H (match or substitution).

Case 2. Align H in WHY against – (indel) in WHAT.

Case 3. Align H in WHAT against – (indel) in WHY.

Filling the alignment matrix (2)



Scoring the alternatives.

Case 1. $S_{2,2} = S_{1,1} + s(2, 2)$

Case 2. $S_{2,2} = S_{1,2} - \delta$

Case 3. $S_{2,2} = S_{2,1} - \delta$

$s(i, j) = 1$ for matching positions,

$s(i, j) = -\mu$ for substitutions.

Choose the case (path) that yields the maximum score.

Keep track of path choices.

Global alignment: formal development

$A = a_1a_2a_3\ldots a_n$,

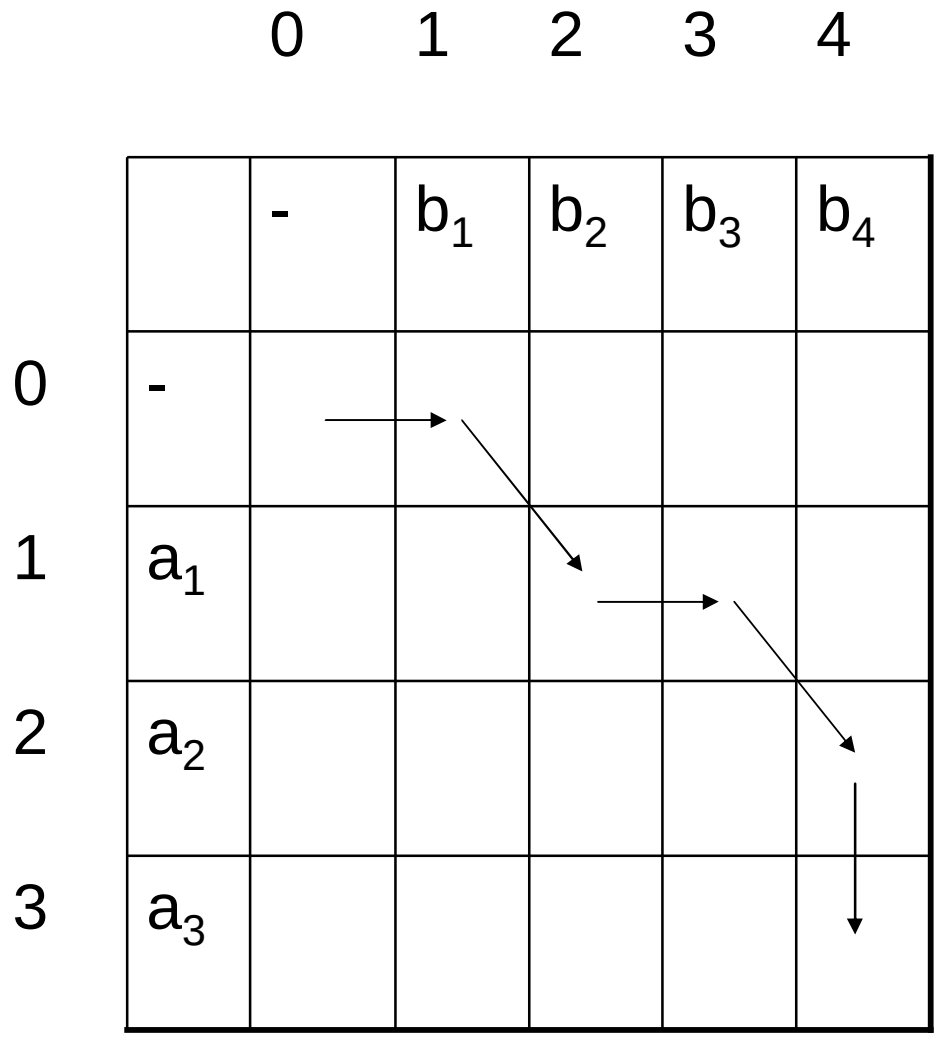
$B = b_1b_2b_3\ldots b_m$

b_1	b_2	b_3	b_4	-
-	a_1	-	a_2	a_3

- Any alignment can be written as a unique path through the matrix

- Score for aligning A and B up to positions i and j:

$S_{i,j} = S(a_1a_2a_3\ldots a_i, b_1b_2b_3\ldots b_j)$



Scoring partial alignments

- Alignment of $A = a_1a_2a_3\ldots a_n$ with $B = b_1b_2b_3\ldots b_m$ can end in three ways
  - Case 1: $(a_1a_2\ldots a_{i-1})\ a_i$

    $(b_1b_2\ldots b_{j-1})\ b_j$
  - Case 2: $(a_1a_2\ldots a_{i-1})\ a_i$

    $(b_1b_2\ldots b_j)$ -
  - Case 3: $(a_1a_2\ldots a_i)$ –

    $(b_1b_2\ldots b_{j-1})\ b_j$

Scoring alignments

- Scores for each case:
  - Case 1: $(a_1 a_2 \ldots a_{i-1})\ a_i$
    $(b_1 b_2 \ldots b_{j-1})\ b_j$
  - Case 2: $(a_1 a_2 \ldots a_{i-1})\ a_i$
    $(b_1 b_2 \ldots b_j)$ –
  - Case 3: $(a_1 a_2 \ldots a_i)$ –
    $(b_1 b_2 \ldots b_{j-1})\ b_j$

$$s(a_i, b_j) = \begin{cases} +1 \text{ if } a_i = b_j \\ -\mu \text{ otherwise} \end{cases}$$

$$s(a_i, -) = s(-, b_j) = -\delta$$

Scoring alignments (2)

- First row and first column correspond to initial alignment against indels:

  $S(i, 0) = -i\,\delta$

  $S(0, j) = -j\,\delta$

- Optimal global alignment score $S(A, B) = S_{n,m}$

		0	1	2	3	4
		-	b_1	b_2	b_3	b_4
0	-	0	$-\delta$	-2δ	-3δ	-4δ
1	a_1	$-\delta$				
2	a_2	-2δ				
3	a_3	-3δ				

Algorithm for global alignment

```
Input sequences A, B, n = |A|, m = |B|
Set S_{i,0} := -δi for all i
Set S_{0,j} := -δj for all j
for i := 1 to n
   for j := 1 to m
      S_{i,j} := max{S_{i-1,j} – δ, S_{i-1,j-1} + s(a_i,b_j), S_{i,j}-1 – δ}
   end
end
```

Algorithm takes O(nm) time and space.

Global alignment: example

$\mu = 1$

$\delta = 2$

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2					
T	-4					
C	-6					
G	-8					
T	-10					?

Global alignment: example (2)

$\mu = 1$

$\delta = 2$

```
ATCGT-
 | ||
-TGGTG
```

	-	T	G	G	T	G
-	0	-2	-4	-6	-8	-10
A	-2	-1	-3	-5	-7	-9
T	-4	-1	-2	-4	-4	-6
C	-6	-3	-2	-3	-5	-5
G	-8	-5	-2	-1	-3	-4
T	-10	-7	-4	-3	0	-2

Sequence Alignment (chapter 6)

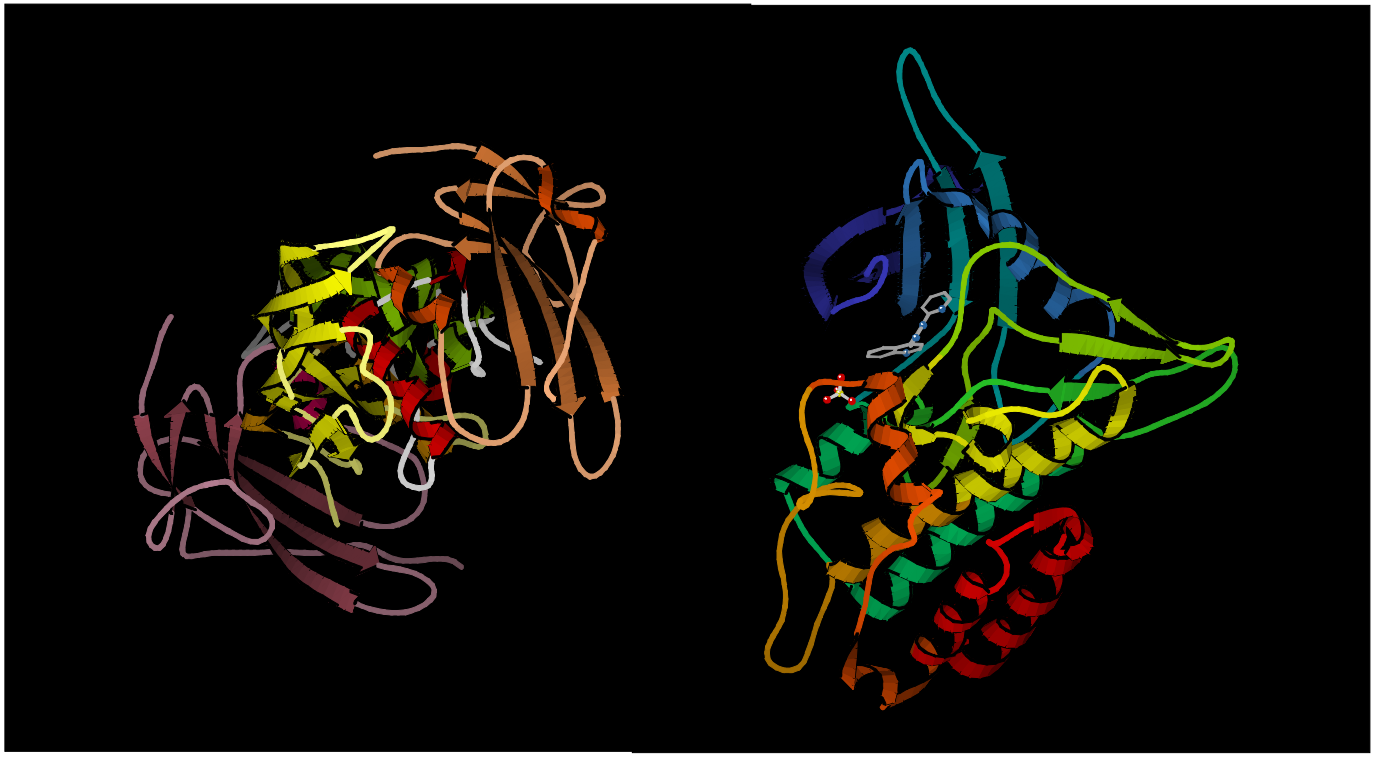
- The biological problem
- Global alignment
- *Local alignment*
- Multiple alignment

Local alignment: rationale

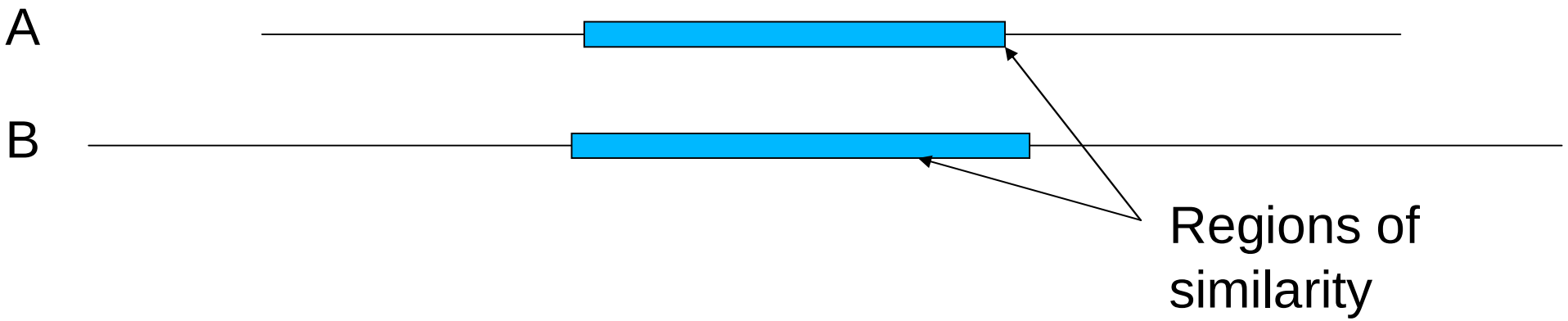
- Otherwise dissimilar proteins may have local regions of similarity
  -> Proteins may share a function

Human bone morphogenic protein receptor type II precursor (left) has a 300 aa region that resembles 291 aa region in TGF-β receptor (right).

The shared function here is protein kinase.



Local alignment: rationale



- Global alignment would be inadequate
- Problem: find the highest scoring *local* alignment between two sequences
- Previous algorithm with minor modifications solves this problem (Smith & Waterman 1981)

From global to local alignment

- Modifications to the global alignment algorithm
  - Look for the highest-scoring path **in** the alignment matrix (not necessarily through the matrix)
  - Allow preceding and trailing indels without penalty

Scoring local alignments

$A = a_1a_2a_3\ldots a_n$, $B = b_1b_2b_3\ldots b_m$

Let I and J be intervals (substrings) of A and B, respectively: $I \subset A$, $J \subset B$

Best local alignment score:

$$M(A, B) = \max\{S(I, J) : I \subset A, J \subset B\}$$

where S(I, J) is the score for substrings I and J.

Allowing preceding and trailing indels

- First row and column initialised to zero:

  $M_{i,0} = M_{0,j} = 0$

		0	1	2	3	4
		-	b_1	b_2	b_3	b_4
0	-	0	0	0	0	0
1	a_1	0				
2	a_2	0				
3	a_3	0				

```
b1  b2  b3
-   -   a1
```

Recursion for local alignment

- $M_{i,j} = \max \{$
  $M_{i-1,j-1} + s(a_i, b_i)$,
  $M_{i-1,j} - \delta$,
  $M_{i,j-1} - \delta$,
  0
  $\}$

	-	T	G	G	T	G
-	0	0	0	0	0	0
A	0	0	0	0	0	0
T	0	1	0	0	1	0
C	0	0	0	0	0	0
G	0	0	1	1	0	1
T	0	1	0	0	2	0

Finding best local alignment

- Optimal score is the highest value in the matrix

$$M(A,B) = \max\{S(I,J) : I \subset A, J \subset B\}$$

$$= \max_{i,j} M_{i,j}$$

- Best local alignment can be found by backtracking from the highest value in M

	-	T	G	G	T	G
-	0	0	0	0	0	0
A	0	0	0	0	0	0
T	0	1	0	0	1	0
C	0	0	0	0	0	0
G	0	0	1	1	0	1
T	0	1	0	0	2	0

Local alignment: example

		0	1	2	3	4	5	6	7	8	9	10
		-	G	G	C	T	C	A	A	T	C	A
0	-	0	0	0	0	0	0	0	0	0	0	0
1	A	0										
2	C	0										
3	C	0										
4	T	0										
5	A	0										
6	A	0										
7	G	0										
8	G	0										

Local alignment: example

Scoring

Match: +2

Mismatch: -1

Indel: -2

		0	1	2	3	4	5	6	7	8	9	10
		-	G	G	C	T	C	A	A	T	C	A
0	-	0	0	0	0	0	0	0	0	0	0	0
1	A	0	0	0	0	0	0	2	2	0	0	2
2	C	0	0	0	2	0	2	0	1	1	2	0
3	C	0	0	0	2	1	2	1	0	0	3	1
4	T	0	0	0	0	4	2	1	0	2	1	2
5	A	0	0	0	0	2	3	4	3	1	1	3
6	A	0	0	0	0	0	1	5	6	4	2	3
7	G	0	2	2	0	0	0	3	4	5	3	1
8	G	0	2	4	2	0	0	1	2	3	4	2

```
C T – A A
C T C A A
```

Non-uniform mismatch penalties

- We used uniform penalty for mismatches:

  s(’A’, ’C’) = s(’A’, ’G’) = … = s(’G’, ’T’) = μ

- Transition mutations (A->G, G->A, C->T, T->C) are approximately twice as frequent than transversions (A->T, T->A, A->C, G->T)
  - use non-uniform mismatch penalties

	A	C	G	T
A	1	-1	-0.5	-1
C	-1	1	-1	-0.5
G	-0.5	-1	1	-1
T	-1	-0.5	-1	1

Gaps in alignment

- Gap is a succession of indels in alignment

```
C T - - - A A
C T C G C A A
```

- Previous model scored a length k gap as $w(k) = -k\delta$
- Replication processes may produce longer stretches of insertions or deletions
  - In coding regions, insertions or deletions of codons may preserve functionality

Gap open and extension penalties (2)

- We can design a score that allows the penalty opening gap to be larger than extending the gap:

$$w(k) = -\alpha - \beta(k - 1)$$

- Gap open cost α, Gap extension cost β
- Our previous algorithm can be extended to use w(k) (not discussed on this course)

Sequence Alignment (chapter 6)

- The biological problem
- Global alignment
- Local alignment
- *Multiple alignment*

Multiple alignment

- Consider a set of n sequences on the right
  - Orthologous sequences from different organisms
  - Paralogs from multiple duplications
- How can we study relationships between these sequences?

aggcgagctgcgagtgcta
cgttagattgacgctgac
ttccggctgcgac
gacacggcgaacgga
agtgtgcccgacgagcgaggac
gcgggctgtgagcgcta
aagcggcctgtgtgcccta
atgctgctgccagtgta
agtcgagccccgagtgc
agtccgagtcc
actcggtgc

Optimal alignment of three sequences

- Alignment of A = $a_1a_2\ldots a_i$ and B = $b_1b_2\ldots b_j$ can end either in (-, b_j), (a_i, b_j) or (a_i, -)
- $2^2 – 1 = 3$ alternatives
- Alignment of A, B and C = $c_1c_2\ldots c_k$ can end in $2^3 – 1$ ways: (a_i, -, -), (-, b_j, -), (-, -, c_k), (-, b_j, c_k), (a_i, -, c_k), (a_i, b_j, -) or (a_i, b_j, c_k)
- Solve the recursion using three-dimensional dynamic programming matrix: $O(n^3)$ time and space
- Generalizes to n sequences but impractical with moderate number of sequences

Multiple alignment in practice

- In practice, real-world multiple alignment problems are usually solved with heuristics
- Progressive multiple alignment
  - Choose two sequences and align them
  - Choose third sequence w.r.t. two previous sequences and align the third against them
  - Repeat until all sequences have been aligned
  - Different options how to choose sequences and score alignments

Multiple alignment in practice

- Profile-based progressive multiple alignment: CLUSTALW
  - Construct a distance matrix of all pairs of sequences using dynamic programming
  - Progressively align pairs in order of decreasing similarity
  - CLUSTALW uses various heuristics to contribute to accuracy

Additional material

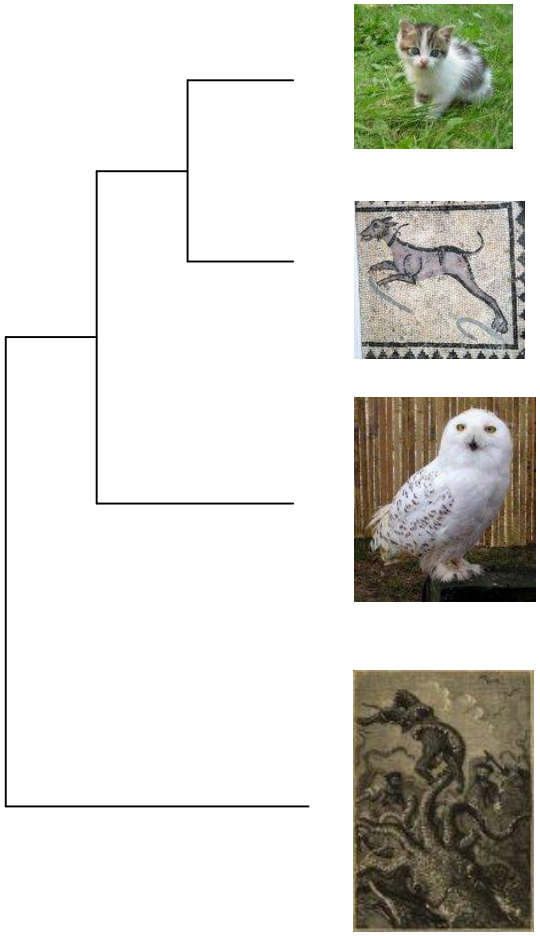
- R. Durbin, S. Eddy, A. Krogh, G. Mitchison: Biological sequence analysis
- Course Biological sequence analysis in Spring 2007

Inferring the Past: Phylogenetic Trees (chapter 12)

- *The biological problem*
- Parsimony and distance methods
- Models for mutations and estimation of distances
- Maximum likelihood methods

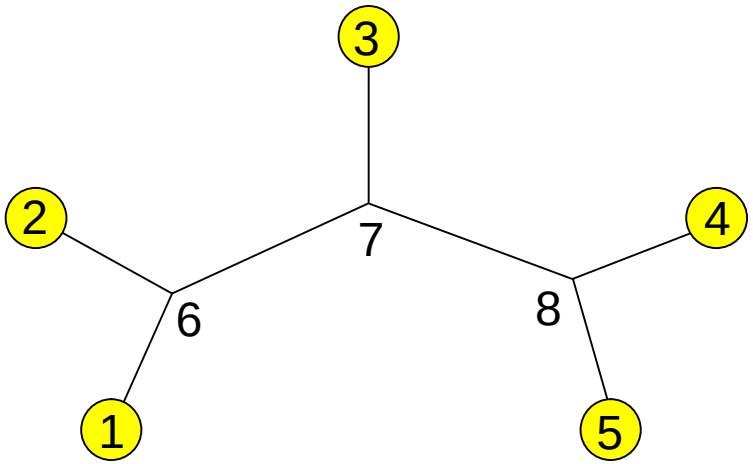
Phylogeny

- We want to study ancestor-descendant relationships, or *phylogeny*, among groups of organisms
- Groups are called *taxa* (singular: *taxon*)
- Organisms are usually called *operational taxonomic units* or *OTU*s in the context of phylogeny



Phylogenetic trees

- Leaves (external nodes) ~ species, observed (OTUs)
- Internal nodes ~ ancestral species/divergence events, not observed
- Unrooted tree does not specify ancestor-descendant relationships beyond the observation ”leaves are not ancestors”

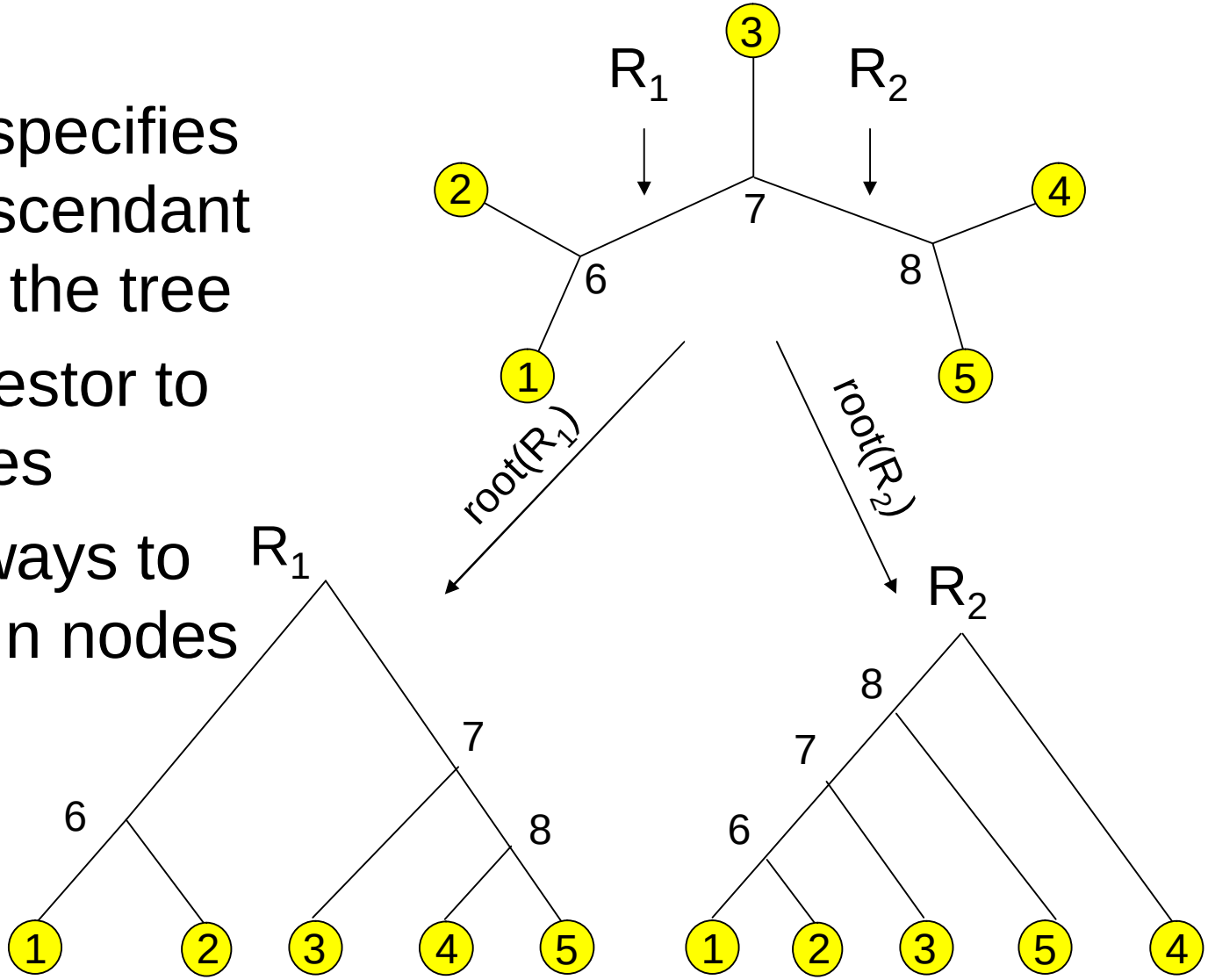


Unrooted tree with 5 leaves and 3 internal nodes.

Is node 7 ancestor of node 6?

Phylogenetic trees

- Rooting a tree specifies all ancestor-descendant relationships in the tree
- Root is the ancestor to the other species
- There are n-1 ways to root a tree with n nodes



Questions

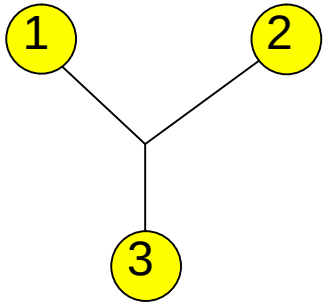
- Can we enumerate all possible phylogenetic trees for *n* species (or sequences?)
- How to score a phylogenetic tree with respect to data?
- How to find the best phylogenetic tree given data?

Finding the best phylogenetic tree: naive method

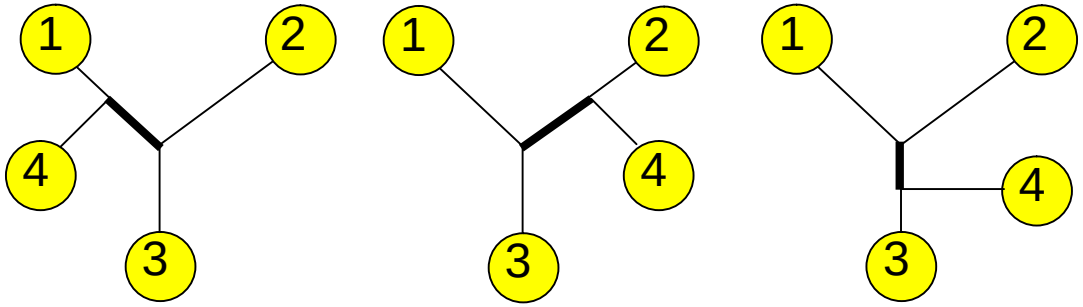
- How can we find the phylogenetic tree that best represents the data?
- Naive method: enumerate all possible trees
- How many different trees are there of *n* species?
- Denote this number by b_n

Enumerating unordered trees

- Start with the only unordered tree with 3 leaves ($b_3 = 1$)



- Consider all ways to add a leaf node to this tree



- Fourth node can be added to 3 different branches (edges), creating 1 new internal branch
- Total number of branches is n external and n – 3 internal branches
- Unrooted tree with n leaves has 2n – 3 branches

Enumerating unordered trees

- Thus, we get the number of unrooted trees

$$b_n = (2(n-1)-3)b_{n-1} = (2n-5)b_{n-1}$$
$$= (2n-5) * (2n-7) * \ldots * 3 * 1$$
$$= (2n-5)! / ((n-3)!2^{n-3}),\ n > 2$$

- Number of rooted trees b'_n is

$$b'_n = (2n-3)b_n = (2n-3)! / ((n-2)!2^{n-2}),\ n > 2$$

that is, the number of unrooted trees times the number of branches in the trees

Number of possible rooted and unrooted trees

n	B_n	b'_n
3	1	3
4	3	15
5	15	105
6	105	945
7	954	10395
8	10395	135135
9	135135	2027025
10	2027025	34459425
20	2.22E+020	8.20E+021
30	8.69E+036	4.95E+038

Too many trees?

- We can’t construct and evaluate every phylogenetic tree even for a smallish number of species
- Better alternative is to
  - Devise a way to evaluate an individual tree against the data
  - Guide the search using the evaluation criteria to reduce the search space

Inferring the Past: Phylogenetic Trees (chapter 12)

- The biological problem
- *Parsimony and distance methods*
- Models for mutations and estimation of distances
- Maximum likelihood methods

Parsimony method

- The parsimony method finds the tree that explains the observed *sequences* with a minimal number of substitutions
- Method has two steps
  - Compute smallest number of substitutions for a given tree with a *parsimony algorithm*
  - Search for the tree with the minimal number of substitutions

Parsimony: an example

- Consider the following short sequences

1 ACTTT

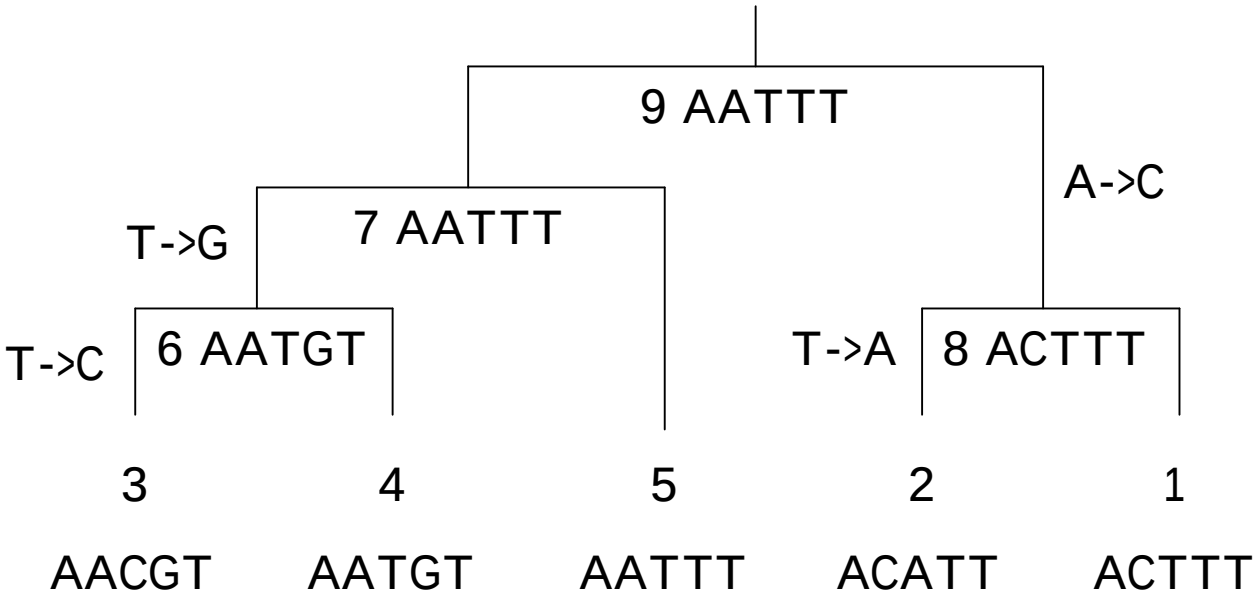
2 ACATT

3 AACGT

4 AATGT

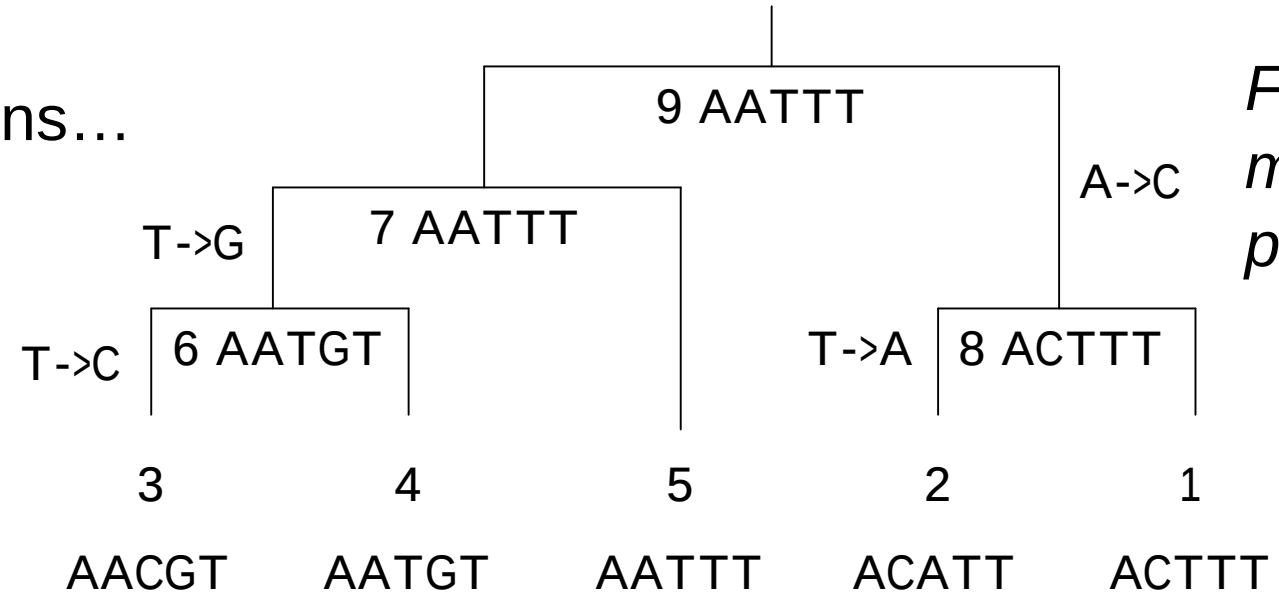
5 AATTT

- There are 105 possible rooted trees for 5 sequences
- Example: which of the following trees explains the sequences with least number of substitutions?



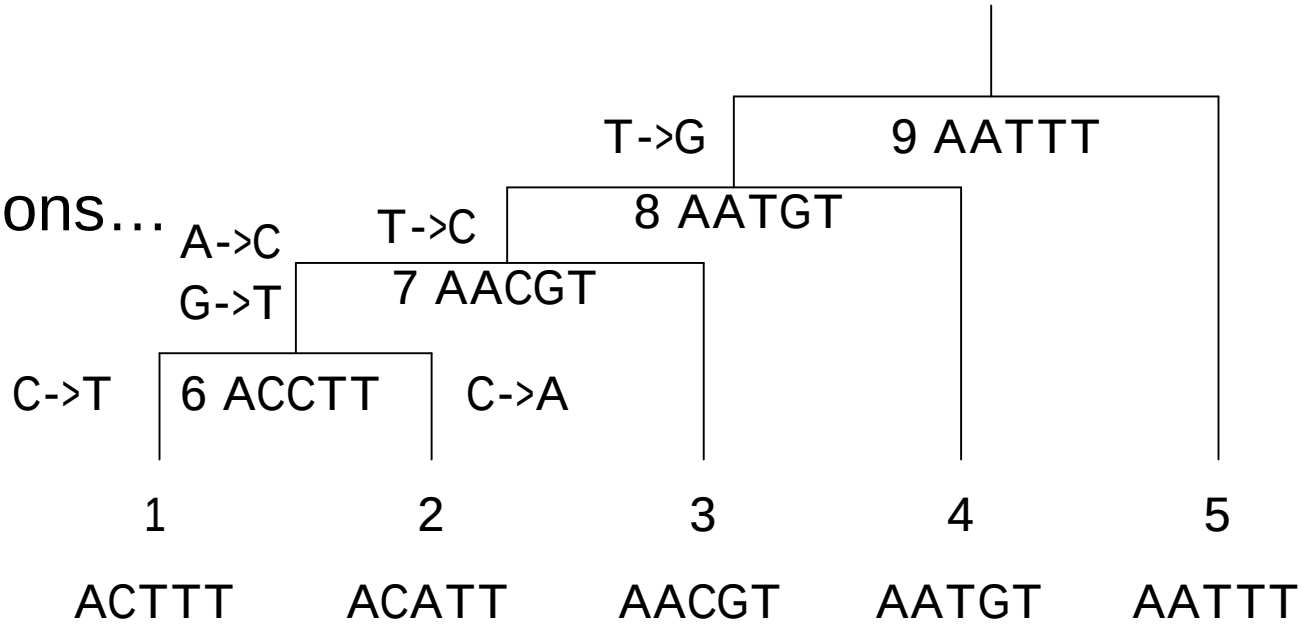
This tree explains the sequences with 4 substitutions

4 substitutions…



First tree is more parsimonious!

6 substitutions…

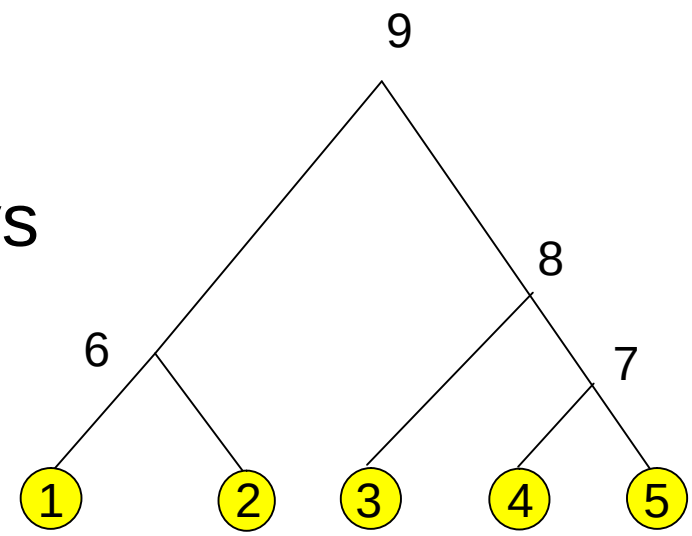


Computing parsimony

- Parsimony treats each site (position in a sequence) independently
- Total parsimony cost is the sum of parsimony costs of each site
- We can compute the minimal parsimony cost for a given tree by
  - First finding out possible assignments at each node, starting from leaves and proceeding towards the root
  - Then, starting from the root, assign a letter at each node, proceeding towards leaves

Labelling tree nodes

- An unrooted tree with *n* leaves contains 2n-2 nodes altogether
- Assign the following labels to nodes in a rooted tree
  - leaf nodes: 1, 2, …, n
  - internal nodes: n+1, n+2, …, 2n-1
  - root node: 2n-1
- The label of a child node is always smaller than the label of the parent node



Parsimony algorithm: first phase

- Find out possible assignments at every node for each site independently. Denote site u in sequence i by $s_{i,u}$

For i := 1, …, n do

  $F_i := \{s_{i,u}\}$ % possible assignments at node i

  $L_i := 0$ % number of substitutions up to node i

For i := n+1, …, 2n-1 do

  Let j and k be the children of node i

  If $F_j \cap F_k = \emptyset$ then $L_i := L_j + L_k + 1$, $F_i := F_j \cup F_k$

    else $L_i := L_j + L_k$, $F_i := F_j \cap F_k$

Parsimony algorithm: first phase

Choose u = 3 (for example)

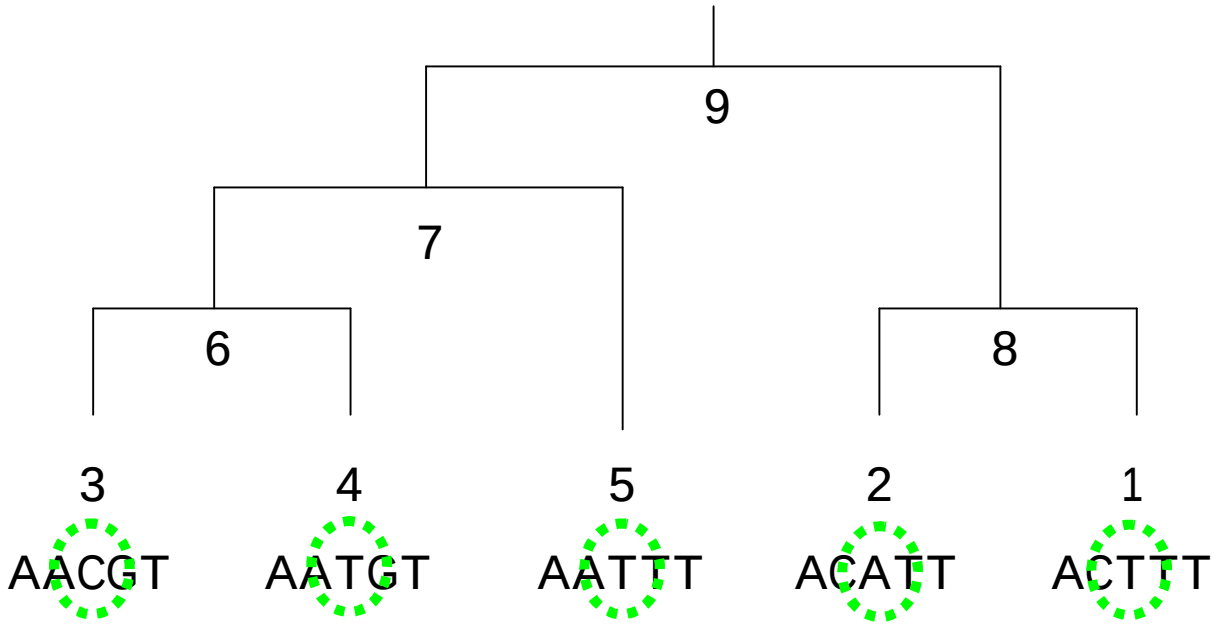
$F_1 := \{T\}$
$L_1 := 0$

$F_2 := \{A\}$
$L_2 := 0$

$F_3 := \{C\}$, $L_3 := 0$

$F_4 := \{T\}$, $L_4 := 0$

$F_5 := \{T\}$, $L_5 := 0$



$F_8 := F_1 \cup F_2 = \{A, T\}$
$L_8 := L_1 + L_2 + 1 = 1$

Parsimony algorithm: first phase

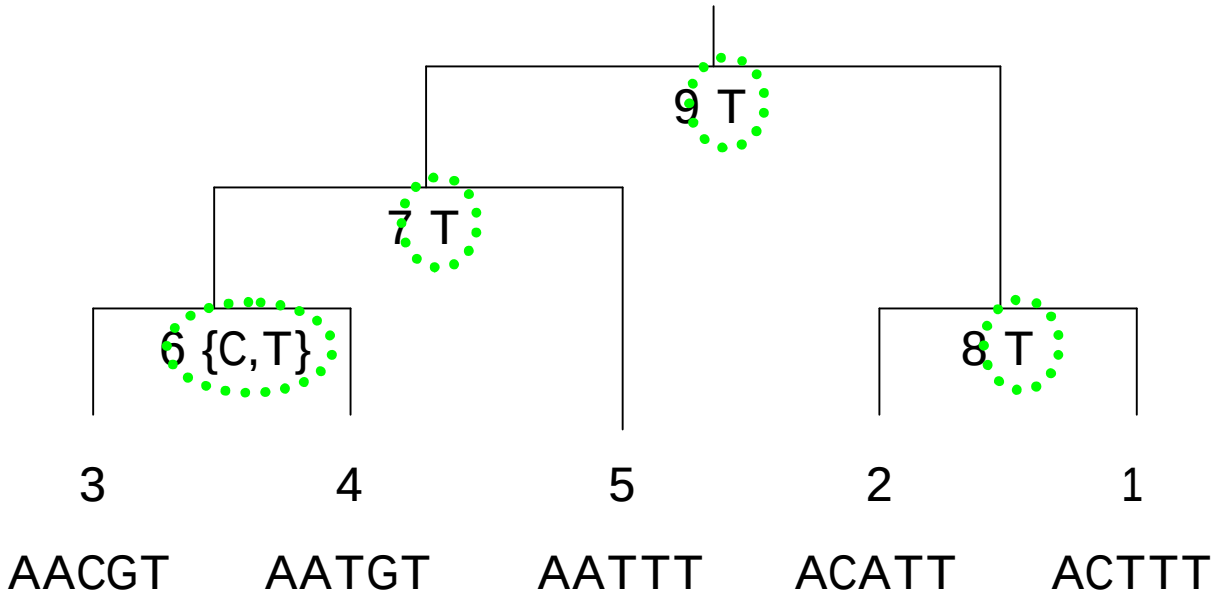
$F_6 := F_3 \cup F_4 = \{C, T\}$

$L_6 := L_3 + L_4 + 1 = 1$

$F_7 := F_5 \cap F_6 = \{T\}$
$L_7 := L_5 + L_6 = 1$

$F_8 := F_1 \cup F_2 = \{A, T\}$
$L_8 := L_1 + L_2 + 1 = 1$

$F_9 := F_7 \cap F_8 = \{T\}$
$L_9 := L_7 + L_8 = 2$



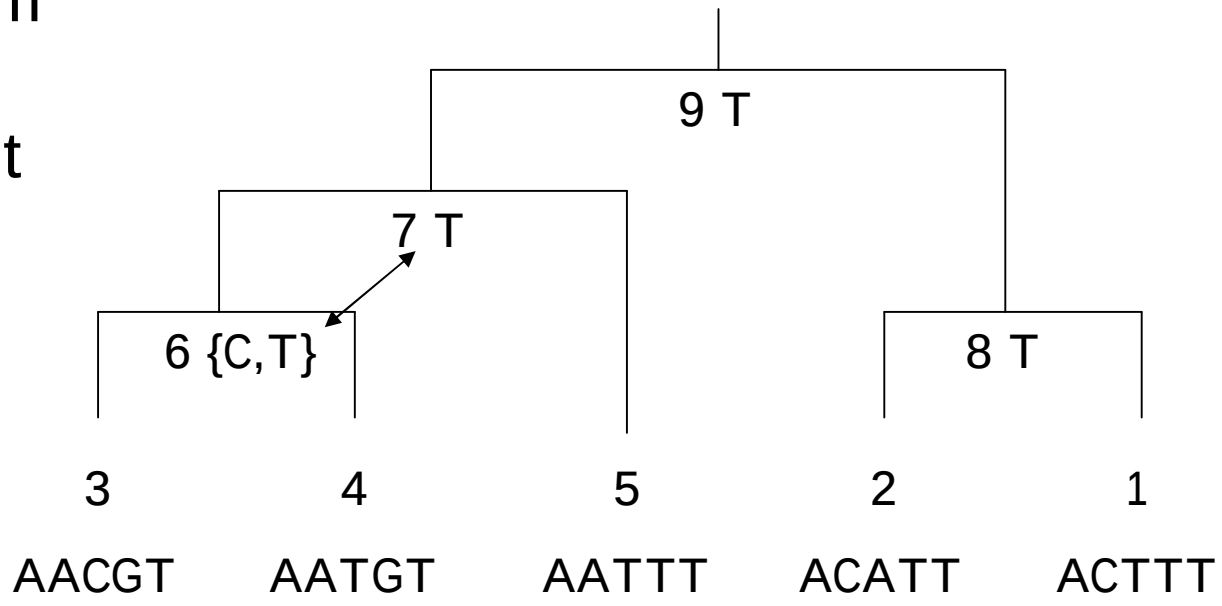
⇨ Parsimony cost for site 3 is 2

Parsimony algorithm: second phase

- Backtrack from the root and assign $x \in F_i$ at each node
- If we assigned y at parent of node i and $y \in F_i$, then assign y
- Else assign $x \in F_i$ by random

Parsimony algorithm: second phase

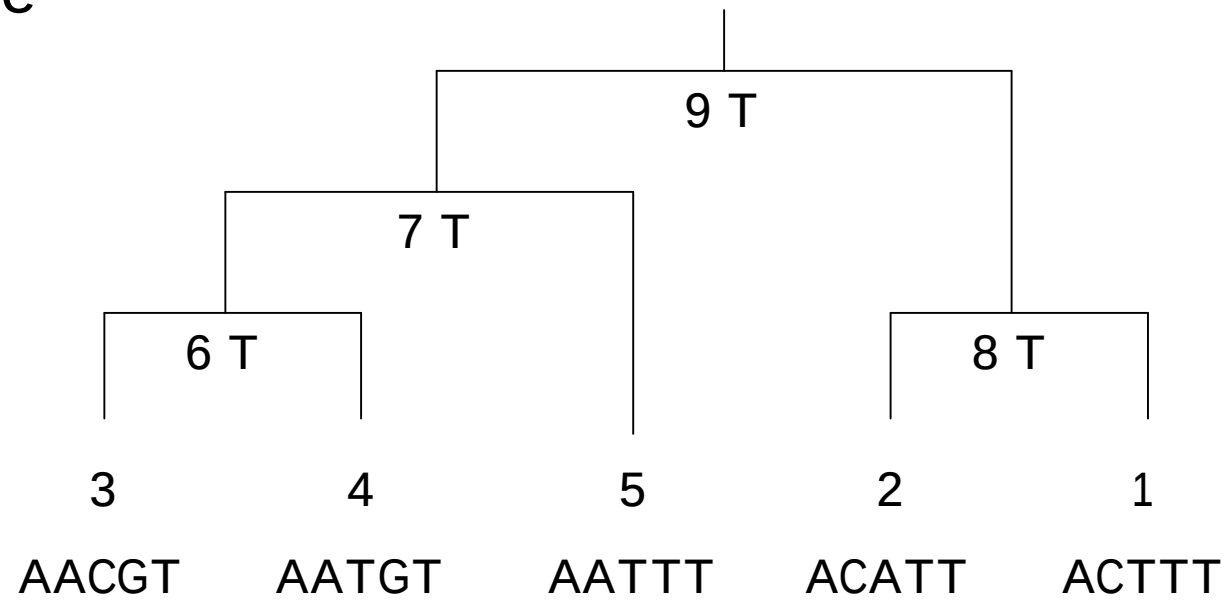
At node 6, the algorithm assigns T because T was assigned to parent node 7 and T $\in F_6$



The other nodes have only one possible letter to assign

Parsimony algorithm

First and second phase are repeated for each site in the sequences,
summing the parsimony costs at each site



Properties of parsimony algorithm

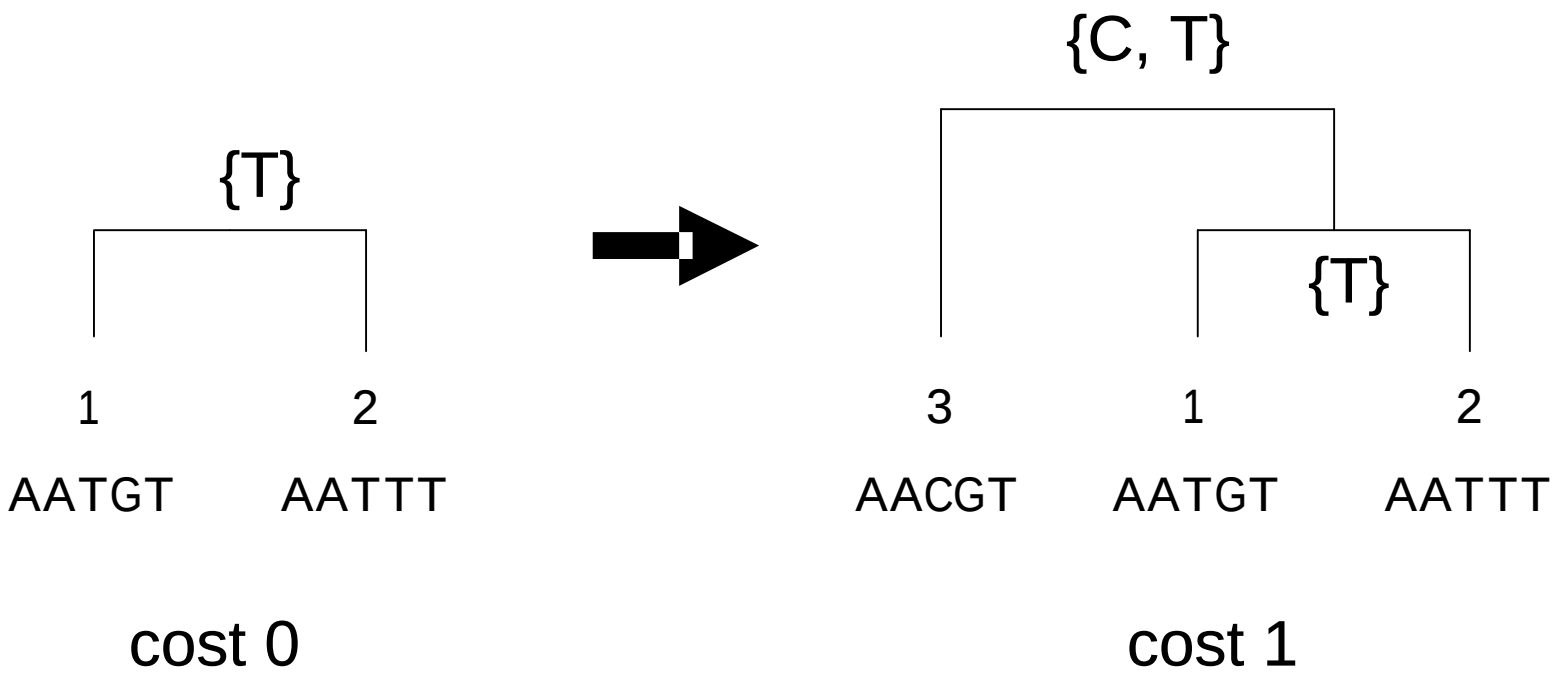
- Parsimony algorithm requires that the sequences are of same length
  - First align the sequences against each other and remove indels
  - Then compute parsimony for the resulting sequences
- Is the most parsimonious tree the correct tree?
  - Not necessarily but it explains the sequences with least number of substitutions
  - We can assume that the probability of having fewer mutations is higher than having many mutations

Finding the most parsimonious tree

- Parsimony algorithm calculates the parsimony cost for a given tree…
- …but we still have the problem of finding the tree with the lowest cost
- Exhaustive search (enumerating all trees) is in general impossible
- More efficient methods exist, for example
  - Probabilistic search
  - Branch and bound

Branch and bound in parsimony

- We can exploit the fact that adding edges to a tree can only increase the parsimony cost



Branch and bound in parsimony

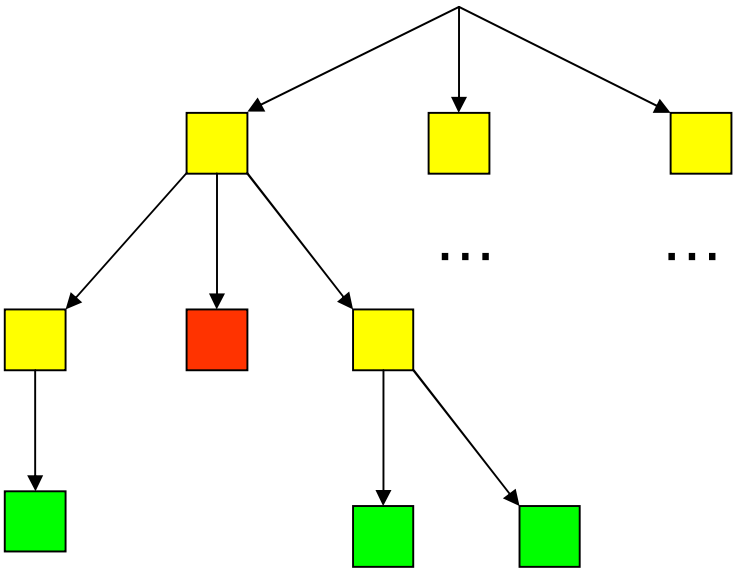
Branch and bound is a general search strategy where

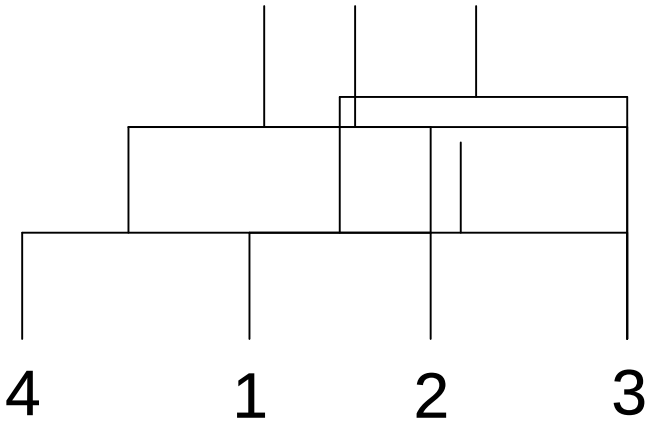
- Each solution is potentially generated
- Track is kept of the best solution found
- If a partial solution cannot achieve better score, we abandon the current search path

In parsimony…

- Start from a tree with 1 sequence
- Add a sequence to the tree and calculate parsimony cost
- If the tree is complete, check if found the best tree so far
- If tree is not complete and cost exceeds best tree cost, do not continue adding edges to this tree

Branch and bound graphically





Partial tree, no best complete tree constructed yet

Complete tree: calculate parsimony cost and store

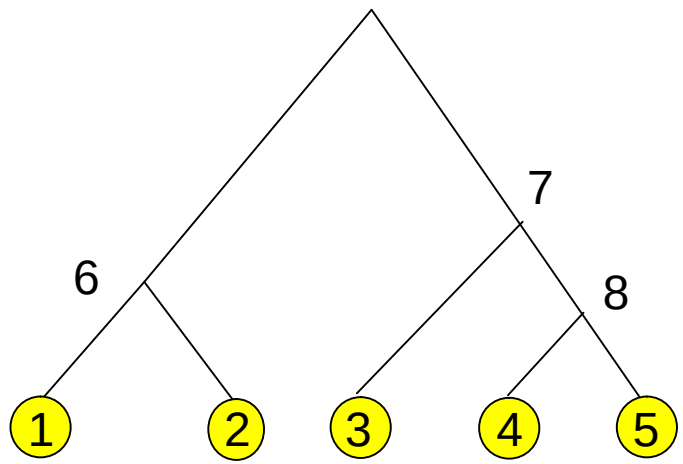
Partial tree, cost exceeds the cost of the best tree this far

Distance methods

- The parsimony method works on sequence (character string) data
- We can also build phylogenetic trees in a more general setting
- *Distance methods* work on a set of pairwise distances d_{ij} for the data
- Distances can be obtained from phenotypes as well as from genotypes (sequences)

Distances in a phylogenetic tree

- Distance matrix D = (d_{ij}) gives pairwise distances for *leaves* of the phylogenetic tree
- In addition, the phylogenetic tree will now specify distances between leaves and internal nodes
  - Denote these with d_{ij} as well



Distance d_{ij} states how far apart species i and j are evolutionary (e.g., number of mismatches in aligned sequences)

Distances in evolutionary context

- Distances d_{ij} in evolutionary context satisfy the following conditions
  - Symmetry: $d_{ij} = d_{ji}$ for each i, j
  - Distinguishability: $d_{ij} \neq 0$ if and only if $i \neq j$
  - Triangle inequality: $d_{ij} \leq d_{ik} + d_{kj}$ for each i, j, k
- Distances satisfying these conditions are called metric
- In addition, evolutionary mechanisms may impose additional constraints on the distances
  - ▷ *additive* and *ultrametric* distances

Additive trees

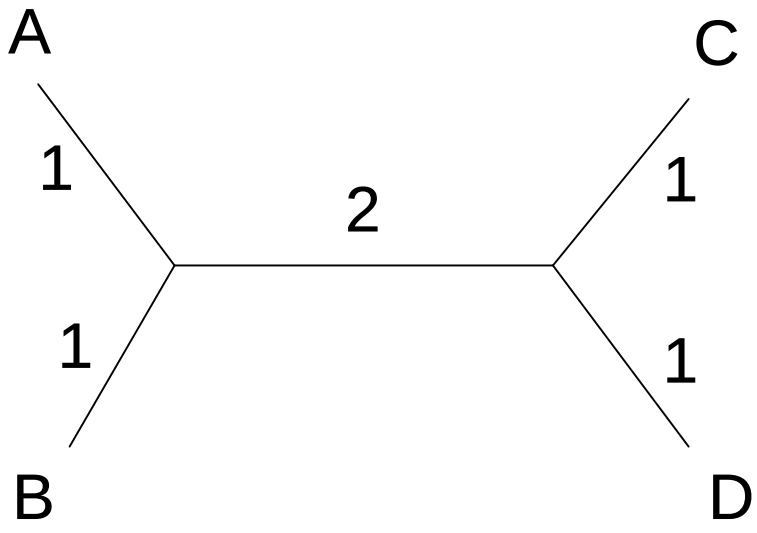
- A tree is called *additive*, if the distance between any pair of leaves (i, j) is the sum of the distances between the leaves and the first node k that they share in the tree

$$d_{ij} = d_{ik} + d_{jk}$$

- ”Follow the path from the leaf i to the leaf j to find the exact distance d_{ij} between the leaves.”

Additive trees: example

	A	B	C	D
A	0	2	4	4
B	2	0	4	4
C	4	4	0	2
D	4	4	2	0



Ultrametric trees

- A rooted additive tree is called a *ultrametric tree*, if the distances between any two leaves i and j, and their common ancestor k are equal

$$d_{ik} = d_{jk}$$

- Edge length d_{ij} corresponds to the time elapsed since divergence of i and j from the common parent
- In other words, edge lengths are measured by a *molecular clock* with a constant rate

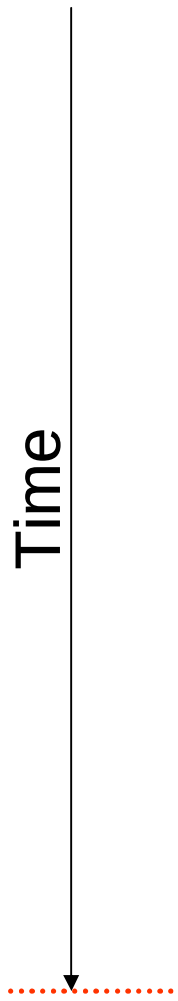
Identifying ultrametric data

- We can identify distances to be ultrametric by the three-point condition:

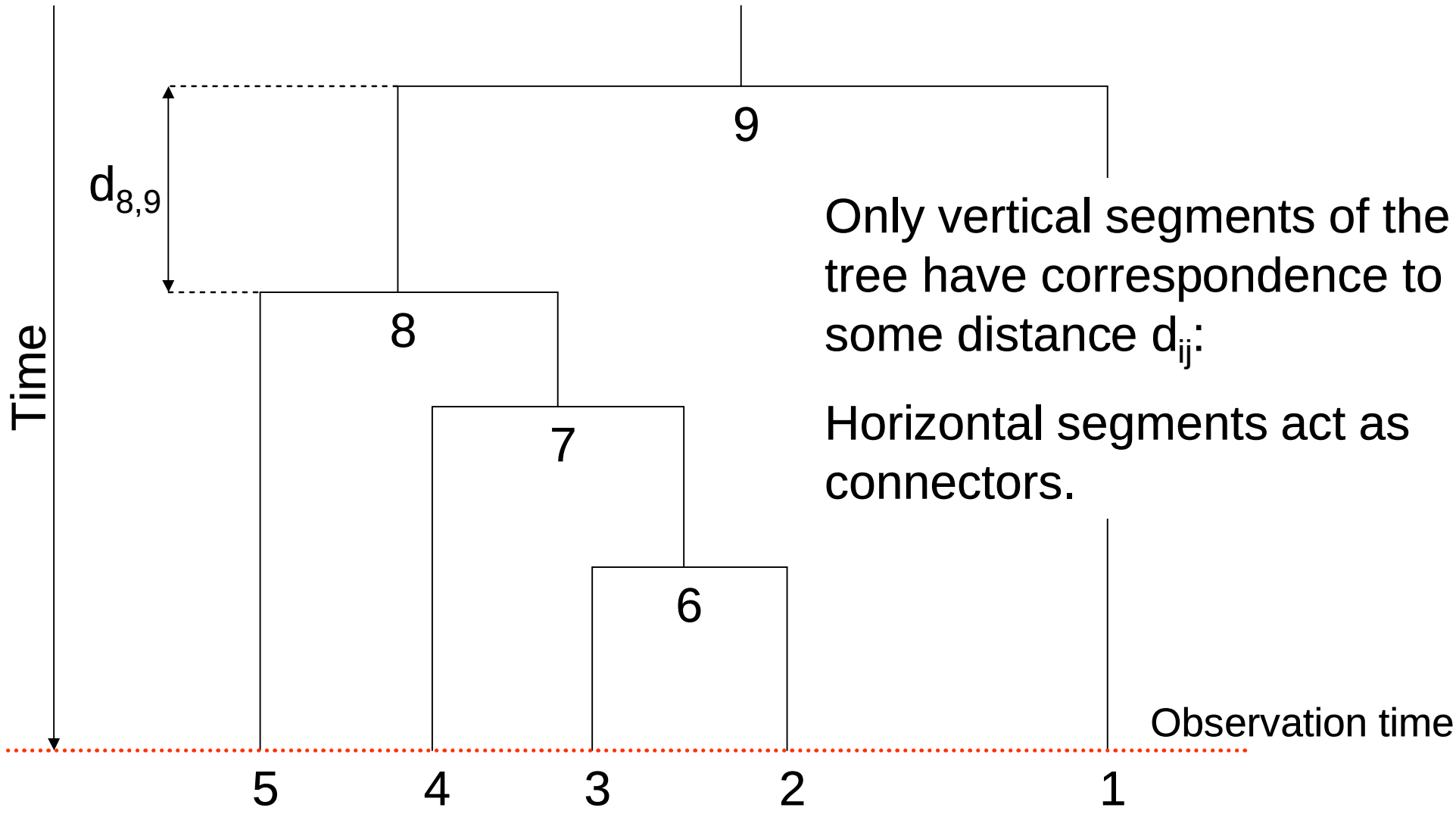
  D corresponds to an ultrametric tree if and only if for any three **sequences** i, j and k, the distances satisfy $d_{ij} \leq \max(d_{ik}, d_{kj})$

- If we find out that the data is ultrametric, we can utilise a simple algorithm to find the corresponding tree

Ultrametric trees



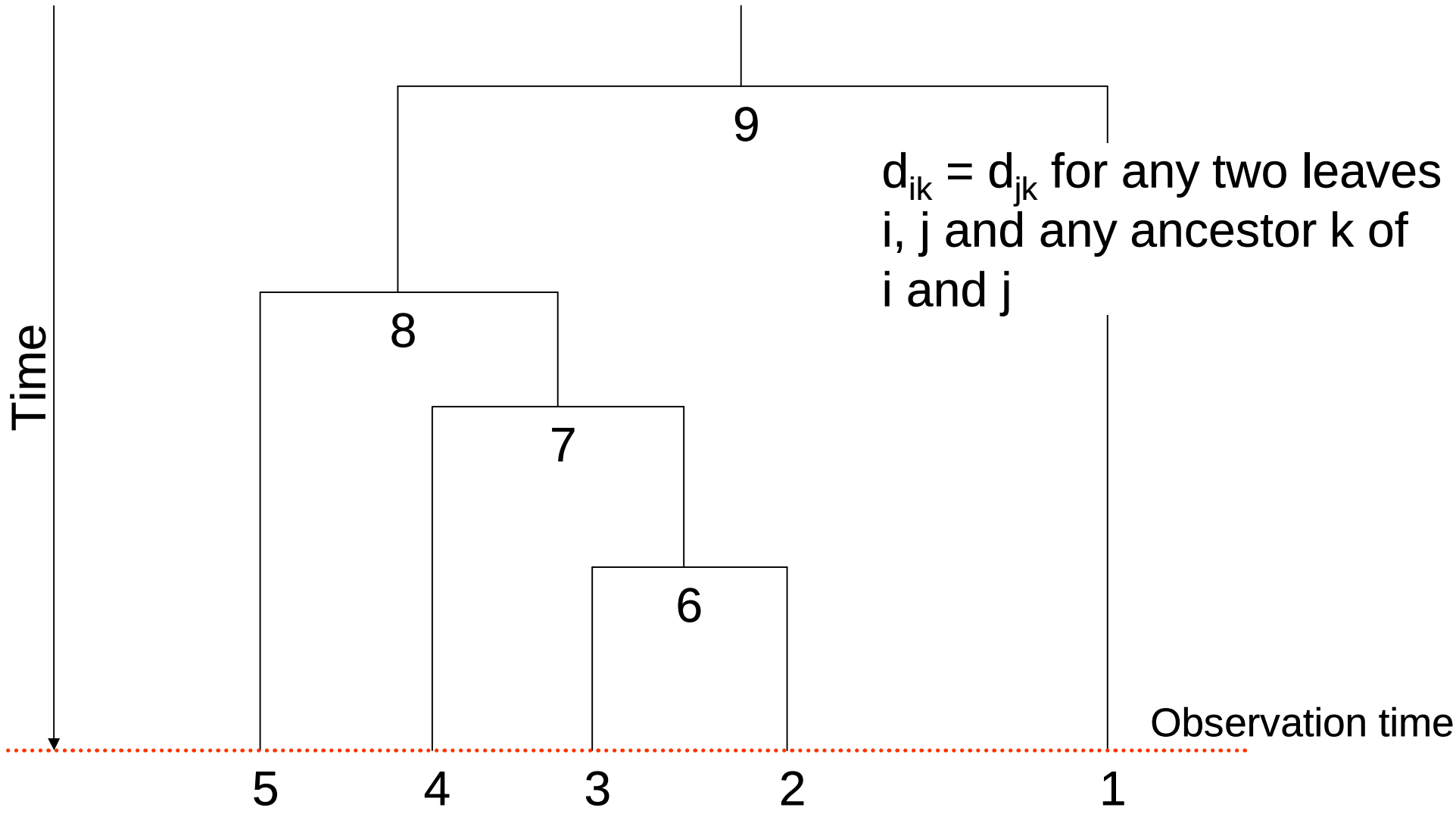
Ultrametric trees



Only vertical segments of the tree have correspondence to some distance d_{ij}:

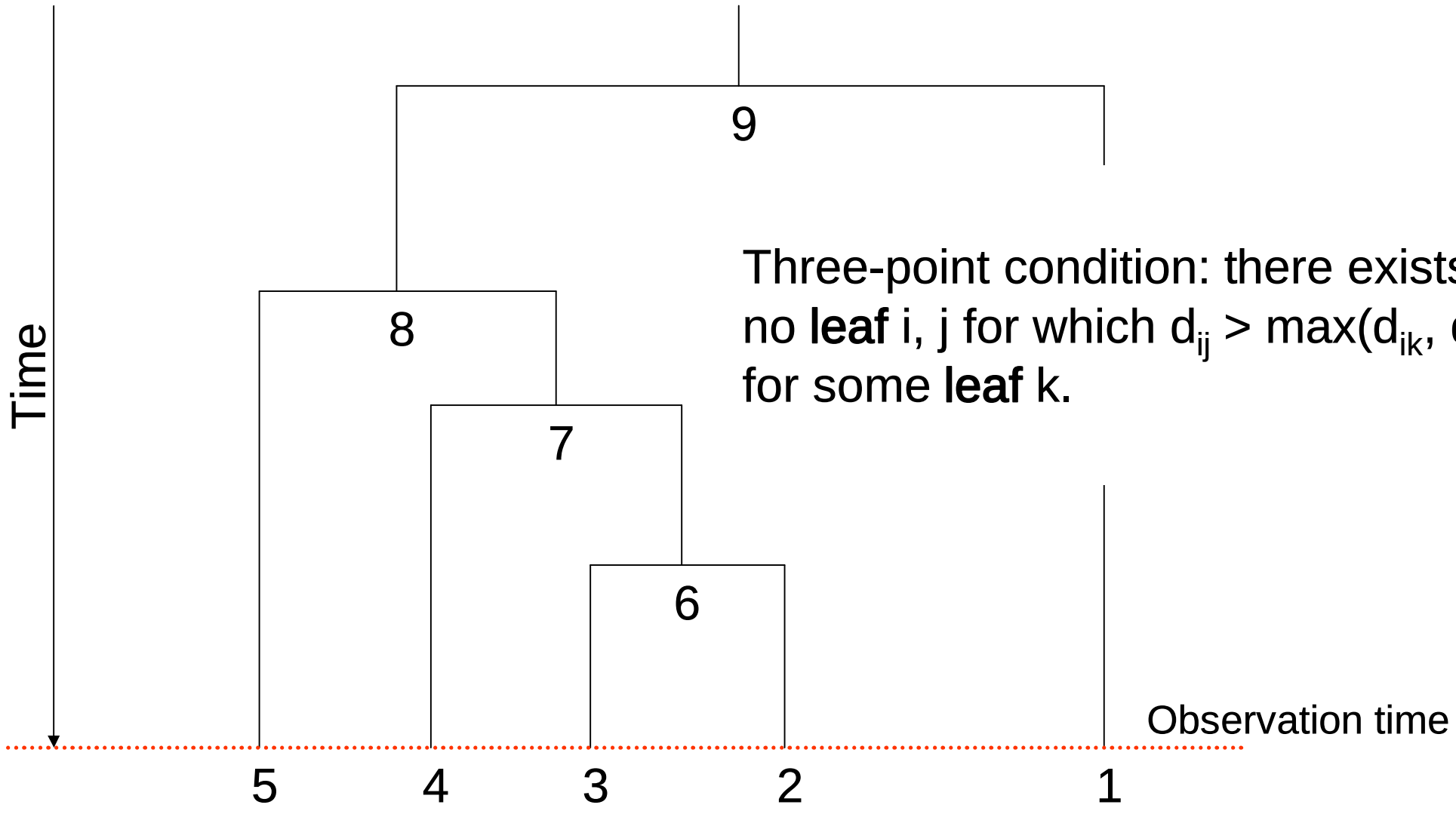
Horizontal segments act as connectors.

Ultrametric trees



$d_{ik} = d_{jk}$ for any two leaves i, j and any ancestor k of i and j

Ultrametric trees



Three-point condition: there exists no **leaf** i, j for which $d_{ij} > \max(d_{ik}, d_{jk})$ for some **leaf** k.

UPGMA algorithm

- UPGMA (unweighted pair group method using arithmetic averages) constructs a phylogenetic tree via clustering
- The algorithm works by at the same time
  - Merging two clusters
  - Creating a new node on the tree
- The tree is built from leaves towards the root
- UPGMA produces a ultrametric tree

Cluster distances

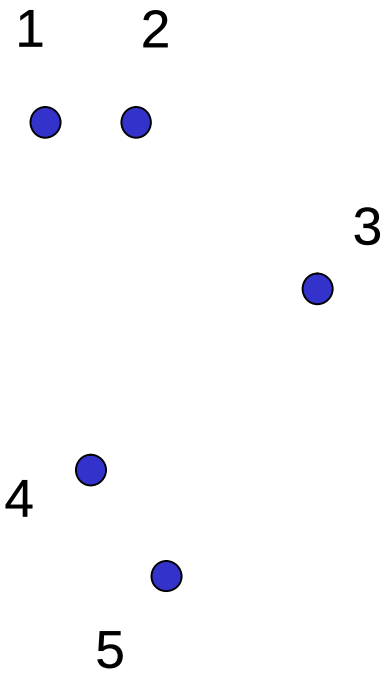
- Let distance d_{ij} between clusters C_i and C_j be

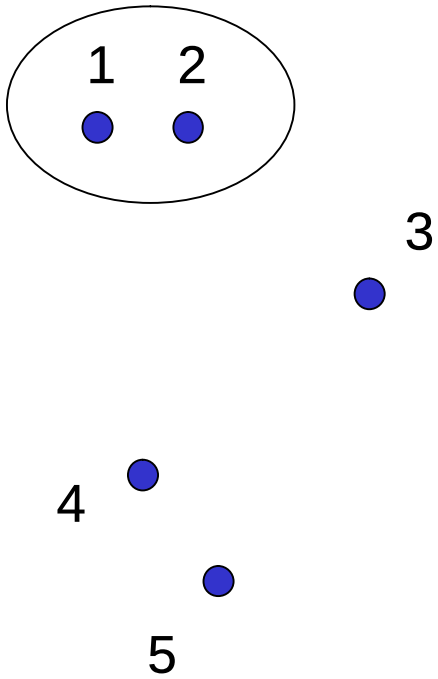
$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

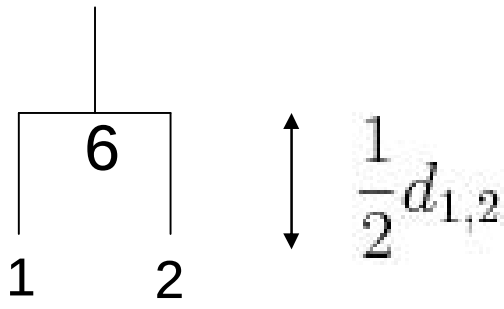
that is, the average distance between points (species) in the cluster.

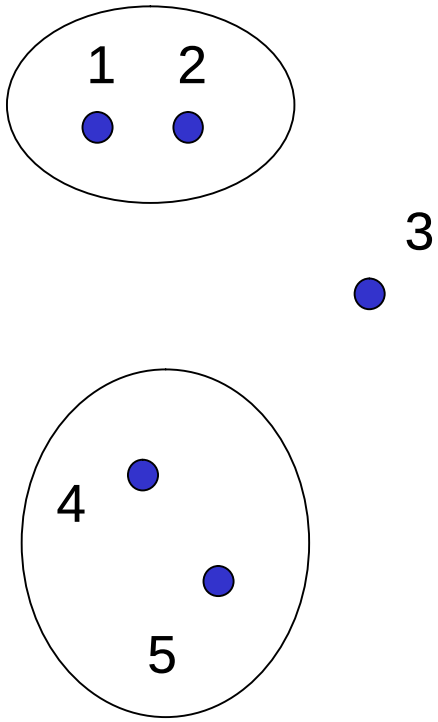
UPGMA algorithm

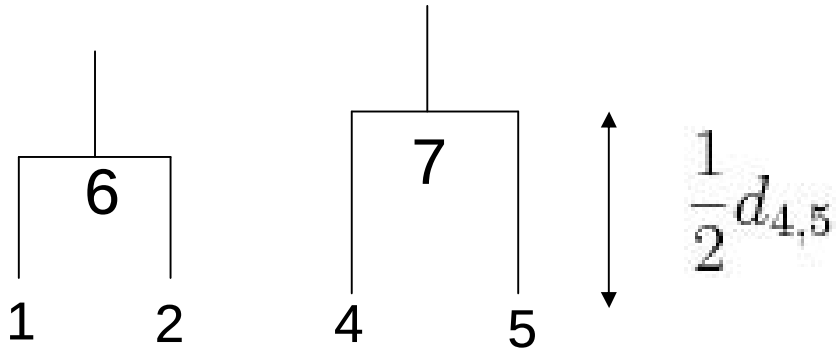
- Initialisation
  - Assign each point i to its own cluster C_i
  - Define one leaf for each sequence, and place it at height zero
- Iteration
  - Find clusters i and j for which d_{ij} is minimal
  - Define new cluster k by $C_k = C_i \cup C_j$, and define d_{kl} for all l
  - Define a node k with children i and j. Place k at height $d_{ij}/2$
  - Remove clusters i and j
- Termination:
  - When only two clusters i and j remain, place root at height $d_{ij}/2$

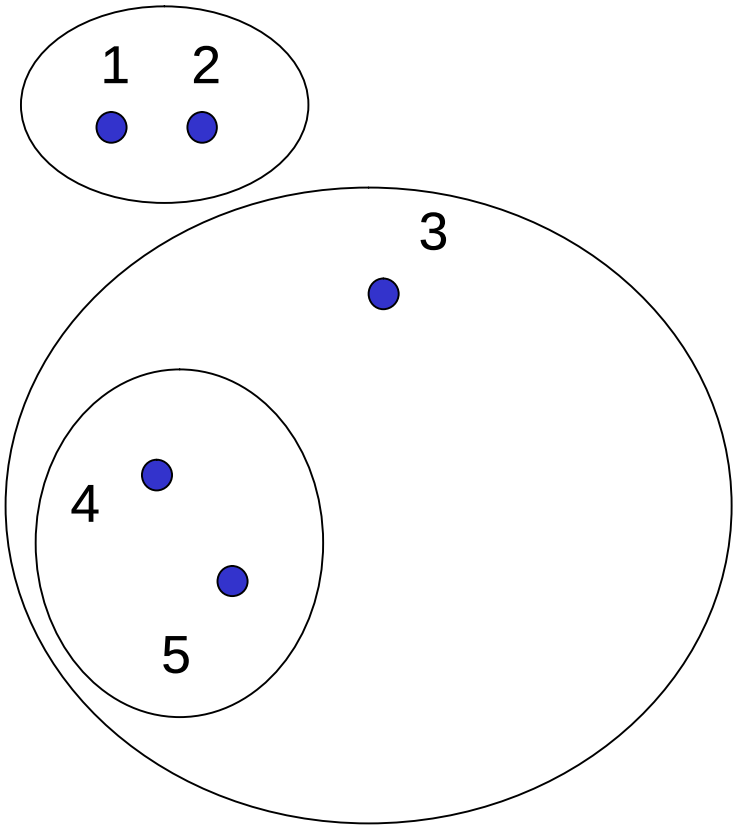

1
2
3
4
5

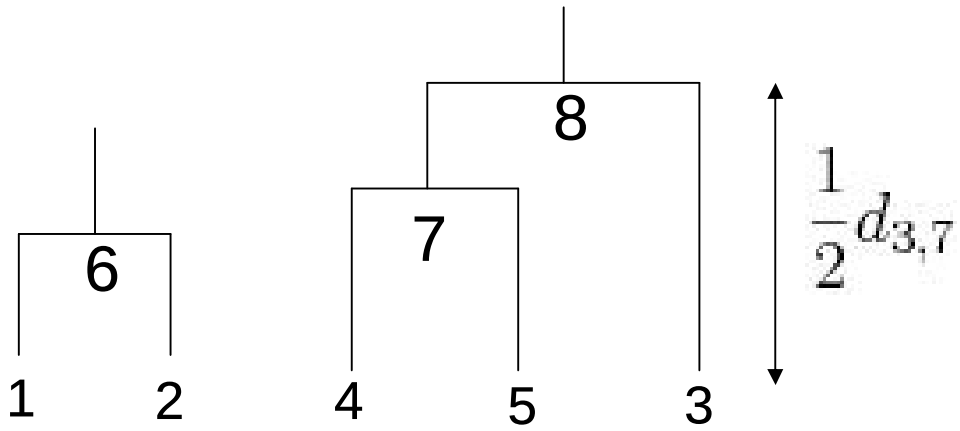

1
2
3
4
5

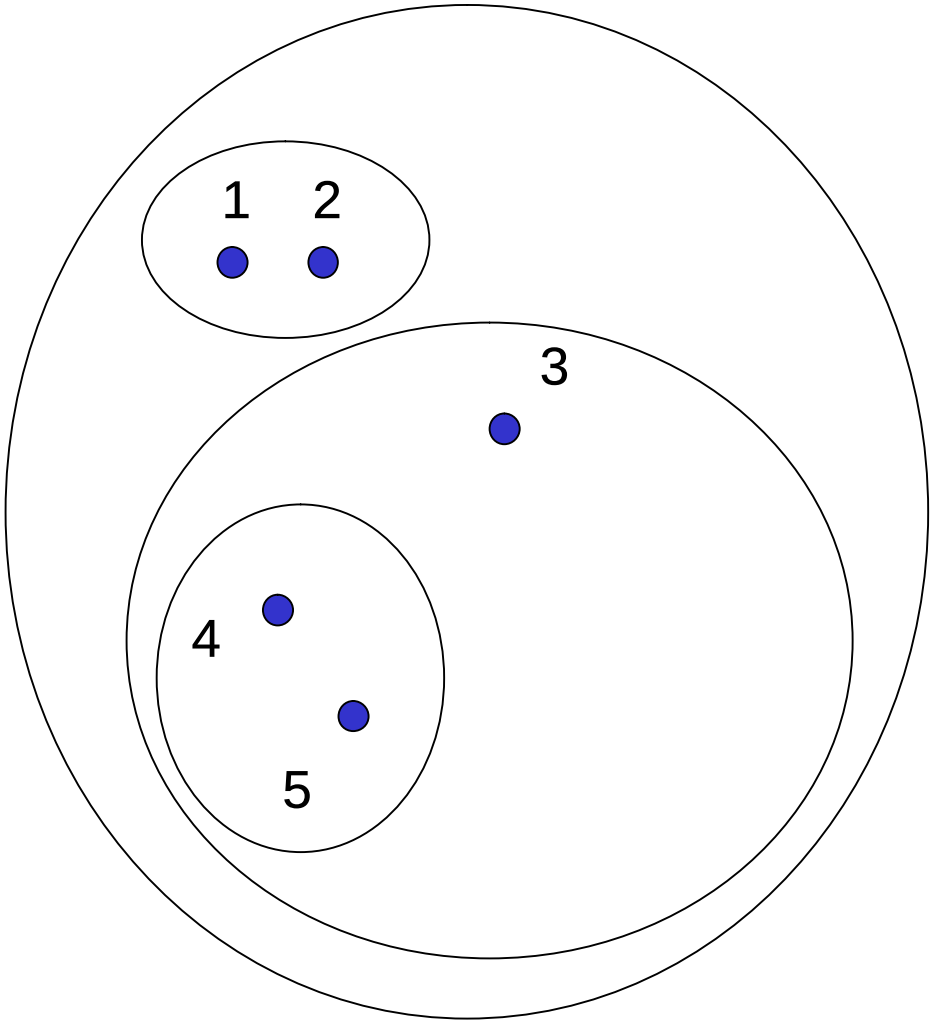

6
1
2
$\frac{1}{2}d_{1,2}$

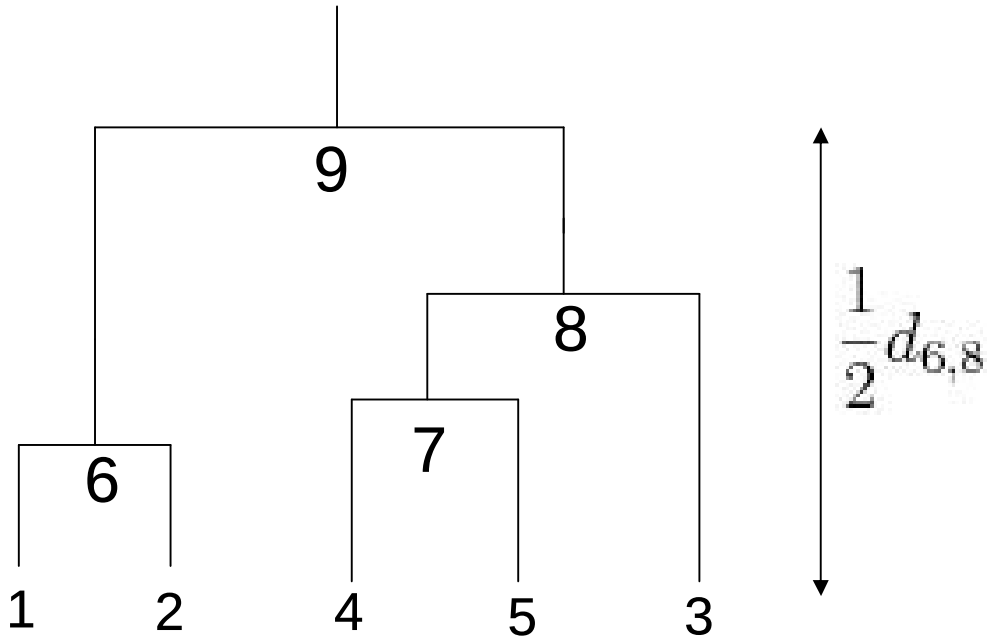

1
2
3
4
5


6
7
1
2
4
5
$\frac{1}{2}d_{4,5}$


1
2
3
4
5


6
1
2
8
7
4
5
3
$\frac{1}{2}d_{3,7}$


1
2
3
4
5


9
8
7
6
1
2
4
5
3
$\frac{1}{2}d_{6,8}$

UPGMA implementation

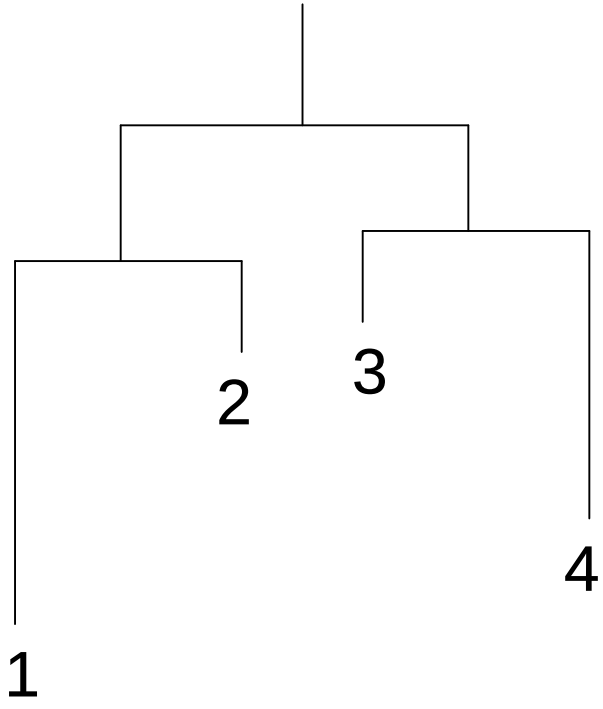
- In naive implementation, each iteration takes $O(n^2)$ time with n sequences => algorithm takes $O(n^3)$ time

- The algorithm can be implemented to take only $O(n^2)$ time (Gronau & Moran, 2006)

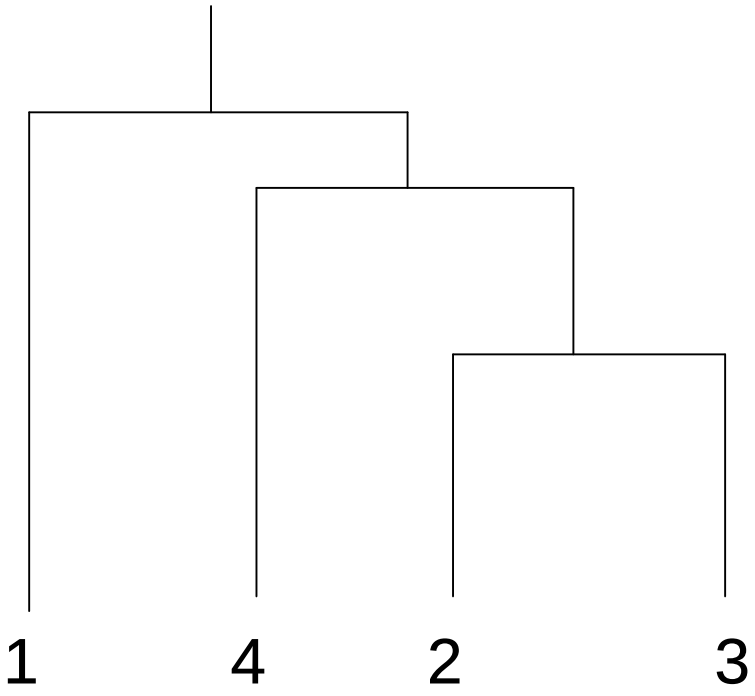
Problem solved?

- We now have a simple algorithm which finds a ultrametric tree
  - If the data is ultrametric, then there is exactly one ultrametric tree corresponding to the data (we skip the proof)
  - The tree found is then the ”correct” solution to the phylogeny problem, if the assumptions hold
- Unfortunately, the data is not ultrametric in practice
  - Measurement errors distort distances
  - *Basic assumption of a molecular clock does not hold usually very well*

Incorrect reconstruction of non-ultrametric data by UPGMA



Tree which corresponds to non-ultrametric distances



Incorrect ultrametric reconstruction by UPGMA algorithm

Finding an additive phylogenetic tree

- Additive trees can be found with, for example, the neighbor joining method (Saitou & Nei, 1987)
- The neighbor joining method produces unrooted trees, which have to be rooted by other means
  - A common way to root the tree is to use an outgroup
  - Outgroup is a species that is known to be more distantly related to every other species than they are to each other
  - Root node candidate: position where the outgroup would join the phylogenetic tree
- However, in real-world data, even additivity usually does not hold very well

Inferring the Past: Phylogenetic Trees (chapter 12)

- The biological problem
- Parsimony and distance methods
- *Models for mutations and estimation of distances*
- Maximum likelihood methods

Estimation of distances

- Many alternative ways to derive the distances d_{ij} exist
- We can construct a simple stochastic model for the evolution of a DNA sequence…
- …and then obtain the distances from the model
- Key points:
  - mutations at sites are rare events in the course of time => poisson process
  - sites evolve individually and by an identical mechanism
  - number of mismatched bases is a sum of mutations at individual sites => binomial variable

A stochastic model for base substitutions

- Consider a single homologous site in two sequences
- Assume the sites diverged for time length t: the sites are separated by time 2t
- Suppose that the number of substitutions in any branch of length t has a Poisson distribution with mean λt
- Probability that k substitutions occur is given by the Poisson probability $e^{-\lambda t}(\lambda t)^k/(k!)$, k = 0, 1, 2, …

Substitutions at one site

- General model: P(substitution results in base j | site was base i) = m_{ij}
- Felsenstein model: $m_{ij} = \pi_j$, with $\pi_j \geq 0$ and $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$
- Assume that the set of probabilities π_j is same at every position in the sequence

Substitutions at one site (2)

- Probability $q_{ij}(t)$ that a base i at time 0 is substituted by a base j a time t later
- $q_{ij}(t) = e^{-\lambda t} + (1 - e^{-\lambda t})\, \pi_j$, if i = j
- $q_{ij}(t) = (1 - e^{-\lambda t})\, \pi_j$, otherwise

Substitutions at one site (3)

- We assume stationarity: distribution of base frequencies is the same for every time t
- In other words, we want that

  P(base a time t later = j) π_j^0

- For our simple model, this can be shown to hold

Estimating distances

- Distances should take into account the mutation mechanism
- Average of λt substitutions occur at a particular site on a branch of length t
- However, some of the substitutions do not change the base (A -> A or A -> G -> A, for example)

Mean number of substitutions in time t

- What is the chance H that a substitution actually changes a base?
- $H = \sum \pi_i(1 - \pi_i) = 1 - \sum \pi_i^2$
- Average number of real substitutions is then $\lambda t H$
- Distance K between two sequences is

$K = 2\lambda t H$

Estimating distances from sequence data

- We want to estimate $K = 2\lambda tH$ from sequence data
- The chance $F_{ij}(t)$ that we observe a base i in one sequence and a base j in another is

$$F_{ij}(t) = \sum_l \pi_l q_{li}(t) q_{lj}(t)$$

by averaging over the possible ancestral nucleotides

Estimating distances from sequence data

- Expression $F_{ij}(t) = \sum_l \pi_l q_{li}(t) q_{lj}(t)$ can be simplified by assuming that the mutation process is reversible:

  $\pi_l m_{ij} = \pi_j m_{ji}$ for all $i \neq j$

- From this it can be shown that

  $\pi_l q_{ij}(t) = \pi_j q_{ji}(t)$ for all i, j and $t > 0$

- Now the model simplifies into $F_{ij}(t) = \pi_i q_{ij}(2t)$

Estimating distances from sequence data

- What is the probabilitity F = F(t) that the letters at a particular position in two immediate descendants from the same node are identical?

$$F = \sum_i \pi_i q_{ii}(2t) = e^{-2\lambda t} + (1 - e^{-2\lambda t})(1 - H)$$

Putting the sites together

- Assume that
  - sites evolve independently of one other and
  - mutation process is identical at each site
  - The two sequences have been aligned against each other and gaps have been removed
- Do the bases at site i in the sequences differ?

$X_i = 1$ if the ith pair of sites differ

$X_i = 0$ otherwise

Putting the sites together (2)

- $P(X_i = 1) = 1 – F = (1 - e^{-2\lambda t})H$
- Now $D = X_1 + … + X_s$ is the number of mismatched pairs of bases
- D is a binomial random variable with parameters s and 1 – F
- Notice that D is the Hamming distance for the sequences

Putting the sites together (3)

- F is unknown and has to be estimated from the sequence data
- Recall that the observed proportion of successes is a good estimator of the binomial success probability: estimate 1 – F with D/s
- $D/s = (1 - e^{-2\lambda t})H$
- $2\lambda t = -\log(1 – D/(sH))$
- Finally, we obtain $K = 2\lambda tH = -H \log(1 – D/(sH))$

Jukes-Cantor formula

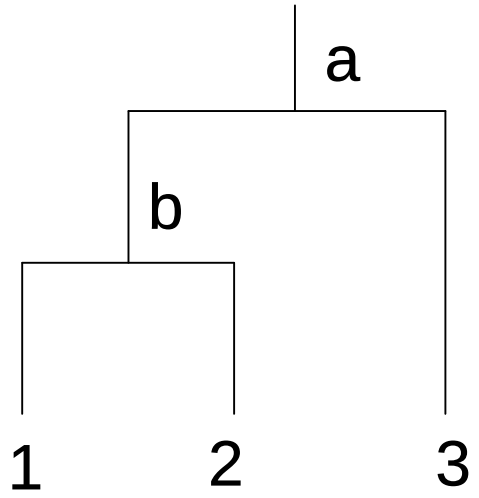
- Estimate $2\lambda tH = -H \log(1 - D/(sH))$ of the distance K is known as the Jukes-Cantor formula
- When H (chance that a substitution actually occurs) approaches 1, the estimate decreases and approaches the Poisson mean $2\lambda t$
- H is usually not known and has to be estimated from the data as well

Inferring the Past: Phylogenetic Trees (chapter 12)

- The biological problem
- Parsimony and distance methods
- Models for mutations and estimation of distances
- *Maximum likelihood methods*

Maximum likelihood methods

- Consider the tree on the right with three sequences
- Probability $p(i_1, i_2, i_3)$ of observing bases i_1, i_2 and i_3 can be computed by summing over all possible ancestral bases,



$$p(i1, i2, i3) = \sum_a \sum_b \pi_a q_{ai3}(t_2) q_{ab}(t_2 - t_1) q_{bi2}(t_1) q_{bi1}(t_1)$$

- Hard to compute for complex trees

Maximum likelihood estimation

- We would like to calculate likelihood $p(i_1, i_2, \ldots, i_n)$ in the general case
- Calculations can be arranged using the peeling algorithm
- Basic idea is to move all summation signs as far to the right as possible

Maximum likelihood estimation

- Likelihood for the data is then obtained by multiplying the likelihoods of individual sites

- General recipe for maximum likelihood estimation:
  - Maximize over all model parameters for a *given* tree
  - Maximize previous expression over *all* possible trees

Problems with tree-building

- Assumptions
  - Sites evolve independently of one other
  - Sites evolve according to the same stochastic model
  - The tree is rooted
  - The sequences are aligned
  - Vertical inheritance

Additional material on phylogenetic trees

- Durbin, Eddy, Krogh, Mitchison: Biological sequence analysis
- Jones, Pevzner: An introduction to bioinformatics algorithms
- Gusfield: Algorithms on strings, trees, and sequences