Near-optimal Algorithm for Constructing Greedy Consensus Tree

Hongxun Wu

Institute for Interdisciplinary Information Sciences, Tsinghua University

Phylogenetic tree

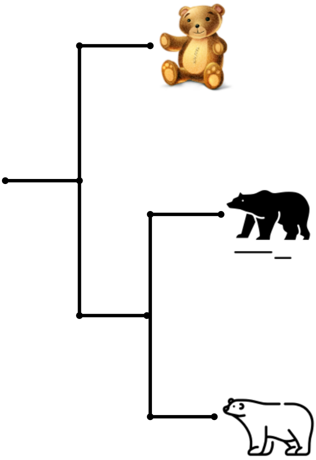
- **Phylogenetic tree** represents evolutionary relations.

Phylogenetic tree

- **Phylogenetic tree** represents evolutionary relations.
- **Leaves** of the tree represent species.

Phylogenetic tree

- **Phylogenetic tree** represents evolutionary relations.
- **Leaves** of the tree represent species.
- Each **inner node** represents the least common ancestor of all leaves in its subtree.



Phylogenetic tree

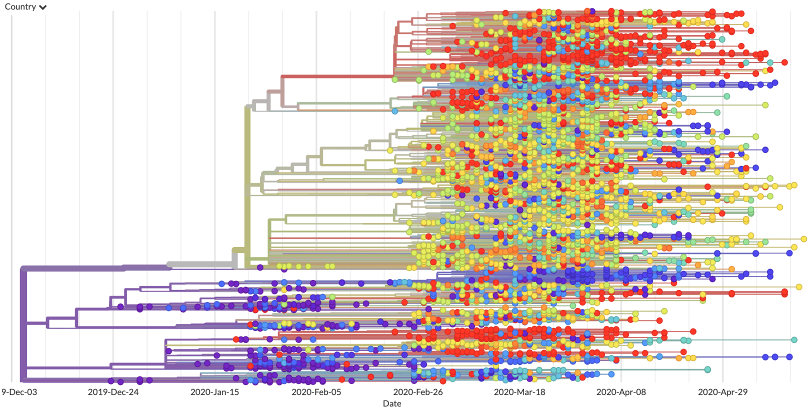
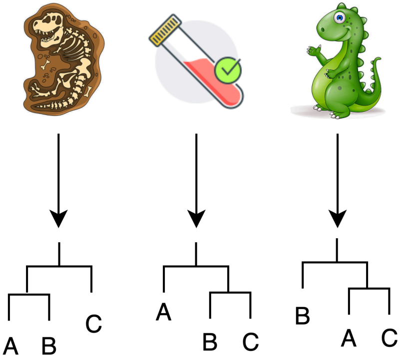


Figure: Phylogenetic Tree of Covid-19[1]

[1] Genomic epidemiology of novel coronavirus. 2020. URL: https://nextstrain.org/ncov/global.

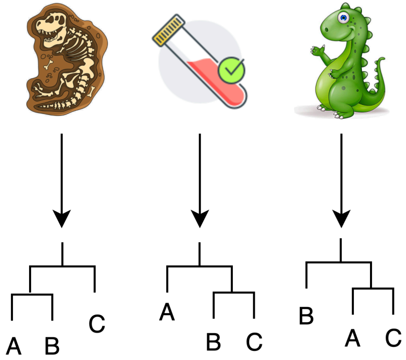
Consensus tree

- Phylogenetic trees from different sources may conflicts



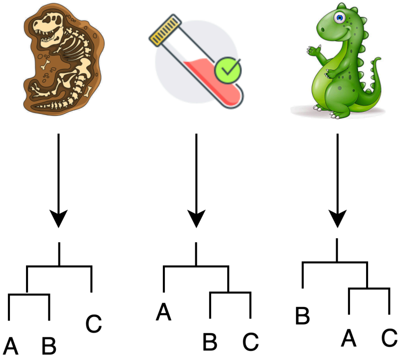
Consensus tree

- Phylogenetic trees from different sources may conflicts
- Consensus tree summarizes their structures to a single tree.



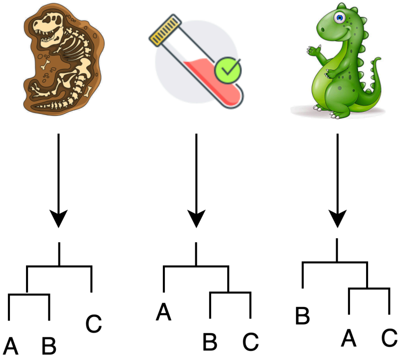
Consensus tree

- Phylogenetic trees from different sources may conflicts
- Consensus tree summarizes their structures to a single tree.
- Let k be the number of phylogenetic trees in the input and n be the number of species in each of them.



Consensus tree

- Phylogenetic trees from different sources may conflicts
- Consensus tree summarizes their structures to a single tree.
- Let k be the number of phylogenetic trees in the input and n be the number of species in each of them.
- The input size is $\Theta(kn)$.



Consensus tree

- Many consensus tree methods were proposed.

Consensus tree method	Running time
Adam's consensus tree	$O(kn \log n)$
Strict consensus tree	$O(kn)$
Loose consensus tree	$O(kn)$
Frequency difference consensus tree	$O(kn \log^2 n)$
Majority-rule consensus tree	$O(kn \log k)$, Randomized $O(kn)$
Majority-rule $(+)$ consensus tree	$O(kn)$
Local consensus tree	$O(kn^3)$
R^* consensus tree	$O(n^2 \log^{k+2} n)$
Greedy consensus tree	$O(kn^{1.5})$, $O(k^2 n)$

Consensus tree

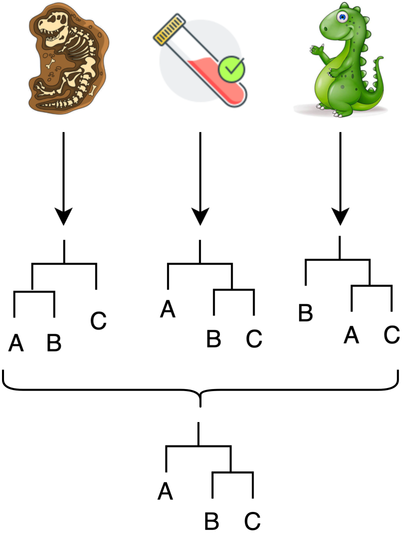
- Many consensus tree methods were proposed.

Consensus tree method	Running time
Adam's consensus tree	$O(kn \log n)$
Strict consensus tree	$O(kn)$
Loose consensus tree	$O(kn)$
Frequency difference consensus tree	$O(kn \log^2 n)$
Majority-rule consensus tree	$O(kn \log k)$, Randomized $O(kn)$
Majority-rule $(+)$ consensus tree	$O(kn)$
Local consensus tree	$O(kn^3)$
R^* consensus tree	$O(n^2 \log^{k+2} n)$
Greedy consensus tree	$O(kn^{1.5})$, $O(k^2 n)$

- Most of them have near-optimal running time. Greedy consensus tree is one of the exceptions.

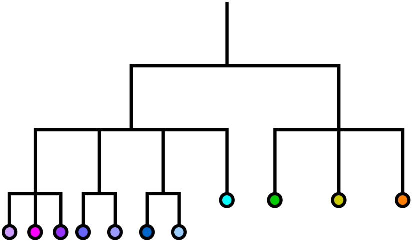
Greedy Consensus Tree

- Input: k phylogenetic trees
- Output: One consensus tree



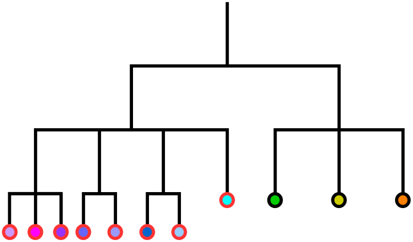
Greedy Consensus Tree

- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.



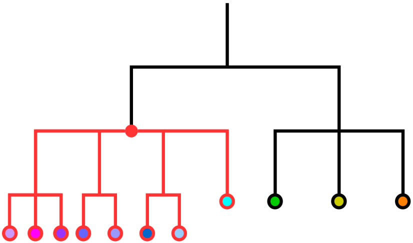
Greedy Consensus Tree

- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.



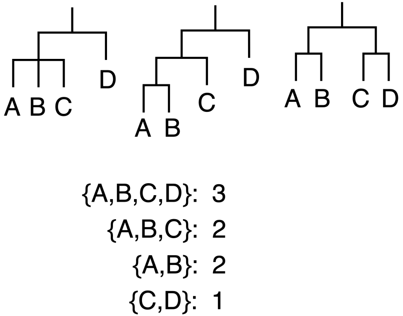
Greedy Consensus Tree

- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.



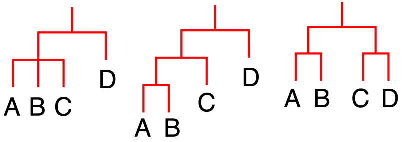
Greedy Consensus Tree

- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.
- A cluster may appear in several input phylogenetic trees.



Greedy Consensus Tree

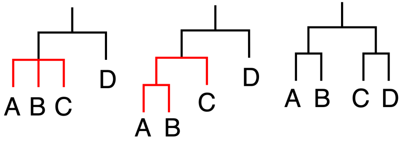
- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.
- A cluster may appear in several input phylogenetic trees.



{A,B,C,D}: 3
{A,B,C}: 2
{A,B}: 2
{C,D}: 1

Greedy Consensus Tree

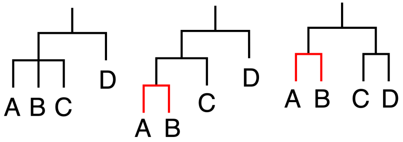
- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.
- A cluster may appear in several input phylogenetic trees.



{A,B,C,D}: 3
{A,B,C}: 2
{A,B}: 2
{C,D}: 1

Greedy Consensus Tree

- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.
- A cluster may appear in several input phylogenetic trees.



{A,B,C,D}: 3

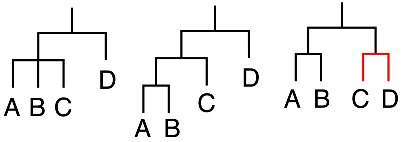
{A,B,C}: 2

{A,B}: 2

{C,D}: 1

Greedy Consensus Tree

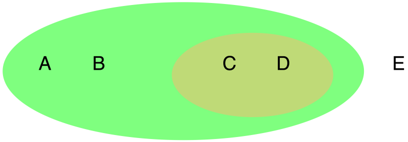
- Input: k phylogenetic trees
- Output: One consensus tree
- A **cluster** $L(v)$ is the set of all leaves in the subtree of an inner node v.
- A cluster may appear in several input phylogenetic trees.



{A,B,C,D}: 3
{A,B,C}: 2
{A,B}: 2
{C,D}: 1

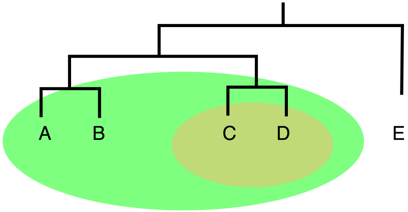
Greedy Consensus Tree

- Two clusters are consistent if and only if one of the following holds:
  - They are disjoint.
  - One contains the other.



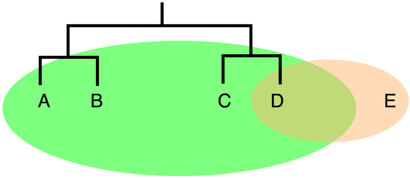
Greedy Consensus Tree

- Two clusters are consistent if and only if one of the following holds:
  - They are disjoint.
  - One contains the other.
- In other words, they can simultaneously occur in a valid phylogenetic tree.



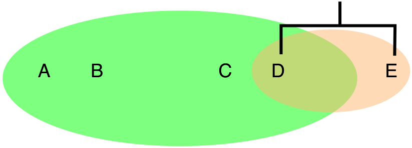
Greedy Consensus Tree

- Two clusters are consistent if and only if one of the following holds:
  - They are disjoint.
  - One contains the other.
- In other words, they can simultaneously occur in a valid phylogenetic tree.



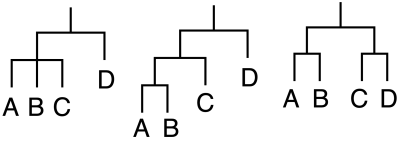
Greedy Consensus Tree

- Two clusters are consistent if and only if one of the following holds:
  - They are disjoint.
  - One contains the other.
- In other words, they can simultaneously occur in a valid phylogenetic tree.



Greedy Consensus Tree

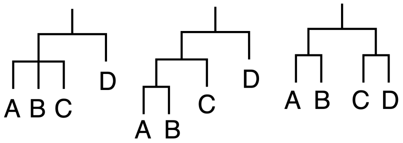
- A **greedy consensus tree** is defined as the output of following greedy algorithm:



{A,B,C,D}: 3
{A,B,C}: 2
{A,B}: 2
{C,D}: 1

Greedy Consensus Tree

- A **greedy consensus tree** is defined as the output of following greedy algorithm:
  1. First, count the frequency of each cluster in the input trees.



{A,B,C,D}: 3
{A,B,C}: 2
{A,B}: 2
{C,D}: 1

Greedy Consensus Tree

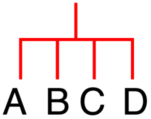
- A **greedy consensus tree** is defined as the output of following greedy algorithm:
  1. First, count the frequency of each cluster in the input trees.
  2. Second, try to insert each cluster into our consensus tree one by one from high frequency to low frequency (ties are broken arbitrarily).

{A,B,C,D}: 3

{A,B,C}: 2

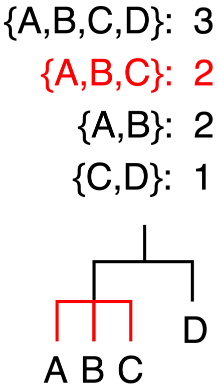
{A,B}: 2

{C,D}: 1



Greedy Consensus Tree

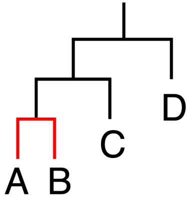
- A **greedy consensus tree** is defined as the output of following greedy algorithm:
  1. First, count the frequency of each cluster in the input trees.
  2. Second, try to insert each cluster into our consensus tree one by one from high frequency to low frequency (ties are broken arbitrarily).



Greedy Consensus Tree

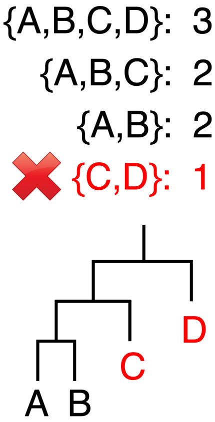
- A **greedy consensus tree** is defined as the output of following greedy algorithm:
  1. First, count the frequency of each cluster in the input trees.
  2. Second, try to insert each cluster into our consensus tree one by one from high frequency to low frequency (ties are broken arbitrarily).

{A,B,C,D}: 3
{A,B,C}: 2
{A,B}: 2
{C,D}: 1

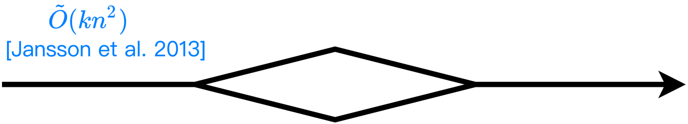


Greedy Consensus Tree

- A **greedy consensus tree** is defined as the output of following greedy algorithm:
  1. First, count the frequency of each cluster in the input trees.
  2. Second, try to insert each cluster into our consensus tree one by one from high frequency to low frequency (ties are broken arbitrarily).

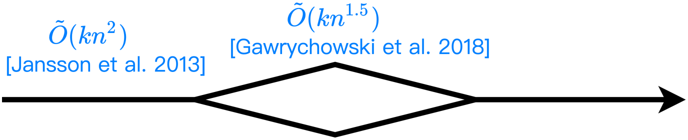


Previous Works



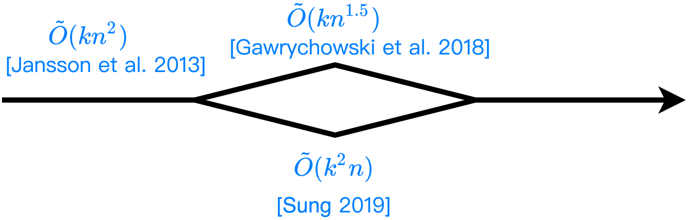
- Jesper Jansson, Chuanqi Shen, Wing-Kin Sung. *SODA 2013*

Previous Works



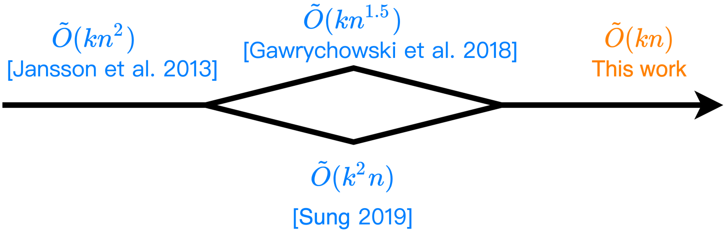
- Jesper Jansson, Chuanqi Shen, Wing-Kin Sung. *SODA 2013*
- Pawel Gawrychowski, Gad M. Landau, Wing-Kin Sung, Oren Weimann. *ICALP 2018*

Previous Works



- Jesper Jansson, Chuanqi Shen, Wing-Kin Sung. *SODA 2013*
- Pawel Gawrychowski, Gad M. Landau, Wing-Kin Sung, Oren Weimann. *ICALP 2018*
- Wing-Kin Sung. *WALCOM 2019*

Previous Works



- Jesper Jansson, Chuanqi Shen, Wing-Kin Sung. *SODA 2013*
- Pawel Gawrychowski, Gad M. Landau, Wing-Kin Sung, Oren Weimann. *ICALP 2018*
- Wing-Kin Sung. *WALCOM 2019*

Characterization of consistency

- Recall greedy consensus tree construction has two steps:

Characterization of consistency

- Recall greedy consensus tree construction has two steps:
  - Count the frequency of each cluster and sort them.

Characterization of consistency

- Recall greedy consensus tree construction has two steps:
  - Count the frequency of each cluster and sort them.
  - Insert each cluster into consensus tree if they are consistent.

Characterization of consistency

- Recall greedy consensus tree construction has two steps:
  - Count the frequency of each cluster and sort them.
  - Insert each cluster into consensus tree if they are consistent.
- Handle first part within $\tilde{O}(kn)$ time is simple.

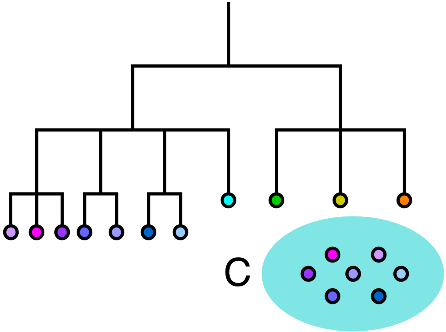
Characterization of consistency

- Recall greedy consensus tree construction has two steps:
  - Count the frequency of each cluster and sort them.
  - Insert each cluster into consensus tree if they are consistent.
- Handle first part within $\tilde{O}(kn)$ time is simple.
- The hard part is to determine whether each cluster is consistent with our current consensus tree.

Characterization of consistency

Lemma

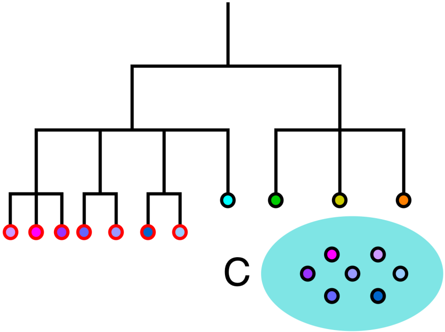
Let v be the LCA of cluster C on consensus tree



Characterization of consistency

Lemma

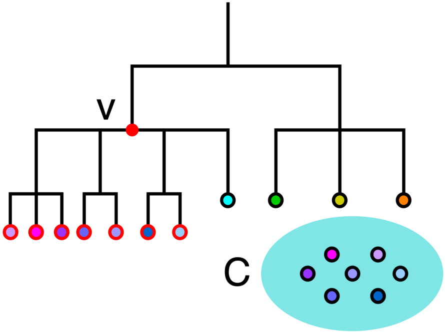
Let v be the LCA of cluster C on consensus tree



Characterization of consistency

Lemma

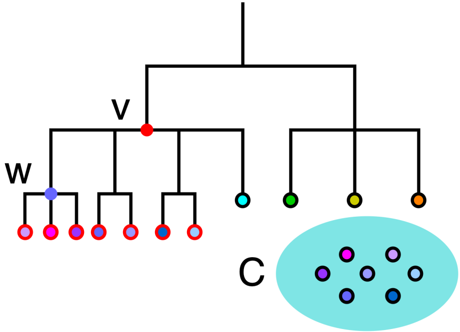
Let v be the LCA of cluster C on consensus tree



Characterization of consistency

Lemma

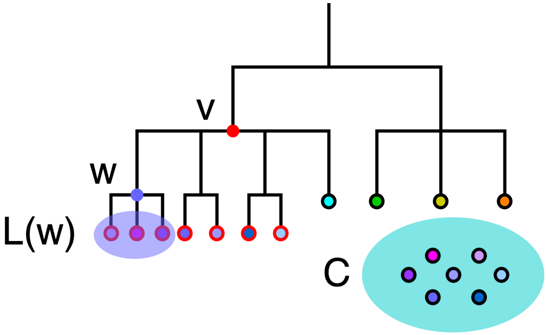
Let v be the LCA of cluster C on consensus tree and w be a child of v.



Characterization of consistency

Lemma

Let v be the LCA of cluster C on consensus tree and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

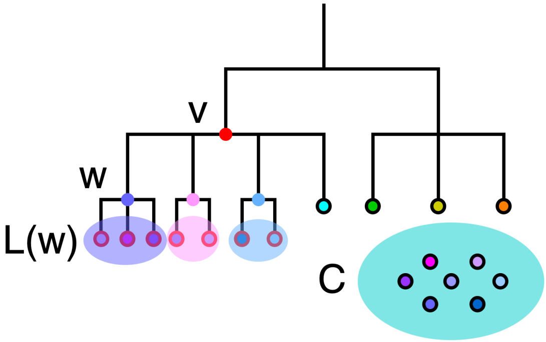


Characterization of consistency

Lemma

Let v be the LCA of cluster C on consensus tree and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

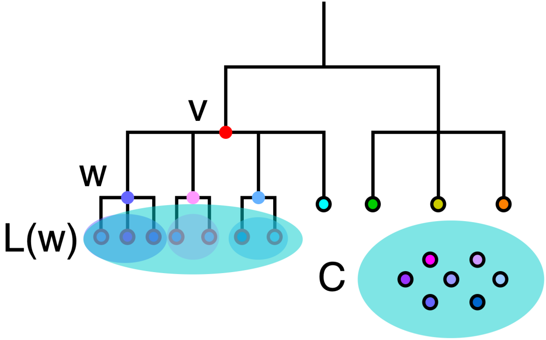


Characterization of consistency

Lemma

Let v be the LCA of cluster C on consensus tree and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

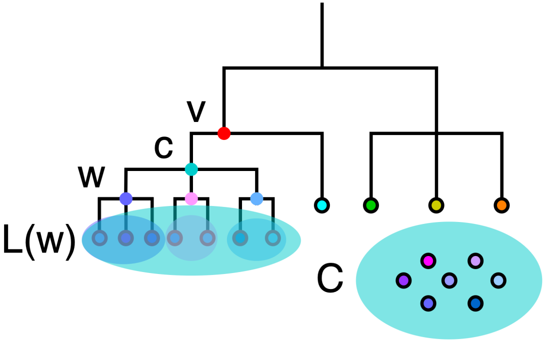


Characterization of consistency

Lemma

Let v be the LCA of cluster C on consensus tree and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.



Implementing the characterization

- To use it, we need to implement the following steps:

Implementing the characterization

- To use it, we need to implement the following steps:
  1. Query the LCA of a new cluster to check the consistency.

Implementing the characterization

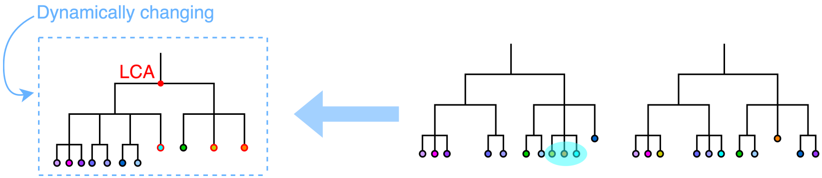
- To use it, we need to implement the following steps:
  1. Query the LCA of a new cluster to check the consistency.
  2. If consistent, insert a new node to the consensus tree.

Main difficulty

- Main difficulty: maintaining LCA of kn static sets on a dynamic tree is hard.

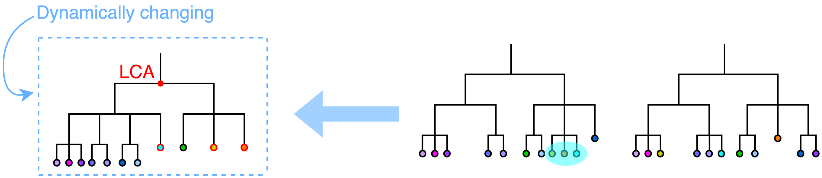
Main difficulty

- Main difficulty: maintaining LCA of kn static sets on a dynamic tree is hard.



Main difficulty

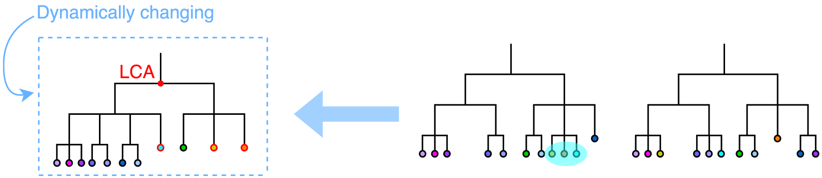
- Main difficulty: maintaining LCA of kn static sets on a dynamic tree is hard.



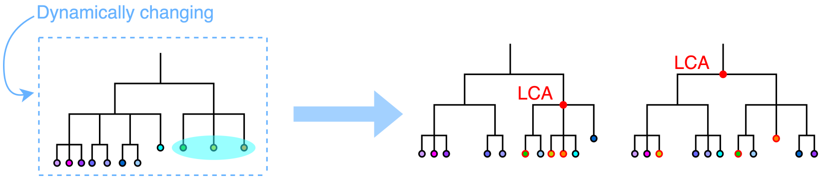
- Our Approach: How about maintaining LCAs of clusters in the dynamic tree on k static trees.

Main difficulty

- Main difficulty: maintaining LCA of kn static sets on a dynamic tree is hard.



- Our Approach: How about maintaining LCAs of clusters in the dynamic tree on k static trees.

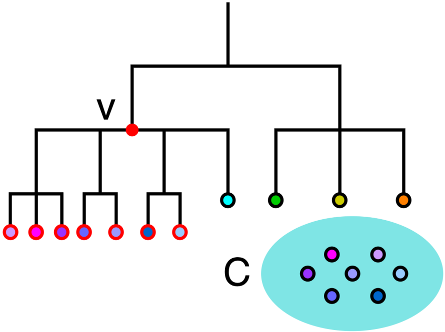


Modified Characterization

Lemma

Let v be the LCA of cluster C on consensus tree and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

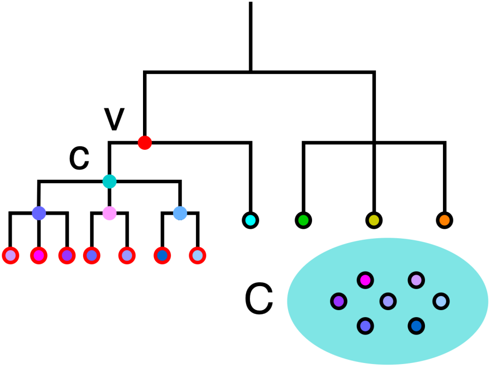


Modified Characterization

Lemma

Let v be the LCA of cluster C on consensus tree and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

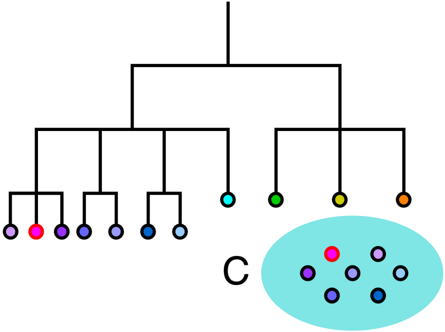


Modified Characterization

Lemma

Let v be the deepest ancestor of a leaf $x_0 \in C$ s.t. $L(v) \not\subseteq C$ and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

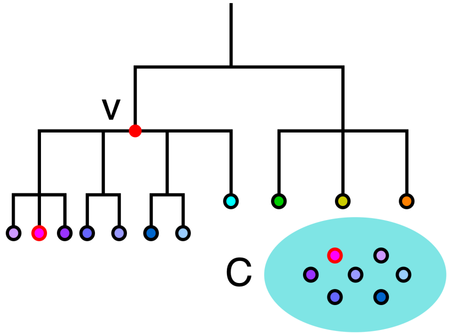


Modified Characterization

Lemma

Let v be the deepest ancestor of a leaf $x_0 \in C$ s.t. $L(v) \not\subseteq C$ and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

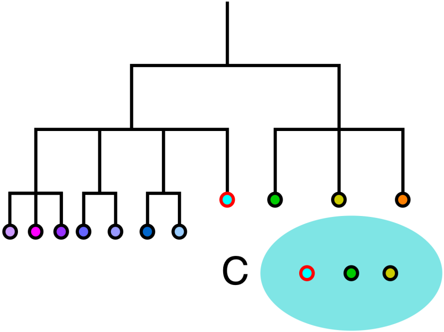


Modified Characterization

Lemma

Let v be the deepest ancestor of a leaf $x_0 \in C$ s.t. $L(v) \not\subseteq C$ and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

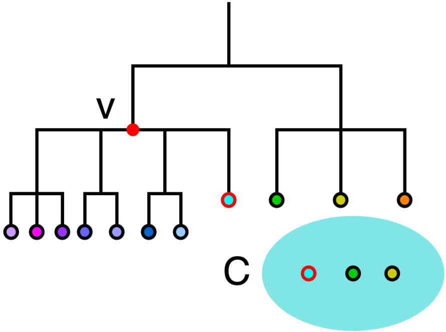


Modified Characterization

Lemma

Let v be the deepest ancestor of a leaf $x_0 \in C$ s.t. $L(v) \not\subseteq C$ and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

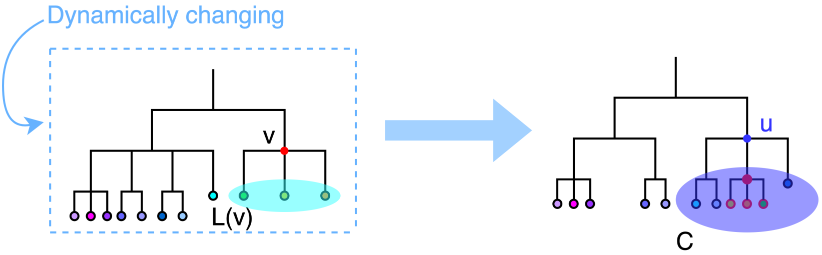
C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.



Modified Characterization

Fact

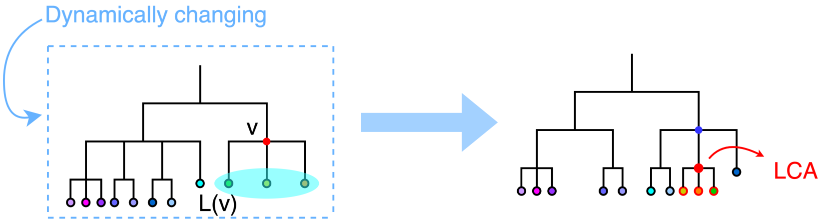
Suppose C is $L(u)$ of inner node u on input phylogenetic tree T_i.



Modified Characterization

Fact

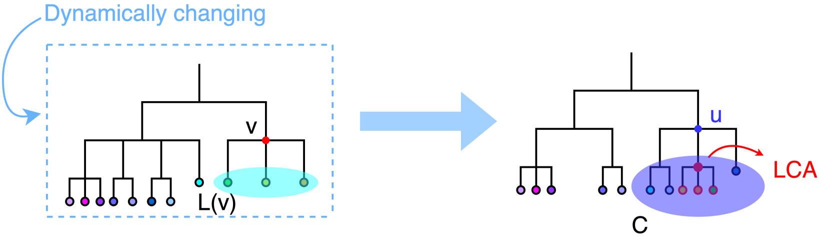
Suppose C is $L(u)$ of inner node u on input phylogenetic tree T_i.
$L(v) \subseteq C$ if and only if the lca of $L(v)$ on T_i



Modified Characterization

Fact

Suppose C is $L(u)$ of inner node u on input phylogenetic tree T_i.
$L(v) \subseteq C$ if and only if the lca of $L(v)$ on T_i is in the subtree of u.



Modified Characterization

Lemma

Let v be the deepest ancestor of a leaf $x_0 \in C$ s.t. $L(v) \nsubseteq C$ and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

- To find deepest such ancestor we can binary search the path from x_0 to root.

Modified Characterization

Lemma

Let v be the deepest ancestor of a leaf $x_0 \in C$ s.t. $L(v) \nsubseteq C$ and w be a child of v. $L(w)$ is defined as the cluster of all species in its subtree.

C is consistent with the consensus tree if and only if it is the union of several such $L(w)$.

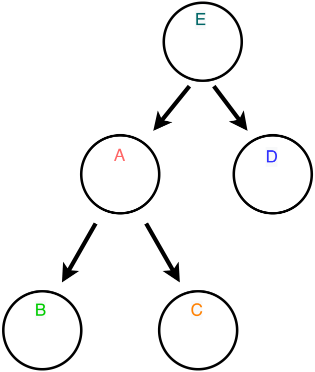
- To find deepest such ancestor we can binary search the path from x_0 to root.
- As a result, the modified characterization can be implemented by maintaining LCAs of clusters in the dynamic tree on k static trees.

BST

- Given a dynamic set S of nodes on static tree T_i, how do we support dynamic addition, deletion, and query $LCA(S)$?

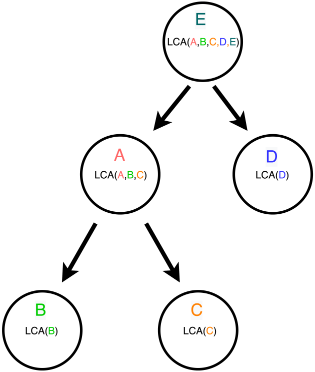
BST

- Given a dynamic set S of nodes on static tree T_i, how do we support dynamic addition, deletion, and query $LCA(S)$?
- Every node in the BST corresponds to a node in S.



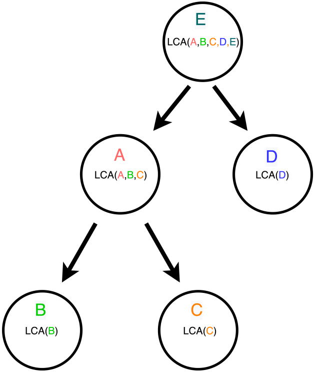
BST

- Given a dynamic set S of nodes on static tree T_i, how do we support dynamic addition, deletion, and query $LCA(S)$?
- Every node in the BST corresponds to a node in S.
- At each node, we maintain the LCA of all nodes in its subtree on BST.



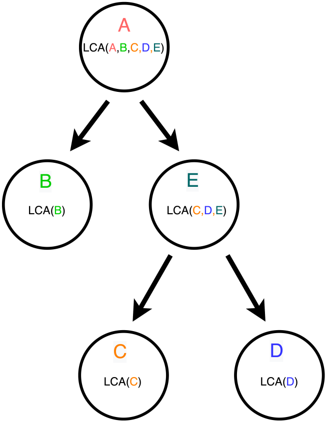
BST

- Given a dynamic set S of nodes on static tree T_i, how do we support dynamic addition, deletion, and query $LCA(S)$?
- Every node in the BST corresponds to a node in S.
- At each node, we maintain the LCA of all nodes in its subtree on BST.
  - This LCA can be computed from the two LCAs maintained at its children.



BST

- Given a dynamic set S of nodes on static tree T_i, how do we support dynamic addition, deletion, and query $LCA(S)$?
- Every node in the BST corresponds to a node in S.
- At each node, we maintain the LCA of all nodes in its subtree on BST.
  - This LCA can be computed from the two LCAs maintained at its children.
- When the children of a node changes, we recompute the LCA of its two children.

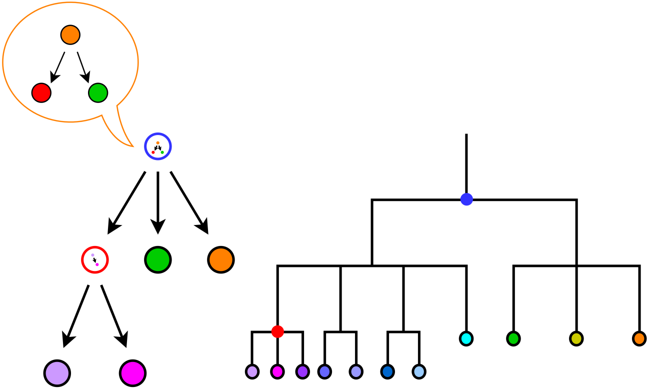


Dynamic Tree + BST

- We use an arbitrary dynamic tree to maintain the consensus tree.

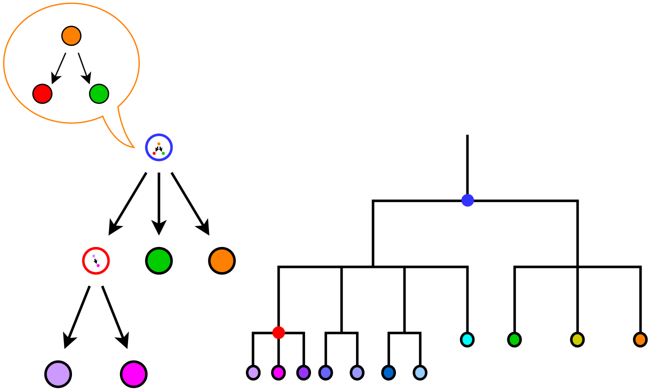
Dynamic Tree + BST

- We use an arbitrary dynamic tree to maintain the consensus tree.
- At the node v of the consensus tree, for each input tree T_i, we maintain a BST and the LCA of $L(v)$.



Dynamic Tree + BST

- We use an arbitrary dynamic tree to maintain the consensus tree.
- At the node v of the consensus tree, for each input tree T_i, we maintain a BST and the LCA of $L(v)$.
  - Each node of the BST is the LCA of one of its children.

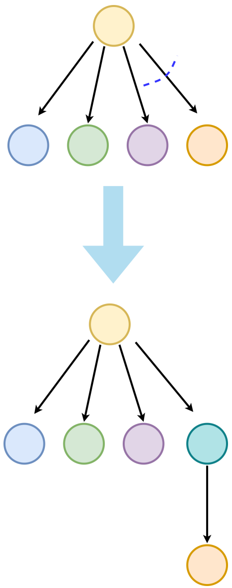


Insert a new node

- While inserting a new node, we split the BST at its parent by deletions.

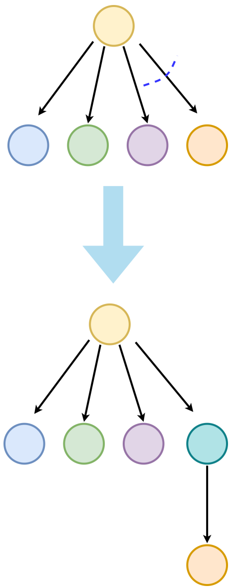
Insert a new node

- While inserting a new node, we split the BST at its parent by deletions.
- If most of the children remain



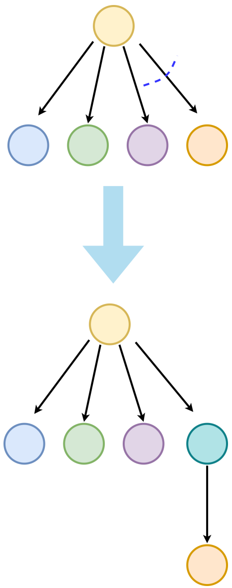
Insert a new node

- While inserting a new node, we split the BST at its parent by deletions.
- If most of the children remain
  - Delete those children in the BST.



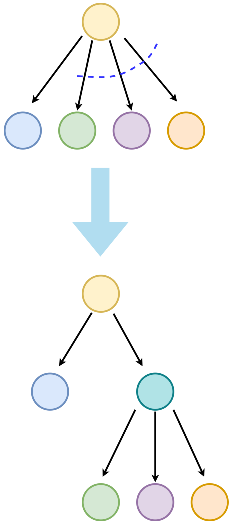
Insert a new node

- While inserting a new node, we split the BST at its parent by deletions.
- If most of the children remain
  - Delete those children in the BST.
- If most of the children go to the new node



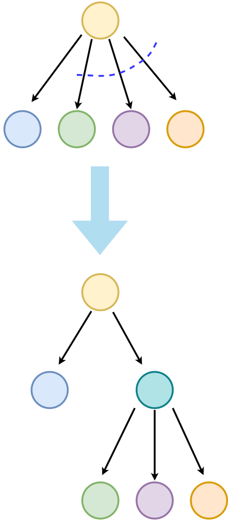
Insert a new node

- While inserting a new node, we split the BST at its parent by deletions.
- If most of the children remain
  - Delete those children in the BST.
- If most of the children go to the new node
  - Delete the other children in the BST,



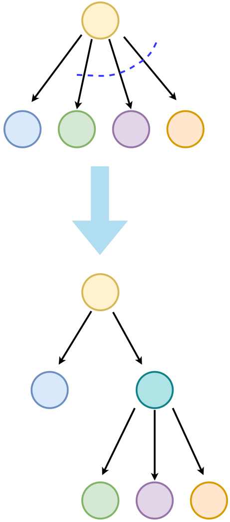
Insert a new node

- While inserting a new node, we split the BST at its parent by deletions.
- If most of the children remain
  - Delete those children in the BST.
- If most of the children go to the new node
  - Delete the other children in the BST,
  - Give the BST to our new node.



Insert a new node

- While inserting a new node, we split the BST at its parent by deletions.
- If most of the children remain
  - Delete those children in the BST.
- If most of the children go to the new node
  - Delete the other children in the BST,
  - Give the BST to our new node.
- Since we always enumerate the smaller part, in total $O(kn \log n)$ deletions in BSTs.



Q & A

Questions?

Thank you!