The worst case complexity of Maximum Parsimony

- Amir Carmel
- Noa Musa-Lempel
- Dekel Tsur
- Michal Ziv-Ukelson

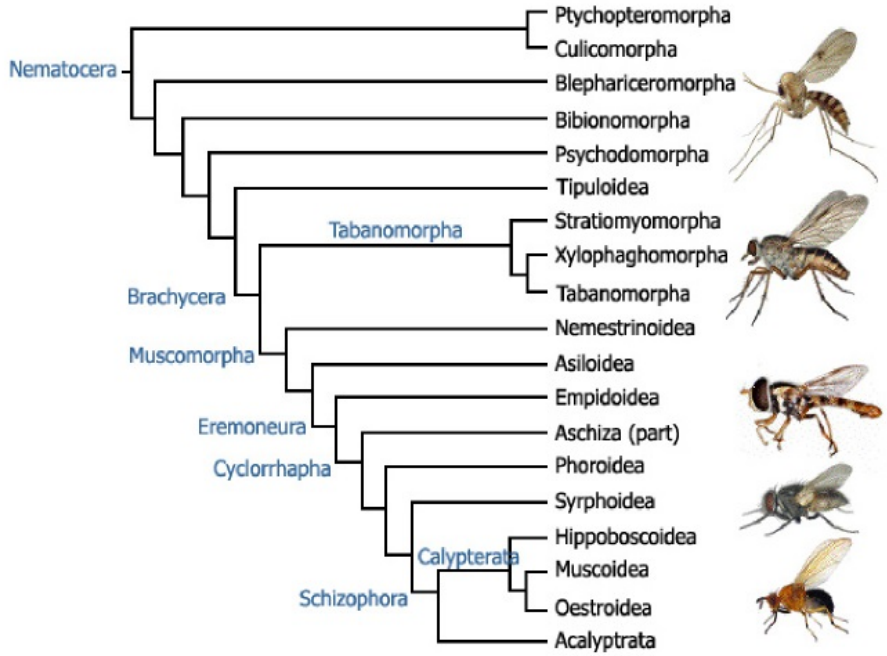
Ben-Gurion University

June 12, 2014

What's a phylogeny

Phylogenies:

- Graph-like structures whose topology describes the inferred evolutionary history among a set of species.
- Modeled as either rooted or unrooted labeled binary trees, where the input entities are assigned to the leaf vertices.

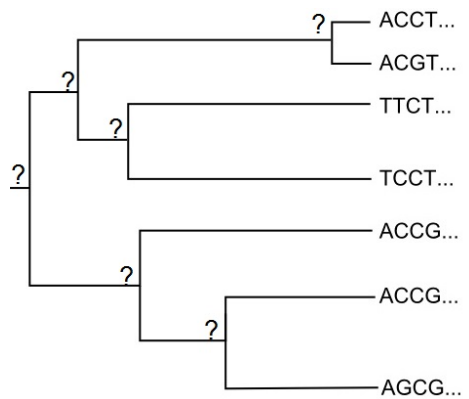


Character based methods for phylogenetic reconstruction

- Each specie is characterized by a sequence of letters.
- We are given a subsitution scoring matrix over the letters.
- Position independence is assumed.

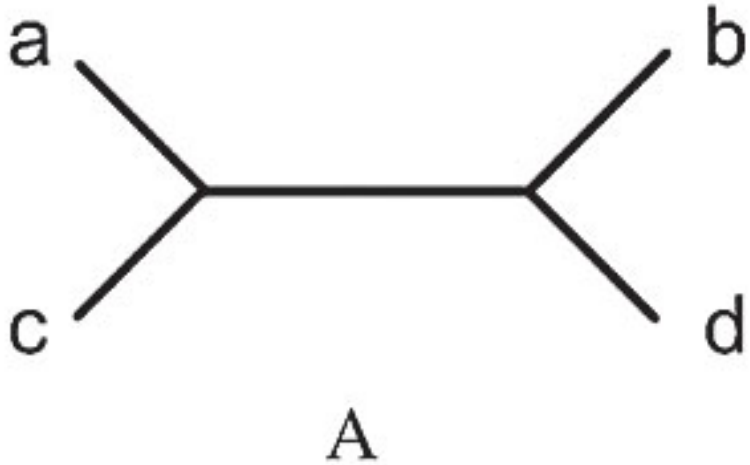
1:	A	C	A	G	G	T	T
2:	C	C	A	G	A	T	T
3:	C	C	G	G	G	T	A
4:	T	G	A	G	G	T	A
5:	T	G	A	G	G	T	T

	A	T	G	C
A	3	0	1	0
T	0	3	0	1
G	1	0	3	0
C	0	1	0	3

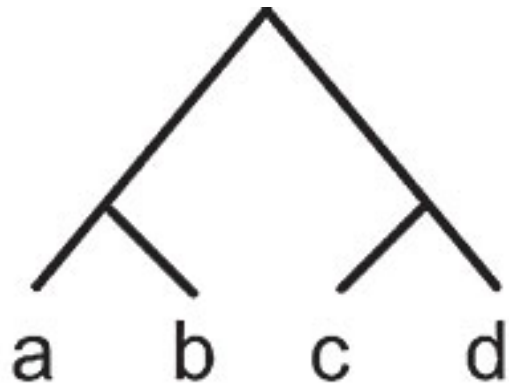


rooted/unrooted phylogeny

- The decision whether to model phylogenies as rooted versus unooted depends on the substitution scoring matrix.
- Modeling phylogenies as unrooted trees requires the assumption of symmetric scoring matrices.
- Today, many applications apply asymmetric scoring matrices.



A



B

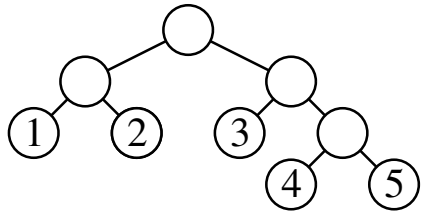
Parsimony Maximization

- A classical approach for phylogenetic reconstruction.
- The Parsimony Maximization approach seeks the phylogenetic tree that supposes the least amount of evolutionary change explaining the observed data.
- There are two classical problems inferred from phylogenetic parsimony maximization: Small Parsimony (SP) and Maxmimum Parsimony (MP).

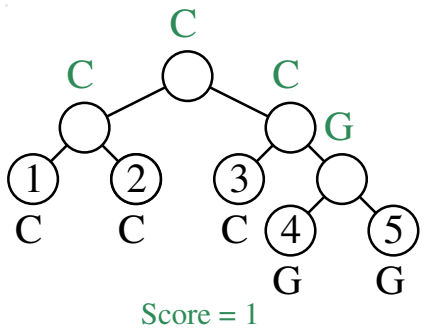
Small Parsimony Problem (SP)

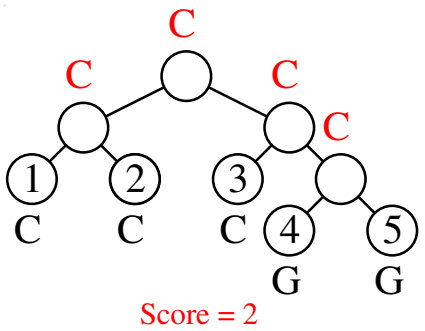
Input: multiple alignment, tree topology on n leaves.

1:	A	C	A	G	G	T	T
2:	C	C	A	G	A	T	T
3:	C	C	G	G	G	T	A
4:	T	G	A	G	G	T	A
5:	T	G	A	G	G	T	T



Goal: Assignment to internal vertices that minimizes the scoring function.

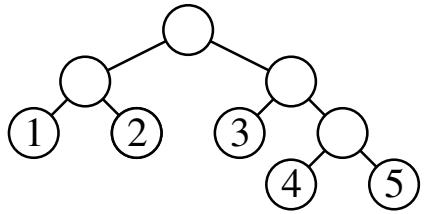




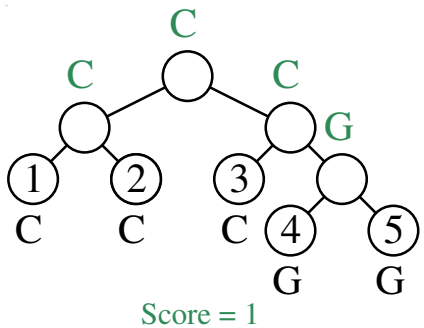
Small Parsimony Problem (SP)

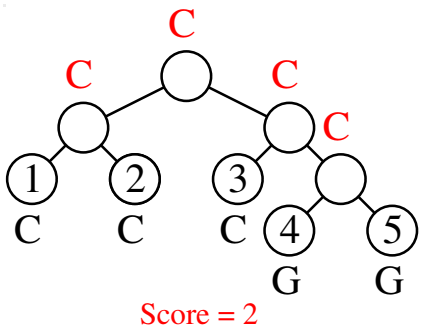
Input: multiple alignment, tree topology on n leaves.

1:	A	C	A	G	G	T	T
2:	C	C	A	G	A	T	T
3:	C	C	G	G	G	T	A
4:	T	G	A	G	G	T	A
5:	T	G	A	G	G	T	T



Goal: Assignment to internal vertices that minimizes the scoring function.





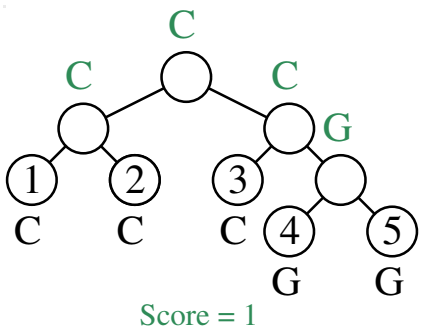
We note that known algorithms for Small Parsimony traverse the tree in a bottom up manner.

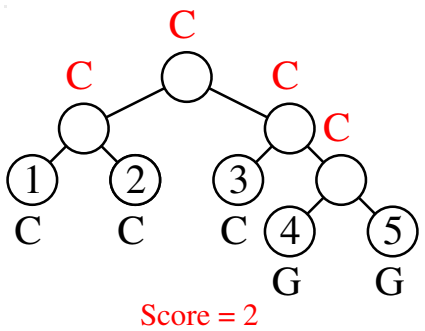
Maximum Parsimony Problem (MP)

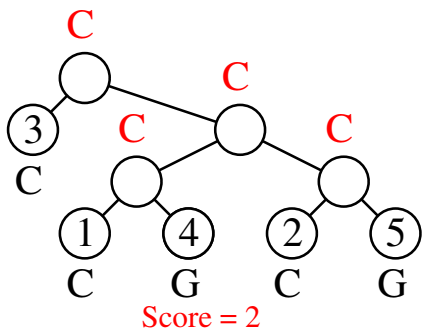
Input: multiple alignment

1:	A	C	A	G	G	T	T
2:	C	C	A	G	A	T	T
3:	C	C	G	G	G	T	A
4:	T	G	A	G	G	T	A
5:	T	G	A	G	G	T	T

Goal: topology and assignments to internal vertices, that minimizes the SP score.





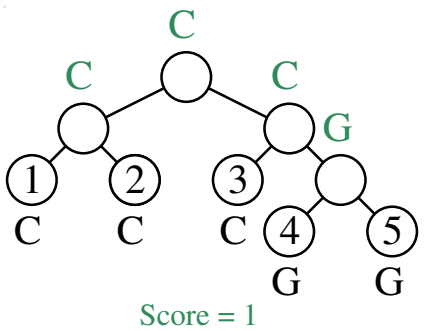


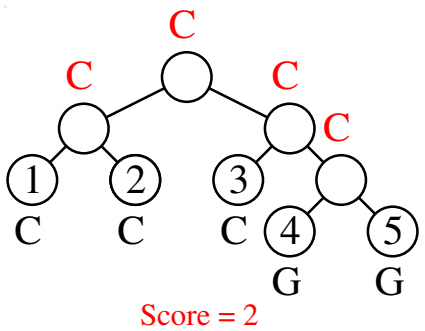
Maximum Parsimony Problem (MP)

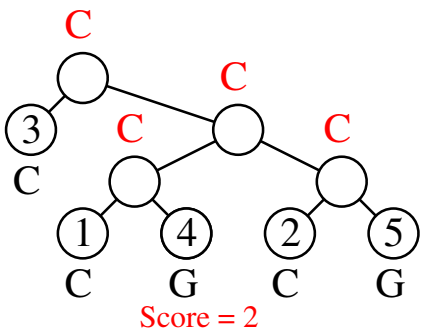
Input: multiple alignment

1:	A	C	A	G	G	T	T
2:	C	C	A	G	A	T	T
3:	C	C	G	G	G	T	A
4:	T	G	A	G	G	T	A
5:	T	G	A	G	G	T	T

Goal: topology and assignments to internal vertices, that minimizes the SP score.







The Maximum Parsimony (MP) problem is NP-hard [L. R. Foulds and R. L. Graham (1982)].

Measuring SP and MP complexity in terms of assignment operations

- *Assignment operation* - time to compute the assignment for a single vertex.
- This depends on the scoring scheme employed, for example: Fitch's algorithm (Hamming distance) $O(m)$, Sankoff's algorithm (weighted edit distance) $O(m\Sigma^2)$.

Our contribution

Previous results:

- Cavalli-Sforza and Edwards (1967) - $(n-1) \cdot (2n-3)!!$ assignment operations.
- Hendy and Penny (1982) - branch&bound algorithm for MP.

Where $(2n-3)!! = 1 \times 3 \times 5 \times \ldots \times (2n-3)$.

Our contribution

New results:

- Cavalli-Sforza and Edwards (1967) - $(n-1) \cdot (2n-3)!!$ assignment operations.
- Hendy and Penny (1982) - branch&bound algorithm for MP. Worst case running time: $\Theta(\sqrt{n} \cdot (2n-3)!!)$ assignment operations.
- A new, faster algorithm which executes $\Theta((2n-3)!!)$ assignment operations.

Where $(2n-3)!! = 1 \times 3 \times 5 \times \ldots \times (2n-3)$

The algorithm of Cavalli-Sforza and Edwards

The algorithm of Cavalli-Sforza and Edwards

- Cavalli-Sforza and Edwards showed that the number of rooted phylogenies with n leaves is $(2n-3)!!$.

The algorithm of Cavalli-Sforza and Edwards

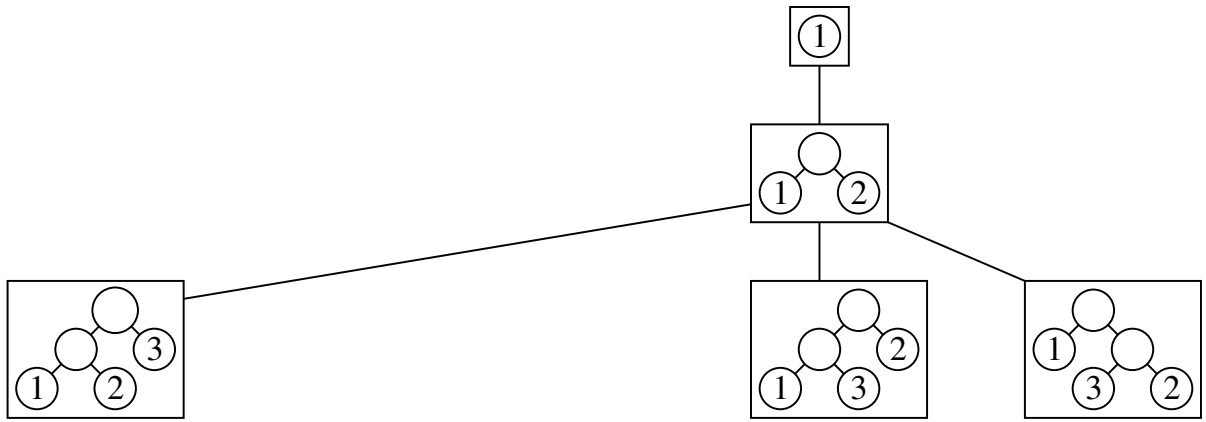
- Cavalli-Sforza and Edwards showed that the number of rooted phylogenies with n leaves is $(2n-3)!!$.
- The algorithm enumerates all phylogenies with n leaves, and then solves the Small Parsimony (SP) problem on each tree.

The algorithm of Cavalli-Sforza and Edwards

- Cavalli-Sforza and Edwards showed that the number of rooted phylogenies with n leaves is $(2n-3)!!$.
- The algorithm enumerates all phylogenies with n leaves, and then solves the Small Parsimony (SP) problem on each tree.
- Each phylogeny has exactly $n-1$ internal vertices, therefore the algorithm has a running time of $(n-1)\cdot(2n-3)!!$ assignment operations.

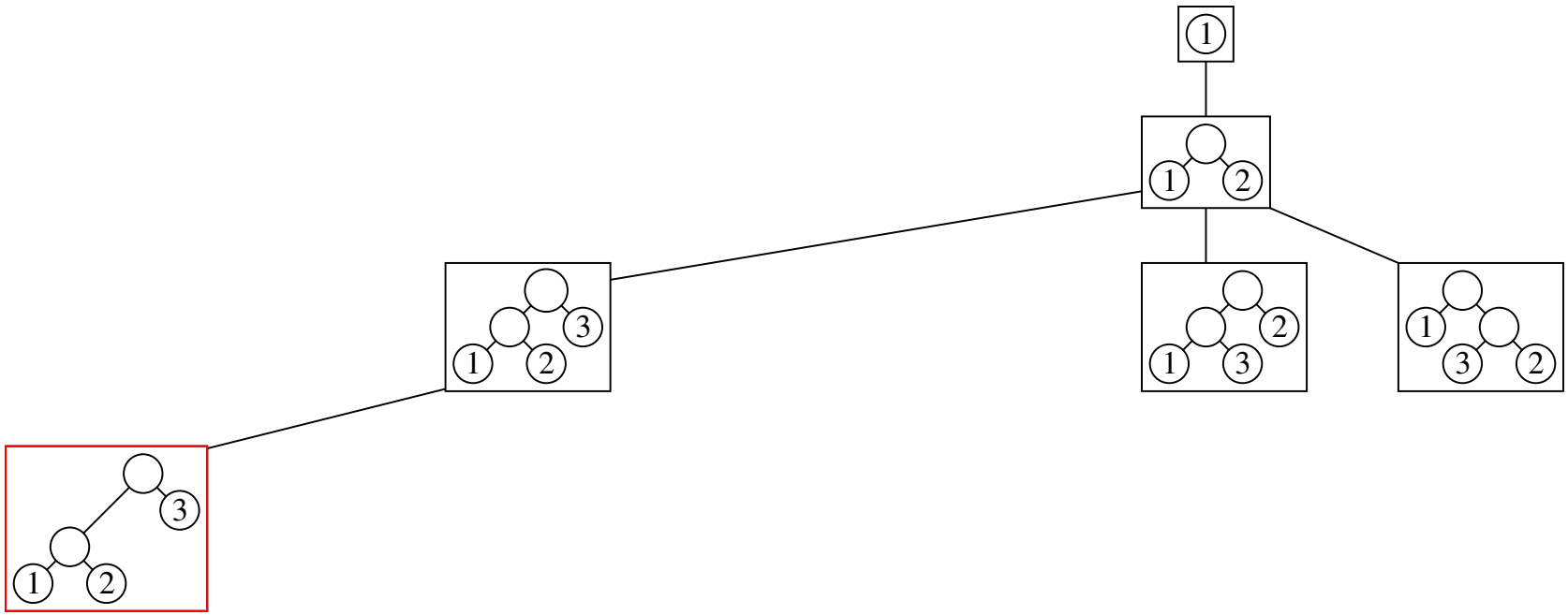
The algorithm of Hendy and Penny

Preliminaries:



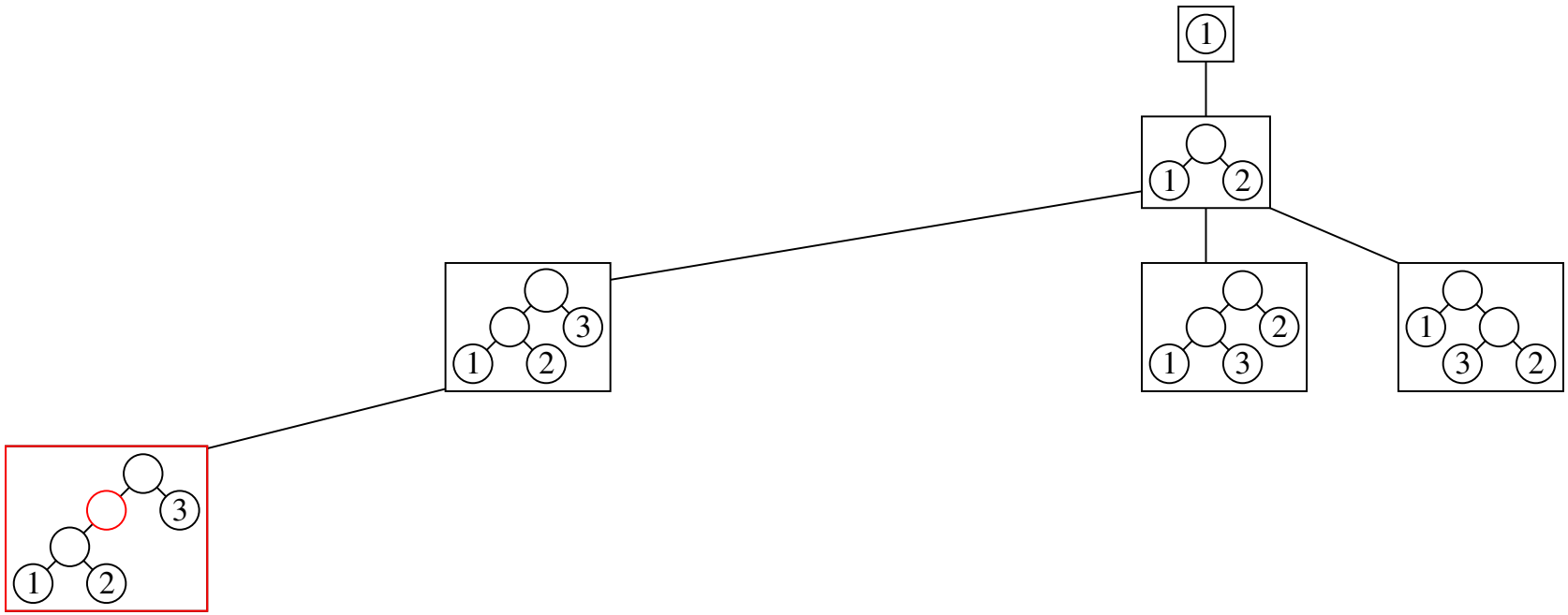
The algorithm of Hendy and Penny

Enumeration space:



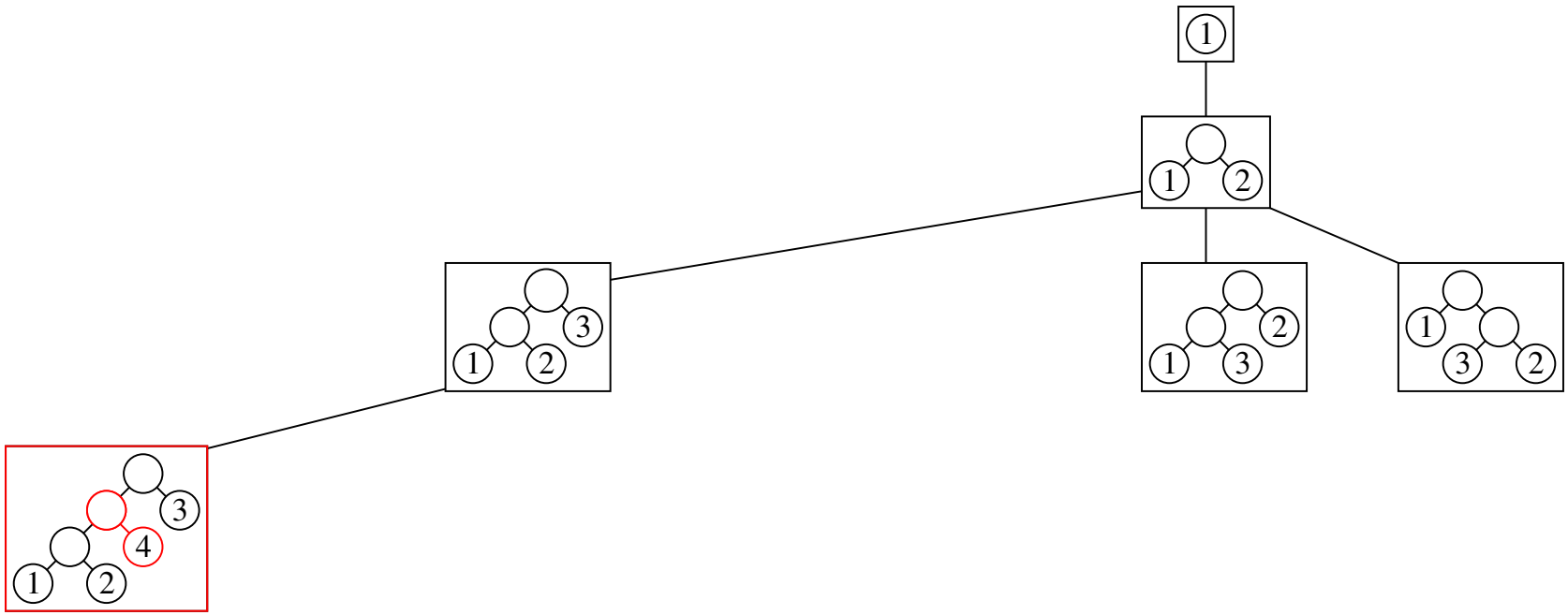
The algorithm of Hendy and Penny

Enumeration space:



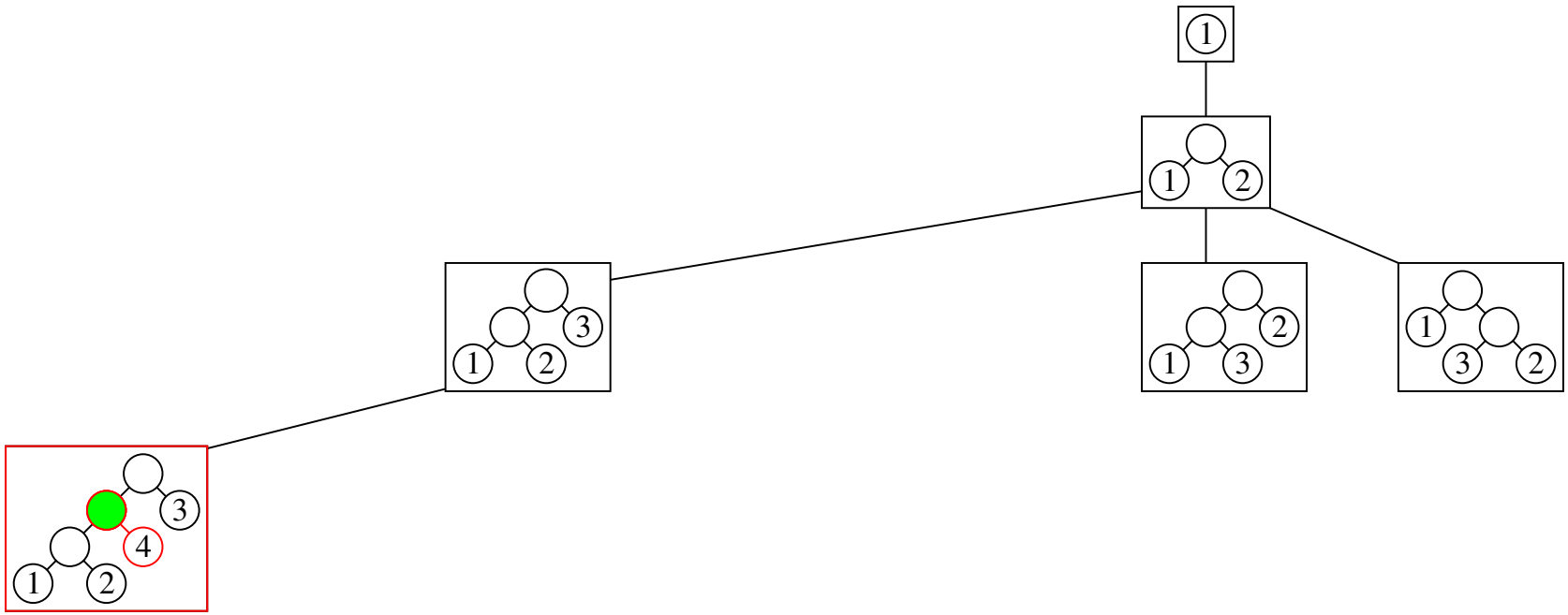
The algorithm of Hendy and Penny

Enumeration space:



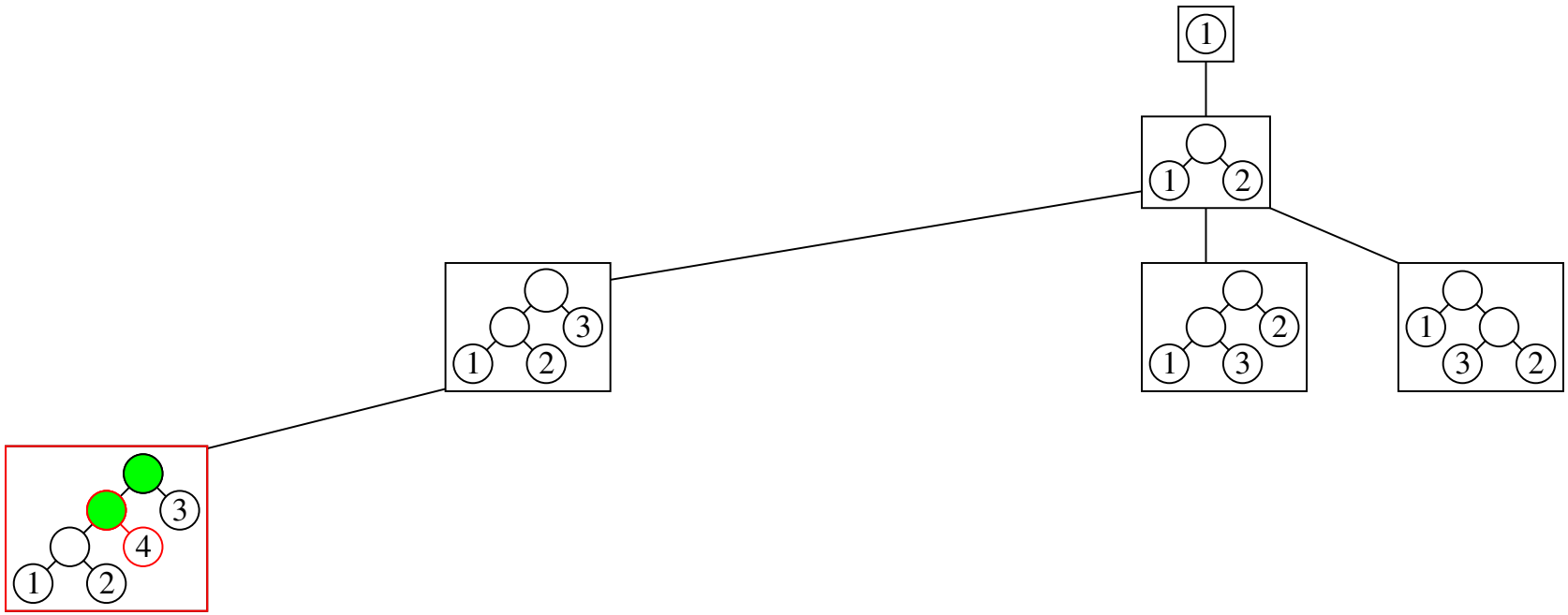
The algorithm of Hendy and Penny

Assignment operations:



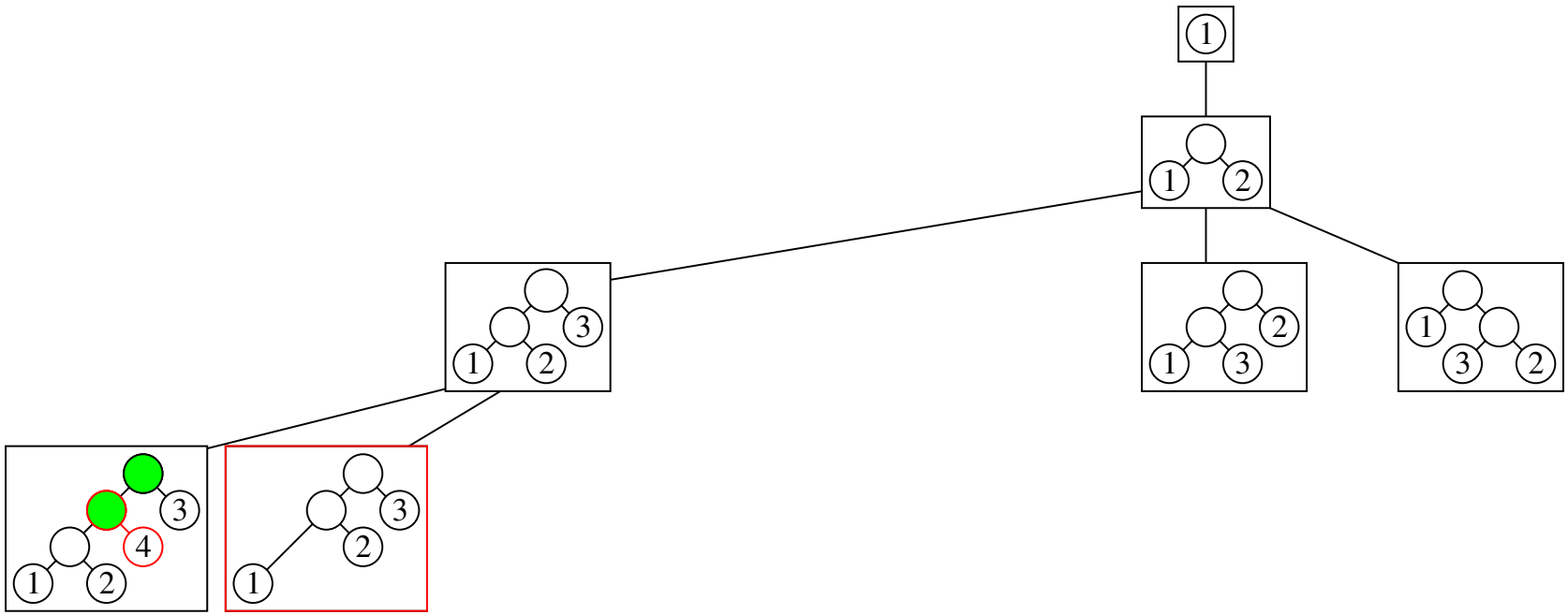
The algorithm of Hendy and Penny

Assignment operations:



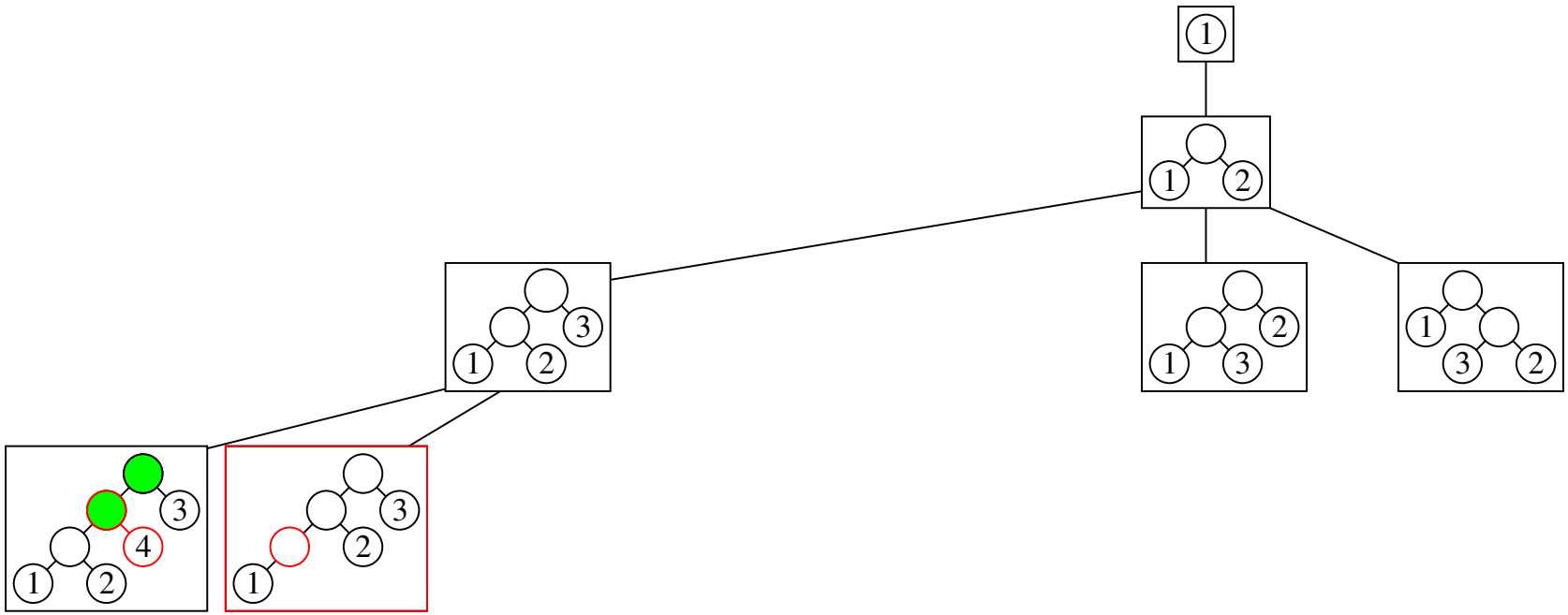
The algorithm of Hendy and Penny

Assignment operations:



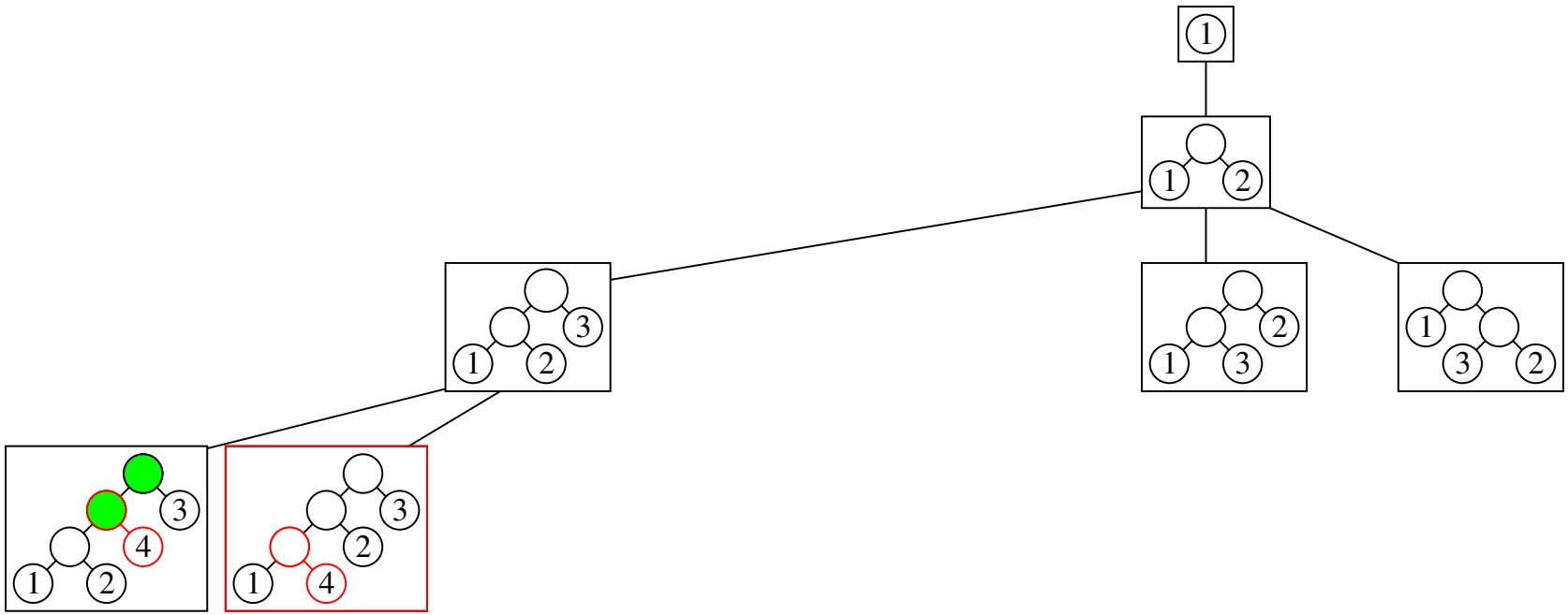
The algorithm of Hendy and Penny

Assignment operations:



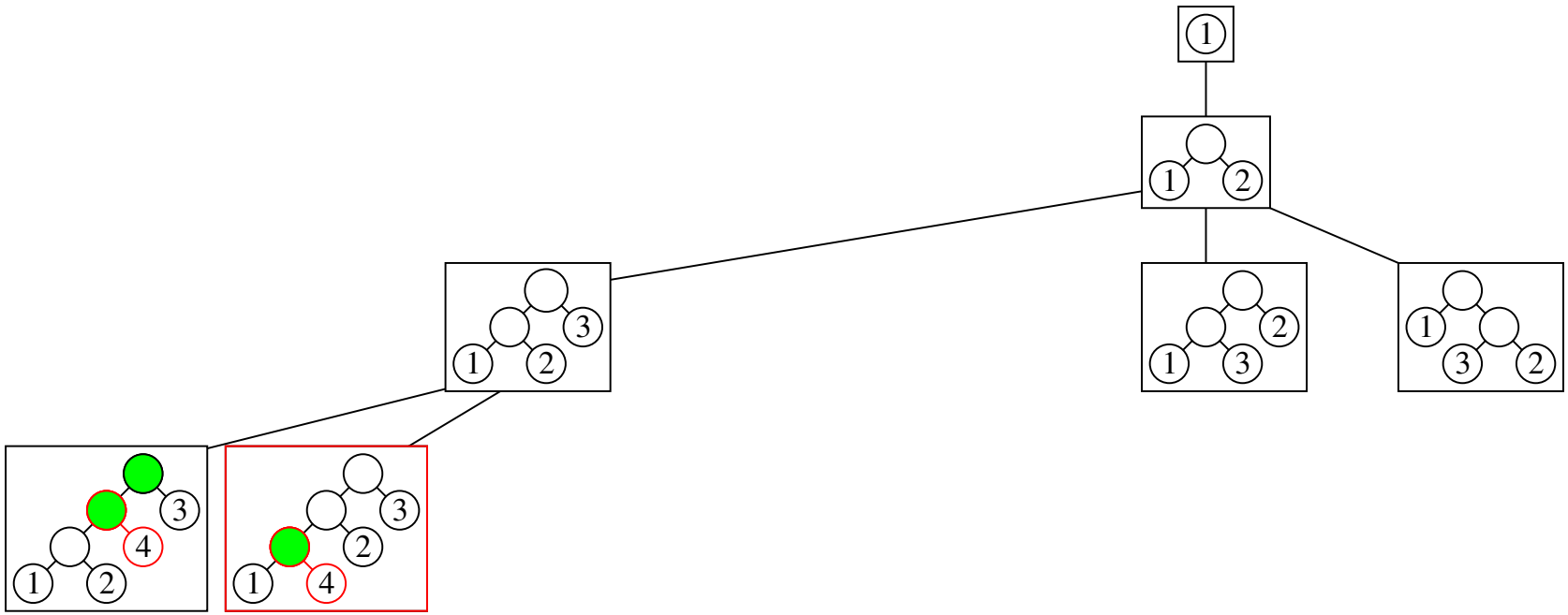
The algorithm of Hendy and Penny

Assignment operations:



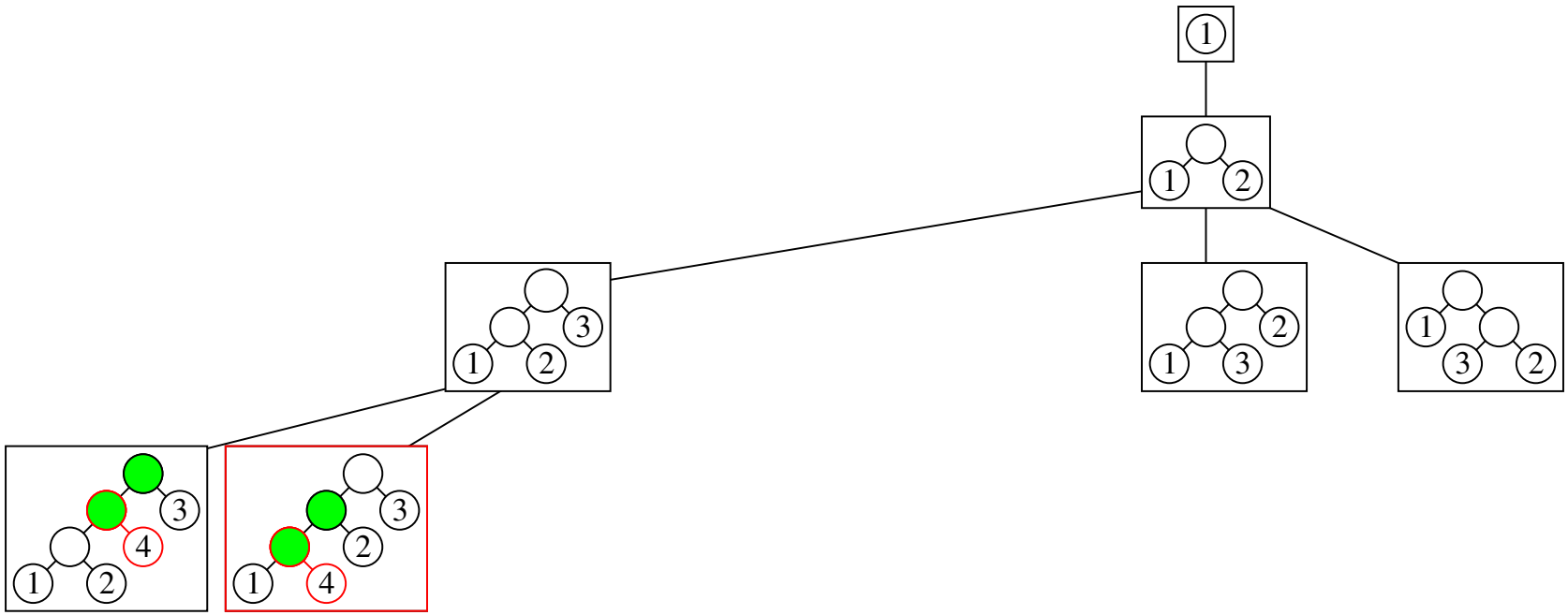
The algorithm of Hendy and Penny

Assignment operations:



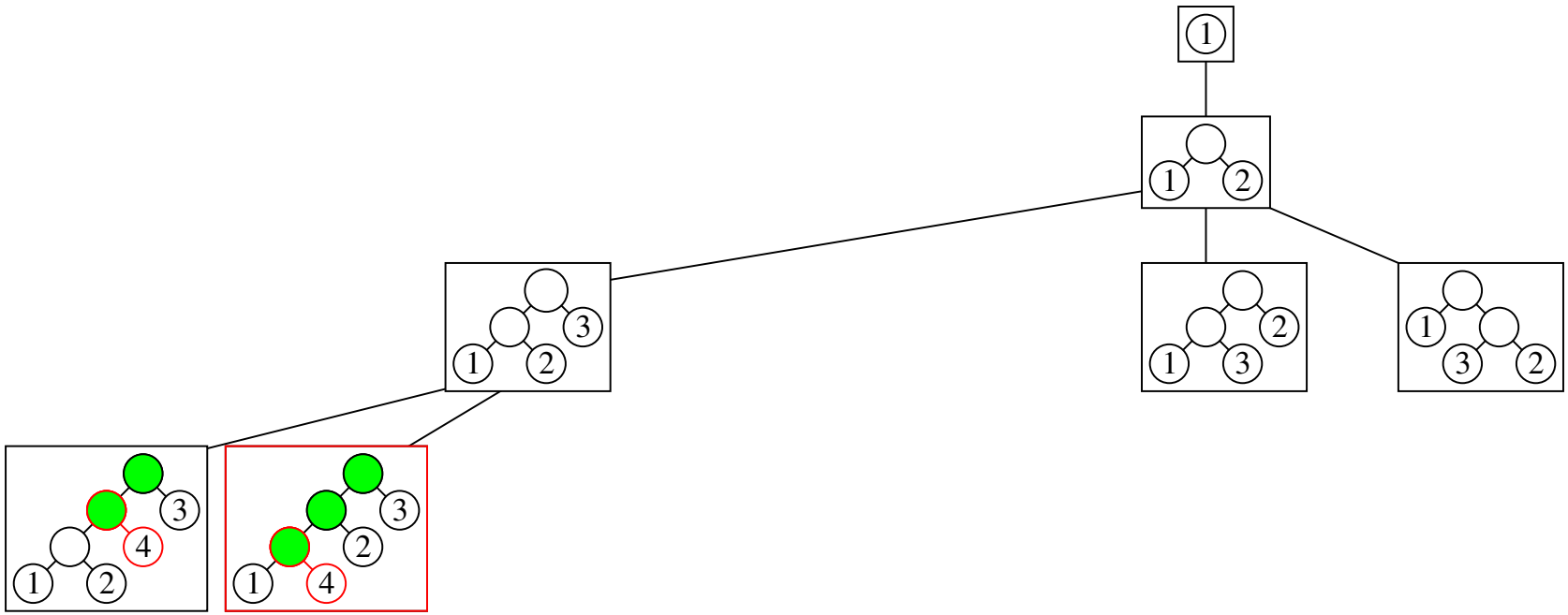
The algorithm of Hendy and Penny

Assignment operations:



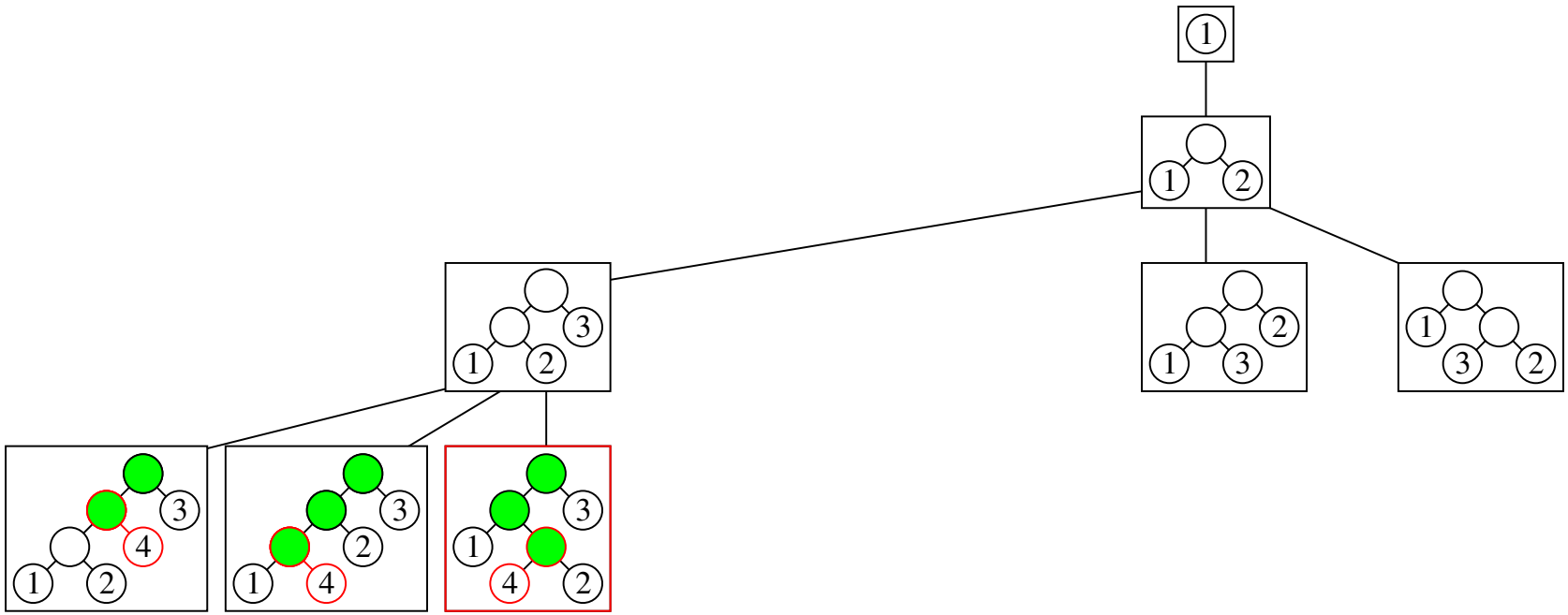
The algorithm of Hendy and Penny

Assignment operations:



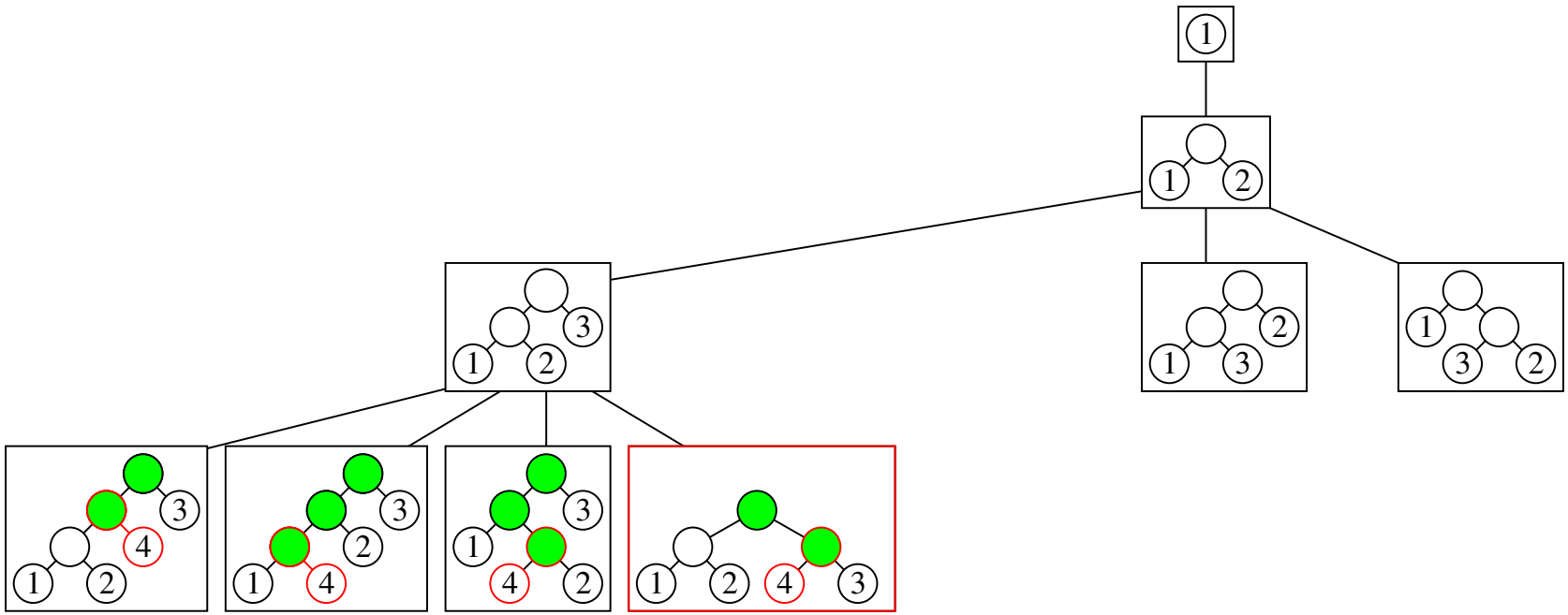
The algorithm of Hendy and Penny

Assignment operations:



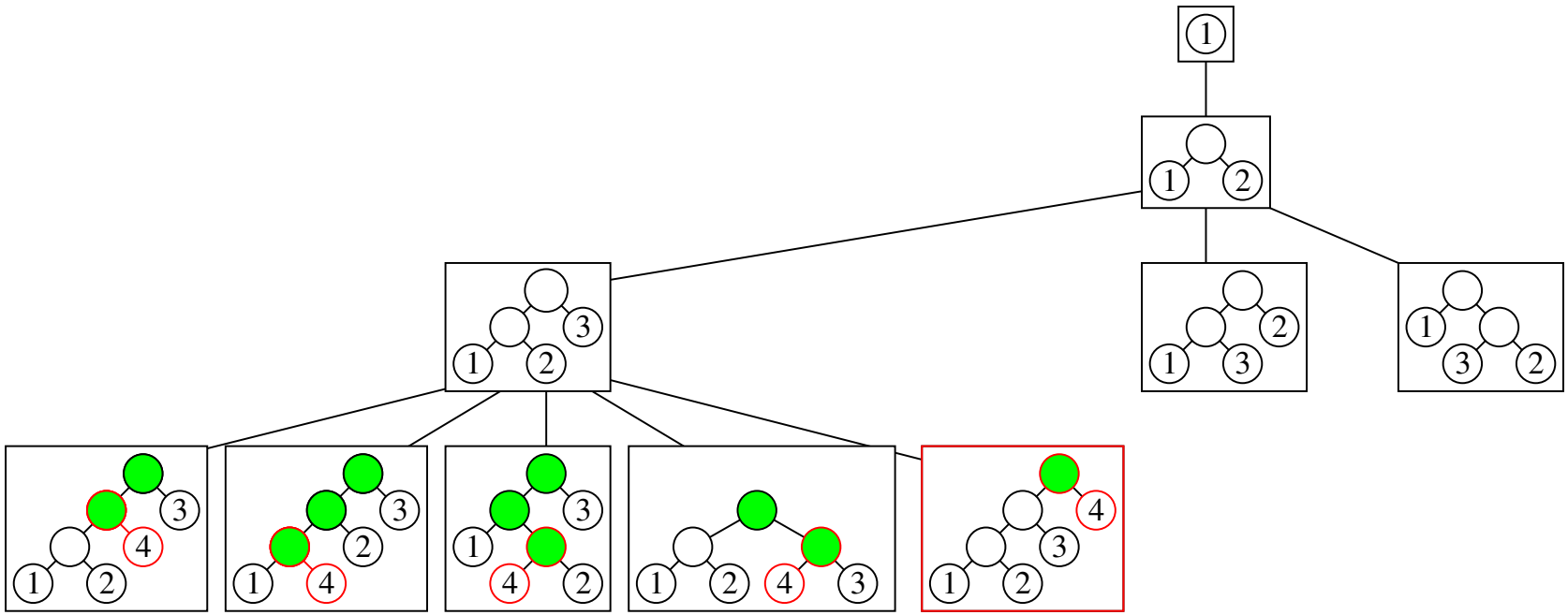
The algorithm of Hendy and Penny

Assignment operations:



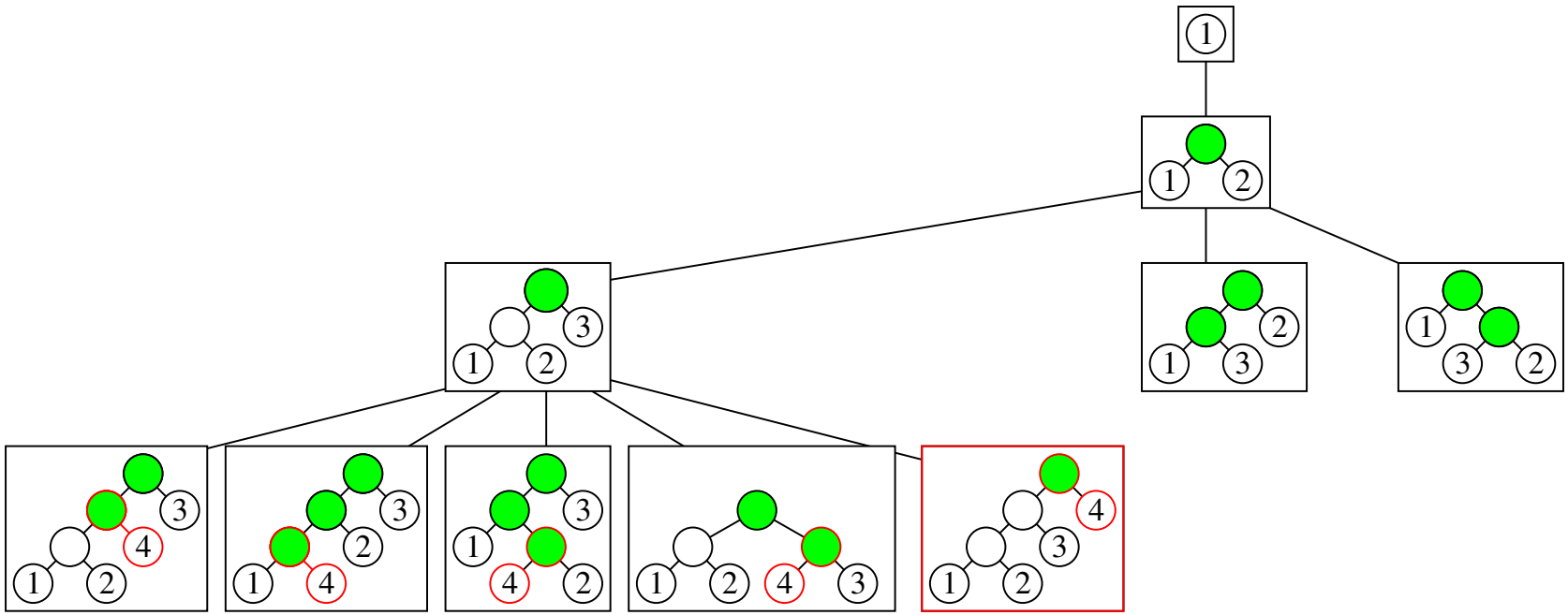
The algorithm of Hendy and Penny

Assignment operations:



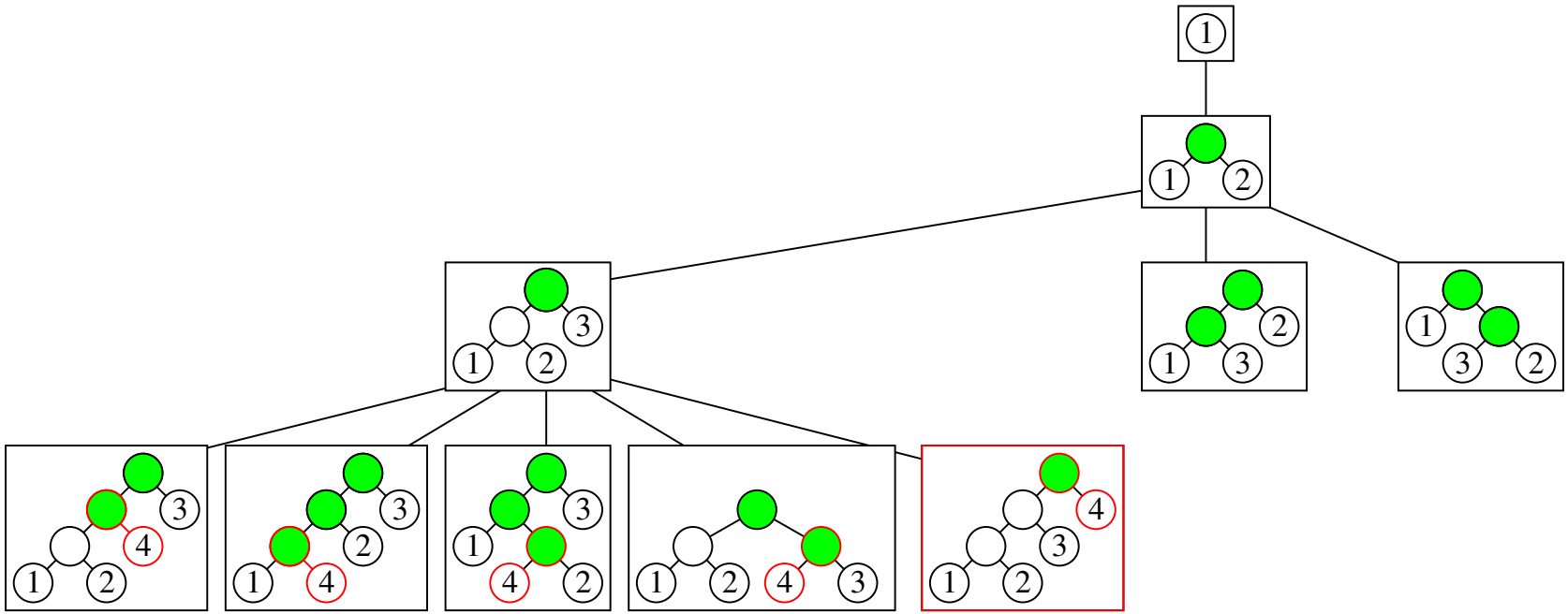
The algorithm of Hendy and Penny

Assignment operations:



The algorithm of Hendy and Penny

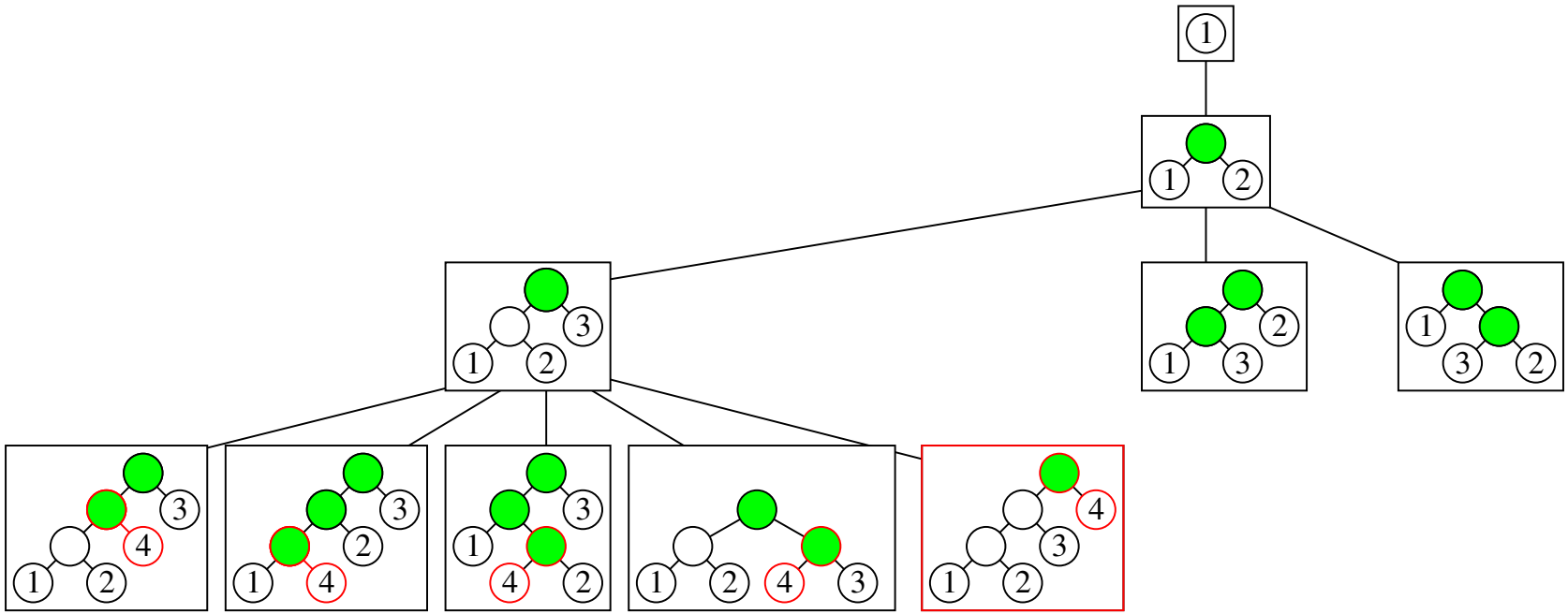
Assignment operations:



The search space tree is developed in top-down order, while the recalculations of assignments is done in a bottom-up order.

The algorithm of Hendy and Penny

Assignment operations:



The complexity of the algorithm equals to the number of assignment operations.

The algorithm of Hendy and Penny

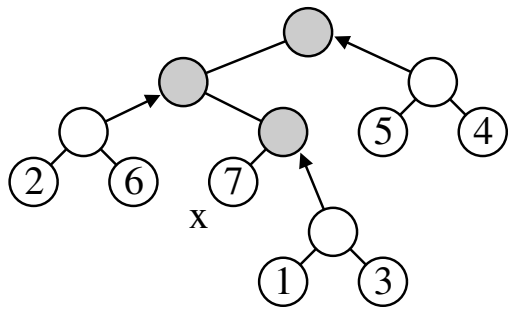
Their algorithm was originally proposed for the purpose of branch and bound and its worst case bound was not previously properly analyzed. Using combinatorial methods we managed to achieve an exact bound.

The number of assignment operations

- Let $\textsc{NumAnc}(v)$ denote the number of ancestors of x in F_v.

The number of assignment operations

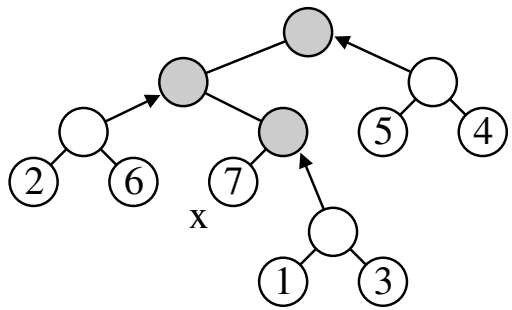
- Let $\textsc{NumAnc}(v)$ denote the number of ancestors of x in F_v.



-

The number of assignment operations

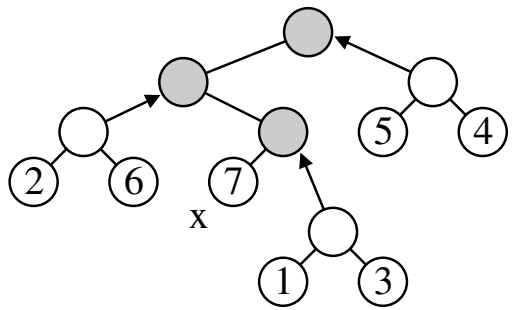
- Let $\textsc{NumAnc}(v)$ denote the number of ancestors of x in F_v.



-
- The number of ancestors of x in F_v is equal to the number of assignment operations executes in node v.

The number of assignment operations

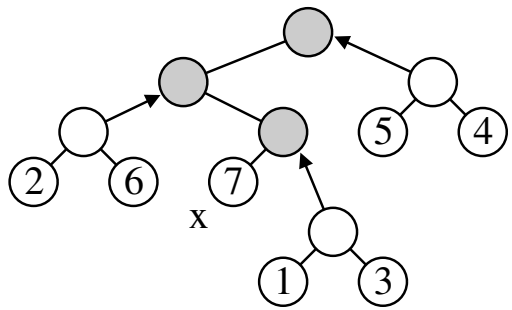
- Let $\textsc{NumAnc}(v)$ denote the number of ancestors of x in F_v.



-
- The number of ancestors of x in F_v is equal to the number of assignment operations executes in node v.
- Let H_i be the sum of $\textsc{NumAnc}(v)$ for all nodes v in level $i+1$.

The number of assignment operations

- Let $\textsc{NumAnc}(v)$ denote the number of ancestors of x in F_v.



-
- The number of ancestors of x in F_v is equal to the number of assignment operations executes in node v.
- Let H_i be the sum of $\textsc{NumAnc}(v)$ for all nodes v in level $i+1$.
- By definition, $\sum \textsc{NumAnc}(v) = \sum_{i=1}^{n-1} H_i$.

The number of assignment operations

Lemma 1

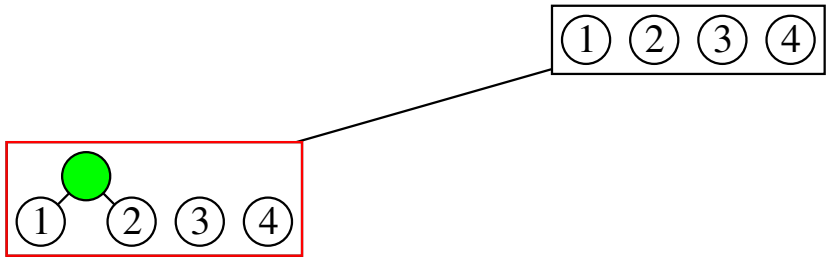
$H_i = (2i)!! - (2i-1)!!$.

Theorem 2

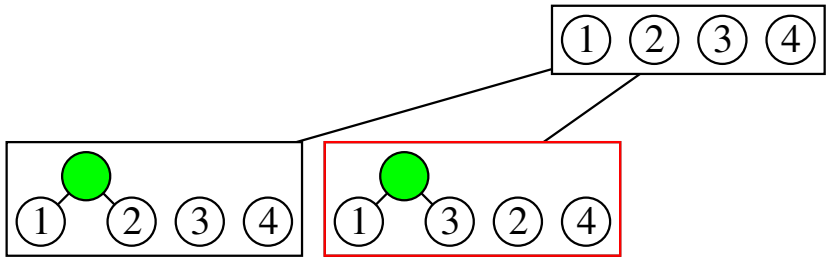
The assignment operations complexity of the algorithm of Hendy and Penny is $\Theta(\sqrt{n}(2n-3)!!)$.

A new, more efficient search space tree

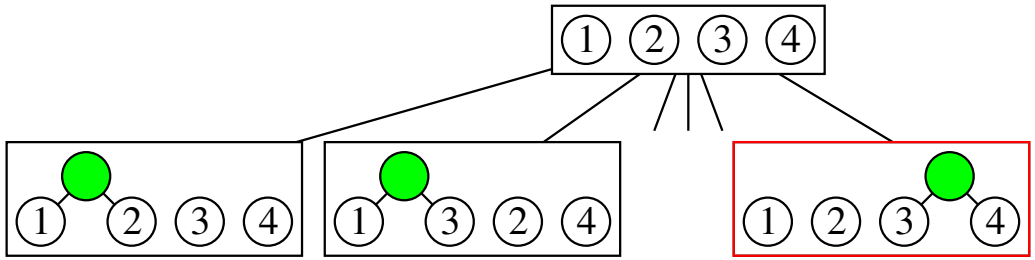
A new, more efficient search space tree



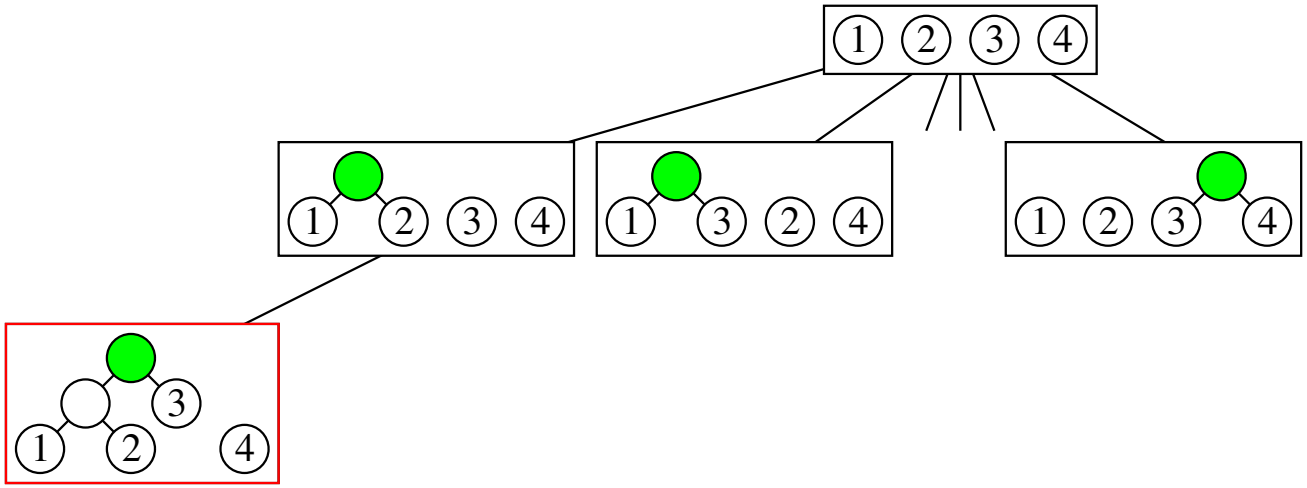
A new, more efficient search space tree



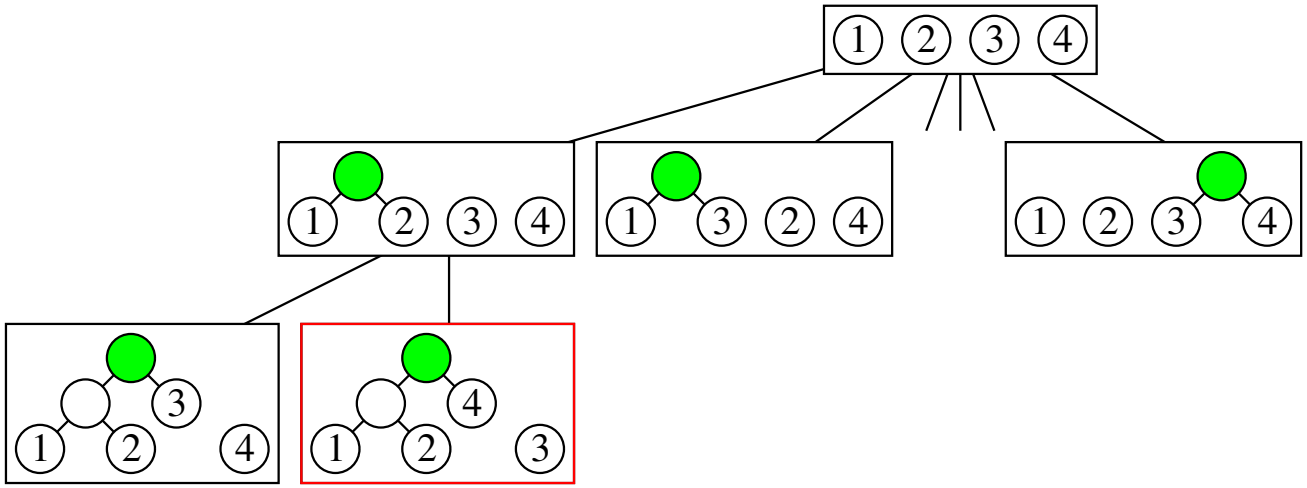
A new, more efficient search space tree



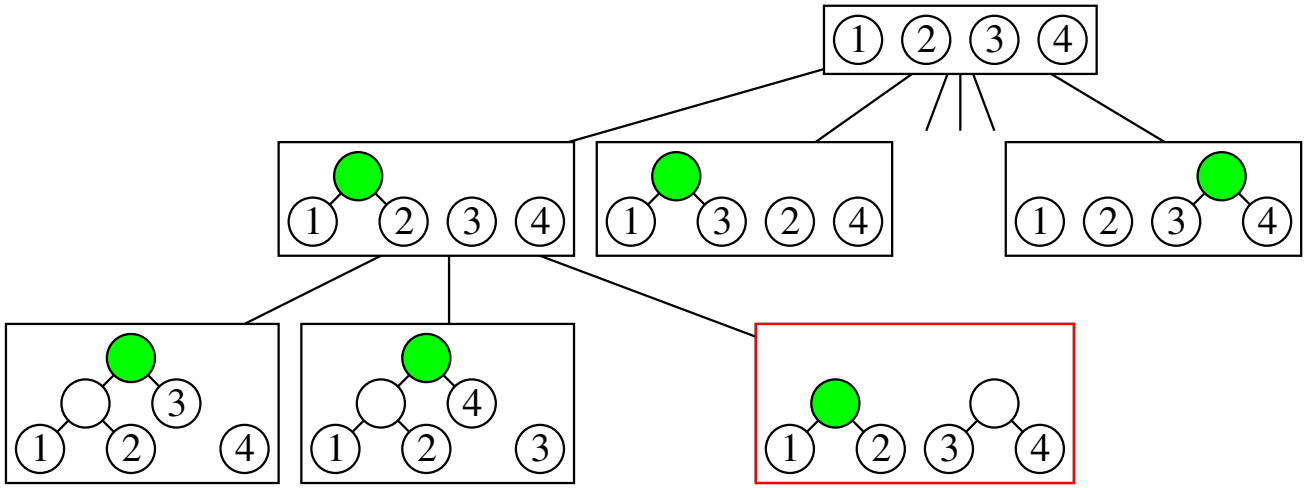
A new, more efficient search space tree



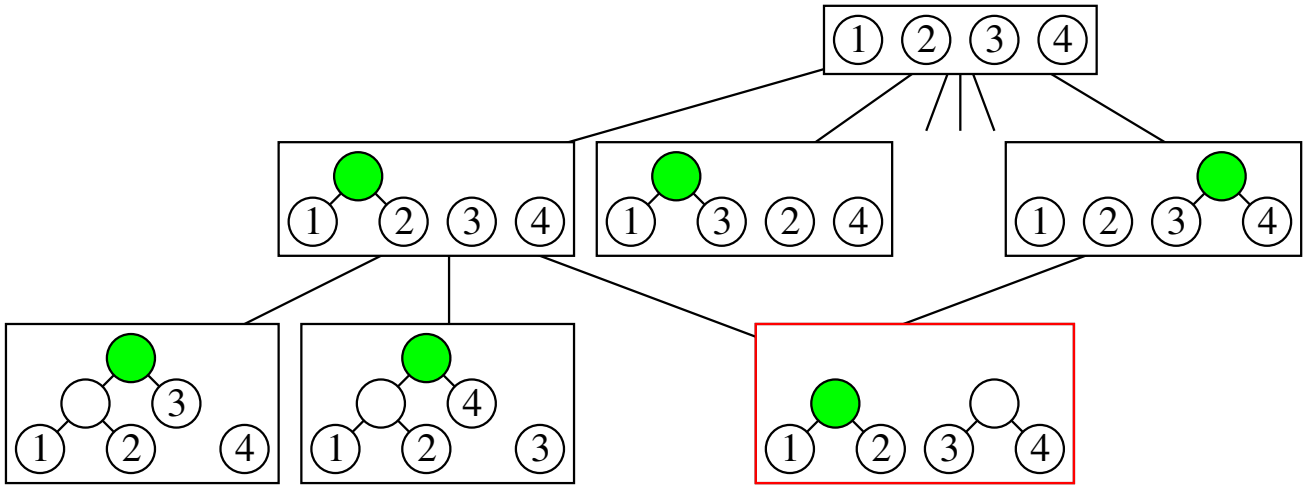
A new, more efficient search space tree



A new, more efficient search space tree

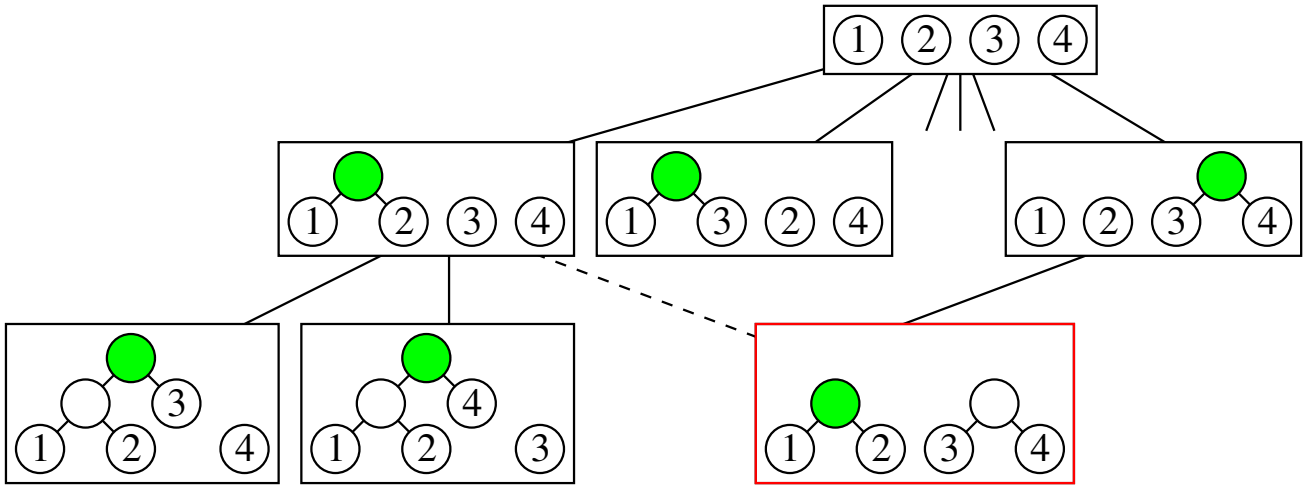


A new, more efficient search space tree



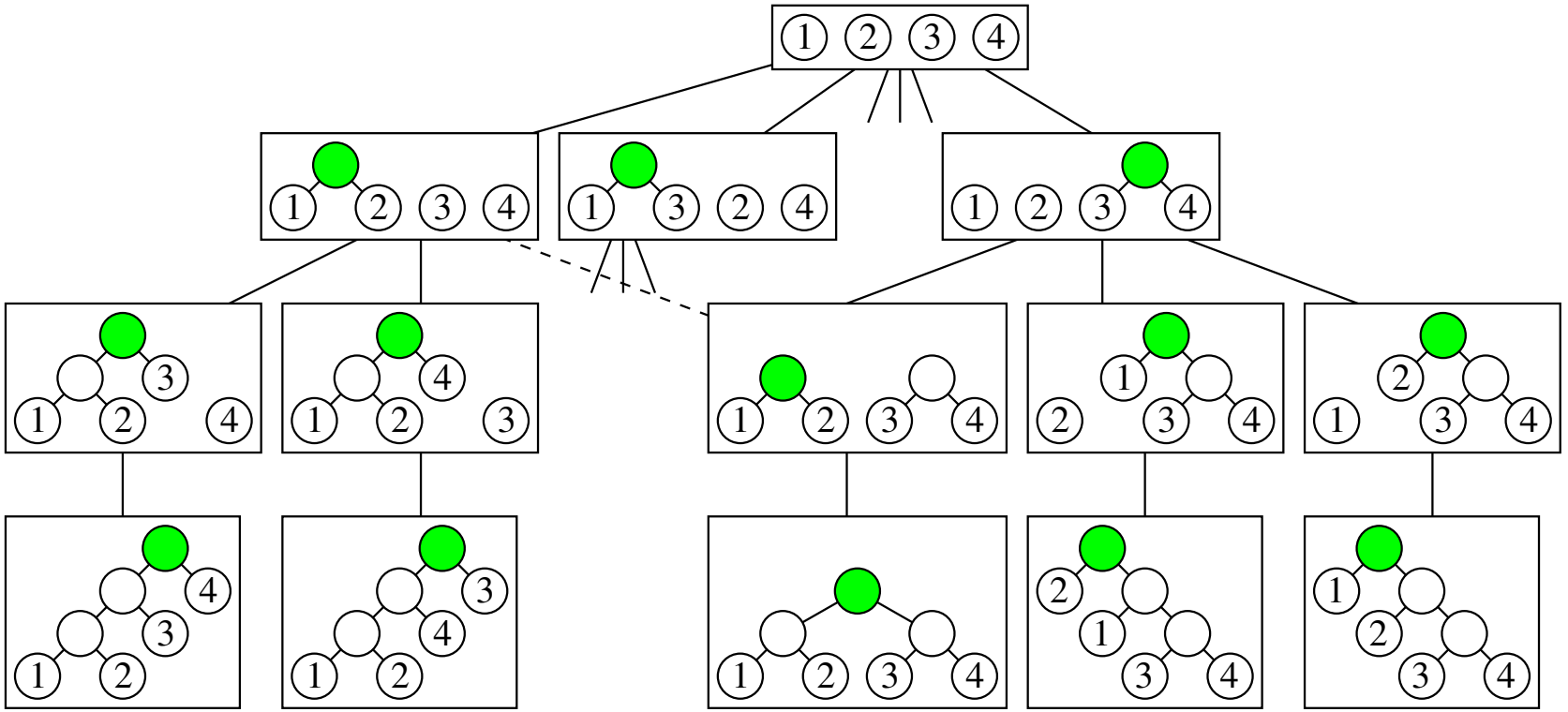
First challenge: The same node can be obtained from two different sequences of merge operations.

A new, more efficient search space tree



First challenge: The same node can be obtained from two different sequences of merge operations.

A new, more efficient search space tree



Second challenge: We need to count the number of nodes in the search space tree.

A new, more efficient search space tree

- The new algorithm develops the search space tree bottom up, which flows along with the natural bottom up direction of Small Parsimony.

A new, more efficient search space tree

- The new algorithm develops the search space tree bottom up, which flows along with the natural bottom up direction of Small Parsimony.
- We first define a graph $\mathcal{G}_n$, and then, using canonical representation for each node, we transform $\mathcal{G}_n$ into a tree $\mathcal{T}_n$ by removing redundant edges.

A new, more efficient search space tree

- The new algorithm develops the search space tree bottom up, which flows along with the natural bottom up direction of Small Parsimony.
- We first define a graph $\mathcal{G}_n$, and then, using canonical representation for each node, we transform $\mathcal{G}_n$ into a tree $\mathcal{T}_n$ by removing redundant edges.
- Using combinatorial analysis we showed that the size of $\mathcal{T}_n$ is approximately $e \cdot (2n-3)!!$.

A new, more efficient search space tree

- The new algorithm develops the search space tree bottom up, which flows along with the natural bottom up direction of Small Parsimony.
- We first define a graph $\mathcal{G}_n$, and then, using canonical representation for each node, we transform $\mathcal{G}_n$ into a tree $\mathcal{T}_n$ by removing redundant edges.
- Using combinatorial analysis we showed that the size of $\mathcal{T}_n$ is approximately $e \cdot (2n-3)!!$.
- Traversing the search space tree requires performing exactly one assignment operation per node, thus the complexity of our new MP algorithm is $\Theta((2n-3)!!)$

From $\mathcal{G}_n$ to $\mathcal{T}_n$

For a node u in the search space, we denote by F_u the corresponding forest.

- For a tree T in a forest define the *label* of T to be

$$\text{label}(T) = \min\{\text{label}(v) : v \text{ is a leaf in } T\}.$$

- The label of a forest F is

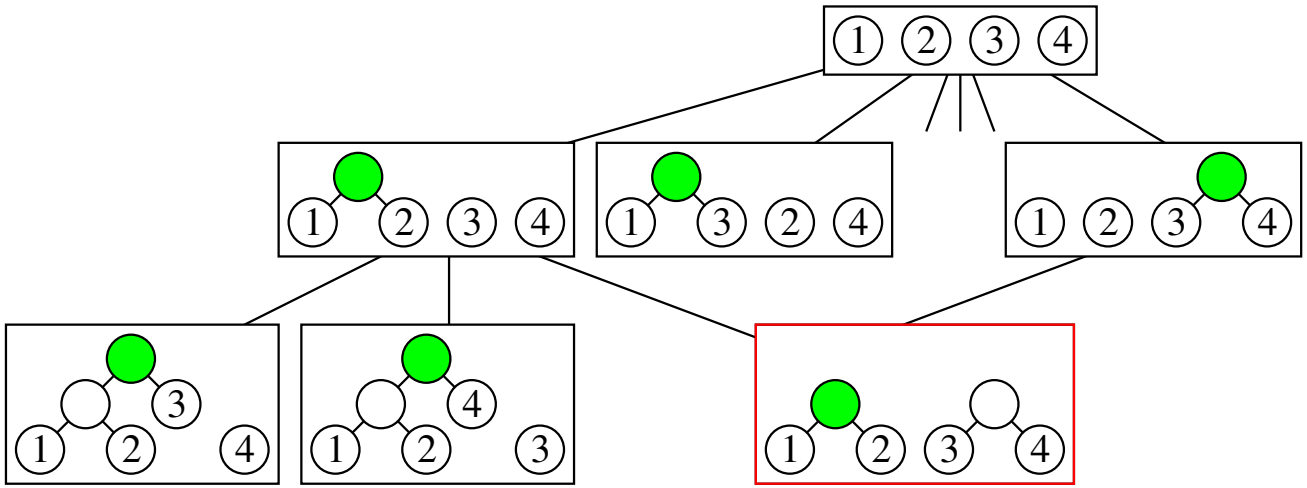
$$\text{label}(F) = \min\{\text{label}(T) : T \text{ is a non-singleton tree in } F\}.$$

- For a node u. let T_u denote the tree in F_u for which $\text{label}(F_u) = \text{label}(T_u)$.

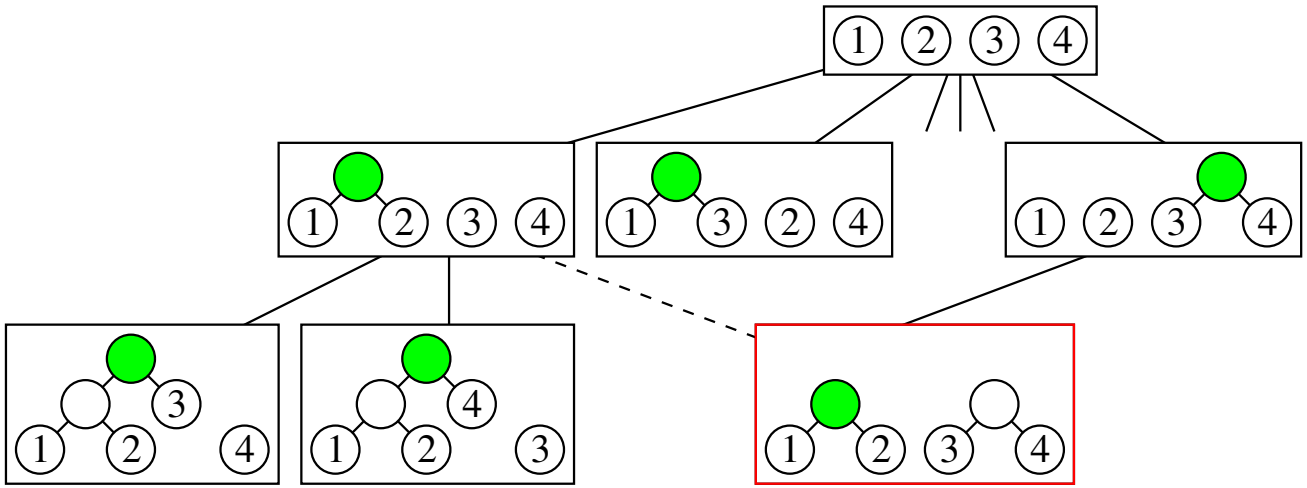
Definition

The search space tree $\mathcal{T}_n$ is a tree whose nodes are the nodes of $\mathcal{G}_n$. A node v is the parent of a node u in $\mathcal{T}_n$ if F_v is obtained from F_u by deleting the root of T_u.

From $\mathcal{G}_n$ to $\mathcal{T}_n$



From $\mathcal{G}_n$ to $\mathcal{T}_n$



From $\mathcal{G}_n$ to $\mathcal{T}_n$

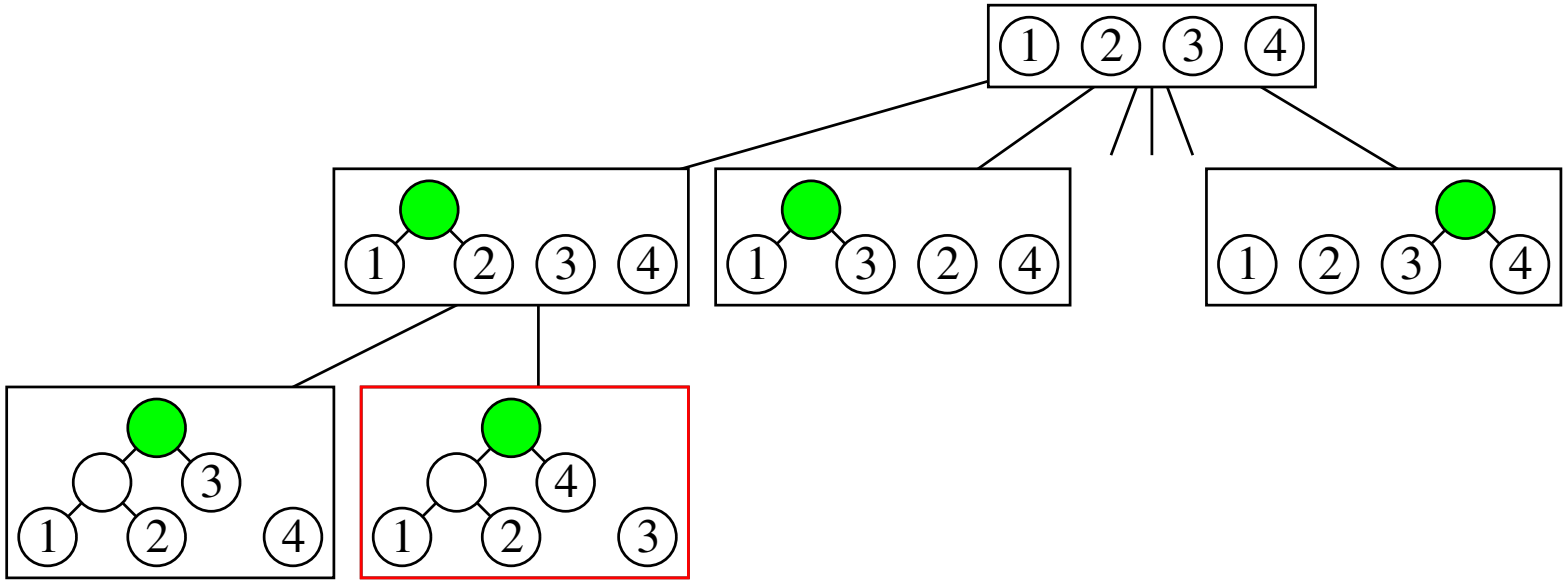
The definition of $\mathcal{T}_n$ gives a characterization for the parent of a node in the tree. In order to perform a top-down traversal of the search space tree, we need a characterization for the children of a node. Such characterization is given in the following lemma.

Lemma 2

A node u is a child of a node v in $\mathcal{T}_n$ if and only if F_u is obtained from F_v by merging two trees T_1 and T_2 from F_v with $\mathrm{label}(T_1) < \mathrm{label}(T_2)$ and either

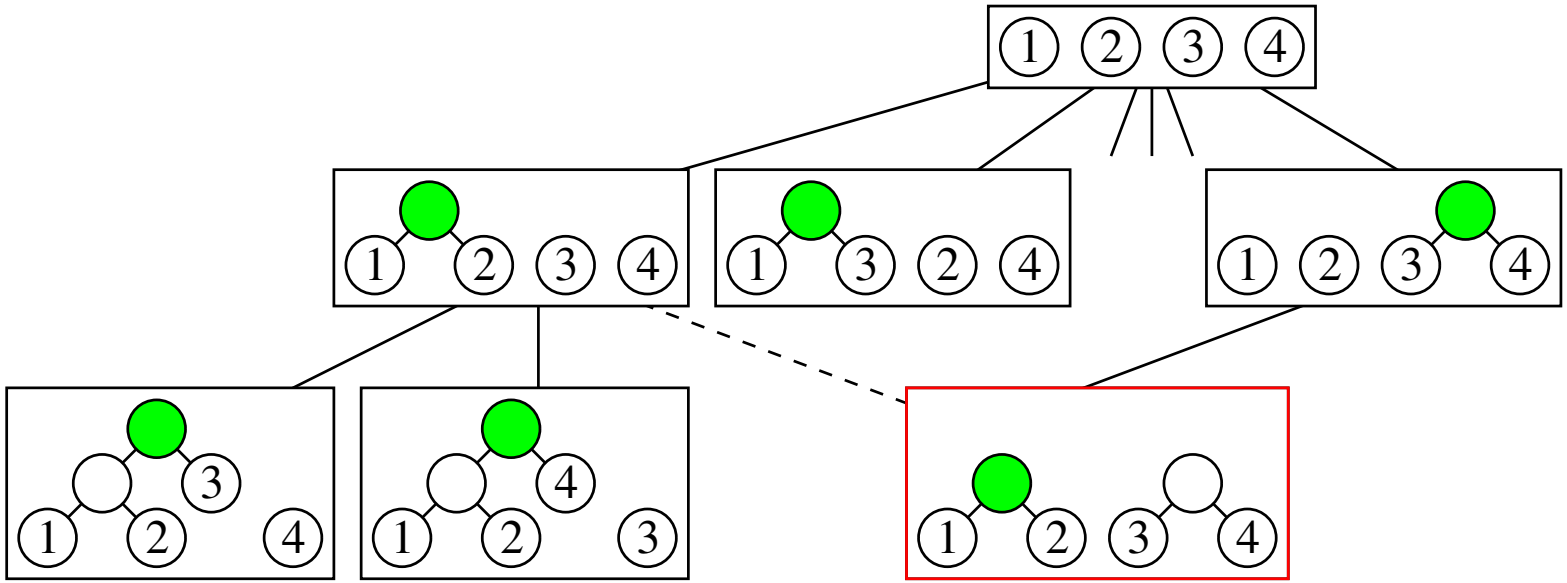
1. $T_1 = T_v$ (and in particular, T_1 is not a singleton), or
2. T_1 is a singleton and $\mathrm{label}(T_1) < \mathrm{label}(T_v)$.

From $\mathcal{G}_n$ to $\mathcal{T}_n$



1. $T_1 = T_v$ (and in particular, T_1 is not a singleton), or
2. T_1 is a singleton and $\mathrm{label}(T_1) < \mathrm{label}(T_v)$.

From $\mathcal{G}_n$ to $\mathcal{T}_n$



1. $T_1 = T_v$ (and in particular, T_1 is not a singleton), or
2. T_1 is a singleton and $\mathrm{label}(T_1) < \mathrm{label}(T_v)$.

Complexity of the new search space

- Let A_i^n denote the number of nodes in level i of $\mathcal{T}_n$.
- Let $L_{i,k}^n$ denote the number of nodes v in level i for which $\mathrm{label}(F_v) = k$.
- For example, $A_0^n = 1$, $A_1^n = \binom{n}{2}$, and $A_{n-1}^n = (2n-3)!!$.
- Let μ_n denote the size of $\mathcal{T}_n$. i.e. $\mu_n = \sum_{i=0}^{n-1} A_i^n$.

Lemma 3

$L_{i+1,k}^n = (n-i-k)\sum_{l=k}^{n-i} L_{i,l}^n$.

Lemma 4

$L_{i,k}^n = (2i-1)!!\binom{n-k+i-1}{2i-1}$.

Lemma 5

$A_i^n = (2i-1)!!\binom{n+i-1}{2i}$.

Complexity of the new search space

- Let A_i^n denote the number of nodes in level i of $\mathcal{T}_n$.
- Let $L_{i,k}^n$ denote the number of nodes v in level i for which $\mathrm{label}(F_v) = k$.
- For example, $A_0^n = 1$, $A_1^n = \binom{n}{2}$, and $A_{n-1}^n = (2n-3)!!$.
- Let μ_n denote the size of $\mathcal{T}_n$. i.e. $\mu_n = \sum_{i=0}^{n-1} A_i^n$.

Lemma 3

$L_{i+1,k}^n = (n-i-k)\sum_{l=k}^{n-i} L_{i,l}^n$.

Lemma 4

$L_{i,k}^n = (2i-1)!!\binom{n-k+i-1}{2i-1}$.

Lemma 5

$A_i^n = (2i-1)!!\binom{n+i-1}{2i}$.

Theorem 3

The assignment operations complexity for solving MP using the new search space is $(1+o(1))\cdot e\cdot(2n-3)!!$, i.e. $\mu_n \sim e\cdot(2n-3)!!$.

Summary

- We studied the classical problem of exact Maximum Parsimony, focusing on the running time complexity of various algorithms for the problem.
- The first approach proposed by Cavalli-Sforza and Edwards yields an assignment operations complexity of $(n-1) \cdot (2n-3)!!$.
- The second approach we analyzed was proposed by Hendy and Penny. Its theoretical running time complexity has not been previously analyzed. We showed that the assignment operations complexity of this approach is smaller by a factor of $\Theta(\sqrt{n})$.
- We proposed a new, faster MP approach, whose assignment operations complexity is smaller by a factor of $\Theta(\sqrt{n})$ than the complexity of the Hendy and Penny approach.

Thank You