



Algorithms in Bioinformatics: A Practical Introduction

Phylogenetic Trees Reconstruction



Evolution

- DNA encodes the information of life.
- Living things pass the DNA information to their children.
- Due to **mutation**, the DNA is changed by a little bit.
- After a long time, different species evolved.
- **Phylogenetics** studies the genetic relationship among different species!



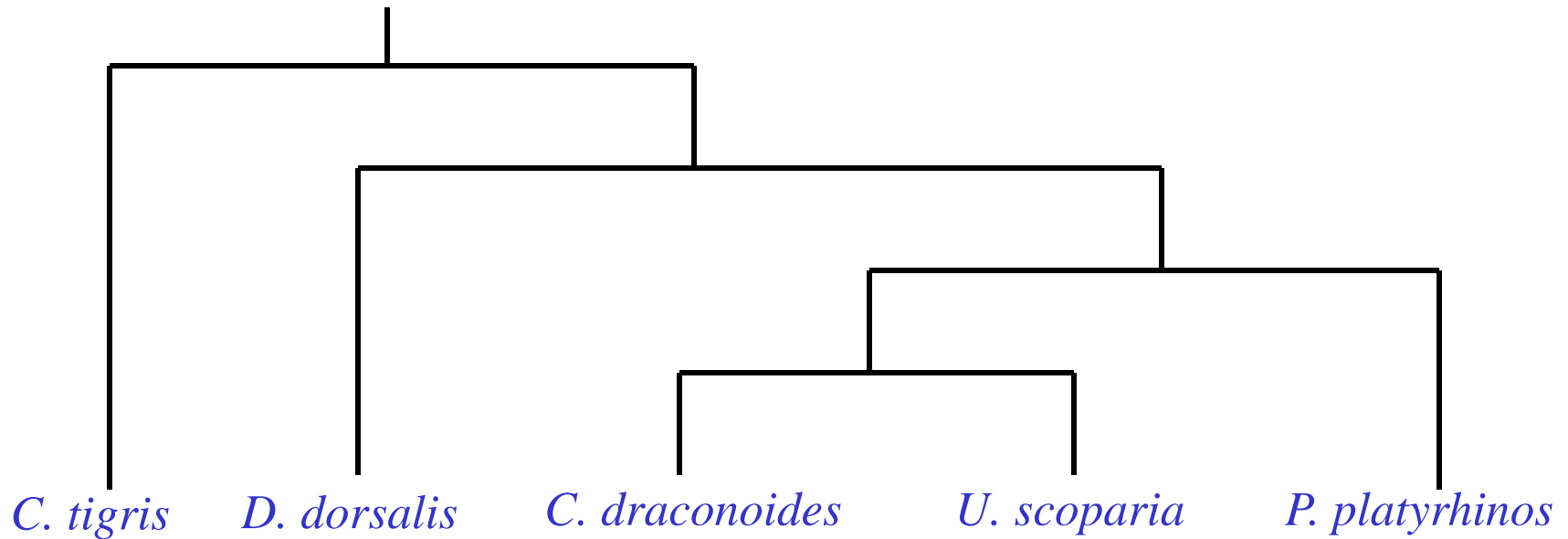
Definition of Phylogeny

- **Phylogeny (or Phylogenetic tree):** reconstruction of the evolutionary history of a set of species.
- Usually, it is a leaf-labeled tree where the internal nodes refer the hypothetical ancestors and the leaves are labeled by the species
- The edges of the tree represent the evolutionary relationships



Example of phylogeny

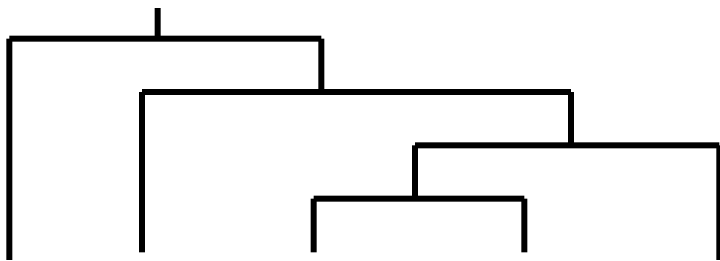
- Phylogeny for lizards





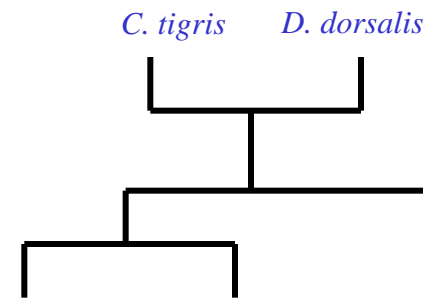
Rooted and Unrooted Tree

- A phylogeny is **rooted**.
- However, since estimating the root is scientifically difficult, the reconstructed tree may be **unrooted**.



C. tigris *D. dorsalis* *C. draconoides* *U. scoparia* *P. platyrhinos*

Rooted



C. draconoides *U. scoparia* *P. platyrhinos*

Unrooted



Rooted a phylogeny by outgroup

- Rooted tree can be reconstructed by systematic biologists based on using **outgroup**.
 - Outgroup is a species which is clearly less related with all other species in the phylogeny
 - E.g. build the phylogenetic tree for human and all bacteria. Then, most probably, human is the outgroup.



Human evolution

- As an example, we can understand the human evolution through phylogenetic study.
- Below, we illustrate the phylogenetic study of
 - mitochondrial Eve
 - Y chromosome Adam



About mitochondrial Eve

- **Human mitochondrial DNA (mtDNA)**
 - Circular double-stranded consisting of 16,500 base pairs
 - Everyone inherits the mtDNA from his/her mother (because mitochondria exists in egg, not in sperm)
 - The pointwise mutation substitution rates of mtDNA is roughly 10 times faster than nuclear DNA
 - Every cell has many mtDNAs.
 - Apparently lack of recombination.
- Therefore, we all inherit the mtDNA from the mother of human (Eve)!

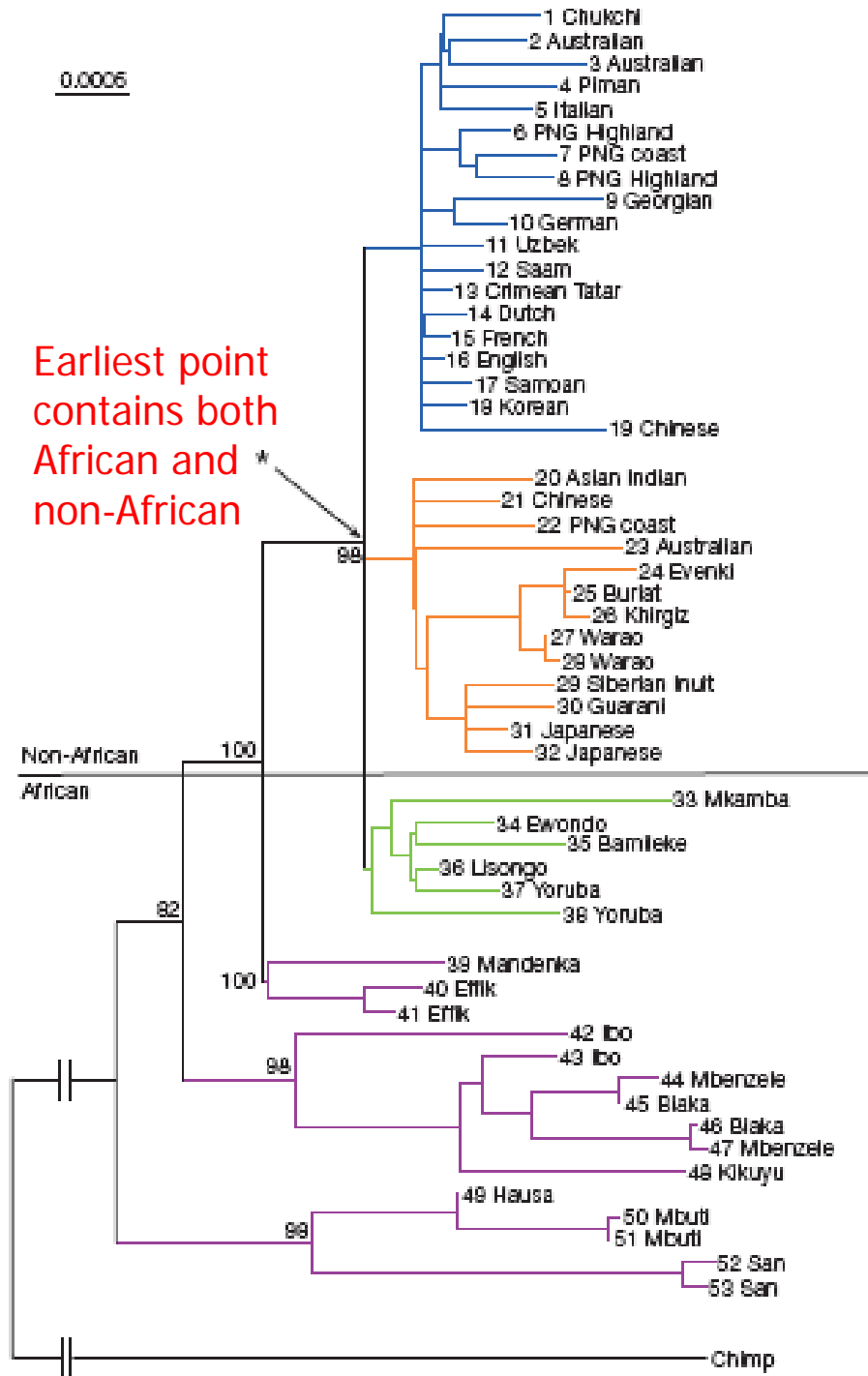


Genetics helps finding the origin of human

- By carrying out a statistical analysis of mtDNAs extracted from the placental tissue of 147 women of different races and from different countries
 - Alan Wilson's group and others construct a phylogenetic tree under the assumption of a **constant molecular clock**.
 - Such phylogenetic tree implies that the common ancestor of modern human appear roughly 100,000-200,000 years ago. (about 143,000 years ago)
- Vigilant, L., Stoneking, M., Harpending, H., Hawkes, K. & Wilson, A. C. African populations and the evolution of human mitochondrial DNA. *Science* 253, 1503-1507 (1991).
- Cann, R. L., Stoneking, M. & Wilson, A. C. Mitochondrial DNA and human evolution. *Nature* 325, 31-36 (1987).

Eve tree

- Tree constructed using neighbour-joining for 53 humans and 1 chimp.
 - chimp is outgroup!
- Complete mtDNA excluding the D-loop.
- M. Ingman, H. Kaessmann, S. Paabo, and U. Gyllensten. Mitochondrial genome variation and the origin of modern humans. Nature, 2000.





About Y chromosome Adam (I)

- Y chromosome is unique to males and it can help to find the father of human.
- However, since the mutation rate of Y chromosome is not as fast as mtDNA,
 - we need more samples to study the evolution of Y chromosome

About Y chromosome Adam (II)

- In Science 1997, at least 93 polymorphic sites have been identified in Y chromosomes of 900 men scanned.
- For one of the site,
 - 15% Khoisan people have A
 - 5-10% of Ethiopians and Sudanese have A
 - Most africans and people outside Africa have T
- This suggested that
 - Khoisan, Ethiopians, and Sudanese (in Africa) may be the closest living relatives to the Y chromosome Adam

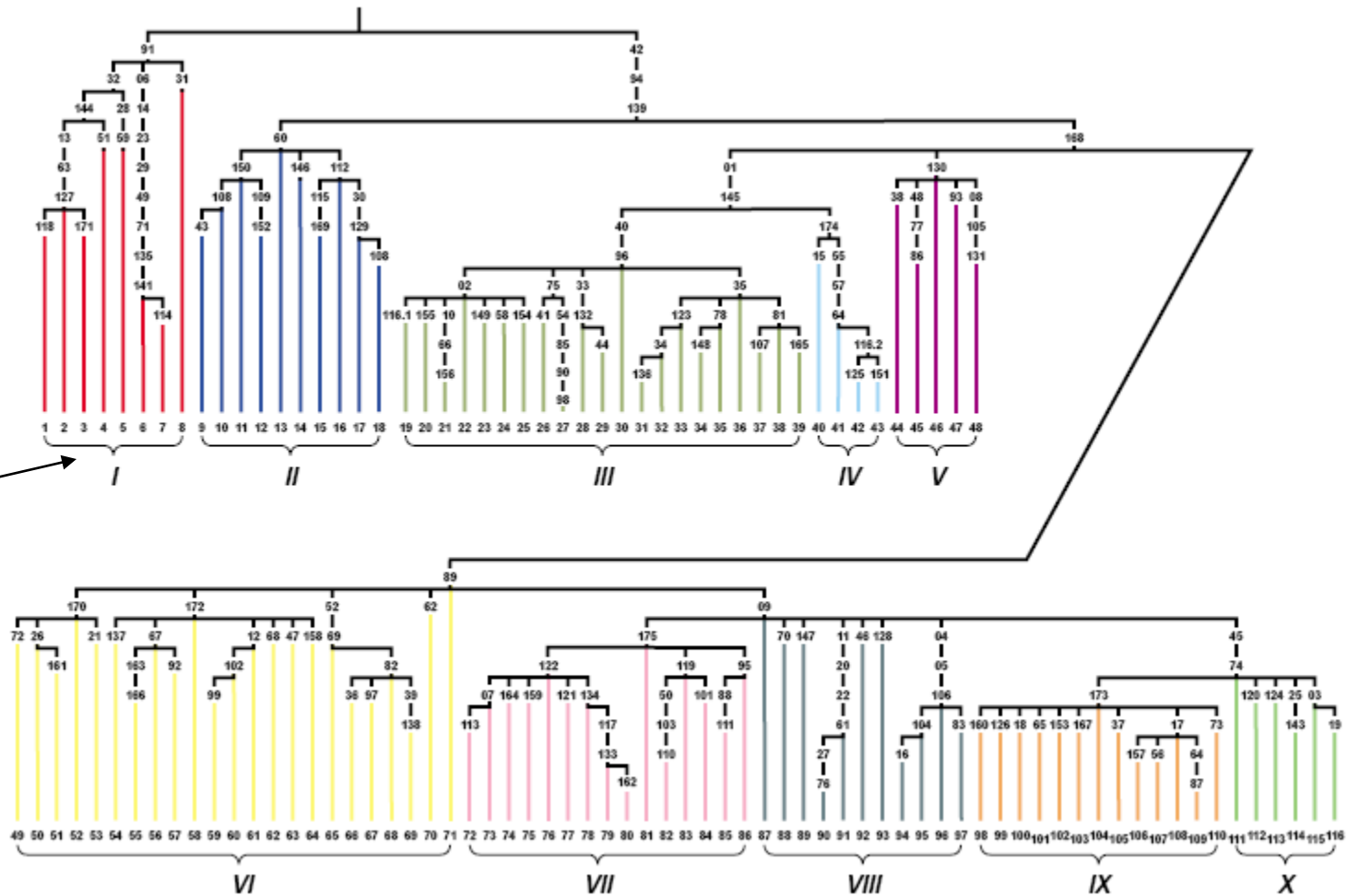




About Y chromosome Adam (III)

- In Nature genetic 2000, by studying Y chromosome of 1062 males from 22 different geographic areas,
 - They identify 167 haplotypes.
 - The common ancestor of the 167 haplotypes is estimated to appear around 59,000 years old.
- Underhill et al. Y chromosome sequence variation and the history of human populations. Nature Genetic, 26:358-361, 2000.

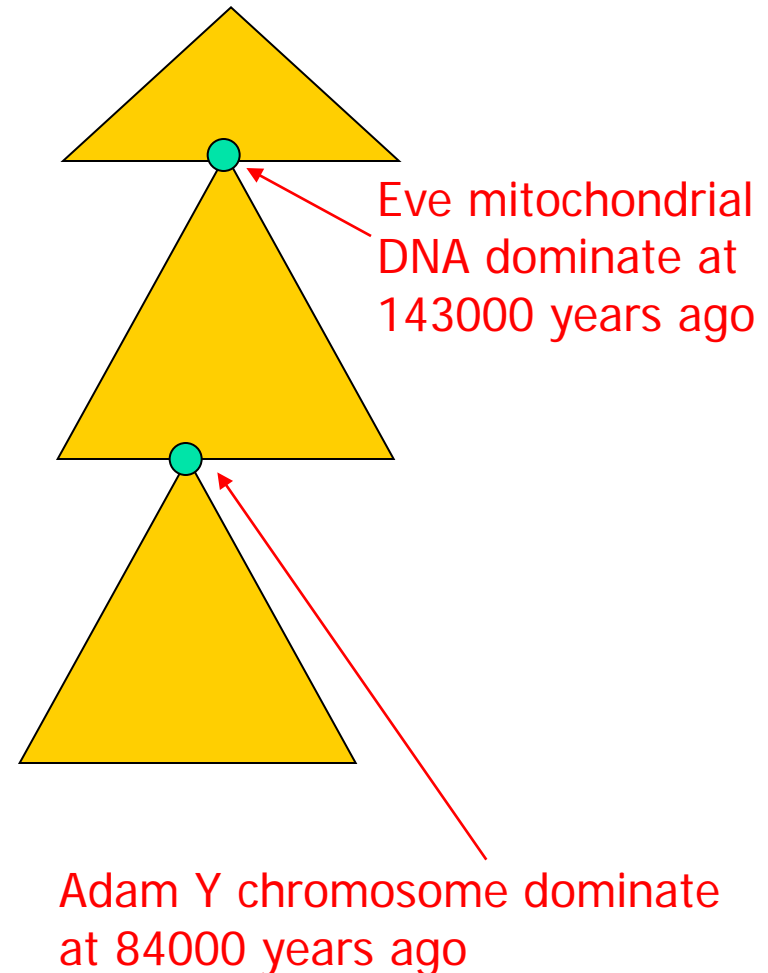
Adam tree



Minority of Africans—mainly Sudanese, Ethiopians and Khoisans

Explanation why Adam and Eve appear in different time

- In around 143,000 years ago,
 - Among different mitochondrial DNA sequences in human population, the Eve mitochondrial DNA had advantages and started to dominate.
 - All other versions of mitochondrial DNA eventually disappear.
- In parallel, different versions of Y chromosomes appear in human population.
 - It took another 84,000 years before the Adam Y chromosome started to take over in the human population.





Applications of Phylogeny

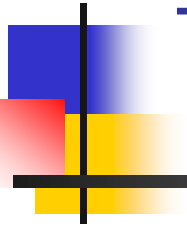
- Apart from understanding the history of life, there are many other applications
 - Understanding rapidly mutating viruses (like HIV)
 - Help to predict protein/RNA structure
 - Help to do multiple sequence alignment
 - Explaining and predicting gene expression
 - Explaining and predicting ligands
 - Help to design enhanced organisms (like rice, wheat)
 - Help to design drug



Computational problem: Phylogeny reconstruction

- Depending on the input, there are two computational problems for reconstructing the phylogeny:
 - Character based
 - Distance Based
- Below, we first describe character based method.

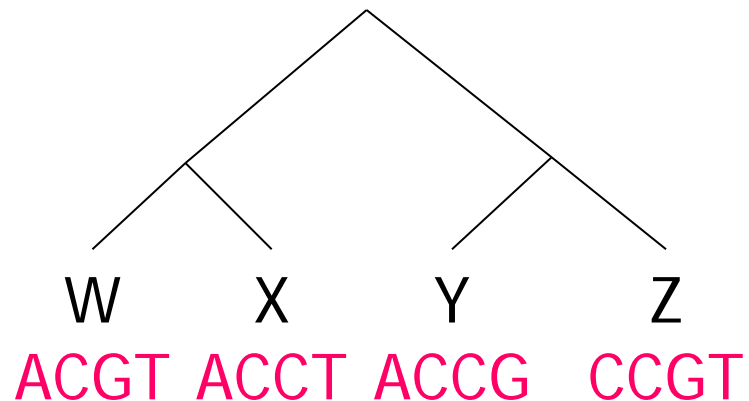
Character Based Phylogenetic Tree Reconstruction



Character Based

- **Input:** each species is described by a set of characters
 - A character can be a base in a specific position in its DNA sequence, the number of eyes of the species, etc
- **Output:** a tree which best explain the input

	1	2	3	4
W	A	C	G	T
X	A	C	C	T
Y	A	C	C	G
Z	C	C	G	T





Outline for Character based methods

- Parsimony
- Compatibility
- Maximum Likelihood

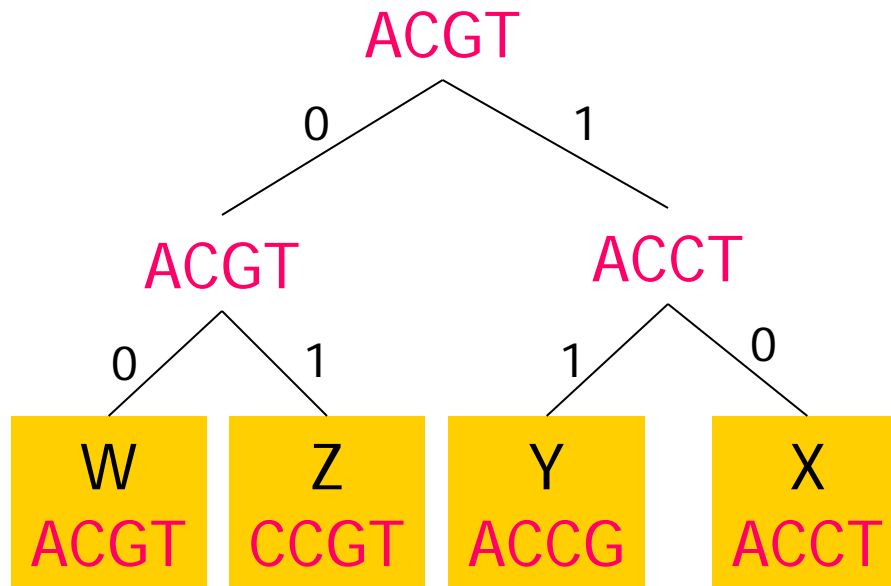


Parsimony

- Most popular method in the systematic biology literature!
- Idea: Build a phylogeny with the fewest point mutations
- Formal Definition:
 - Let S be a set of (DNA or Protein) sequences
 - Denote $H(x, y)$ be the hamming distance between two sequences x and y
 - The **most parsimonious tree** is a tree T leaf-labeled by S and each internal node is assigned a sequence such that $H(T) = \sum_{(x, y) \in E(T)} H(x, y)$ is minimized. Note that $H(T)$ is called the **parsimony length** of T

Example (4 species, each is represented by a sequence of 4 characters)

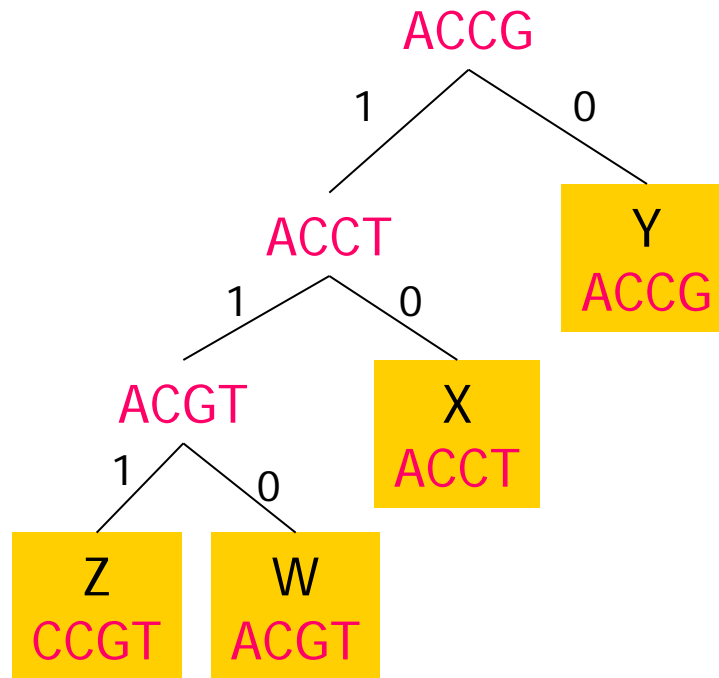
	1	2	3	4
W	A	C	G	T
X	A	C	C	T
Y	A	C	C	G
Z	C	C	G	T



This is the most parsimonious tree
Its parsimony length is 3

Example (4 species, each is represented by a sequence of 4 characters)

	1	2	3	4
W	A	C	G	T
X	A	C	C	T
Y	A	C	C	G
Z	C	C	G	T



This is another most parsimonious tree
Its parsimony length is 3



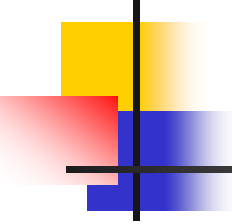
Computational Problems

- **Small Parsimony problem** is to find the parsimony length of a given tree topology
- **Large Parsimony problem** is to find the most parsimonious tree.



Small Parsimony Problem

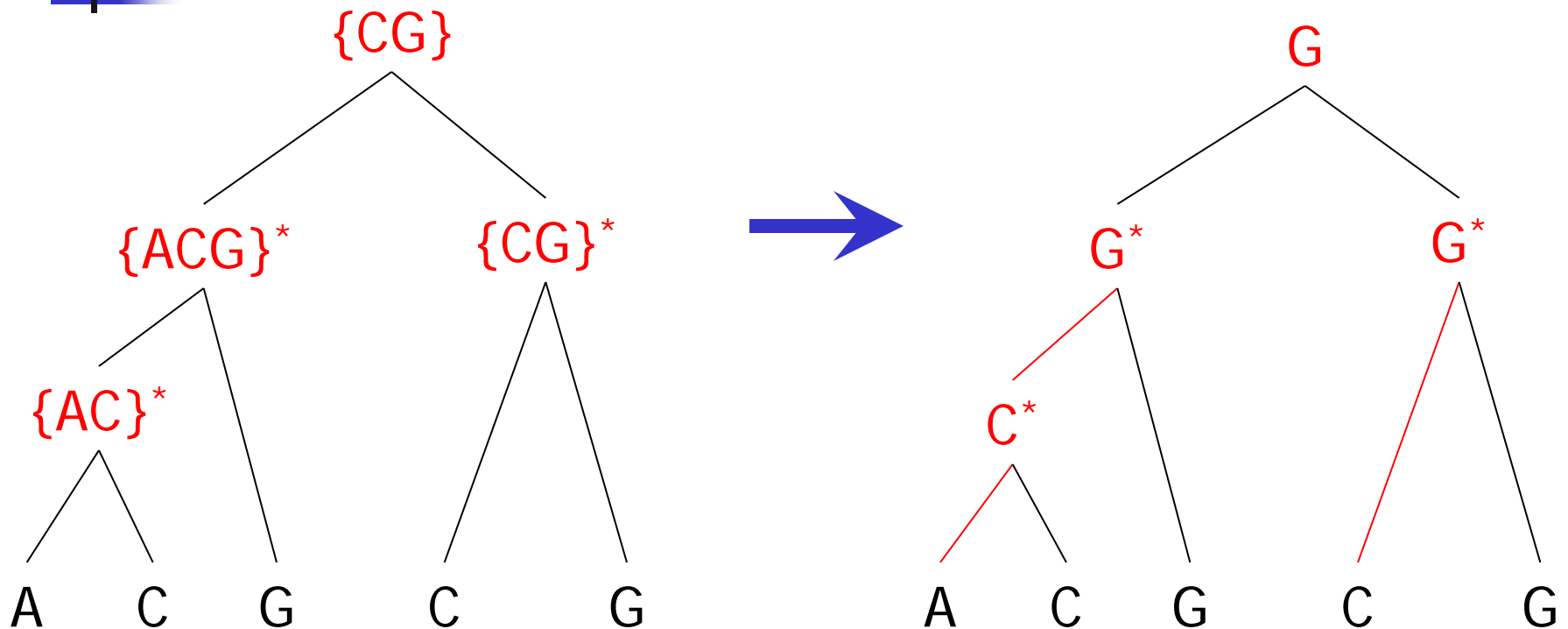
- **Input:** Given a set S of sequences and the topology of a rooted phylogeny T with leaves labeled by S
- **Goal:** Find parsimony length of T
- This problem can be solved in polynomial time using Fitch's algorithm



Simple case: each sequence only has one character

- **Input:** a leaf-labeled tree T where each leaf v is labeled by a single character v_c
- **Output:** a fully-labeled tree which is also the most parsimonious tree of T
 1. For every leaf v , let $S_v = \{v_c\}$.
 2. For every internal node v with children u, w , let
$$S_v = \begin{cases} S_u \cap S_w & \text{if } S_u \cap S_w \neq \Phi \\ S_u \cup S_w & \text{otherwise} \end{cases}$$
 3. For every node v in preorder,
 - Let u be its parent. If $u_c \in S_v$, set $v_c \leftarrow u_c$; otherwise, assign any character in S_v to v_c .

An example



- Each asterisk(*) requires a change in one of the edges to its children
- Time complexity: $O(nk)$ where k is the size of the alphabet (which is 4 for DNA and 20 for protein)



Each sequence has m characters

- Note that the i^{th} character and the j^{th} character are independent for any i and j .
- Thus, this problem can be solved using m instance of the simple case problem.
- Time complexity is $O(mnk)$.



Large Parsimony Problem

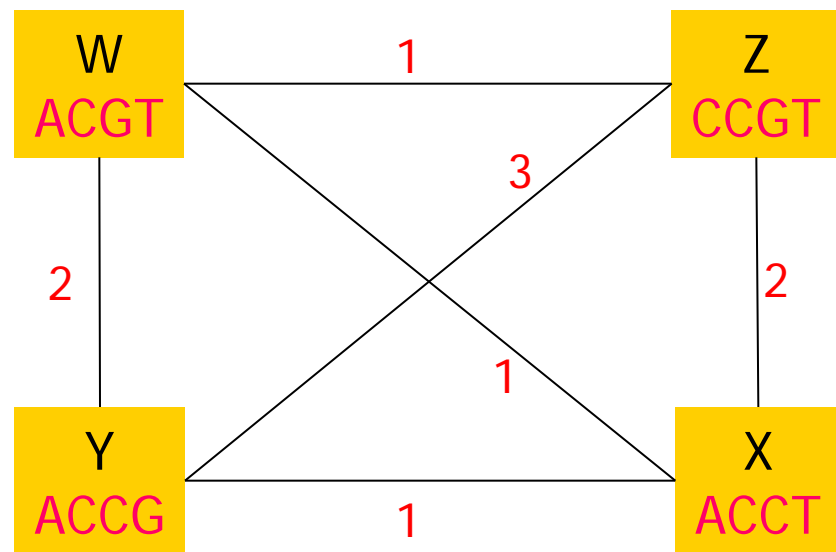
- Input: a set S of sequences
- Output: the most parsimonious tree
 - Large Parsimony Problem is NP-hard
 - Large Parsimony Problem can be 2-approximated in polynomial time

Approximation algorithm

- Given a set S of sequences, define $G(S)$ be a weight complete graph whose nodes are labeled by S and each edge (i, j) has weight $H(i, j)$.

	1	2	3	4
W	A	C	G	T
X	A	C	C	T
Y	A	C	C	G
Z	C	C	G	T

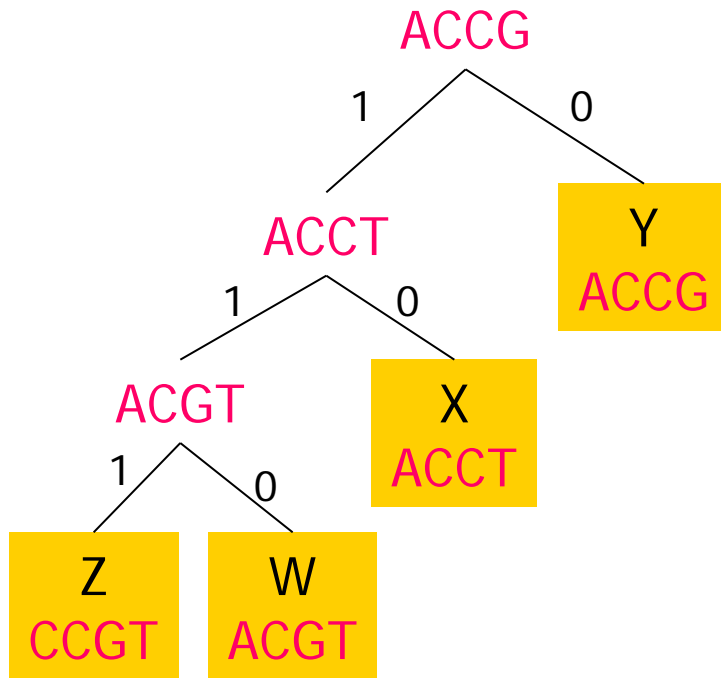
S



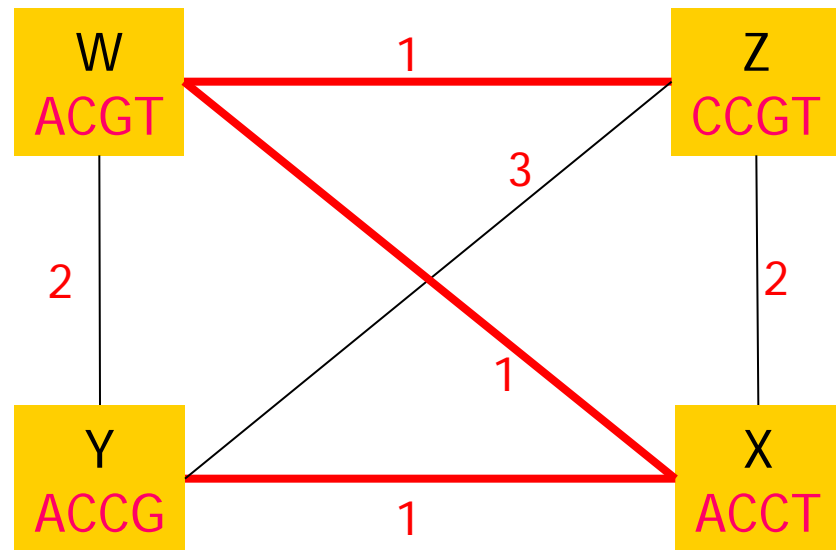
$G(S)$

Approximation algorithm (II)

- Let T be a minimum spanning tree of $G(S)$.



T



$G(S)$



Approximation algorithm (III)

- Theorem: Let T be a minimum spanning tree of $G(S)$. Then, the parsimony length of T is at most twice that of the most parsimonious tree.
 - Proof: Let T^* be the most parsimonious tree.
 - Let C be an Euler cycle of T^* .
 - Let P contains only the nodes of $G(S)$ ordered in the way in which they appear in C .
 - $w(T) \leq w(P) \leq w(C) = 2 w(T^*)$



Final remark for maximum parsimony

- Maximum parsimony is **statistically inconsistent**.
- This means that given long enough sequences, maximum parsimony may not be able to recover the true tree with arbitrarily high probability.



Application of Maximum Parsimony: Predicting evolution of influenza

- Influenza is a fast evolving virus.
- Bush and Fitch et al. show that phylogenetic analyses of the human influenza A (subtype H3) virus can be used to make predictions about the evolutionary course of future human influenza strains.
- The predicted strains of flu virus is included in the vaccine prepared each year to protect against the upcoming influenza season.
 - Bush, R. M., C. A. Bender, et al. (1999) "Predicting the evolution of human influenza A." *Science* 286: 1921-1925.



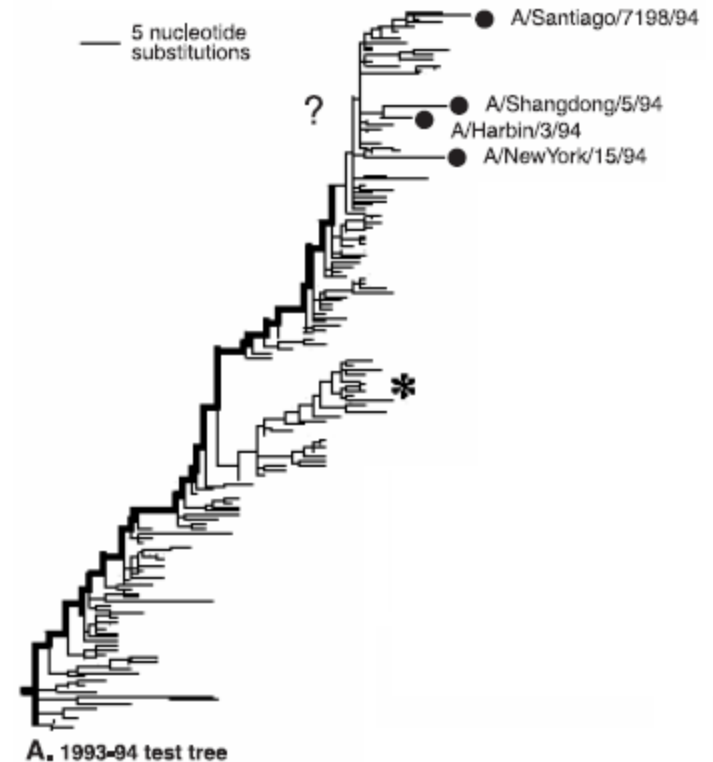
How to build the influenza tree?

- The HA1 domain of the hemagglutinin gene of human influenza A subtype H3
- The HA1 domains are aligned using multiple sequence alignment algorithm. Then, we get the input matrix.
- By maximum parsimony, we build the tree.

Observation from the influenza tree

- The tree shows the evolution of HA1 domain of the hemagglutinin gene of human influenza A subtype H3
 - Build by Maximum Parsimony using isolates from 1983-1994
- There is a selection stress. (The tree is skew.)
 - The bold path shows the single evolutionarily successful linkage.
- At least 18 of the 329 H3 HA1 codons have been under positive selection.

Predictive Isolate: Codon set
A/Shangdong/5/94: Positively selected codons
A/Harbin/3/94: Codons associated with receptor binding
A/Santiago/7198/94: Fastest evolving codons
A/NewYork/15/94: Codons in or near antibody combining sites A and B



Question: What is the trend of the evolution lineage?

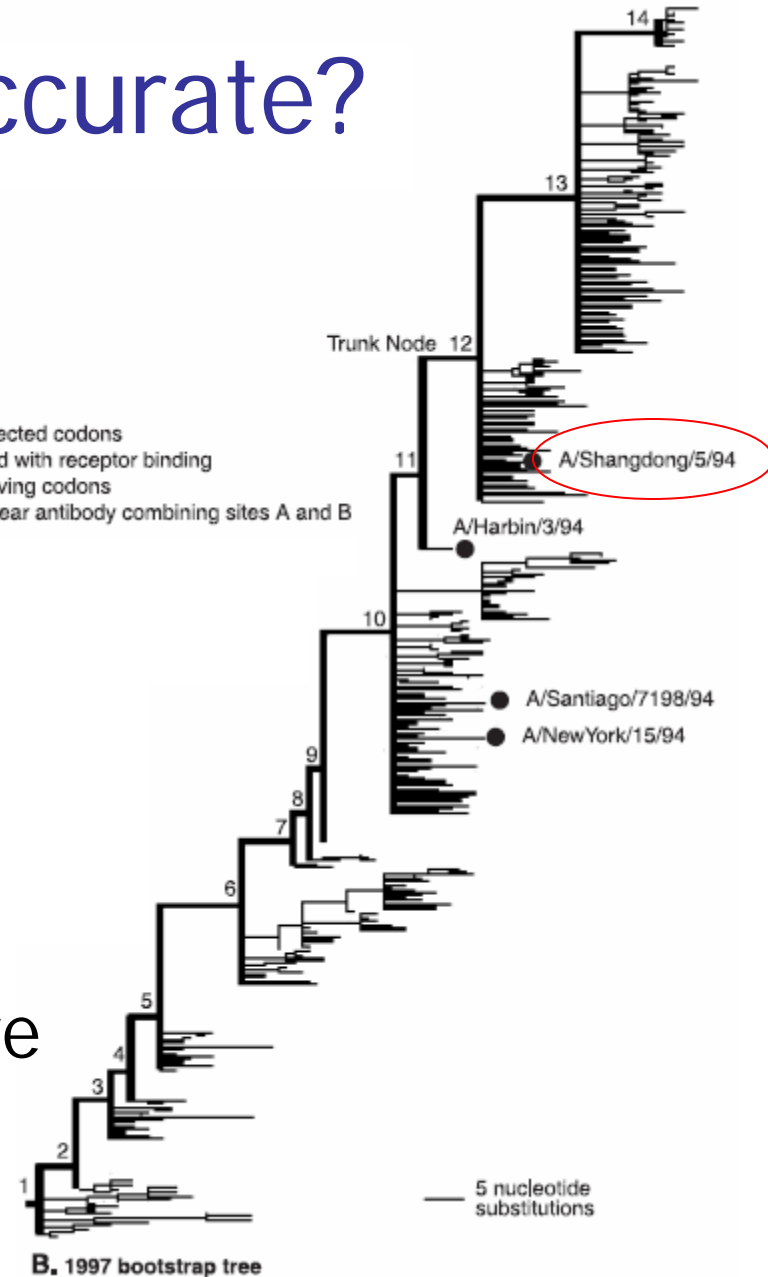
- Hypothesis:
 - If the selective pressure were to evade the host immune response, then viruses sustaining mutations at these 18 codons in the past should have been more fit than other coexisting viruses.
- Based on this idea, the authors predict the future influenza looks similar to **A/Shangdong/5/94**.



Is the prediction accurate?

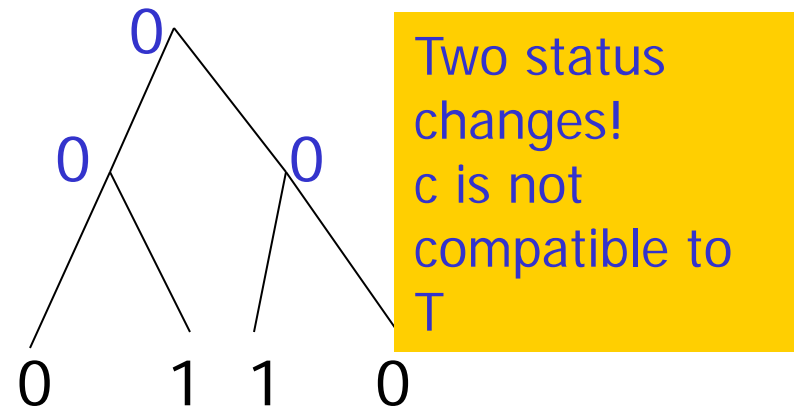
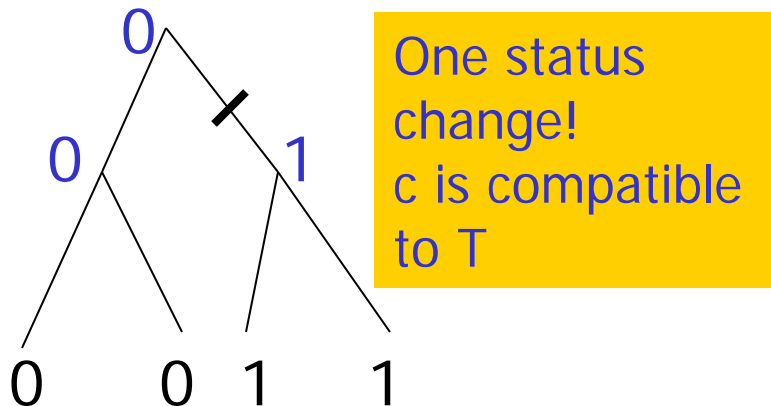
Predictive Isolate: Codon set
A/Shangdong/5/94: Positively selected codons
A/Harbin/3/94: Codons associated with receptor binding
A/Santiago/7198/94: Fastest evolving codons
A/NewYork/15/94: Codons in or near antibody combining sites A and B

- The right tree is reconstructed from the influenza in 1985-1997.
- **A/Shangdong/5/94** is relative more fit to isolates in the future influenza seasons.



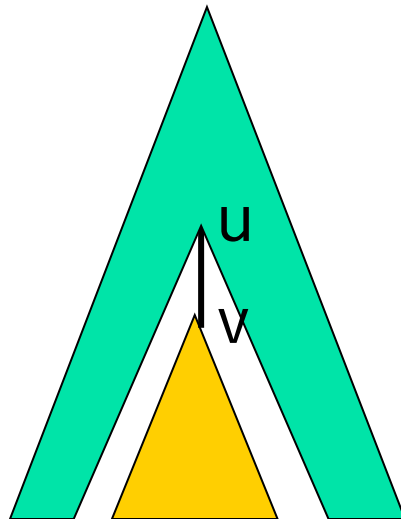
Compatibility

- Compatibility is a simplification of parsimony.
- **Definition:**
 - A binary character c is **compatible** to a leaf-labeled tree T if and only if there exist an assignment of states to the internal nodes of T such that a change of status exists in exactly one edge



More on compatibility

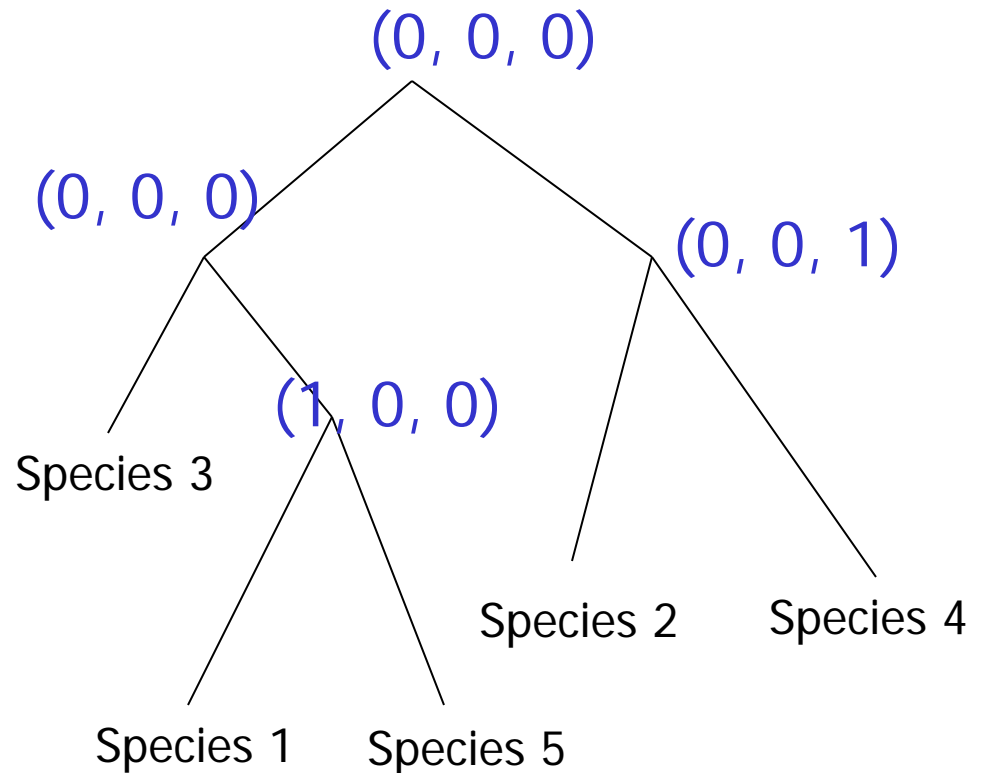
- In fact, if character c is compatible to a tree T , we can identify an edge (u, v) in T so that
 - The leaves in the subtree of v have state s for character c
 - The other leaves have state $(1-s)$ for character c



Example

- Characters 1, 2, and 3 are all compatible!

M	X ₁	X ₂	X ₃
Species 1	1	1	0
Species 2	0	0	1
Species 3	0	0	0
Species 4	0	0	1
Species 5	1	0	0





Perfect phylogeny

- Input: n species, each is characterized by m binary characters.
 - This input can be represented using a binary matrix M with n rows and m columns.
- M admits a **perfect phylogeny** if
 - there exists a rooted tree T for the n species such that all m characters are compatible.



Computational Problems

- Input: Given n species, each characterized by m binary characters. (Represented using a binary matrix M .)
- **Compatibility Problem**
 - Check whether this set of species admits a perfect phylogeny.
- **Perfect Phylogeny Problem (Large Compatibility Problem)**
 - Find a maximum set of characters which admits a perfect phylogeny



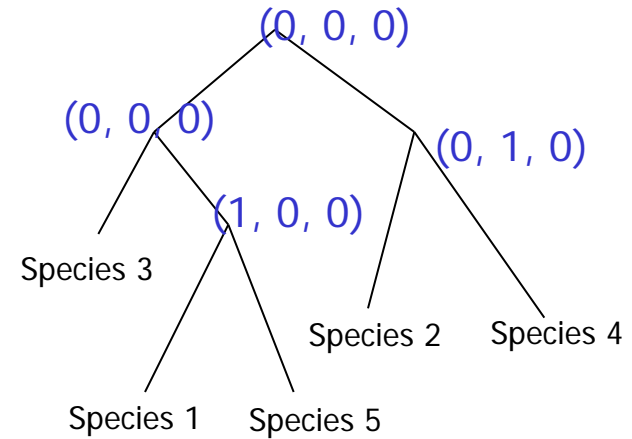
Compatibility problem

- Divide the discussion into two parts:
 1. Check whether M admits a perfect phylogeny
 2. If M admits a perfect phylogeny, recover the tree

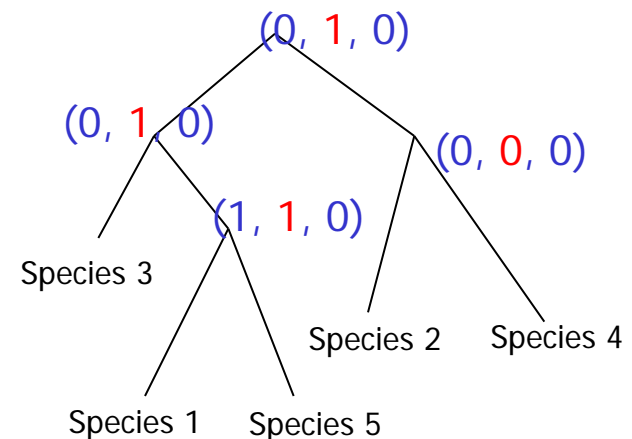
Observation

- If M admits a perfect phylogeny T , after exchanging 0 and 1 in any column, the resulting matrix M' still admits the same perfect phylogeny T .

M	X_1	X_2	X_3
Species 1	1	0	1
Species 2	0	1	0
Species 3	0	0	0
Species 4	0	1	0
Species 5	1	0	0



M'	X_1	X_2	X_3
Species 1	1	1	1
Species 2	0	0	0
Species 3	0	1	0
Species 4	0	0	0
Species 5	1	1	0





Assumption on the input matrix M

- Based on the previous slide, we assume for every column of M ,
 - The number of state 1 $>$ the number of state 0.
- Otherwise, we exchange 0 and 1 and such transformation has no effect on compatibility!

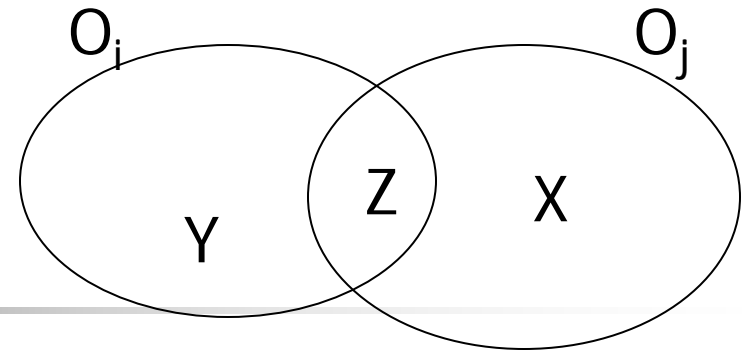


Main lemma

- For every character i , let O_i be the set of species with state 1.
- Characters i and j are **pairwise compatible** if
 - O_i and O_j are disjoint or one of them contains the other.
 - (Note: pairwise compatible \neq compatible!)
- **Lemma**: M admits a perfect phylogeny if and only if for every characters i and j , they are pairwise compatible.



Proof(→)



- Given that M admits a perfect phylogeny
- Note that, for every character i , $|O_i| \leq n/2$.
- Assume that character i and j are not pairwise compatible.
- That is, there exists three species X, Y, Z such that $Y, Z \in O_i$, $X \notin O_i$ and $X, Z \in O_j$, $Y \notin O_j$.
- Since $O_i \cap O_j$ is non-empty, $|O_i \cup O_j| = |O_i| + |O_j| - |O_i \cap O_j| < n$.
 - Thus, there exists a species $W \notin O_i, O_j$.
- By character i , Y and Z are in the same partition in T , while X and W are in another partition
- By character j , X and Z are in the same partition in T and W and Y are in the same partition in T .
- Impossible! We arrived at contradiction!



Proof (←)

- Exercise!



Simple solution for compatibility

- Based on the previous lemma, we get the following algorithm.

Algorithm

- For every characters i and j ,
 - Check whether i and j are pairwise compatible.
 - If no, return “cannot admit a perfect phylogeny”!
- Return “admits a perfect phylogeny”!
- Time complexity: $O(m^2 n)$



Can we get a better algorithm?

- Yes! We can have an $O(mn)$ time algorithm
- Idea:
 - Below, an algorithm is described to check, for all i, j , whether O_i and O_j are disjoint or one of them contains the other
 - If the condition is satisfied, M admits a perfect phylogeny; Otherwise, M does not admit a perfect phylogeny



Step 1

- Relabel the characters so that $|O_i| \geq |O_j|$ if $i < j$

M	X_1	X_2	X_3
Species 1	1	0	1
Species 2	0	1	0
Species 3	0	0	0
Species 4	0	1	0
Species 5	1	0	0

$$\begin{aligned} |O_1| &= 2, \\ |O_2| &= 2, \\ |O_3| &= 1 \end{aligned}$$



Step 2

- For every species i and character j ,
 - If $M_{ij}=1$, let L_{ij} be the biggest $k < j$ such that $M_{ik}=1$.
If such k does not exist, $L_{ij} = -1$
 - If $M_{ij}=0$, let $L_{ij}=0$.

M	X_1	X_2	X_3
Species 1	1	0	1
Species 2	0	1	0
Species 3	0	0	0
Species 4	0	1	0
Species 5	1	0	0

L	X_1	X_2	X_3
Species 1	-1	0	1
Species 2	0	-1	0
Species 3	0	0	0
Species 4	0	-1	0
Species 5	-1	0	0

Technical Lemma^M

- Lemma: For some character j , if there exist two nonzero entries L_{ij} and L_{kj} such that $L_{ij} \neq L_{kj}$,
 - then M does not admit a perfect phylogeny

- Proof:

- Suppose $L_{ij} = x$ and $L_{kj} = x'$. WLOG, $x > x'$.
- By definition, $M_{ij} = M_{kj} = 1$, $M_{ix} = 1$, $M_{kx} = 0$
- Thus, O_j contains species i and species k and O_x contains species i , but not species k . It means that (1) $O_j \cap O_x \neq \emptyset$, (2) O_j is not subset of O_x
- Note that $j > x$. Thus, $|O_x| \geq |O_j|$
- As $k \notin O_x$, O_x should contain some species which does not appear in O_j . So, (3) O_x is not subset of O_j .
- So, by the previous lemma, M does not admit a perfect phylogeny.

	x'	x	...	j		
...					...	
i				1	1	...
...						...
k				0	1	...
...						...

	x'	x	...	j		
...					...	
i					x	...
...						...
k					x'	...
...						...



Step 3

- For every character j , check if there exist i and k such that $L_{ij} \neq L_{kj}$ and both L_{ij} and L_{kj} are nonzero.
- If yes, return “does not admit a perfect phylogeny”.
- Otherwise, “admits a perfect phylogeny”.

L	X_1	X_2	X_3
Species 1	-1	0	1
Species 2	0	-1	0
Species 3	0	0	0
Species 4	0	-1	0
Species 5	-1	0	0

For every character j (column j), we can't find two nonzero positive entries which are different. So, for all i, j , O_i and O_j are disjoint or one of them contains the other



Time complexity

- Step 1 takes $O(mn)$ time (by radix sort)
- Steps 2 and 3 can be computed in $O(mn)$ time!
- Thus, we can decide whether M admits a perfect phylogeny or not in $O(mn)$ time.



Tree reconstruction

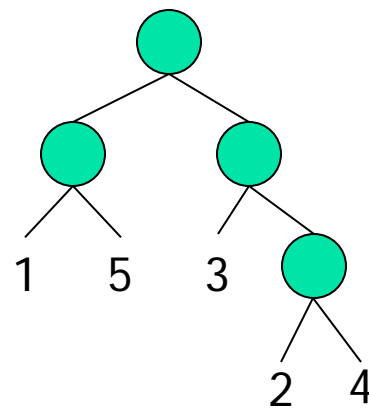
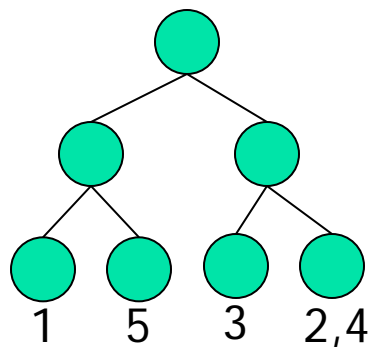
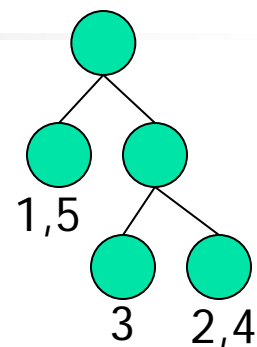
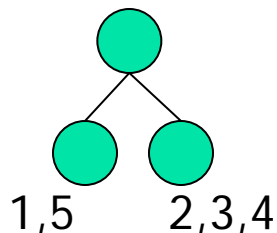
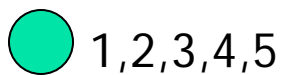
Algorithm

Input: A character-state matrix M with $O_i \geq O_j$ for $1 \leq i < j \leq n$

- Let T be a tree containing the single root node r . $N(r) = \{1, \dots, n\}$
- For every character j where $j=1$ to m
 - Find a leaf $v \in T$ such that
 - $N(v)$ can be partitioned into two non-empty sets N_0 and N_1 where $N_s = \{x \in N(v) \mid \text{character } j \text{ of species } x \text{ is of state } s\}$ for $s=0,1$
 - /* Note: we can only split one leaf v */
 - Create two children v_0 and v_1 for v
 - Set $N(v_0) = N_0$, $N(v_1) = N_1$
 - Set $N(v) = \Phi$
- For every leaf v s.t. $N(v)$ is nonempty,
 - If $|N(v)| > 1$, let the species in $N(v)$ be the children of v
 - If $|N(v)| = 1$, leaf v represents the species in $N(v)$

Example

M	X ₁	X ₂	X ₃
Species 1	1	0	1
Species 2	0	1	0
Species 3	0	0	0
Species 4	0	1	0
Species 5	1	0	0





Time analysis

- For every character j , it takes $O(n)$ time to identify a node and to split the node
- Thus, the total time is $O(nm)$



Large Compatibility Problem

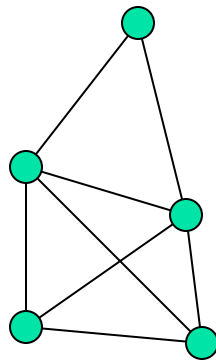
- Find the maximum set of characters which admits a perfect phylogeny!
- This problem is NP-hard!

- We discuss how to solve Large Compatibility Problem by transforming it to CLIQUE Problem.

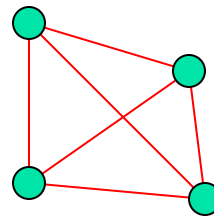


CLIQUE Problem

- Given a graph G , the problem tries to find the maximum size subgraph H such that H is a complete graph.



G



H

- Note: this is an NP-complete problem



Large Compatibility Problem vs CLIQUE Problem

- Given an instance of M , define a graph G where
 - Each vertex i in G corresponds to a character in M
 - (i, j) is an edge in G if i and j are pairwise compatible.
- Note that
 - G can be constructed in polynomial time
 - Note that G contains a clique of size B if and only if M contains a subset of compatible characters whose size is B .
- Thus, we transform the large compatibility problem to a CLIQUE problem.



Algorithm for solving large compatibility problem

Input: M

1. Obtain G based on M
 2. Find the maximum clique in G
 3. Then, recover the maximum subset of compatible characters
 4. Based on the tree construction algorithm in slide 49, recover the phylogeny
- The bottleneck is step 2. So, the time complexity is exponential.



Compatibility for characters with k possible states

- We can generalize the problem when the characters are not binary
- **Definition:**
 - A character c with k possible states is compatible to a leaf-labeled tree T if and only if there exist an assignment of states to the internal nodes of T such that the total number of state changes is exactly $k-1$
- **Result:**
 - **Compatibility Problem**
 - When the number of states is constant, polynomial time algorithm is still feasible
 - When the number of states is variable, NP-complete
 - **Large Compatibility Problem**
 - NP-complete



Maximum Likelihood

- Given a set of data D , Maximum likelihood tries to find a model M such that
 - $\Pr(D|M)$ is maximized!



What is a model?

- A model consists of
 - A rooted tree which models the evolution relationship
 - Every edge is associated with a stochastic model of evolution
- Usually, it is assume that
 - the characters evolve **identically** and **independently**
 - Also, the tree has the **markov** property. That is, the evolution occurs at one subtree is independent to the other parts of the tree.
- Example of models:
 - Cavender-Felsenstein model (also called Cavender-Farris model)
 - Jukes-Cantor model



Cavender-Felsenstein Model (I)

- Simplest possible markov model of evolution
- Assume each character has only two states
- The model consist of
 - the topology T
 - a mutation probability $p(e)$ for each edge e in T
- Assumption:
 - For every $e=(u,v)$ in T , $0 < p_i(e) < 0.5$

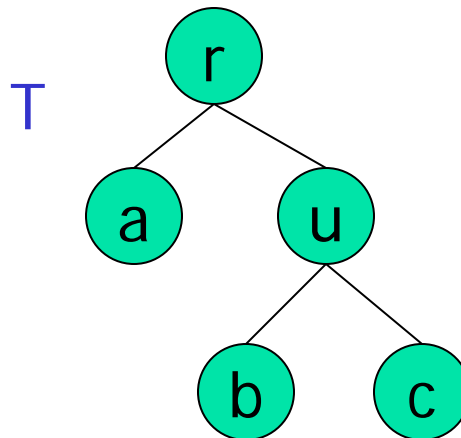
	$u=0$	$u=1$
$v=0$	$\Pr(u=0 v=0)=1-p_i(e)$	$\Pr(u=1 v=0)=p_i(e)$
$v=1$	$\Pr(u=0 v=1)=p_i(e)$	$\Pr(u=1 v=1)=1-p_i(e)$

- $\Pr(u|v) = \Pr(v|u)$
- For the root r , $\Pr(r=0)=\Pr(r=1)=0.5$

Cavender-Felsenstein Model (II)

- Consider 3 species a, b, and c
- For a particular character i, assume the model says that the tree topology is T and the mutation probability for every edge e is $p_i(e)$
- Suppose the data D_i says: $a_i=1$, $b_i=1$, $c_i=0$
- Then, probability that the data is D_i given that the model is (T, p_i) , $\Pr(D_i|T, p_i)$, equals

$$\sum_{\substack{k=0,1 \\ j=0,1}} \Pr(r_i = k) \Pr(a_i = 1 | r_i = k) \Pr(u_i = j | r_i = k) \Pr(b_i = 1 | u_i = j) \Pr(c_i = 0 | u_i = j)$$



Cavender-Felsenstein Model (III)

- Consider m species each is characterized by n characters
- Let the data be $D = D_1 \cup \dots \cup D_n$
- The model consists of the tree topology T and the mutation probability p_i for character i
- $\Pr(D|T, p_e \ e \in T) = \prod_{i=1..n} \Pr(D_i|T, p_e \ e \in T)$



Computational Problems

- Likelihood of a model
 - Given the model M , for any data D , try to compute $\Pr(D|M)$
- Find model with maximum likelihood
 - Given data D , try to find a model M which maximizes $\Pr(D|M)$!



Likelihood of a model

- **Input:**
 - Data D : m species where each species is characterized by n character
 - Model $M = (T, p_e \ e \in T)$
- **Aim:** Compute $\Pr(D|M)$

- $\Pr(D|M)$ can be computed using the formula we stated before.
 - However, it takes exponential time.
- Can we do it better?
 - Yes! By defining the likelihood recursively and compute the value using dynamic programming.



Recursive Definition

- For a particular character i , let $L_i(v, s)$ be the likelihood of the subtree rooted at v , given that character i has state s .
- For every leaf v and state s ,
 - $L_i(v, s) = 1$ if $v_i = s$; 0, otherwise.
- Traverse the tree in postorder, for every internal node v with children, says, u and w ,
 - $$L_i(v, s) = \left[\sum_{y=0,1} L_i(u, y) \Pr(u_i = y \mid v_i = s) \right] \left[\sum_{y=0,1} L_i(w, y) \Pr(w_i = y \mid v_i = s) \right]$$



Time complexity

- Finally, for the root, we have

$$L = \prod_{i=1..m} \left[\sum_{s=1,2} \left(\frac{1}{2} L_i(\text{root}, s) \right) \right]$$

- **Time Complexity:**
 - For every node v and every state s ,
 - $L_i(v, s)$ can be computed in $O(1)$ time according to the recurrence.
 - Since there are n nodes and m characters, all $L_i(v, s)$ can be computed in $O(mn)$ time.
 - For L , it can be computed in $O(m)$ time.
 - In total, Likelihood of a tree can be computed in $O(mn)$ time.



Find model using maximum likelihood

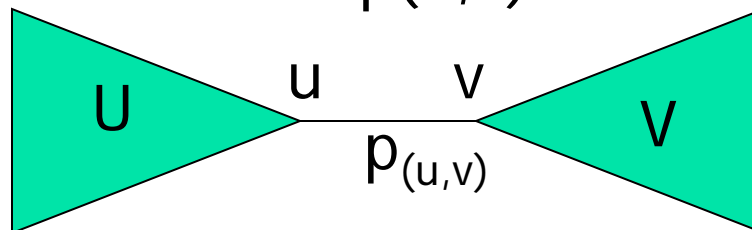
- **Input:**
 - Data D : m species where each species is characterized by n character
- **Aim:** Find $M = (T, p_e \ e \in T)$ which maximizes $\Pr(D|M)$
- This problem is NP-hard.
- Solution: uses heuristic to get close to optima (like DNAmI)

Estimating the weight of an edge

- Let $L(u=s, U)$ and $L(v=s, V)$ be the maximum likelihood score of U and V with the state of the root equals s .
- We would like to find $p_{(u,v)}$ of the edge (u,v) which maximize the likelihood of the combined tree.
- Note that the likelihood of the combined tree is

$$L = \prod_i \sum_{h, h' \in \{0,1\}} L_i(U, h) L_i(V, h') \Pr(u_i = h | v_i = h')$$

- We would like to find $p(u,v)$ which maximizes L .





Find $p_{(u,v)}$ which maximizes L (I)

$$\begin{aligned} L &= \prod_i \sum_{h,h' \in \{0,1\}} L_i(U, h) L_i(V, h') \Pr(u_i = h \mid v_i = h') \\ &= \prod_i p_{(u,v)} \left(\sum_{h \in \{0,1\}} L_i(U, h) L_i(V, h) \right) + (1 - p_{(u,v)}) \left(\sum_{h \in \{0,1\}} L_i(U, h) L_i(V, 1-h) \right) \\ &= \prod_i p_{(u,v)} A_i + (1 - p_{(u,v)}) B_i \end{aligned}$$

$$\ln L = \sum_i p A_i + (1 - p) B_i$$

$$\frac{d \ln L}{d p} = \sum_i \frac{A_i - B_i}{p A_i + (1 - p) B_i} = 0$$



Find $p_{(u,v)}$ which maximizes L (II)

$$m = \sum_i \frac{B_i + p(A_i - B_i)}{pA_i + (1-p)B_i} = \sum_i \frac{B_i}{pA_i + (1-p)B_i}$$

$$p = \frac{1}{m} \sum_i \frac{B_i p}{pA_i + (1-p)B_i}$$

By iterating the following equation, we can approximate $p_{(u,v)}$.

$$p^{(k+1)} = \frac{1}{m} \sum_i \frac{B_i p^{(k)}}{p^{(k)} A_i + (1 - p^{(k)}) B_i}$$



DNAmI

Algorithm DNAmI

- Let $S = \{s_1, s_2, \dots, s_n\}$ be the set of taxa.
- Build the tree T for species $\{s_1, s_2\}$
- For $k = 3$ to n
 - Among all $(2k-5)$ ways to insert s_k into T ,
 - we choose the way with the best likelihood.
 - If $k \geq 4$,
 - While there exists nearest neighbor interchange (NNI) which can improve the likelihood of T ,
 - We perform such NNI



Final remark for Maximum Likelihood

- For the Cavender-Felsenstein model, maximum likelihood is statistically consistent.

Distance Based Phylogenetic Tree Reconstruction





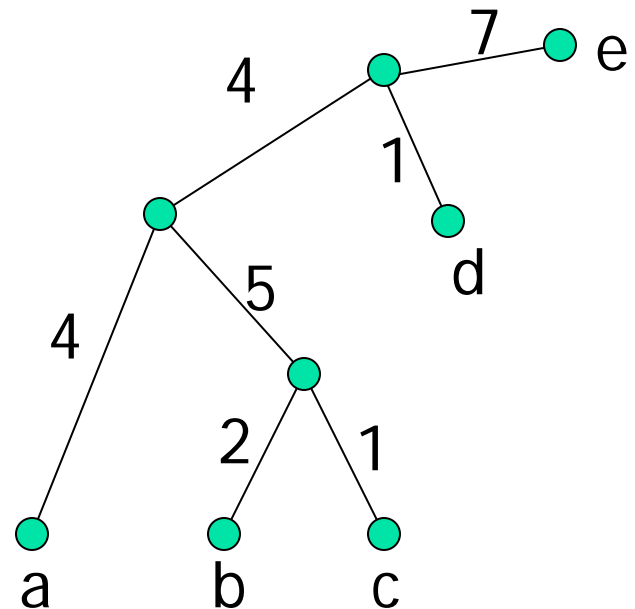
Distance between species

- In character based methods, we try to minimize the number of mutations.
- Intuitively, species which look similar should be evolutionary more related.
- This motivates us to define the **distance** between two species to be the number of mutations need to change one species to another.
- In this lecture, we try to construct a phylogeny using the distance information among species.

Distance Based

- **Input:** a distance matrix M satisfying some constraints
- **Output:** a tree of degree 3 which is consistent with the distance matrix

	a	b	c	d	e
a	0	11	10	9	15
b	11	0	3	12	18
c	10	3	0	11	17
d	9	12	11	0	8
e	15	18	17	8	0





Constraints for the distance matrix M

- There are three assumptions for M
 1. M should satisfy the metric space
 2. M is an additive metric
 3. M is ultrametric (optional)



Metric space

- In the following discussion, we assume that the distance between species satisfy the **metric space**. That is,
 - a distance metric M which satisfies
 - $M_{ij} = M_{ji} \geq 0, M_{ii} = 0$
 - $M_{ij} + M_{jk} \geq M_{ik}$ [triangle inequality]

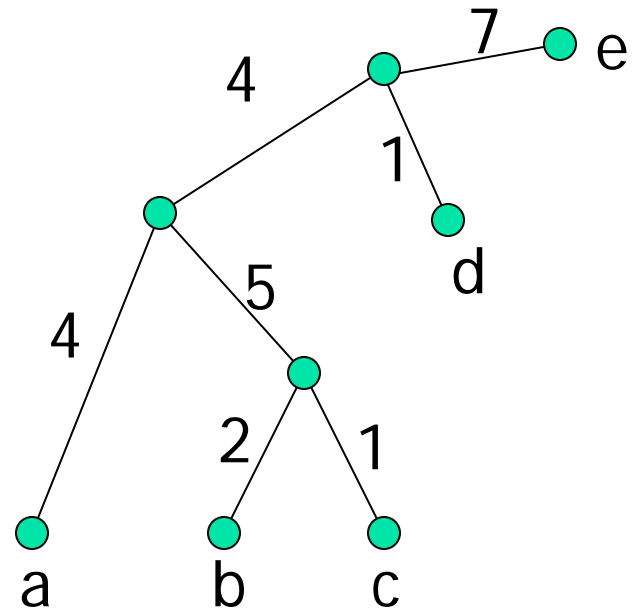


Additive metric

- Let S be a set of species
- Let M be the distance matrix for S
- If there exists a rooted tree T where
 - every edge has a positive weight and every leaf is labeled by a distinct species in S ; and
 - for every $i, j \in S$, M_{ij} = the sum of the edge weights along the path from i to j .
- Then, M is called an **additive metric**
- The corresponding tree T is called **additive tree**

Additive Metric Example

	a	b	c	d	e
a	0	11	10	9	15
b	11	0	3	12	18
c	10	3	0	11	17
d	9	12	11	0	8
e	15	18	17	8	0



- Don't know the root! We can only build an unrooted phylogeny.

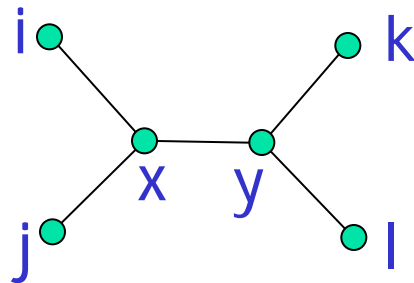


Properties of additive metric

- Buneman's 4-point condition
 - M is additive if and only if
 - for any four species in S , we can label them i, j, k, l such that $M_{ik} + M_{jl} = M_{il} + M_{jk} \geq M_{ij} + M_{kl}$

Proof for the 4-point condition

- **Proof of forward direction:** If M is additive, there exists an additive tree T for S .
- Consider the subtree for the 4 species i, j, k, l . WLOG, the subtree is as follows.



- It can be easily verify that
 - $M_{ik} + M_{jl} = M_{il} + M_{jk} \geq M_{ij} + M_{kl}$
- We will not present the proof for the backward direction.



Criteria for checking if M is additive or not

- Based on the 4-point condition, we can check whether a matrix M is additive or not.



Why additive metric?

- Recall that distance captures the actual number of mutations between a pair of species.
- If (1) the correct tree for a set of species is known and (2) we get the exact number of mutations for each edge,
 - The distance (the number of mutations) between two species i and j should be the sum of the edge weights along the path from i to j .
- **Additive metric seems reasonable!**



Hamming distance is additive?

- For any two species i and j , can we define M_{ij} to be the hamming distance between species i and j ?
 - Example: assume number of characters $m=5$
 - Species i : (A, C, G, C, T)
 - Species j : (C, C, A, C, T)
 - Hamming distance $h_{ij} = 2$
 - No! Hamming distance fails to capture the “multiple” mutations on the same site. It is not an additive metric
- **Solution:**
 - Use **poission correction**
 - corrected distance $M_{ij} = -\ln(1 - h_{ij}/m)$
 - As the number of characters increase, M converges to an additive metric

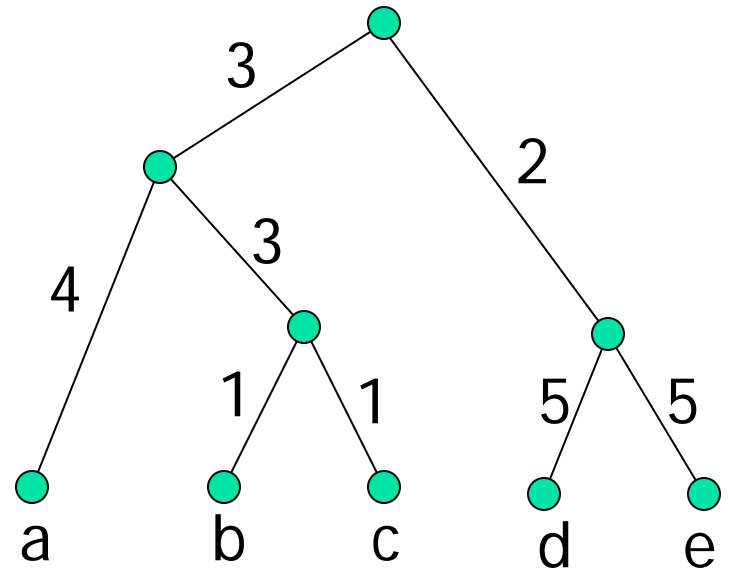


Ultrametric

- Assume M is additive. That is, there exists a tree T such that
 - the distance between any two species i and j equals the sum of the edge weights along the path from i to j .
- If we can further identify a **root** such that the path length from the root of T to every leaf is identical, then M is called an **ultrametric**
- A tree T which satisfies ultrametric is an **ultrametric tree**

Ultrametric Example

	a	b	c	d	e
a	0	8	8	14	14
b	8	0	2	14	14
c	8	2	0	14	14
d	14	14	14	0	10
e	14	14	14	10	0

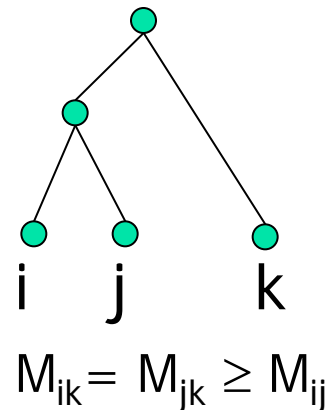


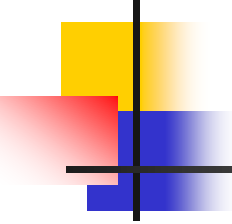
- Every path from root to leaf has the same length!

Properties of ultrametric

- Ultrametric is an additive metric. Thus, it satisfies 4-point condition.
- Additional property: **3-point condition**
 - M is ultrametric if and only if
 - for any three species in S, we can label them i, j, k such that $M_{ik} = M_{jk} \geq M_{ij}$

- **Proof of forward direction:**





Criteria for checking if M is ultrametric or not

- Based on the 3-point condition, we can check whether a matrix M is ultrametric or not.



Constant molecular clock assumption

- **Constant molecular clock** is an assumption in biology.
 - It states that the number of accepted mutations occurring in any time interval is proportional to the length of that interval.
 - Thus, all species evolved at equal rate from a common ancestor.
 - Recall that Alan Wilson found the origin of human based on this clock.
- Ultrametric tree states that the distance from the root to all species are the same. Thus, its correctness is based the constant molecular clock assumption, which is rarely correct!



Computational Problems

- Let M be a distance matrix for a set of species S .
 1. If M is ultrametric, can we reconstruct the corresponding ultrametric tree T in polynomial time?
 2. If M is additive, can we have an polynomial time algorithm to recover the corresponding additive tree T ?
 3. If M is not exactly additive, can we find the nearest additive tree T ?



Ultrametric Tree Reconstruction

- **Input:** Given an ultrametric matrix M for a set of species S
- **Problem:** Can we reconstruct the phylogenetic tree T for S ?

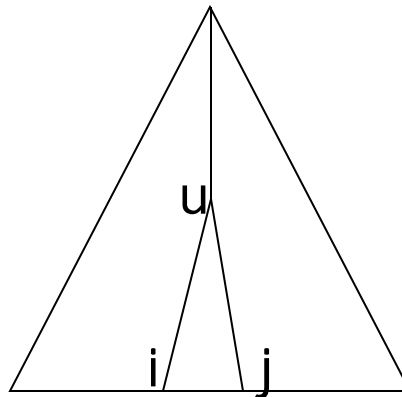


UPGMA (Unweighted Pair Group Method with Arithmetic mean)

- Build an ultrametric tree using a clustering procedure.
- Consider an ultrametric tree T . If a subset of species S form a subtree of T , we call it a **cluster**.
- **Idea:**
 - Every species forms a cluster.
 - Iteratively connect two nearest clusters, until one cluster is left.

Definition - height

- For a node u , define $\text{height}(u)$ be the path length from u to any of its descendent leaf. (Since T is ultrametric, every path should have the same length!)
- Let i and j be the descendent leaves of u in two different subtrees. To ensure that the distance from the root to both i and j are the same, $\text{height}(u) = M_{ij}/2$



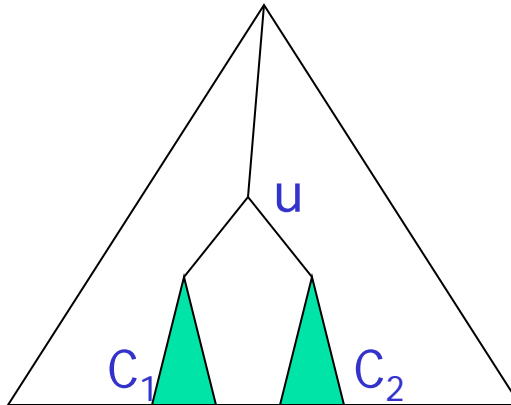
Distance between two clusters

- For any two clusters C_1 and C_2 of T

- Define

$$\text{dist}(C_1, C_2) = \frac{\sum_{i \in C_1, j \in C_2} M_{ij}}{|C_1| \cdot |C_2|}$$

- Note that $\text{dist}(C_1, C_2) = M_{ij}$ for all $i \in C_1$ and $j \in C_2$
- Let u be the lowest common ancestor of i and j .
 $\text{dist}(C_1, C_2) = 2 \text{ height}(u)$!





Idea of the algorithm

- Consider a set \mathbf{Z} of clusters
- Let A, B be two clusters such that $\text{dist}(A, B)$ is minimum.
- Let C be a tree formed by joining A and B with a root.
- Lemma: C is a cluster (subtree) of the ultrametric tree T



Observation

- For any clusters C_1, C_2 , and D ,
- $\text{dist}(C_1 \cup C_2, D) =$
 $(|C_1| \text{dist}(C_1, D) + |C_2| \text{dist}(C_2, D)) / (|C_1 \cup C_2|)$
- Try to prove this!

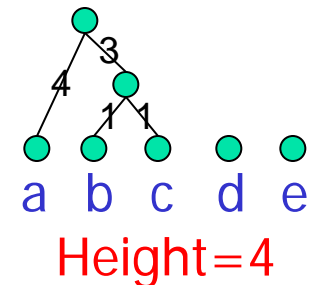
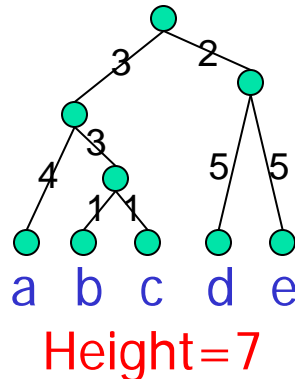
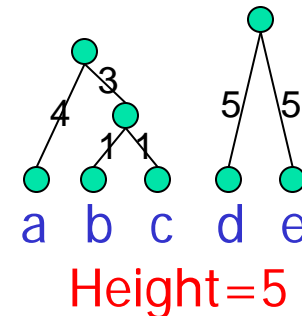
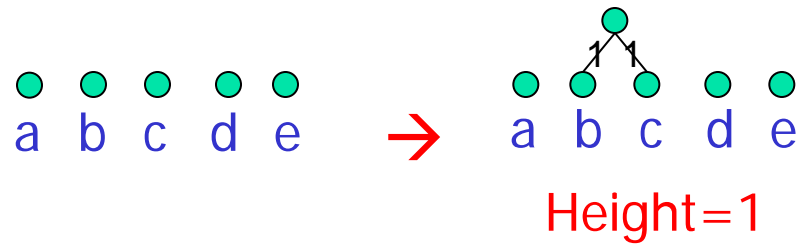


Algorithm

- Input: $n \times n$ ultrametric distance matrix M
- 1. Initialize set \mathbf{Z} to consist of n initial singleton clusters $\{1\}, \{2\}, \dots, \{n\}$
- 2. For all $\{i\}, \{j\} \in \mathbf{Z}$, initialize $\text{dist}(\{i\}, \{j\}) = M_{ij}$
- 3. Repeat $n-1$ times
 1. Determine cluster $A, B \in \mathbf{Z}$ such that $\text{dist}(A, B)$ is minimum.
 2. Define a new cluster $C = A \cup B$
 3. $\mathbf{Z} = \mathbf{Z} - \{A, B\} \cup \{C\}$
 4. Define a new node c and let c be the parent of a and b . Also, define $\text{height}(c) = \text{dist}(A, B)/2$
 5. For all $D \in \mathbf{Z} - \{C\}$, define $\text{dist}(D, C) = \text{dist}(C, D) = (|A|\text{dist}(A, D) + |B|\text{dist}(B, D)) / (|A| + |B|)$

Example

M	a	b	c	d	e
a	0	8	8	14	14
b	8	0	2	14	14
c	8	2	0	14	14
d	14	14	14	0	10
e	14	14	14	10	0





Time complexity

- Initialization can be done in $O(n^2)$ time
- There are $n-1$ iterations,
 - The bottleneck of each iteration is to find the cluster $A, B \in \mathbf{Z}$ such that $\text{dist}(A, B)$ is minimized, which takes $O(n^2)$ time.
- The total time complexity is $O(n^3)$.
- Next slide shows that $O(n)$ time is sufficient to find the cluster $A, B \in \mathbf{Z}$ such that $\text{dist}(A, B)$ is minimized.
 - Hence, the time complexity is $O(n^2)$.



Algorithm

- Input: $n \times n$ ultrametric distance matrix M
- 1. Initialize set \mathbf{Z} to consist of n initial singleton clusters $\{1\}, \{2\}, \dots, \{n\}$
- 2. For all $\{i\}, \{j\} \in \mathbf{Z}$, initialize $\text{dist}(\{i\}, \{j\}) = M_{ij}$
- 3. For every $P \in \mathbf{Z}$, set $\min_P = \text{argmin}_{Q \in \mathbf{Z}} \text{dist}(P, Q)$
- 4. Repeat $n-1$ times
 1. Determine cluster $A \in \mathbf{Z}$ such that $\text{dist}(A, \min_A)$ is minimized.
 2. Let $B = \min_A$. Define a new cluster $C = A \cup B$
 3. $\mathbf{Z} = \mathbf{Z} - \{A, B\} \cup \{C\}$
 4. Set $\min_C = \text{argmin}_{Q \in \mathbf{Z}} \text{dist}(C, Q)$
 5. For every cluster $A \in \mathbf{Z}$, if $\text{dist}(A, C) < \text{dist}(A, \min_A)$, set $\min_A = C$.
 6. Define a new node c and let c be the parent of a and b . Also, define $\text{height}(c) = \text{dist}(A, B)/2$
 7. For all $D \in \mathbf{Z} - \{C\}$, define $\text{dist}(D, C) = \text{dist}(C, D) = (|A|\text{dist}(A, D) + |B|\text{dist}(B, D)) / (|A| + |B|)$



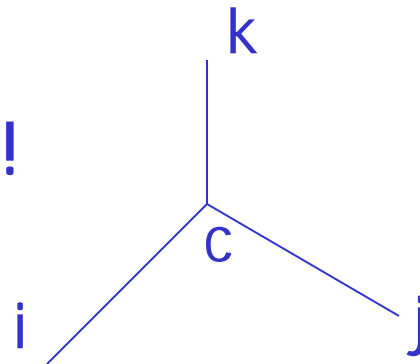
Additive tree reconstruction

- Suppose M is an additive metric. We show an algorithm which reconstructs the additive tree in $O(n^2)$ time.
- For any two species i and j , the additive tree is just an edge with weight M_{ij}



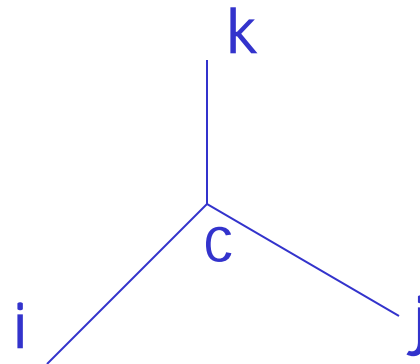
Recovering additive tree for 3 species

- For any three species i, j, k , we can find their center c as follows. [call it 3-star method!]
 - Let d_{xy} be the length of the path from x to y
 - (1) $M_{ik} = d_{ic} + d_{ck}$, (2) $M_{jk} = d_{jc} + d_{ck}$, and (3) $M_{ij} = d_{ic} + d_{cj}$
 - By solving the three equations, we have
 - $d_{ic} = (M_{ij} + M_{ik} - M_{jk})/2$
 - $d_{jc} = (M_{ij} + M_{jk} - M_{ik})/2$
 - $d_{kc} = (M_{ik} + M_{jk} - M_{ij})/2$
- Note: this tree is unique!



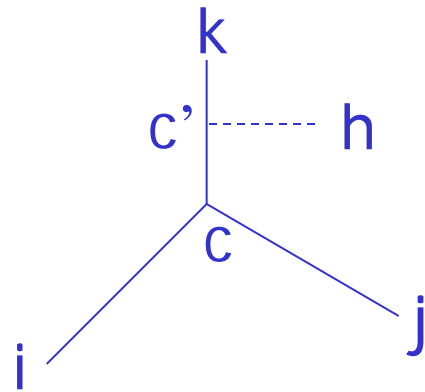
Recovering additive tree for 4 species (I)

- Given four species h, i, j, k , we want to recover the additive tree.
- For species i, j, k , we get the additive tree using the 3-star method
- To include h into the tree, we need to introduce one more internal node c' .
- c' will split either (i, c) , (j, c) or (k, c) .



Recovering additive tree for 4 species (II)

- To check whether c' splits (k, c) , we apply 3-star method for species i, k, h .
- If $d_{kc'} < d_{kc}$, c' splits (k, c) .



- Otherwise, using the same approach to check whether c' splits (i, c) or (j, c) .
- Note: c' can only split exactly one edge. Thus, the additive tree for 4 species is unique.



Recovering additive tree for k species

- Inductively, assume we know how to recover the additive tree for $k-1$ species.
- To recover the additive tree for k species,
 - We first build the additive tree T' for the first $k-1$ species. Then, insert the last species to T'
 - The last species will split one of the edge in T' .
 - For every edge in T' , we check whether the last species will split it using 3-star method.
- Note:
 - The time required is $O(k-1)$.
 - Also, the tree is unique!

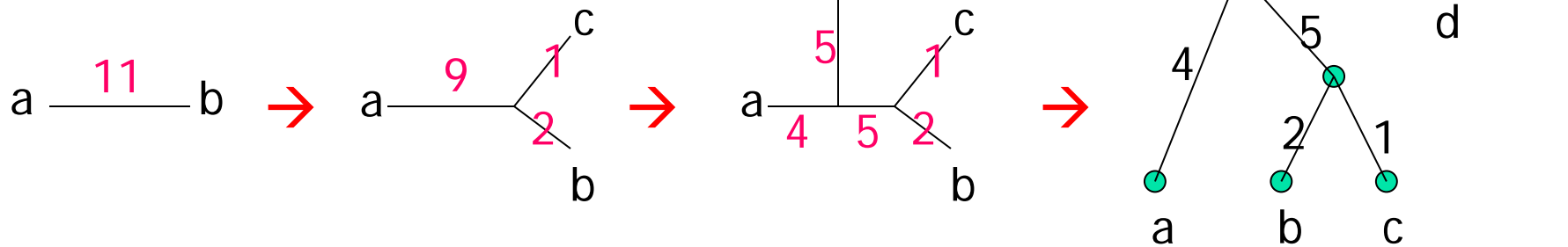


Time complexity

- In summary, to recover an additive tree with n species, the time is $O(1 + 2 + \dots + n) = O(n^2)$.
- Note: the additive tree for M is unique!

Example

M	a	b	c	d	e
a	0	11	10	9	15
b	11	0	3	12	18
c	10	3	0	11	17
d	9	12	11	0	8
e	15	18	17	8	0





Reconstruct nearly additive tree

- If M is not an additive metric, we can find the nearly additive tree using the following methods
 - Least Squares Method
 - Fitch-Margoliash method
 - Neighbor-Joining Method
 - L_∞ -metric



Least Squares Method

- **Input:** a metric M for a set of species S
- **Definition:** For any tree T for the set of species S , let D be its corresponding distance matrix. We define

$$SSQ(T) = \sum_{i=1}^n \sum_{j \neq i} \frac{(D_{ij} - M_{ij})^2}{D_{ij}^2}$$

- **Aim:** Find a tree T which minimizes $SSQ(T)$. Such tree is known as **Least Squares Tree**.
- This problem is NP-hard!



Neighbor joining (NJ)

- Attempts to approximate the additive tree
- **Idea:** join clusters A and B that are
 1. close together [small $\text{dist}(A,B)$]
 2. far away from the rest [big u_A and u_B where $u_A = (\sum_{D \in Z} \text{dist}(D, A))/(n-2)$]
- In other word, minimize $\text{dist}(A,B) - u_A - u_B$



Algorithm

- Input: $n \times n$ distance matrix M
- 1. Initialize set \mathbf{Z} to consist of n initial singleton clusters $\{1\}, \{2\}, \dots, \{n\}$
- 2. For all $\{i\}, \{j\} \in \mathbf{Z}$, initialize $\text{dist}(\{i\}, \{j\}) = M_{ij}$
- 3. Repeat $n-1$ times
 1. For every cluster $A \in \mathbf{Z}$, let $u_A = (\sum_{D \in \mathbf{Z}} \text{dist}(D, A))/(n-2)$
 2. Determine cluster $A, B \in \mathbf{Z}$ such that $\text{dist}(A, B) - u_A - u_B$ is minimized.
 3. Connect A and B by a new internal node r . The resulting cluster is called C .
 4. Calculate branch length: $d_{Ar} = \text{dist}(A, B)/2 + (u_A - u_B)/2$, $d_{Br} = \text{dist}(A, B)/2 + (u_B - u_A)/2$
 5. $\mathbf{Z} = \mathbf{Z} - \{A, B\} \cup \{C\}$
 6. Update $\text{dist}()$: For all $D \in \mathbf{Z} - \{C\}$, define $\text{dist}(D, C) = \text{dist}(C, D) = (\text{dist}(A, D) + \text{dist}(B, D) - \text{dist}(A, B))/2$



Time complexity

- Initialization takes $O(n^2)$ time
- There are n iterations
 - Each iteration takes $O(n^2)$ time, due to step 3.2.
- In total, the time complexity is $O(n^3)$.



L_∞ -metric

- Given two distance matrices M and E for a set of species S ,
 - $L_\infty(M, E) = \max_{i,j} |M_{ij} - E_{ij}|$
- **Input**: a metric M for a set of species S
- **Aim**: Find an additive metric E such that
 - $L_\infty(M, E)$ is minimized
- This problem is NP-hard!
- Agarwala et al. give a 3-approximation algorithm with respect to the L_∞ -metric.



More on neighbor joining

- Let M be any distance matrix and M_T be an additive matrix.
- Suppose $L_\infty(M, M_T) < \mu(T)/2$
 - where $\mu(T)$ is the minimum edge length in T .
 - In this case, M is said to be nearly additive.
- Atteson showed that, given a nearly additive matrix M ,
 - NJ always return the correct tree T .



Can we apply distance based methods on character based data?

- Yes! For any two species i and j , we can compute the distance M_{ij} .
- However, as stated before, we cannot compute the distance M_{ij} as the hamming distance h_{ij} between species i and j .
- Instead, we use a corrected distance.
 - E.g. Assuming the CF model,
 - the corrected distance $M_{ij} = -\ln(1 - h_{ij}/m)$.
 - As the number of characters increase, M converges to an additive metric.



Can we improve the tree generated by distance based methods?

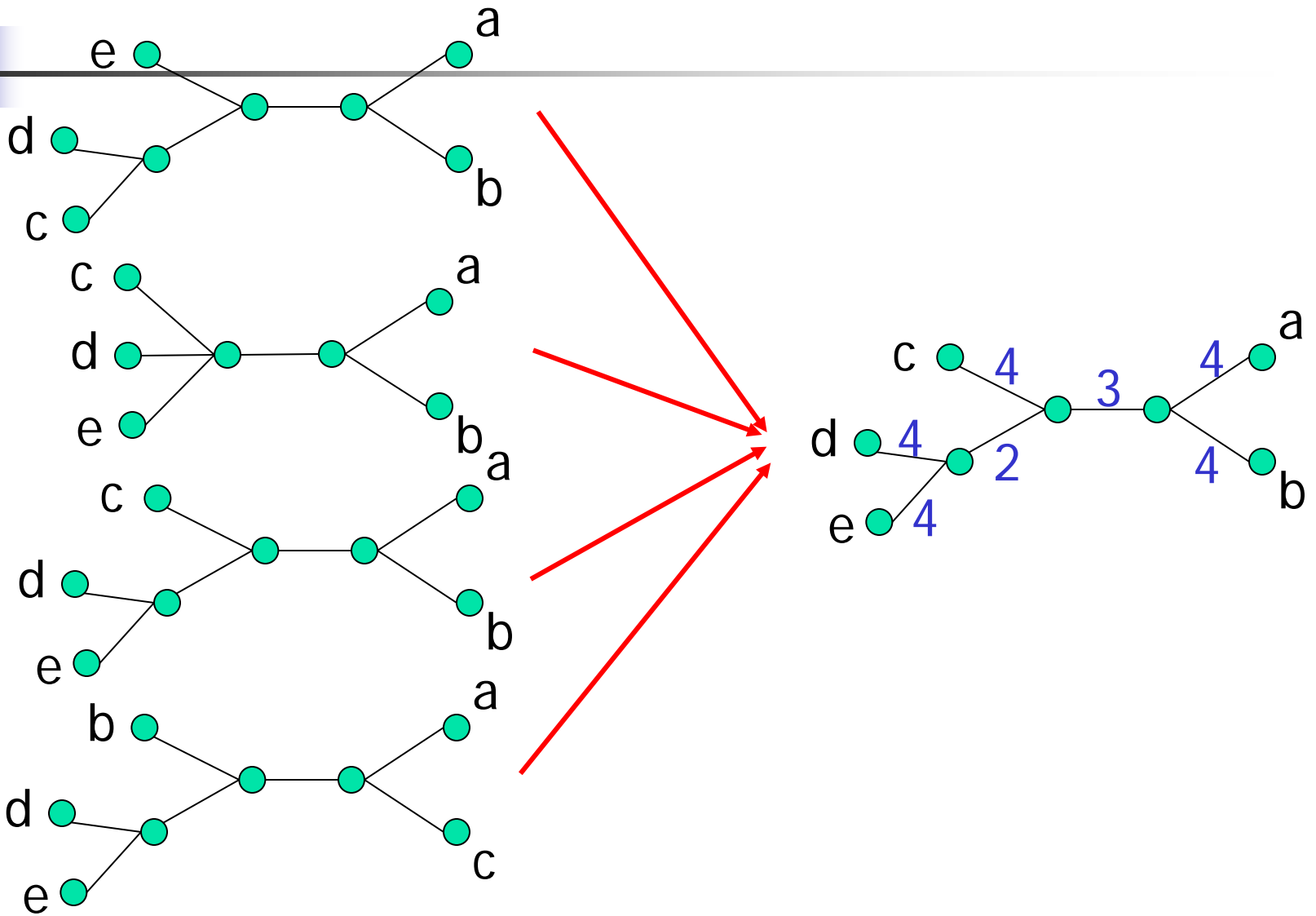
- The tree generated by a distance-based method is usually unstable.
- Bootstrapping helps to identify those stable edges in the tree.

Algorithm Bootstrapping

Input: n species, each is described by m characters

- Repeat x times,
 - Randomly select m characters (with replacement)
 - Build the distance matrix for the n species
 - Build the distance-based tree
- Report the consensus tree

Example: bootstrapping



Can tree reconstruction methods
infer the correct tree?





Can tree reconstruction methods infer the correct tree? (I)

- Experimentally, bacteriophage T7 was propagated and split sequentially in the presence of a mutagen, where each lineage was tracked.
- Out of 135,135 possible phylogenetic trees, the true tree was correctly determined by phylogenetic methods in a blind analysis. Five different phylogenetic methods were used independently, and each one chose the correct tree.
 - DM Hillis, JJ Bull, ME White, MR Badgett, and IJ Molineux. Experimental phylogenetics: generation of a known phylogeny. *Science* 255(5044):589-592, 1992.



Can tree reconstruction methods infer the correct tree? (II)

- In 1998, researchers used 111 modern HIV-1 (AIDS virus) sequences in a phylogenetic analysis to predict the nucleotide sequence of the viral ancestor of which they were all descendants.
- The predicted ancestor sequence closely matched, with high statistical probability, an actual ancestral HIV sequence found in an HIV-1 seropositive African plasma sample collected and archived in the Belgian Congo in 1959
 - Zhu, T., B. Korber, et al. (1998) "An African HIV-1 sequence from 1959 and implications for the origin of the epidemic." Nature 391: 594-597.

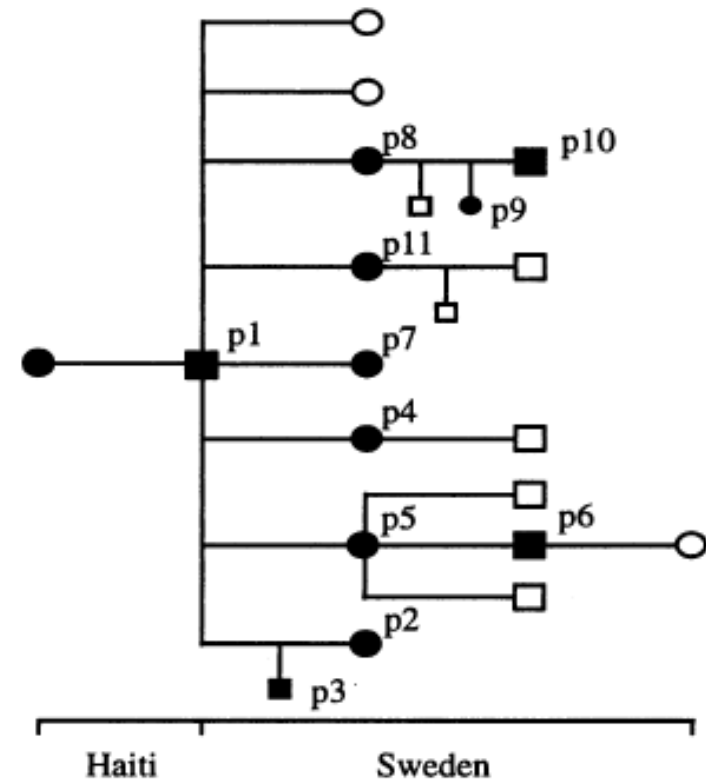


HIV evolution

- HIV evolves approximately one million times faster than the nucleic genomes of higher organisms
- Leitner et al. studied the real evolution tree of 11 HIV-1 samples in a period of 13 years (1981-1994).
- They also collect the sample sequences.
- T Leitner, D Escanilla, C Franzén, M Uhlén, and J Albert. Accurate reconstruction of a known HIV-1 transmission history by phylogenetic tree analysis. PNAS, 93(20):10864–10869, 1996.

Population history of HIV-1 in a Swedish transmission cluster

- HIV-1 transmission time are also known (within a few months)
- Square: male; circle: female
- Solid: HIV-infected; Open: uninfected
- Small symbols: children





The true phylogenetic tree

- The env V3 and p17 gag regions of the HIV-1 genome were directly sequenced from uncultured peripheral blood mononuclear cells of p1 to p11 at different time
- Combining virus transmission time and sample collection time, we get the true phylogenetic tree for 13 HIV-1 genomes.



Phylogenetic tree reconstruction methods

- 7 tree reconstruction methods:
 - Fitch-Margoliash (FM)
 - Neighbor-joining (NJ)
 - Minimum-evolution (ME)
 - Maximum-likelihood (ML)
 - Maximum-parsimony (MP)
 - Unweighted pair group method using arithmetic averages (UPGMA)
 - A FM method assuming a molecular clock (KITSCH)
- They are applied to 13 samples on regions
 - Env V3
 - p17 gag
 - Env V3 + p17 gag



Results

- FM, NJ, ML perform the best
- MP in the middle
- UPGMA and KITSCH, which assume constant molecular clock perform the worst
- All methods tended to overestimate the length of short branches and underestimate long branches.

Method

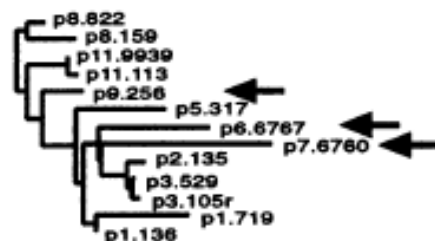
Gene fragment

p17 gag

env V3

p17 gag + env V3

FM



JC / K2' / F84

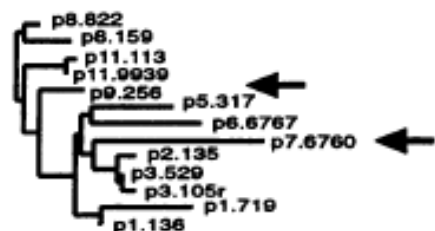


JC / K2' / F84



JC / K2' / F84

NJ



JC / K2' / F84

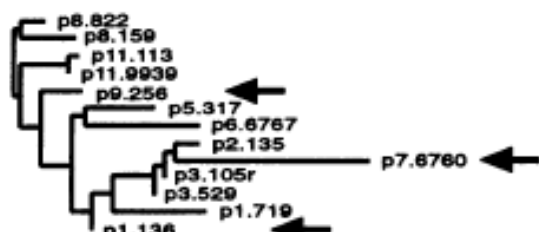


JC / K2' / F84

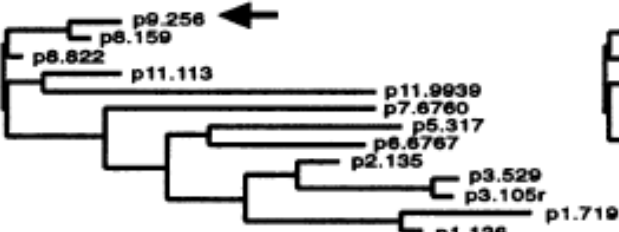


JC / K2' / F84

ML



JC / K2' / F84

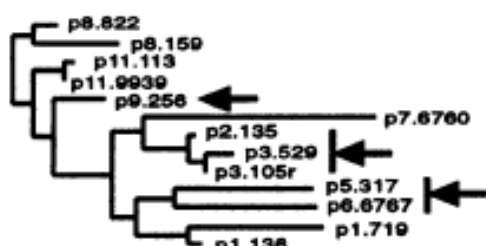


K2' / F84

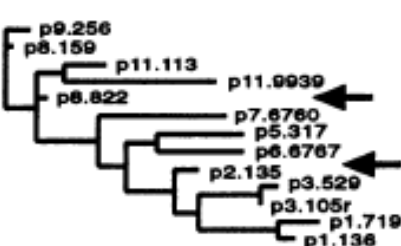


JC / K2' / F84

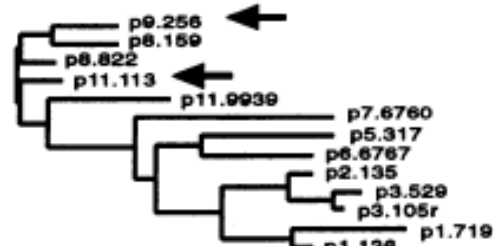
MP



DW



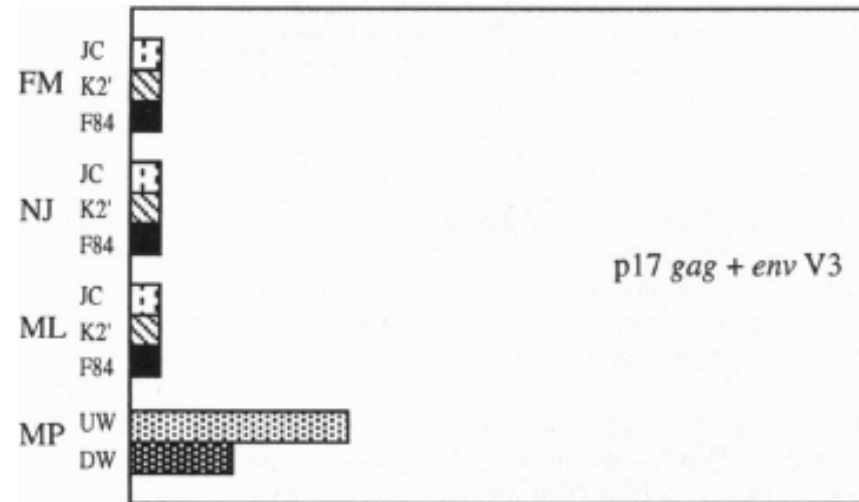
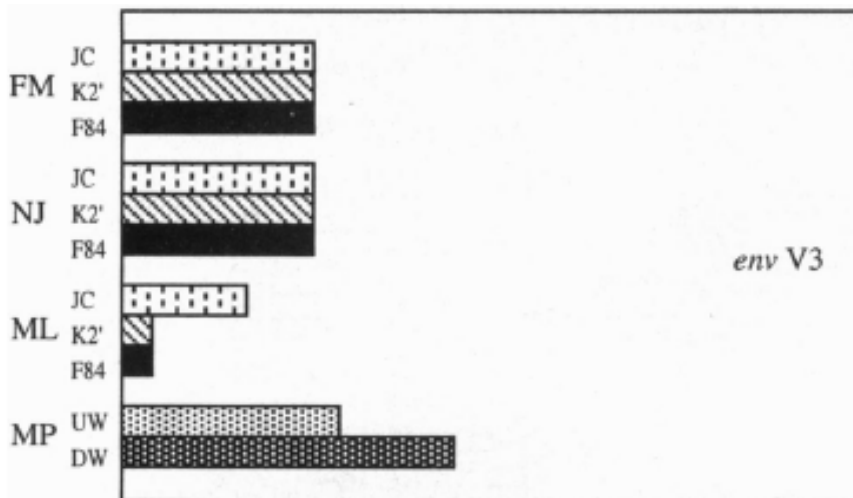
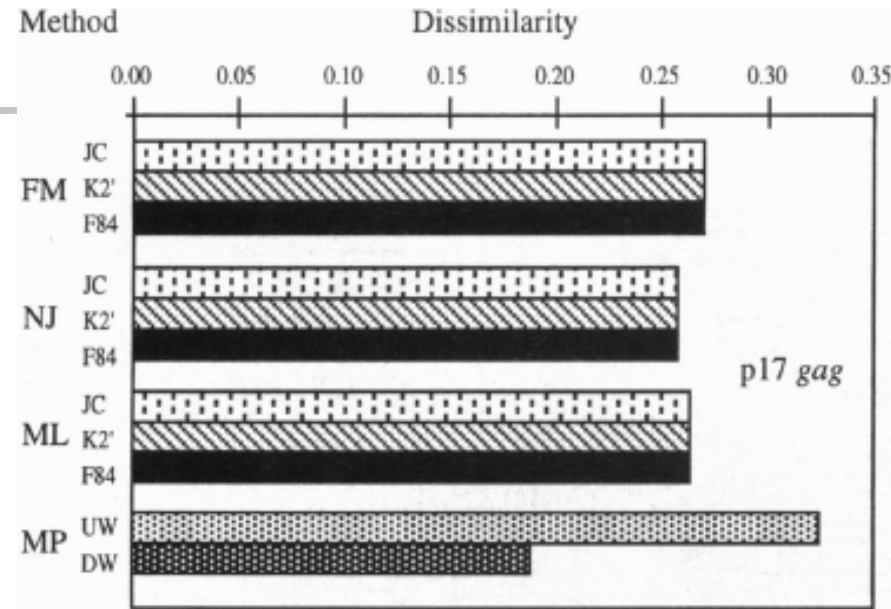
UW



DW

Dissimilarity with the true tree

- Dissimilarity is based on comparing quartets between the true tree and the constructed tree.
- $p17+V3 > V3 > p17$
- $ML, NJ, FM > MP$





Software for constructing phylogenetic tree

- Felsenstein's PHYLIP
 - It offers a large array of methods, including ML, MP and NJ.
 - Command line mode only
 - The most widely used program suite
 - Source code is available
 - Free of charge
 - <http://evolution.genetics.washington.edu/phylip.html>



Methodology for constructing phylogeny

- Multiple alignment
- Bootstrapping (says, 100 times)
- Apply phylogeny reconstruction methods
- Build consensus tree