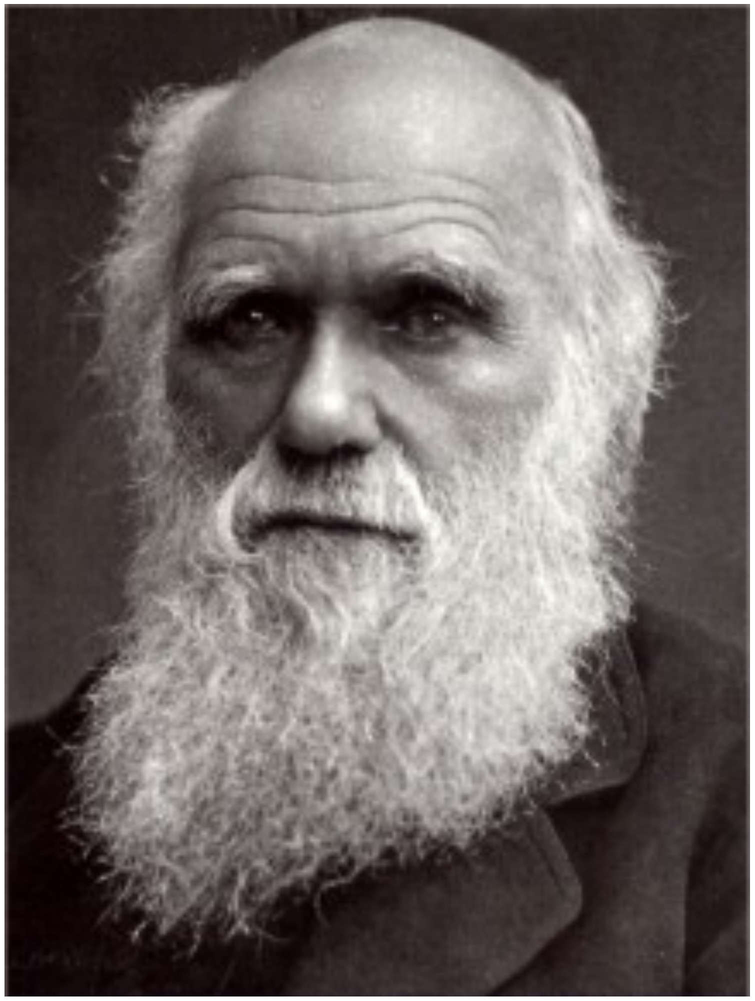
Algorithmic problems related to phylogenetic trees

Evolution

Evolution of new organisms is driven by

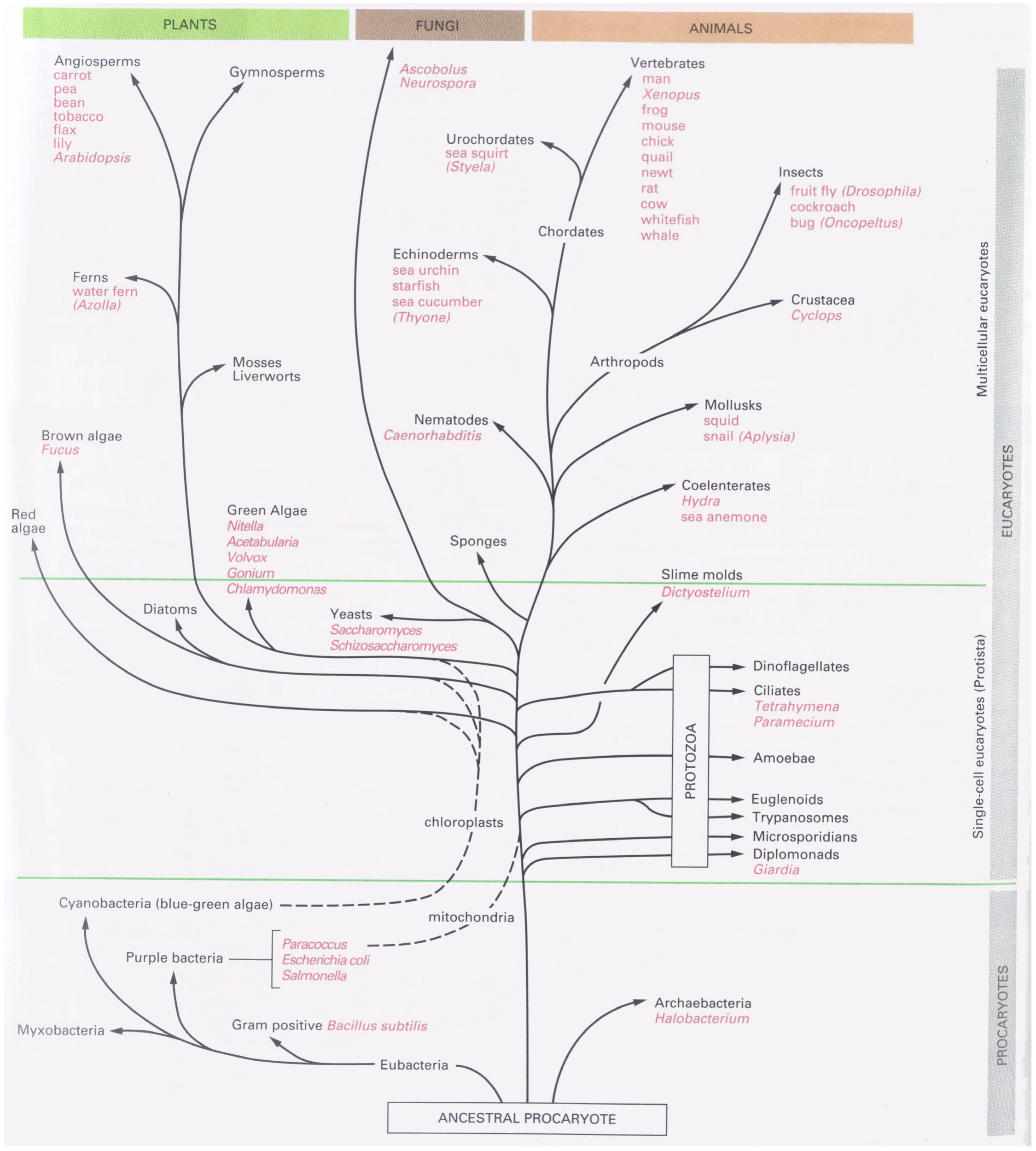
- **Mutations**
  - The DNA sequence can be changed due to single base changes, deletion/insertion of DNA segments, etc.
- **Selection bias**



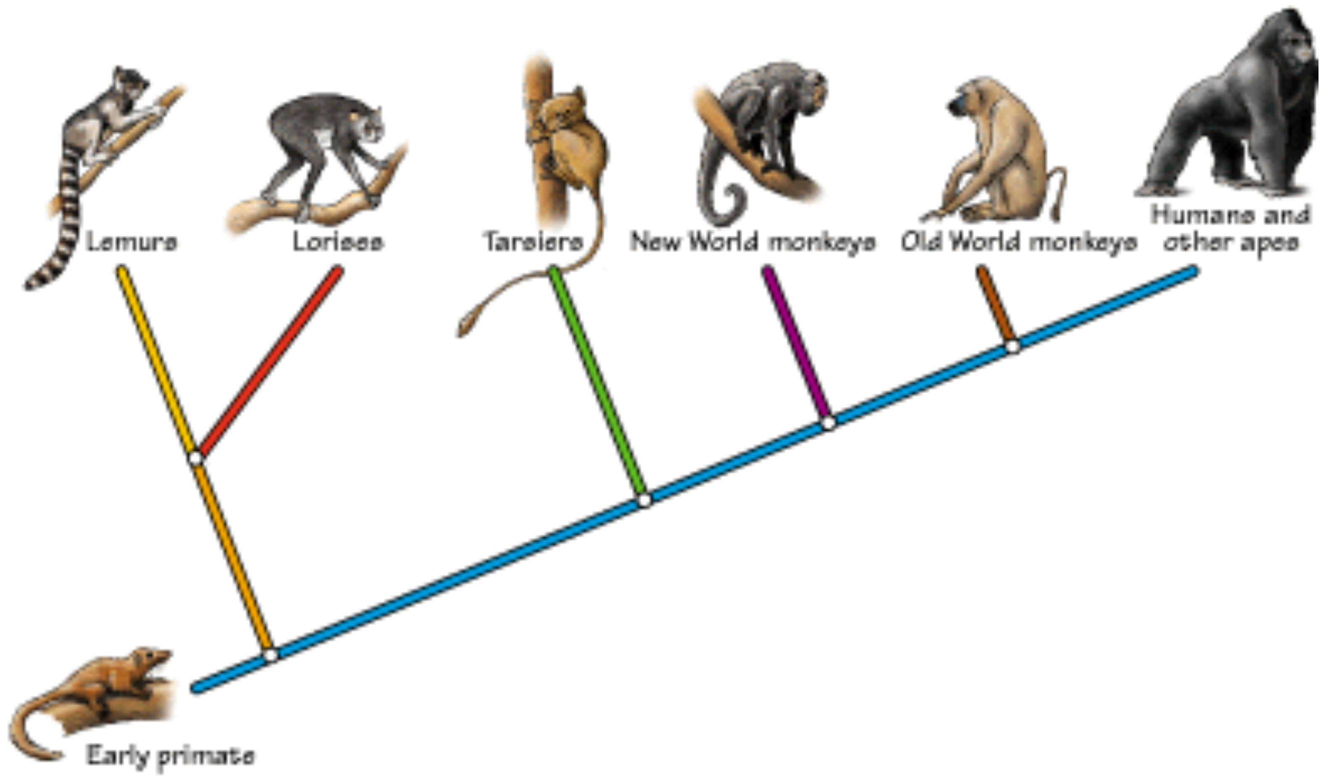
Theory of evolution

- Basic idea
  - **speciation** events lead to creation of different species.
  - Speciation caused by physical separation into groups where different genetic variants become dominant
- Any two species share a (possibly distant) common ancestor

The Tree of Life



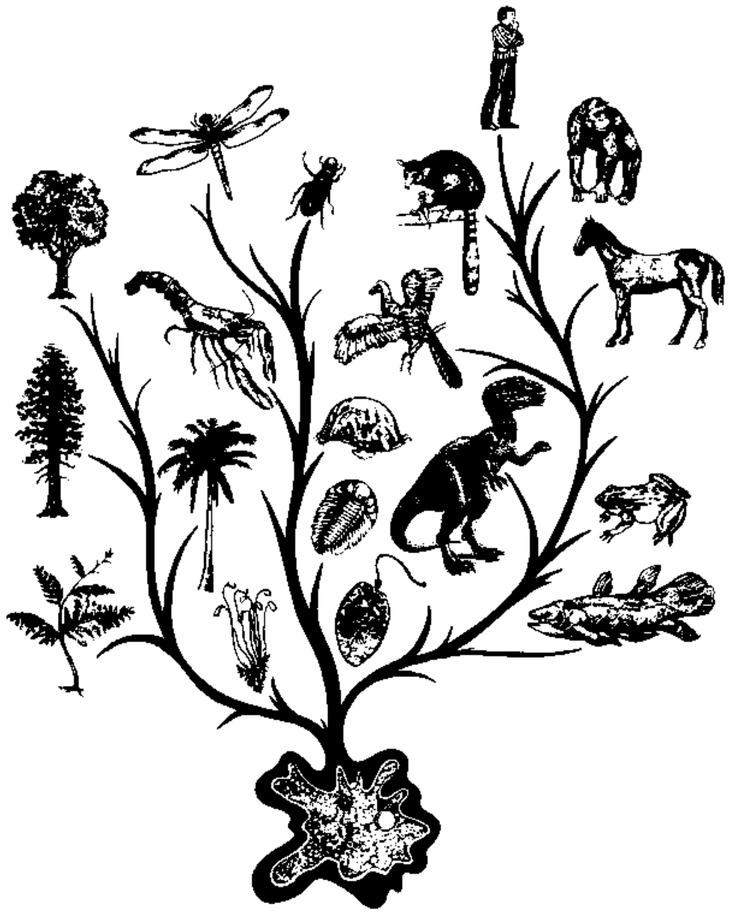
Primate evolution



A **phylogeny** is a tree that describes the sequence of speciation events that lead to the forming of a set of current day species; also called a **phylogenetic tree.**

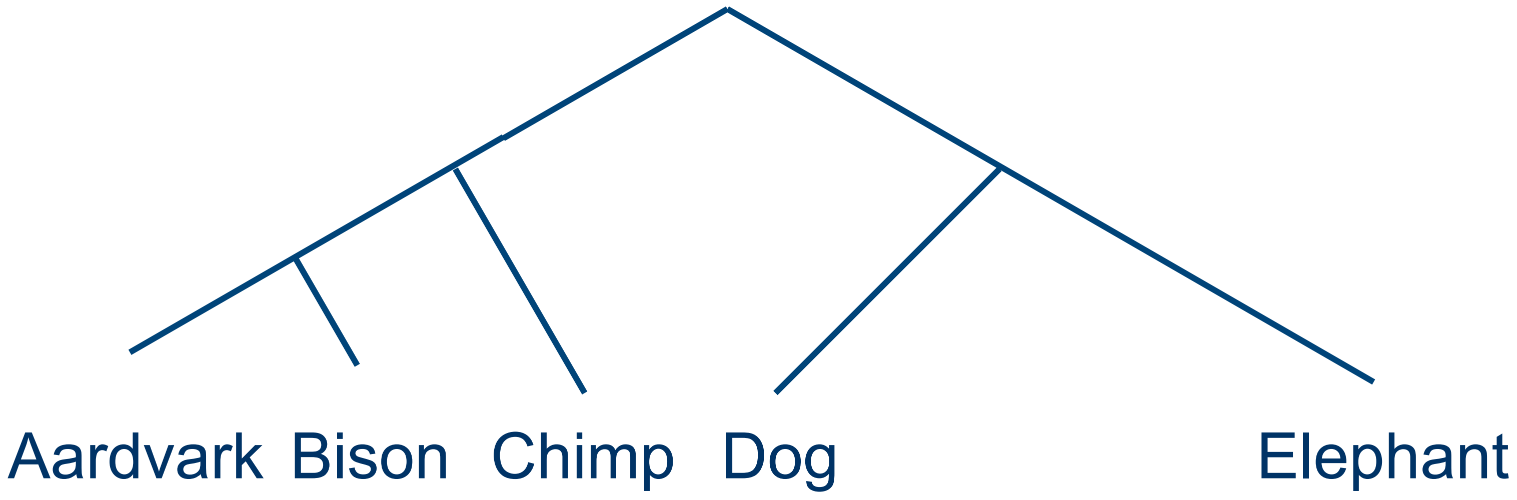
From sequences to a phylogenetic tree

Rat	QEPGGLVVPPTDA
Rabbit	QEPGGMVVPPTDA
Gorilla	QEPGGLVVPPTDA
Cat	REPGGLVVPPTEG



There are many possible types of sequences to use (e.g. Mitochondrial vs Nuclear proteins).

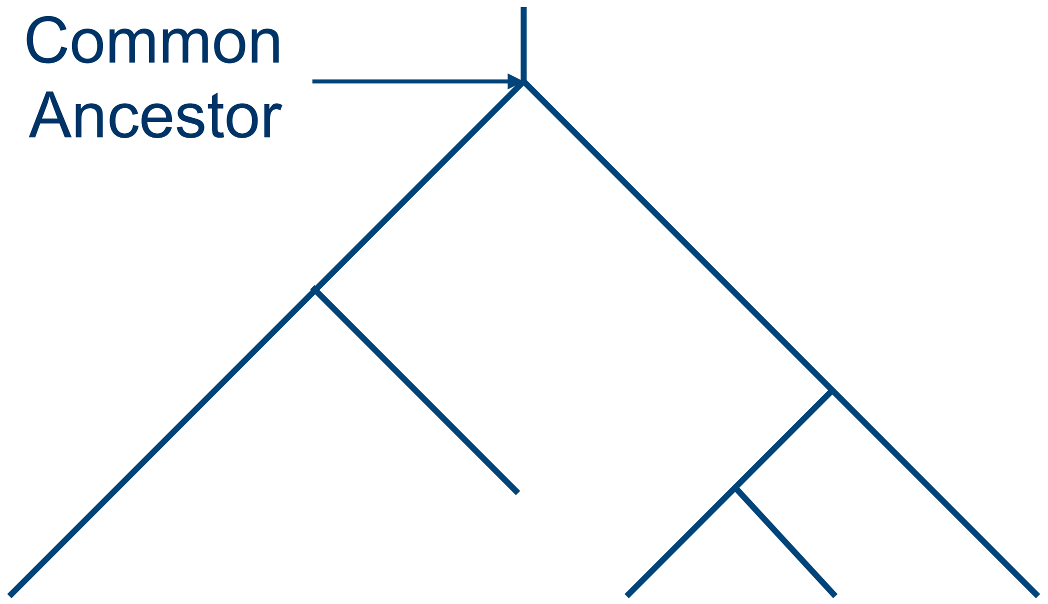
Phylogenenetic trees



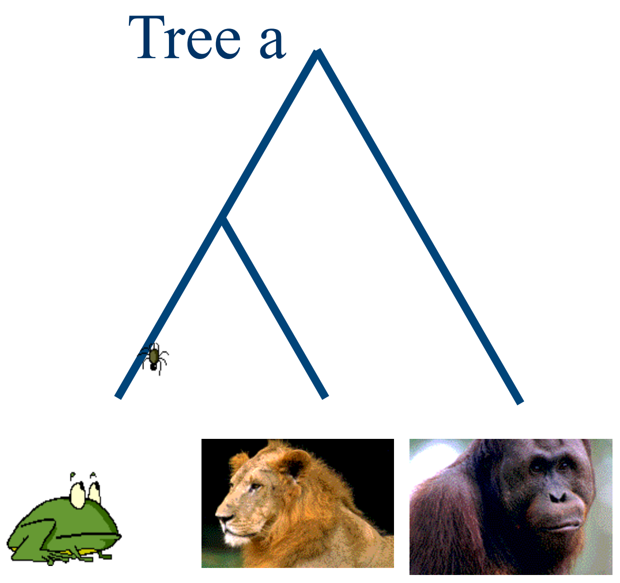
- **Leaves** - current day species (or taxa – plural of taxon)
- **Internal vertices** - hypothetical common ancestors
- **Edges length** - “time” from one speciation to the next

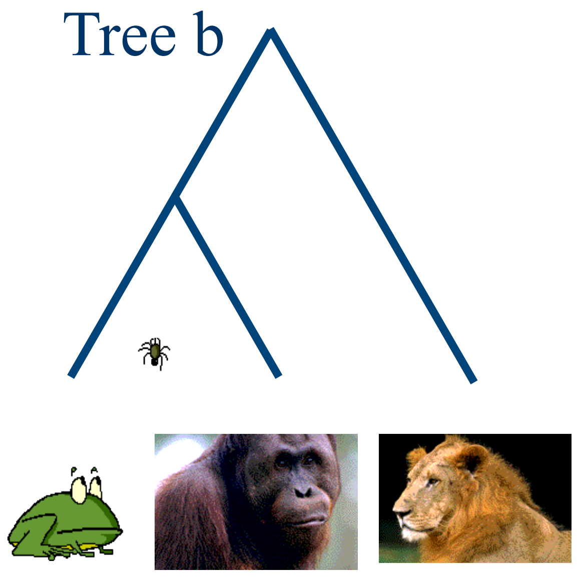
Types of Trees

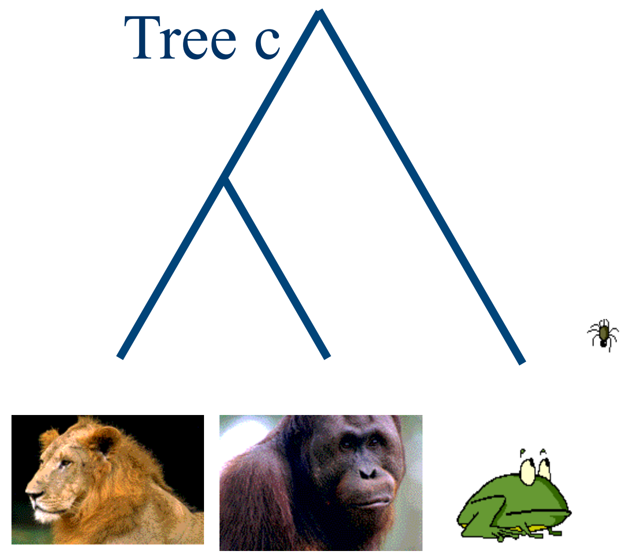
A natural model to consider is that of **rooted** trees

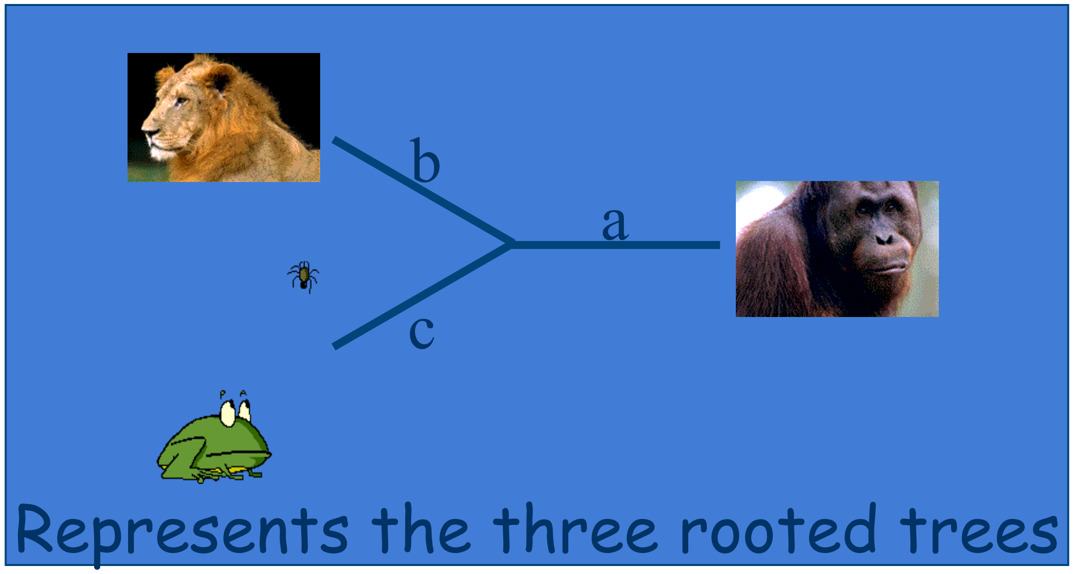


Rooted versus unrooted trees









Represents the three rooted trees

Total numbers of trees

- For N taxa,
  - Rooted bifurcating trees:
    - $(2n-3)!! = (2n-3)!/2^{n-2}(n-2)!$
  - Unrooted bifurcating trees
    - $(2n-5)!!$
  - Tree shapes (with n leaves rooted bifurcating tree): $C(2n-1,n)/(2n-1)$

Type of Data

- **Distance**-based
  - Input is a matrix of distances between species
  - Can be fraction of residue they disagree on, or alignment score between them, or …
- **Character**-based
  - Examine each character (e.g., residue) separately

Character-based methods for constructing phylogenies

In this approach, trees are constructed by comparing the characters of the corresponding species. Characters may be morphological (teeth structures) or molecular (nucleotides in homologous DNA sequences). One common approach is Maximum Parsimony

Common Assumptions:

•Independence of characters (no correlations)

•Best tree is one where minimal changes take place

Character based methods: Input

species	C_1	C_2	C_3	C_4	…													C_m
dog	A	A	C	A	G	G	T	C	T	T	C	G	A	G	G	C	C	C
horse	A	A	C	A	G	G	C	C	T	A	T	G	A	G	A	C	C	C
frog	A	A	C	A	G	G	T	C	T	T	T	G	A	G	T	C	C	C
human	A	A	C	A	G	G	T	C	T	T	T	G	A	T	G	A	C	C
pig	A	A	C	A	G	T	T	C	T	T	C	G	A	T	G	G	C	C
	*	*	*	*	*			*	*			*	*				*	*

- Each character (column) is processed independently.
- The green character will separate the human and pig from frog, horse and dog.
- The red character will separate the dog and pig from frog, horse and human.
- We seek for a tree that will **best explain** all characters simultaneously.

Maximum Parsimony

A Character-based method

Input:

- h sequences (one per species), all of length k.

Goal:

- Find a tree with the input sequences at its leaves,

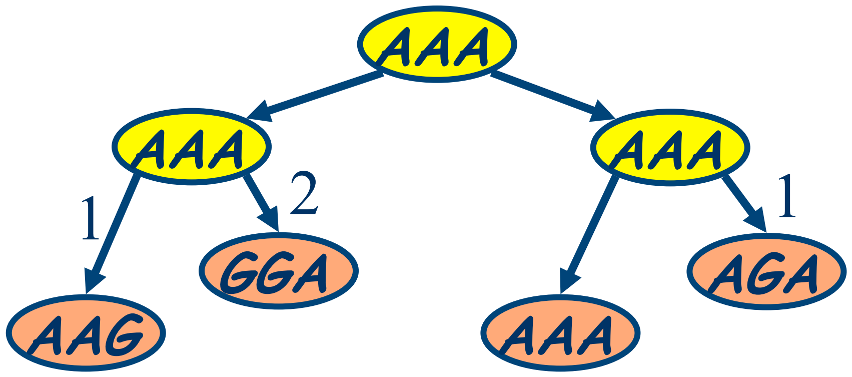
and an assignment of sequences to internal nodes,

such that the total number of substitutions is minimized.

Example

Input: four nucleotide sequences: AAG, AAA, GGA, AGA taken from four species.

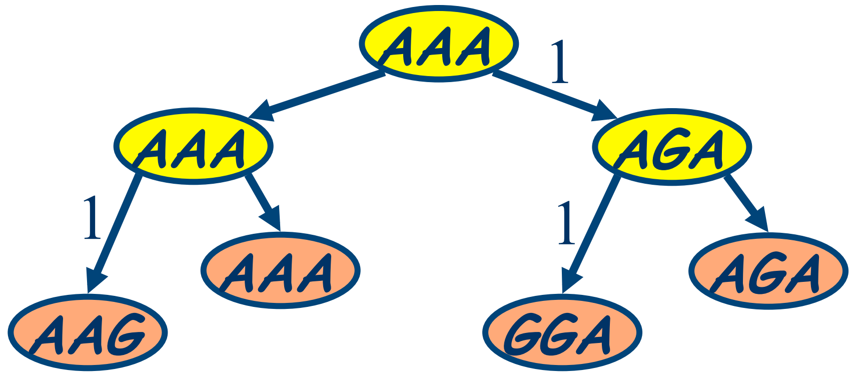
By the parsimony principle, we seek a tree that has a minimum total number of substitutions of symbols between species and their originator in the phylogenetic tree. Here is one possible tree.



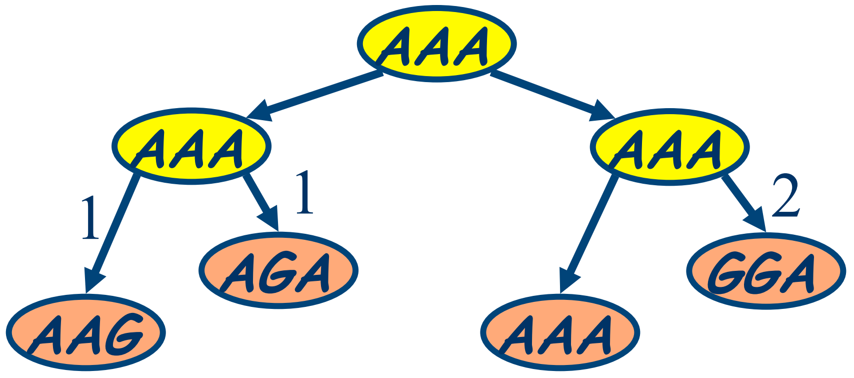
Total #substitutions = 4

Example

There are many assignments for this tree. For example:



Total #substitutions = 3



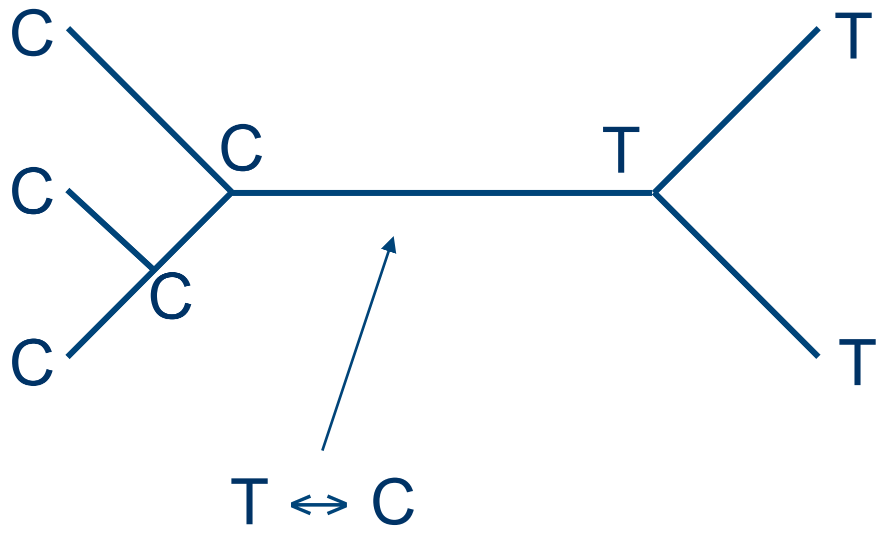
Total #substitutions = 4

The left tree is preferred over the right tree.

The total number of changes is called the **parsimony score**.

Example with one letter sequences

- Suppose we have five species, such that three have ‘C’ and two ‘T’ at a specified position
- Minimal tree has only one evolutionary change:



Parsimony based reconstruction

- Two separate components:

1. A procedure to find the minimum number of changes needed to explain the data for a given tree topology, where species are assigned to leaves.

2. A search through the space of trees.

Finding the right tree: Perfect phylogeny problem

The algorithms of Fitch and Sankoff assume that the tree is known. Finding the optimal tree is harder.

Recall the general problem:

Input: A set of species, specified by strings of characters.

Output: A tree T, and assignment of species to the leaves of T, with minimum parsimony score.

The Perfect Phylogeny Problem

Basic assumption for the perfect phylogeny problem:

A ***character*** is a significant property, which distinguishes between species (e.g. dental structure).

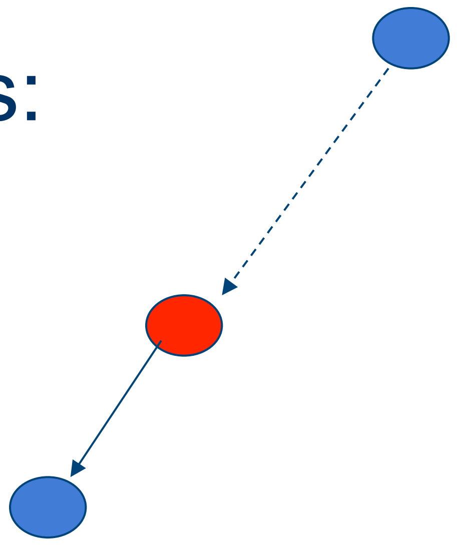
Hence, characters in evolutionary trees should be "Homoplasy free", as we define next.

Homoplasy-free characters 1

Characters in Phylogenetic Trees should avoid:

reversal transitions

- A species regains a state it' s direct ancestor has lost.
- Famous known reversals:
  - Teeth in birds.
  - Legs in snakes.

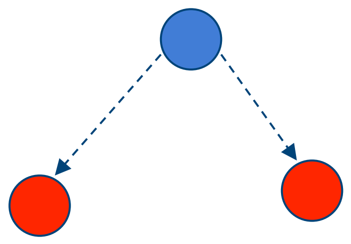


Homoplasy-free characters 2

…and also avoid

convergence transitions

- Two species possess the same state while their least common ancestor possesses a different state.
- Famous known convergence: The marsupials.



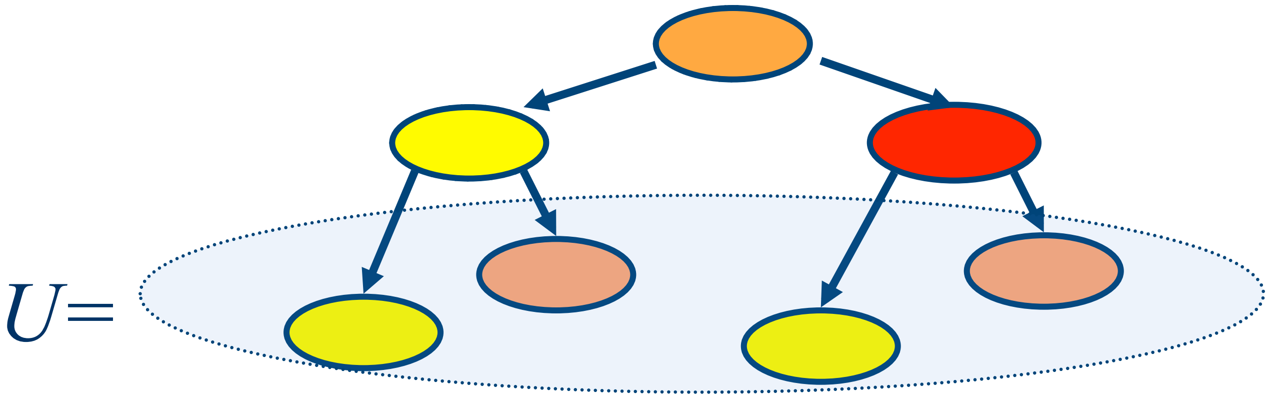
Characters as colorings

A ***coloring*** of a tree T=(V,E) is a mapping

$$C{:}V \rightarrow \text{[set of colors]}$$

A p***artial coloring*** of T is a mapping defined on a subset of the vertices $U \subseteq V$:

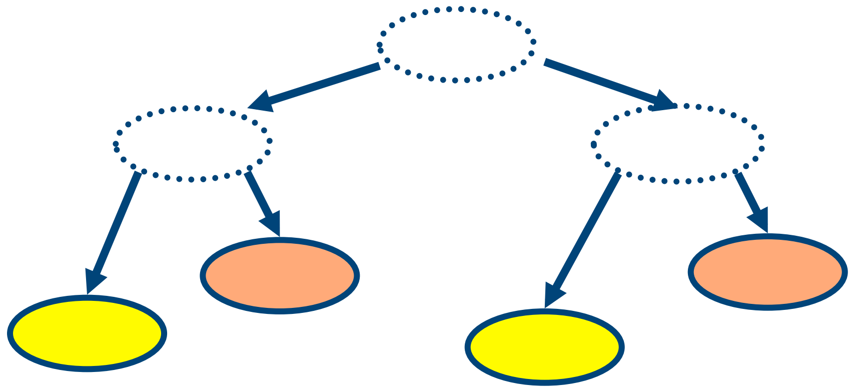
$$C{:}U \rightarrow \text{[set of colors]}$$



Characters as Colorings

Each character defines a (partial) coloring of the corresponding phylogenetic tree:

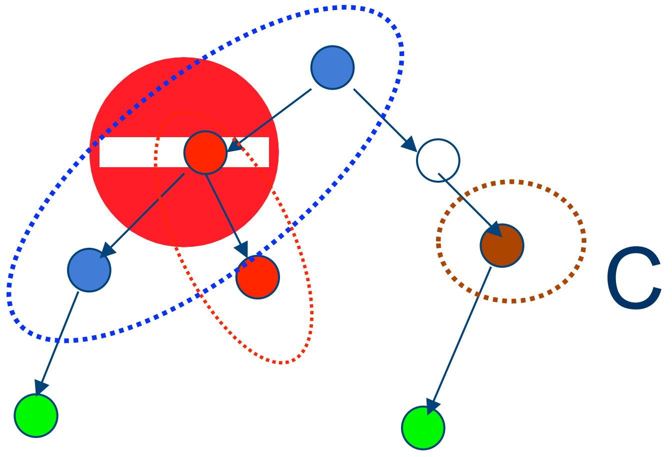
Species ≡ Vertices
States ≡ Colors



Convex colorings (characters)

Let *T*=(*V*,*E*) be a colored tree, and *d* be a color. The *d*-***carrier*** is the minimal subtree of *T* containing all vertices colored *d*

Definition: A (partial/total) coloring of a tree is ***convex*** iff all *d-carriers* are disjoint

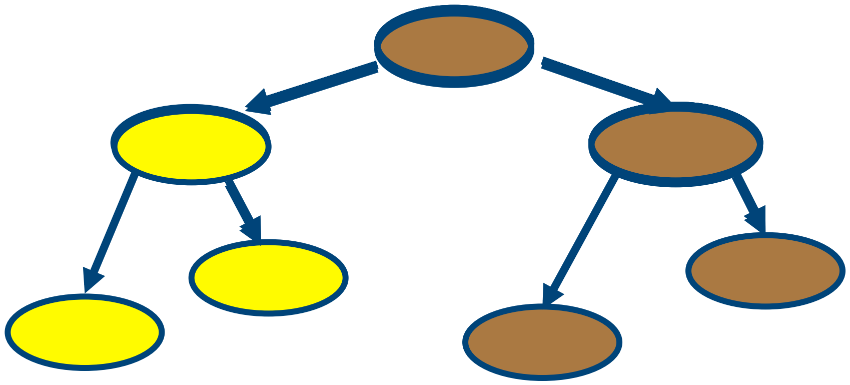


Convexity $\Leftrightarrow$ Homoplasy Freedom

A character is Homoplasy free (avoids reversal and convergence transitions)

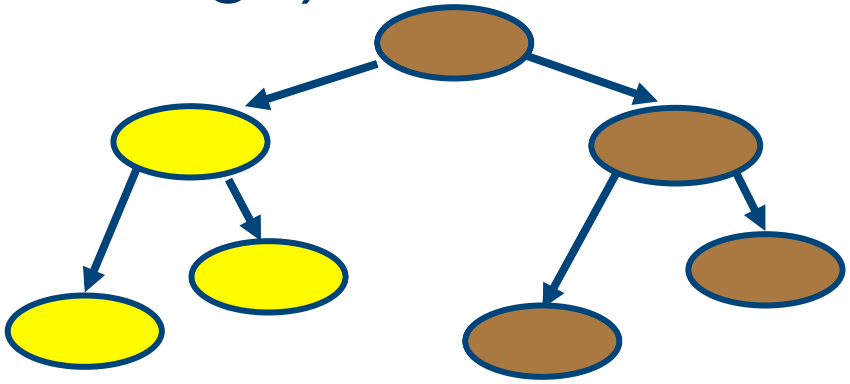
$\updownarrow$

The corresponding (partial) coloring is ***convex***



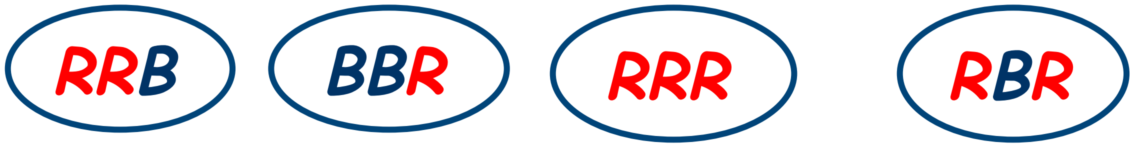
Perfect phylogeny problem

- Input: a set of species, and ***many*** characters.
- Question: is there a tree *T* containing the species as vertices, in which ***all*** the characters (colorings) are convex?



The Perfect phylogeny problem (pure graph theoretic setting)

Input: Partial colorings ($C_1,\ldots,C_k$) of a set of vertices U (in the example: 3 total colorings: left, center, right, each by two colors).



Problem: Is there a tree $T=(V,E)$, s.t. $U\subseteq V$ and for $i=1,\ldots,k$,, C_i is a convex (partial) coloring of T?

NP-Hard In general, in P for some special cases. Next we show a polynomial time algorithm for the case of *binary characters*.

Directed binary perfect phylogeny

Directed binary characters:

- $\mathbf{0}$ – property exists
- $\mathbf{1}$ – property doesn't exist
- Initially (at the root) all propertied do not exist.

Input: binary coloring $(C_1,\dots,C_m)$ of a set S (nxm binary matrix M)

Problem: Find a phylogenetic tree T over S (if one exists), s.t.

1. For $j=1,\dots,m$, the partial coloring induced by C_j is convex in T.
2. The root has state $\mathbf{0}$ in all characters.

➢ We will present a polynomial-time solution

Example

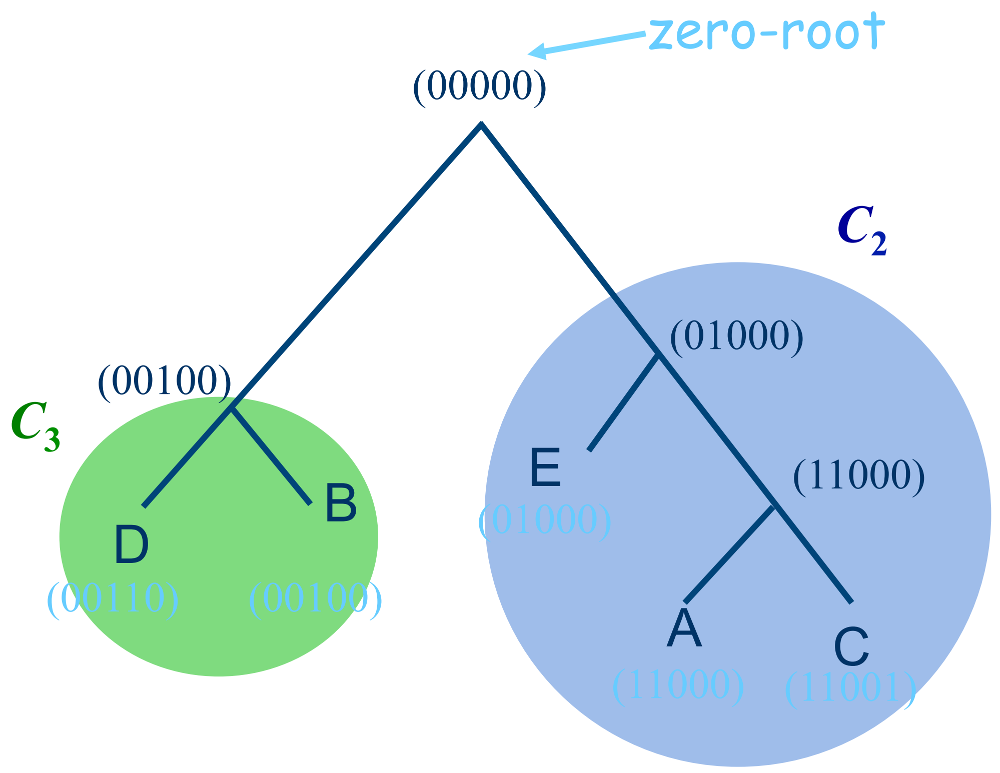
Input:

m characters

n species

$$\begin{array}{c|ccccc|}
 & C_1 & C_2 & C_3 & C_4 & C_5 \\
\hline
A & 1 & 1 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 0 & 0 \\
C & 1 & 1 & 0 & 0 & 1 \\
D & 0 & 0 & 1 & 1 & 0 \\
E & 0 & 1 & 0 & 0 & 0 \\
\hline
\end{array}$$

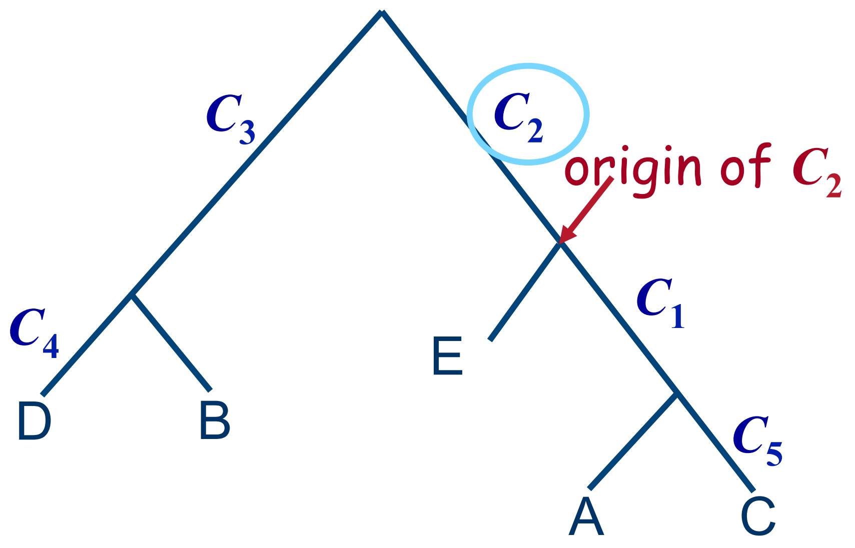
Possible output:



An important observation

A tree is a directed perfect phylogeny for a given **0/1** matrix
iff
we can map each character to an edge/vertex on which this character was "turned on".

Example:



$$
\begin{array}{c|ccccc|}
 & C_1 & C_2 & C_3 & C_4 & C_5 \\
\hline
A & 1 & 1 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 0 & 0 \\
C & 1 & 1 & 0 & 0 & 1 \\
D & 0 & 0 & 1 & 1 & 0 \\
E & 0 & 1 & 0 & 0 & 0 \\
\hline
\end{array}
$$

Laminar matrices

Definitions:

- O_j – set of objects that have character C_j ($O_j = \{i : M_{ij} = 1\}$).
- A collection of sets $\{S_1, \ldots, S_k\}$ is ***laminar*** if
  - for all i, j, either S_i and S_j are disjoint, or one includes the other.

Theorem: A binary matrix M has a perfect phylogenetic tree iff the collection $\{O_1, \ldots, O_m\}$ is ***laminar***.

Laminar

$$\begin{array}{c|ccccc|}
 & C_1 & C_2 & C_3 & C_4 & C_5 \\
\hline
A & 1 & 1 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 0 & 0 \\
C & 1 & 1 & 0 & 0 & 1 \\
D & 0 & 0 & 1 & 1 & 0 \\
E & 0 & 1 & 0 & 0 & 0 \\
\hline
\end{array}$$

Not Laminar

$$\begin{array}{c|ccccc|}
 & C_1 & C_2 & C_3 & C_4 & C_5 \\
\hline
A & 1 & 1 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 0 & 1 \\
C & 1 & 1 & 0 & 0 & 1 \\
D & 0 & 0 & 1 & 1 & 0 \\
E & 0 & 1 & 0 & 0 & 1 \\
\hline
\end{array}$$

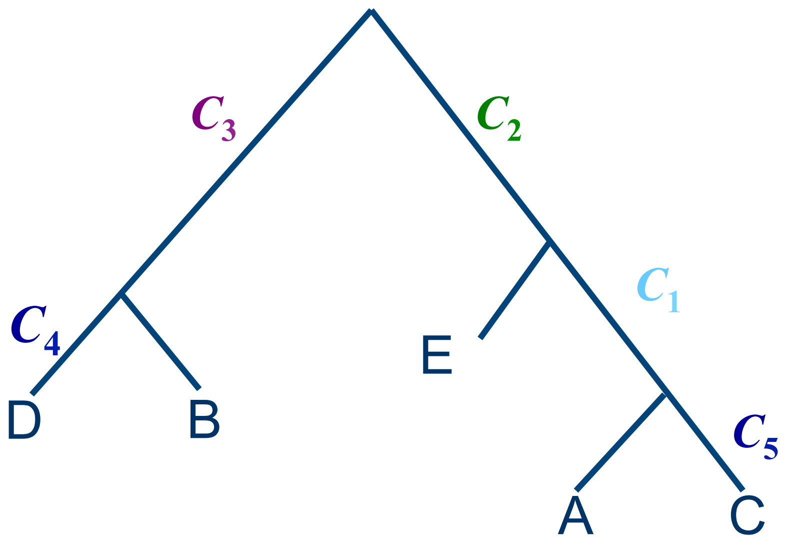
Proof

➔ Assume M has a perfect phylogeny.

Consider the edges labeled C_i and C_j:

If there is a ***root-to-leaf*** path containing both edges (C_1,C_2 below), then O_i includes O_j or vice-versa.

Otherwise, O_i and O_j are disjoint (C_1,C_3 below).



← Assume that the collection $\{O_1, \ldots, O_k\}$ is laminar.

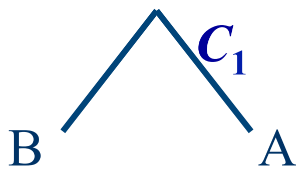
We prove by induction on the number of characters k that M has a perfect phylogenetic tree.

Basis: one character. There are at most two (distinct) objects, one with and one without this character.

$$\begin{array}{c|c|} & C_1 \\ \hline A & 1 \\ B & 0 \\ \hline \end{array}$$

root

$$\Longrightarrow$$



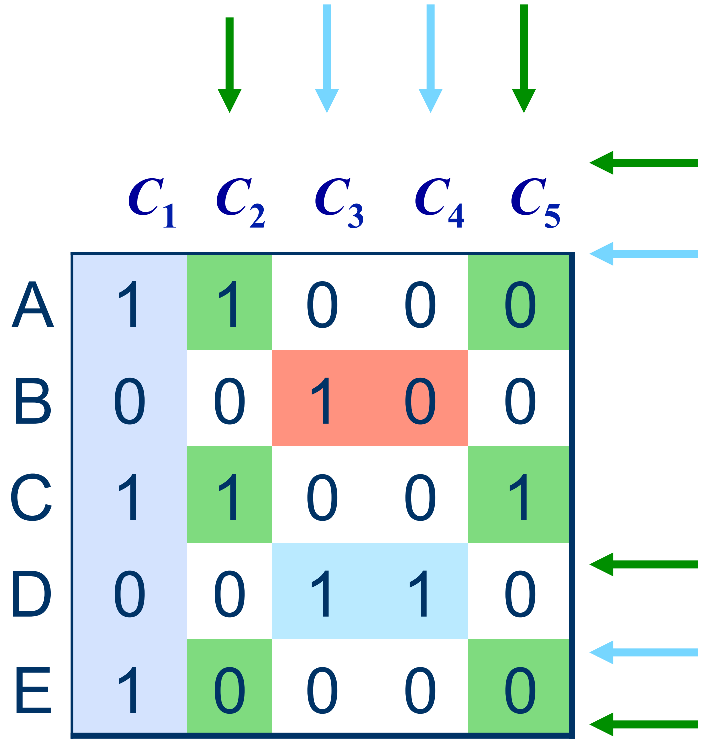
⇐ Assume that the collection $\{O_1, \ldots, O_k\}$ is laminar.

Induction step: assume correctness for n-1 characters.

Consider a matrix with n characters (non-zero columns), and assume WLOG that O_1 is not contained in O_j for all $j > 1$.

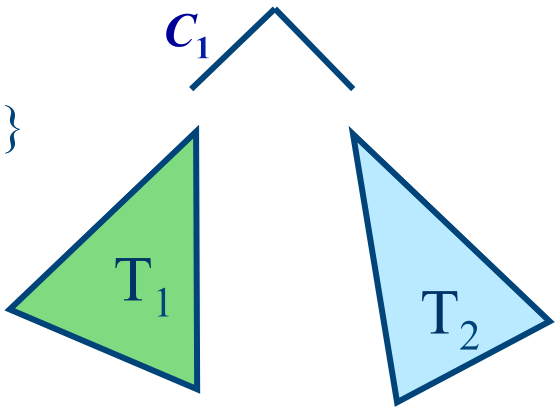
- S_1 – the set of objects i for which $M_{i1} = 1$. S_2 – the remaining objects.
- **Claim:** each character belongs to objects in S_1 or S_2 , but not to both.

By induction there are trees T_1 and T_2 for S_1 and S_2.



$$\begin{array}{c|ccccc|} & C_1 & C_2 & C_3 & C_4 & C_5 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 0 & 1 \\ D & 0 & 0 & 1 & 1 & 0 \\ E & 1 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$S_1 = \{A,C,E\}$
$S_2 = \{B,D\}$



Efficient Implementation

1. Sort the columns (characters) according to decreasing binary value.

Claim: If the binary value of column i is larger than that of column j, then O_i is not a proper subset of O_j.

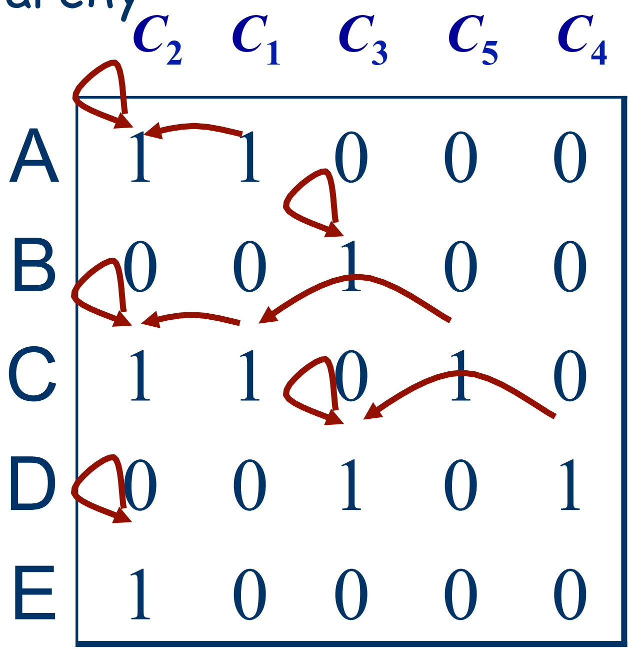
Proof: $O_i > O_j$ means the 1's in O_i are not covered by the 1's in O_j.

$$\begin{array}{c|ccccc} & C_1 & C_2 & C_3 & C_4 & C_5 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 0 & 1 \\ D & 0 & 0 & 1 & 1 & 0 \\ E & 0 & 1 & 0 & 0 & 0 \end{array} \Longrightarrow \begin{array}{c|ccccc} & C_2 & C_1 & C_3 & C_5 & C_4 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 1 & 0 \\ D & 0 & 0 & 1 & 0 & 1 \\ E & 1 & 0 & 0 & 0 & 0 \end{array}$$

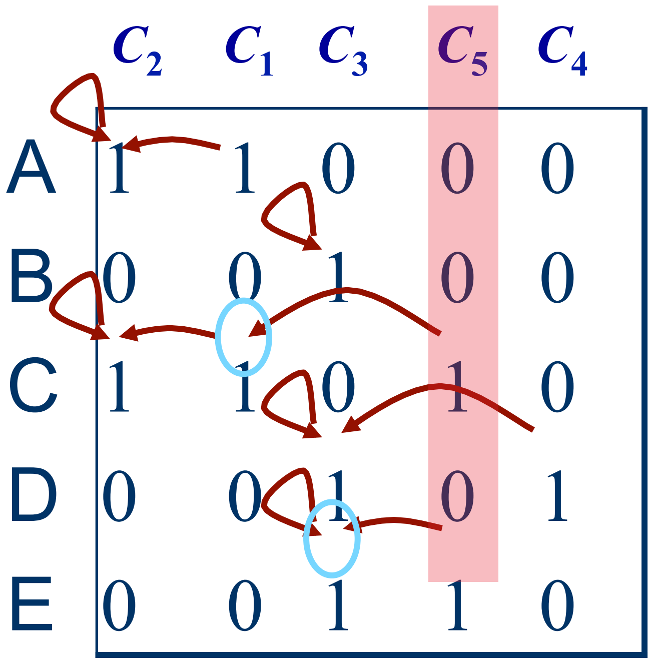
2. Make a backwards linked list of the **1**'s in each row

Claim: If the columns are sorted, then the set of columns is laminar iff for each column i, all the links leaving column i point at the same column.

➔ If the matrix is laminar then these pointers define the inclusion hierarchy

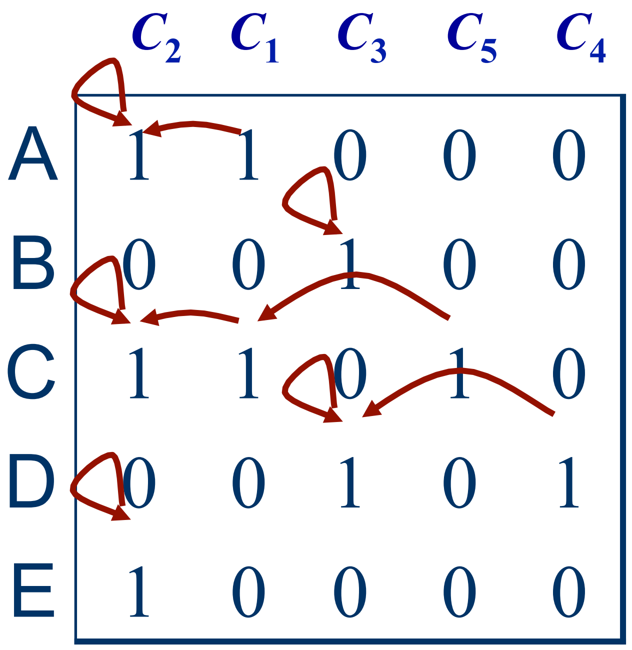


$$\begin{array}{c|ccccc} & C_2 & C_1 & C_3 & C_5 & C_4 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 1 & 0 \\ D & 0 & 0 & 1 & 0 & 1 \\ E & 1 & 0 & 0 & 0 & 0 \end{array}$$

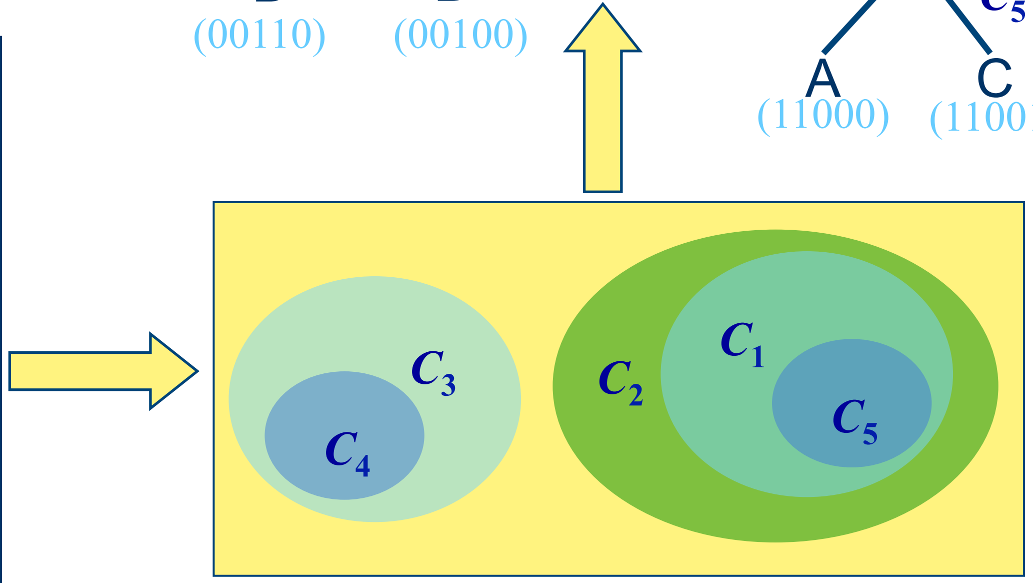


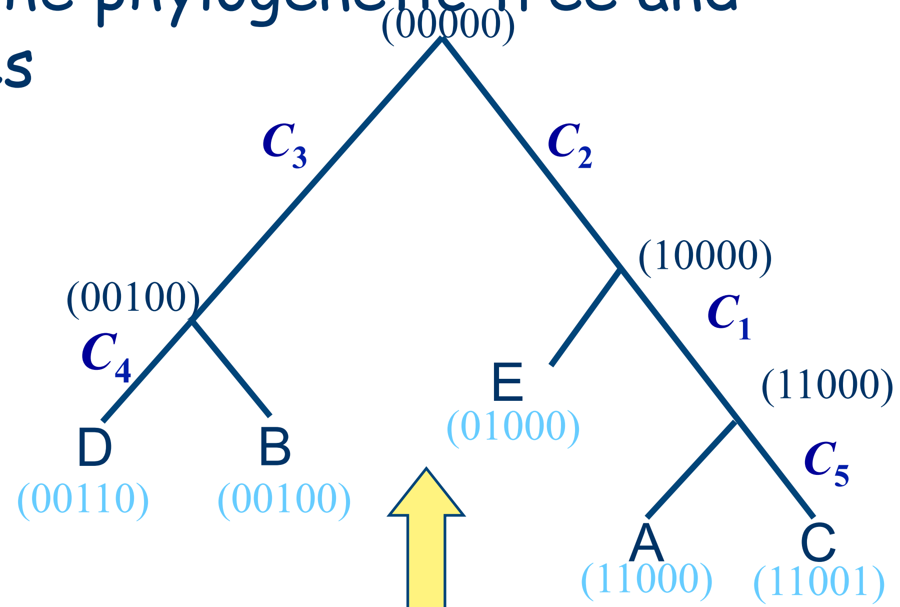
$$\begin{array}{c|ccccc} & C_2 & C_1 & C_3 & C_5 & C_4 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 1 & 0 \\ D & 0 & 0 & 1 & 0 & 1 \\ E & 0 & 0 & 1 & 1 & 0 \end{array}$$

3. If the matrix is laminar, compute the inclusion hierarchy
4. Reconstruct topology of the phylogenetic tree and ancestral character states



$$\begin{array}{c|ccccc} & C_2 & C_1 & C_3 & C_5 & C_4 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 1 & 0 \\ D & 0 & 0 & 1 & 0 & 1 \\ E & 1 & 0 & 0 & 0 & 0 \end{array}$$





1. Sort the columns (characters) according to decreasing binary value.
2. Make a backwards linked list of the **1**'s in each row
3. If the matrix is laminar, compute the inclusion hierarchy
4. Reconstruct topology of the phylogenetic tree and ancestral character states

Complexity: $O(mn)$ – use radix (bucket) sort in stage 1.

$$\begin{array}{c|ccccc|} & C_1 & C_2 & C_3 & C_4 & C_5 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 0 & 1 \\ D & 0 & 0 & 1 & 1 & 0 \\ E & 0 & 1 & 0 & 0 & 0 \end{array} \quad \Longrightarrow \quad \begin{array}{c|ccccc|} & C_2 & C_1 & C_3 & C_5 & C_4 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 1 & 0 \\ D & 0 & 0 & 1 & 0 & 1 \\ E & 1 & 0 & 0 & 0 & 0 \end{array}$$

Genotypes and haplotypes

Each individual has two "copies" of each chromosome.

At each site, each chromosome has one of two alleles (states) denoted by 0 and 1 (motivated by SNPs)

$$\begin{array}{ccccccccc} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline \hline 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Two haplotypes per individual

Merge the haplotypes

2 1 2 1 0 0 1 2 0 Genotype for the individual

SNP Data

- A SNP is a Single Nucleotide Polymorphism - a site in the genome where two different nucleotides appear with sufficient frequency in the population (say each with 5% frequency or more).
- SNP maps in human have been compiled with a density of about 1 site per 1000.
- SNP data is what is mostly collected in populations - it is much cheaper to collect than full sequence data, and focuses on variation in the population, which is what is of interest.

Haplotype Map Project: HAPMAP

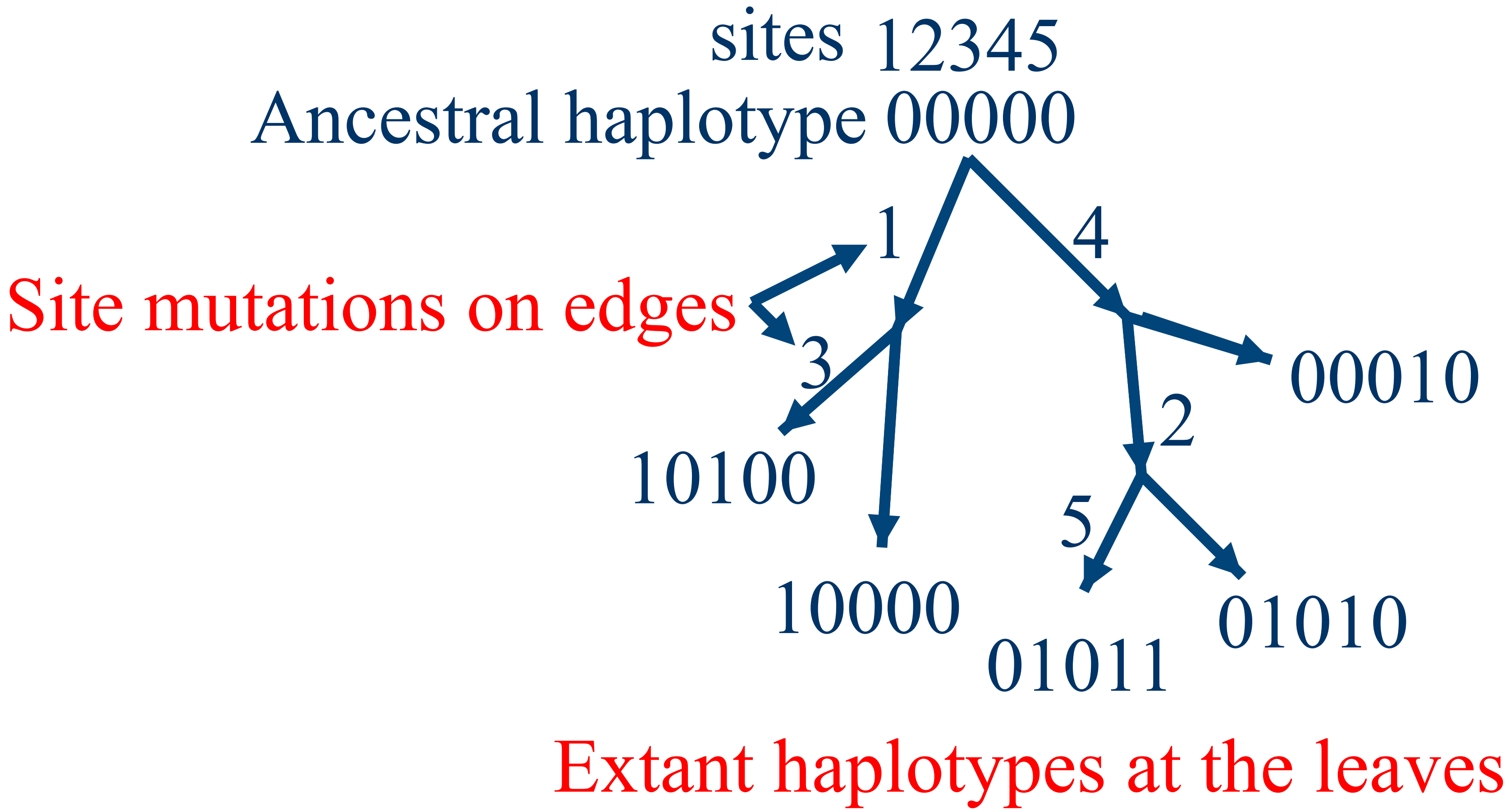
- NIH lead project ($100M) to find common haplotypes in the Human population.
- Used to try to associate genetic-influenced diseases with specific haplotypes, to either find causal haplotypes, or to find the region near causal mutations.
- Haplotyping individuals is expensive.

Haplotyping problem

- Biological Problem: for disease association studies, haplotype data is more valuable than genotype data, but haplotype data is hard to collect. Genotype data is easy to collect.

- Computational problem: given a set of n genotypes, determine the original set of n haplotype pairs that generated the n genotypes. This is hopeless without a genetic model.

Perfect phylogeny model of haplotype evolution



The Perfect phylogeny model

We assume that the evolution of extant haplotypes can be displayed on a rooted, directed tree, with the all-0 haplotype at the root, where each site changes from 0 to 1 on exactly one edge, and each extant haplotype is created by accumulating the changes on a path from the root to a leaf, where that haplotype is displayed.

In other words, the extant haplotypes evolved along a perfect phylogeny with all-0 root.

Justification for perfect phylogeny model

- In the absence of recombination each haplotype of any individual has a single parent, so tracing back the history of the haplotypes in a population gives a tree.
- Recent strong evidence for long regions of DNA with no recombination. Key to the NIH haplotype mapping project.
- Mutations are rare at selected sites, so are assumed non-recurrent.
- Connection with coalescent models.

Perfect Phylogeny Haplotype (PPH)

Given a set of genotypes S, find an explaining set of haplotypes that fits a perfect phylogeny.

sites

S

	1	2
a	2	2
b	0	2
c	1	0

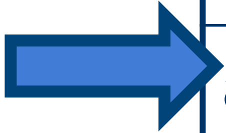
Genotype matrix

A haplotype pair explains a genotype if the merge of the haplotypes creates the genotype. Example: The merge of 0 1 and 1 0 explains 2 2.

The PPH Problem

Given a set of genotypes, find an explaining set of haplotypes that fits a perfect phylogeny

	1	2
a	2	2
b	0	2
c	1	0



	1	2
a	1	0
a	0	1
b	0	0
b	0	1
c	1	0
c	1	0

The Haplotype Phylogeny Problem

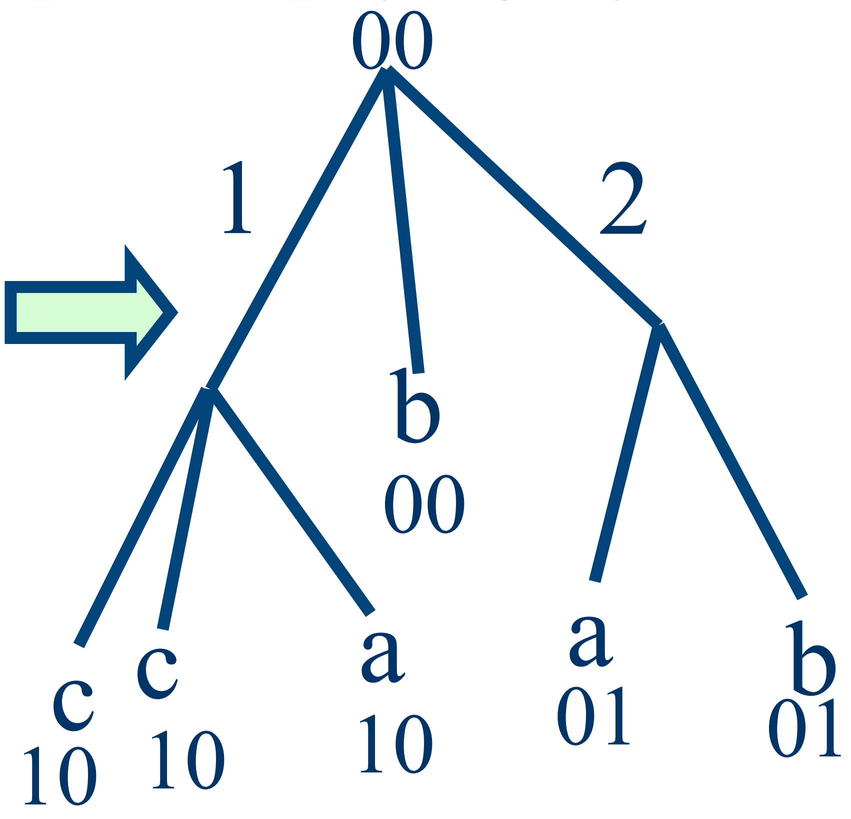
Given a set of genotypes, find an explaining set of haplotypes that fits a perfect phylogeny

	1	2
a	2	2
b	0	2
c	1	0

$\Rightarrow$

	1	2
a	1	0
a	0	1
b	0	0
b	0	1
c	1	0
c	1	0

$\Rightarrow$



The Alternative Explanation

	1	2
a	2	2
b	0	2
c	1	0

	1	2
a	1	1
a	0	0
b	0	0
b	0	1
c	1	0
c	1	0

No tree possible for this explanation

Efficient solutions to the PPH problem - n genotypes, m sites

- Reduction to a graph realization problem (GPPH) - build on Bixby-Wagner or Fushishige solution to graph realization O(nm α(nm)) time. (Gusfield 2002)
- Reduction to graph realization - build on Tutte's graph realization method O(nm^2) time.
- Direct, from scratch combinatorial approach - O(nm^2) (Bafna et al . 2003, Eskin et al. 2003)
- EHK approach - specialize the Tutte solution to the PPH problem - O(nm^2) time.

The case of the 1's

1) For any row i in S, the set of 1 entries in row i specify the exact set of mutations on the path from the root to the least common ancestor of the two leaves labeled i, in every perfect phylogeny for S.
2) The order of those 1 entries on the path is also the same in every perfect phylogeny for S, and is easy to determine by "leaf counting".

S

	1	2	3	4	5	6	7
a	1	0	1	0	0	0	0
b	0	1	0	1	0	0	0
c	1	2	0	0	2	0	2
d	2	2	0	0	0	2	0

Leaf Counting

In any column c, count two for each 1, and count one for each 2. The total is the number of leaves below mutation c, in every perfect phylogeny for S. So if we know the set of mutations on a path from the root, we know their order as well.

S

	1	2	3	4	5	6	7
a	1	0	1	0	0	0	0
b	0	1	0	1	0	0	0
c	1	2	0	0	2	0	2
d	2	2	0	0	0	2	0
Count	5	4	2	2	1	1	1

So Assume

The columns are sorted by leaf-count, largest to the left.

Similarly

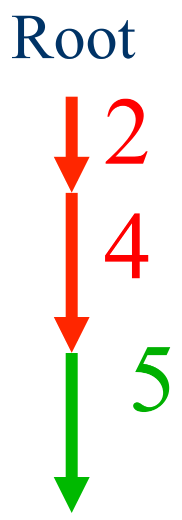
In any perfect phylogeny, the edge corresponding to the leftmost 2 in a row must be on a path just after the 1's for that row.

Simple conclusions

sites

	1	2	3	4	5	6	7
i:	0	1	0	1	2	2	2

Subtree for row i data



The order is known for the red mutations together with the leftmost blue mutation.

But what to do with the remaining blue entries (2’s) in a row?

More simple tools

3) For any row i in S, and any column c, if S(i,c) is 2, then in every perfect phylogeny for S, the path between the two leaves labeled i, must contain the edge with mutation c.

   Further, every mutation c on the path between the two i leaves must be from such a column c.

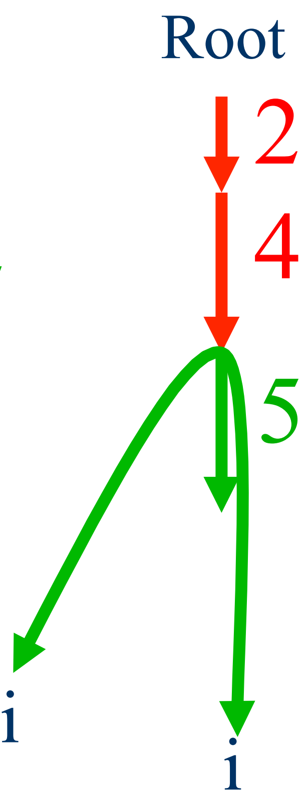
From row data to tree constraints

sites

1 2 3 4 5 6 7

i:0 1 0 1 2 2 2

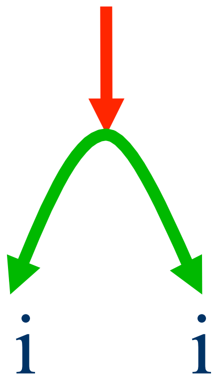
Subtree for row i data



Edges 5, 6 and 7
must be on the blue path,
and 5 is already known to
follow 4, but we don’t
where to put 6 and 7.

The graph theoretic problem

Given a genotype matrix S with n sites, and a red-blue subgraph for each row i,



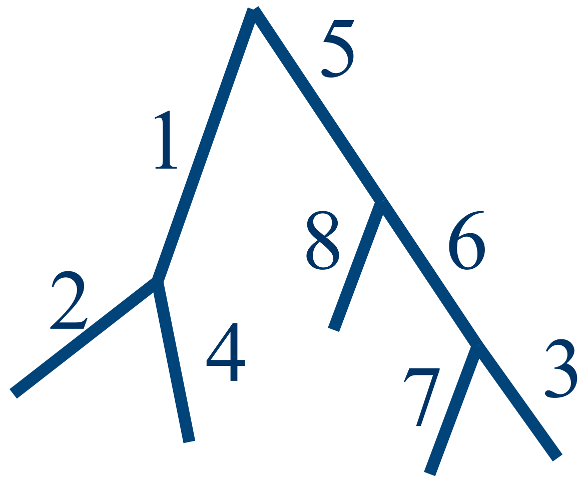
create a directed tree T where each integer from 1 to n labels exactly one edge, so that each subgraph is contained in T.

Powerful tool: graph realization

- Let Rn be the integers 1 to n, and let P be an unordered subset of Rn. P is called a path set.
- A tree T with n edges, where each is labeled with a unique integer of Rn, realizes P if there is a contiguous path in T labeled with the integers of P and no others.
- Given a family P1, P2, P3…Pk of path sets, tree T realizes the family if it realizes each Pi.
- The graph realization problem generalizes the consecutive ones problem, where T is a path.

Graph realization example

P1: 1, 5, 8
P2: 2, 4
P3: 1, 2, 5, 6
P4: 3, 6, 8
P5: 1, 5, 6, 7



Realizing Tree T

Graph realization

Polynomial time (almost linear-time) algorithms exist for the graph realization problem – Whitney, Tutte, Cunningham, Edmonds, Bixby, Wagner, Gavril, Tamari, Fushishige, Lofgren 1930's - 1980's

The algorithms are not simple; none implemented before 2002.

Recognizing graphic Matroids

- The graph realization problem is the same problem as determining if a binary matroid is graphic, and the algorithms come from that literature.
- The fastest algorithm is due to Bixby and Wagner
- Representation methods due to Cunningham et al.

Reducing PPH to graph realization

We solve any instance of the PPH problem by creating appropriate path sets, so that a solution to the resulting graph realization problem leads to a solution to the PPH problem instance.

The key issue: How to encode the needed subgraph for each row, and glue them together at the root.

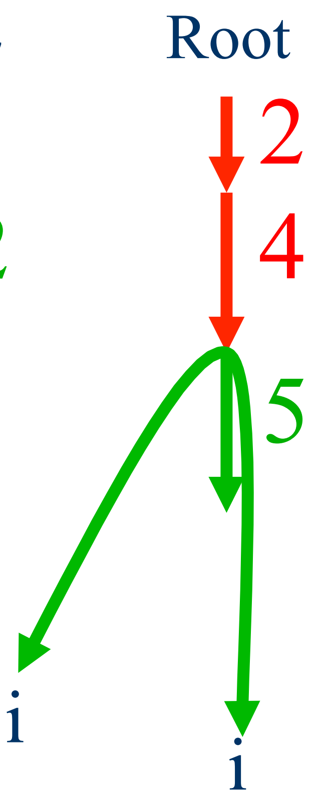
From raw data to tree constraints

sites

1 2 3 4 5 6 7

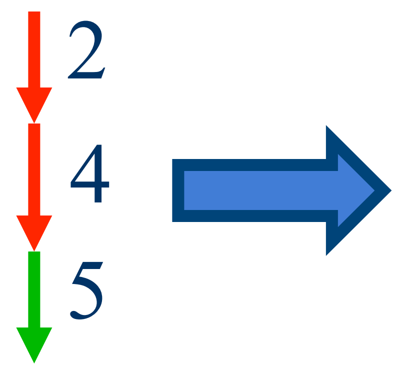
i:0 1 0 1 2 2 2

Subtree for row i data

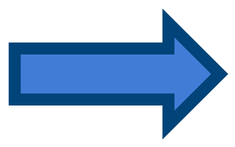


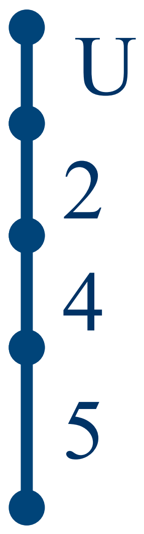
Edges 5, 6 and 7
must be on the blue path,
and 5 is already known to
follow 4.

Encoding a Red-Blue directed path



P1: U, 2
P2: U, 2, 4
P3: 2, 4
P4: 2, 4, 5
P5: 4, 5





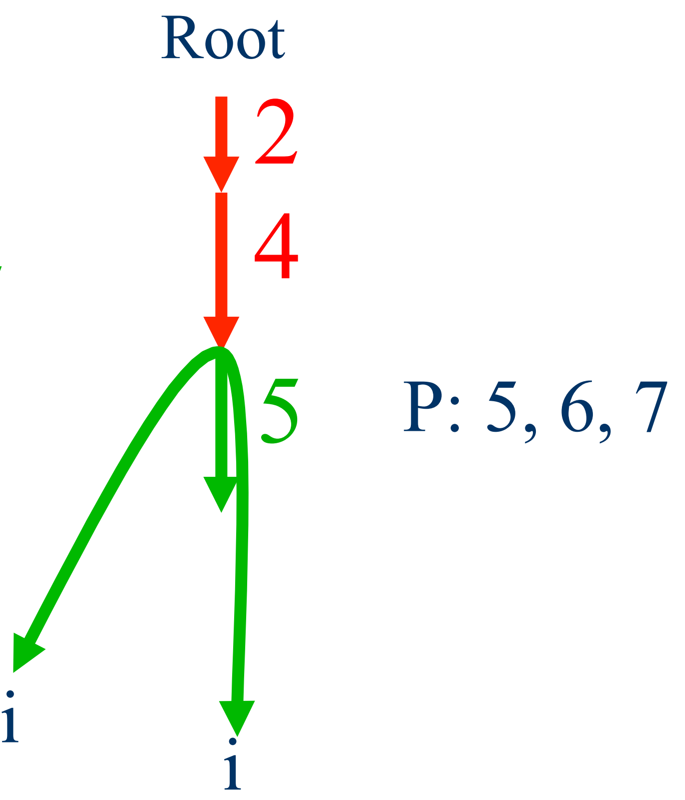
U is a glue edge used to glue together the directed paths from the different rows.

Now add a path set for the blues in row i.

sites

1 2 3 4 5 6 7

i:0 1 0 1 2 2 2



That' s the reduction

The resulting path-sets encode everything that is known about row i in the input.

The family of path-sets are input to the graph-realization problem, and every solution to the that graph-realization problem specifies a solution to the PPH problem, and conversely.

But how is graph realization solved?

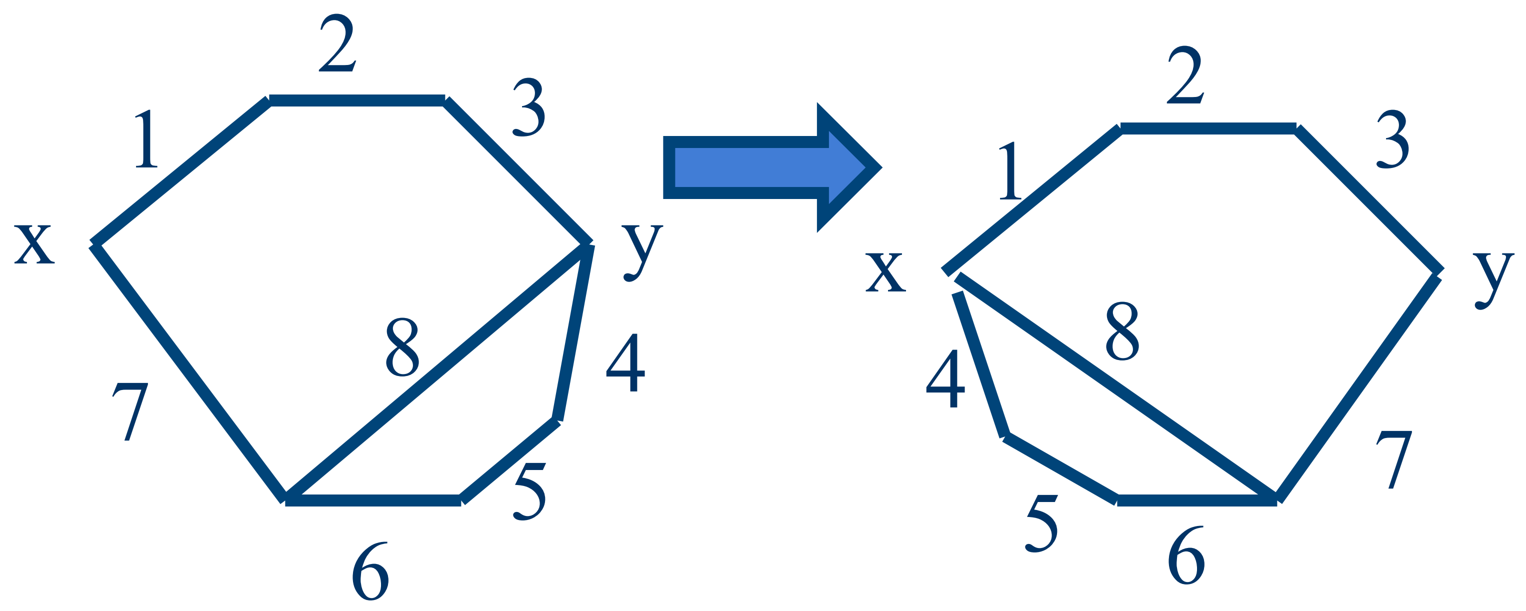
Tutte’s Algorithm for Graph Realization, given a partial solution T.

- Pick an unpicked edge e.
- Determine any other edges that must be on one particular side of e or the other.
- Determine any pair of edges that must be on opposite sides of e. Form a graph G with an edge between any such pair - test if bipartite. If so, assign one side of G to one side of e, and the other side of G to the other side of e.
- Apply the decisions, modifying T, and recursion.

Multiple solutions

- In 1933, Whitney showed how multiple solutions to the graph realization problem are related.
- Partition the edges of G into two connected graphs, each with at least two edges, such that the two graphs have exactly two nodes in common. Then twist one graph around those two nodes.
- Any solution can be transformed to another via a series of such twists.

Twisting Example



All the cycle sets are preserved by twisting

Representing all the solutions

All the solutions to the PPH problem can be implicitly represented in a data structure which can be built in linear time, once one solution is known. Then each solution can be generated in linear time per solution. Method is a small modification of ones developed by Cunningham and Edwards, and by Hopcroft and Tarjan.

The DPPH Method

- Bafna et al. $O(nm^2)$ time
- Based on deeper combinatorial observations about the PPH problem.
- A matrix-centric approach (rather than tree-centric), although a graph is used in the algorithm.

First, we need to understand why some sets of haplotypes have a perfect phylogeny, and some do not.

When does a set of haplotypes fit a perfect phylogeny?

Arrange the haplotypes in a matrix, two haplotypes for each individual. Then (with no duplicate columns), the haplotypes fit a unique perfect phylogeny if and only if no two columns contain all three pairs:

0,1 and 1,0 and 1,1

This is called the 3-Gamete Test

The Alternative Explanation

	1	2
a	2	2
b	0	2
c	1	0

$\Rightarrow$

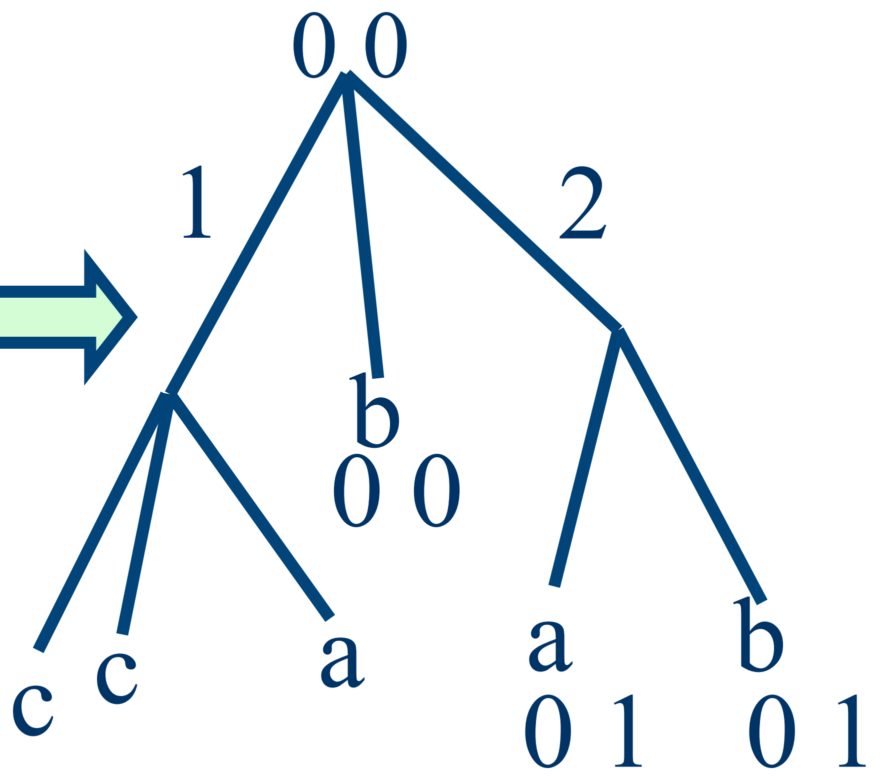
	1	2
a	1	1
a	0	0
b	0	0
b	0	1
c	1	0
c	1	0

$\Rightarrow$

No tree possible for this explanation

	1	2
a	2	2
b	0	2
c	1	0

	1	2
a	1	0
a	0	1
b	0	0
b	0	1
c	1	0
c	1	0



PPH: The combinatorial problem

Input: A ternary matrix (0,1,2) M with 2N rows partitioned into N pairs of rows, where the two rows in each pair are identical.

Def: If a pair of rows (r,r') in the partition have entry values of 2 in a column j then positions (r,j) and (r' ,j) are called Mates.

Output: A binary matrix M' created from M by replacing each 2 in M with either 0 or 1, such that

a) A position is assigned 0 if and only if its Mate is assigned 1.

b) M' passes the 3-Gamete Test, i.e., does not contain a 3x2 submatrix (after row and column permutations) with all three combinations 0,1; 1,0; and 1,1

Initial observations

If two columns of M contain the following rows

```
2 0 |
2 0 | mates
0 2 |
0 2 | mates
```

then M' will contain a row with 1 0 and a row with 0 1 in those columns.

This is a forced expansion.

Initial observations

Similarly, if two columns of M contain the mates

2 1

2 1

then M' will contain a row with 1 1 in those columns.

This is a forced expansion.

If a forced expansion of two columns creates $\begin{matrix}0 & 1\\ 1 & 0\end{matrix}$ in those columns, then any $\begin{matrix}2 & 2\\ 2 & 2\end{matrix}$ in those columns must be set to be

$$\begin{matrix}0 & 1\\ 1 & 0\end{matrix}$$

We say that two columns are forced out-of-phase.

If a forced expansion of two columns creates $\begin{matrix}1 & 1\end{matrix}$ in those columns, then any $\begin{matrix}2 & 2\\ 2 & 2\end{matrix}$ in those columns must be set to be

$$\begin{matrix}1 & 1\\ 0 & 0\end{matrix}$$

We say that two columns are forced in-phase.

	1	2	3
a	1	2	2
a	1	2	2
b	2	0	2
b	2	0	2
c	1	2	2
c	1	2	2
d	1	2	2
d	1	2	2
e	2	2	0
e	2	2	0

Example:

Columns 1 and 2, and 1 and 3 are forced in-phase.
Columns 2 and 3 are forced out-of-phase.

Immediate failure

It can happen that the forced expansion of cells creates a 3x2 submatrix that fails the 3-Gamete Test. In that case, there is no PPH solution for M.

Example:

```
20
20      Will fail the 3-Gamete Test
11
11
02
02
```

An O(nm^2)-time Algorithm

- Find all the forced phase relationships by considering columns in pairs.
- Find all the inferred, invariant, phase relationships.
- Find a set of column pairs whose phase relationship can be arbitrarily set, so that all the remaining phase relationships can be inferred.
- Result: An implicit representation of all solutions to the PPH problem.

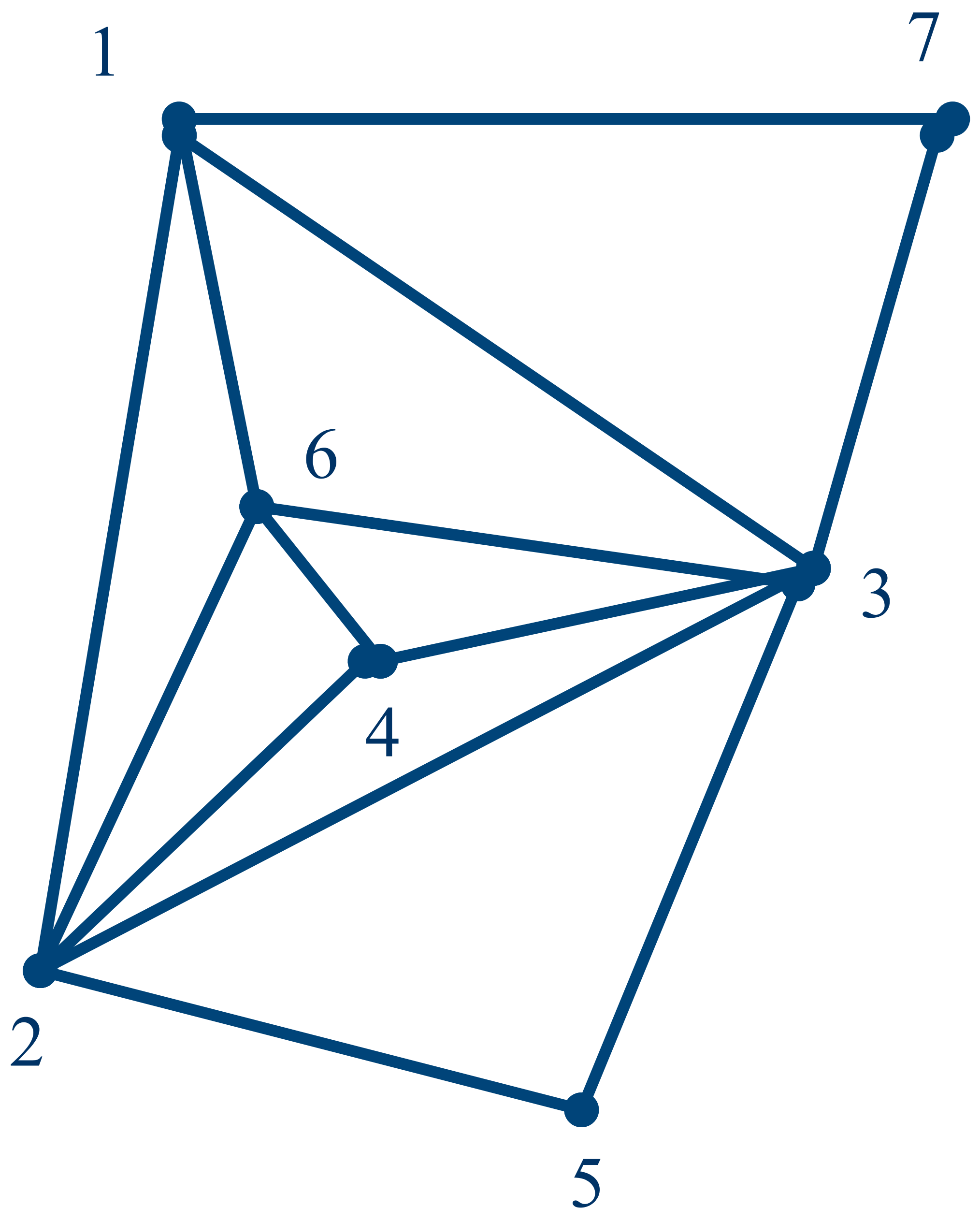
	1	2	3	4	5	6	7
a	1	2	2	2	0	0	0
a	1	2	2	2	0	0	0
b	2	0	2	0	0	0	2
b	2	0	2	0	0	0	2
c	1	2	2	2	0	2	0
c	1	2	2	2	0	2	0
d	1	2	2	0	2	0	0
d	1	2	2	0	2	0	0
e	2	2	0	0	0	2	0
e	2	2	0	0	0	2	0

A running example.

Overview of Bafna et al. algorithm

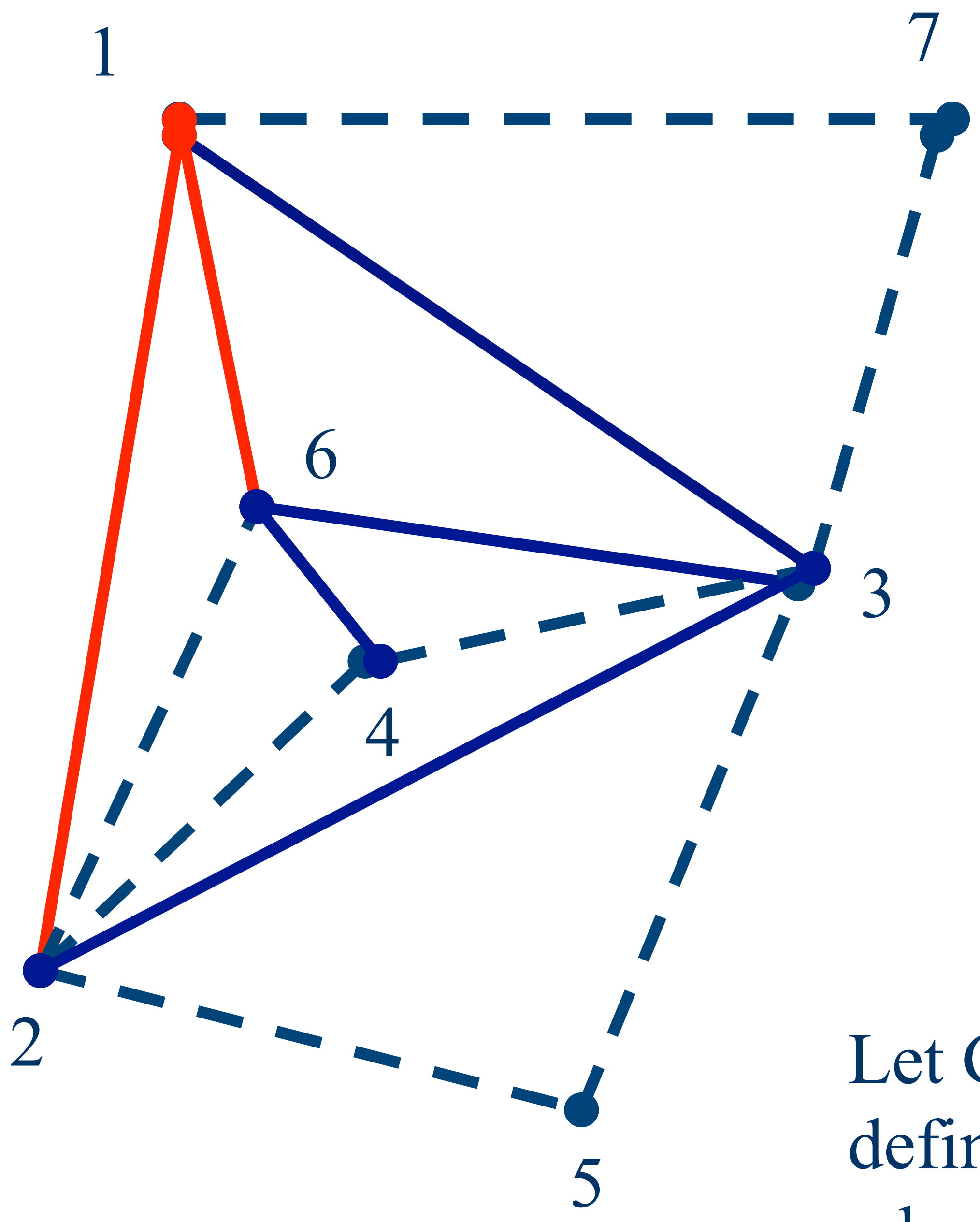
First, represent the forced phase relationships, and the needed decisions, in a graph G.

Graph G



Each node represents a column in M, and each edge indicates that the pair of columns has a row with 2's in both columns.

The algorithm builds this graph, and then checks whether any pair of nodes is forced in or out of phase.

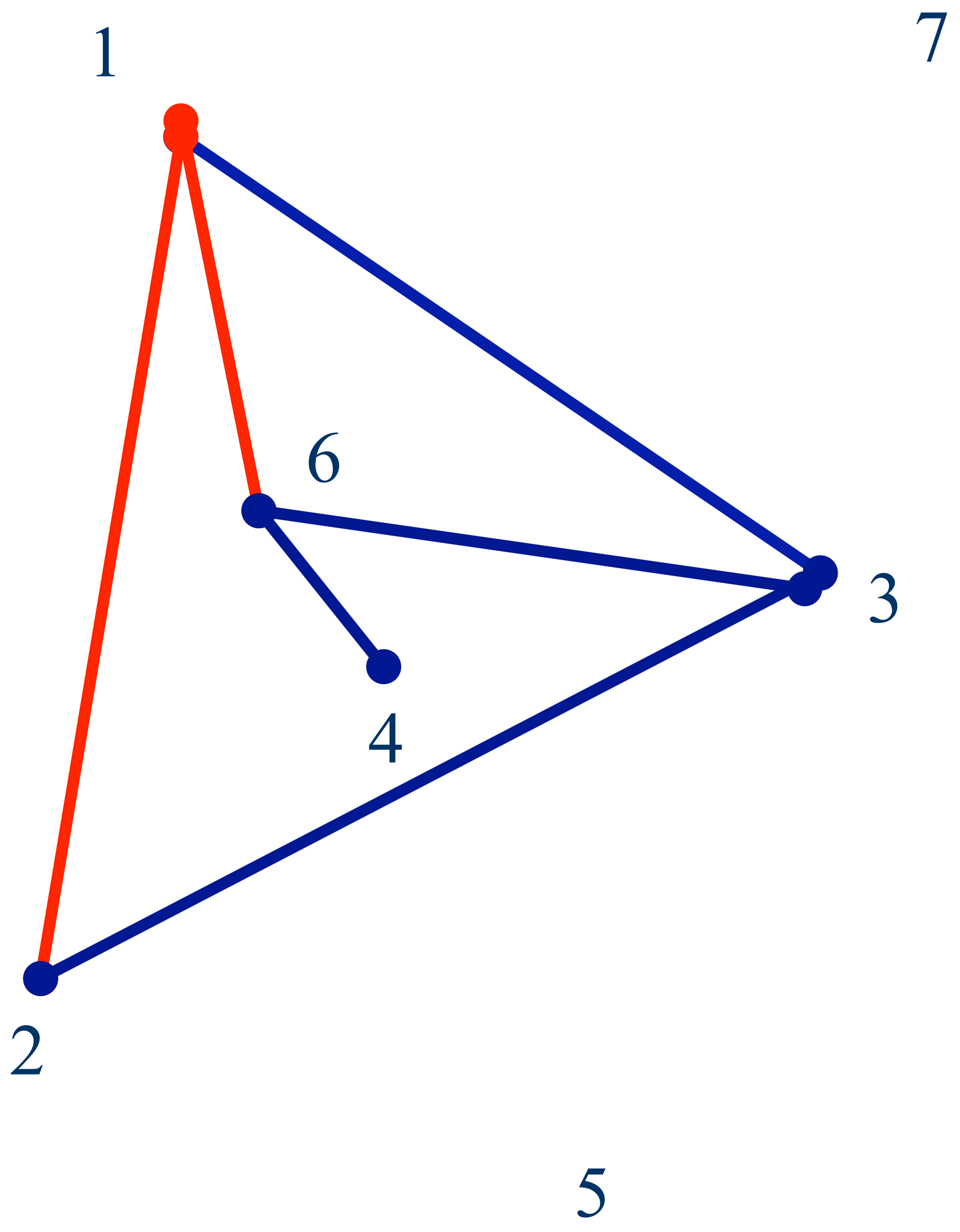


Graph Gc

Each Red edge indicates that the columns are forced in-phase.

Each Blue edge indicates that the columns are forced out-of-phase.

Let Gf be the subgraph of Gc defined by the red and blue edges.



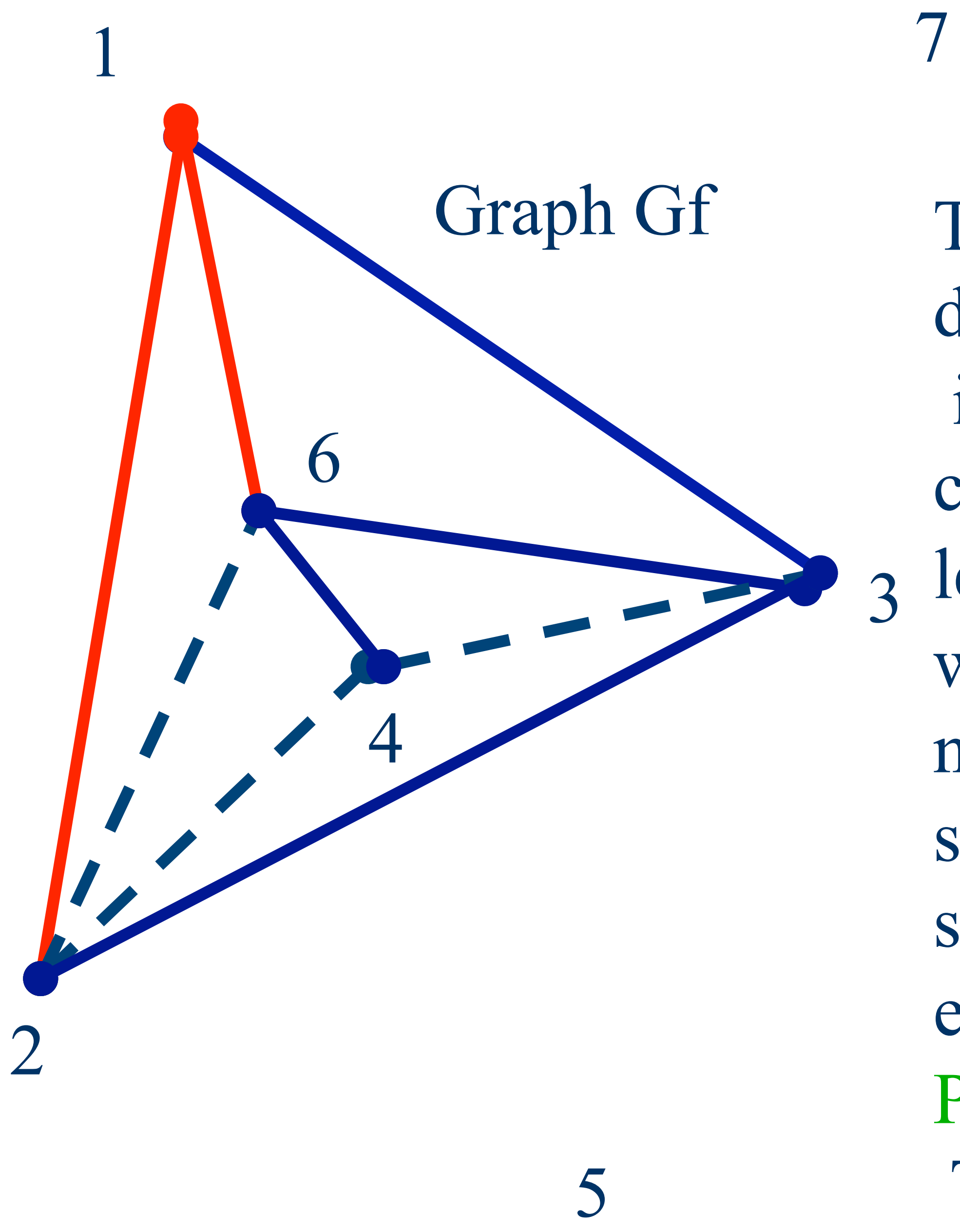
Graph Gf has three connected components.

The central theorem

There is a solution to the PPH problem for M if and only if there is a coloring of the dashed edges of Gc with the following property:

For any triangle (i,j,k) in Gc, where there is one row containing 2's in all three columns i,j and k (any triangle containing at least one dashed edge will be of this type), the coloring makes either 0 or 2 of the edges blue (out-of-phase).

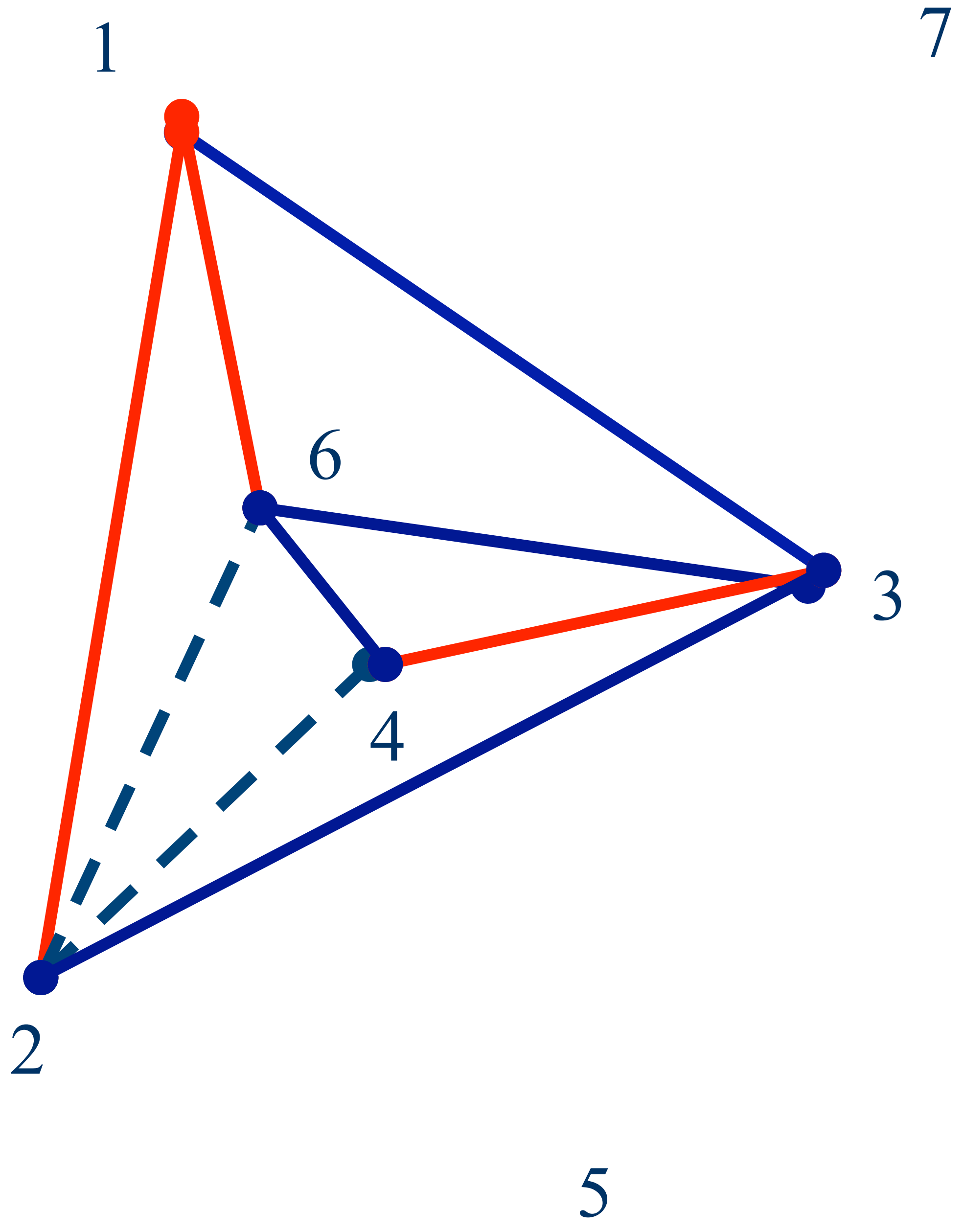
Nice, but how do we find such a coloring?

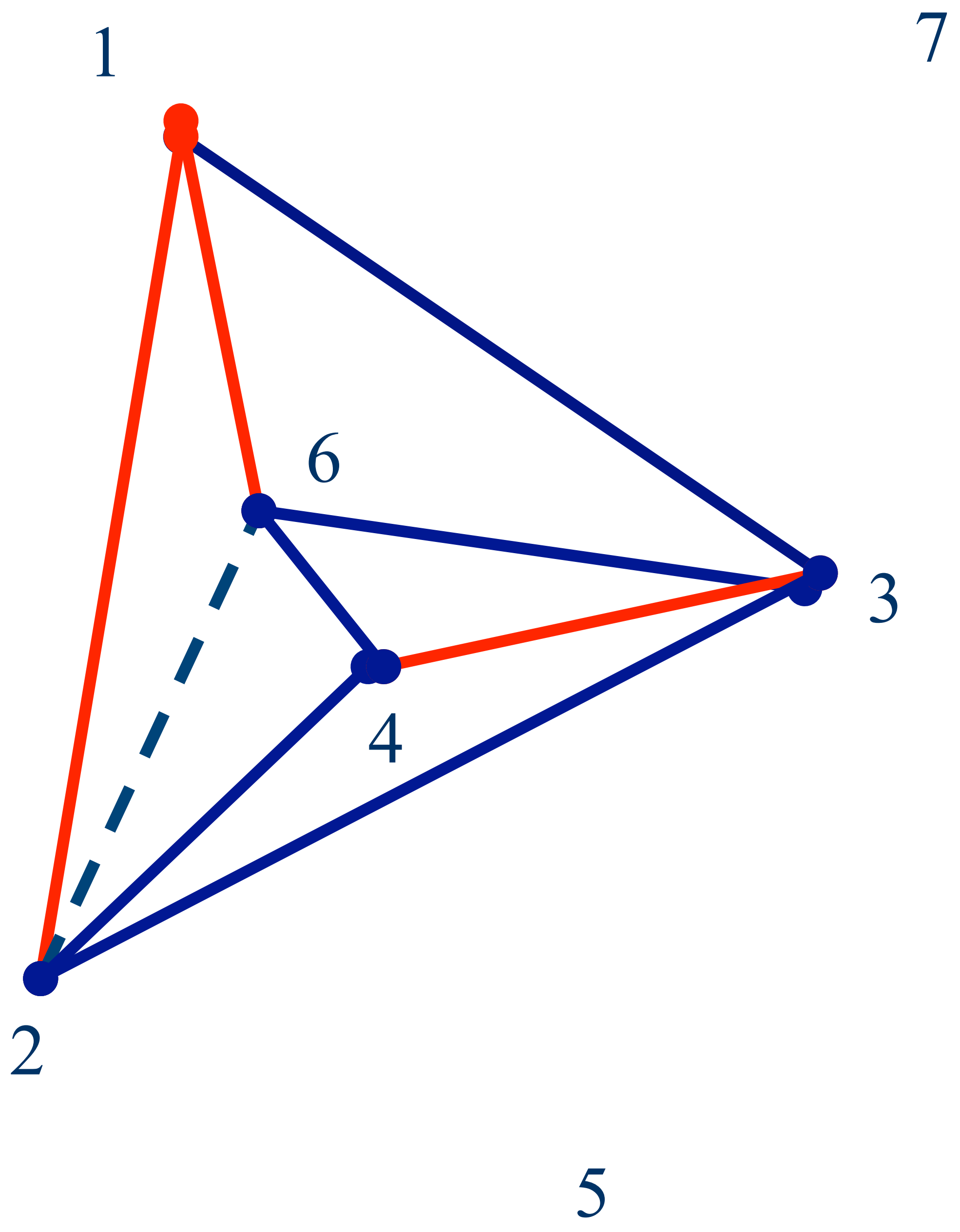


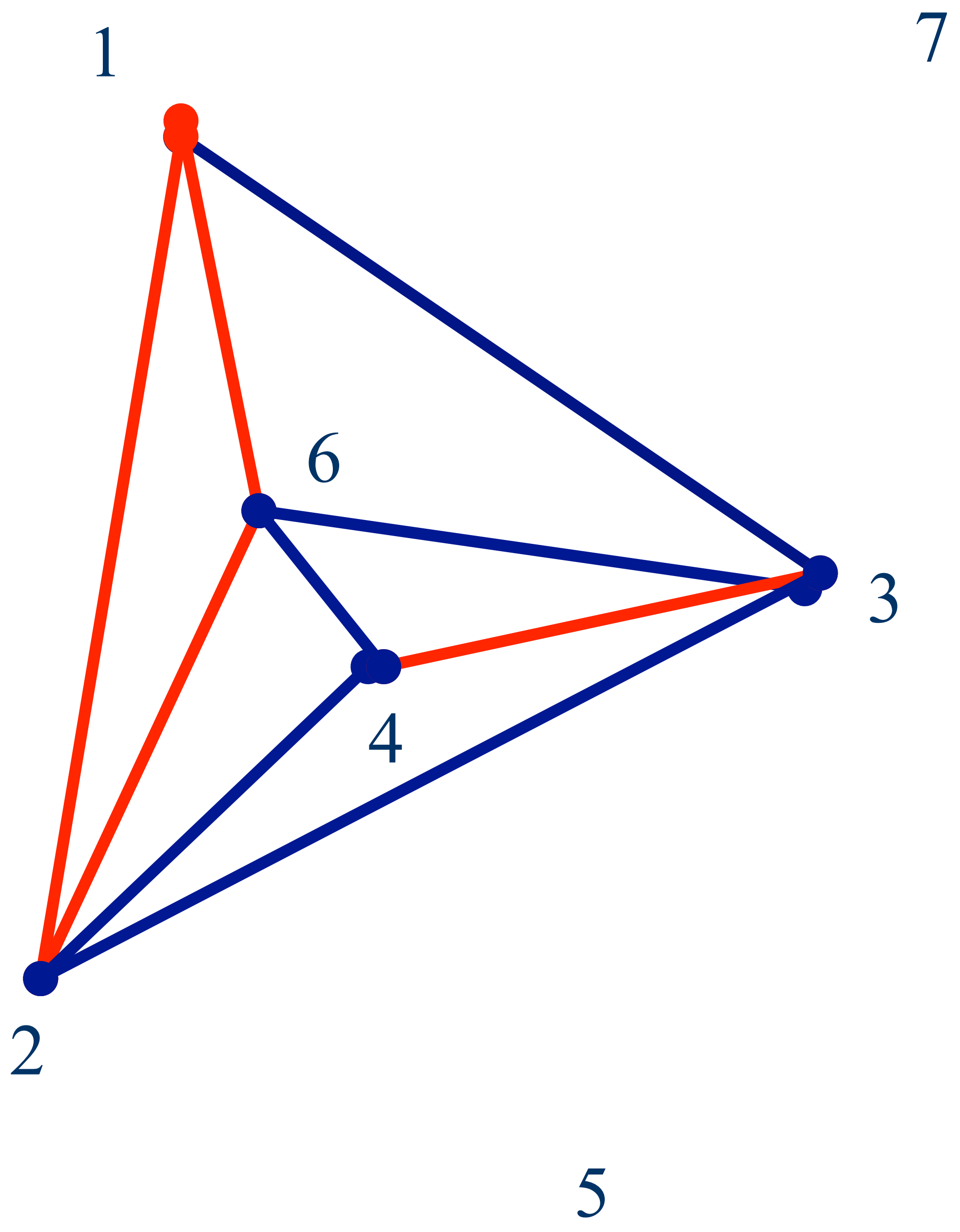
Triangle Rule

Theorem 1: If there are any dashed edges whose ends are in the same connected component of Gf, at least one edge is in a triangle where the other edges are not dashed, and in every PPH solution, it must be colored so that the triangle has an even number of Blue (out of Phase) edges.

This is an "inferred" coloring


1
7
6
3
4
2
5

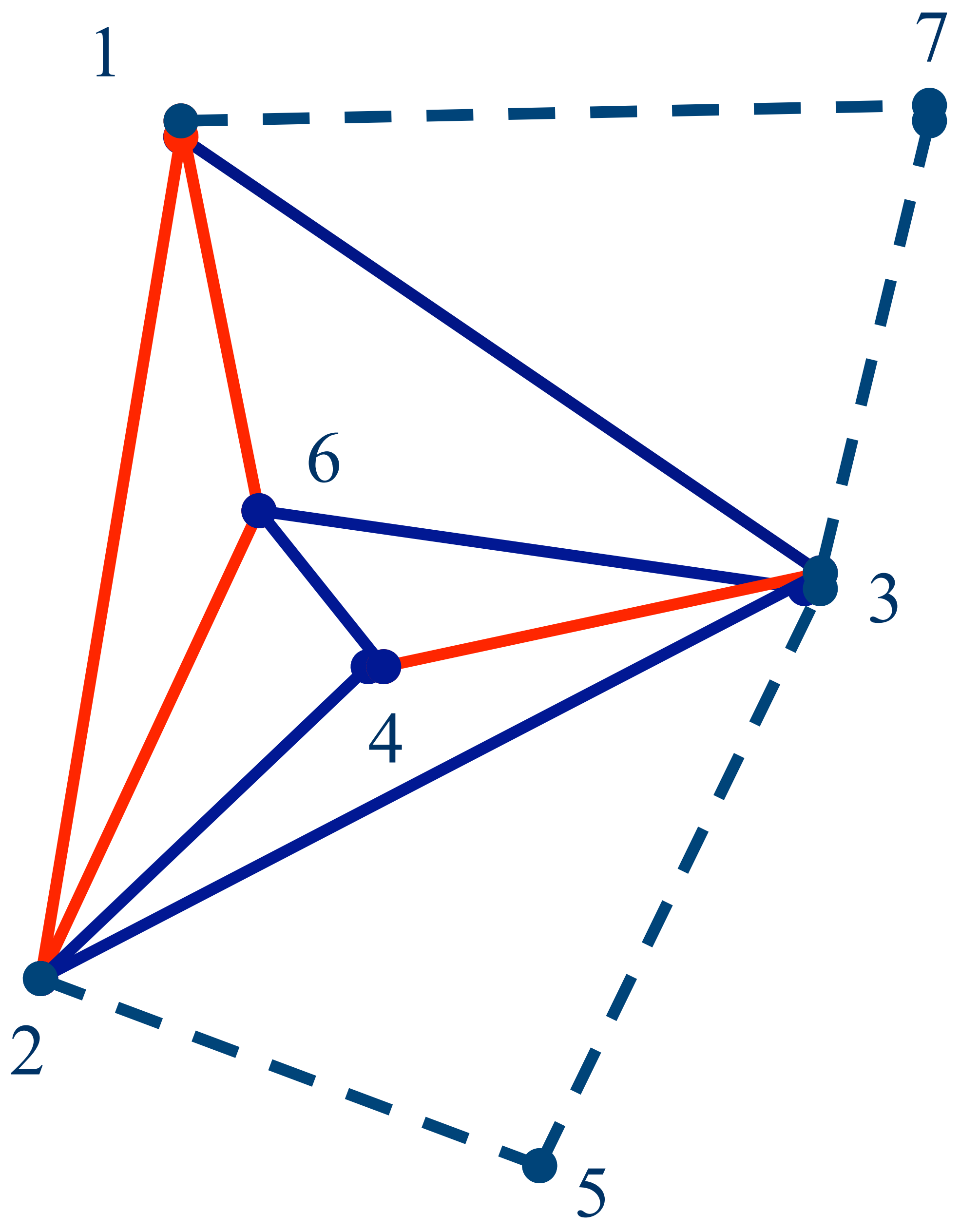

1
7
6
3
4
2
5


1
7
6
3
4
2
5

Corollary

Inside any connected component of Gf, ALL the phase relationships on edges (columns of M) are uniquely determined, either as forced relationships based on pairwise column comparisons,
or by triangle-based inferred colorings.

Hence, the phase relationships of all the columns in a connected component of Gf are INVARIANT over all the solutions to the PPH problem.



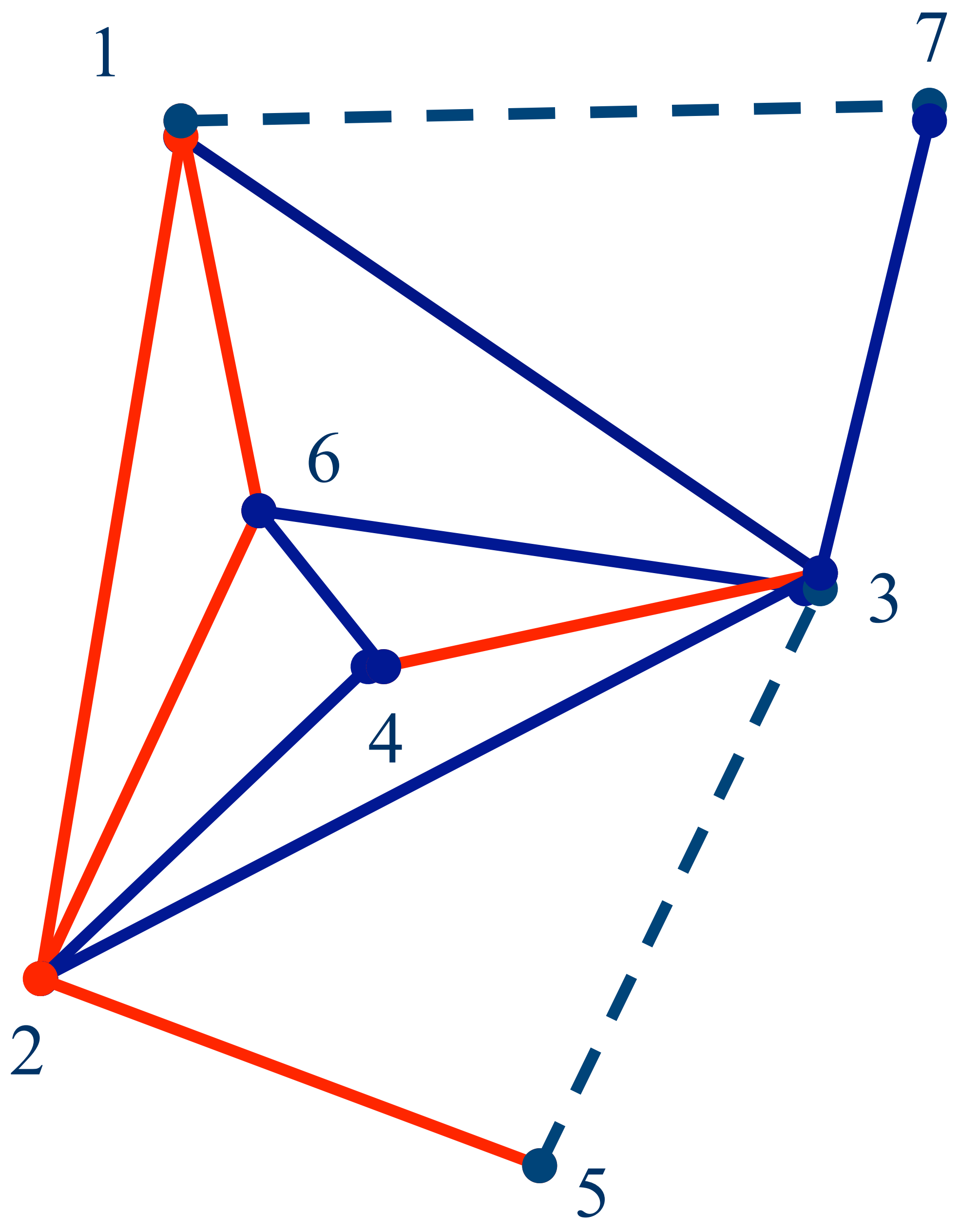
How should we color the remaining dashed edges in a connected component C of Gc?

Answer

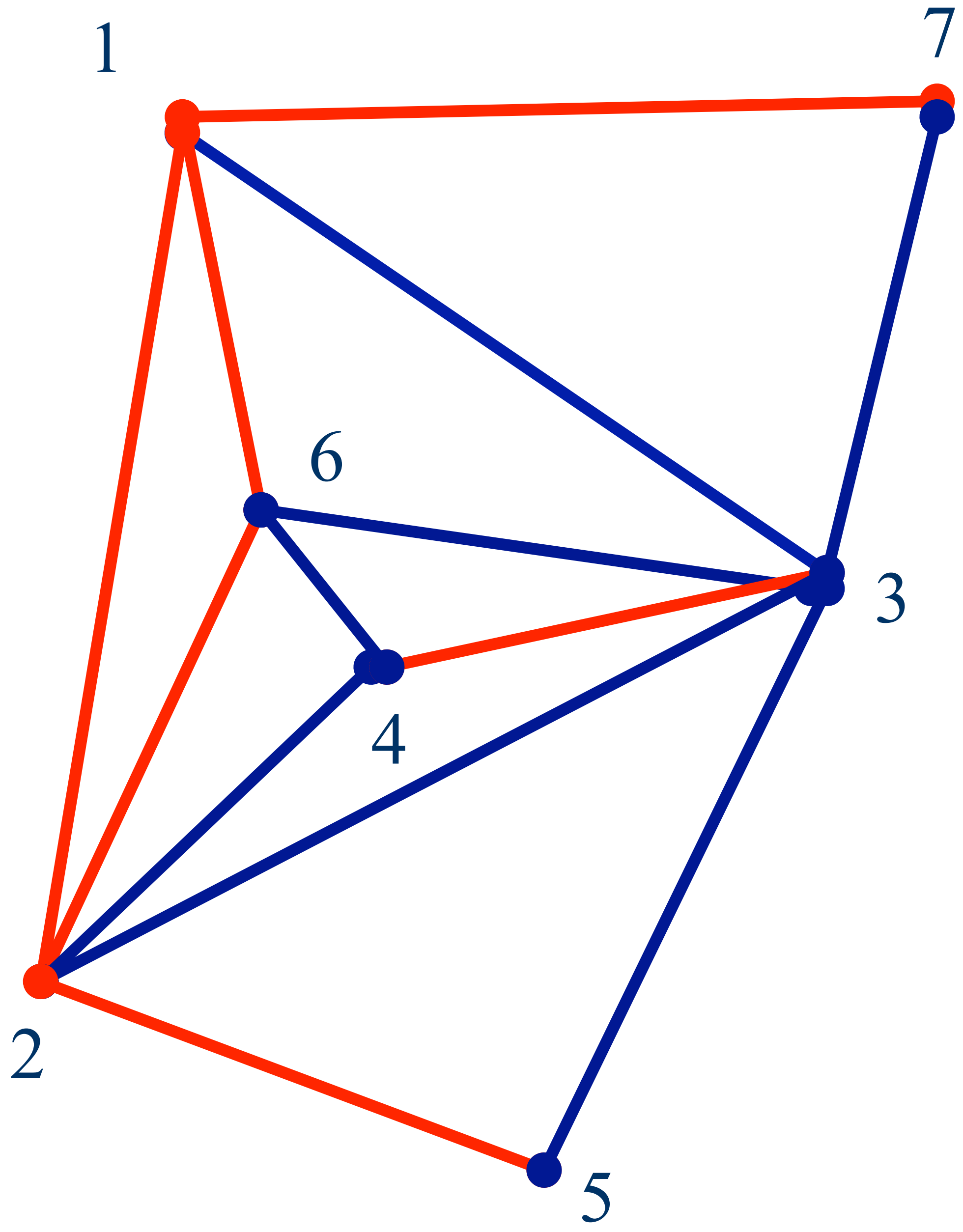
For a connected component C of G with k connected components of Gf, select any
subset S of k-1 dashed edges in C, so that S
together with the red and blue edges span all the nodes of C.

Arbitrarily, color each edge in S either red or blue.

Infer the color of any remaining dashed edges by successive use of the triangle rule.



Pick and color edges (2,5) and (3,7)
The remaining dashed edges are colored by using the triangle rule.


1
7
6
3
4
2
5

Theorem 2

- Any selected S works (allows the triangle rule to work) and any coloring of the edges in S determines the colors of any remaining dashed edges.
- Different colorings of S determine different colorings of the remaining dashed edges.
- Each different coloring of S determines a different solution to the PPH problem.
- All PPH solutions can be obtained in this way, i.e. using just one selected S set, but coloring it in all 2^{k-1} ways.