

WEIGHTED BIPARTITE MATCHING

Input: Weighted bipartite graph $G = (V, E, w)$
 $w: E \rightarrow \mathbb{Z}^+$

Output: Matching $M \subseteq E'$ such that

$$\sum_{e \in M} w(e) \text{ is maximum.}$$

(We showed the special case
with unit weights, $w(e) = 1 \forall e \in E$).

Hopcroft & Karp (1973)

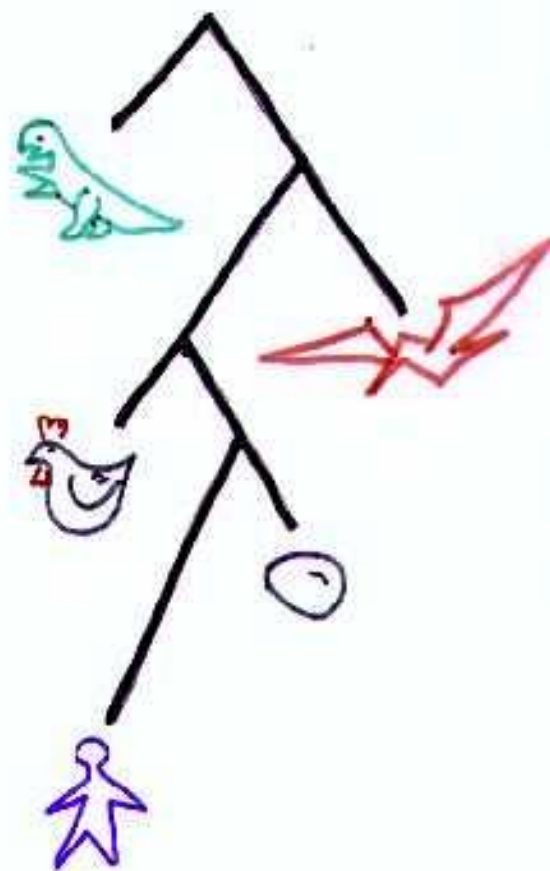
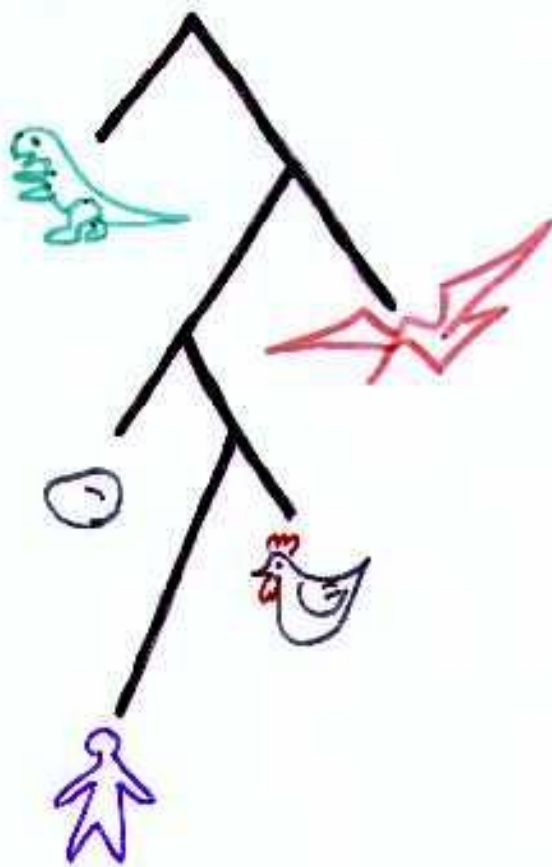
Showed $O(n^{5/2})$ algorithm

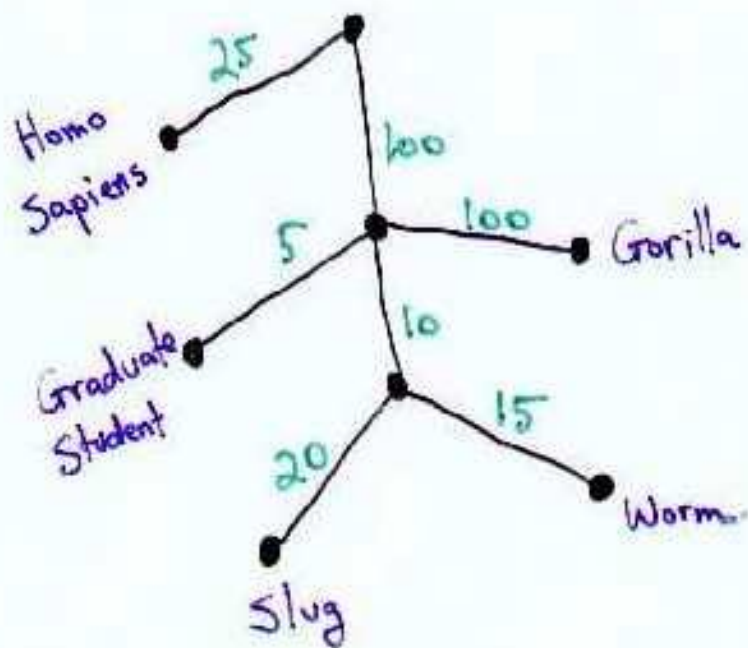
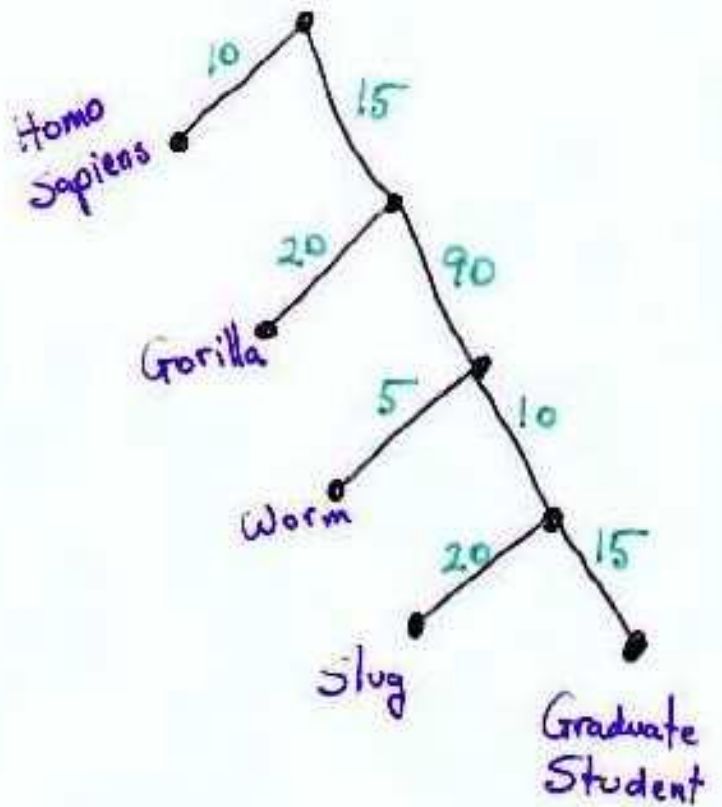
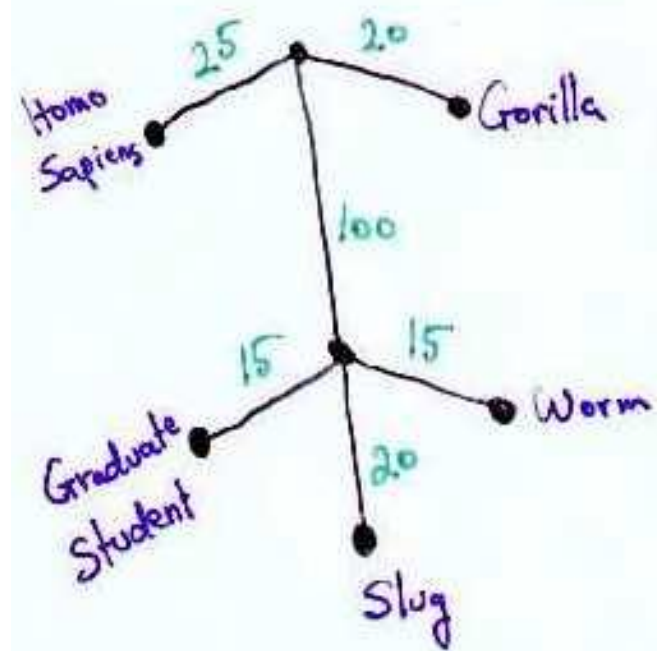
for maximum weighted bipartite matching.

(SIAM J. Computing, 1973, 225-231).

USES OF MATCHING

Famous Open Problem...

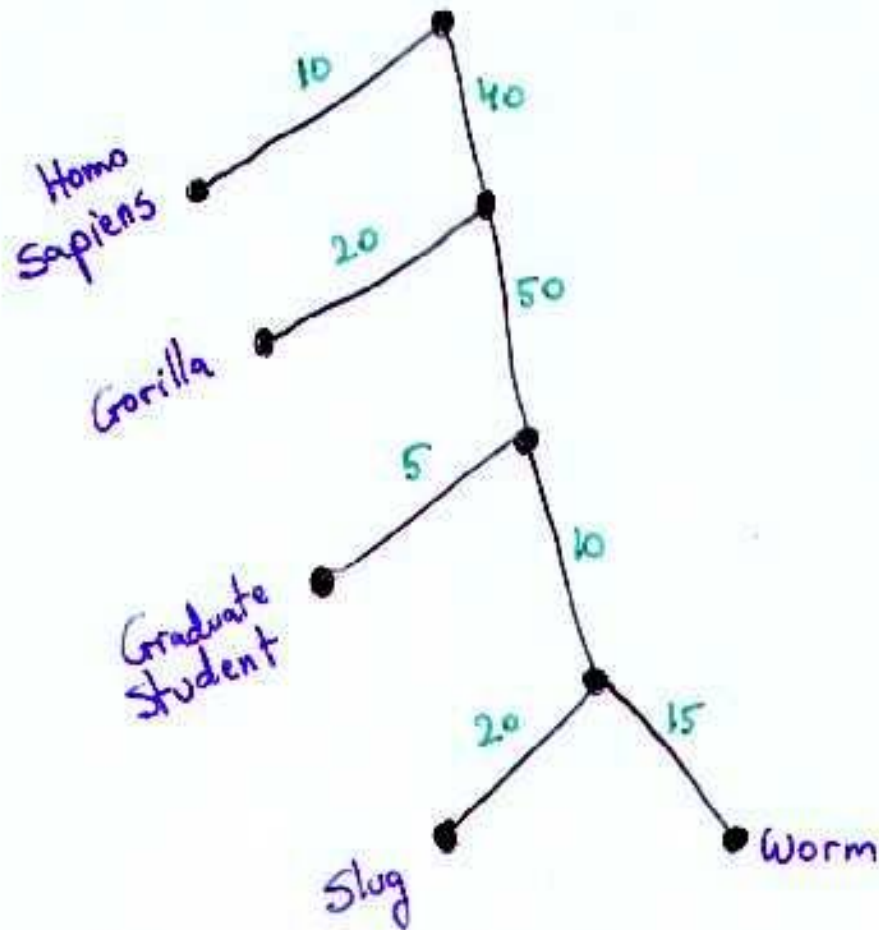




How do we reconcile?

Construct tree "closest" to all.

(e.g. Neumann 83)

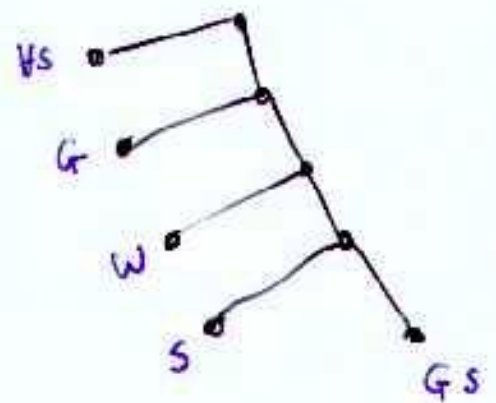
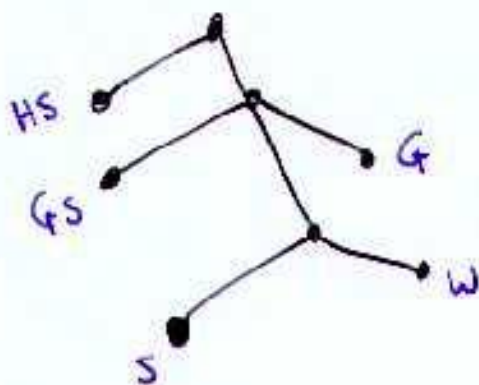


MAXIMUM HOMEOMORPHIC

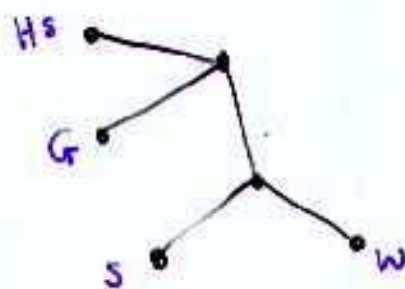
AGREEMENT SUBTREE (MHT)

(Findey-Gordon 85)

Find maximum set of leaves such that minimal homeomorphic subtrees (contract degree-2 nodes) with those leaves are all equal.



MHT



FORMALLY:

INPUT: k trees.

Each has n leaves

labeled by elements from

$$S = \{s_1, s_2, \dots, s_n\}$$

No 2 leaves in a tree have equal label.

OUTPUT: Maximum $S' \subseteq S$ such

that the restrictions of all trees to subtree with labels from S' are homeomorphic.

Naive Solution

- For every subset $S' \subseteq S$ and every tree, construct restriction, check if homeomorphic.
- For all agreements, choose maximum.

Time: $O(n2^n)$

AWFUL!!!

Finden & Gordon (1985) $O(n^3)$ heuristic. Optimal tree not guaranteed.
MHT of 2 trees.

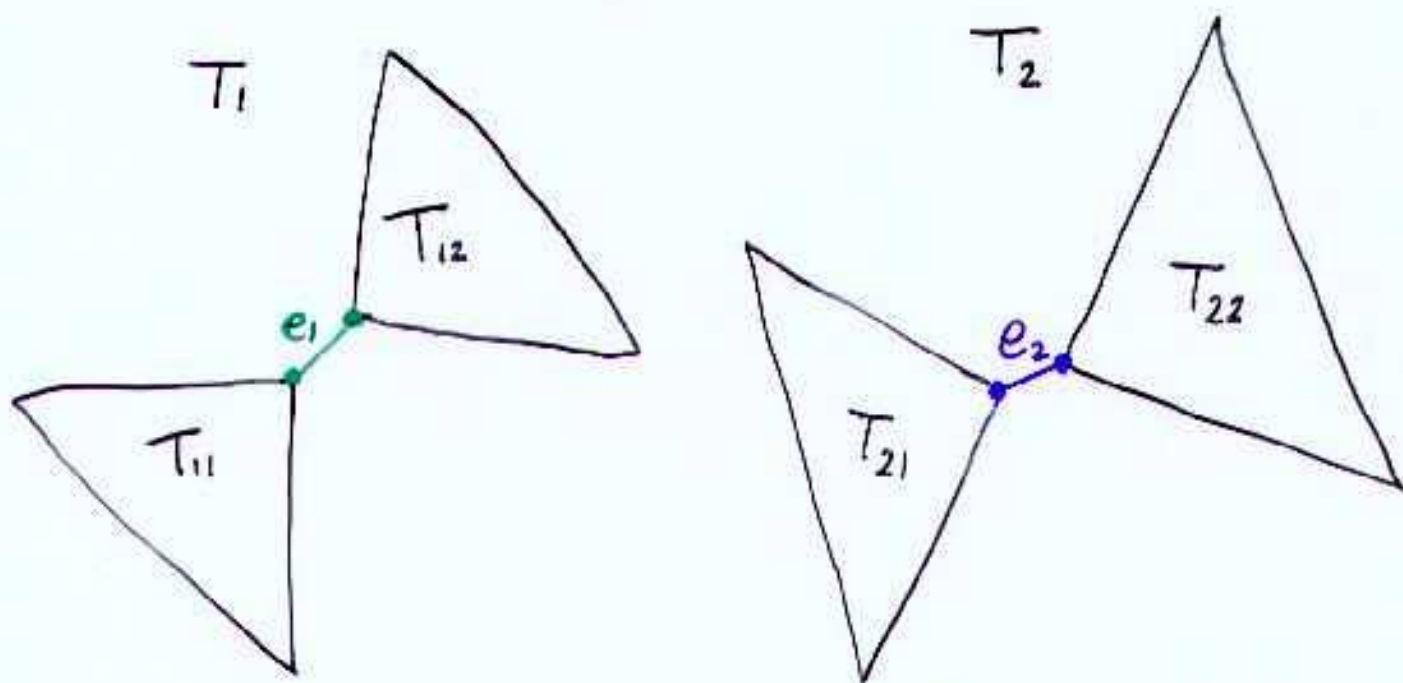
Kubicka, Kubicki & McMorris (1992) $O(n^{\frac{1}{2} + \epsilon \log_2 n})$
MHT of 2 trees.

MHT of 2 Trees

(Steel & Warnow, 1993)

IDEA: Dynamic Programming

choose edge $e_1 \in T_1$ $e_2 \in T_2$



Consider $M = \max$ (
 $MHT(T_{11}, T_{21}) \cup MHT(T_{12}, T_{22})$,
 $MHT(T_{11}, T_{22}) \cup MHT(T_{12}, T_{21})$)

We will prove that an appropriate choice of e_1, e_2 makes $M = MHT(T_1, T_2)$.

We need to: order all rooted subtrees
(a root, its 2 children & all their
descendants.)

How many such rooted subtrees?

Lemma: Let T be an n -leaf binary
tree with the property that
every node has either degree
 1 (leaf) or 3 .

Then T has at most $2n-1$ nodes.

Proof: By induction on # of leaves (exercise) ■

In our case: n leaves \Rightarrow
 $2n-1$ nodes \Rightarrow
 $O(n)$ trees.

Natural Order: $A \leq B$ iff A is a subtree of B .

Do topological sort on trees to get a total order.

We are ready to construct the dynamic programming table.

MHT	T_1^1	T_1^2	T_1^3	...	$T_1^{n_1}$
T_2^1					
T_2^2					
T_2^3					
⋮					
$T_2^{n_2}$					

We fill up the matrix as follows:

- For every (P, Q) where $|P| \neq 1$ ($P = \{p\}$)

$$\text{MHT}(P, Q) \leftarrow \begin{cases} 1 & \text{if } p \in Q \\ 0 & \text{otherwise} \end{cases}$$

So we can easily fill first n rows.

- Cell (P, Q) where P has 2 children p_1, p_2 and Q has 2 children q_1, q_2

is filled as follows:

Let P_i be the subtree rooted at p_i

Q_i the subtree rooted at q_i

$i=1, 2$.

$$\begin{aligned} \text{MHT}(P, Q) \leftarrow \max & \left(\text{MHT}(P_1, Q_1) \cup \text{MHT}(P_2, Q_2), \right. \\ & \text{MHT}(P_1, Q_2) \cup \text{MHT}(P_2, Q_1), \\ & \text{MHT}(P, Q_1), \text{MHT}(P, Q_2), \\ & \left. \text{MHT}(P_1, Q), \text{MHT}(P_2, Q) \right). \end{aligned}$$

• For the entire trees T_1, T_2 (unrooted):

For every pair of edges $e_1 \in T_1$
 $e_2 \in T_2$

which split T_1 to T_{11}, T_{12}
and T_2 to T_{21}, T_{22}

compute

$\max (MHT(T_{11}, T_{21}) \cup MHT(T_{12}, T_{22}),$
 $MHT(T_{11}, T_{22}) \cup MHT(T_{12}, T_{21}))$

choose maximum over all pairs of edges.

Implementation Note:

Both the table and entire tree handling may just keep track of the sizes of the MHT's with pointers that allow computing MHT at the end.

Time:

1) Matrix size is $O(n^2)$.

- Cells where both trees have more than one element are clearly computed in constant time.

- One-element trees require a "find" operation. Since matrix size is $O(n^2)$ we can afford construction of bit-vector for every set (space & time $O(n^2)$). "finds" are now also constant time.

2) In final stage, constant work per pair of edges, $O(n^2)$ pairs.

3) Constructing $O(n)$ subtrees:

Start from singletons and do "unions".

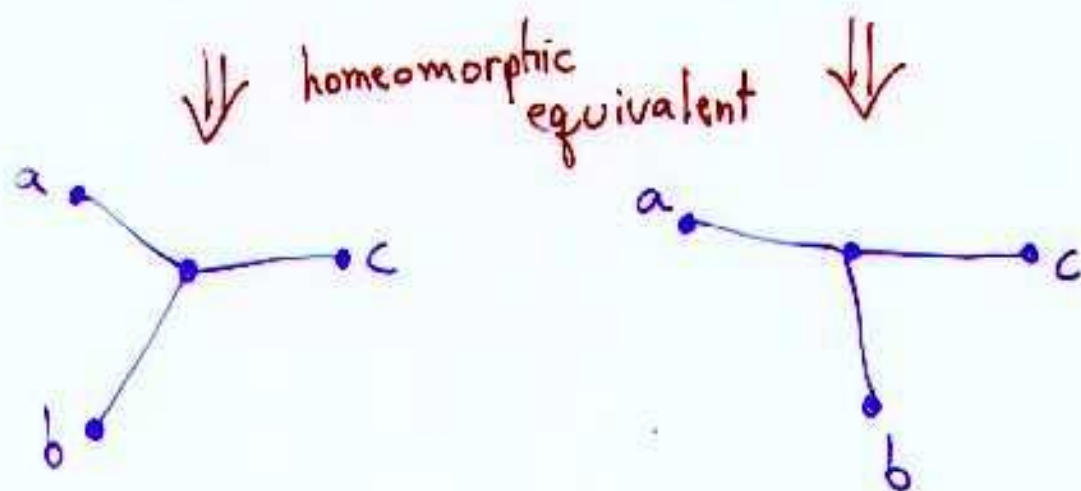
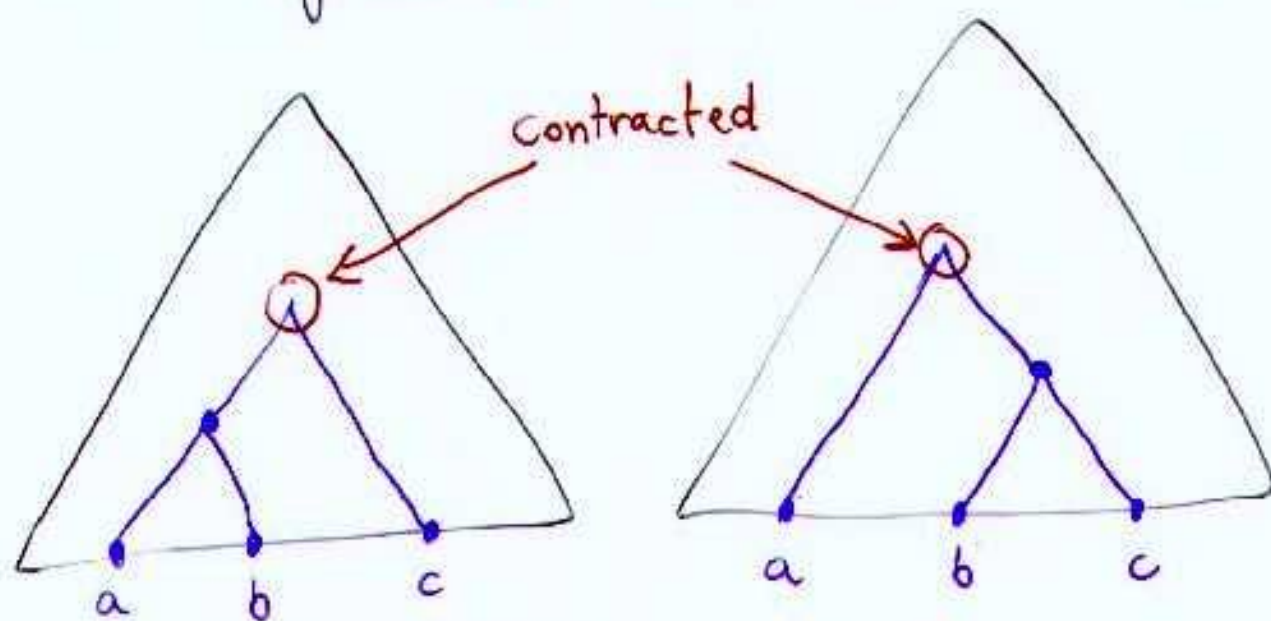
$O(n \log n)$	straightforward	} union-find algorithm.
$O(n \log^* n)$	Ullman	
$O(n \alpha(n))$	Tarjan	

Total Time: $O(n^2)$

Correctness: Need to show that MHT is found.

Note: For any two trees with $n \geq 3$, the MHT has at least 3 leaves.

Reason: Any triple creates a homeomorphic agreement subtree.



Consider $A = \text{MHT}(T_1, T_2)$.

- By note, there is at least one edge e that splits A into two subtrees, creating a bipartition of A 's labels.

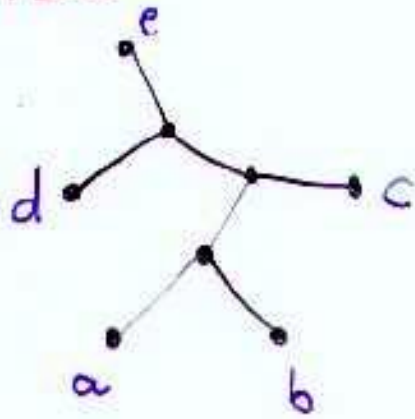
- But A is an MHT of T_1, T_2 .

So $\exists e_1 \in T_1, e_2 \in T_2$ that create a bipartition of the labels that preserve A 's bipartition.

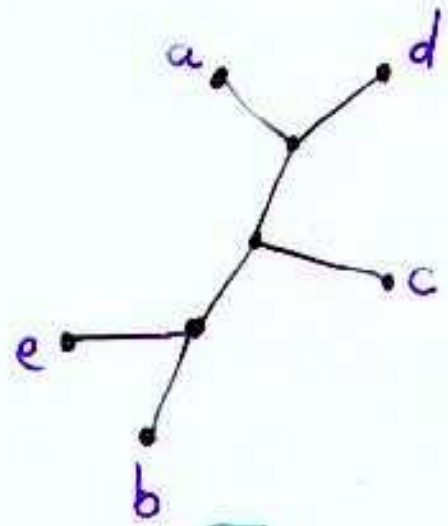
Since our algorithm tries *all* pairs of edges, it will try (e_1, e_2) among the others.

Since it chooses maximum, it will choose (e_1, e_2) or an equivalent pair and produce an MHT. ■

EXAMPLE:



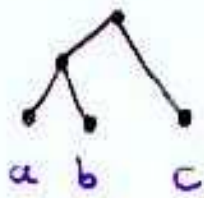
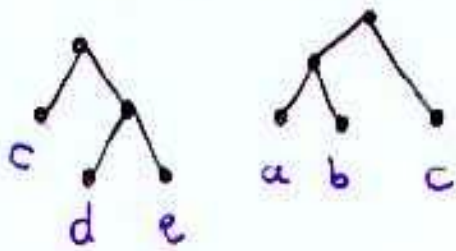
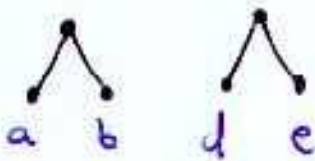
T_1



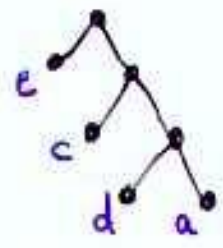
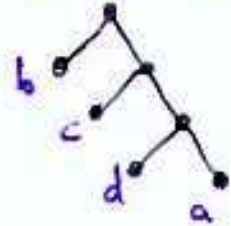
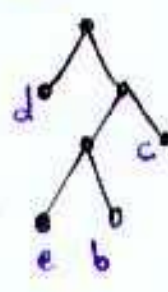
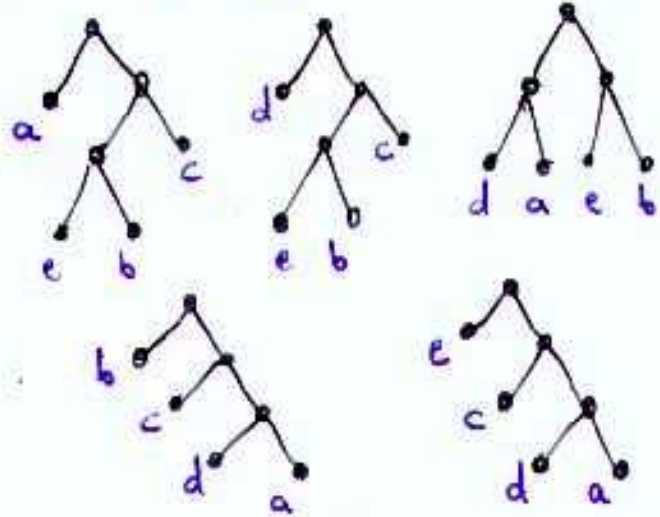
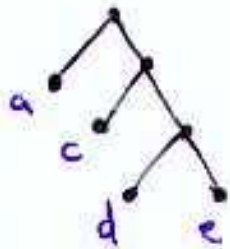
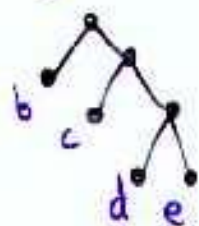
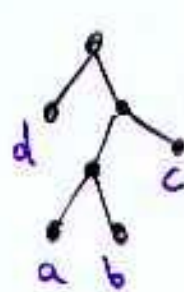
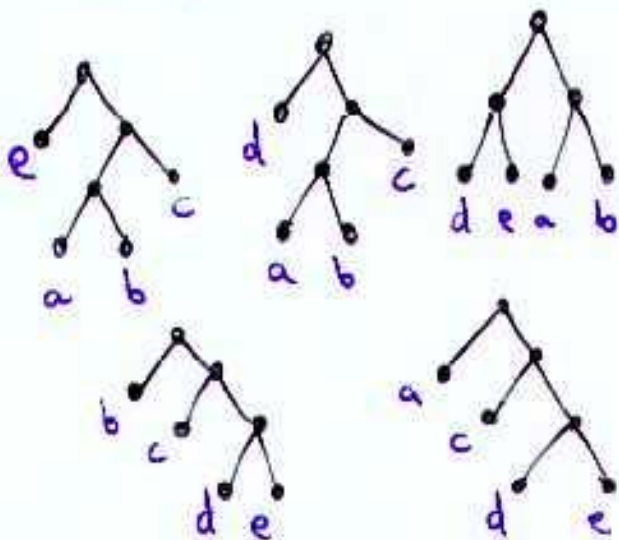
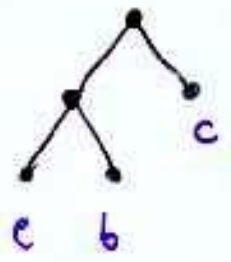
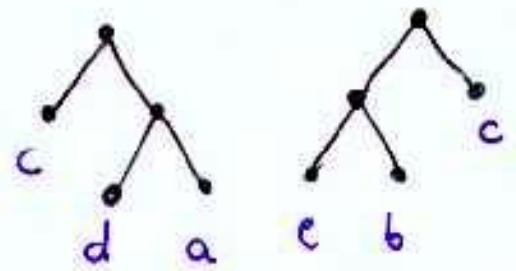
T_2

Subtrees:

a b c d e



a b c d e



$$\text{MHT} \left(\begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ d \quad \text{Node} \\ \swarrow \quad \searrow \\ a \quad b \quad c \end{array} \right), \begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ \text{Node} \quad \text{Node} \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ d \quad a \quad e \quad b \end{array} \right) =$$

$$\max \left(\text{MHT} \left(\begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ \text{Node} \quad c \\ \swarrow \quad \searrow \\ a \quad b \end{array} \right), \begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ d \quad a \end{array} \right) \cup \text{MHT} \left(\begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ d \quad \text{Node} \\ \swarrow \quad \searrow \\ e \quad b \end{array} \right),$$

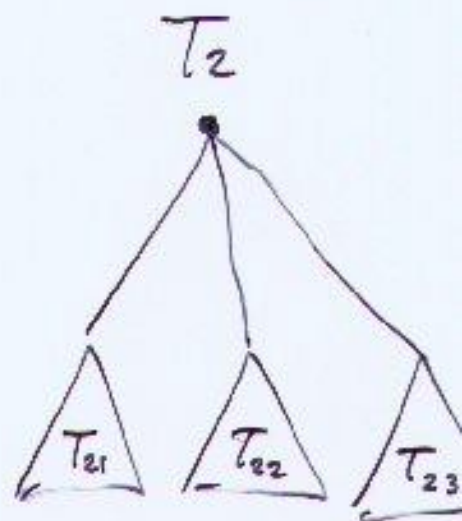
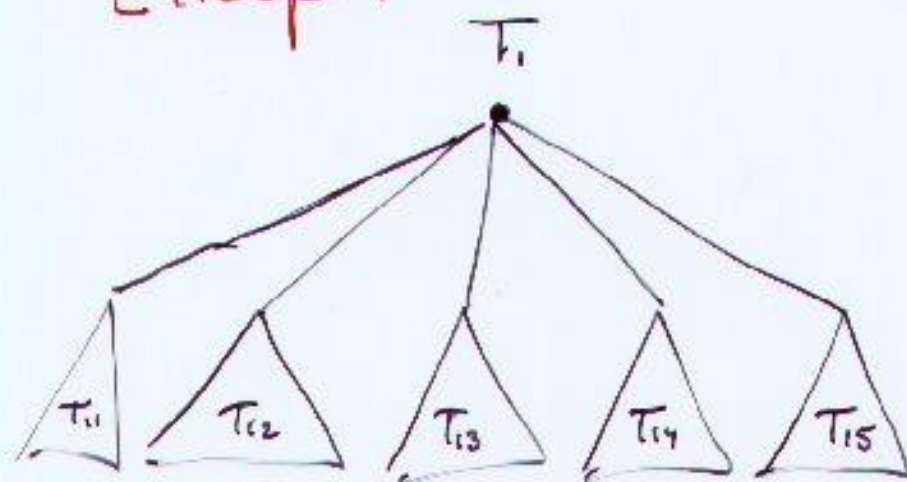
$$\text{MHT} \left(\begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ \text{Node} \quad c \\ \swarrow \quad \searrow \\ a \quad b \end{array} \right), \begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ \text{Node} \quad b \\ \swarrow \quad \searrow \\ e \quad \text{Node} \\ \swarrow \quad \searrow \\ d \quad a \end{array} \right) \cup \text{MHT} \left(\begin{array}{c} \text{Root} \\ \swarrow \quad \searrow \\ d \quad \text{Node} \\ \swarrow \quad \searrow \\ e \quad b \end{array} \right)$$

$$\max \left(\{a\} \cup \emptyset, \{b\} \cup \{d\} \right) = \{b\} \cup \{d\} = \{b, d\}$$

What happens when degree of tree > 3 ?

- All stages of dynamic programming remain same.

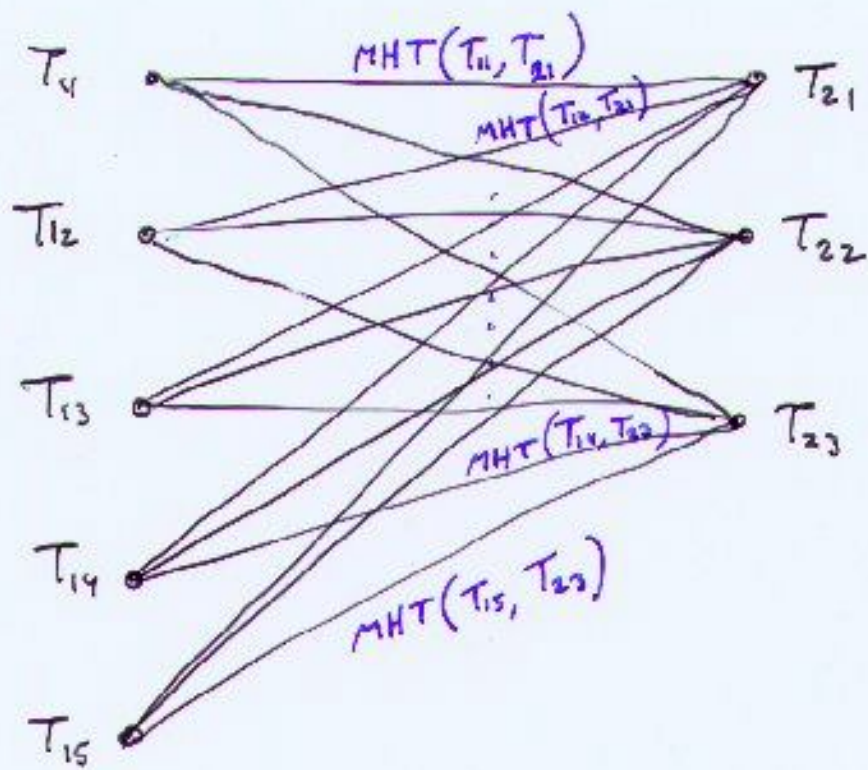
Except:



We have: $MHT(T_{1i}, T_{2j})$ $i=1, \dots, 5$
 $j=1, \dots, 3$

We want: The maximum combination.

How do we do it in polynomial time?
- Weighted Matching.



So Total Time:

Table size $O(m^2)$.

Weighted matching at every

cell: $O(m^{3/2})$

Total: $O(m^{4.5})$

NP-Completeness of 3-MHT and 3-MWT (unbounded degree).

Decision Problem: Seek agreement subtree of given size rather than **maximum**.

3-Dimensional Matching

INPUT: $M \subseteq W \times X \times Y$

where W, X, Y are disjoint
 q element sets.

DECIDE: If $\exists M' \subseteq M$ of size q where
no 2 elements of M' agree on
any coordinate.

EXAMPLE:

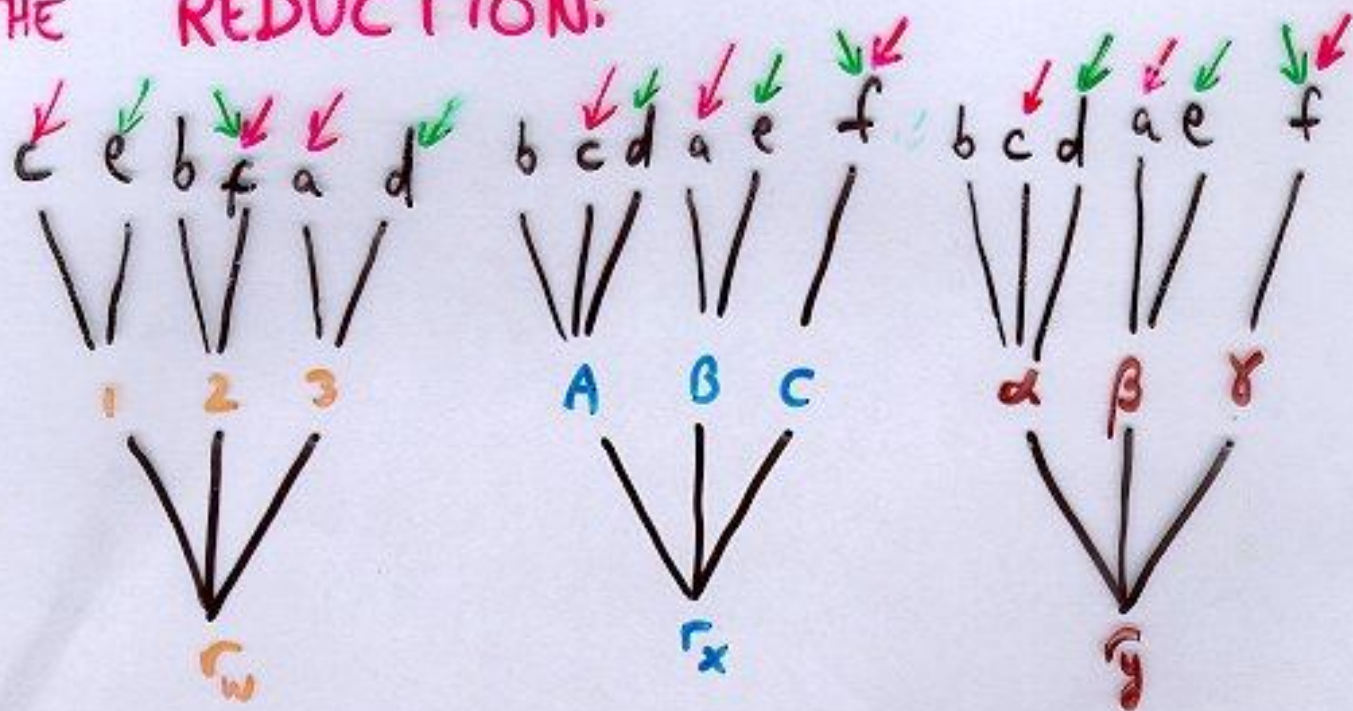
$$W = \{1, 2, 3\} \quad X = \{A, B, C\}$$

$$Y = \{\alpha, \beta, \gamma\}$$

$$M = \{a: \langle 3 B \beta \rangle, b: \langle 2 A \alpha \rangle, c: \langle 1 A \alpha \rangle, \\ d: \langle 3 A \alpha \rangle, e: \langle 1 B \beta \rangle, f: \langle 2 C \gamma \rangle\}$$

$$M' = \{a: \langle 3 B \beta \rangle, c: \langle 1 A \alpha \rangle, f: \langle 2 C \gamma \rangle\}$$

THE REDUCTION:



$\langle w, x, y \rangle$ is child of w in Γ_w , x in Γ_x and y in Γ_y .

(For MIT - "anchor" roots)

APPROXIMATION

Approximate set of leaves **NOT** in MHT.
(approximate the **complement**)

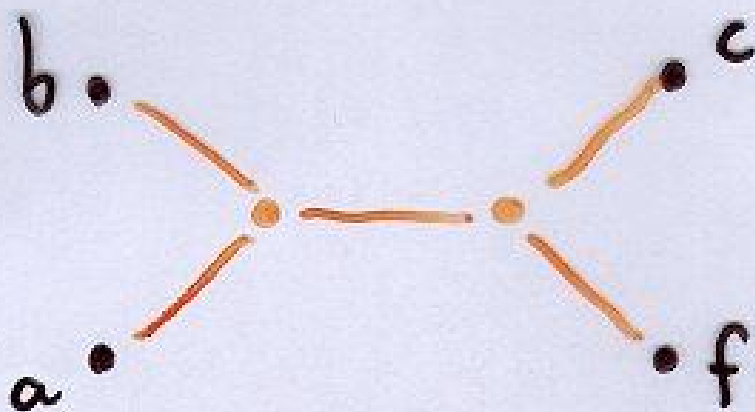
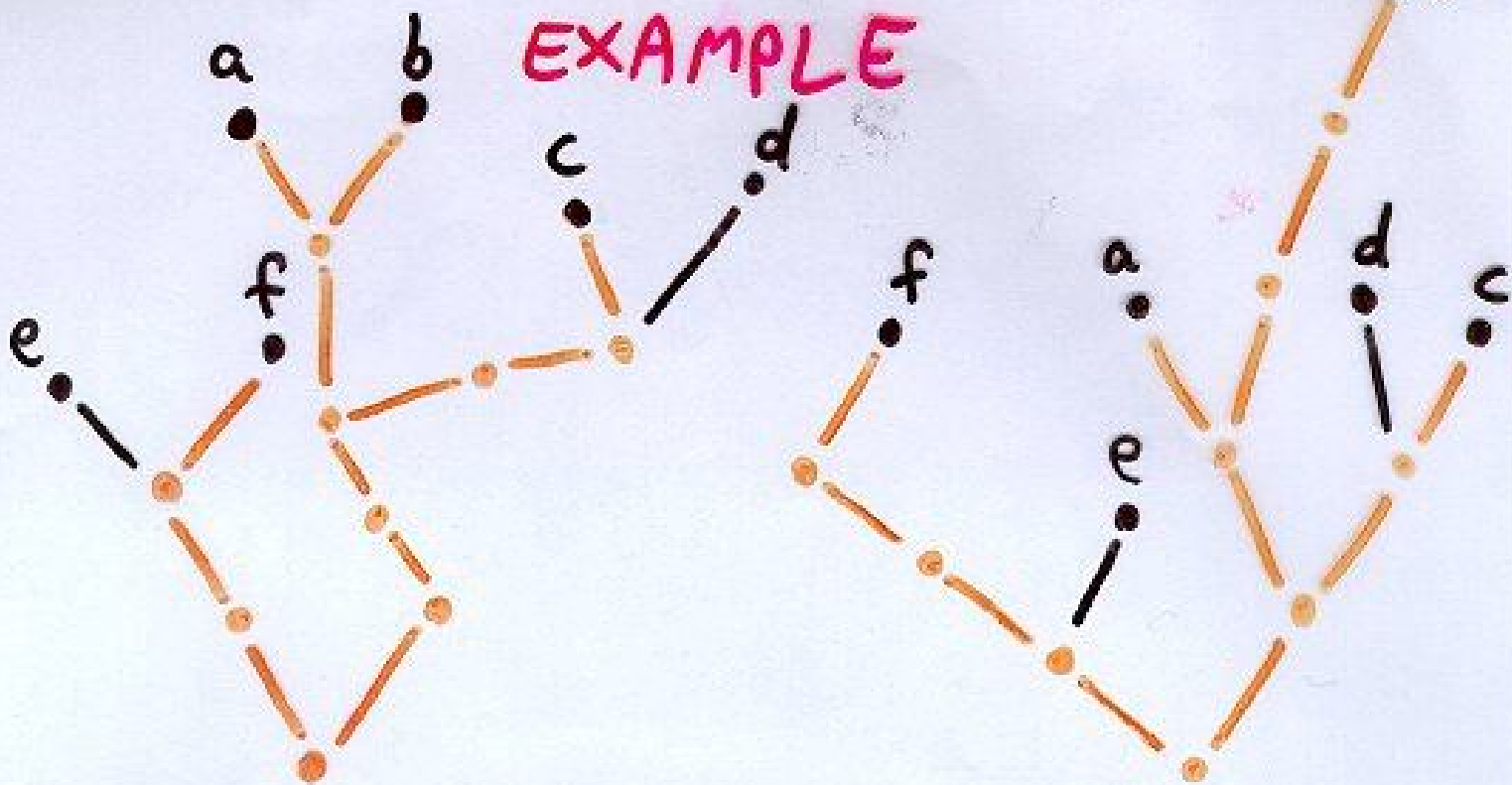
Theorem: (Bandelt & Dress 86)

Two leaf labeled trees are
homeomorphic

iff

all generated 4-leaf subtrees
are homeomorphic

EXAMPLE



IDEA: Generalize Vertex Cover

S_4 : All 4-element subsets of S that **do not** generate a homeomorphic subtree in all trees.

$S' \subseteq S$ is a **cover** of S_4 if

$$\forall \{a, b, c, d\} \in S_4$$

$$\exists e \in S'$$

$$e \in \{a, b, c, d\}$$

S' is **minimum cover** if its size is smallest

From Thm: S' min cover \Rightarrow
homeomorphic restriction of $S-S'$
is MHT.

APPROXIMATE MIN COVER

$S_a \leftarrow \emptyset$

While $S_4 \neq \emptyset$ do

 choose $\alpha \in S_4$

$S_a \leftarrow S_a \cup \alpha$

 Delete from S_4 all β
 such that $\alpha \cap \beta \neq \emptyset$

end

Time: $O(|S_4|) = O(n^4)$

Constructing S_4 : $O(kn^5)$

Size of S_a : At most 4 times
minimum cover.