Which Animal Gave Us SARS?

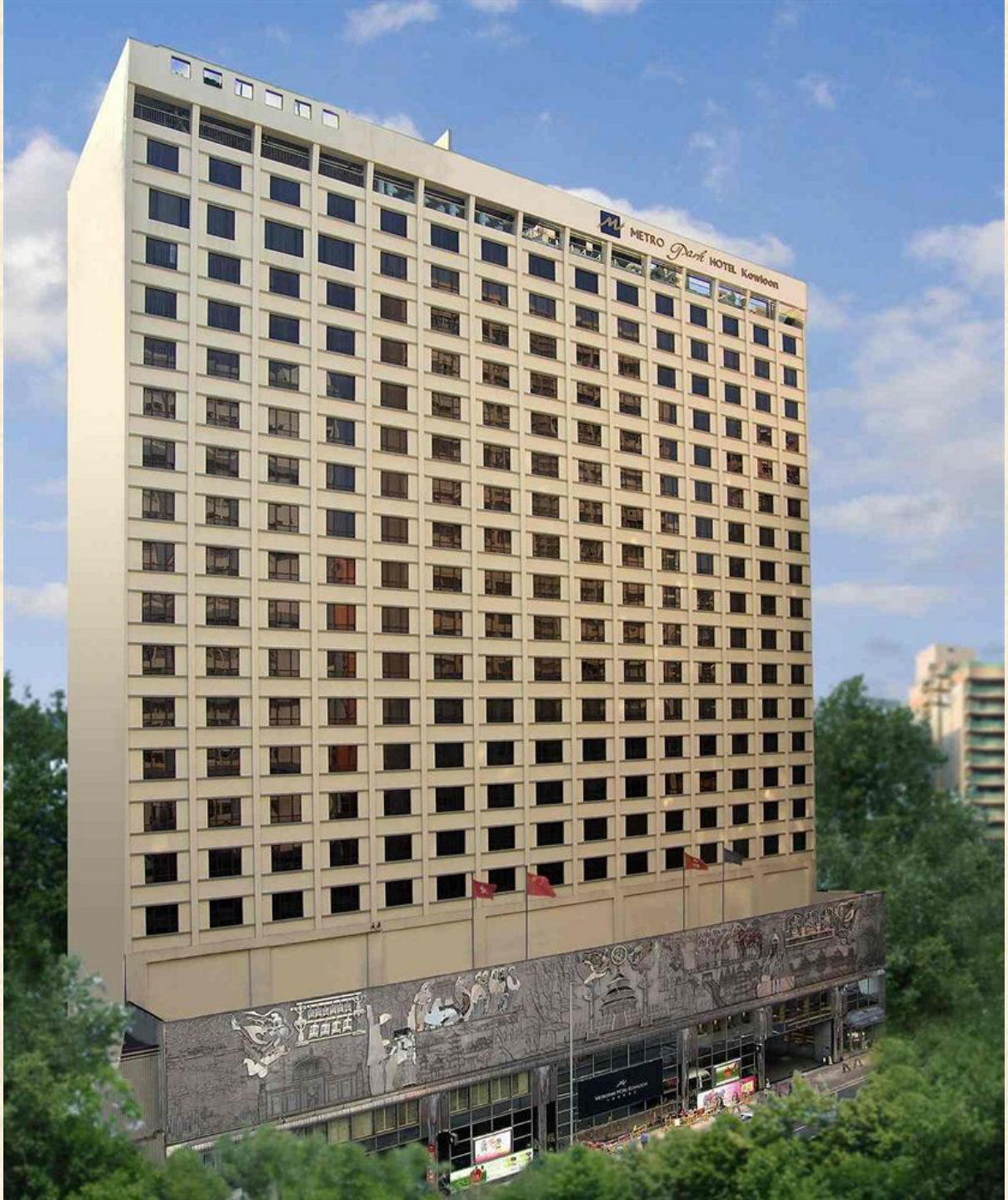
Evolutionary Tree Reconstruction

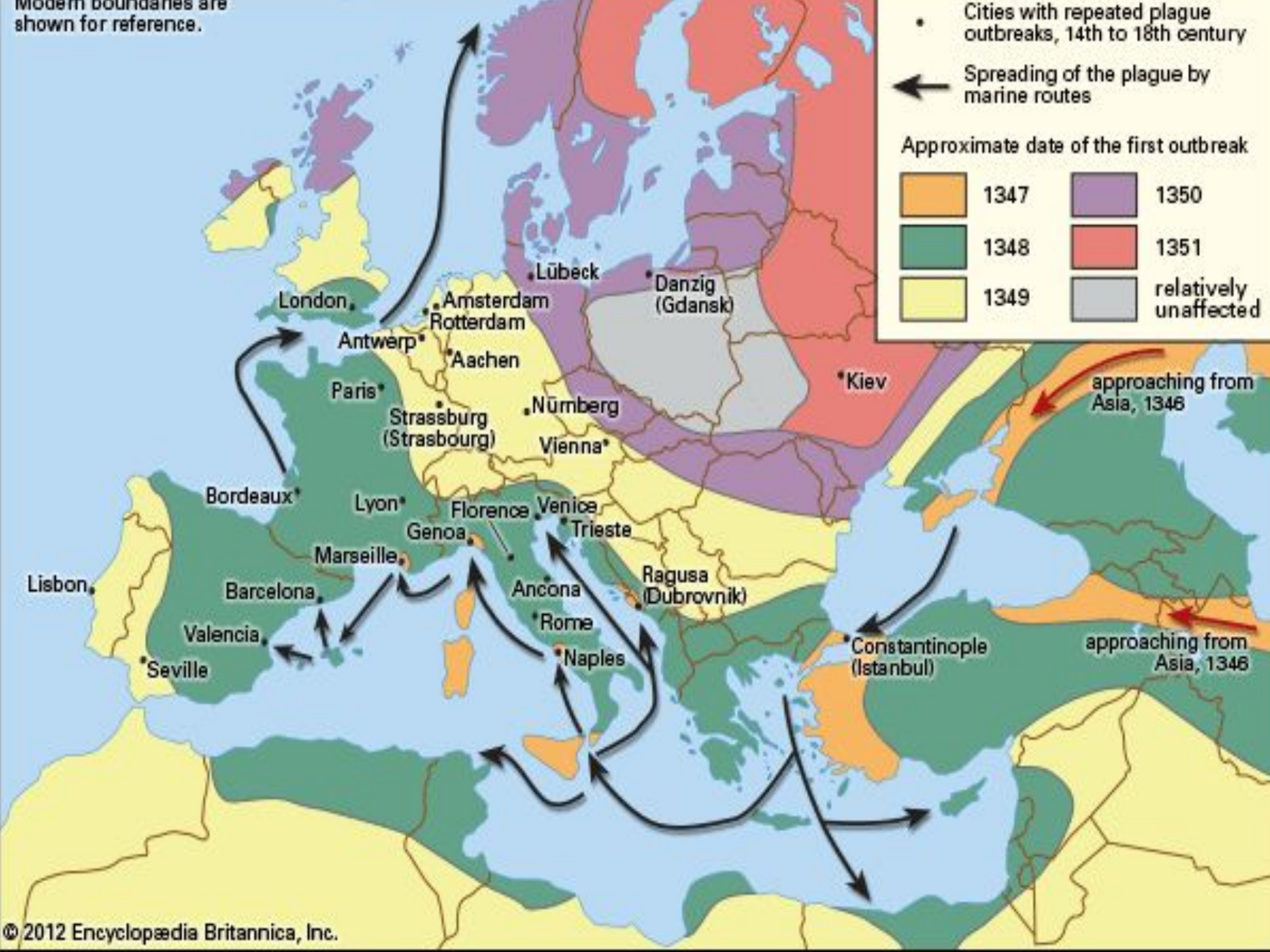
Phillip Compeau and Pavel Pevzner

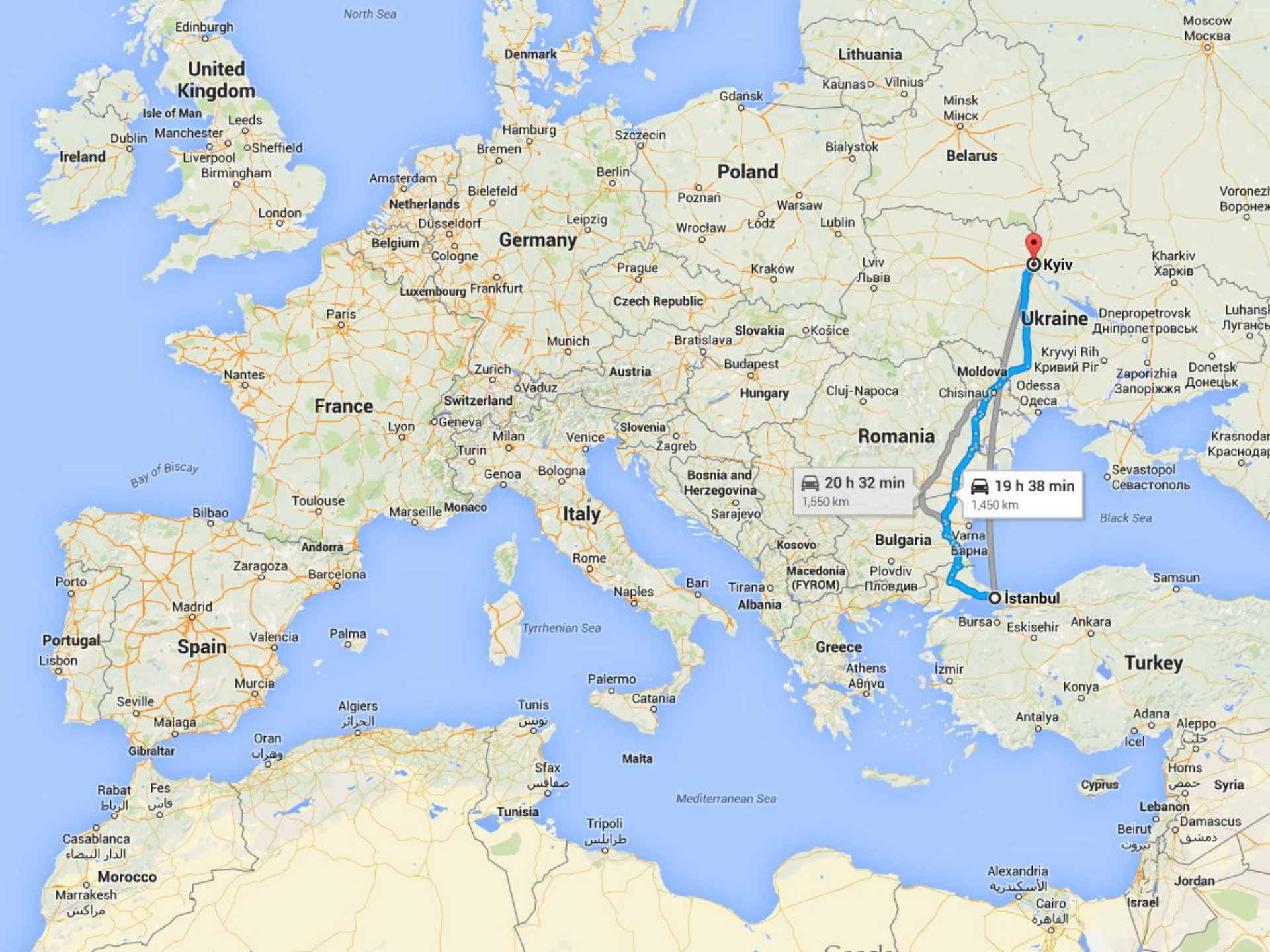
Bioinformatics Algorithms: An Active Learning Approach

Outline

- **The Fastest Outbreak**
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

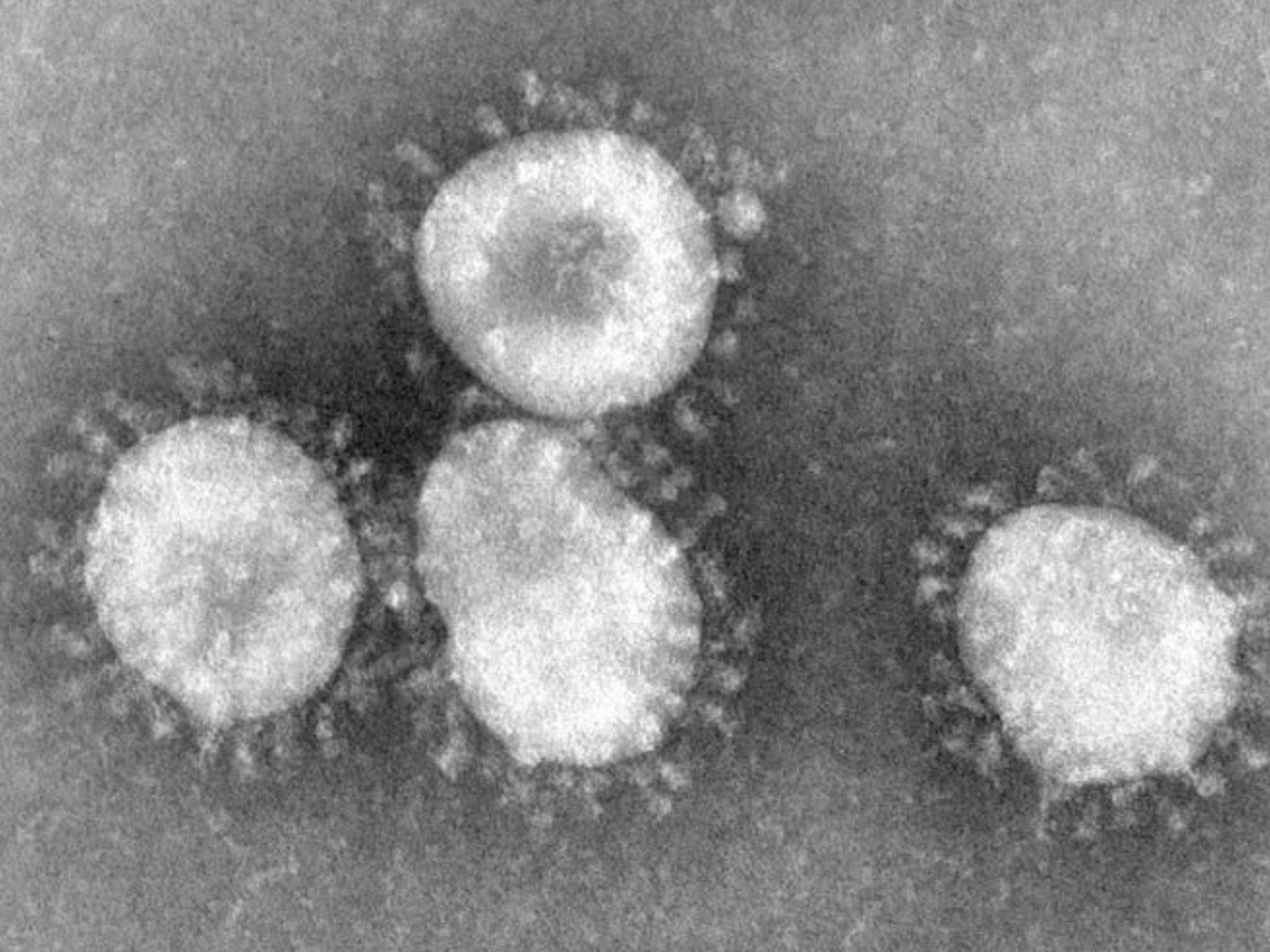

METRO Park HOTEL Kowloon




20 h 32 min
1,550 km
19 h 38 min
1,450 km
Kyiv
İstanbul
North Sea
Black Sea
Mediterranean Sea
Tyrrhenian Sea
Bay of Biscay
United Kingdom
Ireland
Isle of Man
Edinburgh
Dublin
Manchester
Leeds
Liverpool
Sheffield
Birmingham
London
Denmark
Netherlands
Amsterdam
Belgium
Luxembourg
Germany
Hamburg
Bremen
Düsseldorf
Cologne
Frankfurt
Paris
France
Nantes
Toulouse
Bilbao
Marseille
Monaco
Switzerland
Zurich
Geneva
Vaduz
Lyon
Milan
Turin
Venice
Genoa
Bologna
Italy
Rome
Naples
Bari
Andorra
Spain
Portugal
Madrid
Barcelona
Zaragoza
Valencia
Palma
Porto
Lisbon
Seville
Málaga
Murcia
Gibraltar
Algiers
Oran
Tunis
Sfax
Tunisia
Malta
Palermo
Catania
Tripoli
Rabat
Fes
Casablanca
Marrakesh
Morocco
Poland
Gdańsk
Szczecin
Berlin
Poznań
Warsaw
Wrocław
Łódź
Lublin
Leipzig
Prague
Czech Republic
Kraków
Munich
Austria
Slovakia
Košice
Bratislava
Budapest
Hungary
Slovenia
Zagreb
Bosnia and Herzegovina
Sarajevo
Kosovo
Macedonia (FYROM)
Albania
Tirana
Greece
Athens
Bulgaria
Varna
Plovdiv
Romania
Cluj-Napoca
Chisinau
Moldova
Ukraine
Lithuania
Kaunas
Vilnius
Minsk
Belarus
Bialystok
Lviv
Moscow
Kharkiv
Dnepropetrovsk
Kryvyi Rih
Zaporizhia
Donetsk
Odessa
Sevastopol
Krasnodar
Turkey
Samsun
Bursa
Eskisehir
Ankara
İzmir
Konya
Antalya
Adana
Aleppo
Icel
Homs
Cyprus
Syria
Lebanon
Beirut
Damascus
Jordan
Israel
Alexandria
Cairo


Ireland
1 case
China
Canada
29 cases
United States
1 case
Guangdong
Province
Metropole
Hotel
Hong Kong
Special
Administrative
Region
195 cases
Philippine
Sea
Vietnam
58 cases
Arabian
Sea
Bay
of
Bengal
South
China
Sea
Singapore
71 cases
Indian
The Spread
of ?????

The Spread of SARS



28261 aauaauacug cgucuugguu cacagcucuc acucagcaug gcaaggagga acuuagauuc
28321 ccucgaggcc agggcguucc aaucaacacc aauagugguc cagaugacca aauuggcuac
28381 uaccgaagag cuacccgacg aguucguggu ggugacggca aaaugaaaga gcucagcccc
28441 agaugguacu ucuauuaccu aggaacuggc ccagaagcuu cacuucccua cggcgcuaac
28501 aaagaaggca ucguaugggu ugcaacugag ggagccuuga auacacccaa agaccacauu
28561 ggcacccgca auccuaauaa caaugcugcc accgugcuac aacuuccuca aggaacaaca
28621 uugccaaaag gcuucuacgc agagggaagc agaggcggca gucaagccuc uucucgcucc
28681 ucaucacgua gucgcgguaa uucaagaaau ucaacuccug gcagcaguag gggaaauucu
28741 ccugcucgaa uggcuagcgg agguggugaa acugcccucg cgcuauugcu gcuagacaga
28801 uugaaccagc uugagagcaa aguuucuggu aaaggccaac aacaacaagg ccaaacuguc
28861 acuaagaaau cugcugcuga ggcaucuaaa aagccucgcc aaaaacguac ugccacaaaa
28921 caguacaacg ucacucaagc auuugggaga cgugguccag aacaaaccca aggaaauuuc
28981 ggggaccaag accuaaucag acaaggaacu gauuacaaac auuggccgca aauugcacaa
29041 uuugcuccaa gugccucugc auucuuugga augucacgca uuggcaugga agucacaccu
29101 ucgggaacau ggcugacuua ucauggagcc auuaaauugg augacaaaga uccacaauuc
29161 aaagacaacg ucauacugcu gaacaagcac auugacgcau acaaaacauu cccaccaaca
29221 gagccuaaaa aggacaaaaa gaaaaagacu gaugaagcuc agccuuugcc gcagagacaa
29281 aagaagcagc ccacugugac ucuucuuccu gcggcugaca uggaugauuu cuccagacaa
29341 cuucaaaauu ccaugagugg agcuucugcu gauucaacuc aggcauaaac acucaugaug
29401 accacacaag gcagaugggc uauguaaacg uuuucgcaau uccguuuacg auacauaguc
29461 uacucuugug cagaaugaau ucucguaacu aaacagcaca aguagguuua guuaacuuua
29521 aucucacaua gcaaucuuua aucaaugugu aacauuaggg aggacuugaa agagccacca
29581 cauuuucauc gaggccacgc ggaguacgau cgaggguaca gugaauaaug cuagggagag
29641 cugccuauau ggaagagccc uaauguguaa aauuaauuuu aguagugcua uccccaugug
29701 auuuuaauag cuucuuagga gaaugacaaa aaaaaaaaaa aaaaaaaaaa a

28261 aauaauacug cgucuugguu cacagcucuc acucagcaug gcaaggagga acuuagauuc
28321 ccucgaggcc agggcguucc aaucaacacc aaCagugguc cagaugacca aauuggcuac
28381 uacAgaagag cuacccgacg aguucguggu ggugacggca aaaugaaaga gcucagcccc
28441 agaugguacu ucuauuaccu aggaacuggc ccagaagcuu cacuucccua cggcgcuaac
28501 aaagaaggca ucguaugggu ugcaacugag ggagccuuga auacCcccaa agaccacauu
28561 ggcacccgca auccuaauaa caaugcugcc accgugcuac aacuuccuca aggaacaaca
28621 uugccaaaag gcuucuacgc agagggaagc agaggcggca gucaagccuc uucucgcucc
28681 ucaucacgua gucgcgguaa uucaagaaau ucaacuccug gcagcaguag gggaaauucu
28741 ccugcucgaa uggcuagcgg agguggugaa acugcccucg cgcuauugcu gcuagacaga
28801 uugaaccagc uugagagcaa aguuucuggu aaaggccaac aacaacaagg ccaaacuguc
28861 acuaagaaau cugcugcuga ggcaucuaaa aagccucgcc aaaaacguac ugccacaaaa
28921 caguacaacg ucacucaagc auuugggaga cgugguccag aacaaaccca aggaaauuuc
28981 ggggaccaag acUuaaucag acaaggaacu gauuacaaac auuggccgca aauugcacaa
29041 uuugcuccaa gugccucugc auucuuugga augucacgca uuggcaugga agucacaccu
29101 ucgggaacau ggcugacuua ucauggagcc auuaaauugg augacaaaga uccacaauuc
29161 aaagacaacg ucauacugcu gaacaagcac auugacgcau acaUaacauu cccaccaaca
29221 gaUccuaaaa aggacaaaaa gaaaaagacu gaugaagcuc agccuuugcc gcagagacaa
29281 aagaagcagc ccacugugac ucuucuuccu gcggcugaca uggaugauuu cuccagacaa
29341 cuucaaaauu ccaugagugg agcuucugcu gauucaacuc aggcauaaac acucaugaug
29401 accacacaag gcagaugggc uauguaaacg uuuucgcaau uccguuuacg auacauaguc
29461 uacucuugug cagaaugaau ucucguaacu aaacagcaca aguagguuua guuaacuuua
29521 aucucacaua gcaaucuuua aucaaGgugu aacauuaggg aggacuugaa agagccacca
29581 cauuAucauc gaggccacgc ggaguacgau cgaggguaca gugaauaaug cuagggagag
29641 cugccuauau ggaagagccc uaauguguaa aauuaauuuu aguagugcua uccccaugug
29701 auuuuaauag cuucuuagga gaaugacaaa aaaaaaaaaa aaaaaaaaaa a

28261 aauaauacug cgucuugguu cacagcucuc acucagcaug gcaaggagga acuuagauuc
28321 ccucgaggcc agggcguucc aaucaacacc aaCaguggUc cagaugacca aauuggcuac
28381 uacAgaagag cuacccgacg aguucguggu ggugacggca aaaugaaaga gcuGagcccc
28441 agaugguacu ucuauuaccu aggaacuggc ccagaagcuu cacuucccua cggcgcuaac
28501 aaagaaggca ucguauUggu ugcaacugag ggagccuuga auacCcccaa agaccacauu
28561 ggcacccgca auccuaauaa caaugcugcc accgugcuac aacuuccuca aggaacaaca
28621 uugccaaaag gcuucAacgc agagggaagc agaggcggca gucaagccuc uucucgcucc
28681 ucaucacgua gucgcgguaa uucaagaaau ucaacuccug gcagcaguag gggaaauucu
28741 cGugcucgaa uggcuagcgg agguggugaa acugcccucg cgcuauugcu gcuagacaga
28801 uugaaccagc uugagagcaa aguuucuggu aaaggccaac aacaacaagg ccaGacuguc
28861 acuaagaaau cugcugcuga ggcaucuCaa aagccucgcc aaaaacguac ugccacaaaa
28921 caguacaacg ucacucaagc auuugggaga cgugguccag aacaaaccca aggaaauuuc
28981 ggggaccaag acUuaaucag acaaggaacu gauuacaaac auuggccgca aauugcacaa
29041 uuugcuccaa gugccucugc auucuuugga auguUacgca uuggcaugga agucacaccu
29101 ucgggaacau ggcugacuua ucauggagcc auuaaauugg augacaaaga uccacaauuc
29161 aaagacaacg ucauacugcu gaacaagcac auugacgcau acaUaacauu cccaccaaca
29221 gaUccuaaaa aggacaaaaa gaaaaagacu gaugaagcuc agccuuugcc gcagagacaa
29281 aagaagcagc ccacugugac ucuucuuccu gcggcugaca uggaugauuu cuccagacaa
29341 cuucaaaauu ccaugagugg aAcuucugcu gauucaacuc agUcauaaac acucaugaug
29401 accaAacaag gcagaugggc uauguaaacg uuuucgcaau uccguuuacg auacauaguc
29461 uacucuugug cagaaugaau ucucguaacu aaacagcaca aguagguuua guuaacAuua
29521 aucucacaua gcaaucuuua aucaaGgugu aacauuaggg aggacuugaa agagccacca
29581 cauuAucauc gaggccacgc ggaguacgau cgaggguaca gugaauaaug cuagggagag
29641 cugccuauau ggaagagccc uaauguguaa aauuaauuuu aguagugcua uccccaugug
29701 auuuuaauag cuucuuagga gaaugacaaa aaaaaaaaaa aaaaaaaaaa a

28261 Cauaauacug cgucuugguu cacaUcucuc acucagcaug gcaaggagga acuuagauuc
28321 ccucgaggcc agggcguucc aaucaacacc aaCagugguc cagaugacca aauuggcuac
28381 uacAgaagag cuacccgacg aguucguggu ggugacggca aaaugaaaga gcuGagcccc
28441 agaugguacu ucuauuaccu aggaacuggc ccagaagcuu cacuucccua cggcgcuaac
28501 aaagaaggca ucguauUggu ugcaacugag ggagccuuga auacCcccaa agaccacauu
28561 ggcacccgca auccuaauaa caaugcugcc accgugcuac aacuuccuca aggaacaaca
28621 uugccaaaag gcuucAacgc agagggaagc agaggcggca gucaagccuc uucucgcucc
28681 ucaucacgua gucgcgguaa uucaagUaau ucaacuccug gcaAcaguag gggaaauucu
28741 cGugcucgaa uggcuagcgg agguggugaa acugcccucg cgcuauugcu gcuagacaga
28801 uugaaccagc uugagagcaa aguuucuggu aaaggccaac aacaacaagg ccaGacuguc
28861 acuaagaaau cugcugcuga ggcaucuCaa aagccucgcc aaaaacguac ugccacaaaa
28921 caguacaacg ucacucaagc auuugggaga cgugguccag aacaaaccca aggaaauuuc
28981 ggggaccaag acUuaaucag acaaggaacu gauuacaaac auuggccgca aauugcacaa
29041 uuuAcuccaa gugccucugc auucuuugga auguUacgca uuggcaugga agucacaccu
29101 ucgggaacau ggcugacuua ucauggagcc auuaaauugg augacaaaga uccacaauuc
29161 aaagacaacg ucauacugcu gaacaagcac auugacgcau acaUaacauu cccaccaaca
29221 gaUccuaaaa aggGcaaaaa gaaaaagacu gaugaagcuc agccuuugcc gcagagacaa
29281 aagaagcagc ccacugugac ucuucuuccu gcggcugaca uggaugauuu cuccaUacaa
29341 cuucaaaauu ccaugagugg aAcuucugcu gauucaacuc agUcauaaac acucaugaug
29401 accaAacaag gcagaugggc uauguaaacg uuuucgcaau uccguuuacg auacauaguc
29461 uacucuugug cagaaugaau ucucguaacu aaacagcaca aguagguuua guuaacAuua
29521 aucucacaua gcaaucuuua aucaaGgugu aacauuaggg aggacuugaa agagccacca
29581 cauuAucauc gaggccacgc ggaguacgau cgaggguaca gugaauaaug cuagggagag
29641 cugccuauau ggaagagccc uaauguguaa aauuaauuuu aUuagugcua uccccaugug
29701 auuuuaauag cuucuuagga gaaugacaaa aaaaaaaaaa aaaaaaaaaa a

28261 Cauaauacug cgucuugguu cacaUcucuc acucagcaug gcaaggagga acuuagauuc
28321 ccucgaggcc agggcguCcc aaucaacacc aaCagugguc cagaugacca aauuggcuac
28381 uacAgaagag cuacccgacg aguucguggu ggugacggca aaaugaaaga gcuGagcccc
28441 agaugguacu ucuauuaccu aggaacuggc ccagaagcuu cacuucccua cggcgcuaac
28501 aaagaaggca ucguauUggu ugcaacuUag ggagccuuga auacCGccaa agaccacauu
28561 ggcacccgca auccuaauaa caaugcugcc accgugcuac aacuuccuca aggaacaaca
28621 uugccaaaag gcuucAacgc agagggaagc agaggcggca gucaagccuc uucucgcucc
28681 ucaucacgua gucgcgguaa uucaagUaau ucaacuccug gcaAcaguag gggaaauucu
28741 cGugcucgaa uggcuagcgg agguggugaa acugcccucg cgcuauugcu gcuagacaga
28801 uugaaccagc uugagagcaa aguuucuggu aaaggccaac aacaacaagg ccaGacuguc
28861 acuaagaaau cugcugcuga ggcaucuCaa aagccucgcc aaaaacguac ugccacaaaa
28921 caguacaacg ucacucCagc auuugggaga cgugguccag aacaaaccca aggaaauuuc
28981 ggggaccaag acUuaaucag acaaggaacu gauuacaaac auuggccgca aauugcacaa
29041 uuuAcuccaC gugccucugc auucuuugga auguUacgca uuggcaugga agucacaccu
29101 ucgggaacau ggcugacuua ucauggagcc auuaaauugg augacaaaga uccacaauuc
29161 aaagacaacg ucauacugcu gaacaagcac auugacgcau acaUaaGauu cccaccaaca
29221 gaUccuaaaa aggGcaaaaa gaaaaagacu gaugaagcuc agccuuugcc gcagagacaa
29281 aagaagcagc ccacugugac ucuucuuccu gcggcugaca uggaugauuu cuccaUacaa
29341 cuucaaaauu ccaugagugg aAcuucugcu gauucaacuc agUcauaaac acucaugaug
29401 accaAacaag gcagaugggc uauguaaacg uuuucgcaau uccguuuacg auacauaguc
29461 uacucuugug cagaaugaau ucucguaacu aaacagcaca aguagguuua guuaacAuua
29521 aucucacaua gcaaucuuua aucaaGgugu aacauuaggg aggacuugaa agagccacca
29581 cauuAucauc gaggccacgc ggaguacgau cgaggguaca gugaauaaug cuaAggagag
29641 cugccuauau ggaagagccc uaauAuguaa aauuaauuuu aUuagugcua uccccaugug
29701 auuuuaauag cuucuuagga gaaugacaaa aaaaaaaaaa aaaaaaaaaa a

Questions about SARS

- Which animal gave us SARS?

Questions about SARS

- Which animal gave us SARS?
- How were we first infected?

Questions about SARS

- Which animal gave us SARS?
- How were we first infected?
- How did SARS spread around the world?

Questions about SARS

- Which animal gave us SARS?
- How were we first infected?
- How did SARS spread around the world?
- All these questions relate to constructing **evolutionary trees** (a.k.a. **phylogenies**).

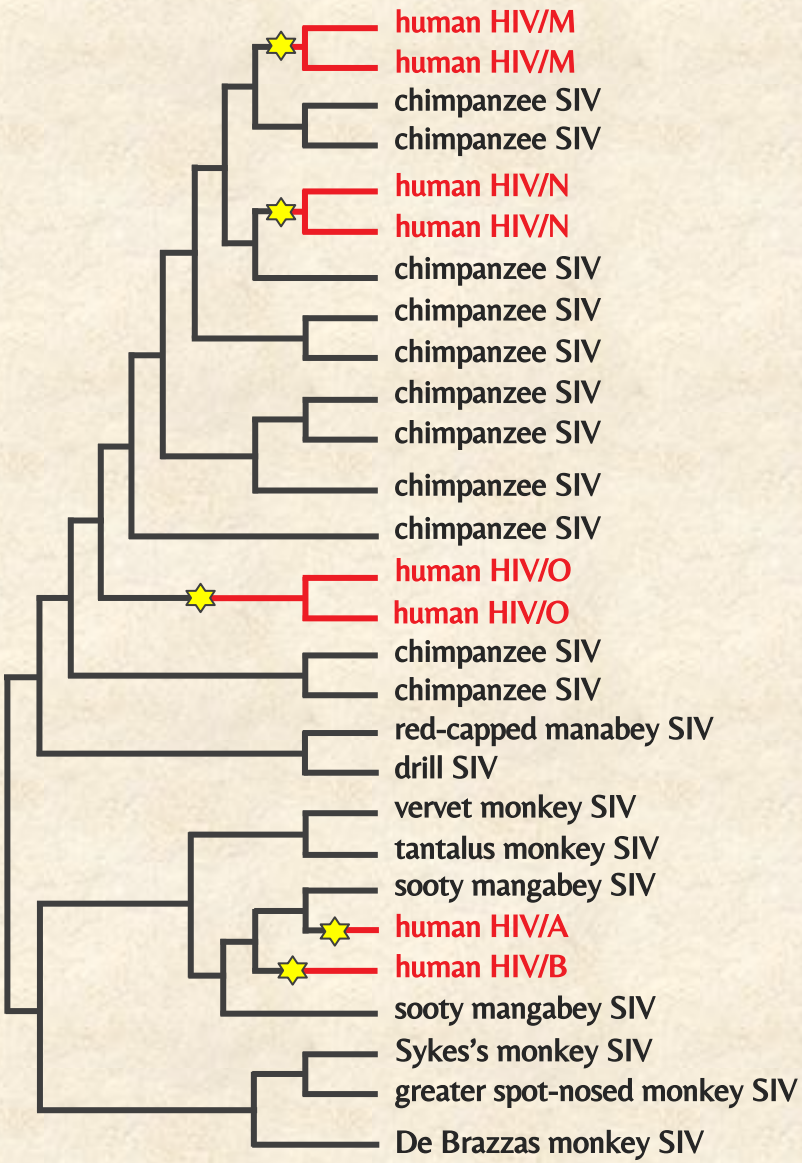
B
D
C
A
I

HIV Evolutionary Tree

SIVs (monkeys)

HIV (human)

human infection



Outline

- The Fastest Outbreak
- **Transforming Distance Matrices into Evolutionary Trees**
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

Constructing a Distance Matrix

Species	Alignment
Chimp	ACGTAGGCCT
Human	ATGTAAGACT
Seal	TCGAGAGCAC
Whale	TCGAAAGCAT

Constructing a Distance Matrix

$D_{i,j}$ = number of differing symbols between *i*-th and *j*-th rows of a multiple alignment.

Species	Alignment	Distance Matrix			
		Chimp	Human	Seal	Whale
Chimp	ACGTAGGCCT	0	3	6	4
Human	ATGTAAGACT	3	0	7	5
Seal	TCGAGAGCAC	6	7	0	2
Whale	TCGAAAGCAT	4	5	2	0

Constructing a Distance Matrix

$D_{i,j}$ = number of differing symbols between *i*-th and *j*-th rows of a multiple alignment.

Species	Alignment	Distance Matrix			
		Chimp	Human	Seal	Whale
Chimp	ACGTAGGCCT	0	3	6	4
Human	ATGTAAGACT	3	0	7	5
Seal	TCGAGAGCAC	6	7	0	2
Whale	TCGAAAGCAT	4	5	2	0

Constructing a Distance Matrix

$D_{i,j}$ = number of differing symbols between *i*-th and *j*-th rows of a multiple alignment.

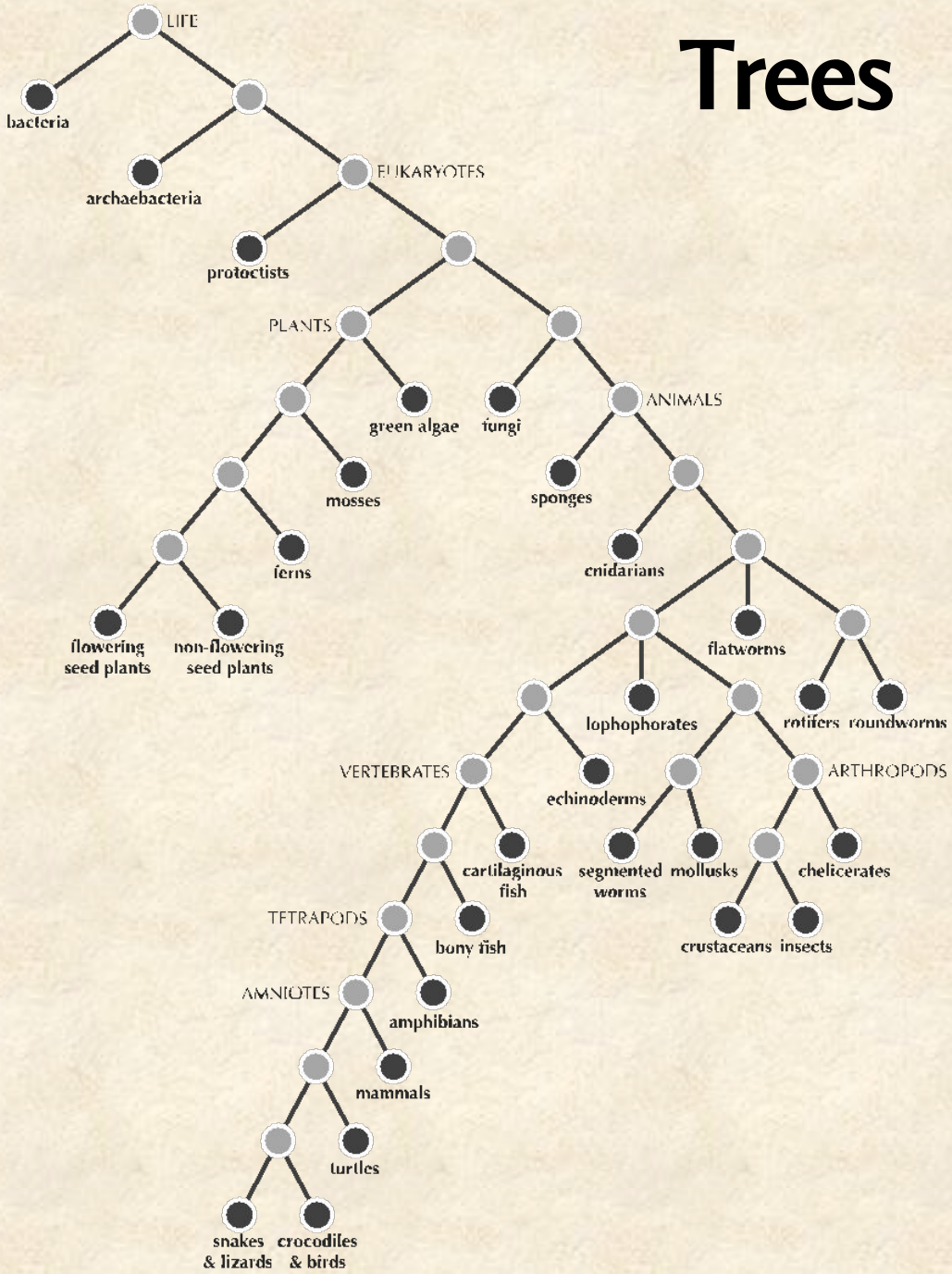
Species	Alignment	Distance Matrix			
		Chimp	Human	Seal	Whale
Chimp	ACGTAGGCCT	0	3	6	4
Human	ATGTAAGACT	3	0	7	5
Seal	TCGAGAGCAC	6	7	0	2
Whale	TCGAAAGCAT	4	5	2	0

STOP and Think: How else could we form a distance matrix?

Spike Protein

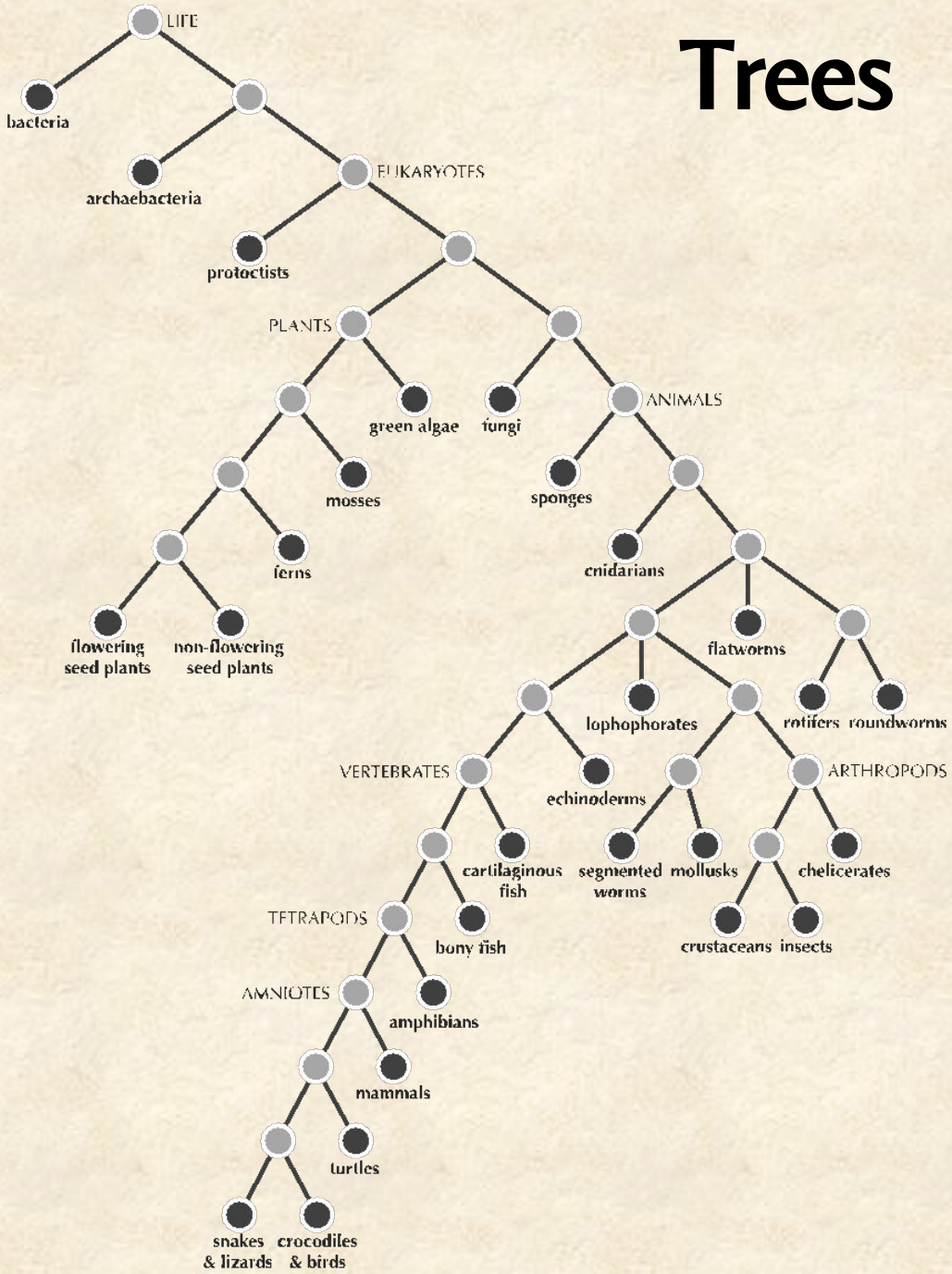
```
   1 mfifllfltl tsgsdldrct tfddvqapny tqhtssmrgv yypdeifrsd tlyltqdlfl
  61 pfysnvtgfh tinhtfgnpv ipfkdgiyfa ateksnvvrg wvfgstmnnk sqsviiinns
 121 tnvviracnf elcdnpffav skpmgtqtht mifdnafnct feyisdafsl dvseksgnfk
 181 hlrefvfknk dgflyvykgy qpidvvrdlp sgfntlkpif klplginitn frailtafsp
 241 aqdiwgtsaa ayfvgylkpt tfmlkydeng titdavdcsq nplaelkcsv ksfeidkgiy
 301 qtsnfrvvps gdvvrfpnit nlcpfgevfn atkfpsvyaw erkkisncva dysvlynstf
 361 fstfkcygvs atklndlcfs nvyadsfvvk gddvrqiapg qtgviadyny klpddfmgcv
 421 lawntrnida tstgnynyky rylrhgklrp ferdisnvpf spdgkpctpp alncywplnd
 481 ygfytttgig yqpyrvvvls fellnapatv cgpklstdli knqcvnfnfn gltgtgvltp
 541 sskrfqpfqq fgrdvsdftd svrdpktsei ldispcafgg vsvitpgtna ssevavlyqd
 601 vnctdvstai hadqltpawr iystgnnvfq tqagcligae hvdtsyecdi pigagicasy
 661 htvsllrsts qksivaytms lgadssiays nntiaiptnf sisittevmp vsmaktsvdc
 721 nmyicgdste canlllqygs fctqlnrals giaaeqdrnt revfaqvkqm yktptlkyfg
 781 gfnfsqilpd plkptkrsfi edllfnkvtl adagfmkqyg eclgdinard licaqkfngl
 841 tvlppllttdd miaaytaalv sgtatagwtf gagaalqipf amqmayrfng igvtqnvlye
 901 nqkqianqfn kaisqiqesl tttstalgkl qdvvnqnaqa lntlvkqlss nfgaissvln
 961 dilsrldkve aevqidrlit grlqslqtyv tqqliraaei rasanlaatk msecvlgqsk
1021 rvdfcgkgyh lmsfpqaaph gvvflhvtyv psqernftta paichegkay fpregvfvfn
1081 gtswfitqrn ffspqiittd ntfvsgncdv vigiinntvy dplqpeldsf keeldkyfkn
1141 htspdvdlgd isginasvvn iqkeidrlne vaknlnesli dlqelgkyeq yikwpwyvwl
1201 gfiagliaiv mvtillccmt sccsclkgac scgscckfde ddsepvlkgv klhyt
```

Trees



Tree: Connected graph containing no cycles.

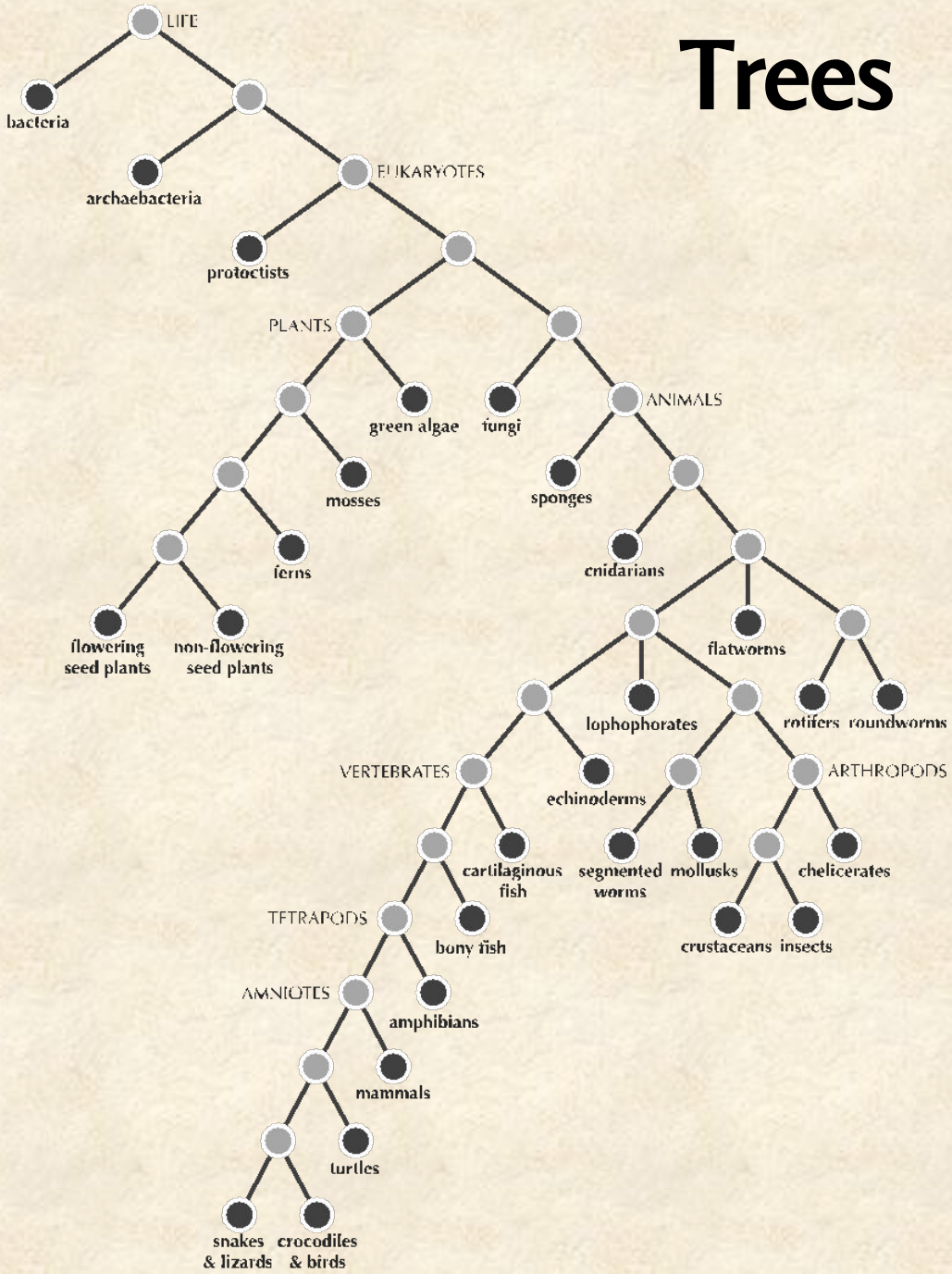
Trees



Tree: Connected graph containing no cycles.

Leaves (degree = 1): present-day species

Trees

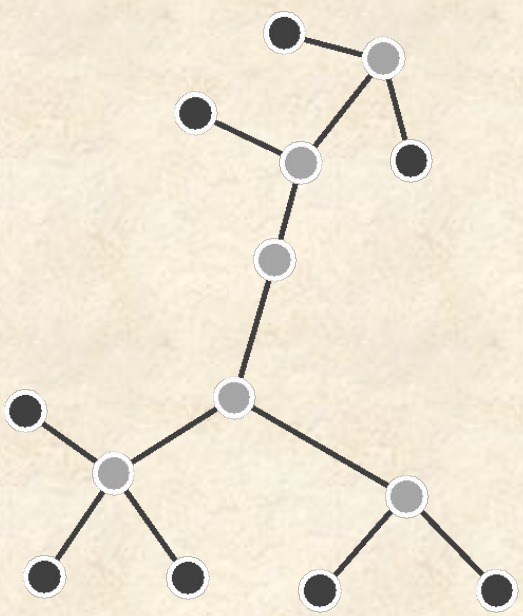


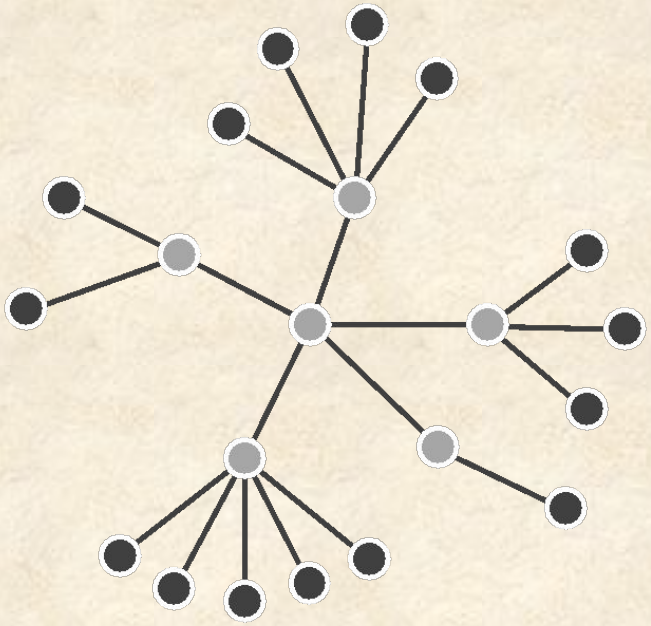
Tree: Connected graph containing no cycles.

Leaves (degree = 1): present-day species

Internal nodes (degree ≥ 1): ancestral species

Trees

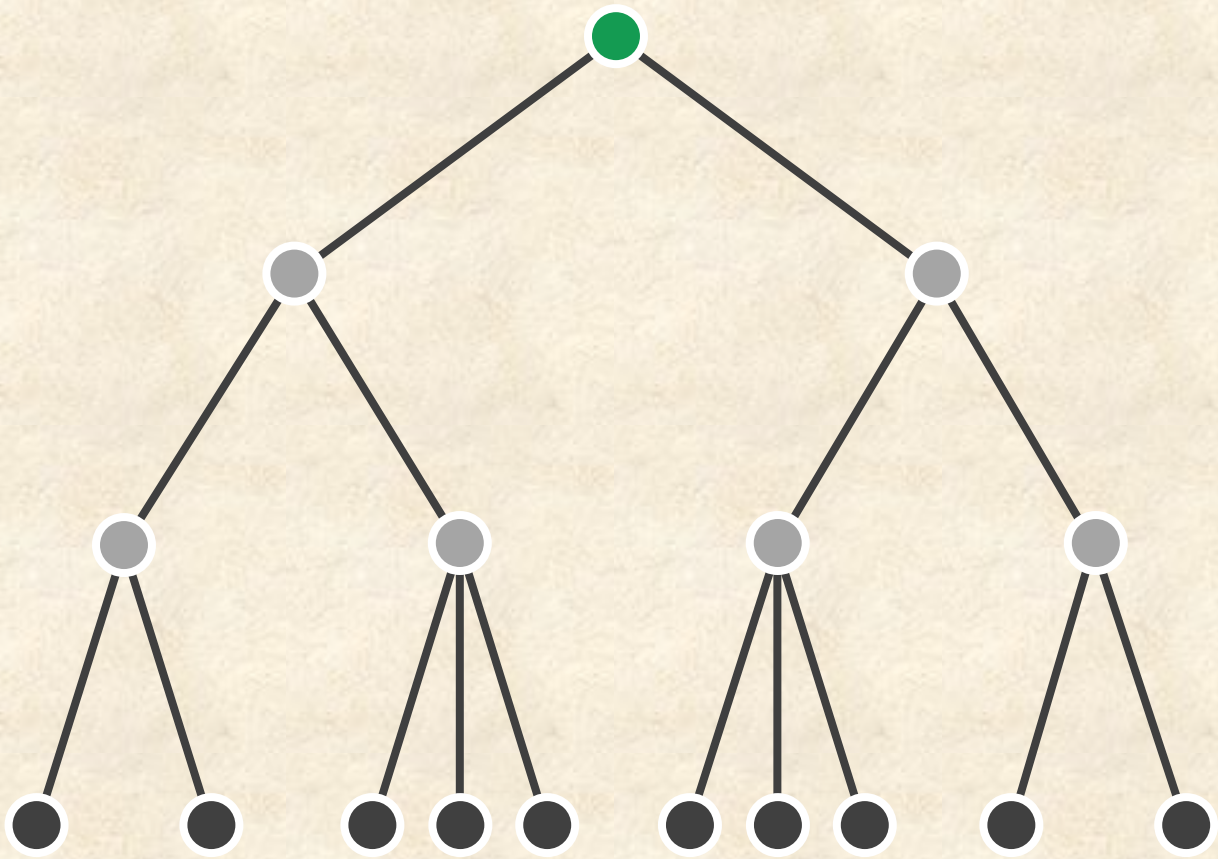




Exercise Break: Prove the following two statements.

1. Every tree with at least two nodes has at least two leaves.
2. Every tree with n nodes has exactly $n - 1$ edges.

Trees



Rooted tree: one node is designated as the **root** (most recent common ancestor)

Distance-Based Phylogeny

Distance-Based Phylogeny Problem: *Construct an evolutionary tree from a distance matrix.*

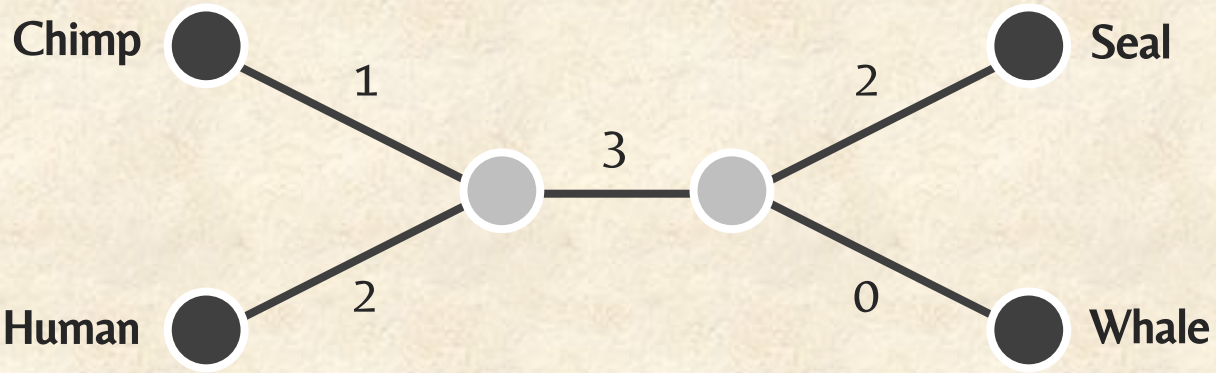
- **Input:** A distance matrix.
- **Output:** The unrooted tree "fitting" this distance matrix.

Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

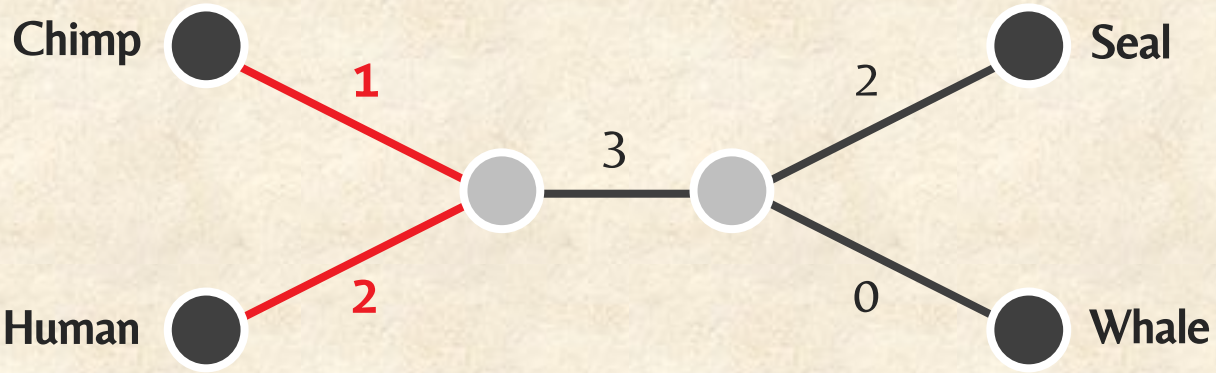
Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



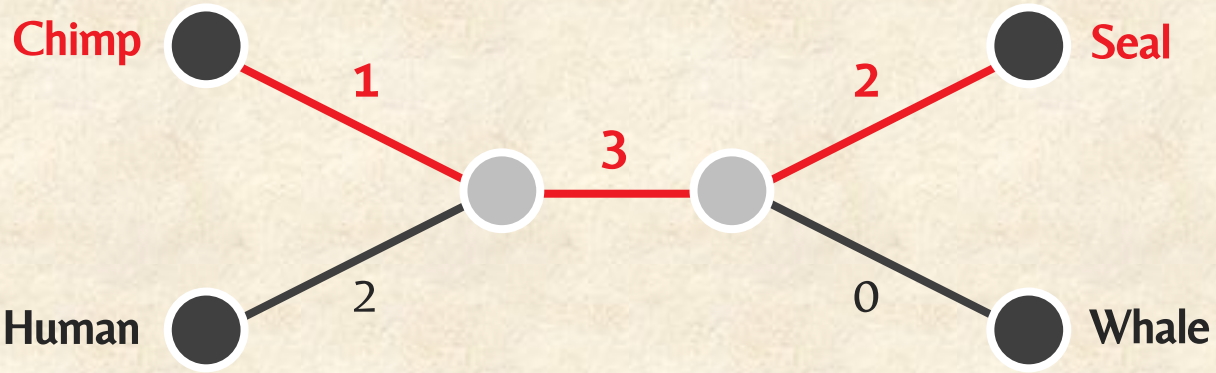
Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



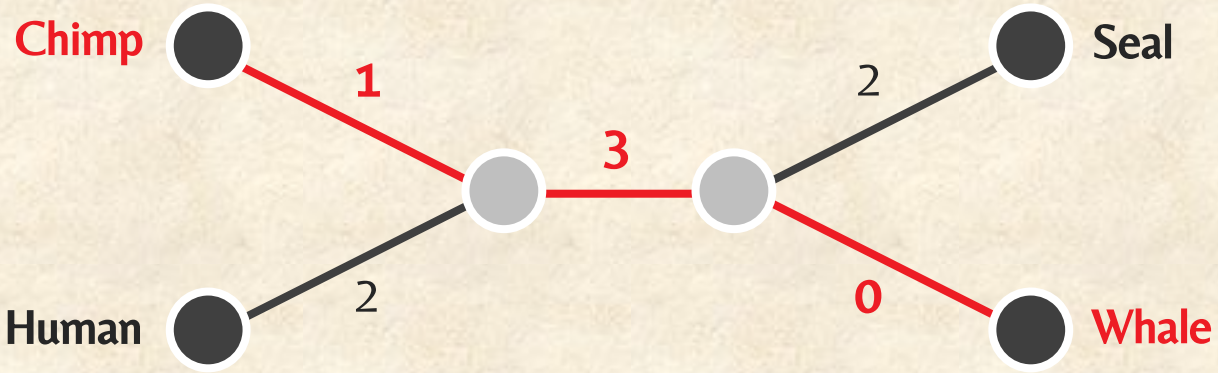
Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



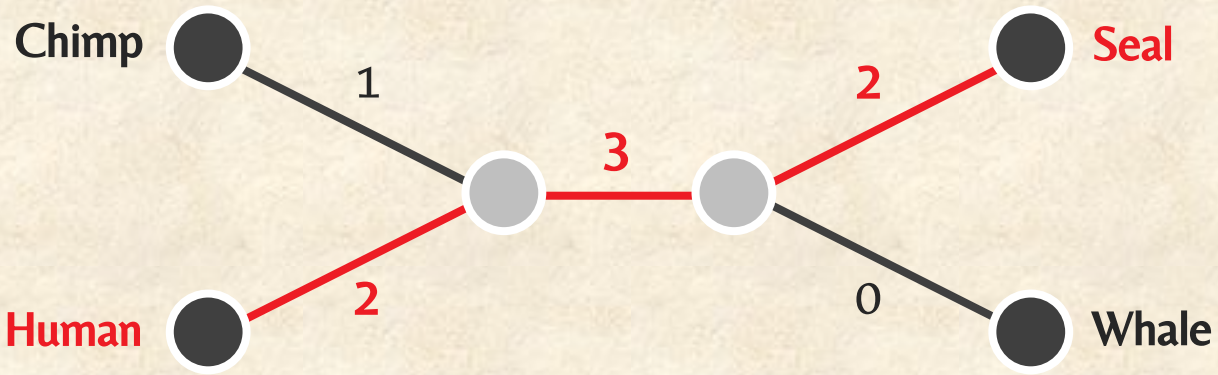
Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



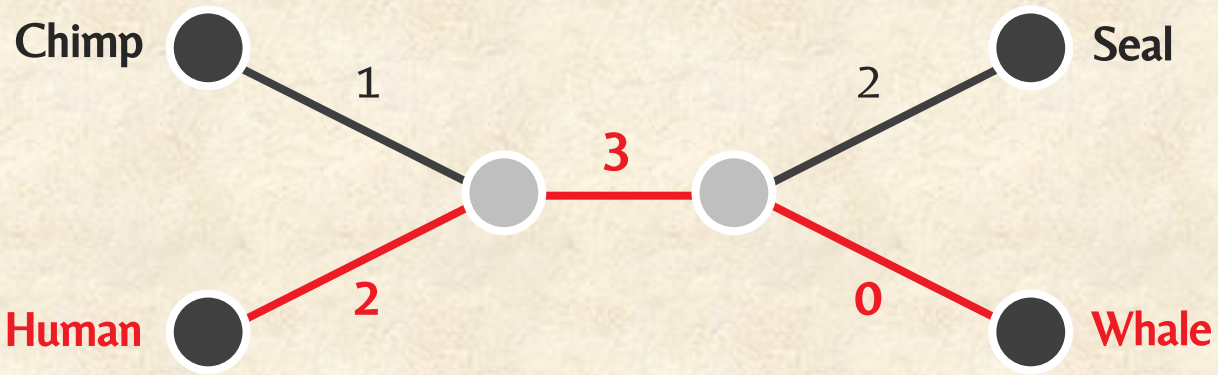
Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



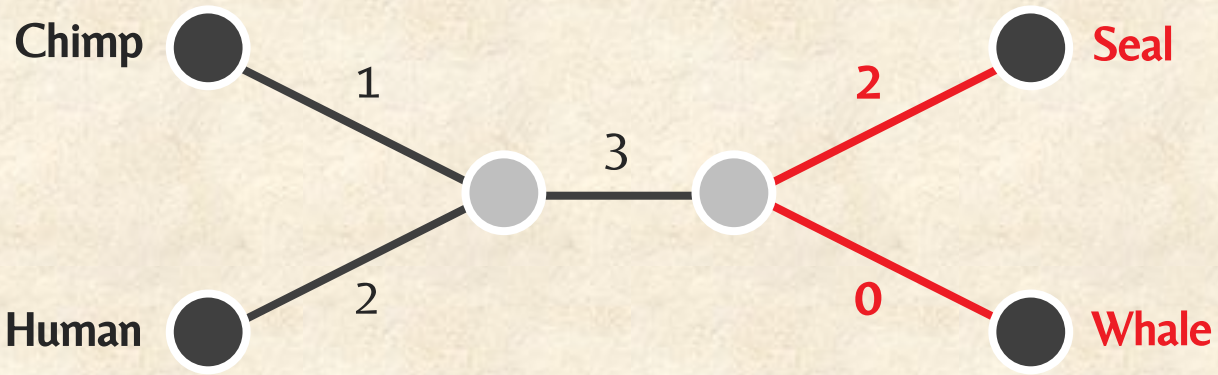
Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



Fitting a Tree to a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



Distance Between Leaves

Distance Between Leaves Problem: *Compute the distances between leaves in a weighted tree.*

- **Input:** A weighted tree with n leaves.
- **Output:** An n x n matrix $(d_{i,j})$, where $d_{i,j}$ is the length of the path between leaves i and j.

Code Challenge: Solve this problem.

Return to Distance-Based Phylogeny

Distance-Based Phylogeny Problem: *Construct an evolutionary tree from a distance matrix.*

- **Input:** A distance matrix.
- **Output:** The unrooted tree fitting this distance matrix.

STOP and Think: Now is this problem well-defined?

Return to Distance-Based Phylogeny

Exercise Break: Try fitting a tree to the following matrix.

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

No Tree Fits a Matrix

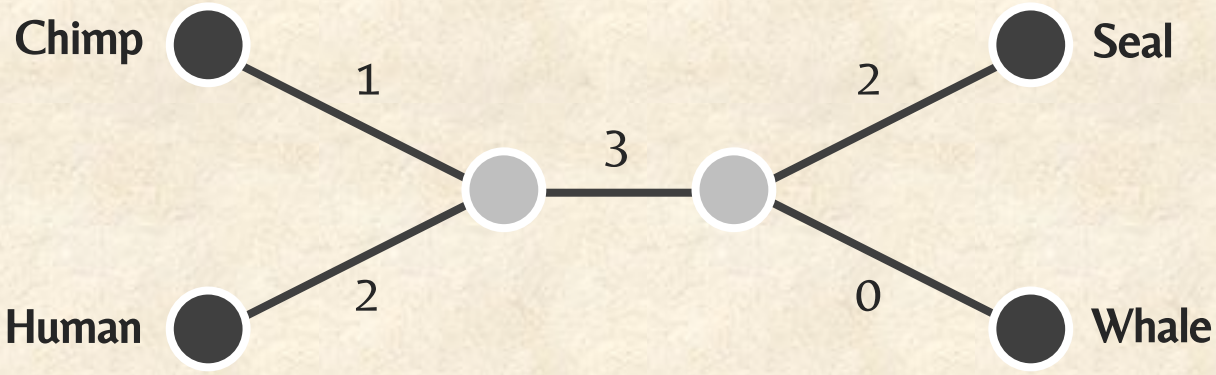
Exercise Break: Try fitting a tree to the following matrix.

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

Additive matrix: distance matrix such that there exists an unrooted tree fitting it.

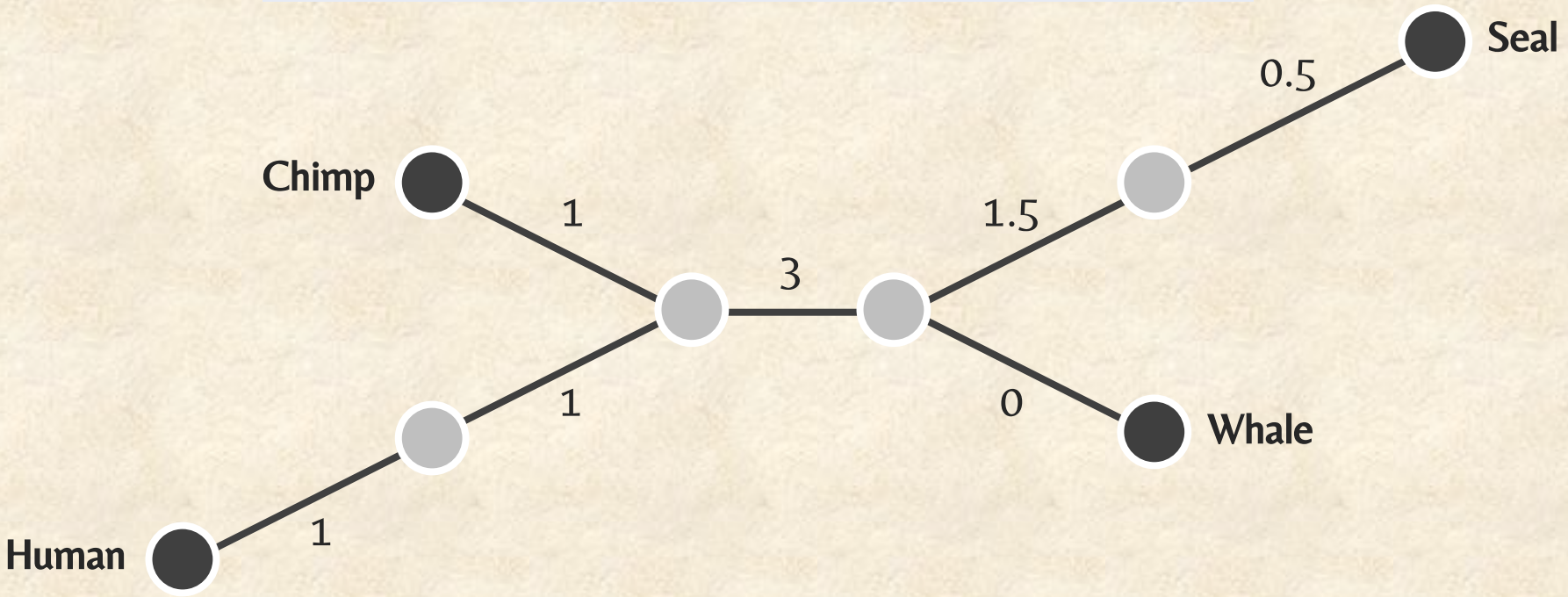
More Than One Tree Fits a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

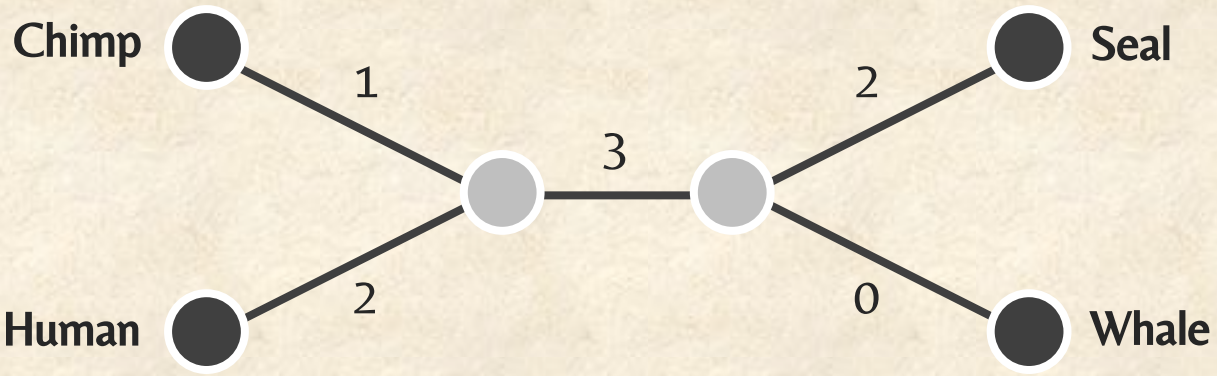


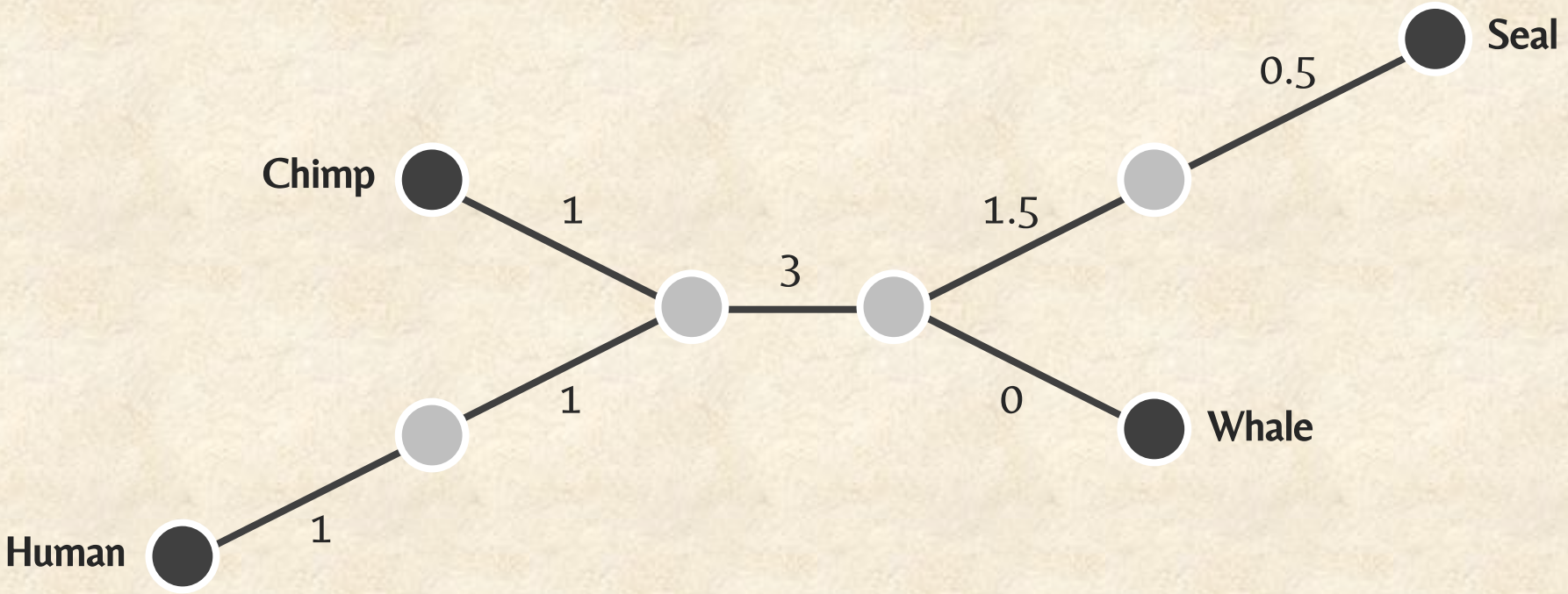
More Than One Tree Fits a Matrix

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

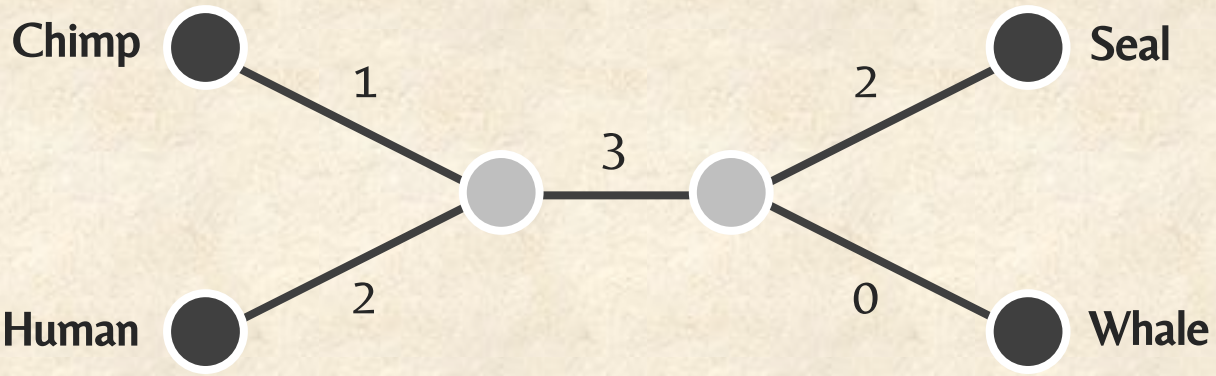


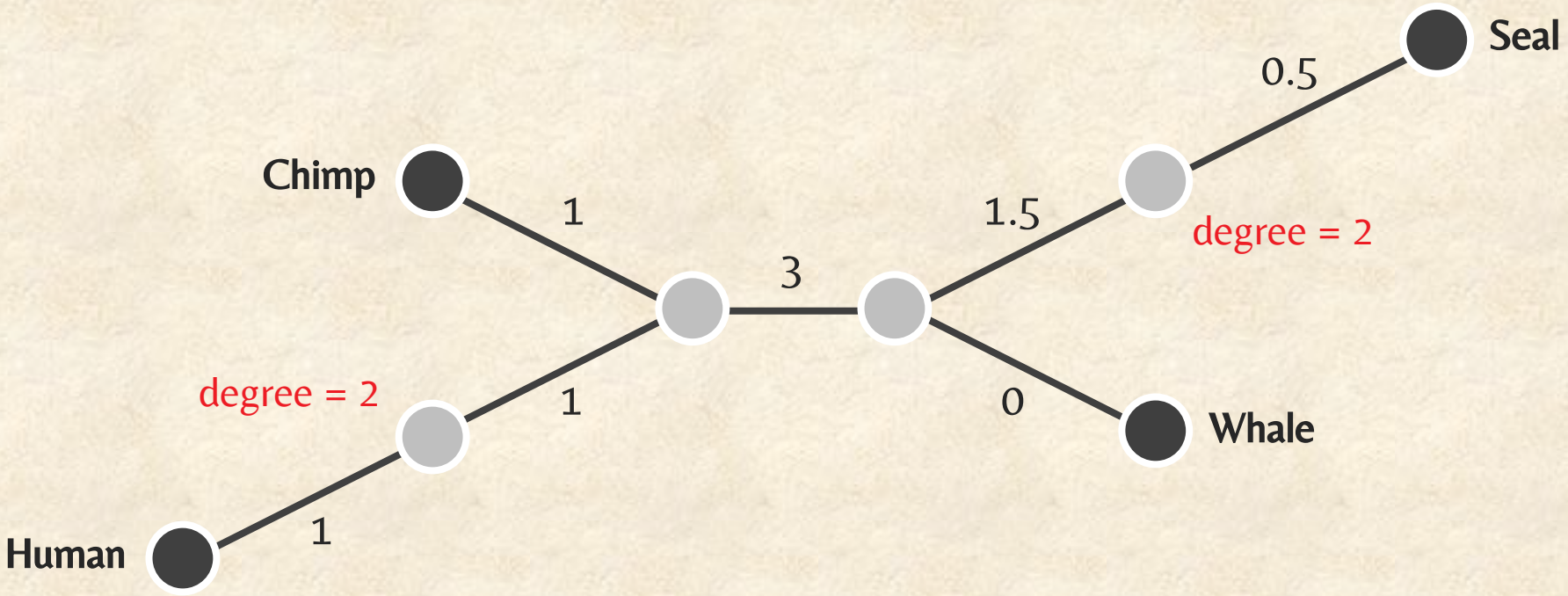
Which Tree is “Better”?



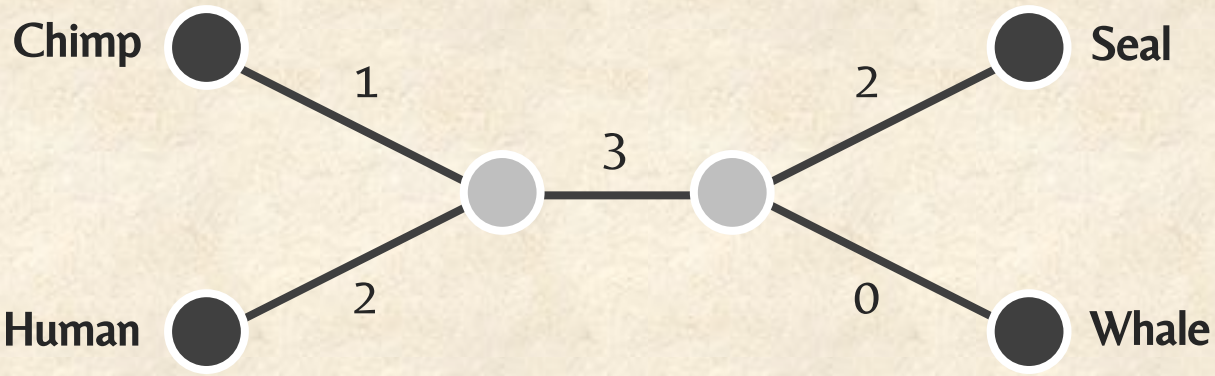


Which Tree is "Better"?



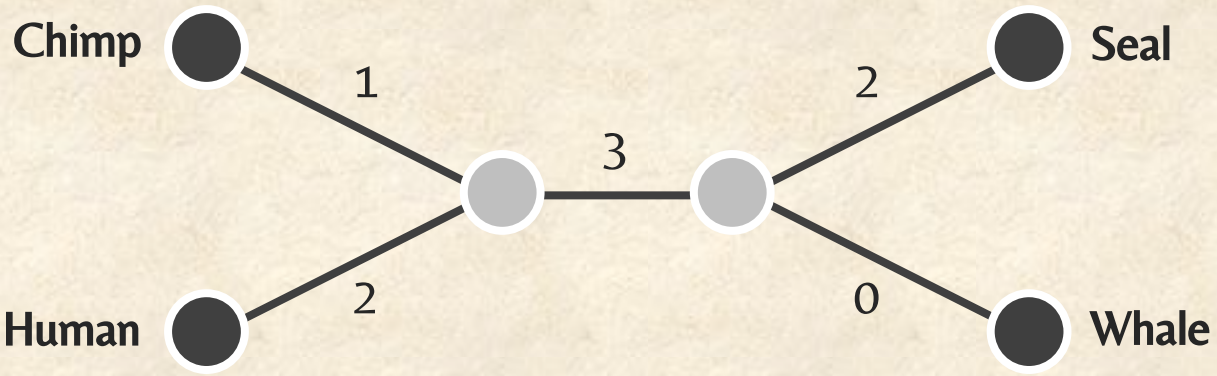


Which Tree is "Better"?



Simple tree: tree with no nodes of degree 2.

Which Tree is "Better"?



Simple tree: tree with no nodes of degree 2.

Theorem: There is a unique *simple* tree fitting an *additive* matrix.

Reformulating Distance-Based Phylogeny

Distance-Based Phylogeny Problem: *Construct an evolutionary tree from a distance matrix.*

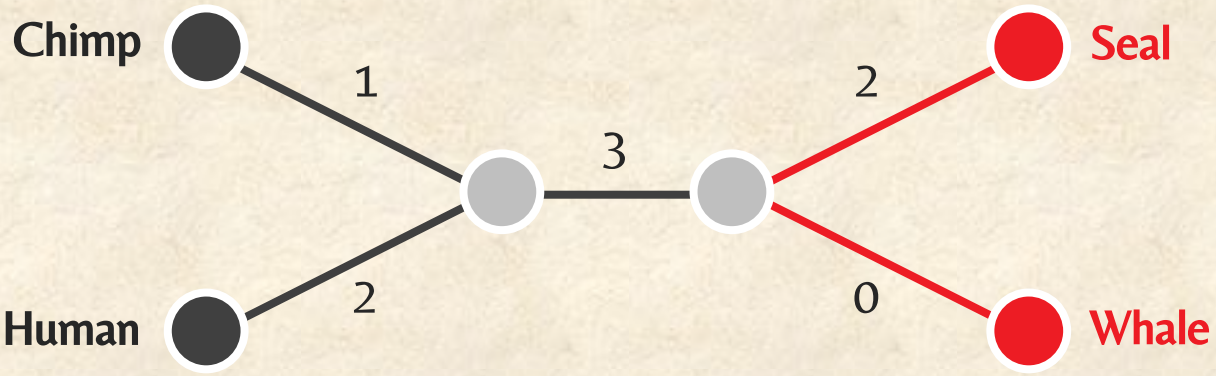
- **Input:** A distance matrix.
- **Output:** The simple tree fitting this distance matrix (if this matrix is additive).

Outline

- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- **Toward an Algorithm for Distance-Based Phylogeny Construction**
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

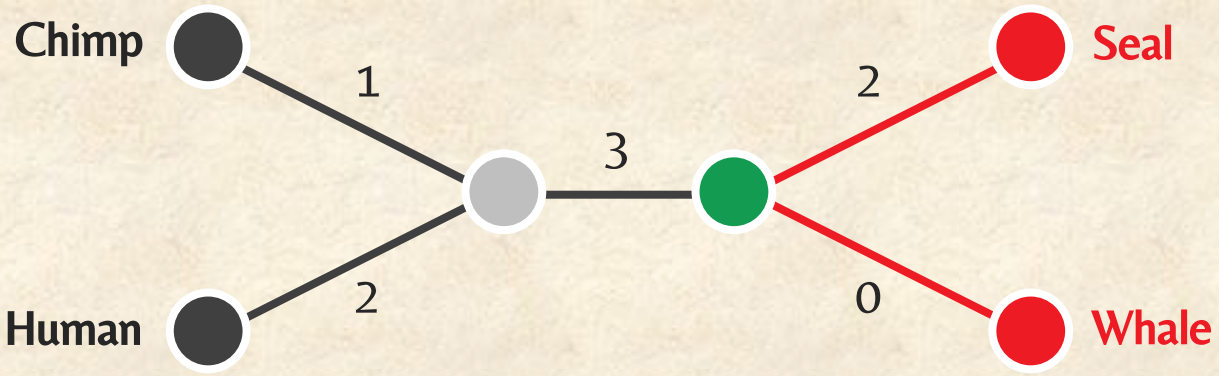
An Idea for Distance-Based Phylogeny

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



An Idea for Distance-Based Phylogeny

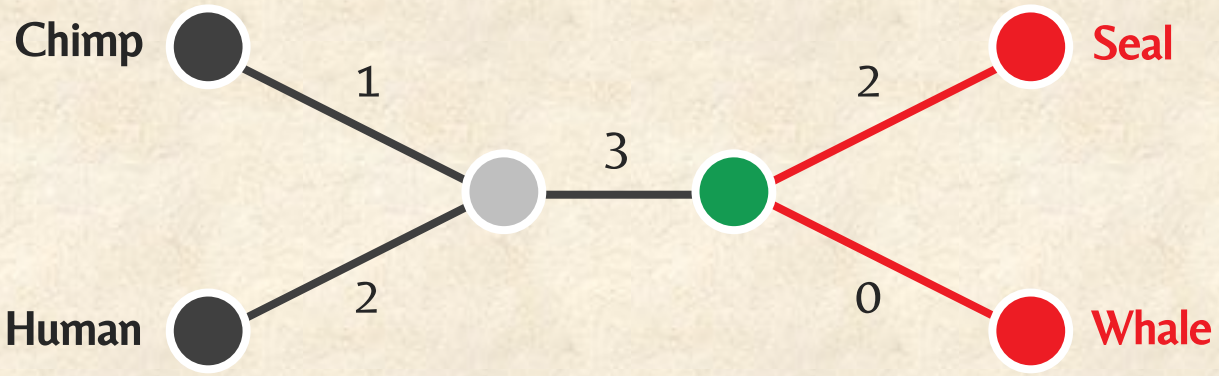
Seal and whale are **neighbors** (meaning they share the same **parent**).



An Idea for Distance-Based Phylogeny

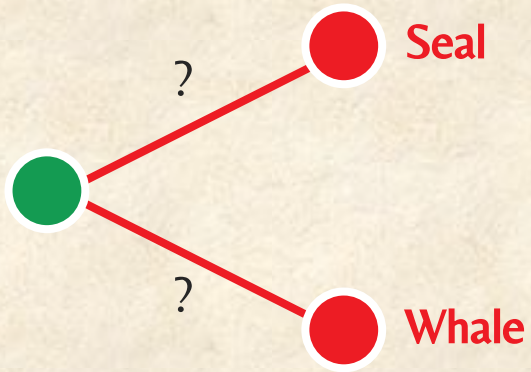
Seal and whale are **neighbors** (meaning they share the same **parent**).

Theorem: Every simple tree with at least two nodes has at least one pair of neighboring leaves.



An Idea for Distance-Based Phylogeny

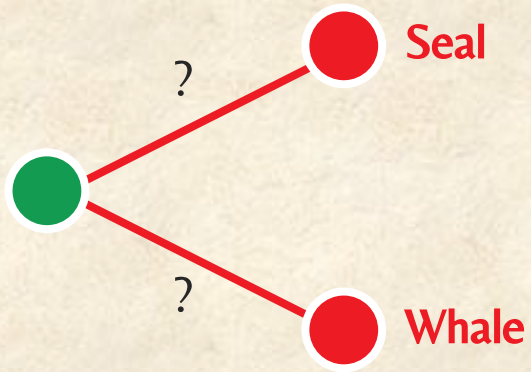
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



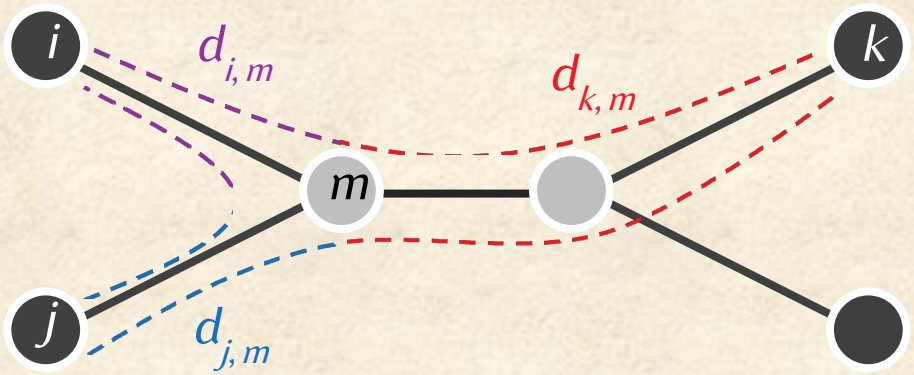
An Idea for Distance-Based Phylogeny

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

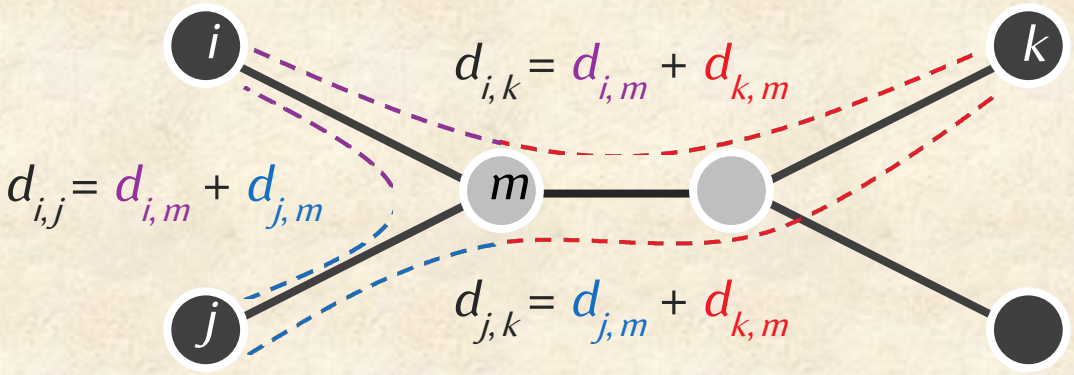
STOP and Think: How do we compute the unknown distances?



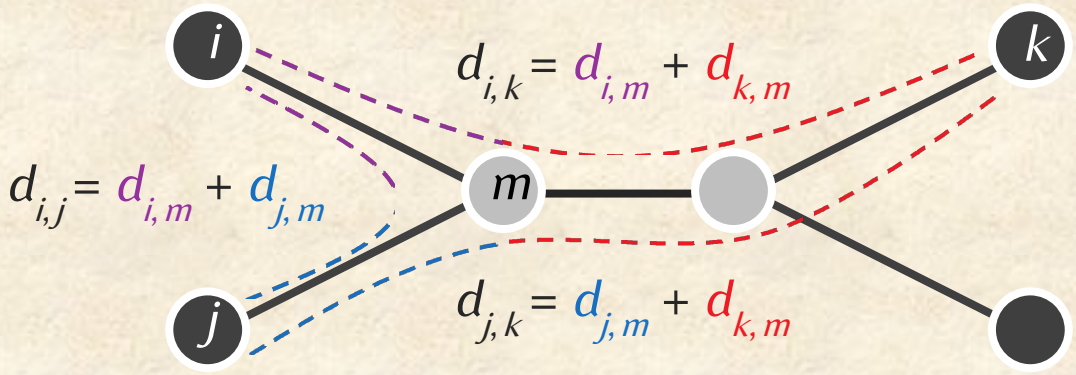
Toward a Recursive Algorithm



Toward a Recursive Algorithm

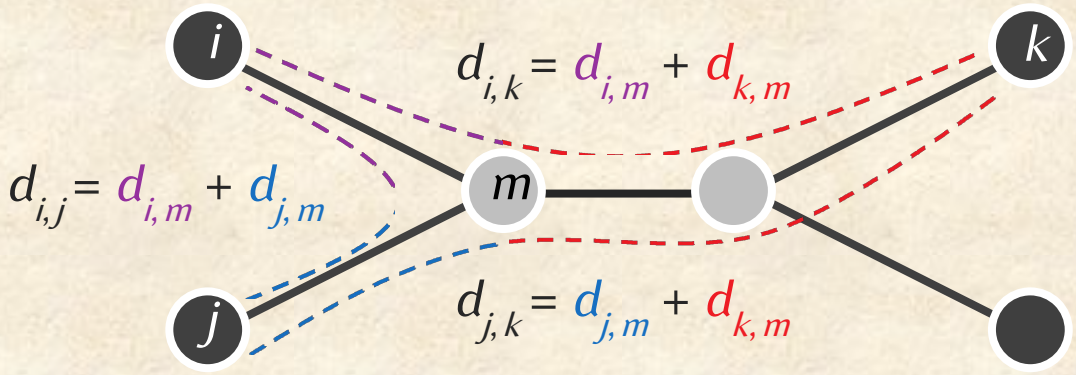


Toward a Recursive Algorithm



$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

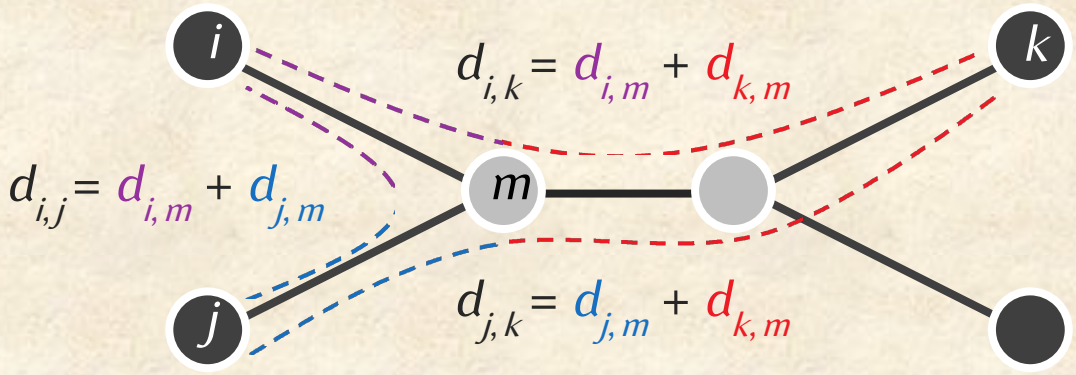
Toward a Recursive Algorithm



$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

$$d_{k,m} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

Toward a Recursive Algorithm

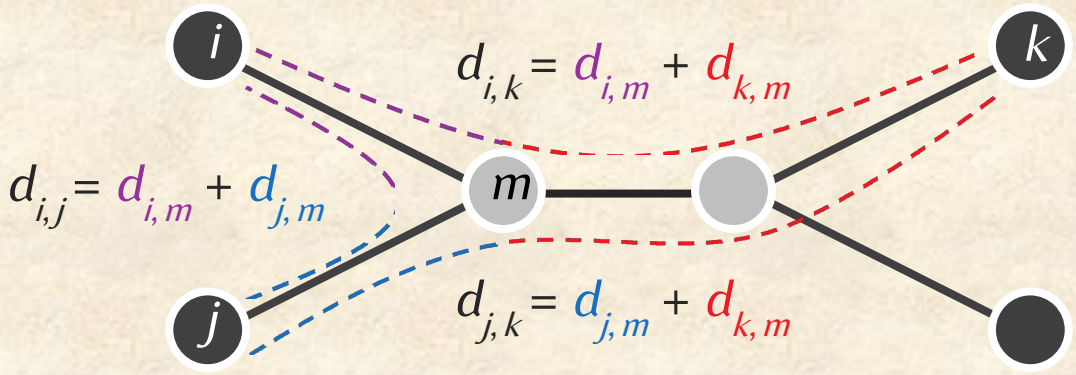


$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

$$d_{k,m} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

$$d_{k,m} = (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

Toward a Recursive Algorithm



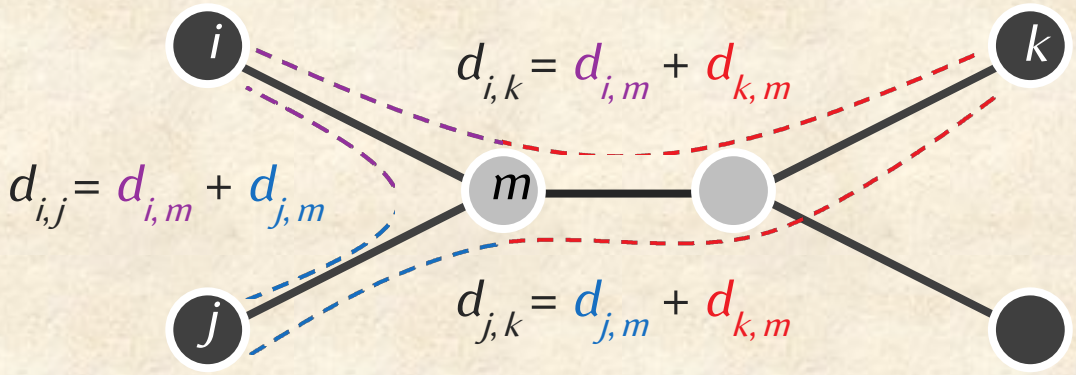
$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

$$d_{k,m} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

$$d_{k,m} = (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

$$\therefore d_{i,m} = D_{i,k} - (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

Toward a Recursive Algorithm



$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

$$d_{k,m} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

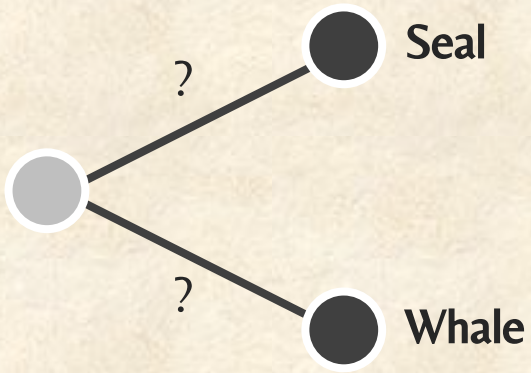
$$d_{k,m} = (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

$$\therefore d_{i,m} = D_{i,k} - (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

$$d_{i,m} = (D_{i,k} + D_{i,j} - D_{j,k}) / 2$$

An Idea for Distance-Based Phylogeny

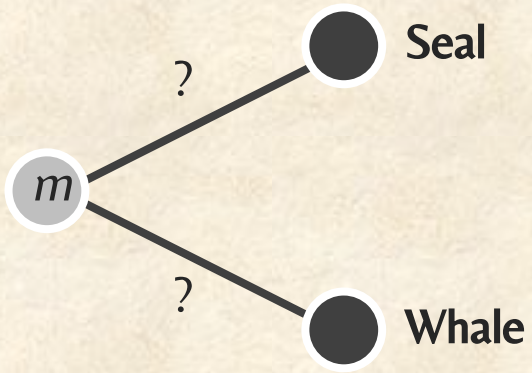
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$$d_{i,m} = (D_{i,k} + D_{i,j} - D_{j,k}) / 2$$

An Idea for Distance-Based Phylogeny

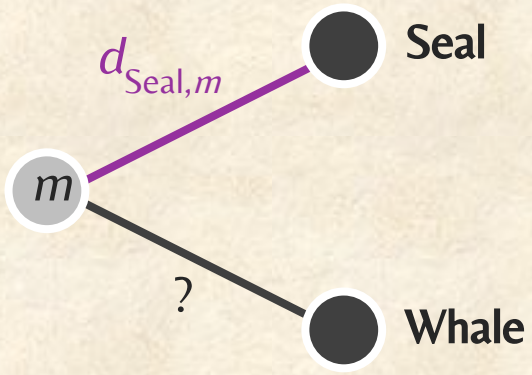
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$$d_{i,m} = (D_{i,k} + D_{i,j} - D_{j,k}) / 2$$

An Idea for Distance-Based Phylogeny

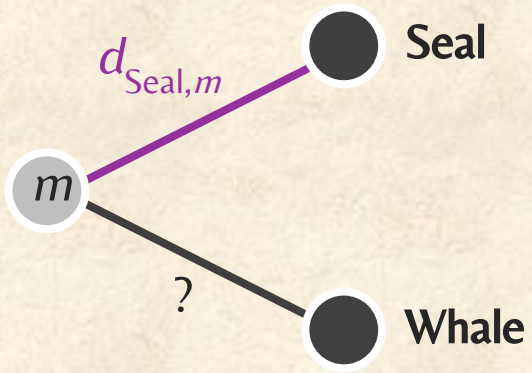
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$$d_{Seal,m} = (D_{Seal,k} + D_{Seal,j} - D_{j,k}) / 2$$

An Idea for Distance-Based Phylogeny

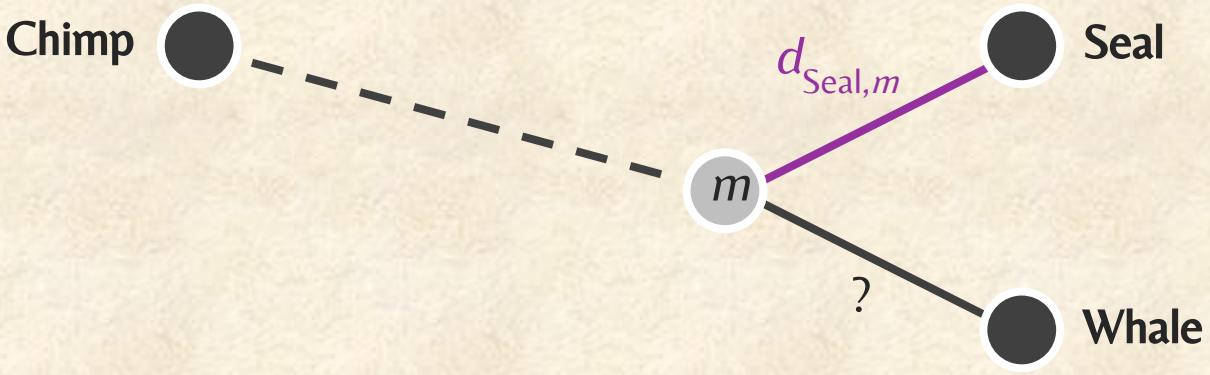
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$$d_{Seal,m} = (D_{Seal,k} + D_{Seal,Whale} - D_{Whale,k}) / 2$$

An Idea for Distance-Based Phylogeny

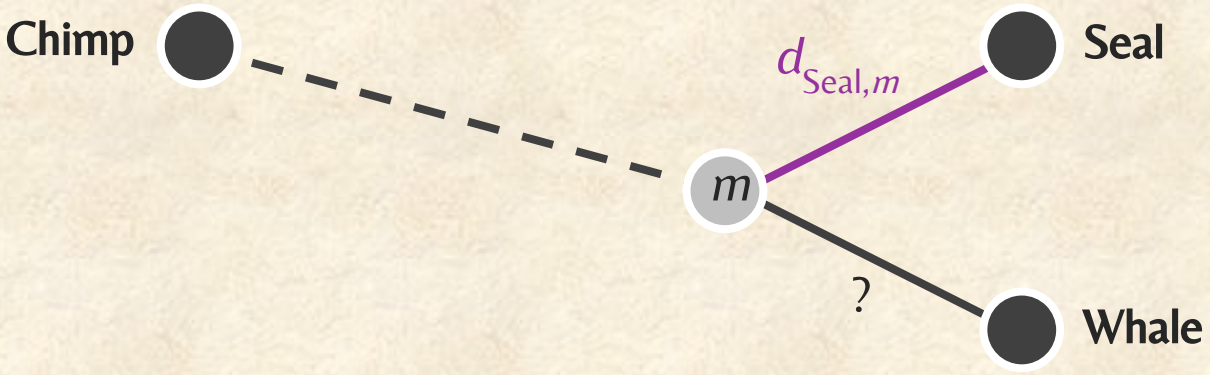
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$$d_{Seal,m} = (D_{Seal,Chimp} + D_{Seal,Whale} - D_{Whale,Chimp}) / 2$$

An Idea for Distance-Based Phylogeny

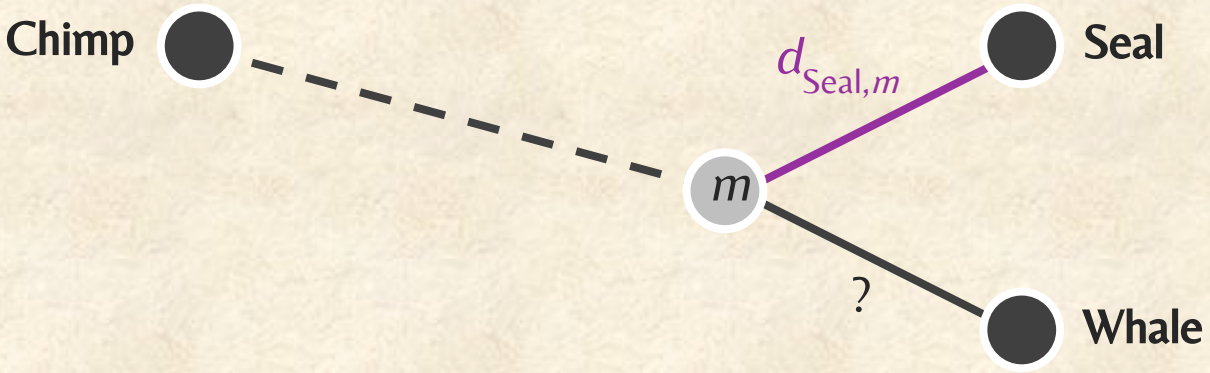
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$$d_{Seal,m} = (D_{Seal,Chimp} + D_{Seal,Whale} - D_{Whale,Chimp}) / 2$$

An Idea for Distance-Based Phylogeny

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

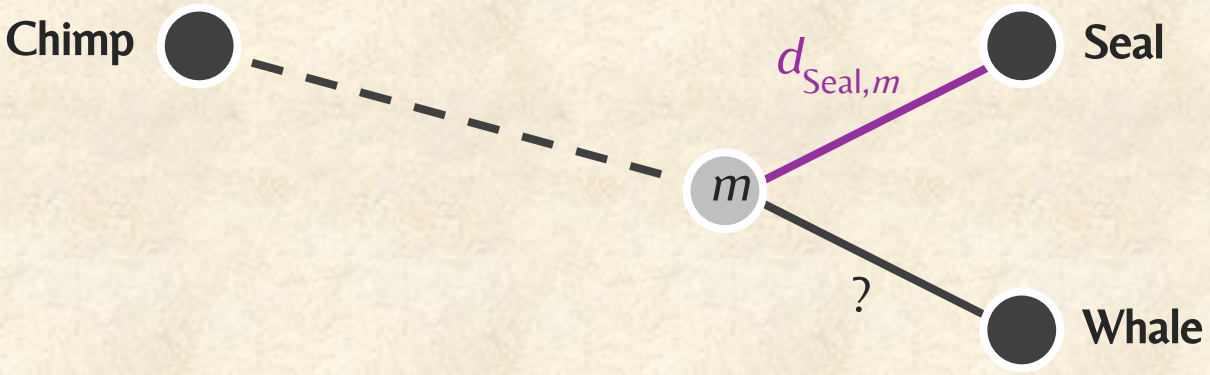


$$d_{Seal,m} = (D_{Seal,Chimp} + D_{Seal,Whale} - D_{Whale,Chimp}) / 2$$

$$= (6 + 2 - 4) / 2$$

An Idea for Distance-Based Phylogeny

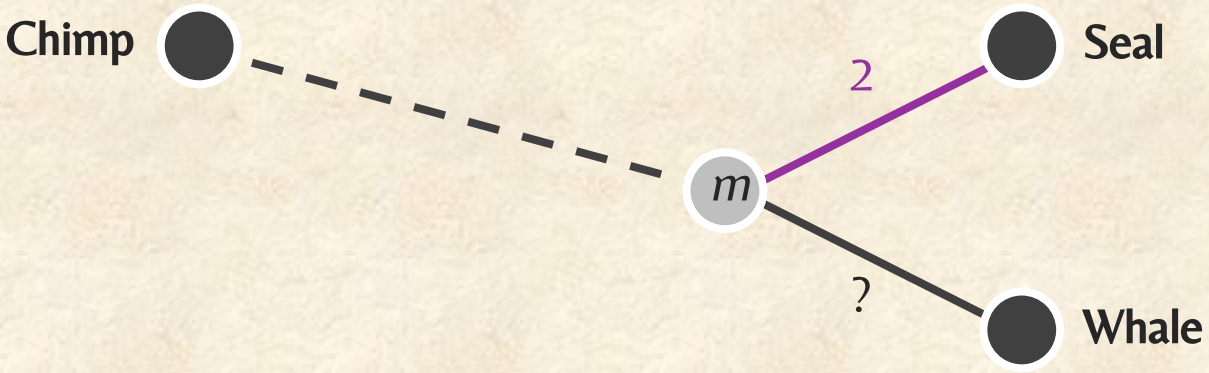
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$$d_{Seal,m} = (D^{6}_{Seal,Chimp} + D^{2}_{Seal,Whale} - D^{4}_{Whale,Chimp}) / 2$$

An Idea for Distance-Based Phylogeny

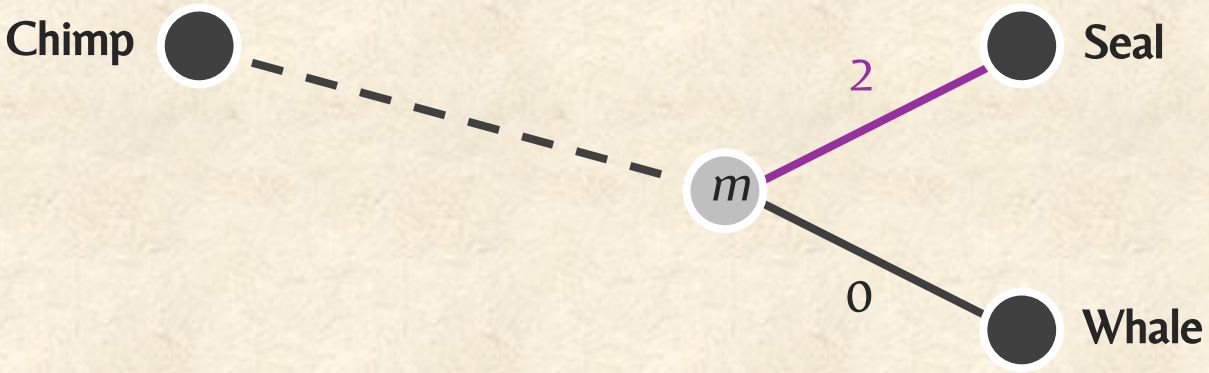
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$d_{Seal,m} = 2$

An Idea for Distance-Based Phylogeny

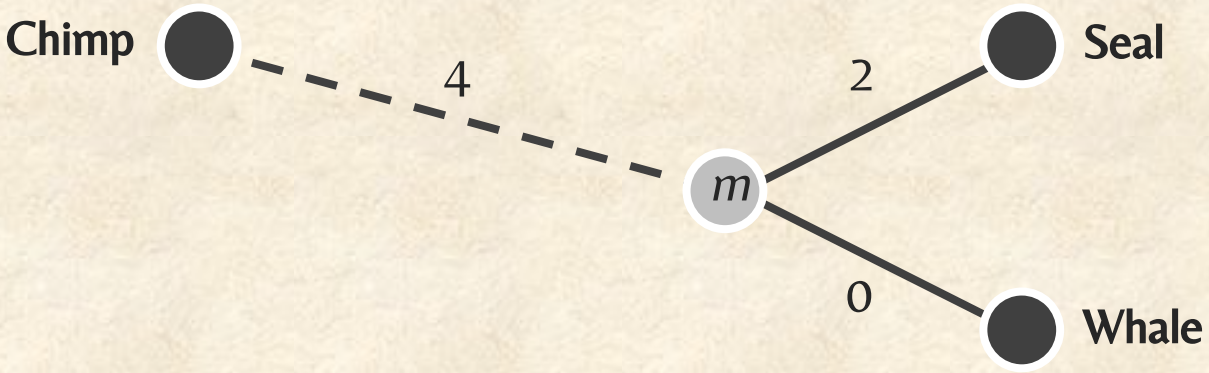
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



$d_{Seal,m} = 2$

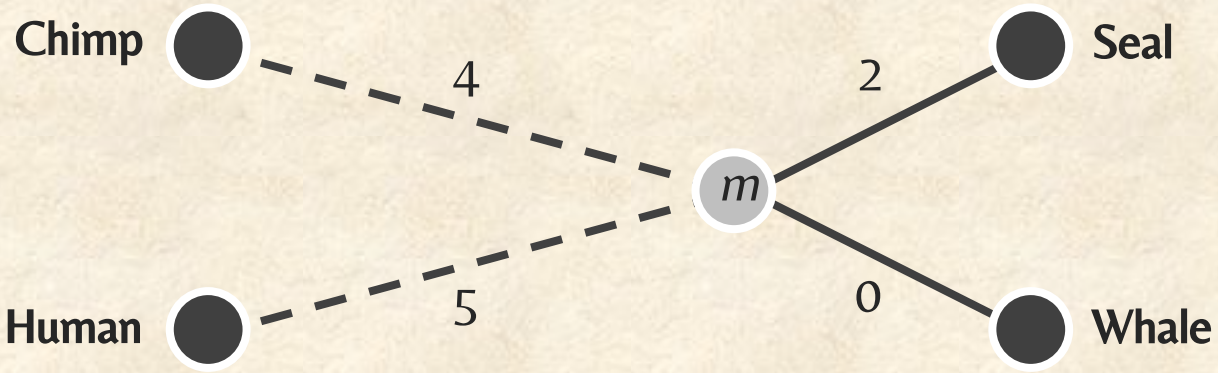
An Idea for Distance-Based Phylogeny

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



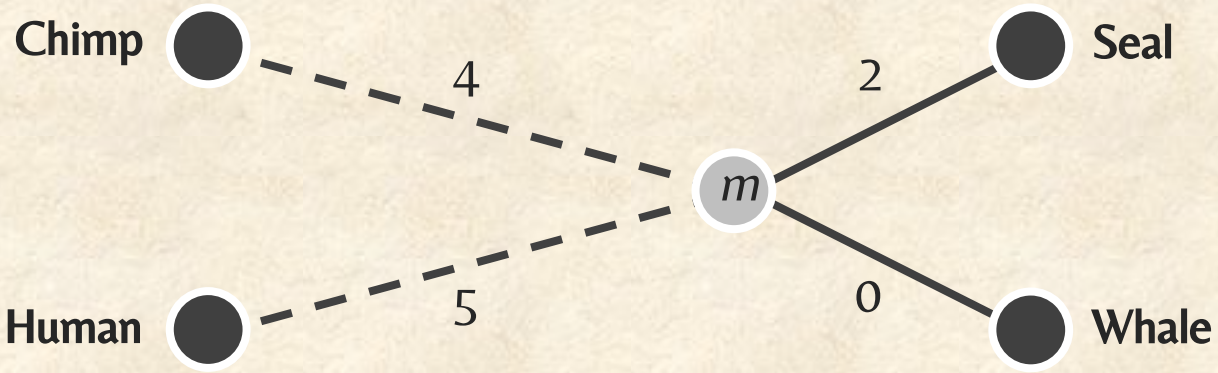
An Idea for Distance-Based Phylogeny

	Chimp	Human	Seal	Whale	*m*
Chimp	0	3	6	4	4
Human	3	0	7	5	5
Seal	6	7	0	2	2
Whale	4	5	2	0	0
m	4	5	2	0	0



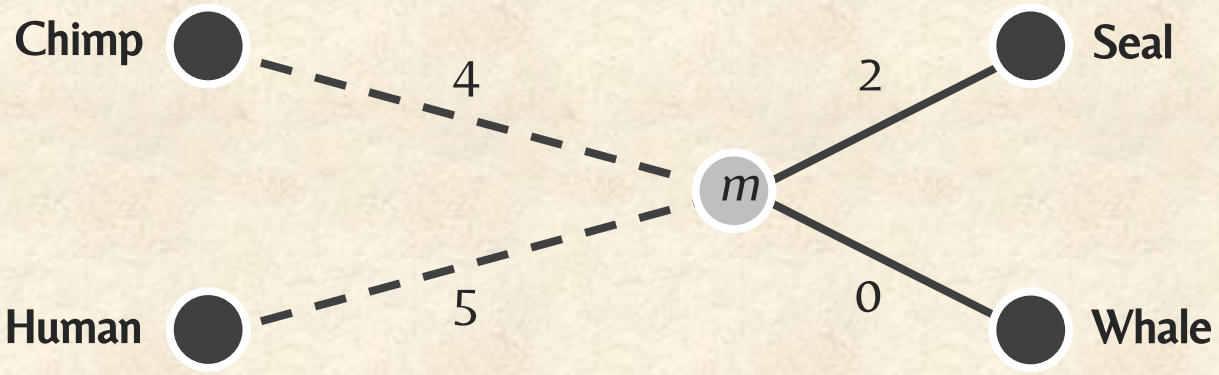
An Idea for Distance-Based Phylogeny

	Chimp	Human	Seal	Whale	*m*
Chimp	0	3	6	4	4
Human	3	0	7	5	5
Seal	6	7	0	2	2
Whale	4	5	2	0	0
m	4	5	2	0	0



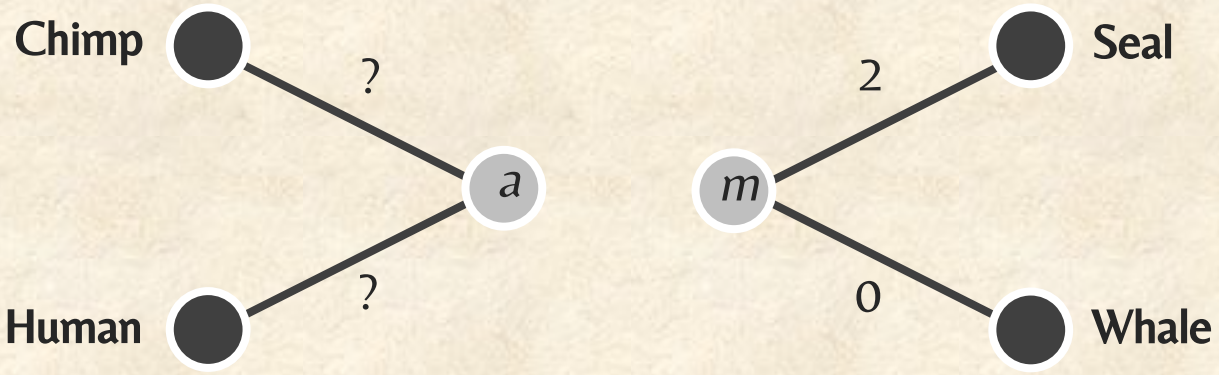
An Idea for Distance-Based Phylogeny

	Chimp	Human	*m*
Chimp	0	3	4
Human	3	0	5
m	4	5	0



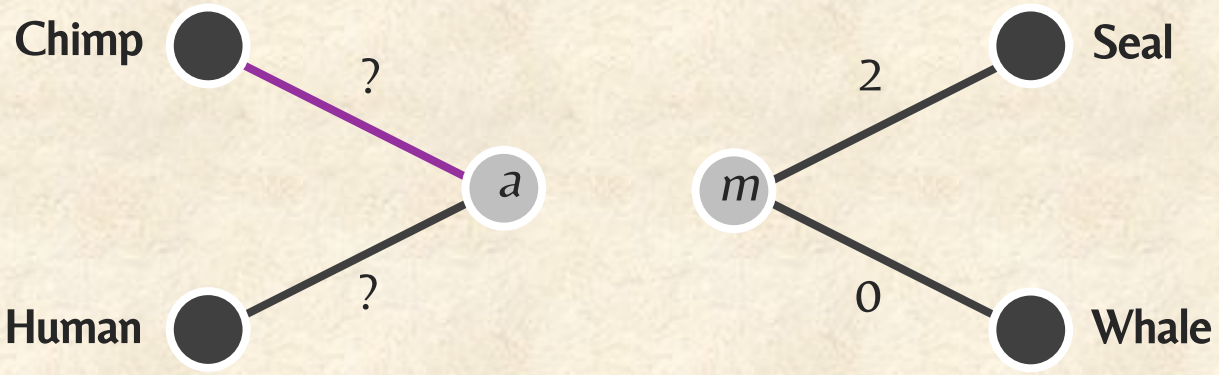
An Idea for Distance-Based Phylogeny

	Chimp	Human	*m*
Chimp	0	3	4
Human	3	0	5
m	4	5	0



An Idea for Distance-Based Phylogeny

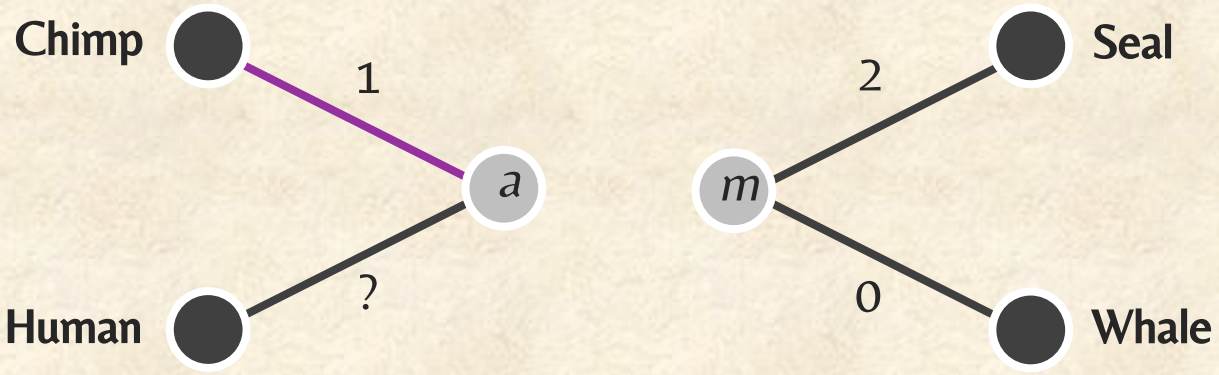
	Chimp	Human	*m*
Chimp	0	3	4
Human	3	0	5
m	4	5	0



$$d_{\text{Chimp},a} = (D_{\text{Chimp},m} + D_{\text{Chimp,Human}} - D_{\text{Human},m}) / 2$$

An Idea for Distance-Based Phylogeny

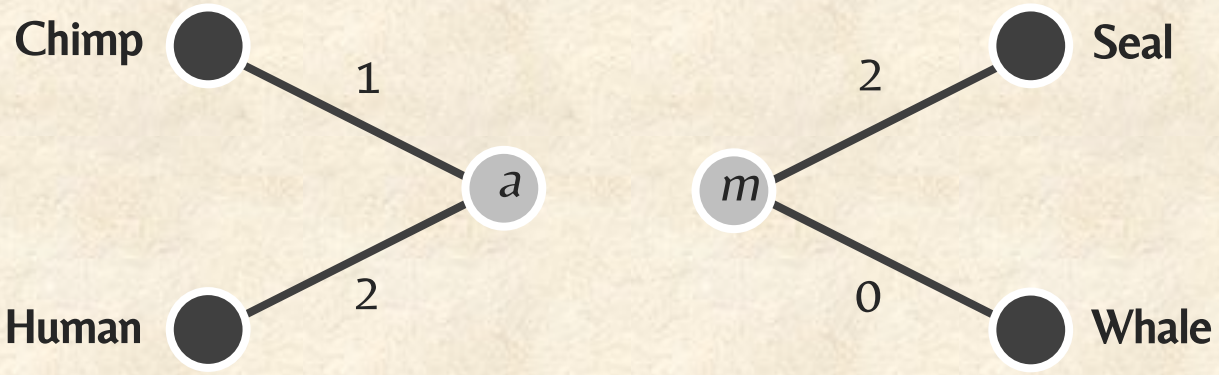
	Chimp	Human	*m*
Chimp	0	3	4
Human	3	0	5
m	4	5	0



$d_{\text{Chimp},a} = 1$

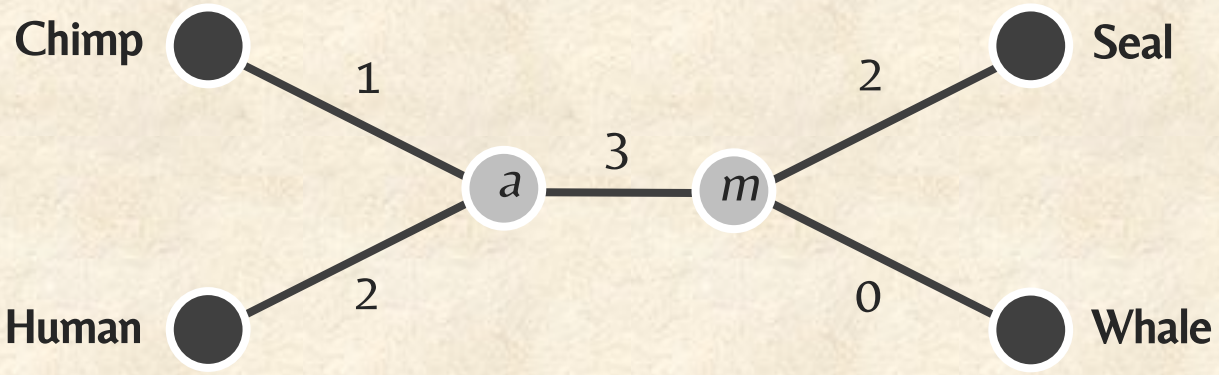
An Idea for Distance-Based Phylogeny

	Chimp	Human	*m*
Chimp	0	3	4
Human	3	0	5
m	4	5	0



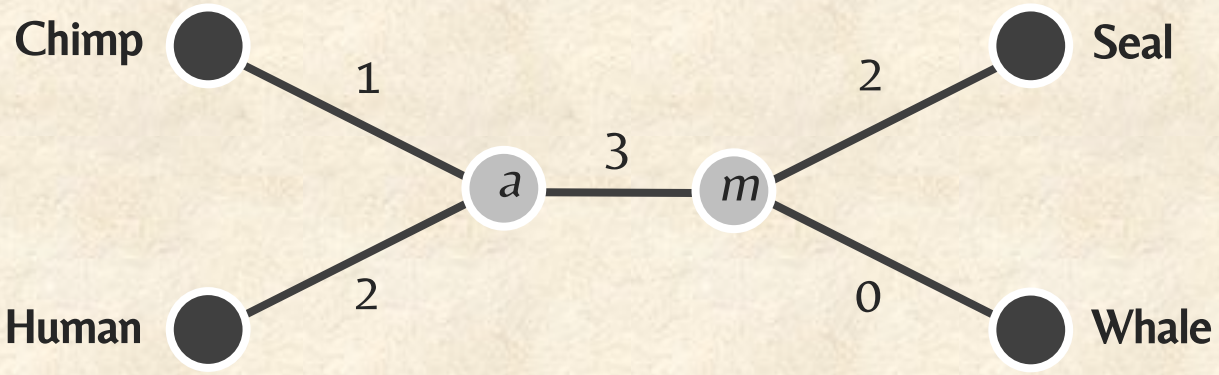
An Idea for Distance-Based Phylogeny

	Chimp	Human	*m*
Chimp	0	3	4
Human	3	0	5
m	4	5	0



An Idea for Distance-Based Phylogeny

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



An Idea for Distance-Based Phylogeny

Exercise Break: Apply this recursive approach to the distance matrix below.

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

Outline

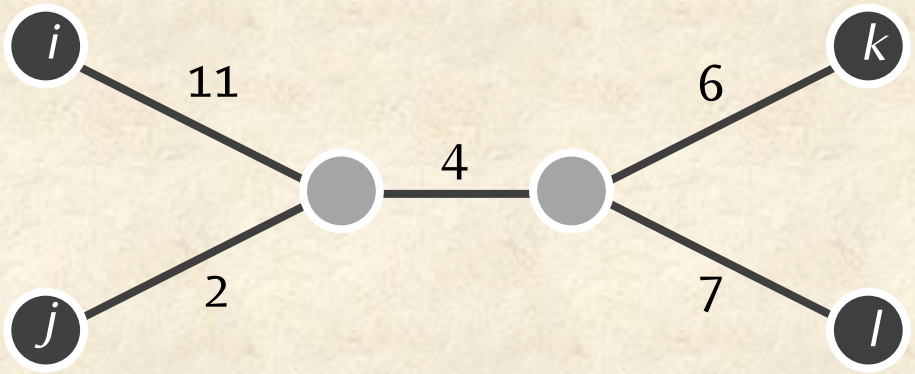
- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- **Additive Phylogeny**
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

What Was Wrong With Our Algorithm?

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

What Was Wrong With Our Algorithm?

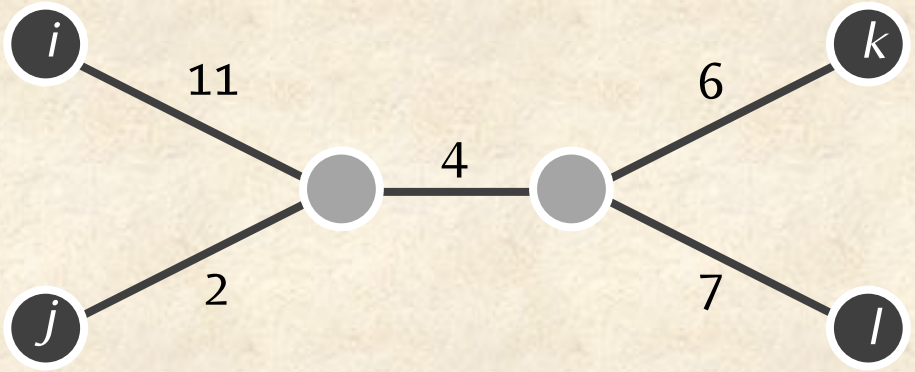
	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0



What Was Wrong With Our Algorithm?

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	**12**	13
k	21	12	0	13
l	22	13	13	0

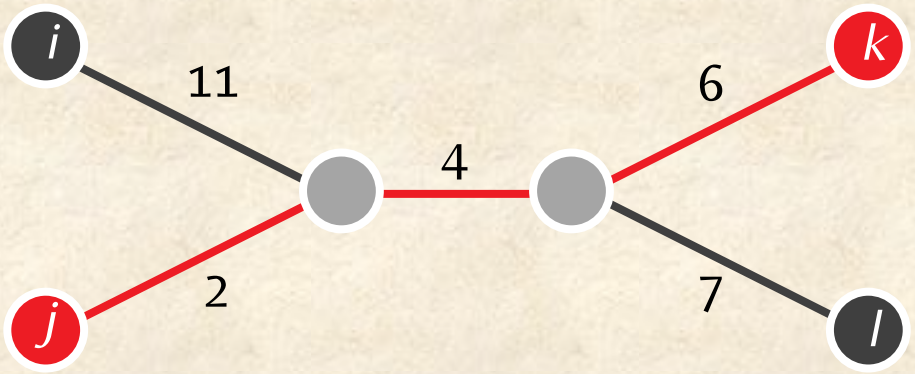
minimum element is $D_{j,k}$



What Was Wrong With Our Algorithm?

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	**12**	13
k	21	12	0	13
l	22	13	13	0

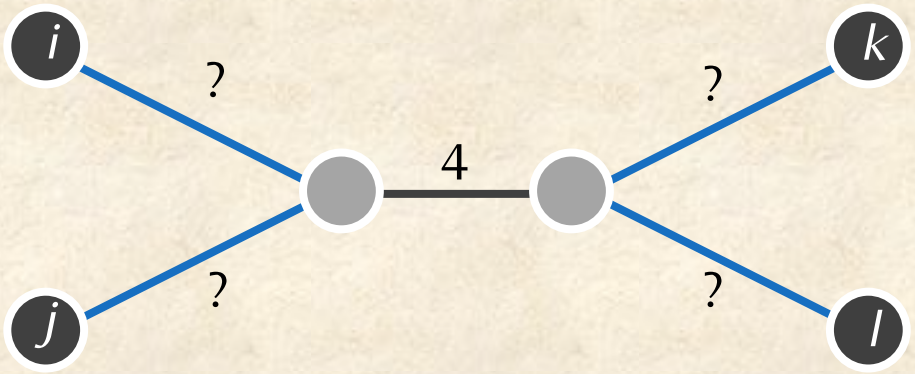
minimum element is $D_{j,k}$



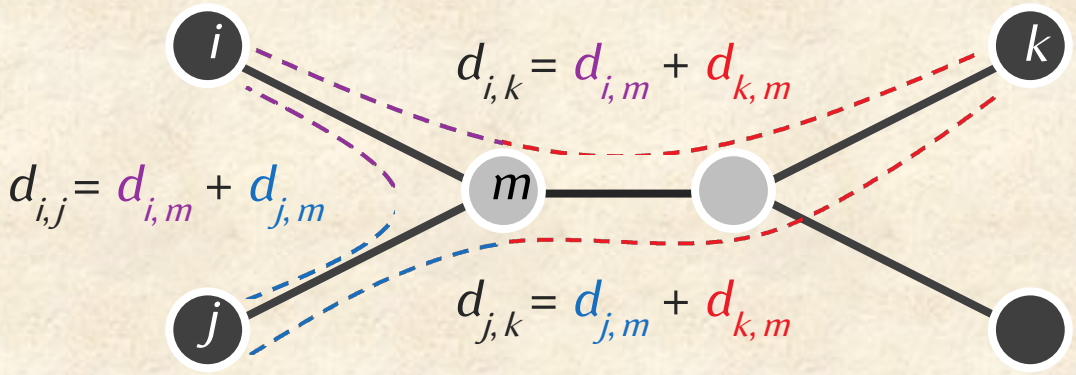
j and *k* are **not** neighbors!

From Neighbors to Limbs

Rather than trying to find **neighbors**, let's instead try to compute the length of **limbs**, the edges attached to leaves.



From Neighbors to Limbs



$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

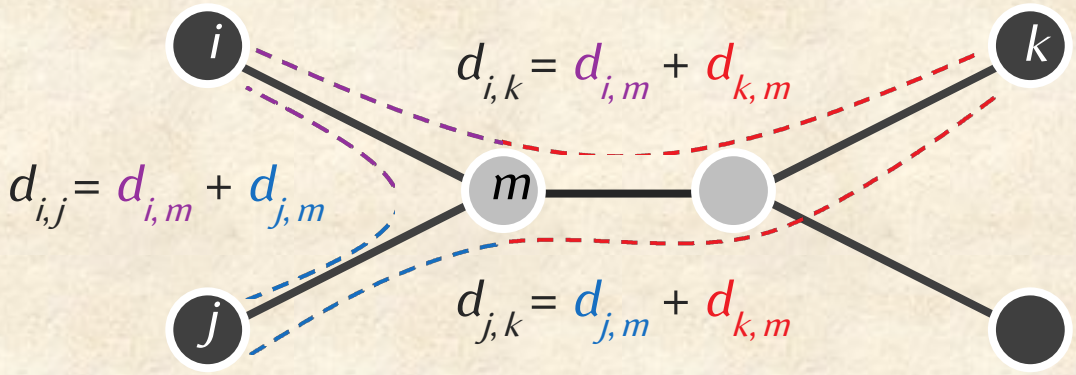
$$d_{k,m} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

$$d_{k,m} = (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

$$\therefore d_{i,m} = D_{i,k} - (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

$$d_{i,m} = (D_{i,k} + D_{i,j} - D_{j,k}) / 2$$

From Neighbors to Limbs



$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

$$d_{k,m} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

$$d_{k,m} = (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

$$\therefore d_{i,m} = D_{i,k} - (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

$$d_{i,m} = (D_{i,k} + D_{i,j} - D_{j,k}) / 2$$

Assumes that *i* and *j* are *neighbors*...

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(*i*) is equal to the minimum value of $(D_{i,k} + D_{i,j} - D_{j,k})/2$ over all leaves j and k.

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(*i*) is equal to the minimum value of $(D_{i,k} + D_{i,j} - D_{j,k})/2$ over all leaves *j* and *k*.

Limb Length Problem: *Compute the length of a limb in the simple tree fitting an additive distance matrix.*

- **Input:** An additive distance matrix *D* and an integer *j*.
- **Output:** The length of the limb connecting leaf *j* to its parent, *LimbLength*(*j*).

Code Challenge: Solve the Limb Length Problem.

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(chimp) is equal to the minimum value of $(D_{\text{chimp},k} + D_{\text{chimp},j} - D_{j,k})/2$ over all leaves j and k.

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(chimp) is equal to the minimum value of $(D_{\text{chimp},k} + D_{\text{chimp},j} - D_{j,k})/2$ over all leaves j and k.

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

$$(D_{\text{chimp, human}} + D_{\text{chimp, seal}} - D_{\text{human, seal}}) / 2 \qquad = (3 + 6 - 7) / 2 = 1$$

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(chimp) is equal to the minimum value of $(D_{\text{chimp},k} + D_{\text{chimp},j} - D_{j,k})/2$ over all leaves j and k.

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

$$(D_{\text{chimp, human}} + D_{\text{chimp, seal}} - D_{\text{human, seal}}) / 2 = (3 + 6 - 7) / 2 = 1$$

$$(D_{\text{chimp, human}} + D_{\text{chimp, whale}} - D_{\text{human, whale}}) / 2 = (3 + 4 - 5) / 2 = 1$$

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(chimp) is equal to the minimum value of $(D_{\text{chimp},k} + D_{\text{chimp},j} - D_{j,k})/2$ over all leaves j and k.

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

$$(D_{\text{chimp, human}} + D_{\text{chimp, seal}} - D_{\text{human, seal}}) / 2 = (3 + 6 - 7) / 2 = 1$$

$$(D_{\text{chimp, human}} + D_{\text{chimp, whale}} - D_{\text{human, whale}}) / 2 = (3 + 4 - 5) / 2 = 1$$

$$(D_{\text{chimp, whale}} + D_{\text{chimp, seal}} - D_{\text{whale, seal}}) / 2 = (6 + 4 - 2) / 2 = 4$$

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(chimp) is equal to the minimum value of $(D_{\text{chimp},k} + D_{\text{chimp},j} - D_{j,k})/2$ over all leaves j and k.

	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0

$$(D_{\text{human, chimp}} + D_{\text{chimp, seal}} - D_{\text{human, seal}}) / 2 = (3 + 6 - 7) / 2 = \mathbf{1}$$

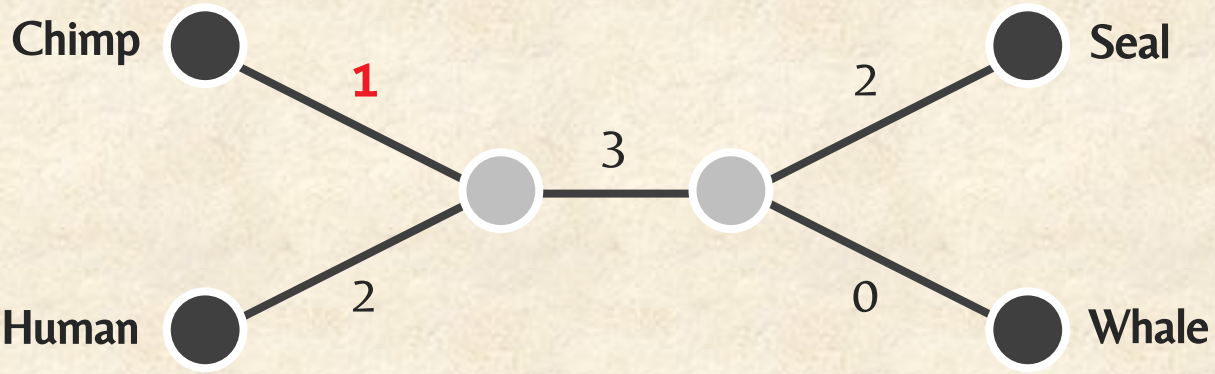
$$(D_{\text{human, chimp}} + D_{\text{chimp, whale}} - D_{\text{human, whale}}) / 2 = (3 + 4 - 5) / 2 = \mathbf{1}$$

$$(D_{\text{whale, chimp}} + D_{\text{chimp, seal}} - D_{\text{whale, seal}}) / 2 = (6 + 4 - 2) / 2 = 4$$

Computing Limb Lengths

Limb Length Theorem: *LimbLength*(chimp) is equal to the minimum value of $(D_{\text{chimp},k} + D_{\text{chimp},j} - D_{j,k})/2$ over all leaves j and k.

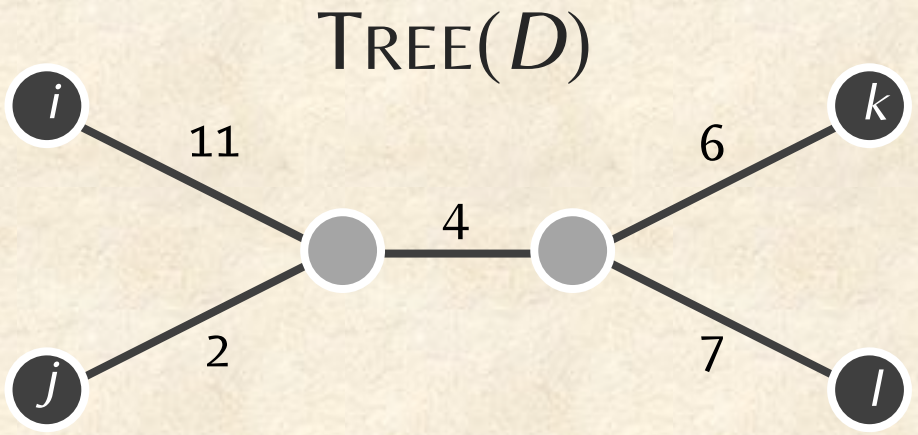
	Chimp	Human	Seal	Whale
Chimp	0	3	6	4
Human	3	0	7	5
Seal	6	7	0	2
Whale	4	5	2	0



AdditivePhylogeny In Action

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0



AdditivePhylogeny In Action

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

1. Pick an arbitrary leaf *j*.

AdditivePhylogeny In Action

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

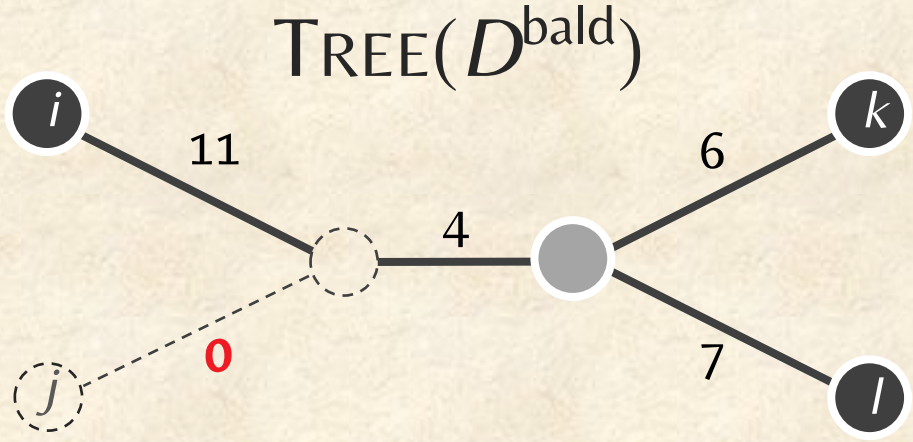
LimbLength(*j*) = 2

2. Compute its limb length, *LimbLength*(*j*).

AdditivePhylogeny In Action

D^{bald}

	i	*j*	*k*	*l*
i	0	11	21	22
j	11	0	10	11
k	21	10	0	13
l	22	11	13	0



3. Subtract *LimbLength*(*j*) from each row and column to produce D^{bald} in which *j* is a **bald limb** (length 0).

AdditivePhylogeny In Action

D^{trim}

	i	*j*	*k*	*l*
i	0	11	21	22
j	11	0	10	11
k	21	10	0	13
l	22	11	13	0

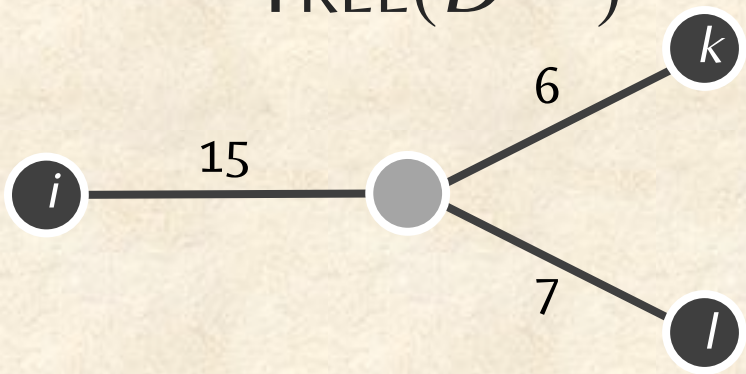
4. Remove the *j*-th row and column of the matrix to form the $(n-1)$ x $(n-1)$ matrix D^{trim}.

AdditivePhylogeny In Action

D^{trim}

	i	*j*	*k*	*l*
i	0	11	21	22
j	11	0	10	11
k	21	10	0	13
l	22	11	13	0

TREE(D^{trim})

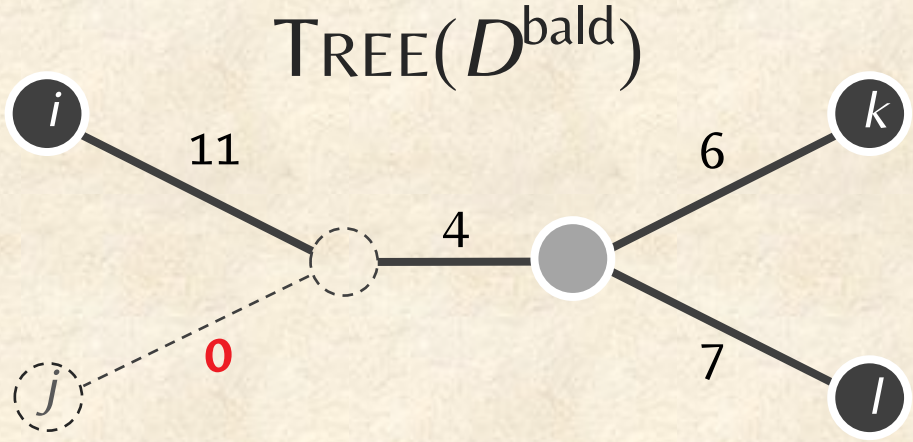


5. Construct *Tree*(D^{trim}).

AdditivePhylogeny In Action

D^{bald}

	i	*j*	*k*	*l*
i	0	11	21	22
j	11	0	10	11
k	21	10	0	13
l	22	11	13	0



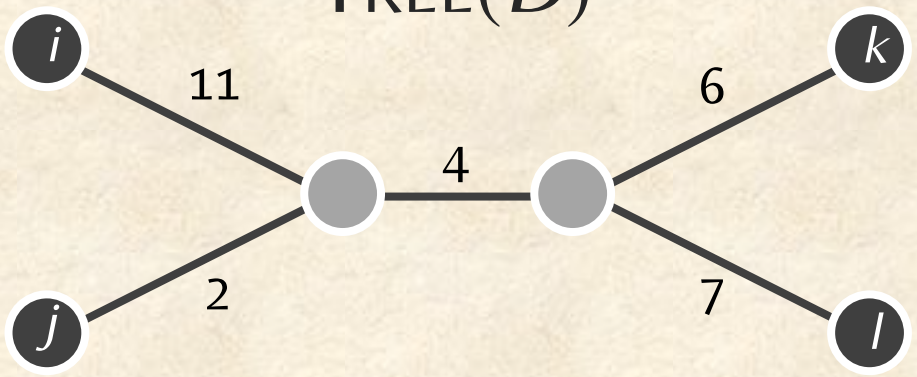
6. Identify the point in *Tree*(D^{trim}) where leaf *j* should be attached.

AdditivePhylogeny In Action

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

TREE(D)



LimbLength(j) = 2

7. Attach j by an edge of length *LimbLength*(j) in order to form *Tree*(D).

AdditivePhylogeny

AdditivePhylogeny(*D*):

1. Pick an arbitrary leaf *j*.
2. Compute its limb length, *LimbLength*(*j*).
3. Subtract *LimbLength*(*j*) from each row and column to produce D^{bald} in which *j* is a bald limb (length 0).
4. Remove the *j*-th row and column of the matrix to form the $(n - 1)$ x $(n - 1)$ matrix D^{trim}.
5. Construct *Tree*(D^{trim}).
6. Identify the point in *Tree*(D^{trim}) where leaf *j* should be attached.
7. Attach *j* by an edge of length *LimbLength*(*j*) in order to form *Tree*(*D*).

AdditivePhylogeny

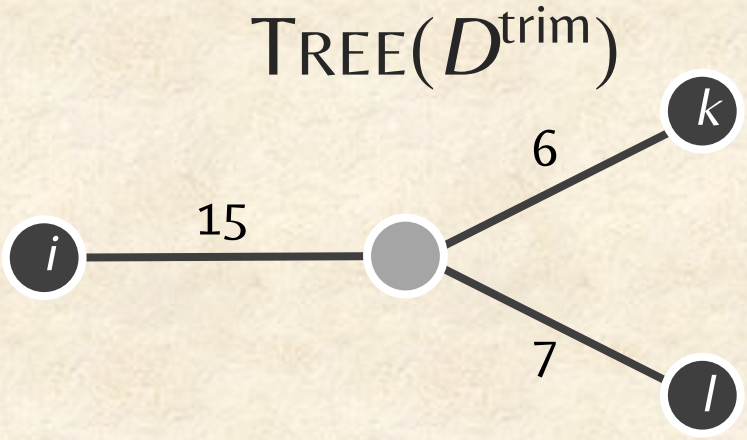
AdditivePhylogeny(D):

1. Pick an arbitrary leaf j.
2. Compute its limb length, $LimbLength(j)$.
3. Subtract $LimbLength(j)$ from each row and column to produce D^{bald} in which j is a bald limb (length 0).
4. Remove the j-th row and column of the matrix to form the $(n-1)$ x $(n-1)$ matrix D^{trim}.
5. Construct $Tree(D^{trim})$.
6. **Identify the point in $Tree(D^{trim})$ where leaf j should be attached.**
7. Attach j by an edge of length $LimbLength(j)$ in order to form $Tree(D)$.

Attaching a Limb

D^{bald}

	i	*j*	*k*	*l*
i	0	11	21	22
j	11	0	10	11
k	21	10	0	13
l	22	11	13	0

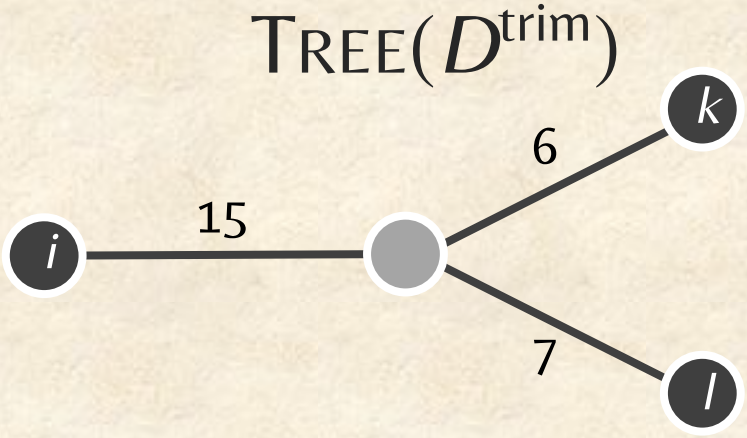


Limb Length Theorem: the length of the limb of j is equal to the minimum value of $(D^{bald}_{i,j} + D^{bald}_{j,k} - D^{bald}_{i,k})/2$ over all leaves i and k.

Attaching a Limb

D^{bald}

	i	*j*	*k*	*l*
i	0	**11**	**21**	22
j	11	0	**10**	11
k	21	10	0	13
l	22	11	13	0



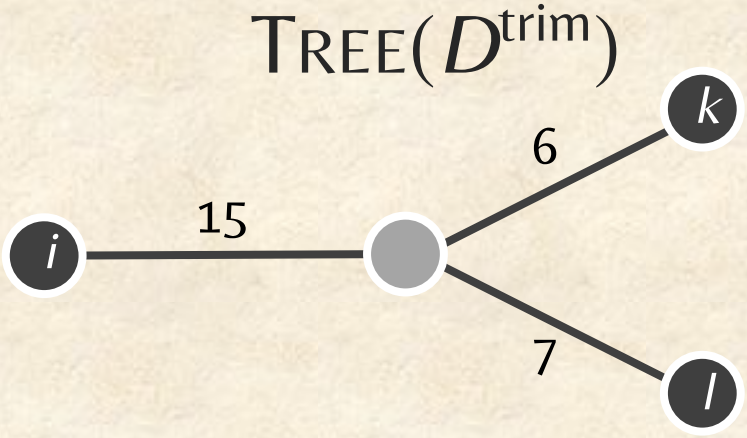
Limb Length Theorem: the length of the limb of *j* is equal to the minimum value of $(D^{bald}_{i,j} + D^{bald}_{j,k} - D^{bald}_{i,k})/2$ over all leaves *i* and *k*.

$$(D^{bald}_{i,j} + D^{bald}_{j,k} - D^{bald}_{i,k})/2 = \mathbf{0}$$

Attaching a Limb

D^{bald}

	i	*j*	*k*	*l*
i	0	**11**	**21**	22
j	11	0	**10**	11
k	21	10	0	13
l	22	11	13	0



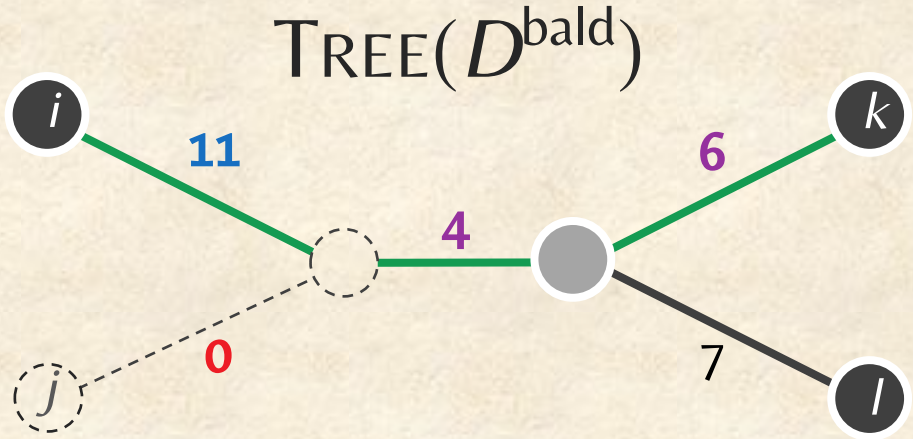
$$(D^{bald}_{i,j} + D^{bald}_{j,k} - D^{bald}_{i,k})/2 = \mathbf{0}$$

$$D^{bald}_{i,j} + D^{bald}_{j,k} = D^{bald}_{i,k}$$

Attaching a Limb

D^{bald}

	i	*j*	*k*	*l*
i	0	**11**	**21**	22
j	11	0	**10**	11
k	21	10	0	13
l	22	11	13	0



The attachment point for *j* is found on the path between leaves *i* and *k* at distance $D^{bald}_{i,j}$ from *i*.

$$D^{bald}_{i,j} + D^{bald}_{j,k} = D^{bald}_{i,k}$$

AdditivePhylogeny

AdditivePhylogeny(*D*):

1. Pick an arbitrary leaf *j*.
2. Compute its limb length, *LimbLength*(*j*).
3. Subtract *LimbLength*(*j*) from each row and column to produce D^{bald} in which *j* is a bald limb (length 0).
4. Remove the *j*-th row and column of the matrix to form the $(n - 1)$ x $(n - 1)$ matrix D^{trim}.
5. Construct *Tree*(D^{trim}).
6. Identify the point in *Tree*(D^{trim}) where leaf *j* should be attached.
7. Attach *j* by an edge of length *LimbLength*(*j*) in order to form *Tree*(*D*).

Code Challenge: Implement **AdditivePhylogeny**.

Spike Protein Distance Matrix

	Cow	Pig	Horse		Mouse		Dog	Cat	Turkey	Civet	Human
Cow	0	226	249	436	958	916	730	787	785		
Pig	226	0	292	436	903	905	744	802	813		
Horse	249	292	0	426	927	907	735	795	791		
Mouse	436	436	426	0	917	946	725	767	782		
Dog	958	903	927	917	0	706	730	844	846		
Cat	916	905	907	946	706	0	736	840	836		
Turkey	730	744	735	725	730	736	0	763	760		
Civet	787	802	795	767	844	840	763	0	16		
Human	785	813	791	782	846	836	760	16	0		

This matrix isn't additive! Let's fudge it a little...

Spike Protein Distance Matrix

	Cow	Pig	Horse	Mouse	Dog	Cat	Turkey	Civet	Human
Cow	0	295	306	497	1081	1091	1003	956	954
Pig	295	0	309	500	1084	1094	1006	959	957
Horse	306	309	0	489	1073	1083	995	948	946
Mouse	497	500	489	0	1092	1102	1014	967	965
Dog	1081	1084	1073	1092	0	818	1056	1053	1051
Cat	1091	1094	1083	1102	818	0	1066	1063	1061
Turkey	1003	1006	995	1014	1056	1066	0	975	973
Civet	956	959	948	967	1053	1063	975	0	16
Human	954	957	946	965	1051	1061	973	16	0

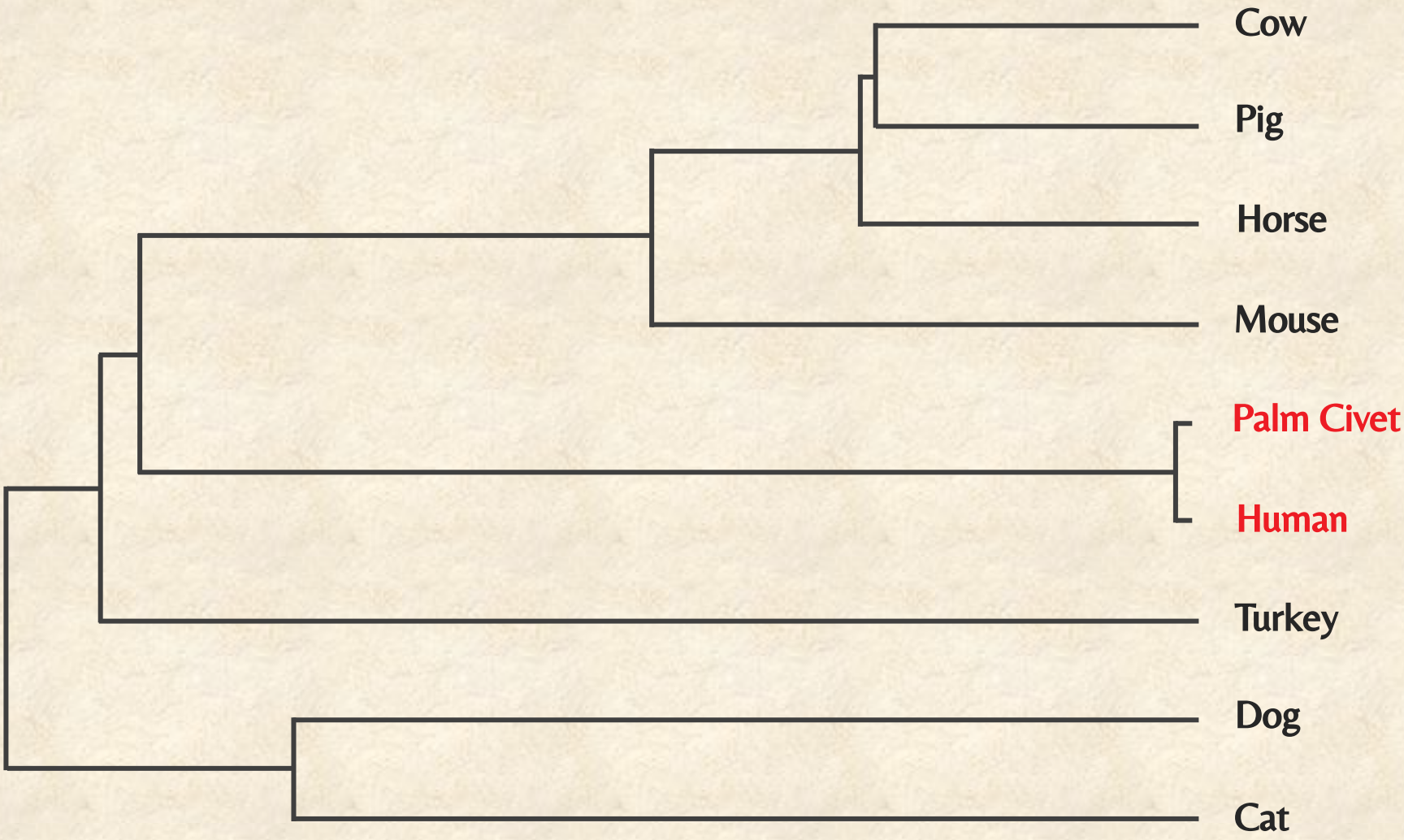
STOP and Think: Which animal gave us SARS?

Spike Protein Distance Matrix

	Cow	Pig	Horse	Mouse	Dog	Cat	Turkey	Civet	Human
Cow	0	295	306	497	1081	1091	1003	956	954
Pig	295	0	309	500	1084	1094	1006	959	957
Horse	306	309	0	489	1073	1083	995	948	946
Mouse	497	500	489	0	1092	1102	1014	967	965
Dog	1081	1084	1073	1092	0	818	1056	1053	1051
Cat	1091	1094	1083	1102	818	0	1066	1063	1061
Turkey	1003	1006	995	1014	1056	1066	0	975	973
Civet	956	959	948	967	1053	1063	975	0	**16**
Human	954	957	946	965	1051	1061	973	16	0

STOP and Think: Which animal gave us SARS?

Coronavirus Phylogeny



Spike Protein Distance Matrix

	Cow	Pig	Horse	Mouse	Dog	Cat	Turkey	Civet	Human
Cow	0	226	249	436	958	916	730	787	785
Pig	226	0	292	436	903	905	744	802	813
Horse	249	292	0	426	927	907	735	795	791
Mouse	436	436	426	0	917	946	725	767	782
Dog	958	903	927	917	0	706	730	844	846
Cat	916	905	907	946	706	0	736	840	836
Turkey	730	744	735	725	730	736	0	763	760
Civet	787	802	795	767	844	840	763	0	16
Human	785	813	791	782	846	836	760	16	0

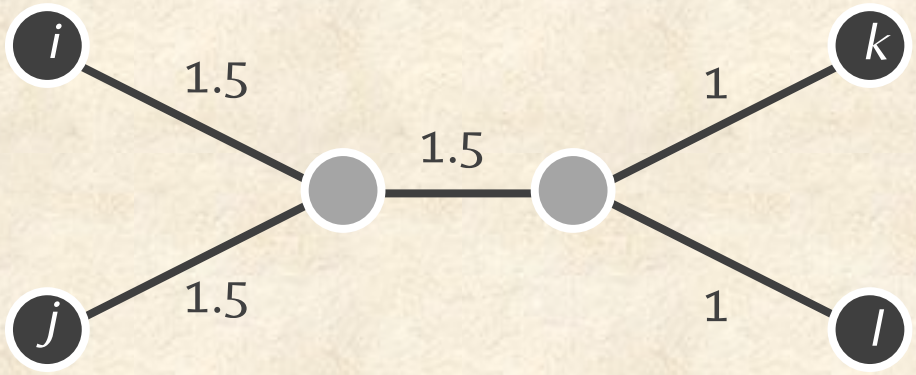
But what should we do about *non-additive* matrices?

Outline

- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- **Using Least-Squares to Construct Distance-Based Phylogenies**
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

Sum of Squared Errors

T

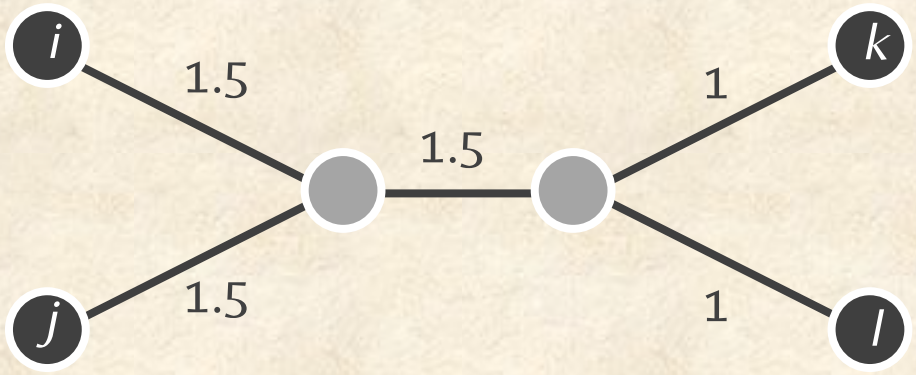


D

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

Sum of Squared Errors

T



D

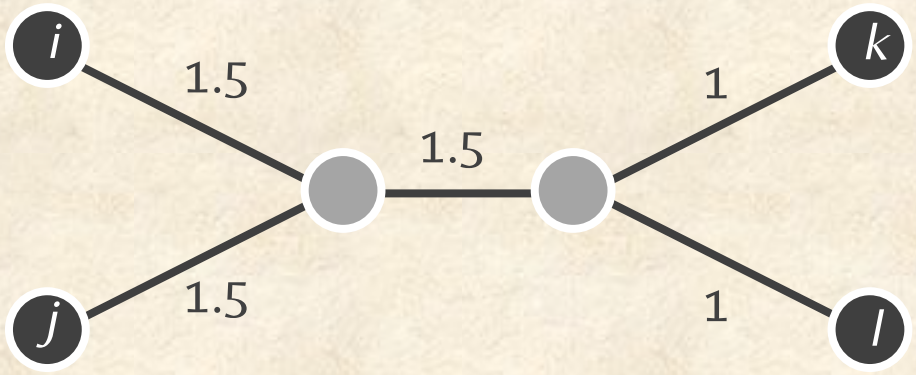
	i	j	k	l
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

d

	i	j	k	l
i	0	3	4	4
j	3	0	4	4
k	4	4	0	2
l	4	4	2	0

Sum of Squared Errors

T



D

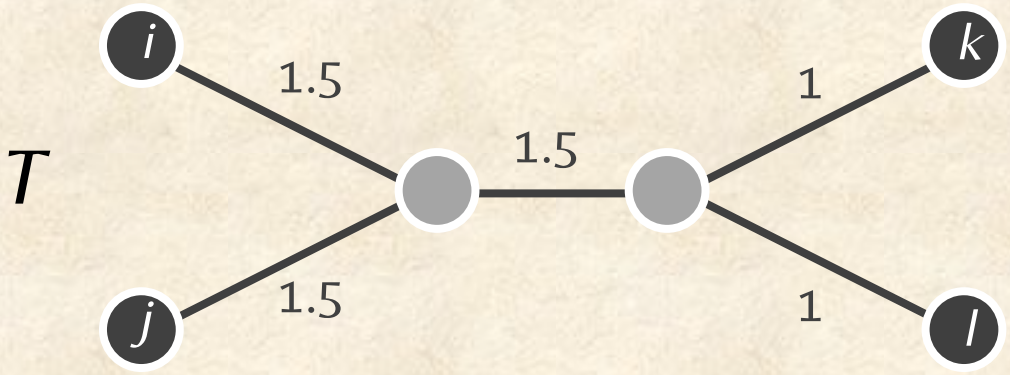
	i	j	k	l
i	0	3	4	**3**
j	3	0	4	**5**
k	4	4	0	2
l	3	5	2	0

d

	i	j	k	l
i	0	3	4	**4**
j	3	0	4	**4**
k	4	4	0	2
l	4	4	2	0

Sum of Squared Errors

$$Discrepancy(T, D) = \Sigma_{1 \le i < j \le n} (d_{i,j}(T) - D_{i,j})^2$$



D

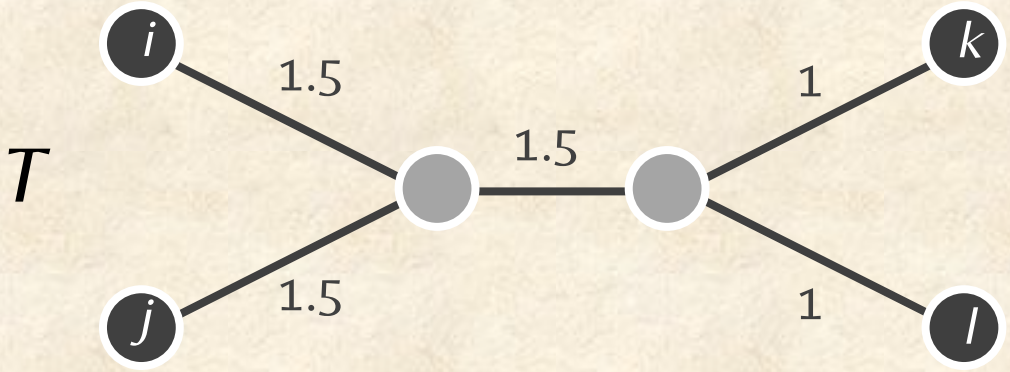
	i	j	k	l
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

d

	i	j	k	l
i	0	3	4	4
j	3	0	4	4
k	4	4	0	2
l	4	4	2	0

Sum of Squared Errors

$$Discrepancy(T, D) = \sum_{1 \le i < j \le n} (d_{i,j}(T) - D_{i,j})^2$$
$$= 1^2 + 1^2 = 2$$



D

	i	j	k	l
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

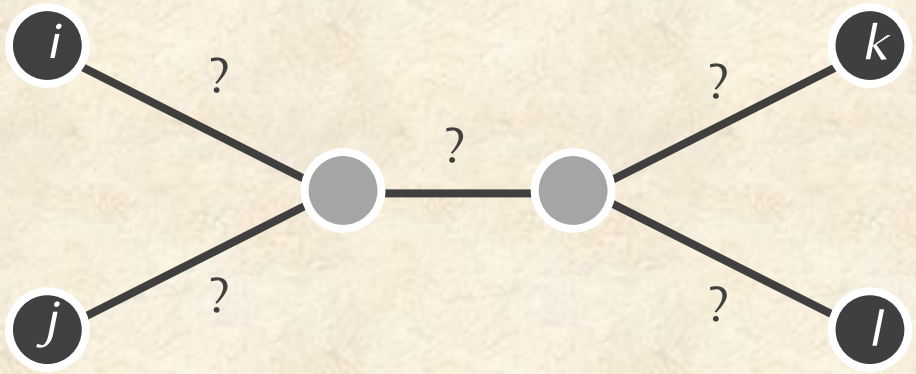
d

	i	j	k	l
i	0	3	4	4
j	3	0	4	4
k	4	4	0	2
l	4	4	2	0

Sum of Squared Errors

Exercise Break: Assign lengths to edges in T in order to minimize *Discrepancy*(T, D).

T



D

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

d

	i	*j*	*k*	*l*
i	0	?	?	?
j	?	0	?	?
k	?	?	0	?
l	?	?	?	0

Least-Squares Phylogeny

Least-Squares Distance-Based Phylogeny Problem:
Given a distance matrix, find the tree that minimizes the sum of squared errors.

- **Input:** An n x n distance matrix D.
- **Output:** A weighted tree T with n leaves minimizing *Discrepancy*(T, D) over all weighted trees with n leaves.

Least-Squares Phylogeny

Least-Squares Distance-Based Phylogeny Problem:
Given a distance matrix, find the tree that minimizes the sum of squared errors.

- **Input:** An n x n distance matrix D.
- **Output:** A weighted tree T with n leaves minimizing *Discrepancy*(T, D) over all weighted trees with n leaves.

Unfortunately, this problem is *NP*-Complete…

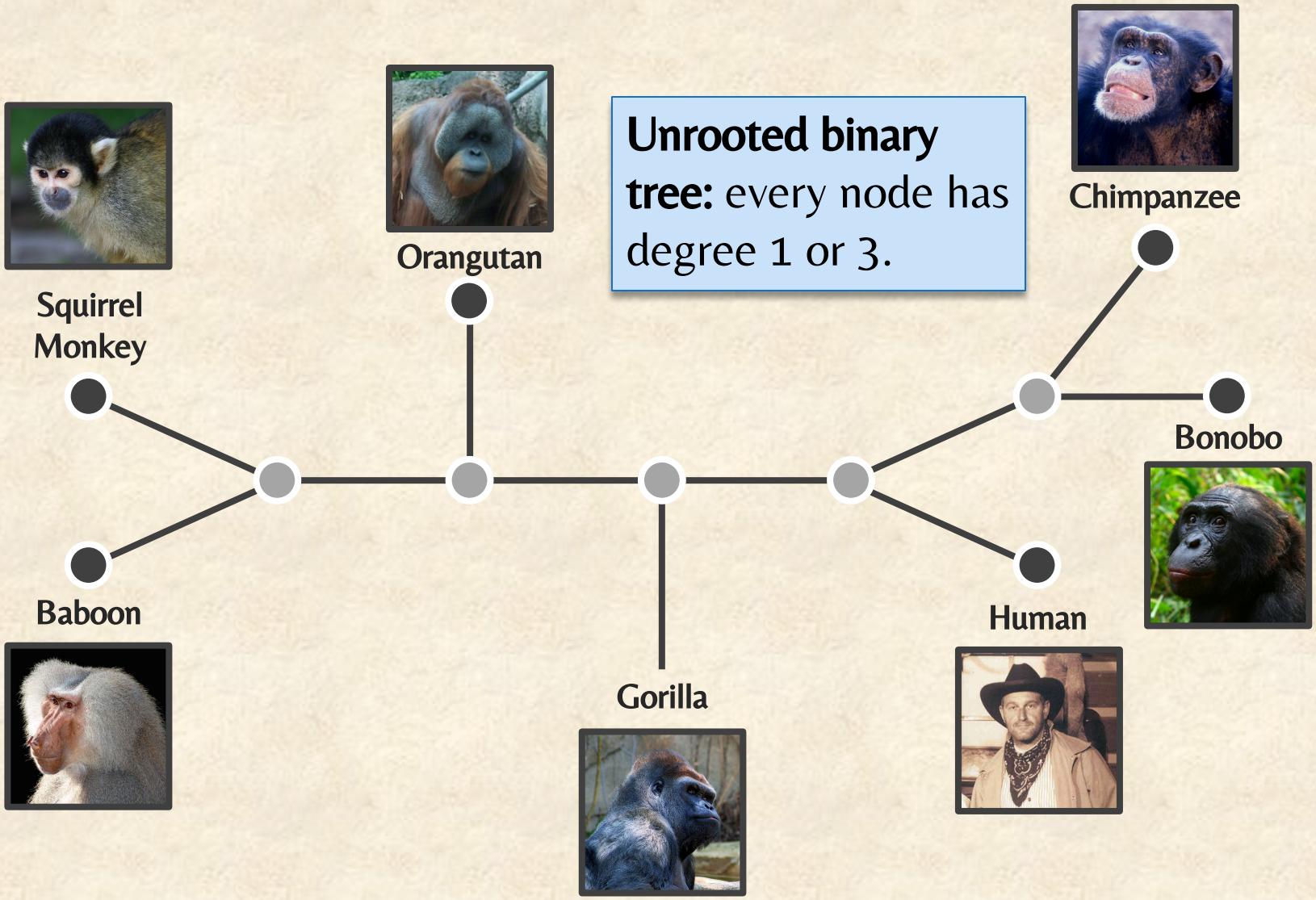
Outline

- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- **Ultrametric Evolutionary Trees**
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

Modeling Speciations

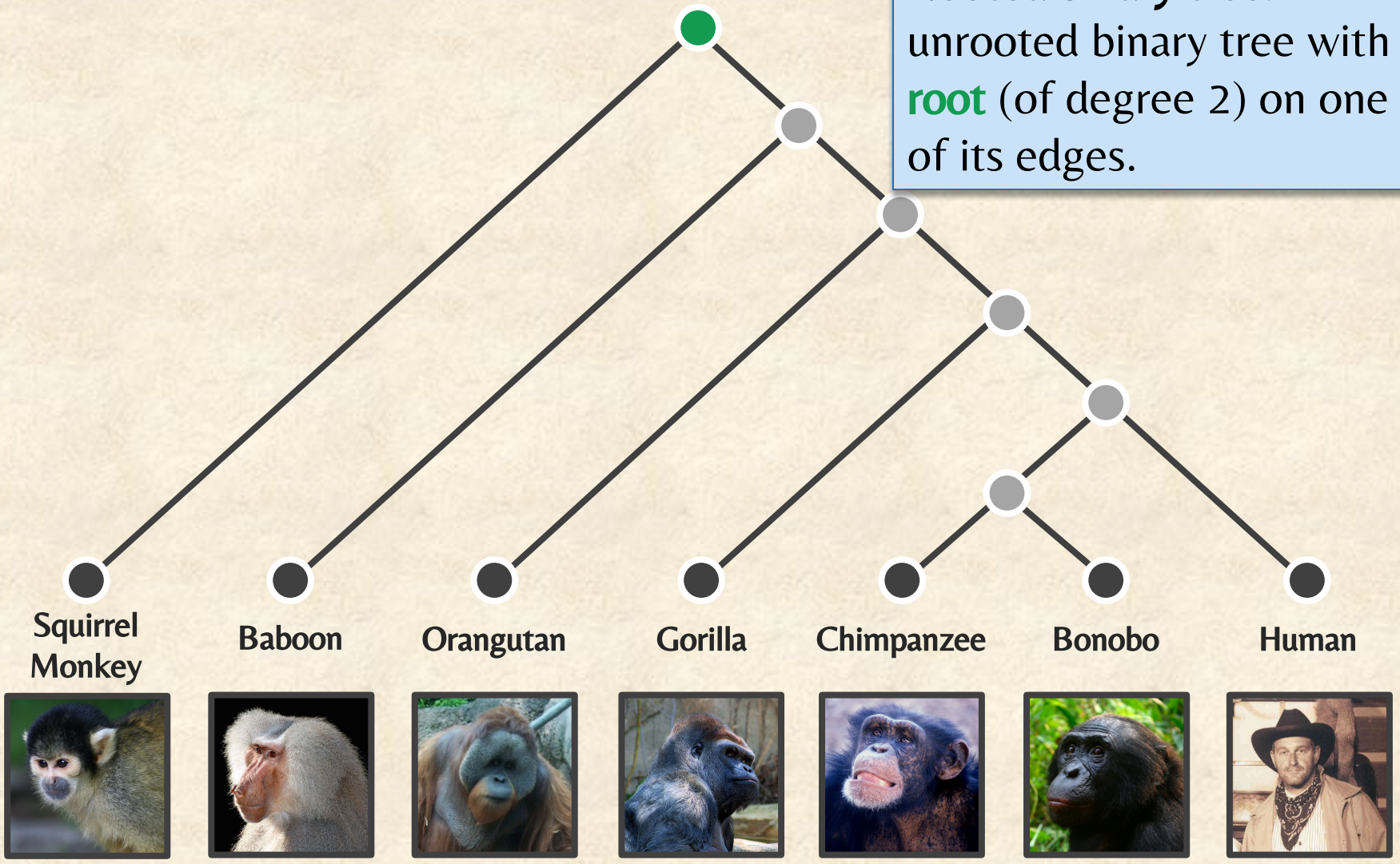
Researchers often assume that all internal nodes correspond to **speciations**, where one species splits into two.

Modeling Speciations



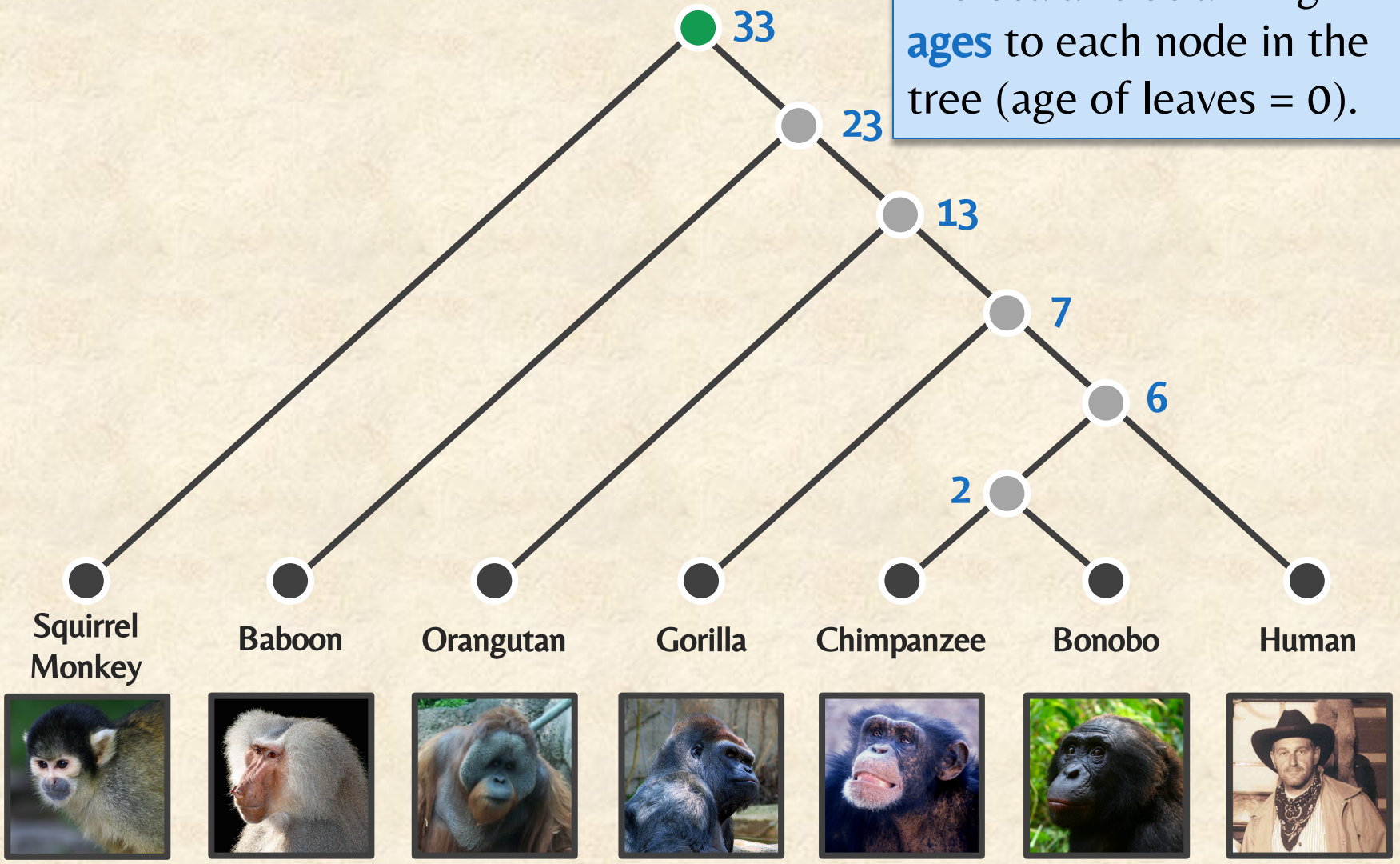
Modeling Speciations

Rooted binary tree: an unrooted binary tree with a **root** (of degree 2) on one of its edges.



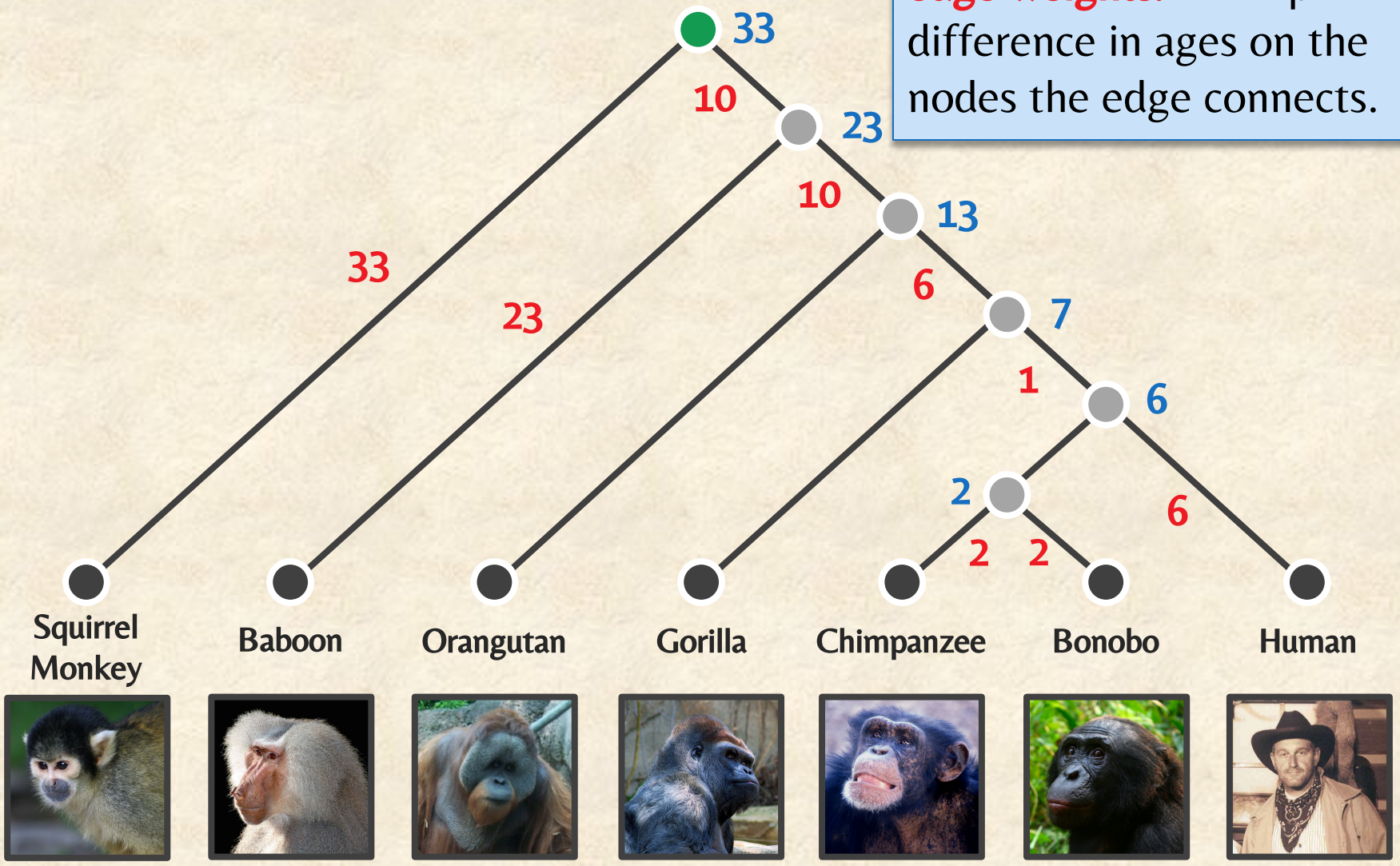
Ultrametric Trees

Molecular clock: assigns **ages** to each node in the tree (age of leaves = 0).



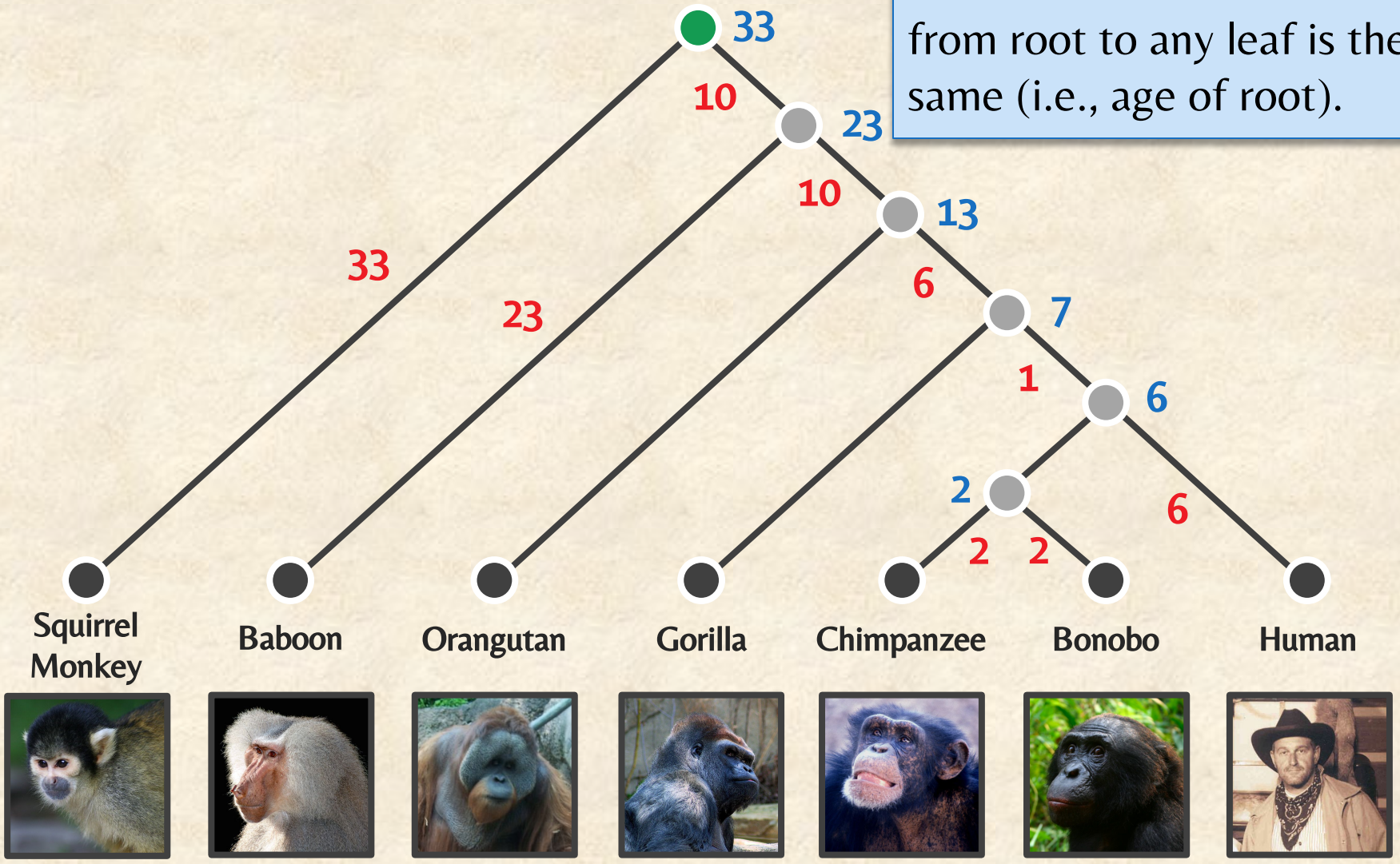
Ultrametric Trees

edge weights: correspond to difference in ages on the nodes the edge connects.



Ultrametric Trees

Ultrametric tree: distance from root to any leaf is the same (i.e., age of root).



UPGMA: A Clustering Heuristic

1. Form a cluster for each present-day species, each containing a single leaf.

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

UPGMA: A Clustering Heuristic

2. Find the two closest clusters C_1 and C_2 according to the average distance

$$D_{\text{avg}}(C_1, C_2) = \Sigma_{i \text{ in } C1,\, j \text{ in } C2}\, D_{i,j} / |C_1| \bullet |C_2|$$

where $|C|$ denotes the number of elements in C.

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	**2**
l	3	5	**2**	0

UPGMA: A Clustering Heuristic

3. Merge C_1 and C_2 into a single cluster C.

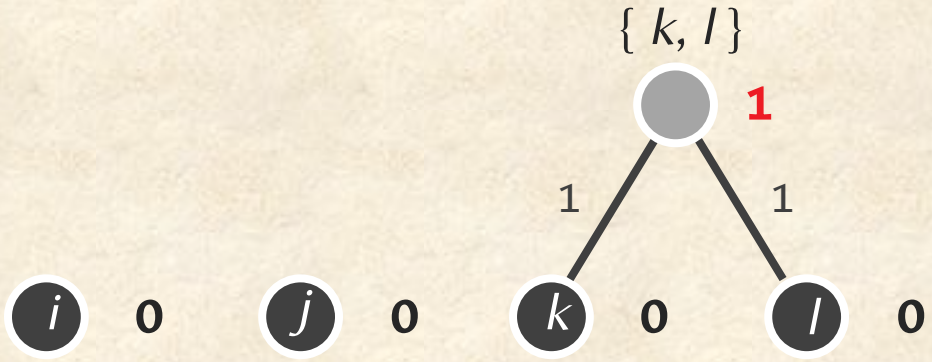
	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	**2**
l	3	5	**2**	0

{ *k*, *l* }

UPGMA: A Clustering Heuristic

4. Form a new node for C and connect to C_1 and C_2 by an edge. Set age of C as $D_{avg}(C_1, C_2)/2$.

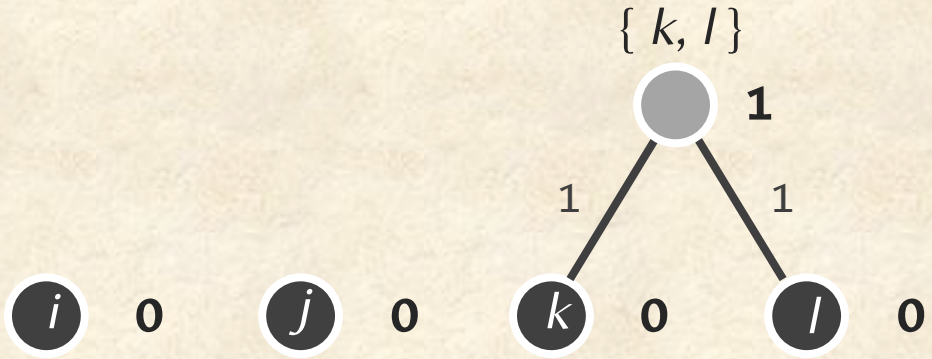
	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	**2**
l	3	5	**2**	0



UPGMA: A Clustering Heuristic

5. Update the distance matrix by computing the average distance between each pair of clusters.

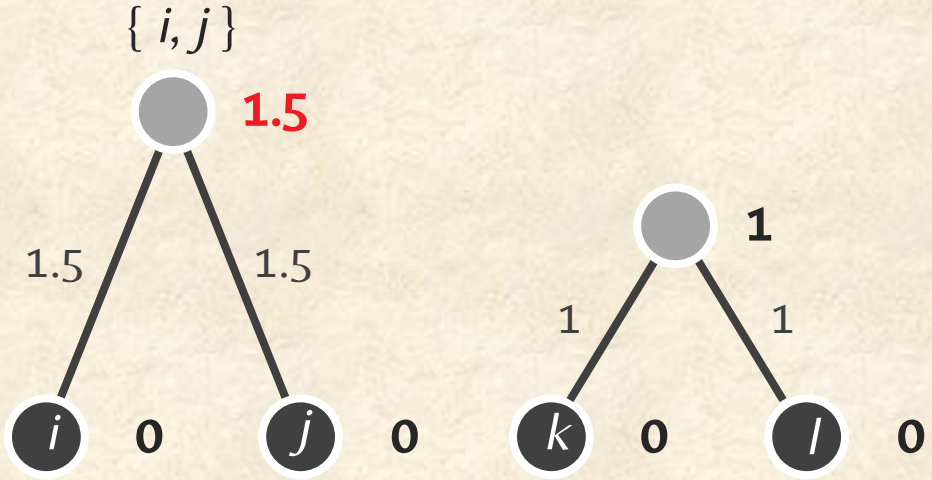
	i	*j*	{ *k*, *l* }
i	0	3	3.5
j	3	0	4.5
{ *k*, *l* }	3.5	4.5	0



UPGMA: A Clustering Heuristic

6. Iterate until a single cluster contains all species.

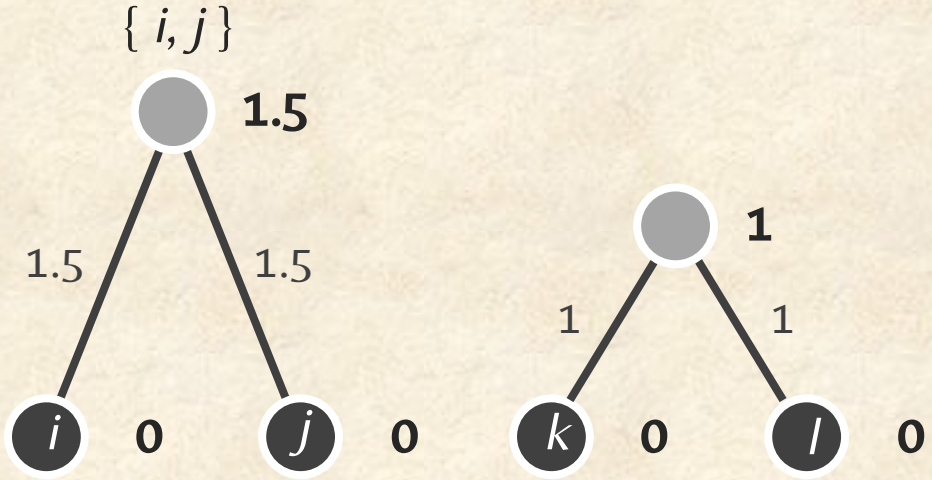
	i	*j*	{ *k*, *l* }
i	0	**3**	3.5
j	**3**	0	4.5
{ *k*, *l* }	3.5	4.5	0



UPGMA: A Clustering Heuristic

6. Iterate until a single cluster contains all species.

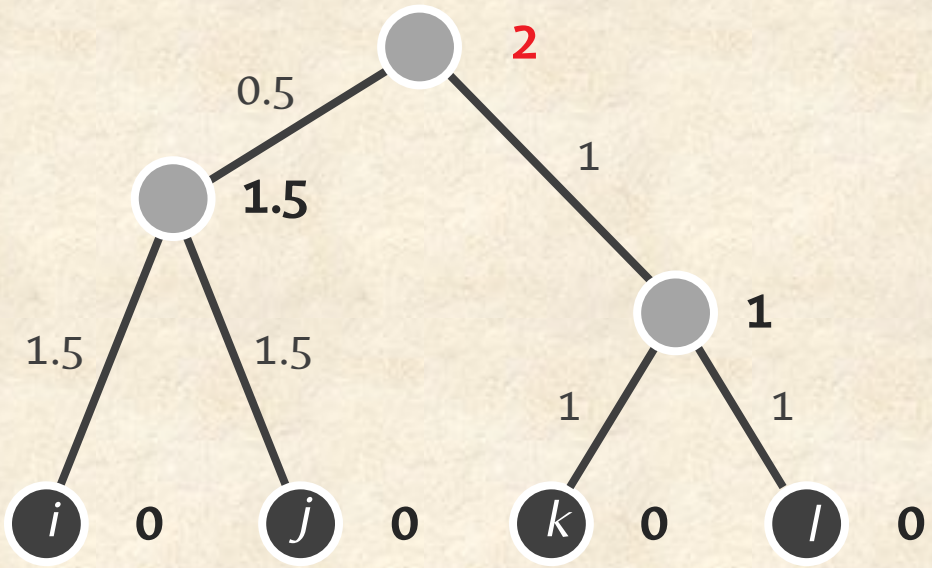
	$\{i, j\}$	$\{k, l\}$
$\{i, j\}$	0	4
$\{k, l\}$	4	0



UPGMA: A Clustering Heuristic

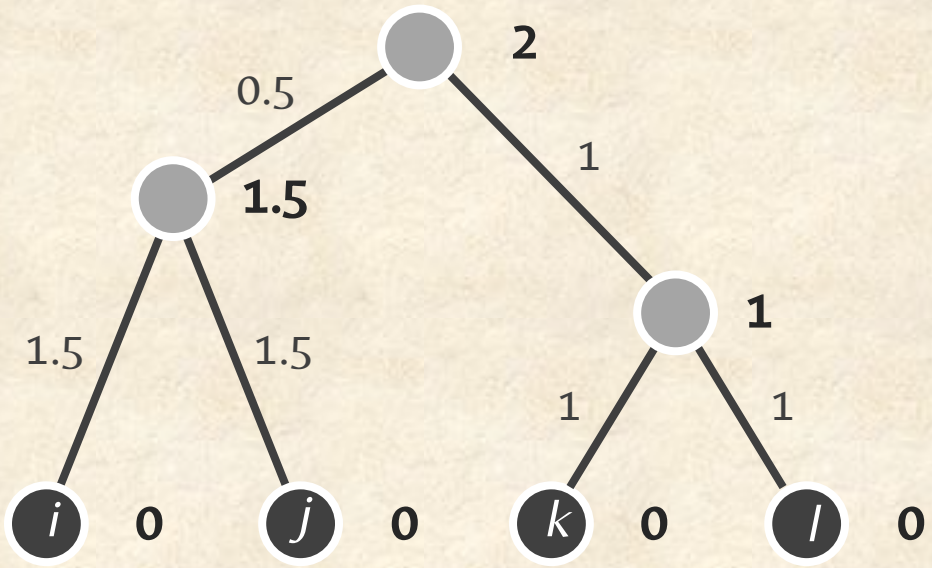
6. Iterate until a single cluster contains all species.

	$\{i, j\}$	$\{k, l\}$
$\{i, j\}$	0	4
$\{k, l\}$	4	0



UPGMA: A Clustering Heuristic

6. Iterate until a single cluster contains all species.



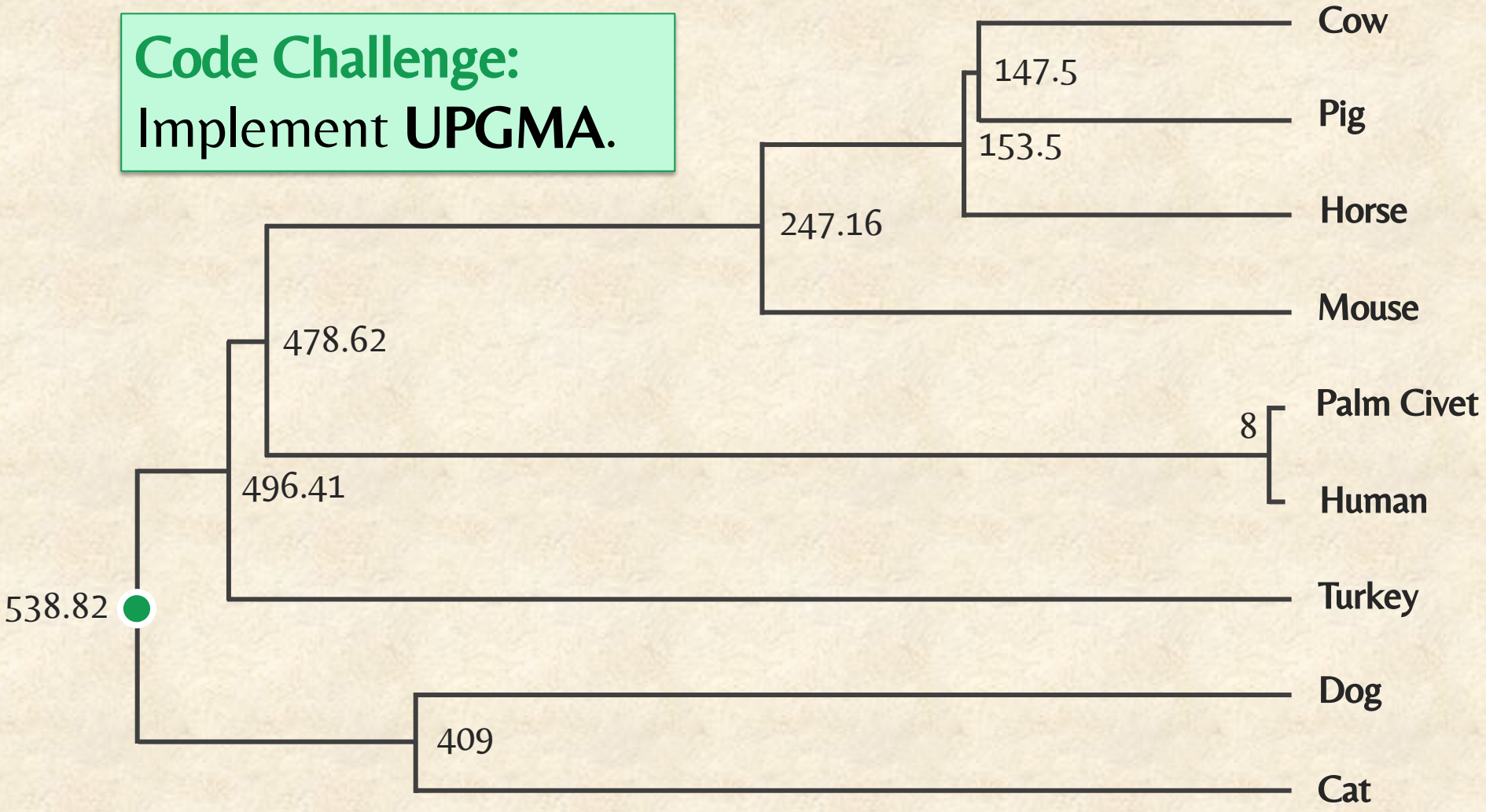
UPGMA: A Clustering Heuristic

UPGMA(*D*):

1. Form a cluster for each present-day species, each containing a single leaf.
2. Find the two closest clusters C_1 and C_2 according to the average distance
   $$D_{avg}(C_1, C_2) = \Sigma_{i \text{ in } C1,\, j \text{ in } C2}\, D_{i,j} / |C_1| \bullet |C_2|$$
   where $|C|$ denotes the number of elements in C
3. Merge C_1 and C_2 into a single cluster C.
4. Form a new node for C and connect to C_1 and C_2 by an edge. Set age of C as $D_{avg}(C_1, C_2)/2$.
5. Update the distance matrix by computing the average distance between each pair of clusters.
6. Iterate steps 2-5 until a single cluster contains all species.

Applying UPGMA to Spike Proteins

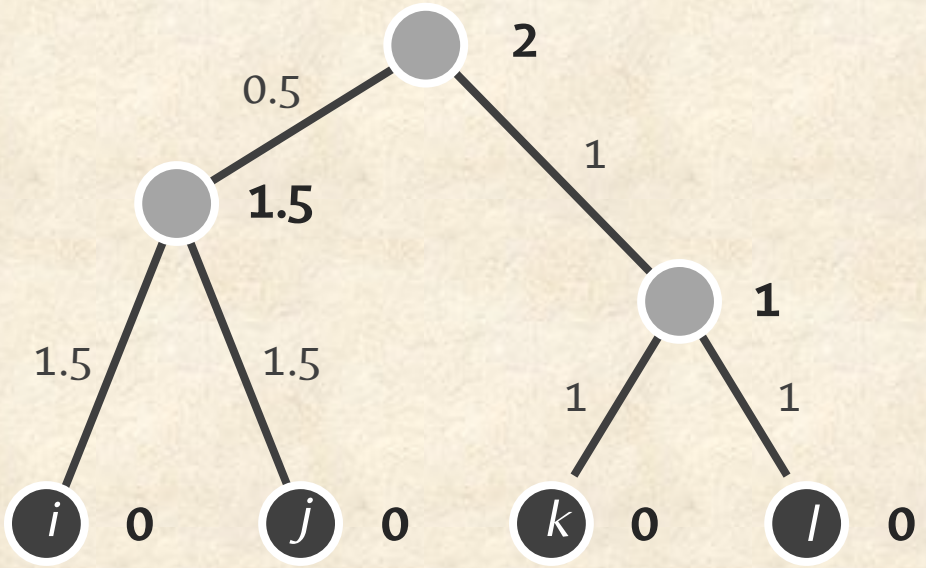
Code Challenge:
Implement **UPGMA**.




You never stated a computational problem!

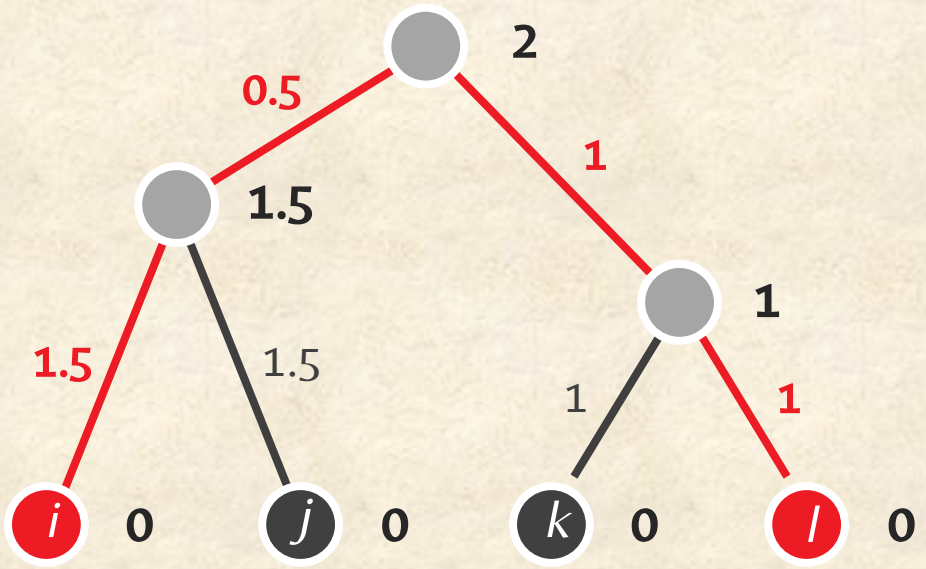
UPGMA Doesn't "Fit" a Tree to a Matrix

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0



UPGMA Doesn't "Fit" a Tree to a Matrix

	i	*j*	*k*	*l*
i	0	3	4	**3**
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0



In Summary…

- AdditivePhylogeny:

In Summary…

- **AdditivePhylogeny:**
  - good: produces the tree fitting an *additive* matrix

In Summary…

- **AdditivePhylogeny:**
  - good: produces the tree fitting an *additive* matrix
  - bad: fails completely on a *non-additive* matrix

In Summary...

- **AdditivePhylogeny:**
  - good: produces the tree fitting an *additive* matrix
  - bad: fails completely on a *non-additive* matrix
- **UPGMA:**
  - good: produces a tree for any matrix

In Summary…

- **AdditivePhylogeny:**
  - good: produces the tree fitting an *additive* matrix
  - bad: fails completely on a *non-additive* matrix
- **UPGMA:**
  - good: produces a tree for any matrix
  - bad: tree doesn't necessarily fit an additive matrix

In Summary…

- **AdditivePhylogeny:**
  - good: produces the tree fitting an *additive* matrix
  - bad: fails completely on a *non-additive* matrix
- **UPGMA:**
  - good: produces a tree for any matrix
  - bad: tree doesn't necessarily fit an additive matrix
- **?????:**
  - good: produces the tree fitting an additive matrix
  - good: provides heuristic for a non-additive matrix

Outline

- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- **The Neighbor-Joining Algorithm**
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

Neighbor-Joining Theorem

Given an n x n distance matrix D, its **neighbor-joining matrix** is the matrix D^* defined as

$$D^*_{i,j} = (n - 2) \cdot D_{i,j} - TotalDistance_D(i) - TotalDistance_D(j)$$

where $TotalDistance_D(i)$ is the sum of distances from i to all other leaves.

Neighbor-Joining Theorem

Given an n x n distance matrix D, its **neighbor-joining matrix** is the matrix D^* defined as

$$D^*_{i,j} = (n-2) \cdot D_{i,j} - TotalDistance_D(i) - TotalDistance_D(j)$$

where $TotalDistance_D(i)$ is the sum of distances from i to all other leaves.

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

Neighbor-Joining Theorem

Given an n x n distance matrix D, its **neighbor-joining matrix** is the matrix D^* defined as

$$D^*_{i,j} = (n-2) \cdot D_{i,j} - TotalDistance_D(i) - TotalDistance_D(j)$$

where $TotalDistance_D(i)$ is the sum of distances from i to all other leaves.

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

$TotalDistance_D$
56
38
46
48

Neighbor-Joining Theorem

Given an n x n distance matrix D, its **neighbor-joining matrix** is the matrix D^* defined as

$$D^*_{i,j} = (n-2) \cdot D_{i,j} - TotalDistance_D(i) - TotalDistance_D(j)$$

where $TotalDistance_D(i)$ is the sum of distances from i to all other leaves.

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

$TotalDistance_D$
56
38
46
48

D^*

	i	*j*	*k*	*l*
i	0	-68	-60	-60
j	-68	0	-60	-60
k	-60	-60	0	-68
l	-60	-60	-68	0

Neighbor-Joining Theorem

Neighbor-Joining Theorem: If D is additive, then the smallest element of D^* corresponds to neighboring leaves in $Tree(D)$.

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

$TotalDistance_D$
56
38
46
48

D^*

	i	*j*	*k*	*l*
i	0	-68	-60	-60
j	-68	0	-60	-60
k	-60	-60	0	-68
l	-60	-60	-68	0

Neighbor-Joining Theorem

Neighbor-Joining Theorem: If D is additive, then the smallest element of D^* corresponds to neighboring leaves in $Tree(D)$.

D

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	**12**	13
k	21	**12**	0	13
l	22	13	13	0

$TotalDistance_D$
56
38
46
48

D^*

	i	*j*	*k*	*l*
i	0	**-68**	-60	-60
j	**-68**	0	-60	-60
k	-60	-60	0	**-68**
l	-60	-60	**-68**	0

Neighbor-Joining in Action

D^*

	i	*j*	*k*	*l*
i	0	-68	-60	-60
j	-68	0	-60	-60
k	-60	-60	0	-68
l	-60	-60	-68	0

$TotalDistance_D$
56
38
46
48

1. Construct neighbor-joining matrix D^* from D.

Neighbor-Joining in Action

D^*

	i	*j*	*k*	*l*
i	0	**-68**	-60	-60
j	**-68**	0	-60	-60
k	-60	-60	0	**-68**
l	-60	-60	**-68**	0

$TotalDistance_D$
56
38
46
48

2. Find a minimum element $D^*_{i,j}$ of D^*.

Neighbor-Joining in Action

D^*

	i	*j*	*k*	*l*
i	0	**-68**	-60	-60
j	**-68**	0	-60	-60
k	-60	-60	0	-68
l	-60	-60	-68	0

$TotalDistance_D$
56
38
46
48

2. Find a minimum element $D^*_{i,j}$ of D^*.

Neighbor-Joining in Action

D^*

	i	*j*	*k*	*l*
i	0	-68	-60	-60
j	-68	0	-60	-60
k	-60	-60	0	-68
l	-60	-60	-68	0

$TotalDistance_D$
56
38
46
48

$\Delta_{i,j} = (56 - 38) / (4 - 2)$
$= 9$

3. Compute $\Delta_{i,j} = (TotalDistance_D(i) - TotalDistance_D(j)) / (n - 2)$.

Neighbor-Joining in Action

D

	i	*j*	*k*	*l*	$TotalDistance_D$
i	0	13	21	22	56
j	13	0	12	13	38
k	21	12	0	13	46
l	22	13	13	0	48

$\Delta_{i,j} = (56 - 38) / (4 - 2) = 9$

$LimbLength(i) = \frac{1}{2}(13 + 9) = 11$

$LimbLength(i) = \frac{1}{2}(13 - 9) = 2$

4. Set $LimbLength(i)$ equal to $\frac{1}{2}(D_{i,j} + \Delta_{i,j})$ and $LimbLength(j)$ equal to $\frac{1}{2}(D_{i,j} - \Delta_{i,j})$.

Neighbor-Joining in Action

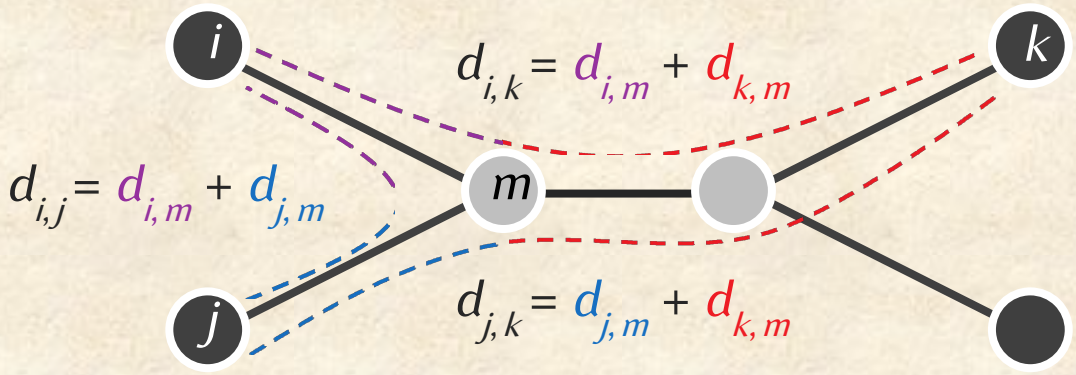
D'

	m	*k*	*l*
m	0	10	11
k	10	0	13
l	11	13	0

$TotalDistance_D$
21
23
24

5. Form a matrix D' by removing i-th and j-th row/column from D and adding an m-th row/column such that for any k, $D_{k,m} = (D_{i,k} + D_{j,k} - D_{i,j}) / 2$.

Flashback: Computation of $d_{k,m}$



$$d_{k,m} = [(d_{i,m} + d_{k,m}) + (d_{j,m} + d_{k,m}) - (d_{i,m} + d_{j,m})] / 2$$

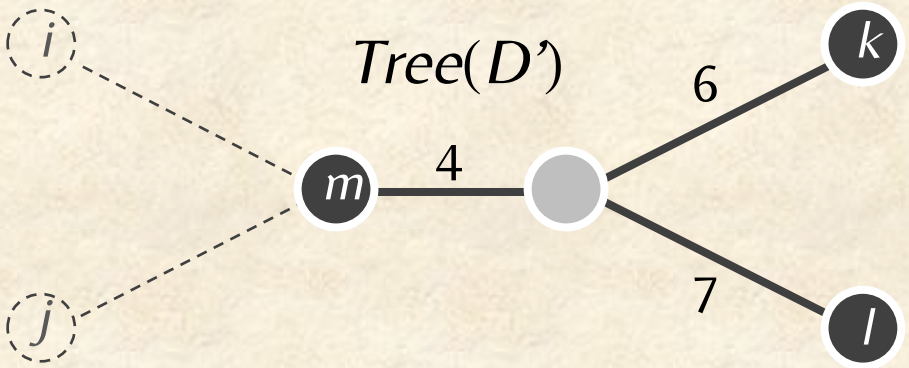
$$d_{k,m} = (d_{i,k} + d_{j,k} - d_{i,j}) / 2$$

$$d_{k,m} = (D_{i,k} + D_{j,k} - D_{i,j}) / 2$$

Neighbor-Joining in Action

D'

	m	*k*	*l*
m	0	10	11
k	10	0	13
l	11	13	0

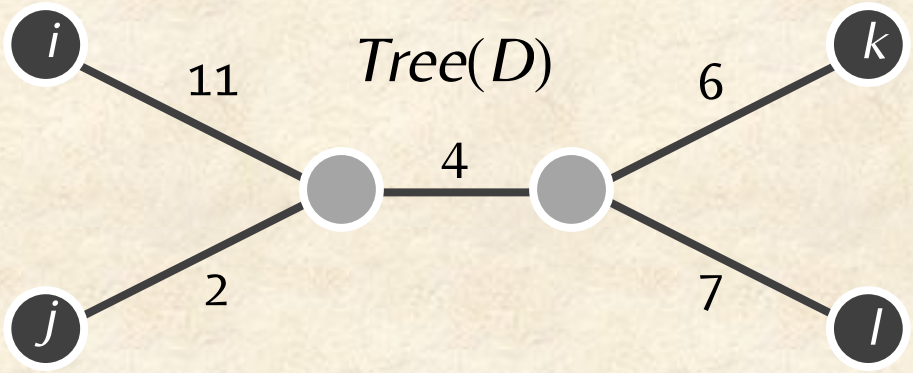


6. Apply **NeighborJoining** to *D'* to obtain *Tree*(*D'*).

Neighbor-Joining in Action

D'

	m	*k*	*l*
m	0	10	11
k	10	0	13
l	11	13	0



$LimbLength(i) = \frac{1}{2}(13 + 9) = 11$

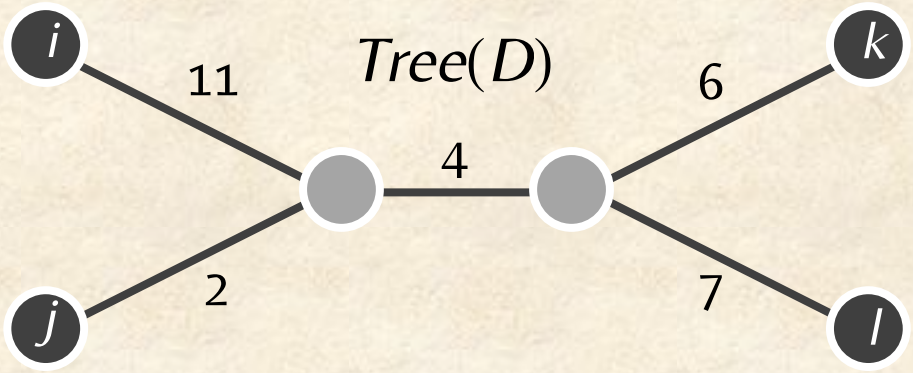
$LimbLength(i) = \frac{1}{2}(13 - 9) = 2$

7. Reattach limbs of *i* and *j* to obtain *Tree*(*D*).

Neighbor-Joining in Action

D'

	m	*k*	*l*
m	0	10	11
k	10	0	13
l	11	13	0



7. Reattach limbs of *i* and *j* to obtain *Tree*(*D*).

Neighbor-Joining

NeighborJoining(*D*):

1. Construct neighbor-joining matrix D^* from D.
2. Find a minimum element $D^*_{i,j}$ of D^*.
3. Compute $\Delta_{i,j} = (TotalDistance_D(i) - TotalDistance_D(j)) / (n - 2)$.
4. Set $LimbLength(i)$ equal to $\frac{1}{2}(D_{i,j} + \Delta_{i,j})$ and $LimbLength(j)$ equal to $\frac{1}{2}(D_{i,j} - \Delta_{i,j})$.
5. Form a matrix D' by removing i-th and j-th row/column from D and adding an m-th row/column such that for any k, $D_{k,m} = (D_{k,i} + D_{k,j} - D_{i,j}) / 2$.
6. Apply **NeighborJoining** to D' to obtain $Tree(D')$.
7. Reattach limbs of i and j to obtain $Tree(D)$.

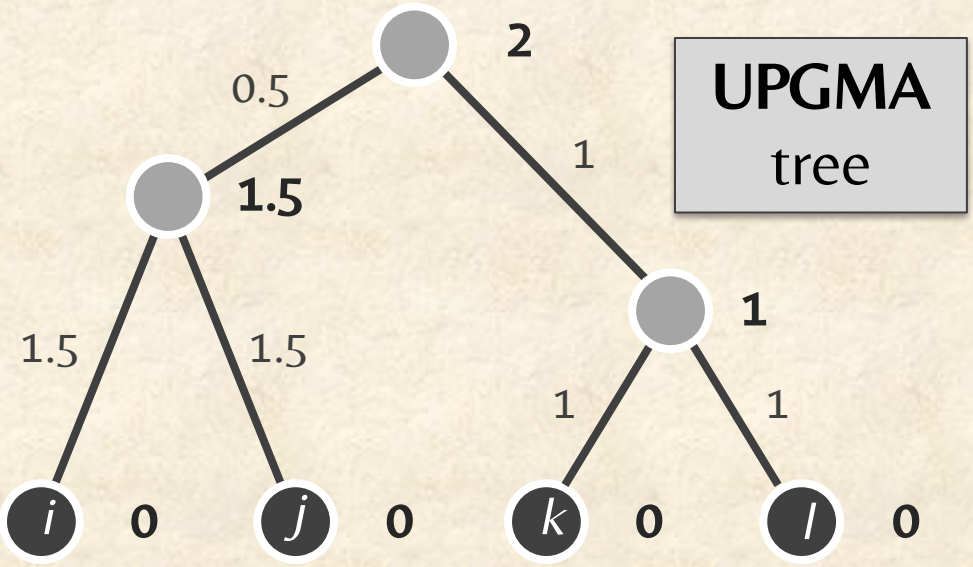
Code Challenge: Implement **NeighborJoining**.

Neighbor-Joining

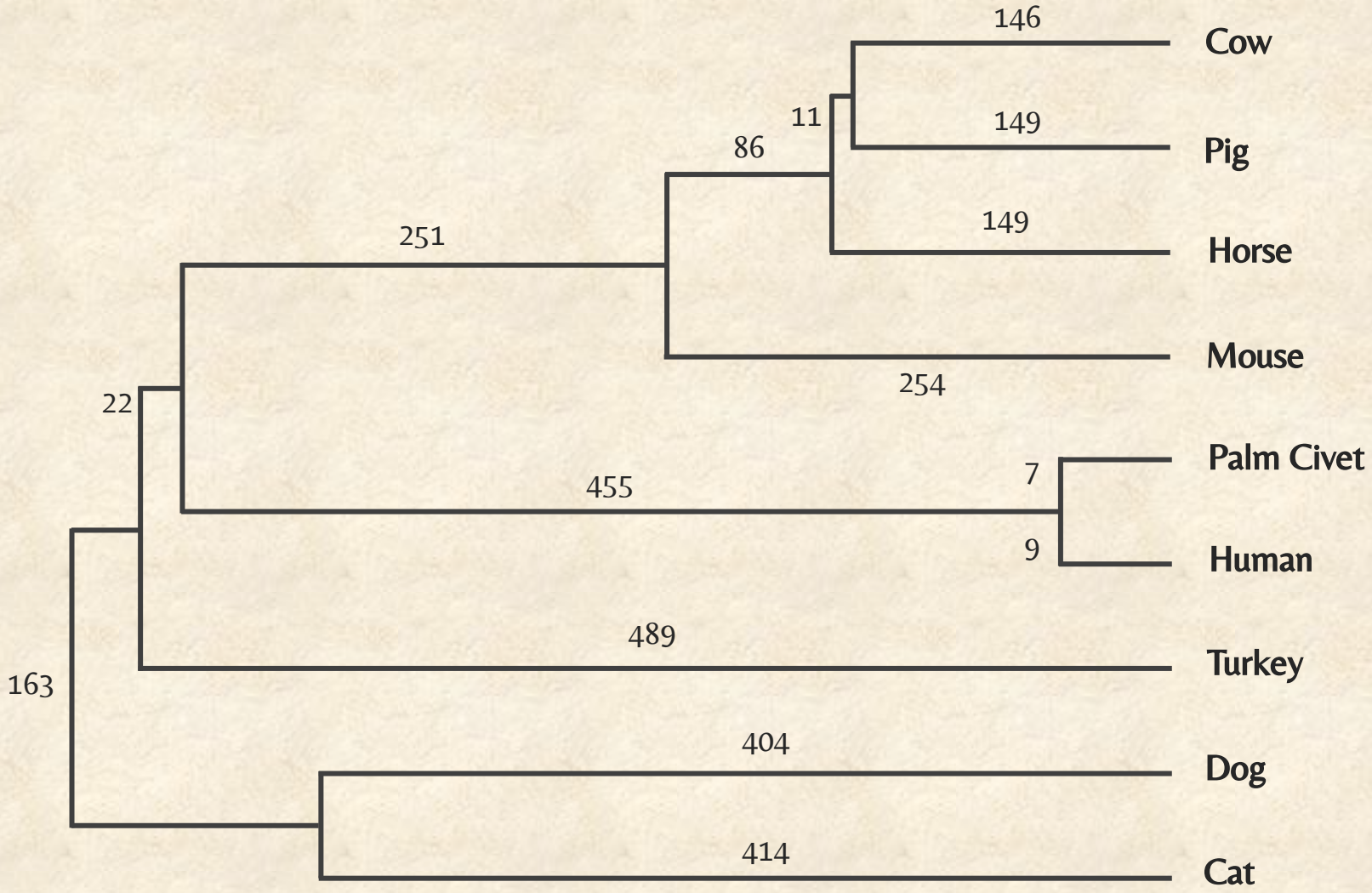
Exercise Break: Find the tree returned by **NeighborJoining** on the following non-additive matrix. How does the result compare with the tree produced by **UPGMA**?

D

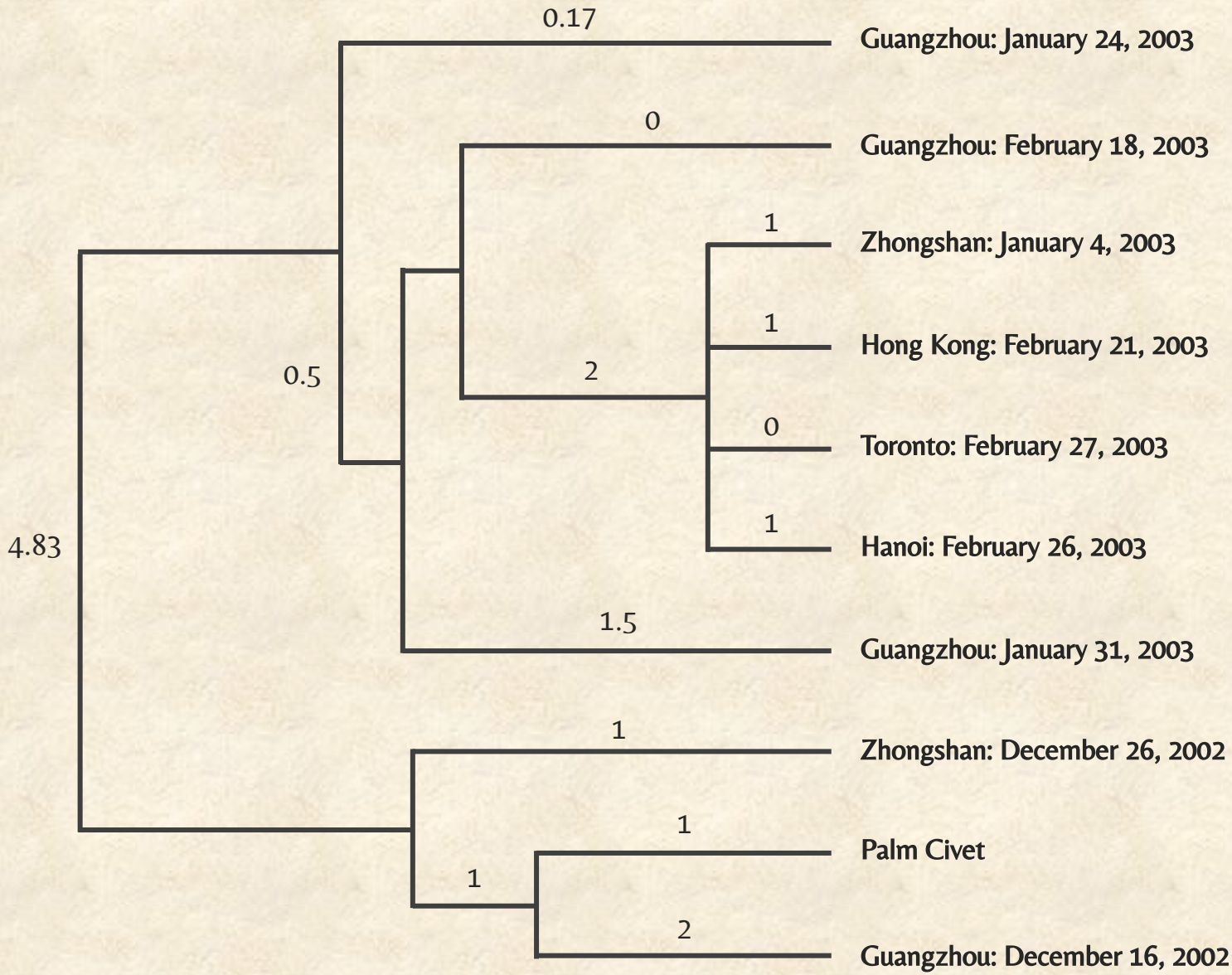
	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0



Neighbor-Joining on Coronaviruses



Neighbor-Joining on Coronaviruses



Weakness of Distance-Based Methods

Distance-based algorithms for evolutionary tree reconstruction say nothing about ancestral states at internal nodes.

Weakness of Distance-Based Methods

Distance-based algorithms for evolutionary tree reconstruction say nothing about ancestral states at internal nodes.

We *lost* information when we converted a multiple alignment to a distance matrix…

Species	Alignment	Distance Matrix			
		Chimp	**Human**	**Seal**	**Whale**
Chimp	ACGTAGGCCT	0	3	6	4
Human	ATGTAAGACT	3	0	7	5
Seal	TCGAGAGCAC	6	7	0	2
Whale	TCGAAAGCAT	4	5	2	0

Outline

- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- **Character-Based Tree Reconstruction**
- The Small Parsimony Problem
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

Character Tables

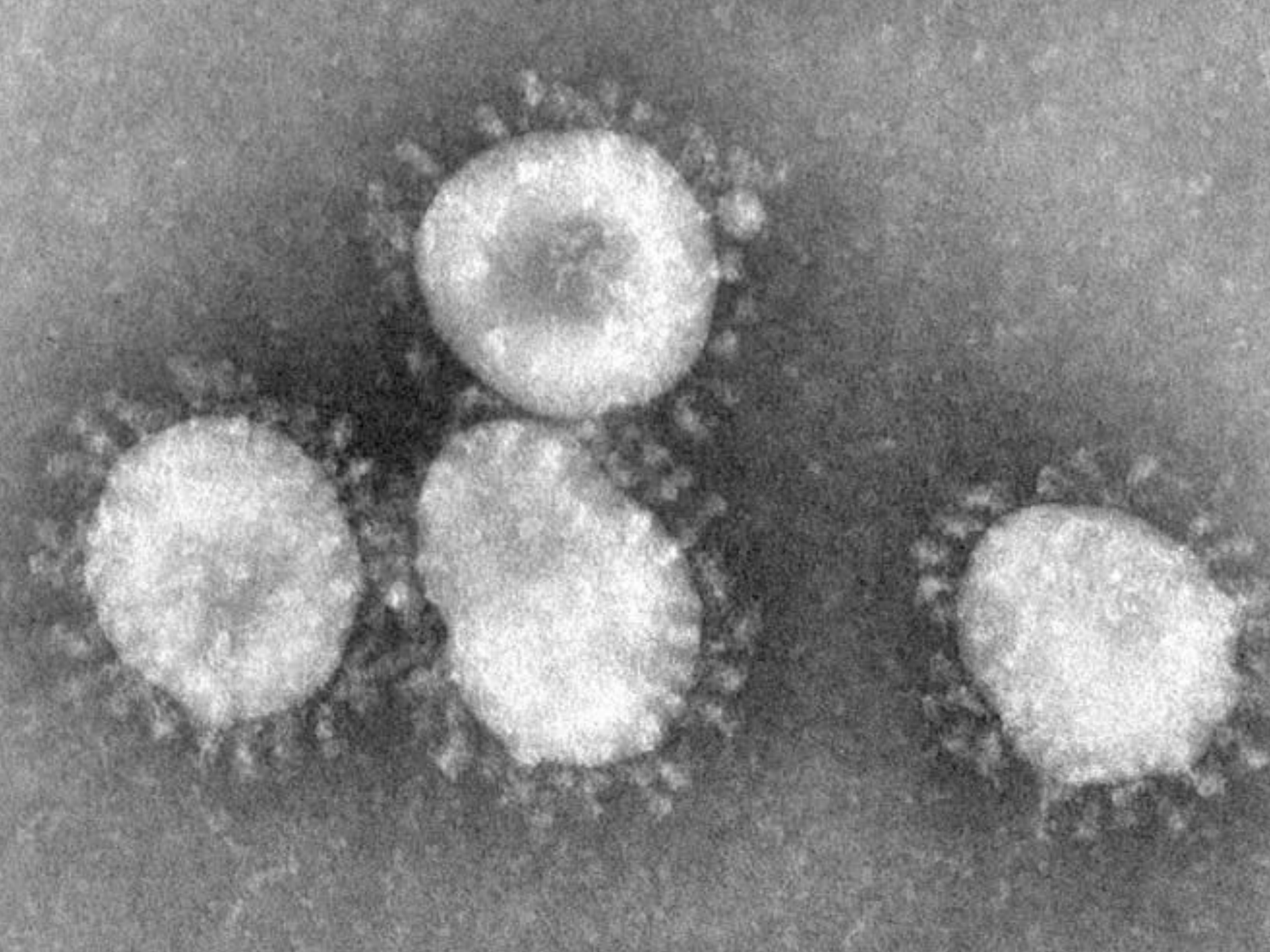
Fifty years ago, researchers constructed trees from anatomical/physiological properties called **characters**.

	wings	legs
winged stick insect	Yes	6
wingless stick insect	No	6
giant centipede	No	42

Character-Based Phylogeny

Character-Based Phylogeny Problem: *Reconstruct a phylogeny from characters.*

- **Input:** An n x m character table for n species and m characters.
- **Output:** A tree in which species with similar character values occur near each other.

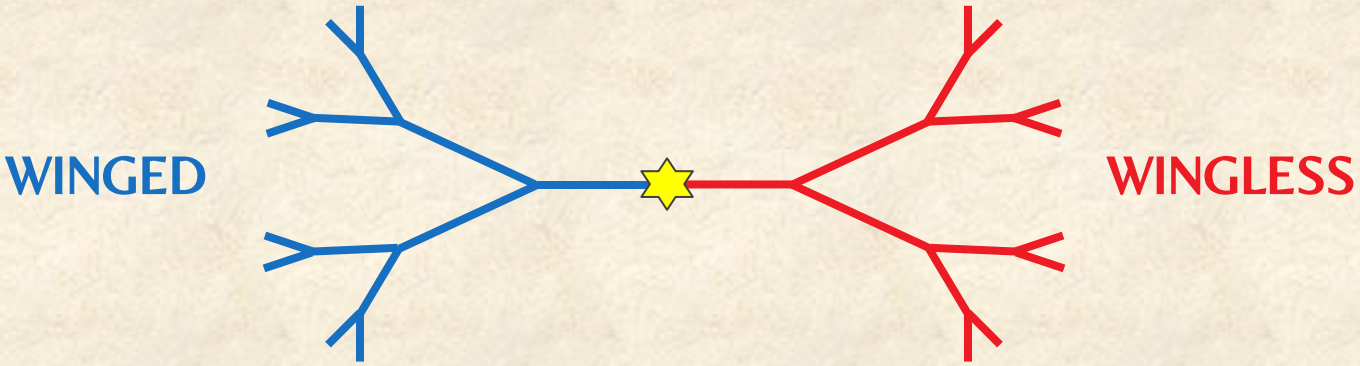


From Characters to a Phylogeny

STOP and Think: How would you construct an evolutionary tree from characters?

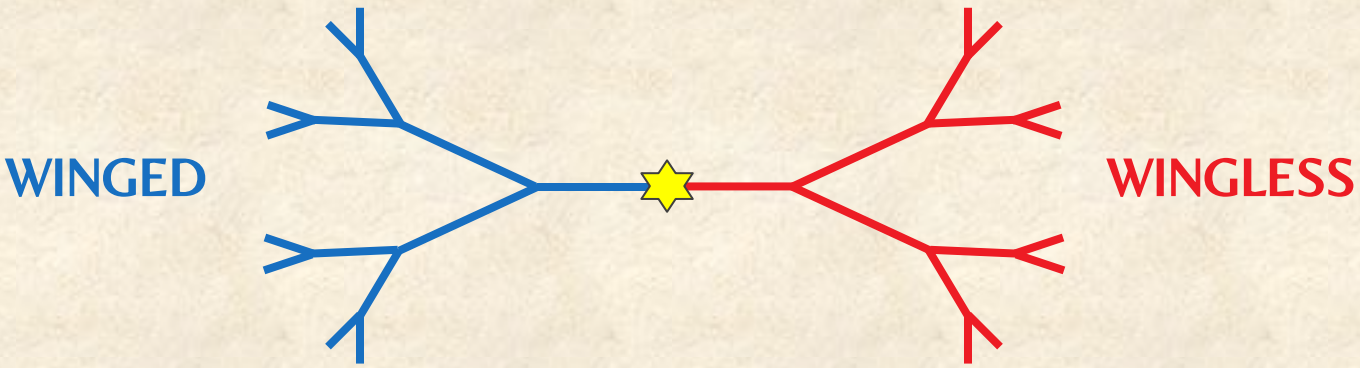
From Characters to a Phylogeny

STOP and Think: How would you construct an evolutionary tree from characters?



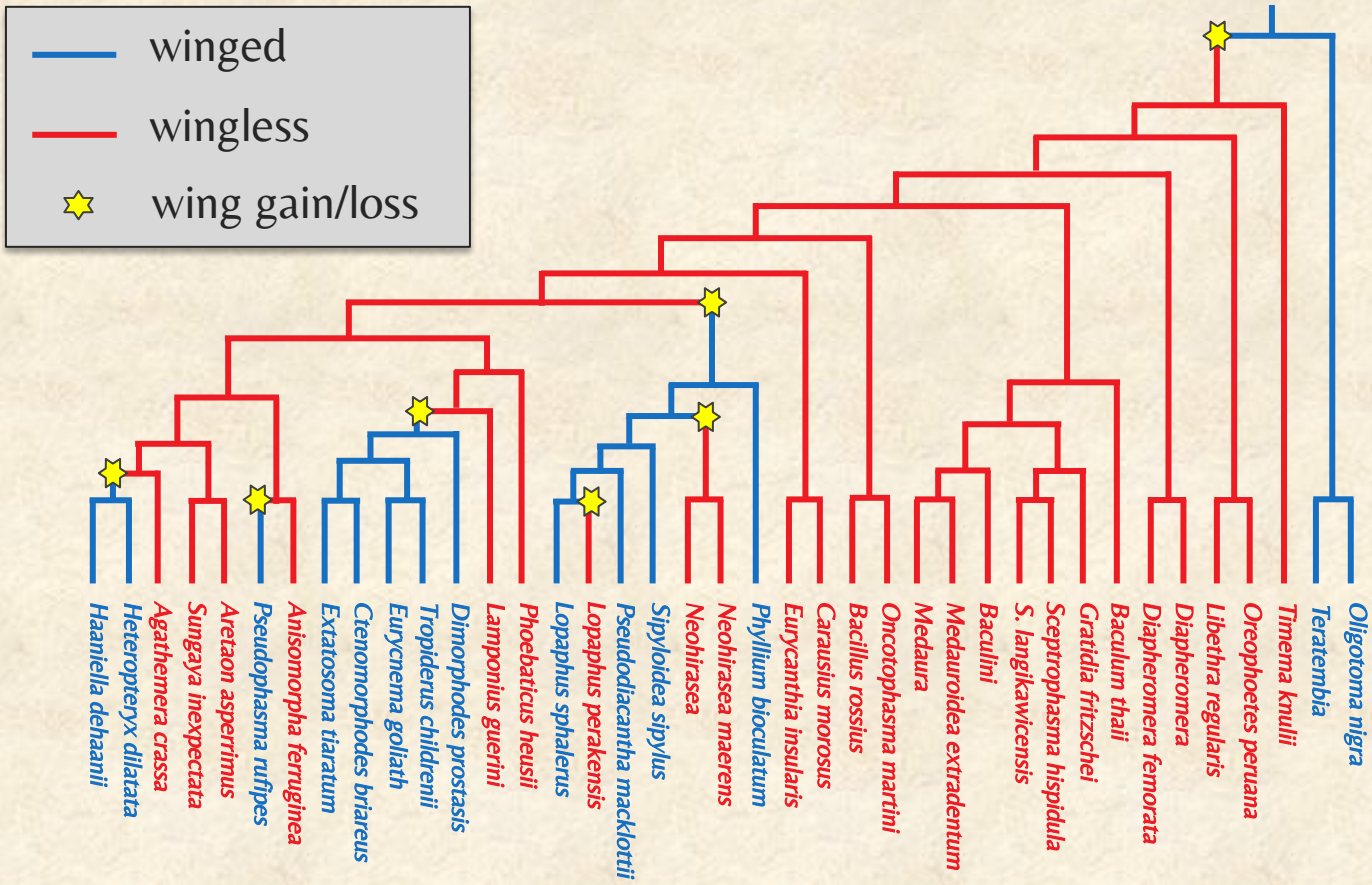
From Characters to a Phylogeny

STOP and Think: How would you construct an evolutionary tree from characters?

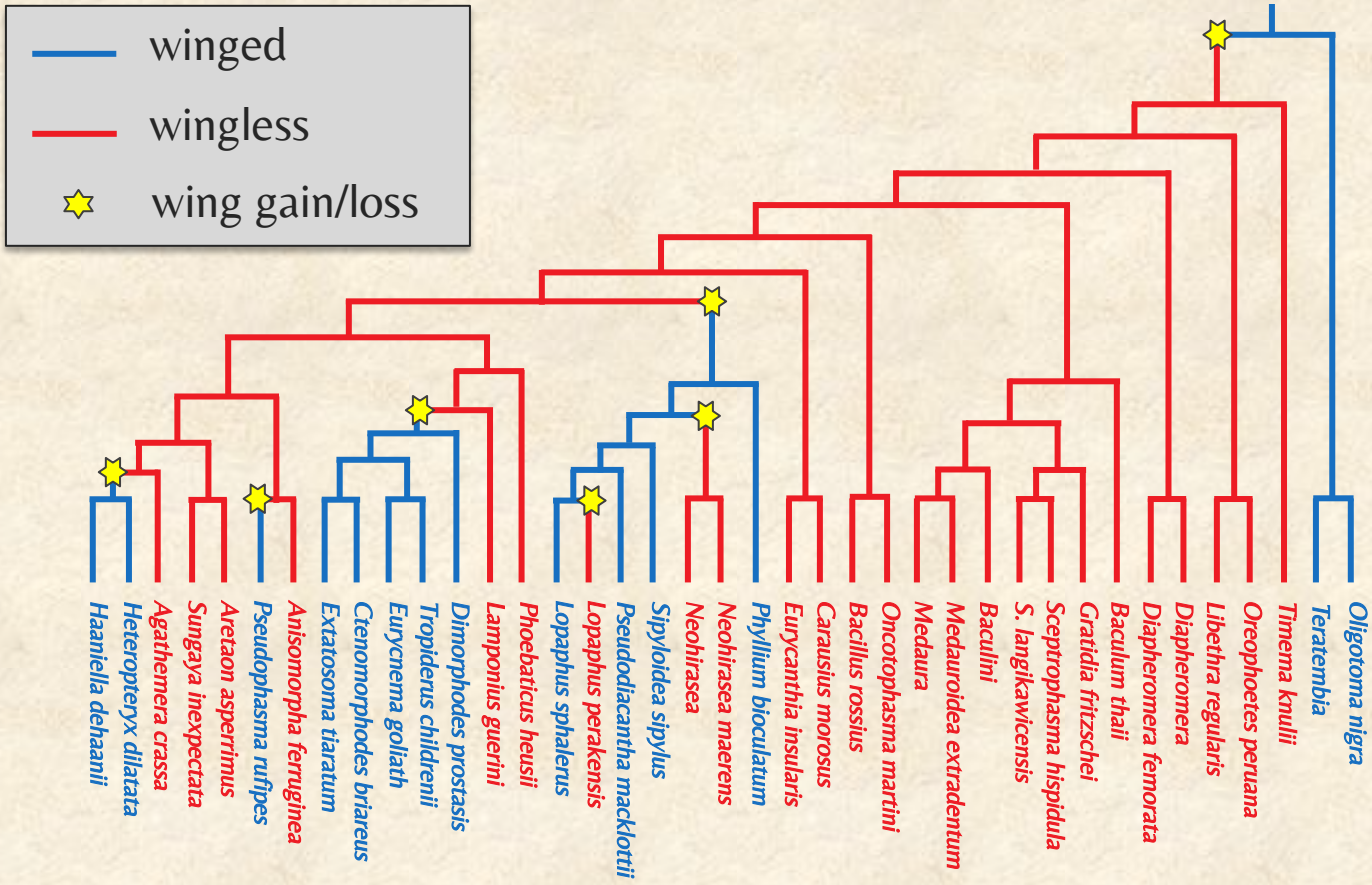


Dollo's principle of irreversibility (1893): evolution doesn't reinvent the same organ (e.g. insect wings).

Dollo's Principle Violated



Dollo's Principle Violated



STOP and Think: What do you think has happened?

An Alignment As a Character Table

Species	Alignment
Chimp	ACGTAGGCCT
Human	ATGTAAGACT
Seal	TCGAGAGCAC
Whale	TCGAAAGCAT

An Alignment As a Character Table

Species	Alignment
Chimp	ACGTAGGCCT
Human	ATGTAAGACT
Seal	TCGAGAGCAC
Whale	TCGAAAGCAT

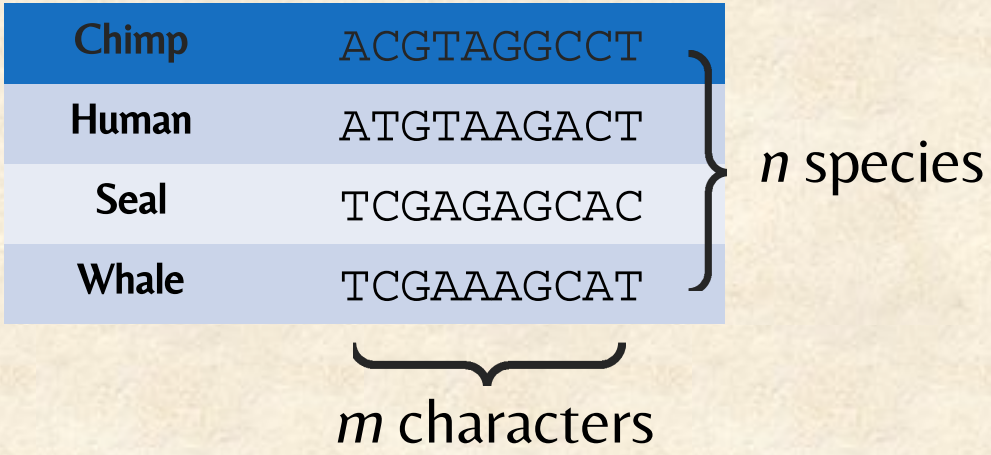
n species

m characters

Outline

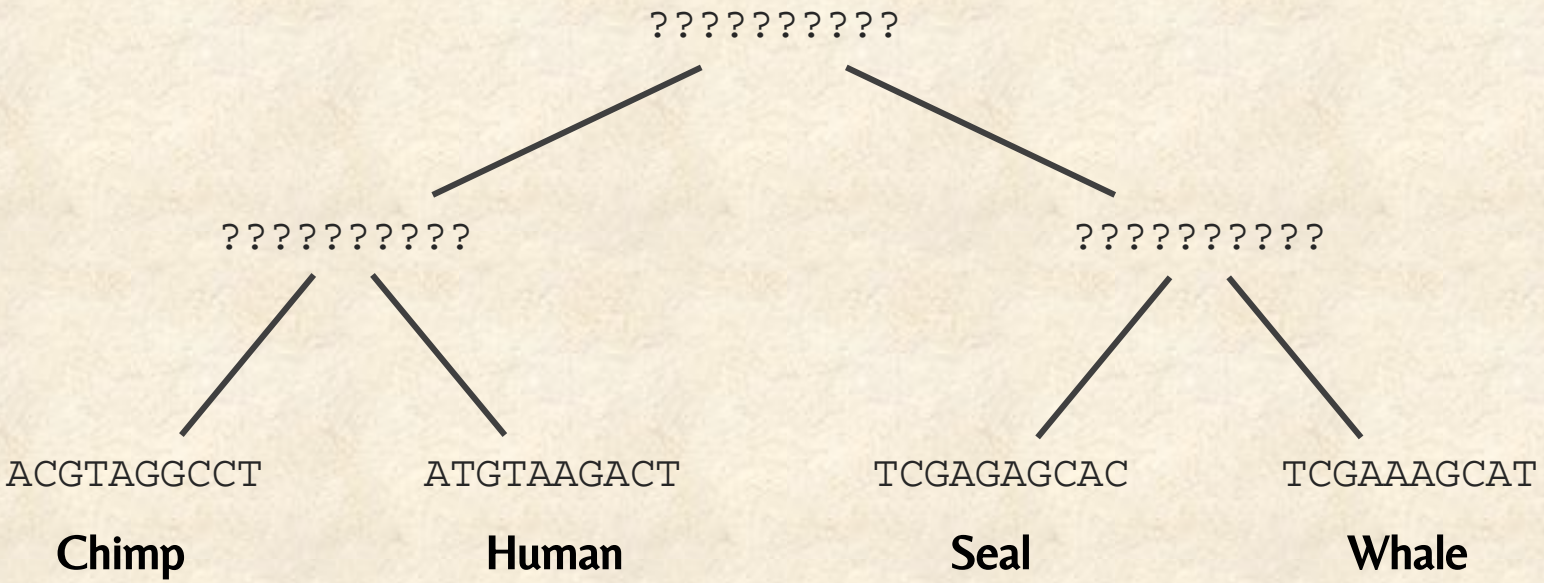
- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- **The Small Parsimony Problem**
- The Large Parsimony Problem
- Evolutionary Tree Reconstruction in the Modern Era

Toward a Computational Problem

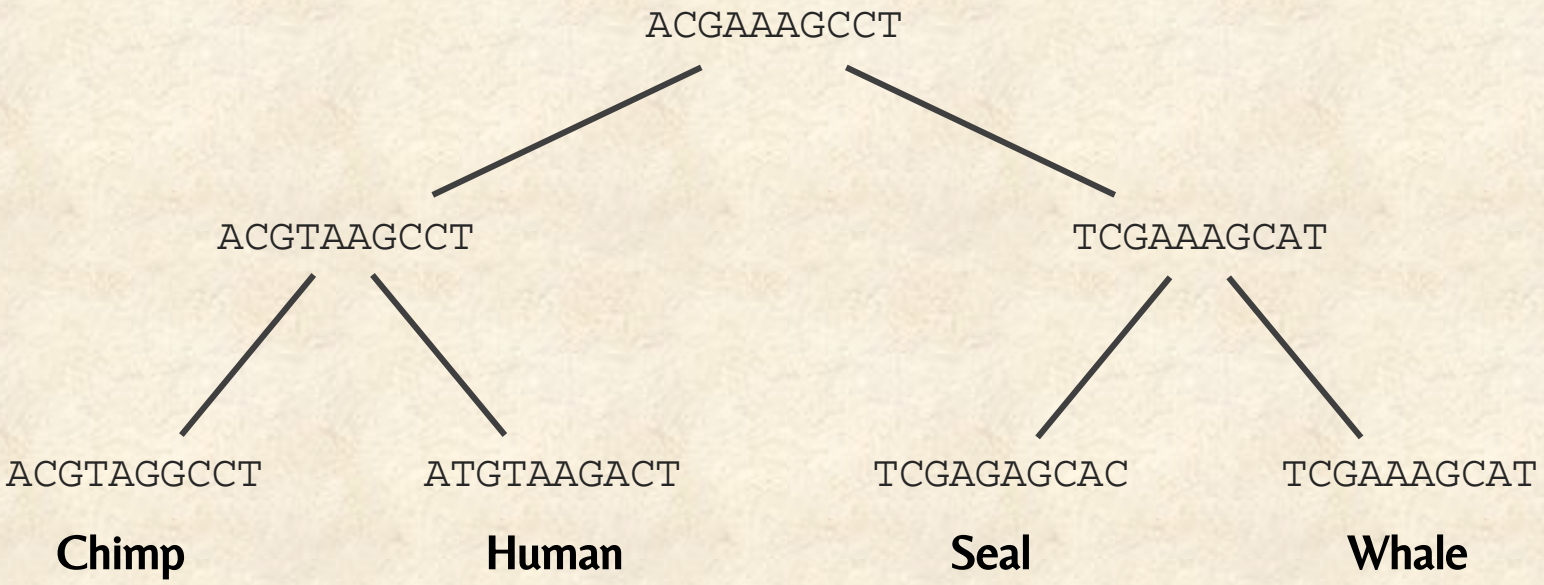


Toward a Computational Problem

Chimp	ACGTAGGCCT
Human	ATGTAAGACT
Seal	TCGAGAGCAC
Whale	TCGAAAGCAT

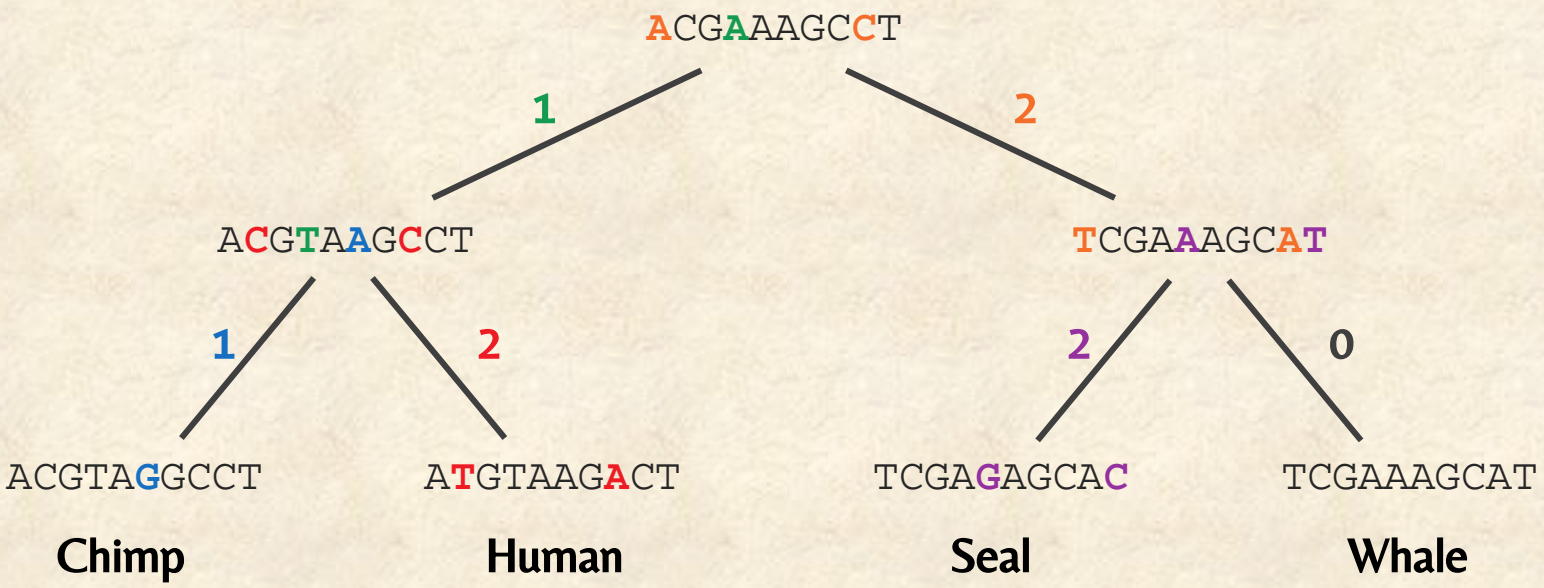


Toward a Computational Problem



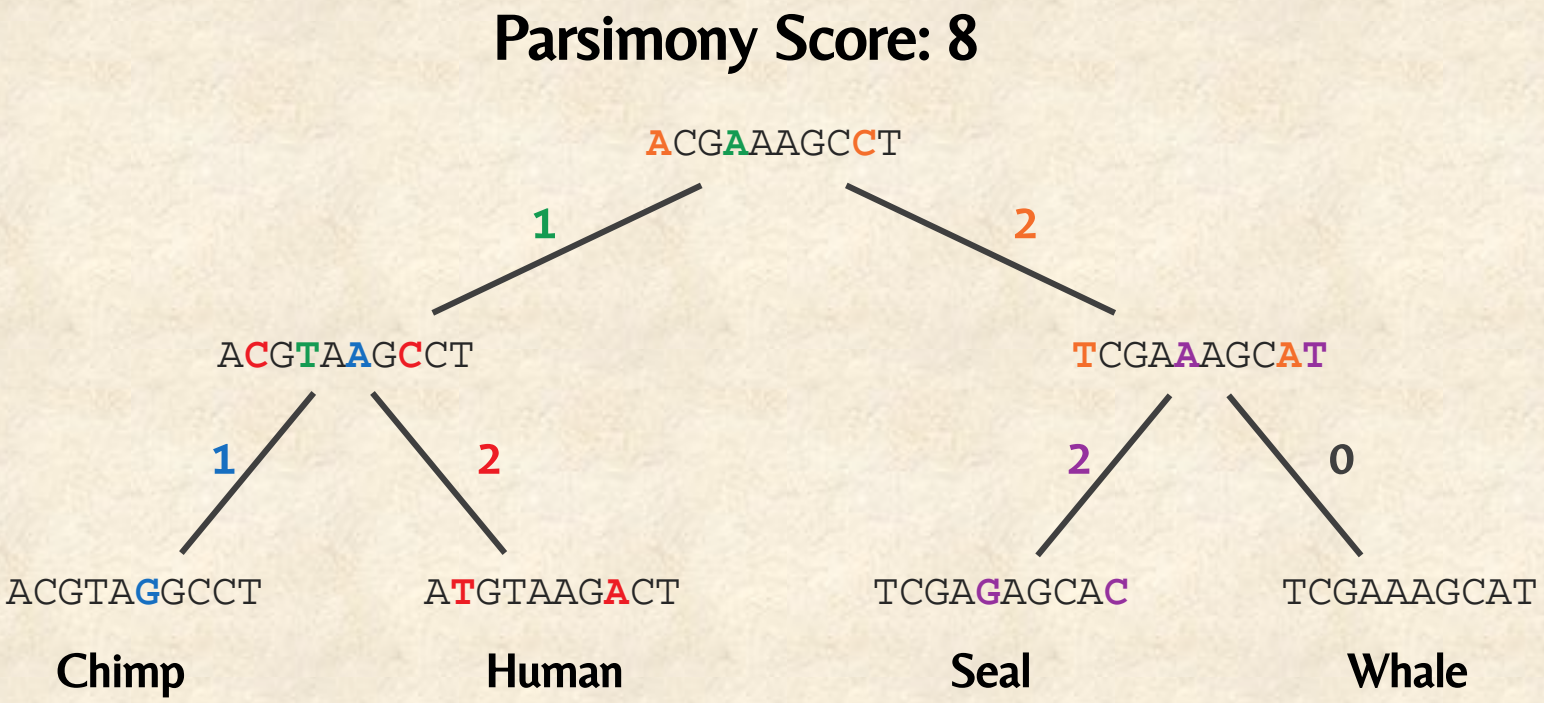
Toward a Computational Problem

Parsimony score: sum of Hamming distances along each edge.



Toward a Computational Problem

Parsimony score: sum of Hamming distances along each edge.



Toward a Computational Problem

Small Parsimony Problem: *Find the most parsimonious labeling of the internal nodes of a rooted tree.*

- **Input:** A rooted binary tree with each leaf labeled by a string of length *m*.
- **Output:** A labeling of all other nodes of the tree by strings of length *m* that minimizes the tree's parsimony score.

Toward a Computational Problem

Small Parsimony Problem: *Find the most parsimonious labeling of the internal nodes of a rooted tree.*

- **Input:** A rooted binary tree with each leaf labeled by a string of length *m*.
- **Output:** A labeling of all other nodes of the tree by strings of length *m* that minimizes the tree's parsimony score.

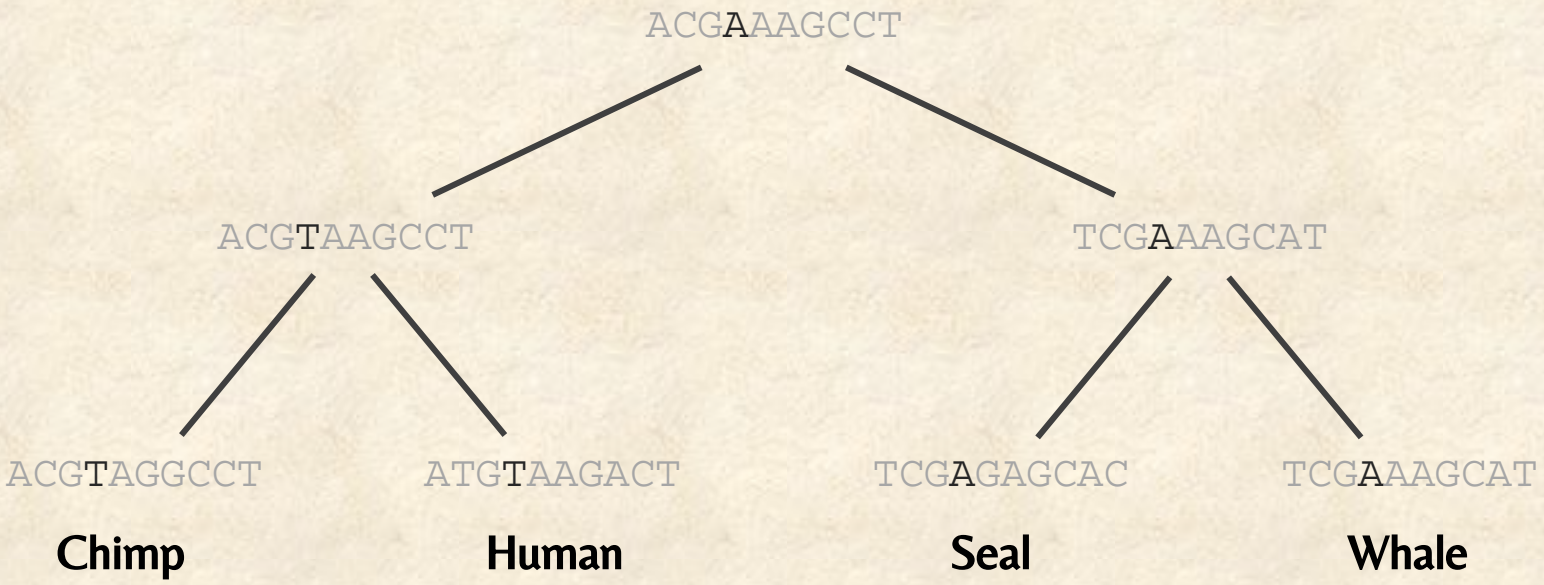
STOP and Think: Is there any way we can simplify this problem statement?

Toward a Computational Problem

Small Parsimony Problem: *Find the most parsimonious labeling of the internal nodes of a rooted tree.*

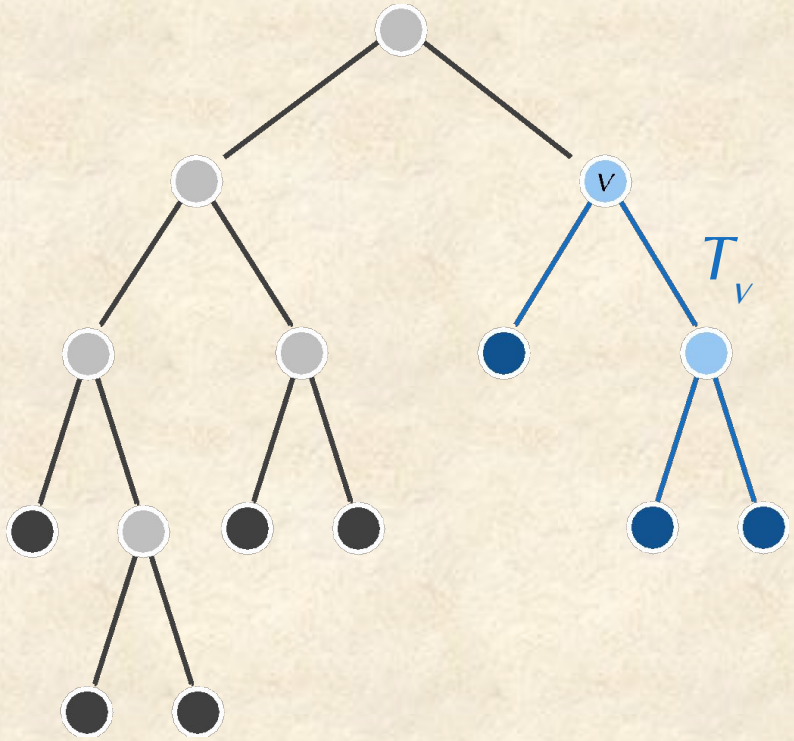
- **Input:** A rooted binary tree with each leaf labeled by a single symbol.
- **Output:** A labeling of all other nodes of the tree by single symbols that minimizes the tree's parsimony score.

Toward a Computational Problem



A Dynamic Programming Algorithm

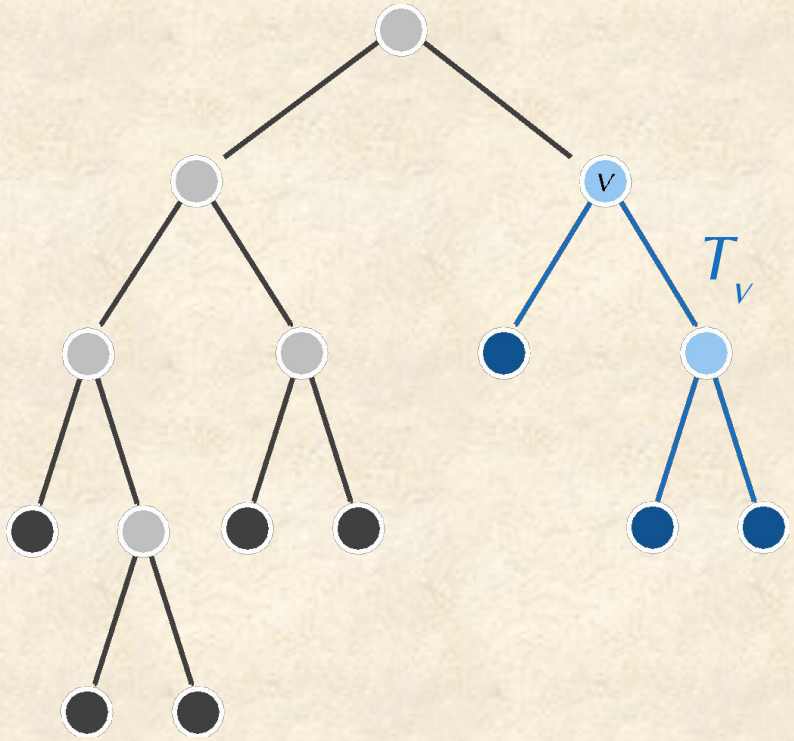
Let T_v denote the subtree of T whose root is v.



A Dynamic Programming Algorithm

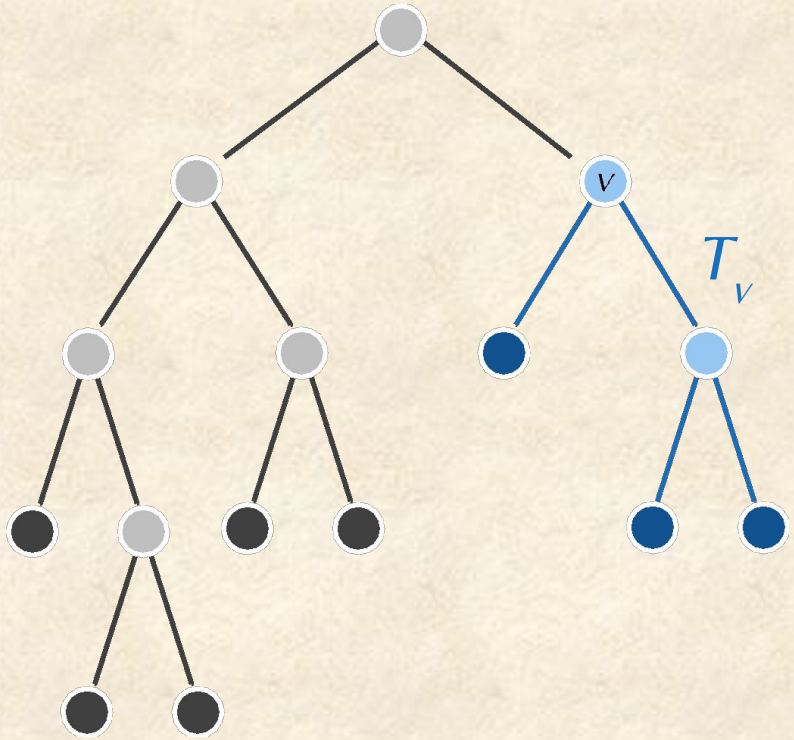
Let T_v denote the subtree of T whose root is v.

Define $s_k(v)$ as the minimum parsimony score of T_v over all labelings of T_v, assuming that v is labeled by k.



A Dynamic Programming Algorithm

Let T_v denote the subtree of T whose root is v.



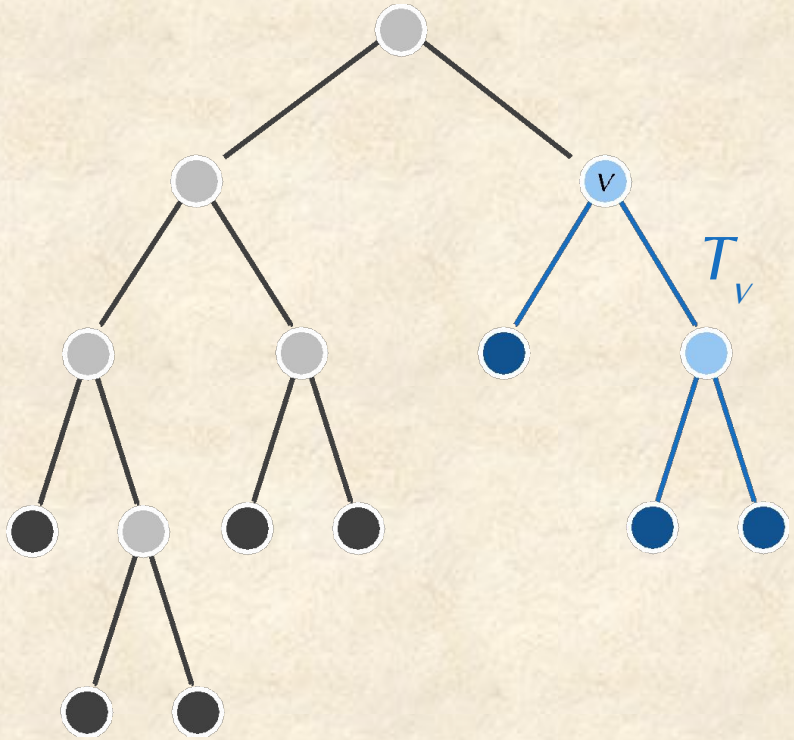
Define $s_k(v)$ as the minimum parsimony score of T_v over all labelings of T_v, assuming that v is labeled by k.

The minimum parsimony score for the tree is equal to the minimum value of $s_k(root)$ over all symbols k.

A Dynamic Programming Algorithm

For symbols i and j, define

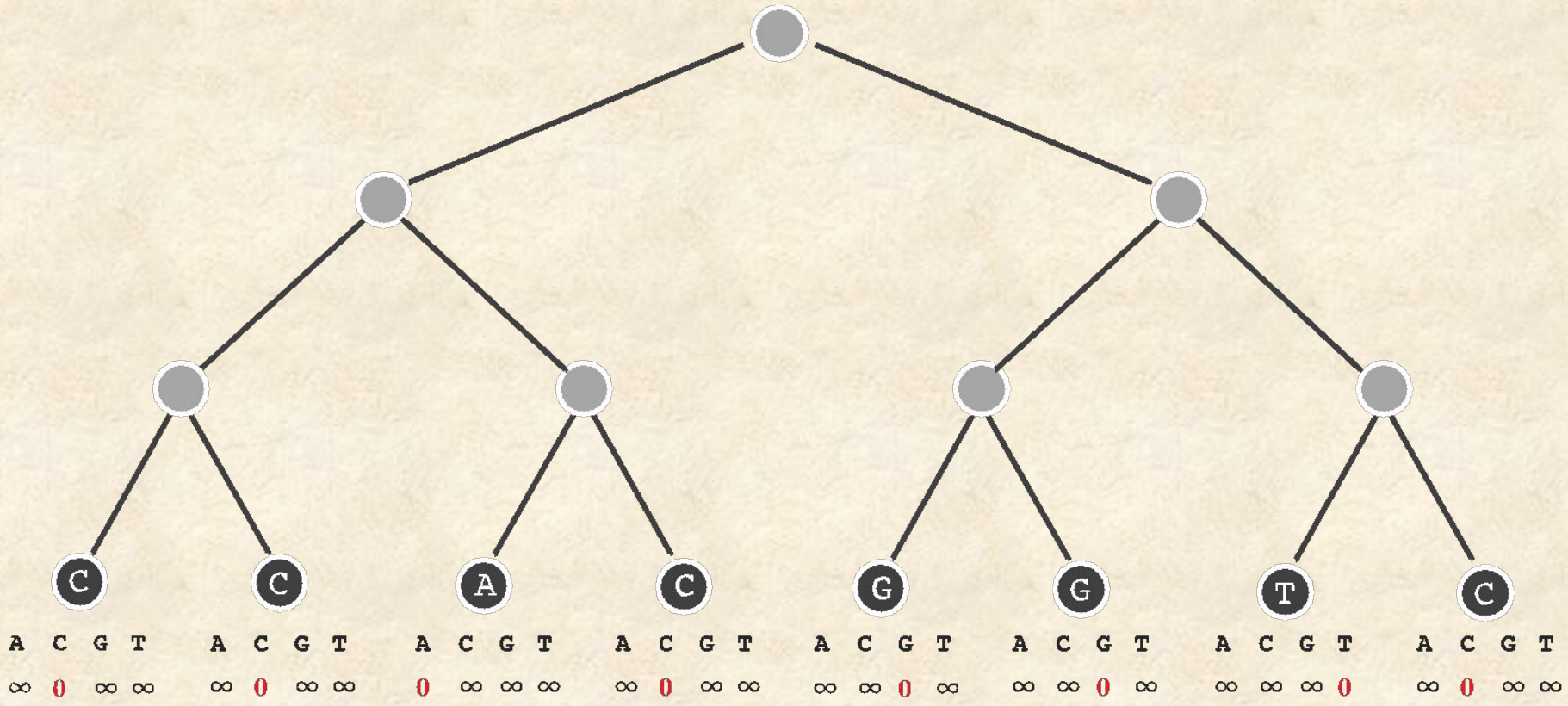
- $\Delta_{i,j} = 0$ if $i = j$
- $\Delta_{i,j} = 1$ otherwise.



Exercise Break: Prove the following recurrence relation:

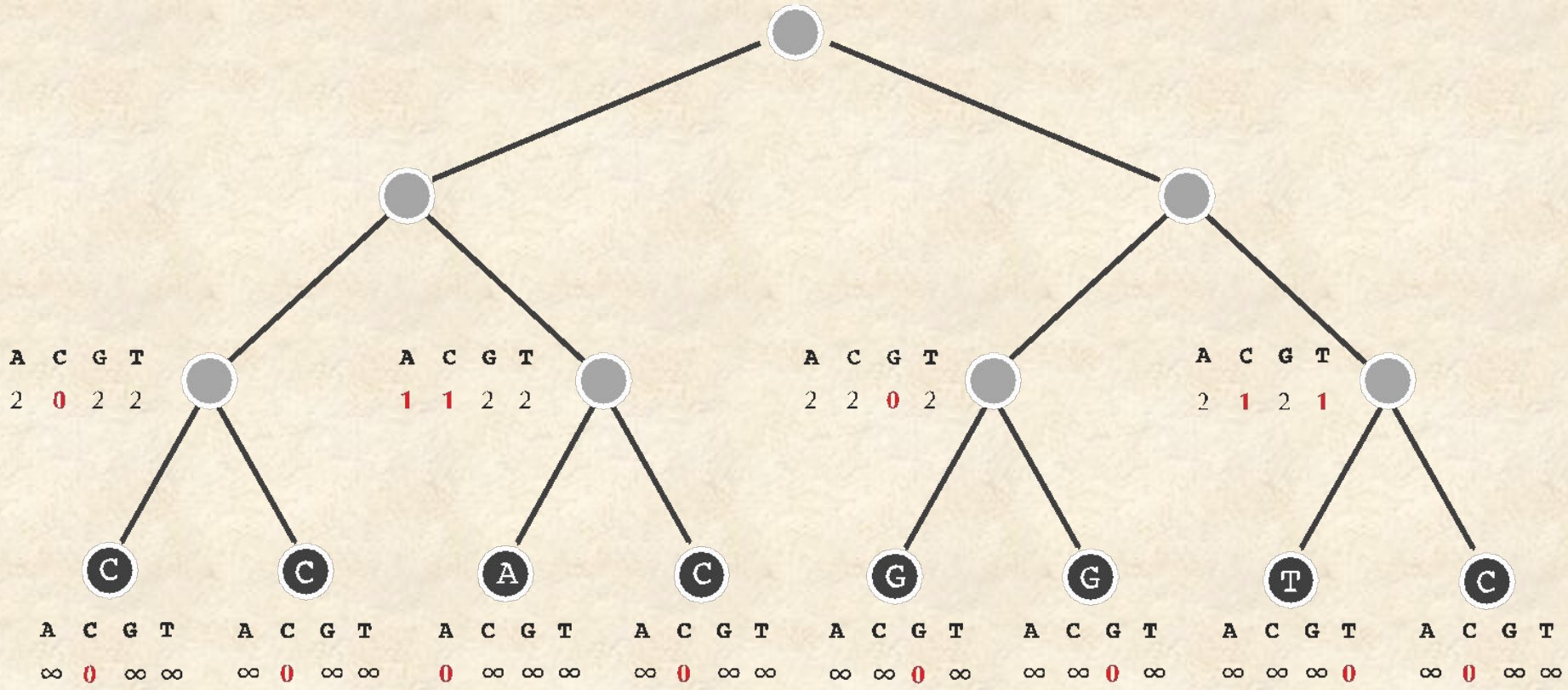
$$s_k(v) = \min_{\text{all symbols } i} \{s_i(Daughter(v)) + \Delta_{i,k}\} + \min_{\text{all symbols } i} \{s_i(Son(v)) + \Delta_{j,k}\}$$

A Dynamic Programming Algorithm



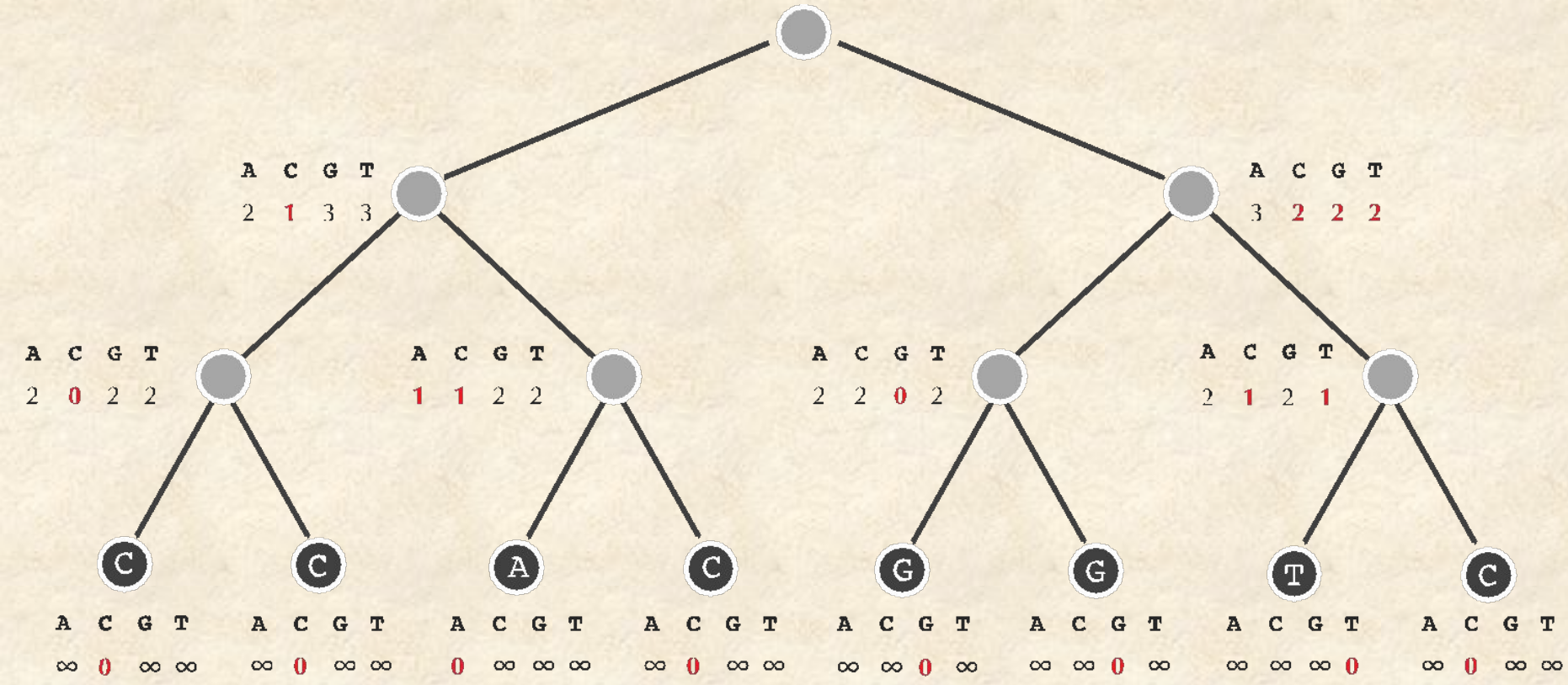
$$s_k(v) = \min_{\text{all symbols } i} \{ s_i(Daughter(v)) + \Delta_{i,k} \} + \min_{\text{all symbols } i} \{ s_i(Son(v)) + \Delta_{j,k} \}$$

A Dynamic Programming Algorithm



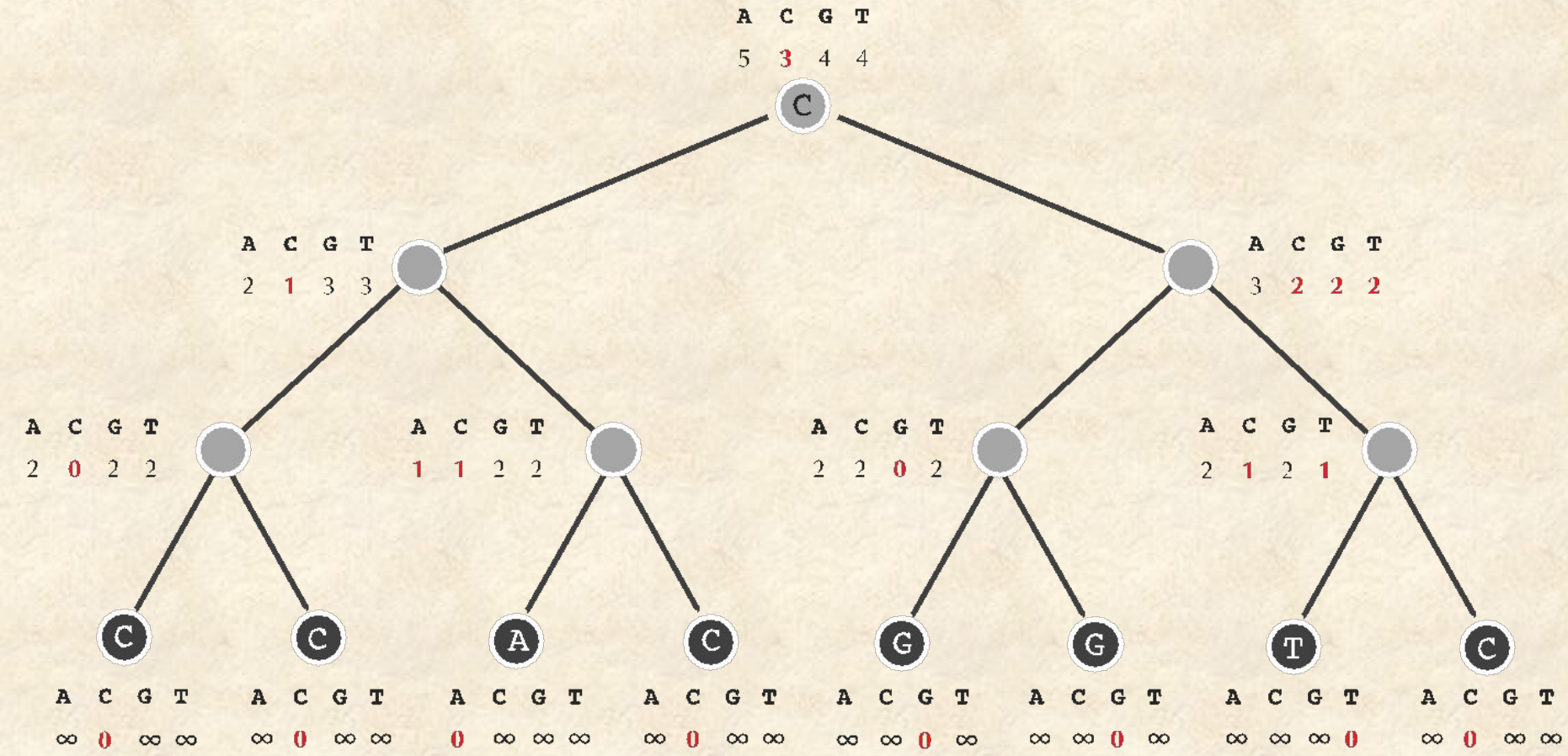
$$s_k(v) = \min_{\text{all symbols } i} \{s_i(Daughter(v)) + \Delta_{i,k}\} + \min_{\text{all symbols } i} \{s_i(Son(v)) + \Delta_{j,k}\}$$

A Dynamic Programming Algorithm



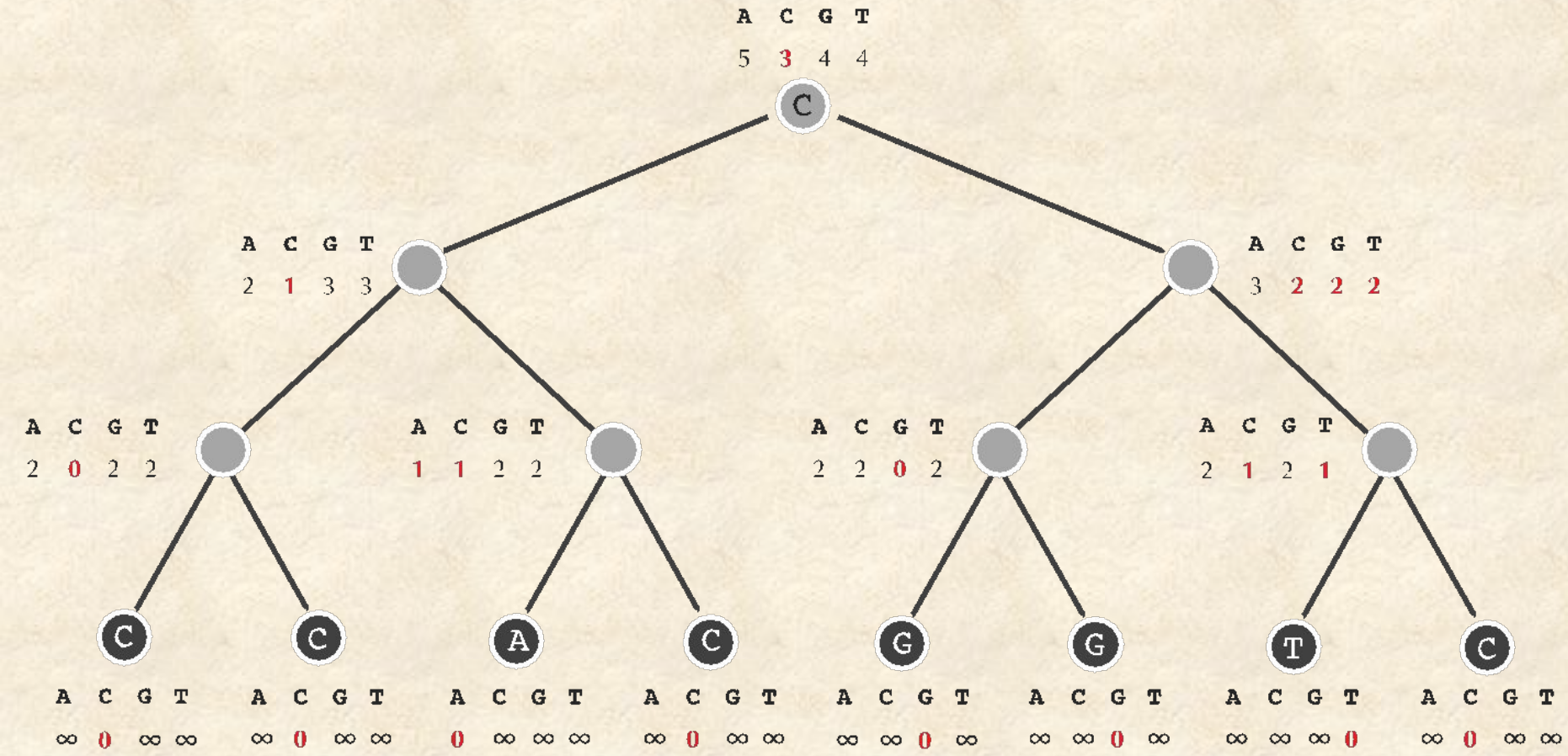
$$s_k(v) = \min_{\text{all symbols } i} \{ s_i(Daughter(v)) + \Delta_{i,k} \} + \min_{\text{all symbols } i} \{ s_i(Son(v)) + \Delta_{j,k} \}$$

A Dynamic Programming Algorithm



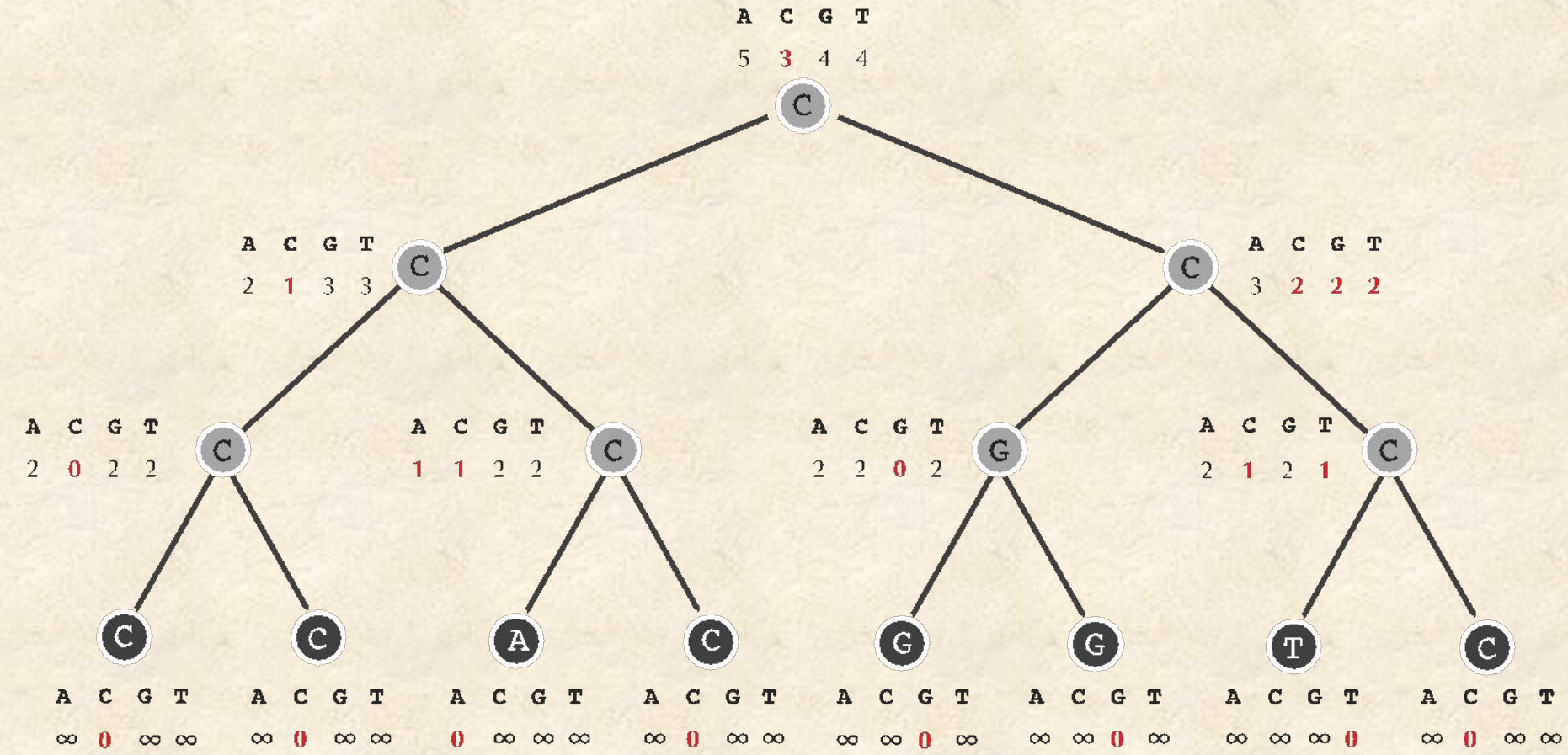
$$s_k(v) = \min_{\text{all symbols } i} \{ s_i(Daughter(v)) + \Delta_{i,k} \} + \min_{\text{all symbols } i} \{ s_i(Son(v)) + \Delta_{j,k} \}$$

A Dynamic Programming Algorithm

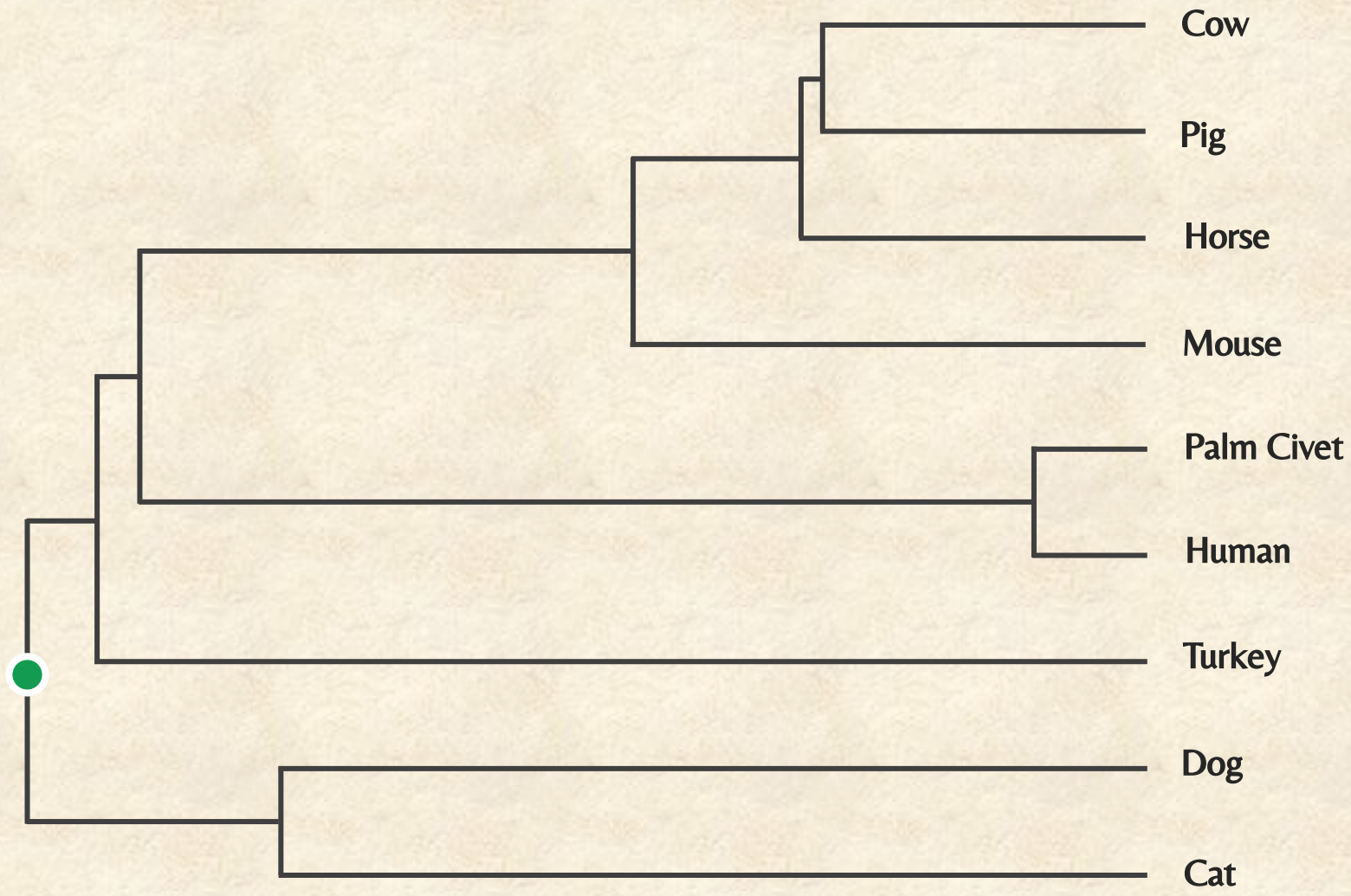


Exercise Break: "Backtrack" to fill in the remaining nodes of the tree.

A Dynamic Programming Algorithm



Code Challenge: Solve the Small Parsimony Problem.



Exercise Break: Apply **SmallParsimony** to this tree to reconstruct ancestral coronavirus sequences.

Small Parsimony for Unrooted Trees

Small Parsimony in an Unrooted Tree Problem: *Find the most parsimonious labeling of the internal nodes of an unrooted tree.*

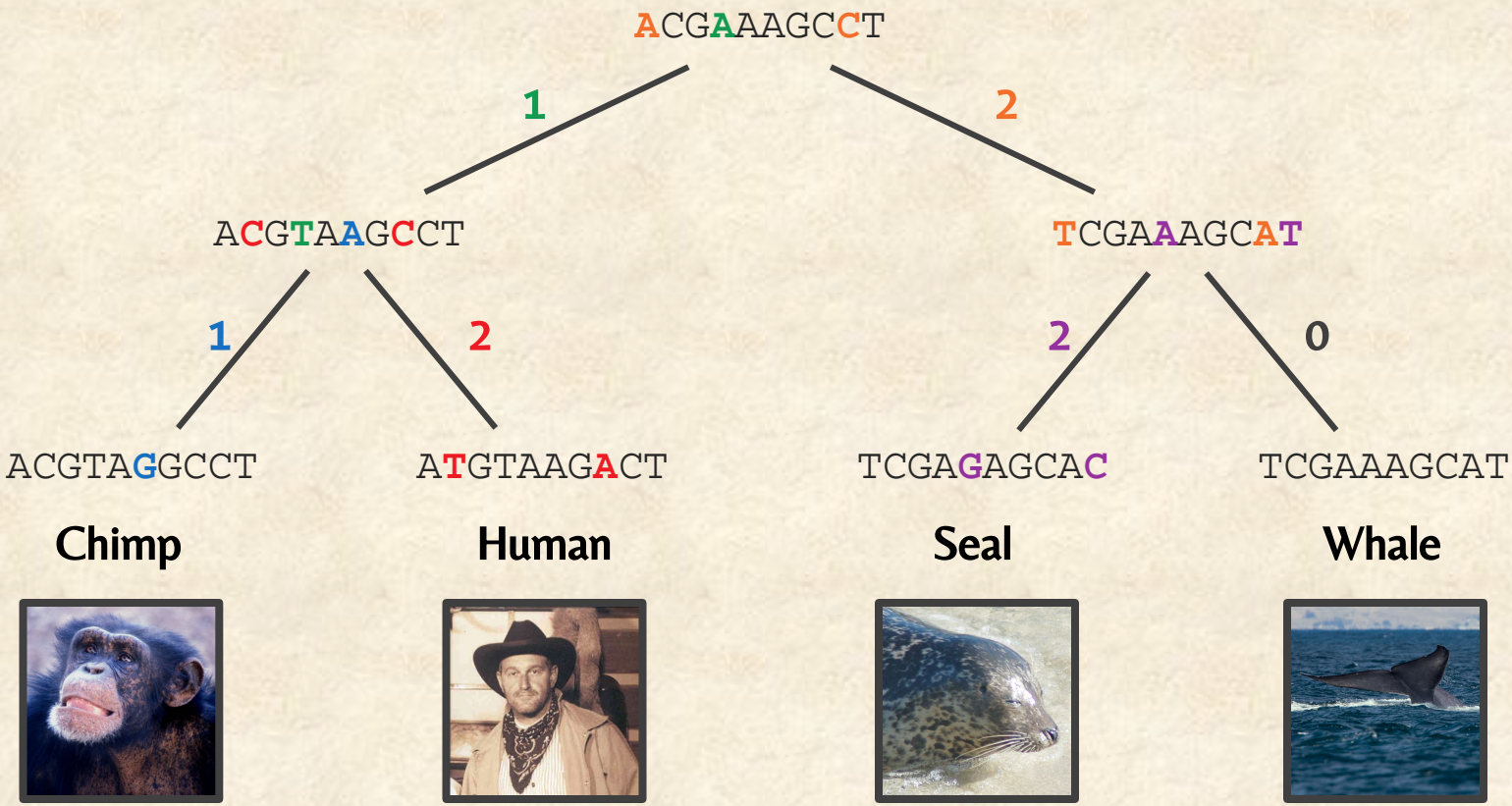
- **Input:** An unrooted binary tree with each leaf labeled by a string of length *m*.
- **Output:** A position of the root and a labeling of all other nodes of the tree by strings of length *m* that minimizes the tree's parsimony score.

Code Challenge: Solve this problem.

Outline

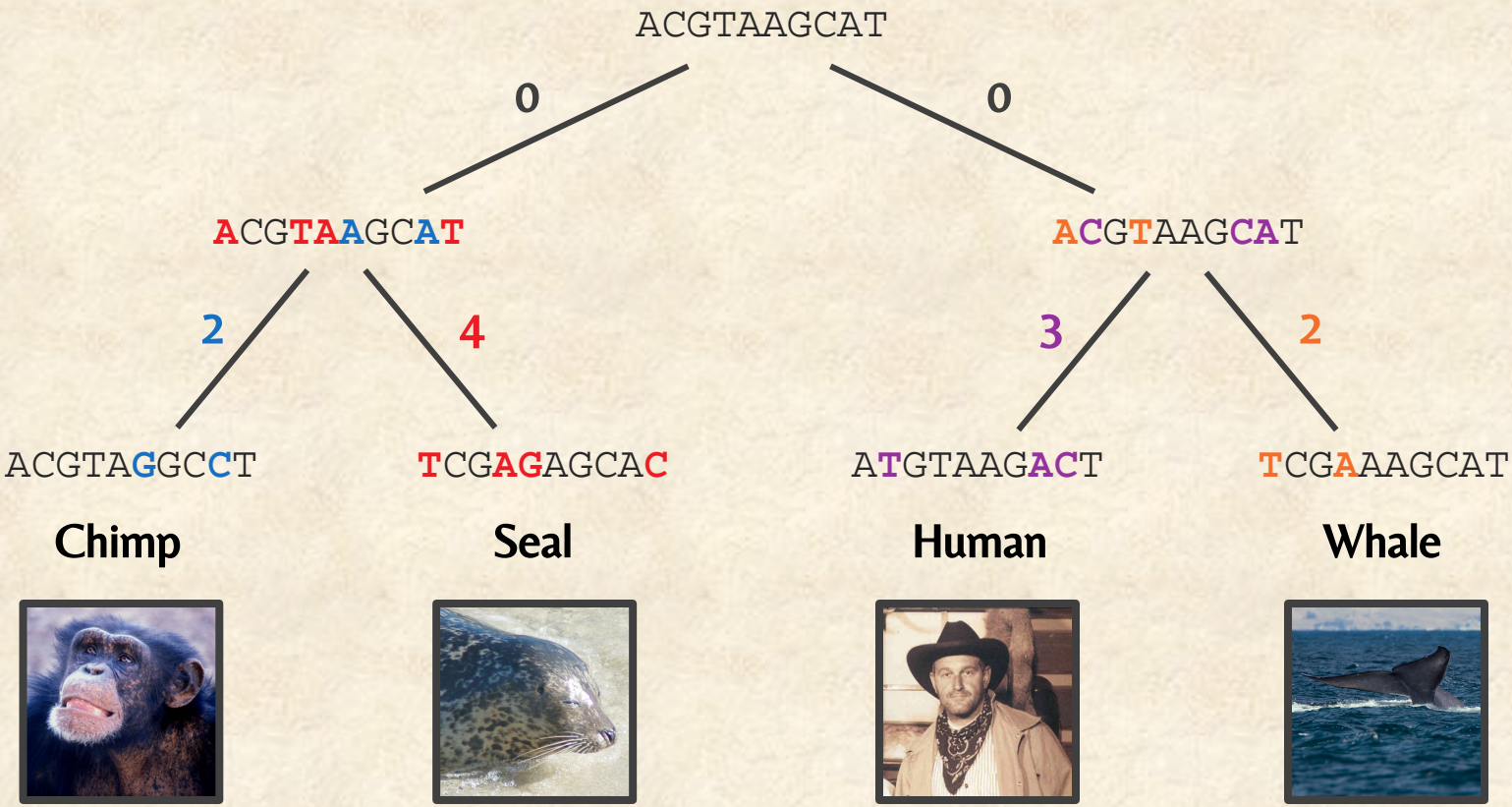
- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- **The Large Parsimony Problem**
- Evolutionary Tree Reconstruction in the Modern Era

Finding the Most Parsimonious Tree



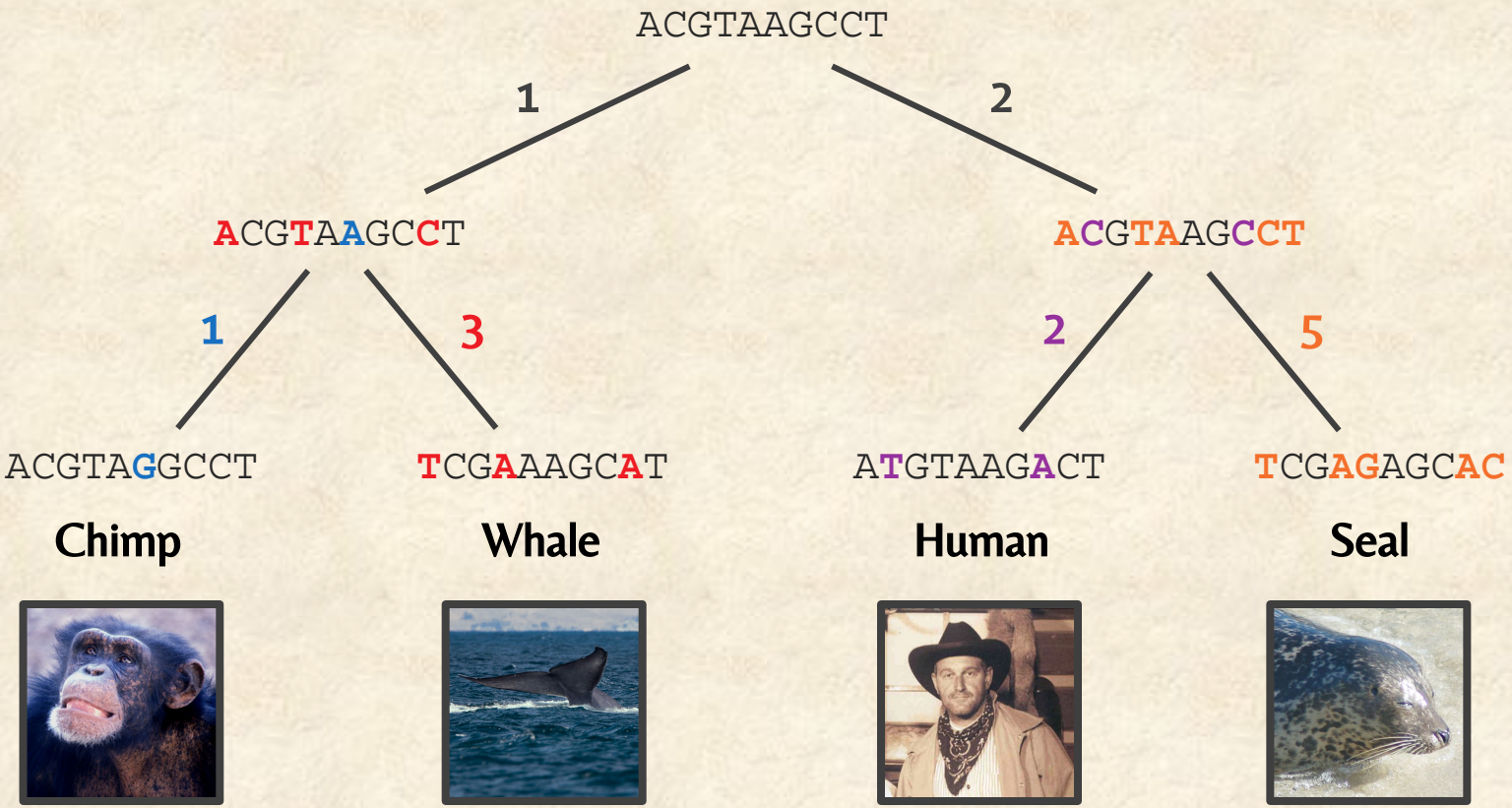
Parsimony Score: 8

Finding the Most Parsimonious Tree



Parsimony Score: 11

Finding the Most Parsimonious Tree



Parsimony Score: 14

Finding the Most Parsimonious Tree

Large Parsimony Problem: *Given a set of strings, find a tree* (*with leaves labeled by all these strings*) *having minimum parsimony score.*

- **Input:** A collection of strings of equal length.
- **Output:** A rooted binary tree *T* that minimizes the parsimony score among all possible rooted binary trees with leaves labeled by these strings.

Finding the Most Parsimonious Tree

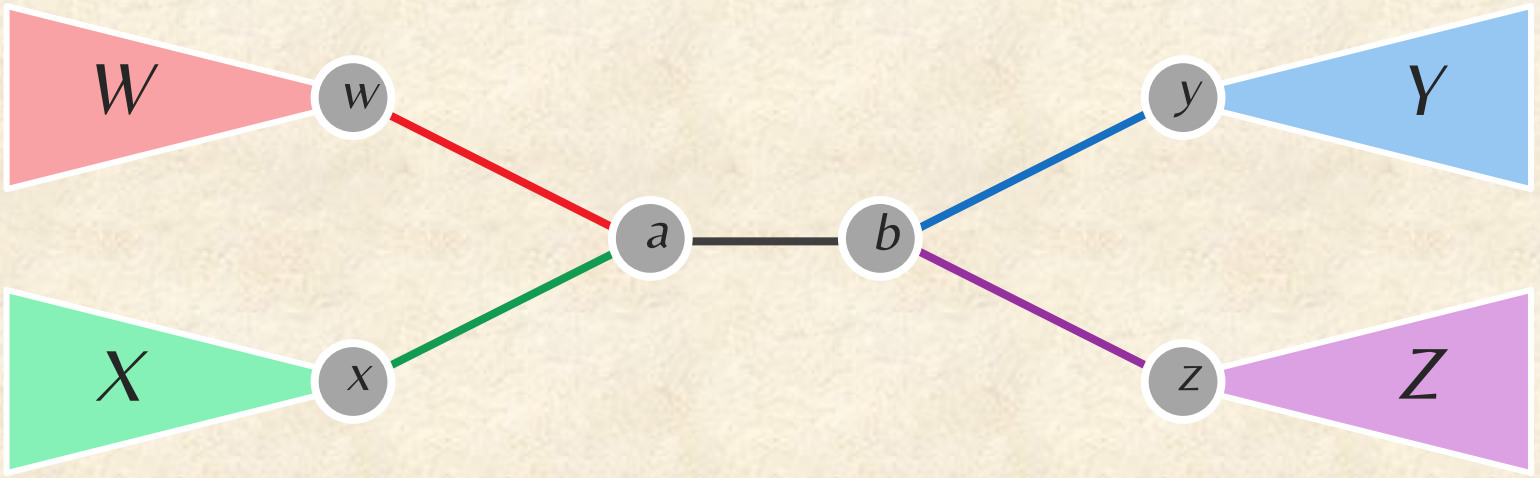
Large Parsimony Problem: *Given a set of strings, find a tree* (*with leaves labeled by all these strings*) *having minimum parsimony score.*

- **Input:** A collection of strings of equal length.
- **Output:** A rooted binary tree *T* that minimizes the parsimony score among all possible rooted binary trees with leaves labeled by these strings.

Unfortunately, this problem is *NP*-Complete…

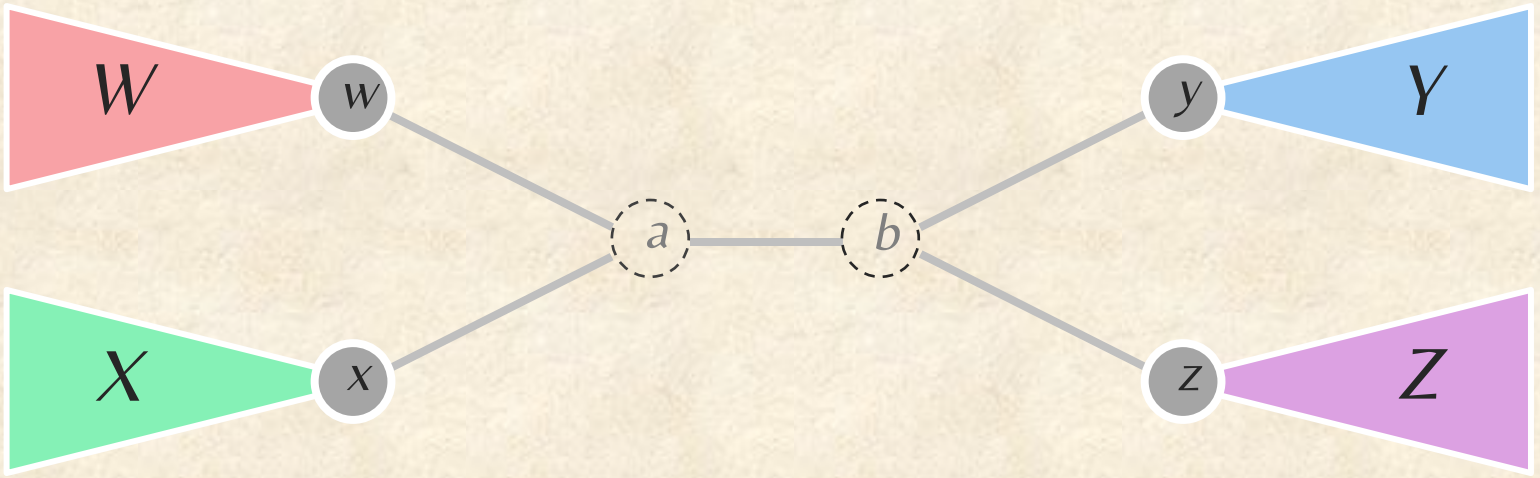
A Greedy Heuristic for Large Parsimony

Note that removing an **internal edge**, an edge connecting two internal nodes (along with the nodes), produces four subtrees (W, X, Y, Z).



A Greedy Heuristic for Large Parsimony

Note that removing an **internal edge**, an edge connecting two internal nodes (along with the nodes), produces four subtrees (W, X, Y, Z).

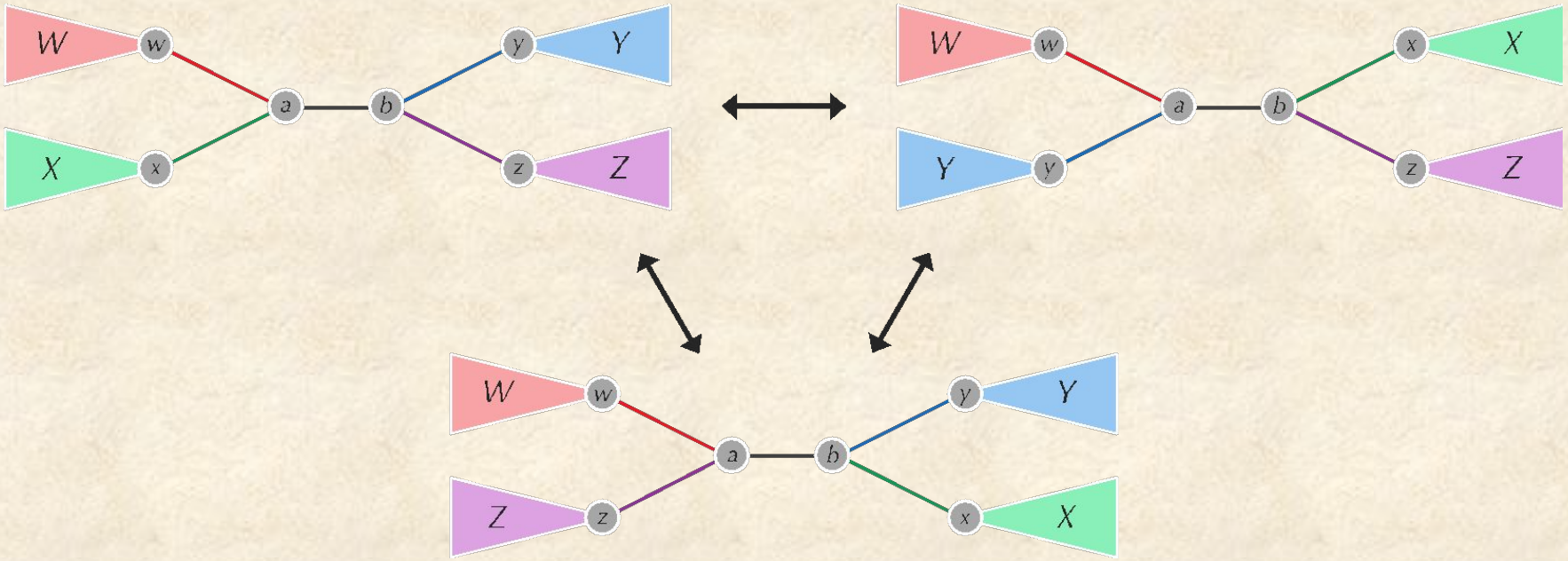


A Greedy Heuristic for Large Parsimony

Note that removing an **internal edge**, an edge connecting two internal nodes (along with the nodes), produces four subtrees (W, X, Y, Z).

A Greedy Heuristic for Large Parsimony

Rearranging these subtrees is called a **nearest neighbor interchange.**



A Greedy Heuristic for Large Parsimony

Nearest Neighbors of a Tree Problem: *Given an edge in a binary tree, generate the two neighbors of this tree.*

- **Input:** An internal edge in a binary tree.
- **Output:** The two nearest neighbors of this tree (for the given internal edge).

Code Challenge: Solve this problem.

A Greedy Heuristic for Large Parsimony

Nearest Neighbor Interchange Heuristic:

1. Set current tree equal to arbitrary binary rooted tree structure.

A Greedy Heuristic for Large Parsimony

Nearest Neighbor Interchange Heuristic:

1. Set current tree equal to arbitrary binary rooted tree structure.
2. Go through all internal edges and perform all possible nearest neighbor interchanges.

A Greedy Heuristic for Large Parsimony

Nearest Neighbor Interchange Heuristic:

1. Set current tree equal to arbitrary binary rooted tree structure.
2. Go through all internal edges and perform all possible nearest neighbor interchanges.
3. Solve Small Parsimony Problem on each tree.

A Greedy Heuristic for Large Parsimony

Nearest Neighbor Interchange Heuristic:

1. Set current tree equal to arbitrary binary rooted tree structure.
2. Go through all internal edges and perform all possible nearest neighbor interchanges.
3. Solve Small Parsimony Problem on each tree.
4. If any tree has parsimony score improving over optimal tree, set it equal to the current tree. Otherwise, return current tree.

A Greedy Heuristic for Large Parsimony

Nearest Neighbor Interchange Heuristic:

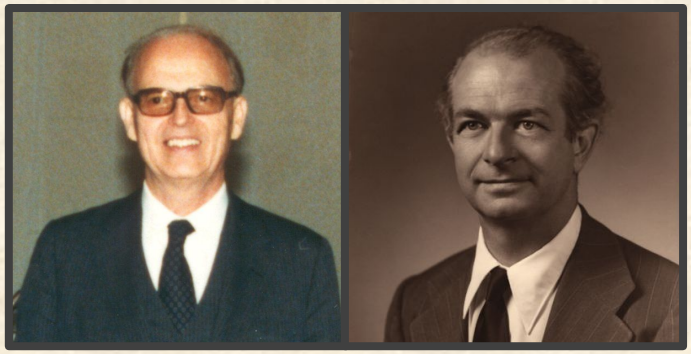
1. Set current tree equal to arbitrary binary rooted tree structure.
2. Go through all internal edges and perform all possible nearest neighbor interchanges.
3. Solve Small Parsimony Problem on each tree.
4. If any tree has parsimony score improving over optimal tree, set it equal to the current tree. Otherwise, return current tree.

Code Challenge: Implement the nearest-neighbor interchange heuristic.

Outline

- The Fastest Outbreak
- Transforming Distance Matrices into Evolutionary Trees
- Toward an Algorithm for Distance-Based Phylogeny Construction
- Additive Phylogeny
- Using Least-Squares to Construct Distance-Based Phylogenies
- Ultrametric Evolutionary Trees
- The Neighbor-Joining Algorithm
- Character-Based Tree Reconstruction
- The Small Parsimony Problem
- The Large Parsimony Problem
- **Evolutionary Tree Reconstruction in the Modern Era**

1963: Paradigm Shift in Genetic Analysis

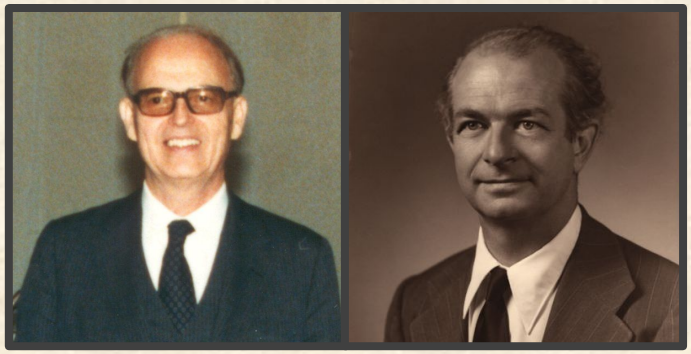


Emile Zuckerkandl

Linus Pauling

From the point of view of hemoglobin structure, it appears that gorilla is just an abnormal human.

1963: Paradigm Shift in Genetic Analysis



Emile Zuckerkandl

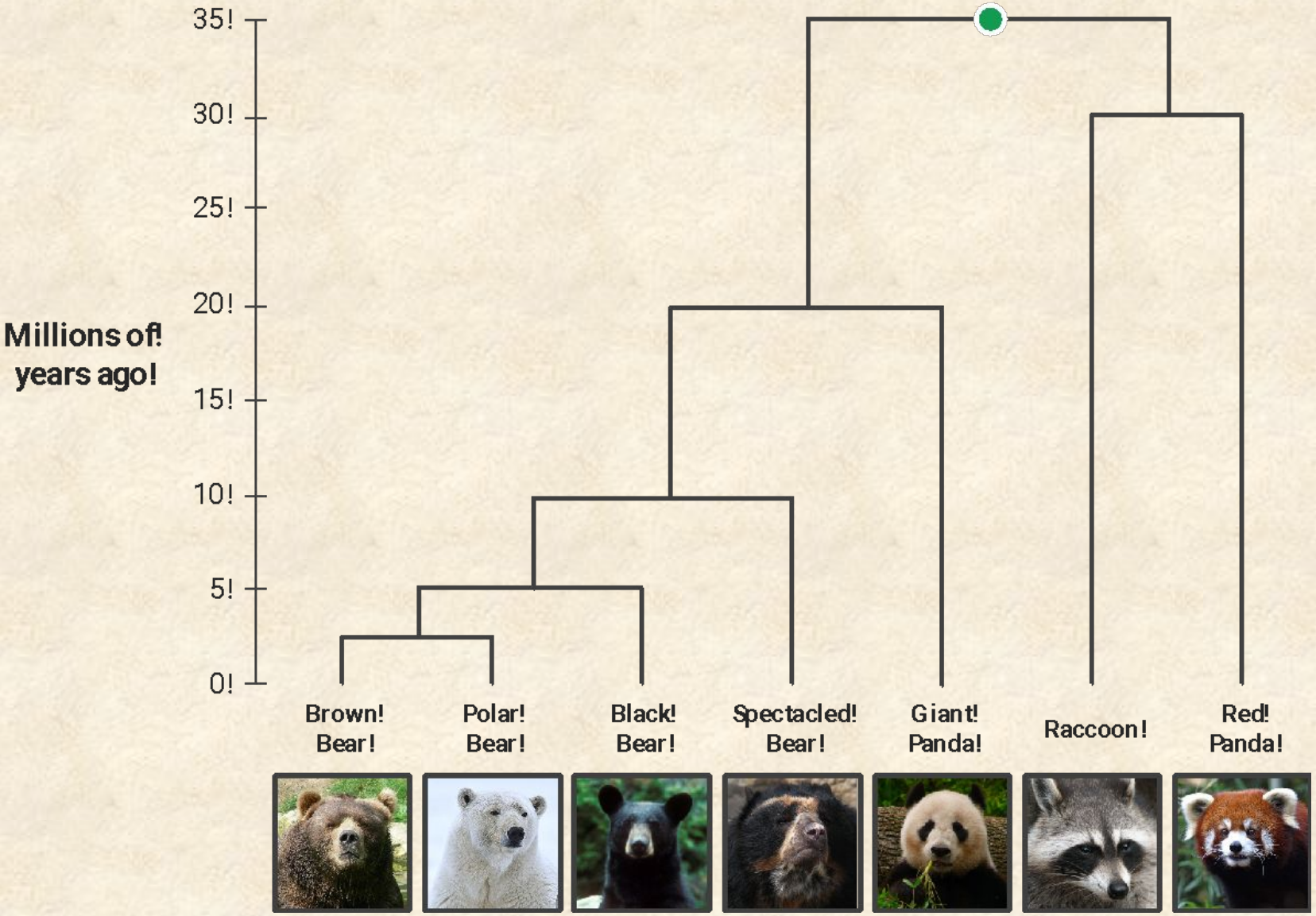
Linus Pauling

From the point of view of hemoglobin structure, it appears that gorilla is just an abnormal human.

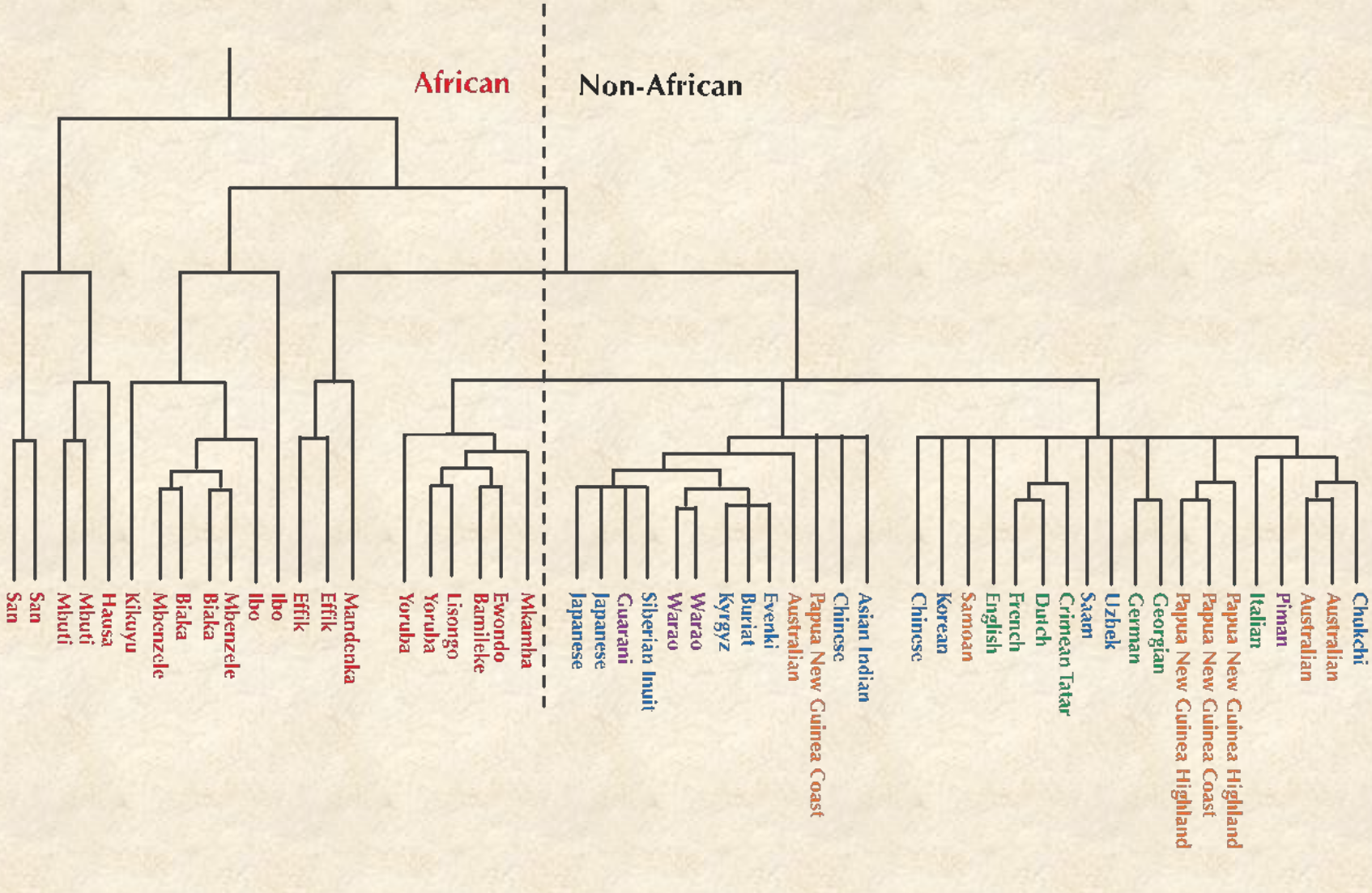
Gaylord Simpson

...that is of course nonsense. What the comparison really indicates is that hemoglobin is a bad choice and has nothing to tell us about attributes, or indeed tells us a lie.

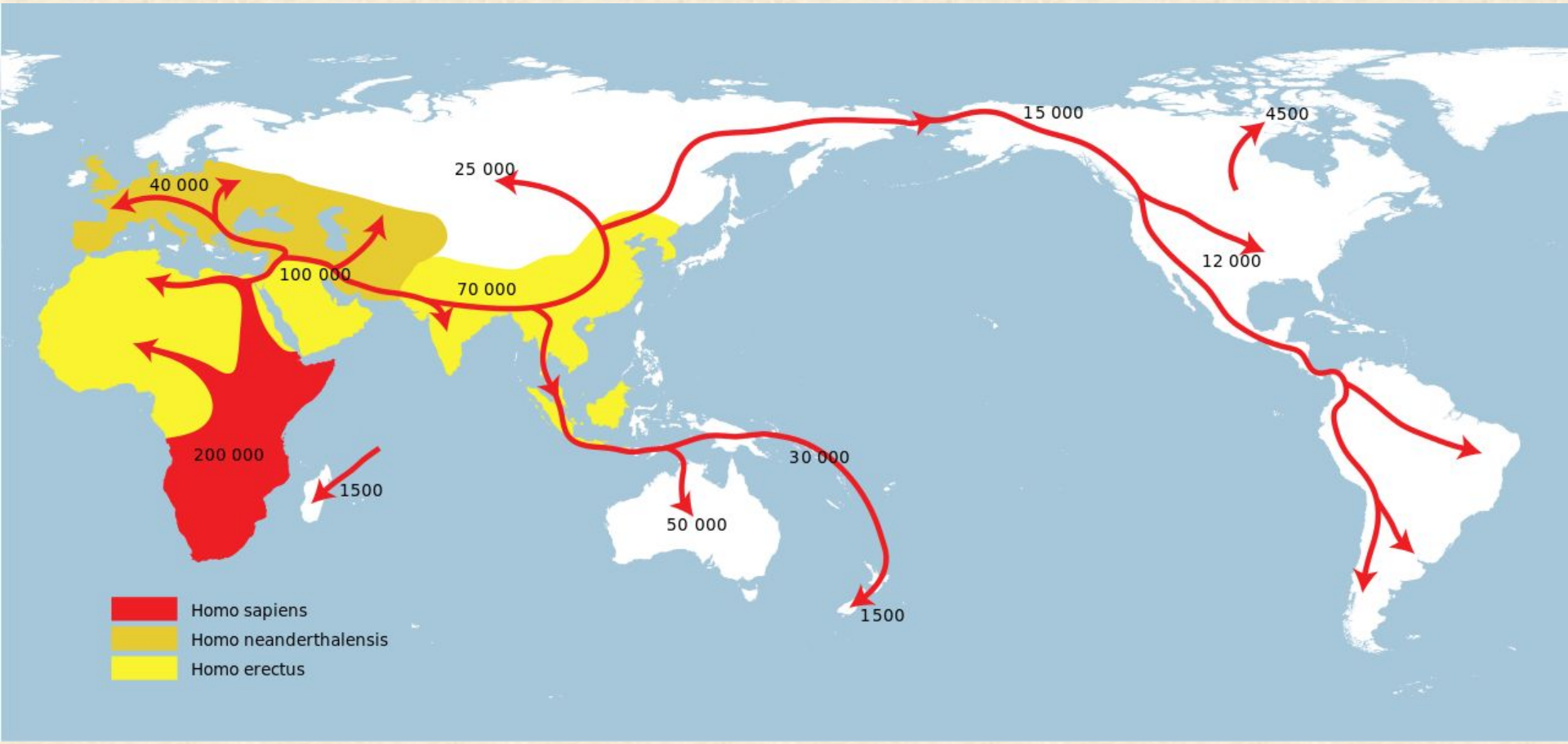
1985: Panda's Pedigree Decoded



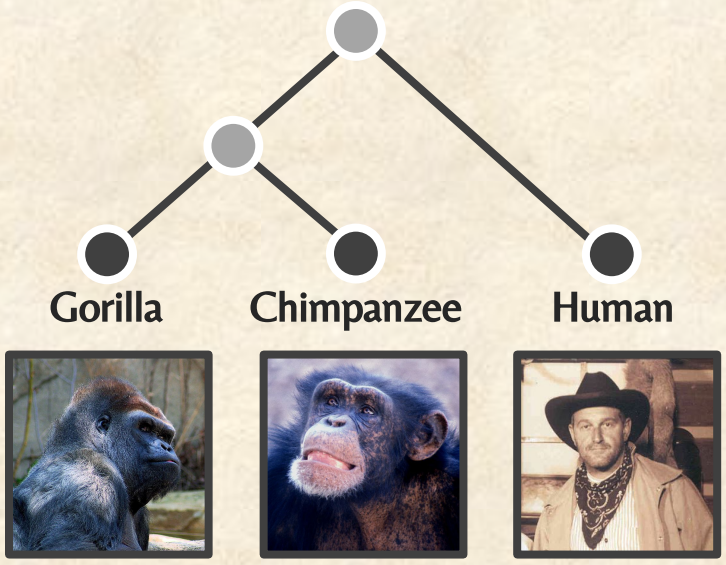
1987: Tracing Human Origins



1987: Tracing Human Origins

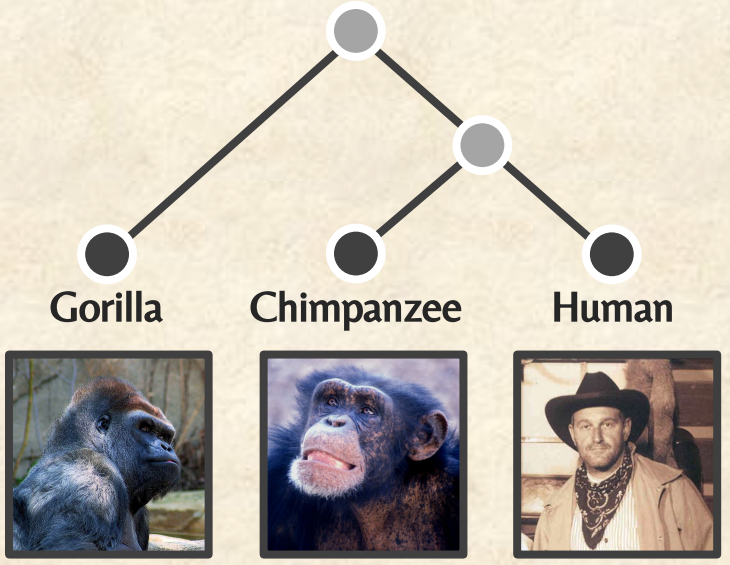


1996: Human-Chimp-Gorilla Ancestry



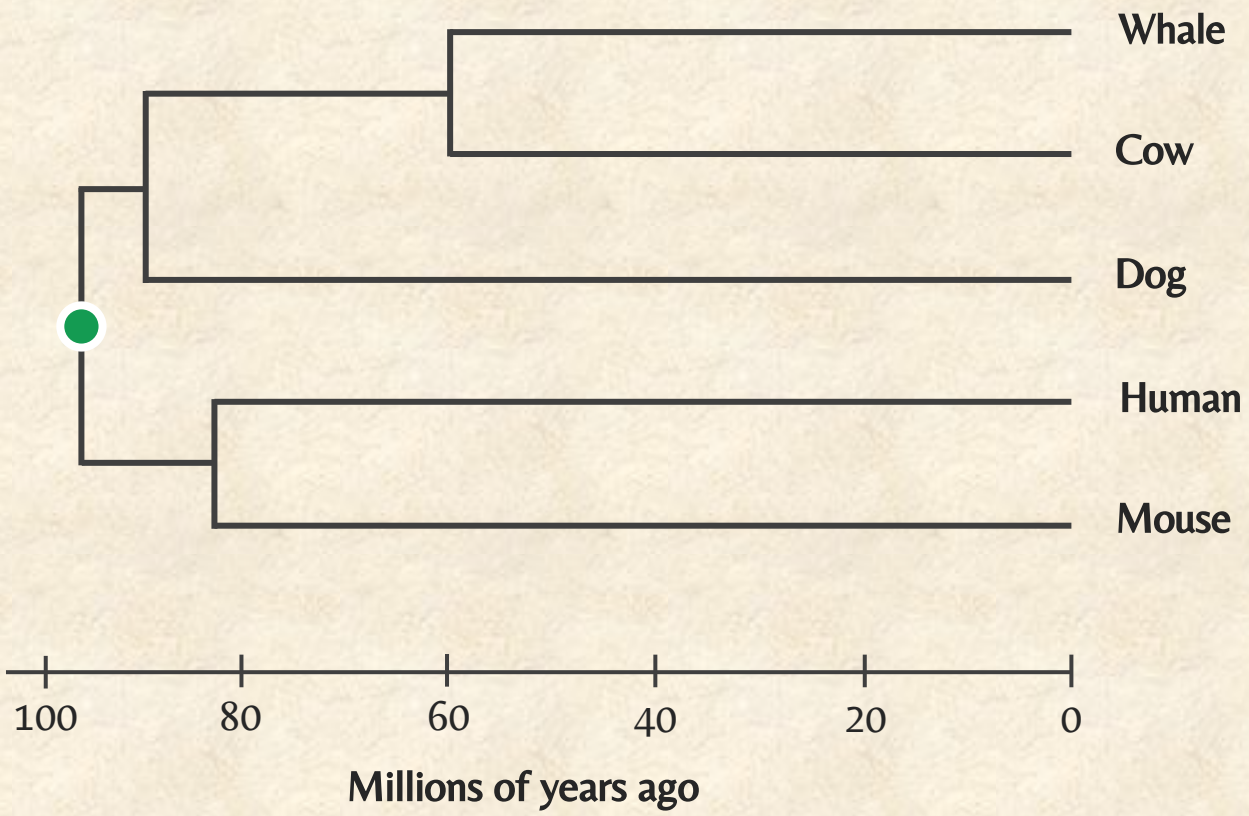
dopamine D4 receptor

VS.

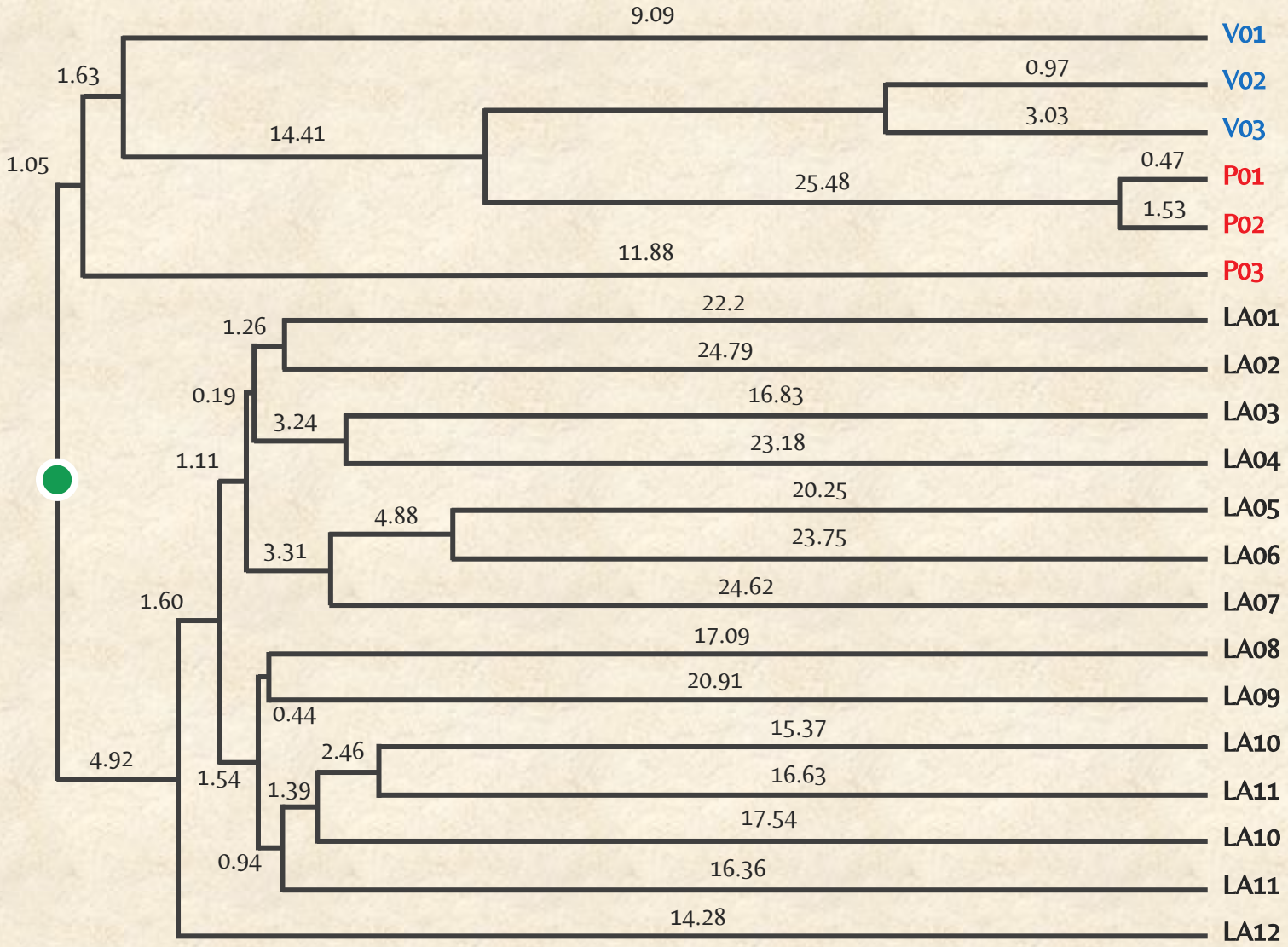


beta-globin

2000s: Humans and Mice Are Relatives



1998: Evolutionary Trees Fight Crime



1988: Evolutionary Trees Fight Crime

Challenge Problem: Given HIV sequences from AIDS patients in Lafayette, construct the evolutionary tree for each of nine HIV proteins. Does each tree support conviction? Reconstruct the ancestral HIV sequences at the internal nodes of the resulting trees.