Edit Distance between Unrooted Trees in Cubic Time

Bartłomiej Dudek[1] Paweł Gawrychowski[1]

[1]University of Wrocław, Poland

July 10, 2018

Edit distance between strings

Elementary operations:

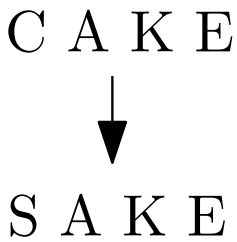
- insert a symbol
- delete a symbol
- substitute a symbol

C A K E

Edit distance between strings

Elementary operations:

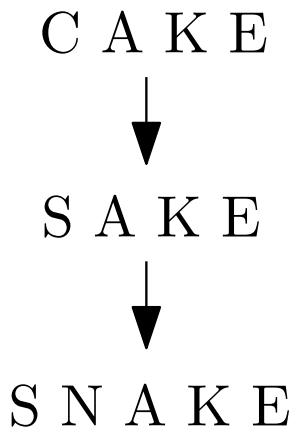
- insert a symbol
- delete a symbol
- substitute a symbol



Edit distance between strings

Elementary operations:

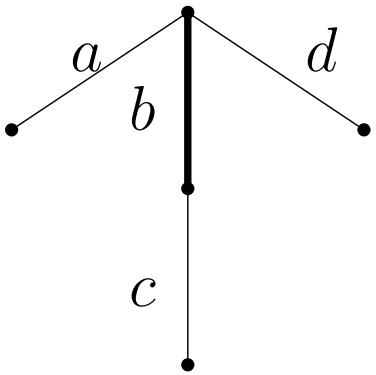
- insert a symbol
- delete a symbol
- substitute a symbol



Operations on trees

Elementary operations:

- add an edge α: $c_{ins}(\alpha)$
- remove (contract) an edge α: $c_{del}(\alpha)$
- change label of an edge from α to β: $c_{match}(\alpha, \beta)$

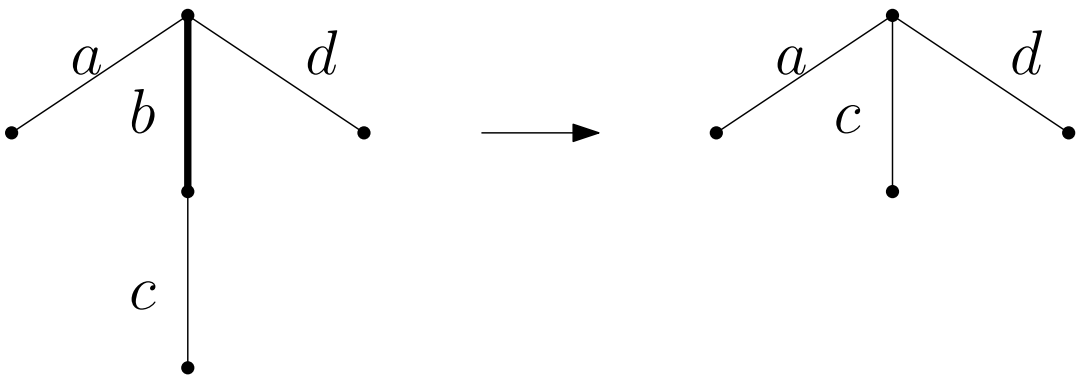


Trees are ordered!

Operations on trees

Elementary operations:

- add an edge α: $c_{ins}(\alpha)$
- remove (contract) an edge α: $c_{del}(\alpha)$
- change label of an edge from α to β: $c_{match}(\alpha, \beta)$

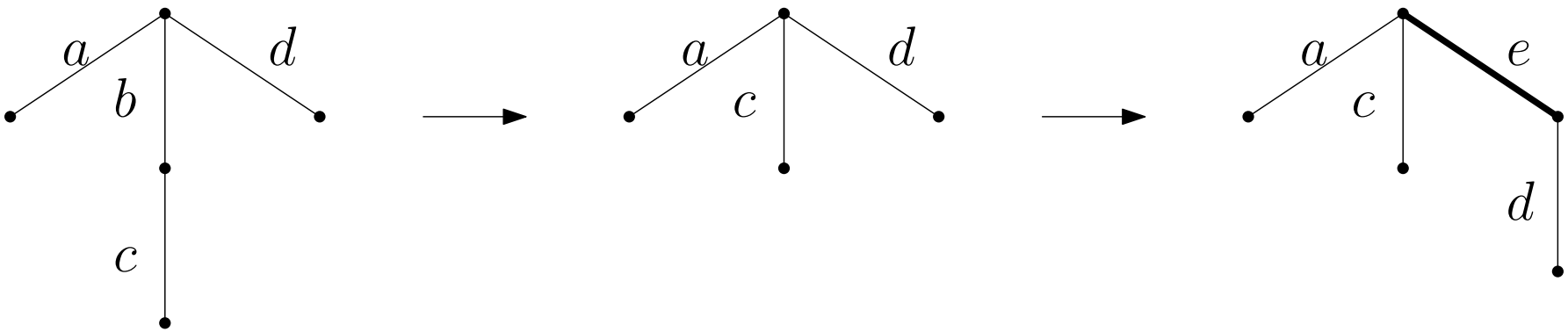


Trees are ordered!

Operations on trees

Elementary operations:

- add an edge α: $c_{ins}(\alpha)$
- remove (contract) an edge α: $c_{del}(\alpha)$
- change label of an edge from α to β: $c_{match}(\alpha, \beta)$

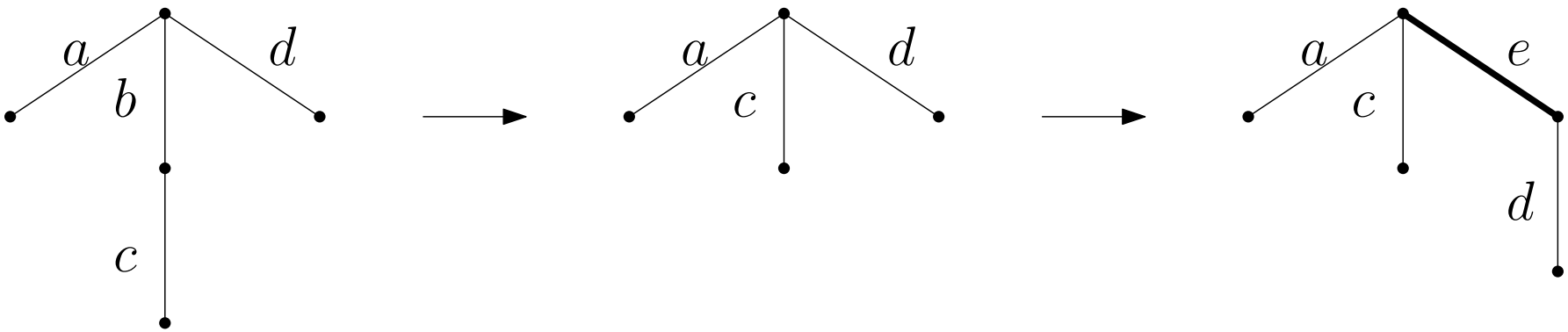


Trees are ordered!

Operations on trees

Elementary operations:

- add an edge α: $c_{ins}(\alpha)$
- remove (contract) an edge α: $c_{del}(\alpha)$
- change label of an edge from α to β: $c_{match}(\alpha, \beta)$



Trees are ordered!

Edit distance between trees

Two definitions:

Transform the first tree to the second:

$$T_1 \rightarrow T_2$$

both adding and removing edges.

Transform both trees to a common tree:

$$T_1 \rightarrow T' \leftarrow T_2$$

only removing edges.

Observation

If $c_{del}(\alpha) = c_{ins}(\alpha)$ and $c_{match}(\alpha, \beta) = c_{match}(\beta, \alpha)$ then these two definitions are equivalent.

Edit distance between trees

Two definitions:

Transform the first tree to the second:

$$T_1 \to T_2$$

both adding and removing edges.

Transform both trees to a common tree:

$$T_1 \to T' \leftarrow T_2$$

only removing edges.

Observation

If $c_{del}(\alpha) = c_{ins}(\alpha)$ and $c_{match}(\alpha, \beta) = c_{match}(\beta, \alpha)$ then these two definitions are equivalent.

Edit distance between trees

Two definitions:

Transform the first tree to the second:

$$T_1 \to T_2$$

both adding and removing edges.

Transform both trees to a common tree:

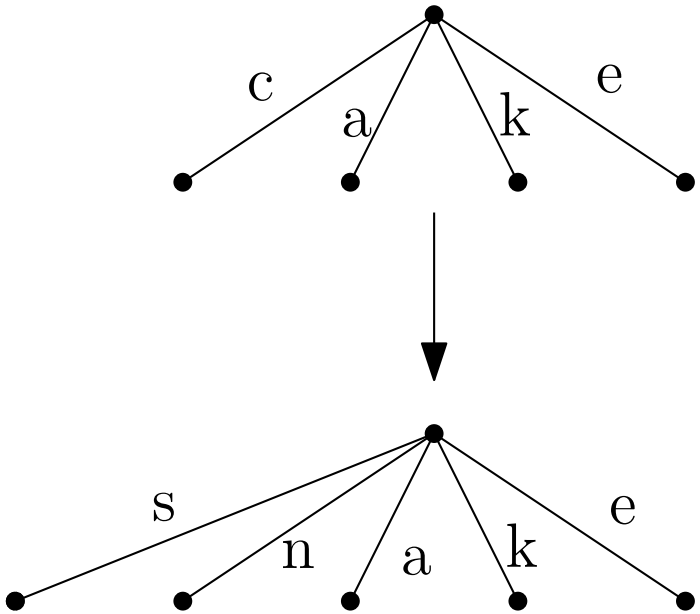
$$T_1 \to T' \leftarrow T_2$$

only removing edges.

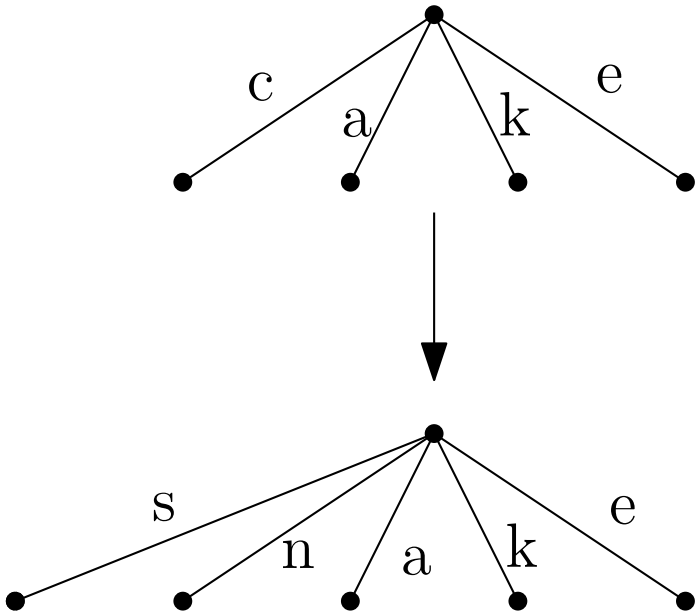
Observation

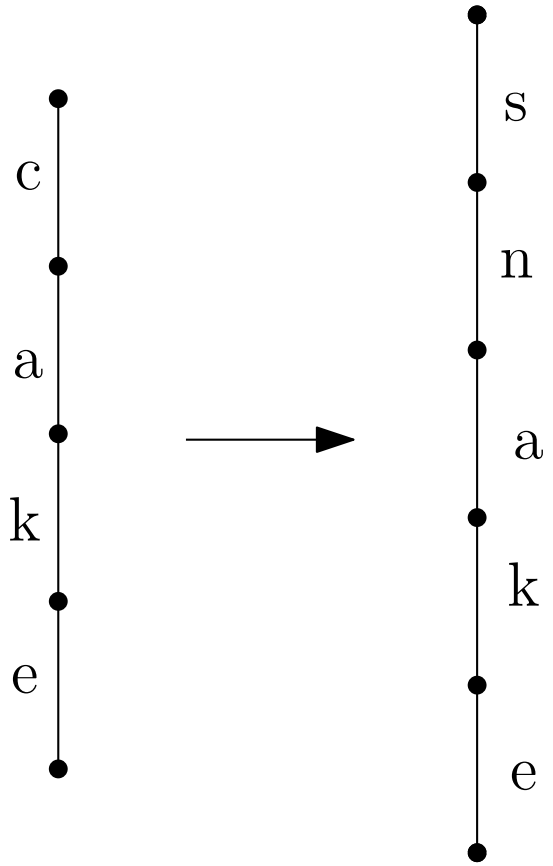
If $c_{del}(\alpha) = c_{ins}(\alpha)$ and $c_{match}(\alpha, \beta) = c_{match}(\beta, \alpha)$ then these two definitions are equivalent.

Trees generalize strings

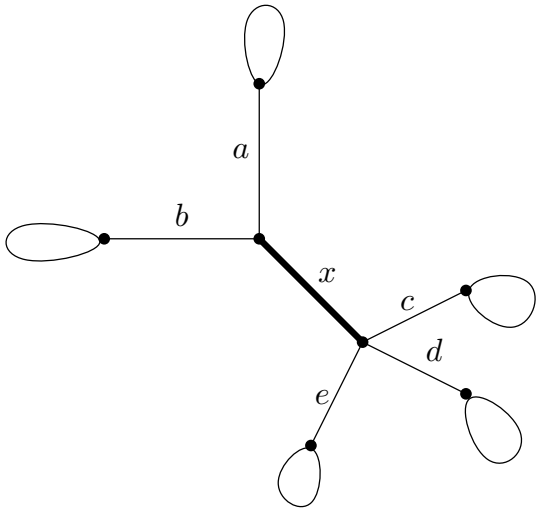


Trees generalize strings

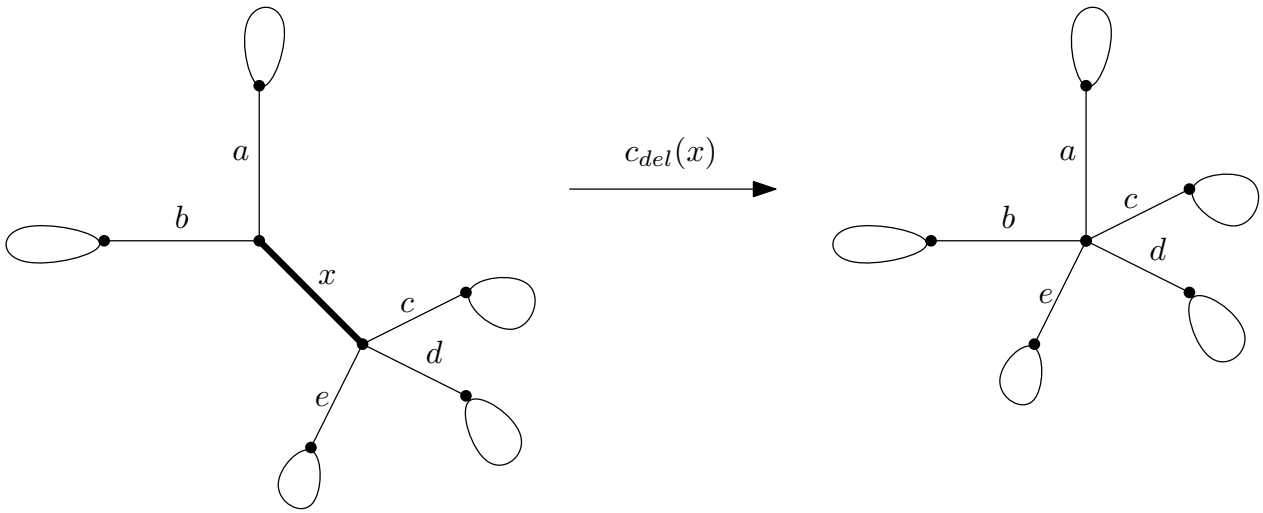




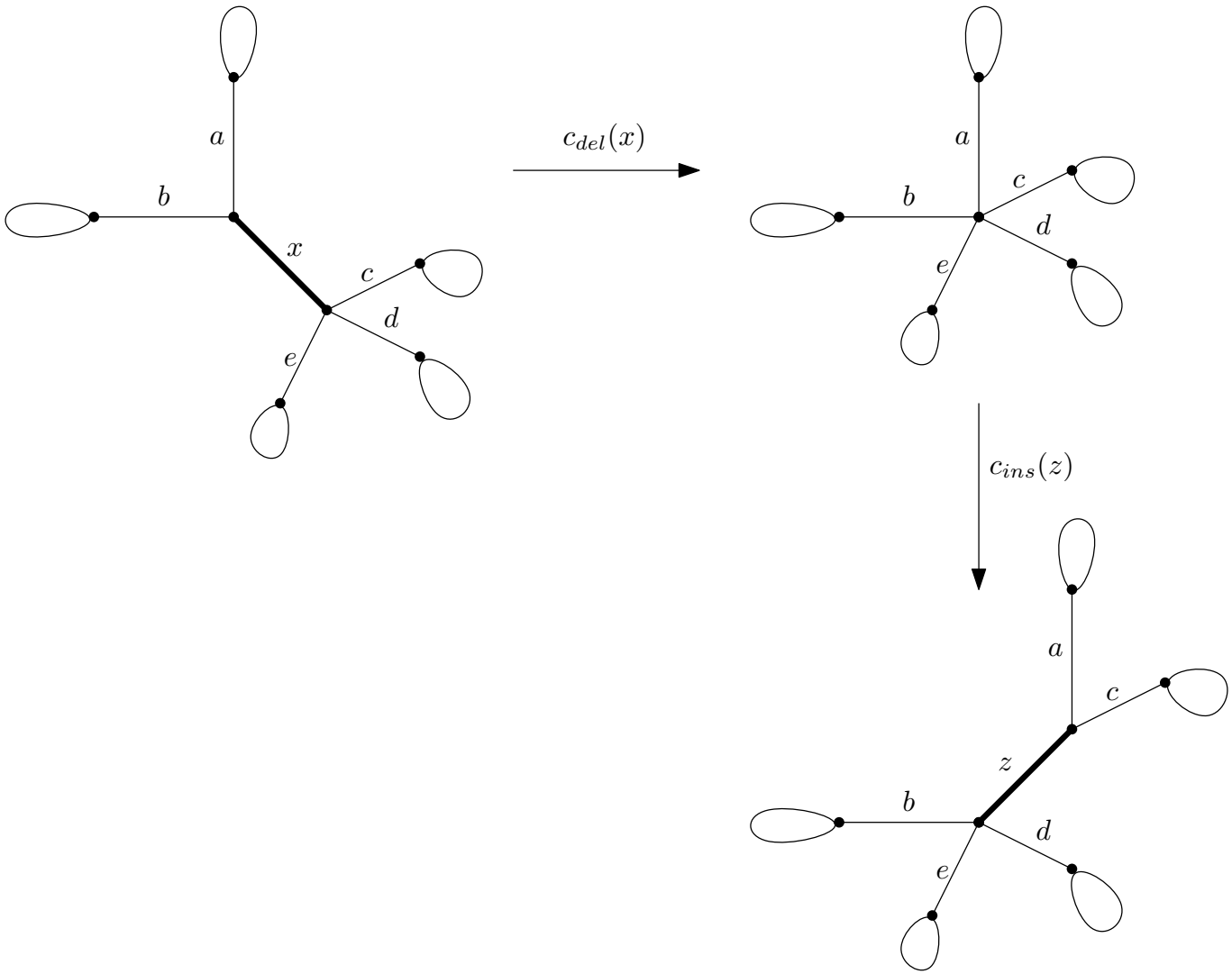
Operations on unrooted trees



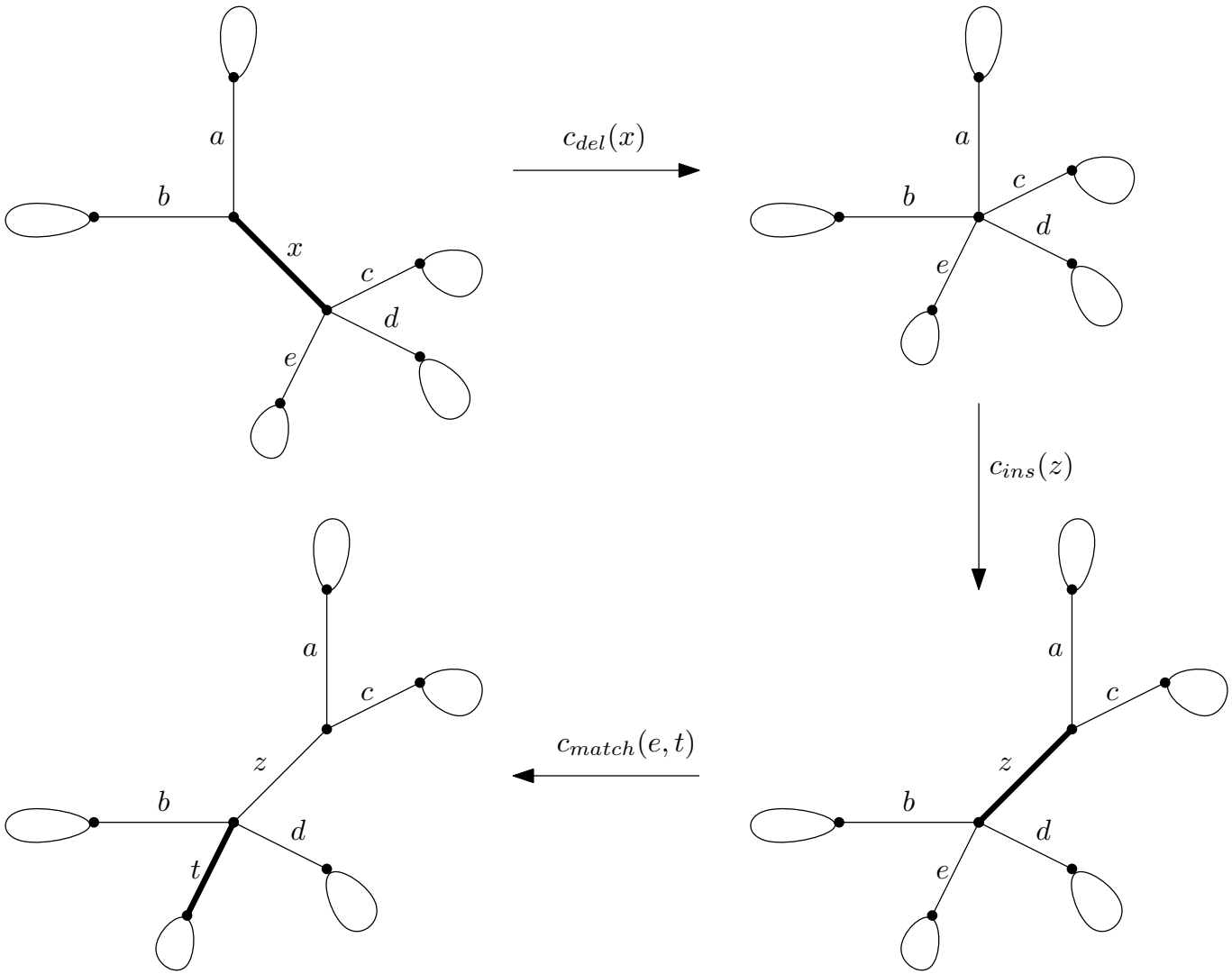
Operations on unrooted trees



Operations on unrooted trees



Operations on unrooted trees

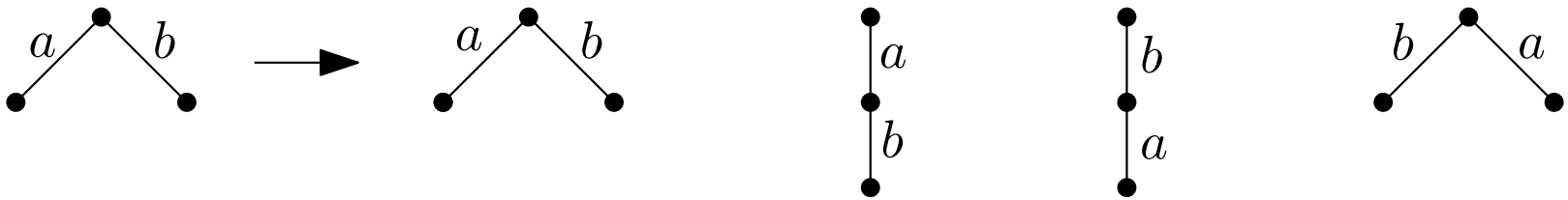


Edit distance between unrooted trees

Definition

$\delta_{unrooted}(T_1, T_2) = \min\{\delta_{rooted}(R_1, R_2) : \text{rooting } R_i \text{ of } T_i\}$

1 tree: 4 rootings:



Lemma

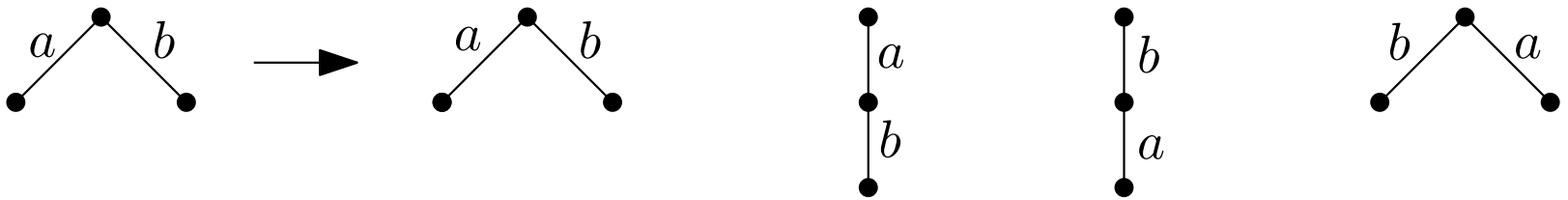
It suffices to root arbitrarily first of the trees and check all rootings of the second.

Edit distance between unrooted trees

Definition

$\delta_{unrooted}(T_1, T_2) = \min\{\delta_{rooted}(R_1, R_2) : \text{rooting } R_i \text{ of } T_i\}$

1 tree: 4 rootings:



Lemma

It suffices to root arbitrarily first of the trees and check all rootings of the second.

History

Authors	Year	Rooted	Unrooted
Tai	1979	$\mathcal{O}(n^6)$	no
Zhang, Shasha	1990	$\mathcal{O}(n^4)$	yes
Klein	1998	$\mathcal{O}(n^3 \log n)$	yes
Demaine et al.	2009	$\mathcal{O}(n^3)$	no
this paper	2018	$\mathcal{O}(n^3)$	yes

Bringmann et al. [SODA '18], conjecture

For any $\varepsilon > 0$, Tree Edit Distance on two n-node trees cannot be computed in $\mathcal{O}(n^{3-\varepsilon})$ time.

History

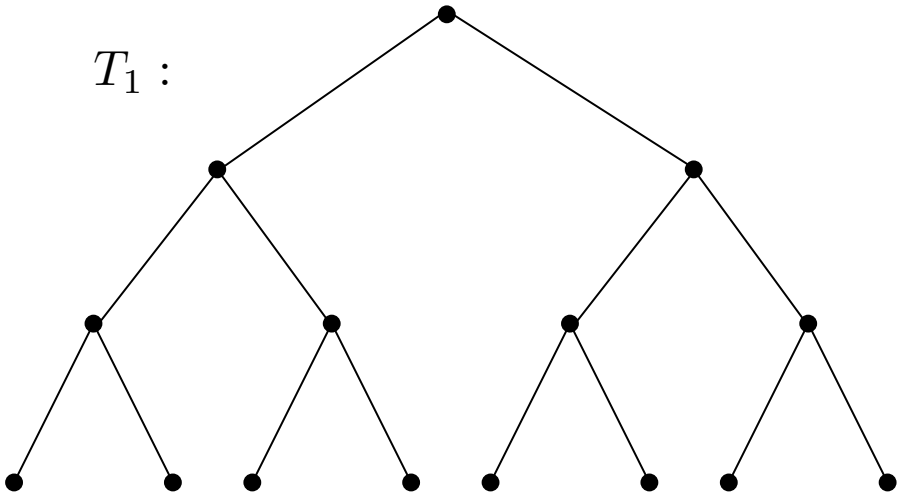
Authors	Year	Rooted	Unrooted
Tai	1979	$\mathcal{O}(n^6)$	no
Zhang, Shasha	1990	$\mathcal{O}(n^4)$	yes
Klein	1998	$\mathcal{O}(n^3 \log n)$	yes
Demaine et al.	2009	$\mathcal{O}(n^3)$	no
this paper	2018	$\mathcal{O}(n^3)$	yes

Bringmann et al. [SODA '18], conjecture

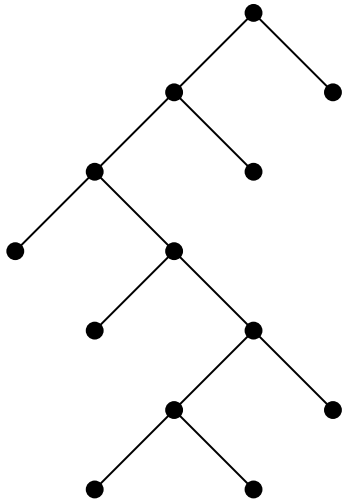
For any $\varepsilon > 0$, Tree Edit Distance on two n-node trees cannot be computed in $\mathcal{O}(n^{3-\varepsilon})$ time.

Return to two **rooted** trees

T_1 :



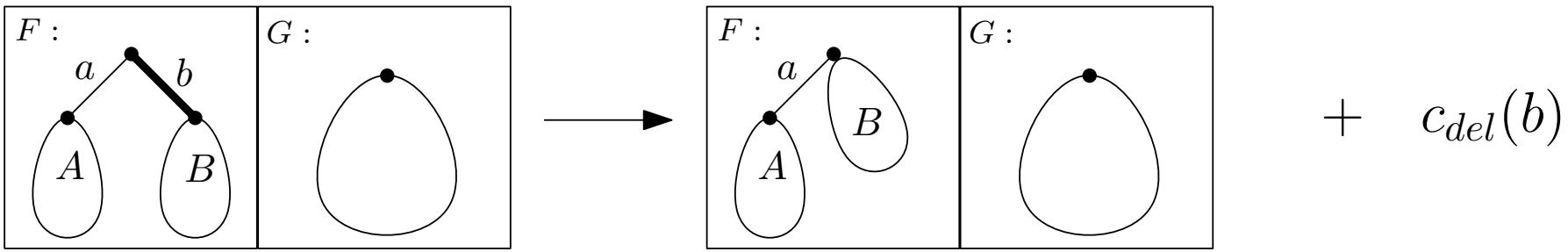
T_2 :



Recursive formula for $\delta(F, G)$:

- $\delta(\emptyset, \emptyset) = 0$
- $\delta(F, G) =$

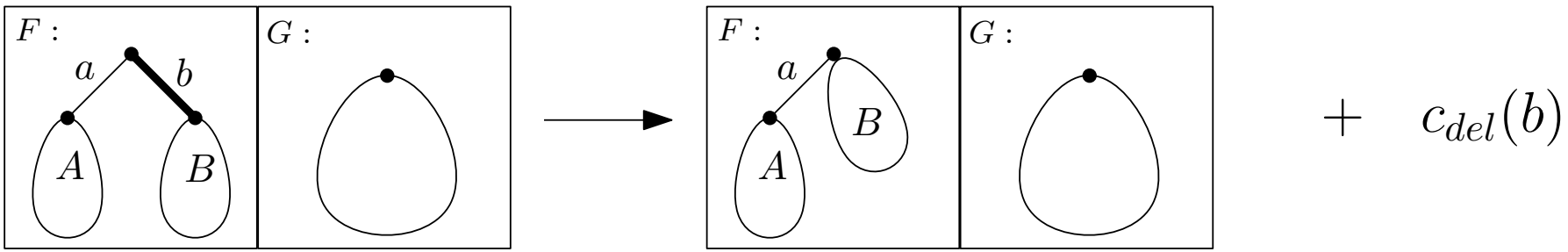
$$\min \begin{cases} \delta(F - r_F, G) + c_{del}(r_F) & \text{if } F \neq \emptyset \\ \delta(F, G - r_G) + c_{del}(r_G) & \text{if } G \neq \emptyset \\ \delta(R_F, R_G) + \delta(F - R_F, G - R_G) + c_{match}(r_F, r_G) & \text{if } F, G \neq \emptyset \end{cases}$$



Recursive formula for $\delta(F, G)$:

- $\delta(\emptyset, \emptyset) = 0$
- $\delta(F, G) =$

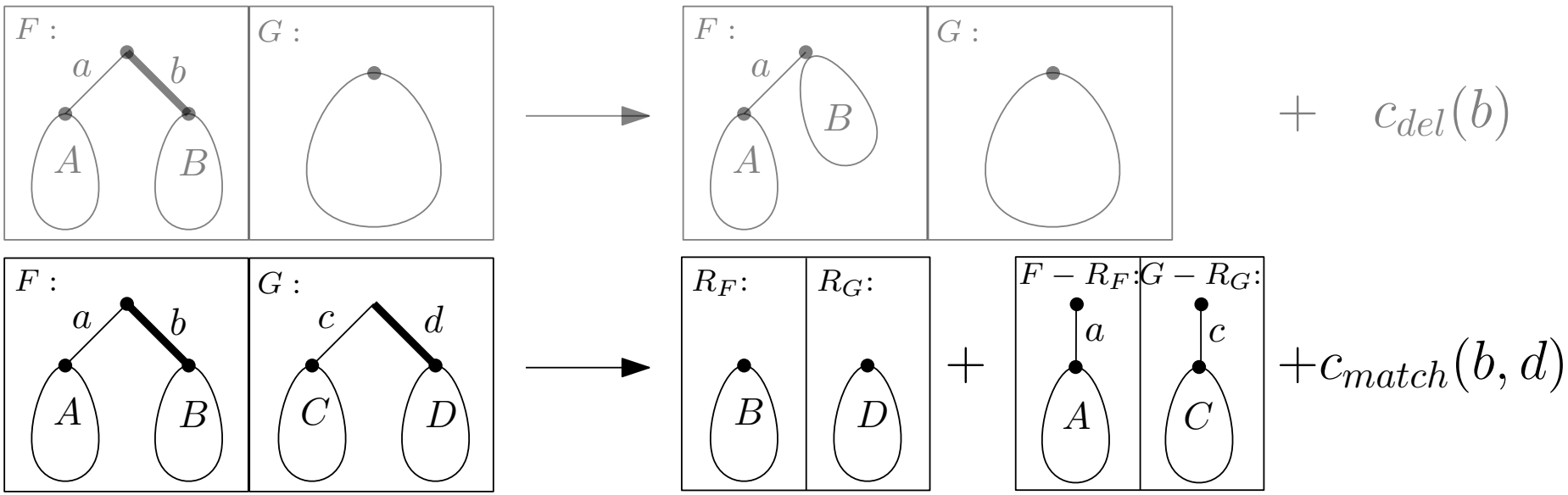
$$\min \begin{cases} \delta(F - r_F, G) + c_{del}(r_F) & \text{if } F \neq \emptyset \\ \delta(F, G - r_G) + c_{del}(r_G) & \text{if } G \neq \emptyset \\ \delta(R_F, R_G) + \delta(F - R_F, G - R_G) + c_{match}(r_F, r_G) & \text{if } F, G \neq \emptyset \end{cases}$$



Recursive formula for $\delta(F, G)$:

- $\delta(\emptyset, \emptyset) = 0$
- $\delta(F, G) =$

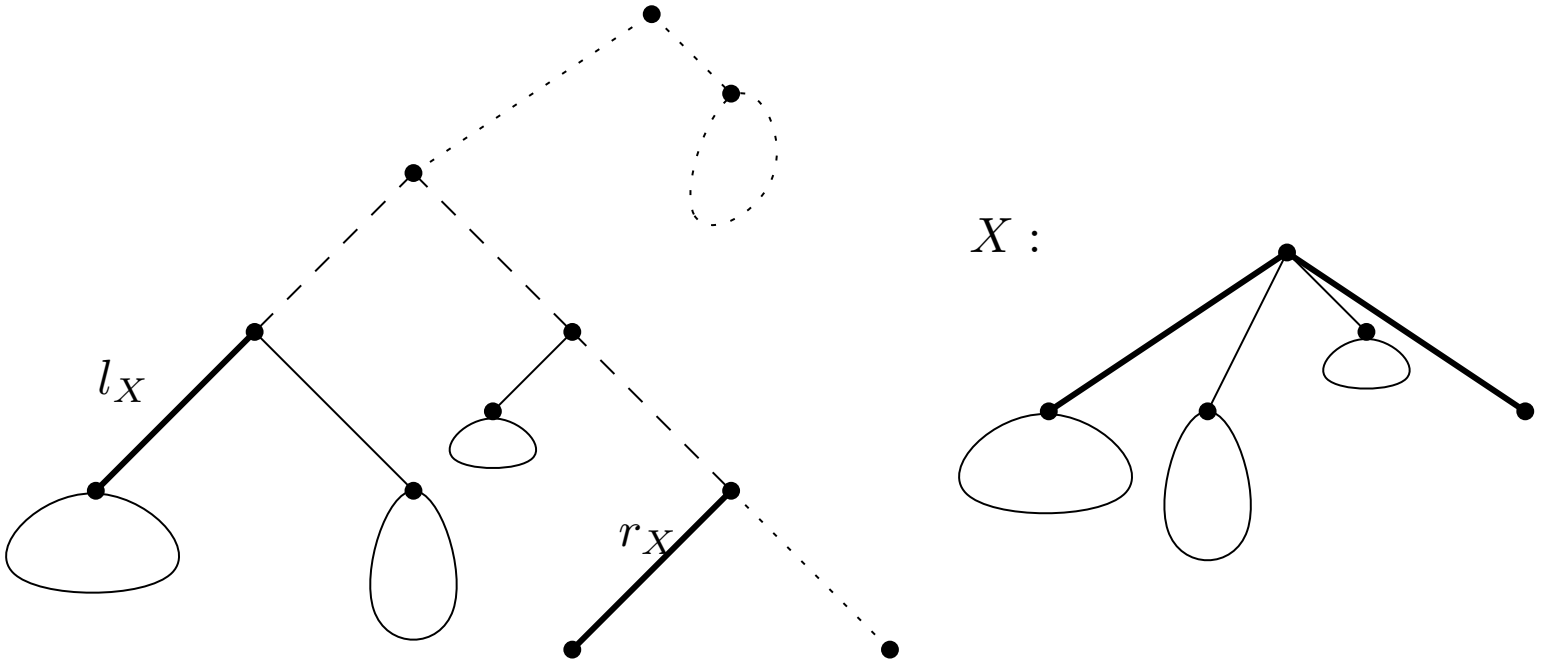
$$\min \begin{cases} \delta(F - r_F, G) + c_{del}(r_F) & \text{if } F \neq \emptyset \\ \delta(F, G - r_G) + c_{del}(r_G) & \text{if } G \neq \emptyset \\ \delta(R_F, R_G) + \delta(F - R_F, G - R_G) + c_{match}(r_F, r_G) & \text{if } F, G \neq \emptyset \end{cases}$$



Analysis of Zhang & Shasha's algorithm

Tree representation

Every obtained tree F can be uniquely represented by the pair (l_F, r_F).



Corollary

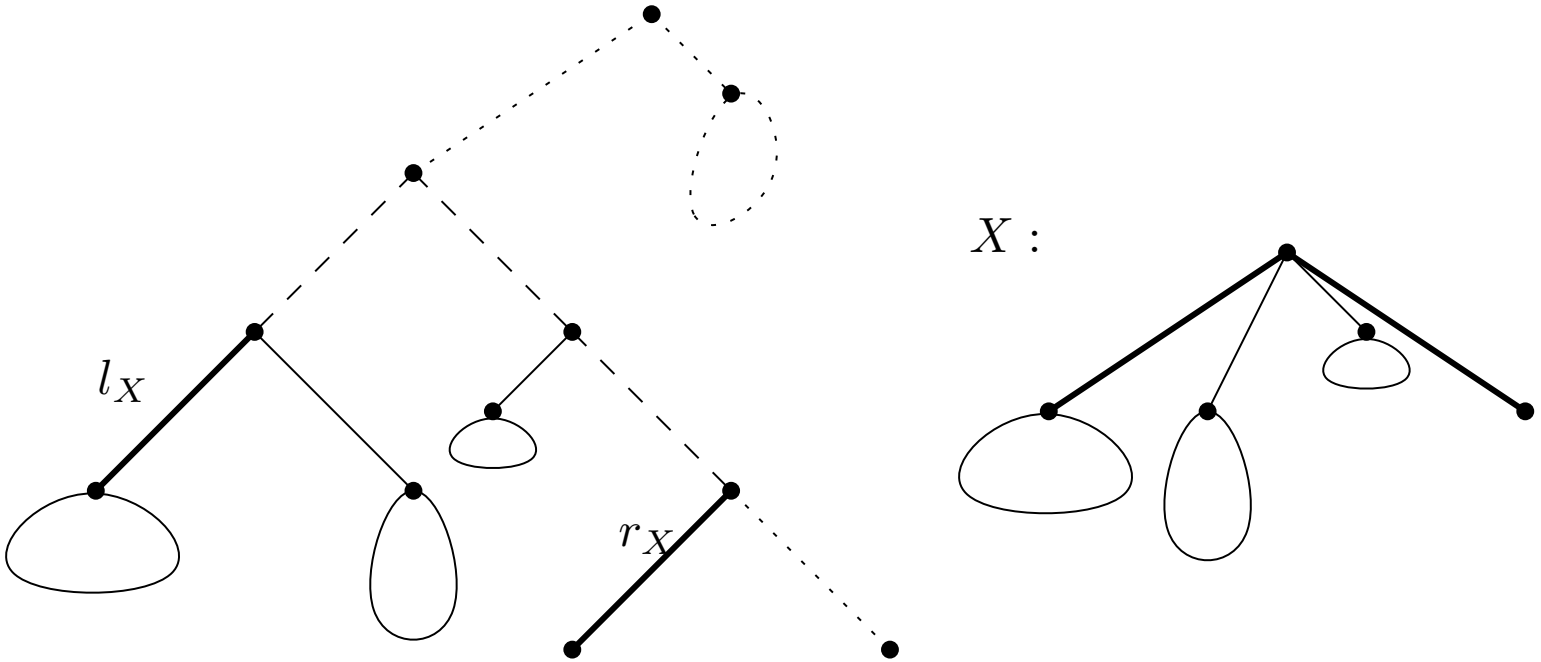
There are $\mathcal{O}(n^2)$ subtrees of a tree that can be obtained.

Zhang & Shasha's algorithm: $\mathcal{O}(n^4)$.

Analysis of Zhang & Shasha's algorithm

Tree representation

Every obtained tree F can be uniquely represented by the pair (l_F, r_F).



Corollary

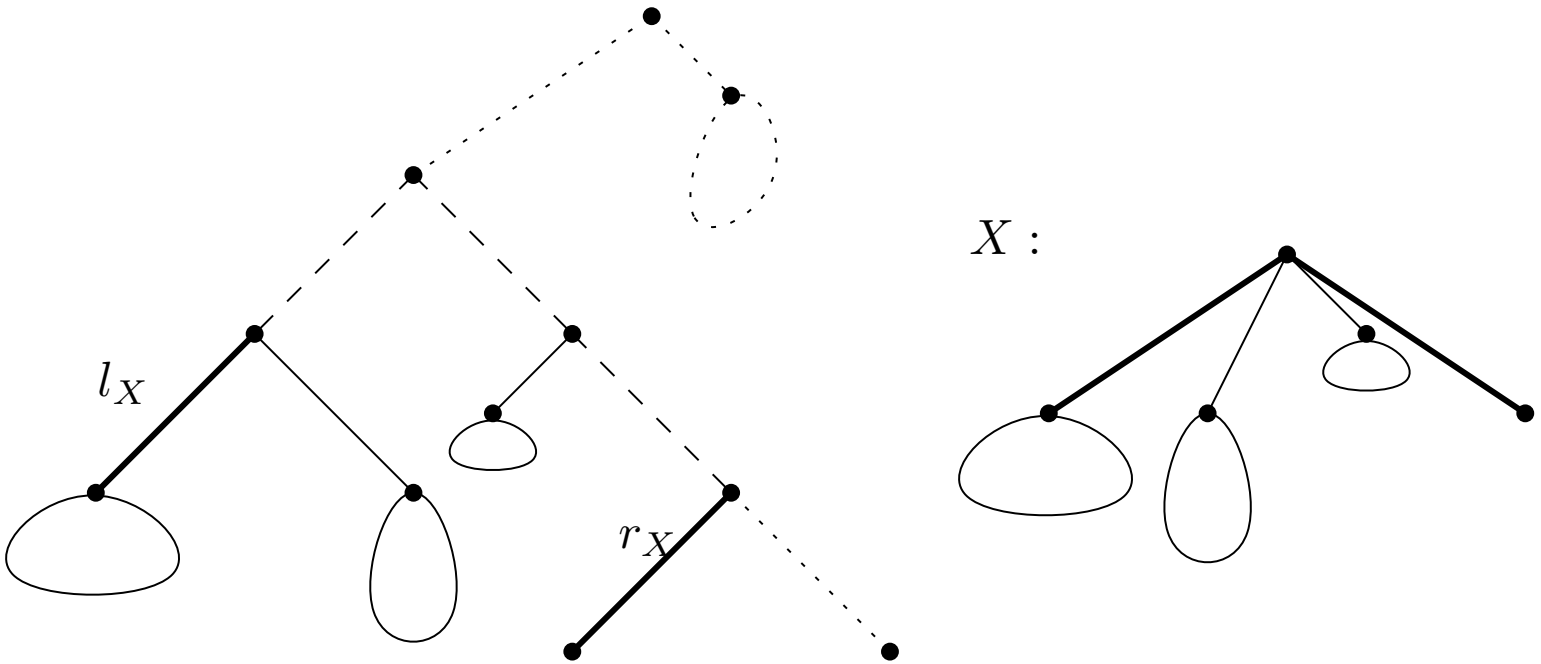
There are $\mathcal{O}(n^2)$ subtrees of a tree that can be obtained.

Zhang & Shasha's algorithm: $\mathcal{O}(n^4)$.

Analysis of Zhang & Shasha's algorithm

Tree representation

Every obtained tree F can be uniquely represented by the pair (l_F, r_F).



Corollary

There are $\mathcal{O}(n^2)$ subtrees of a tree that can be obtained.

Zhang & Shasha's algorithm: $\mathcal{O}(n^4)$.

How to improve Zhang & Shasha

Recall the DP:

$$\delta(F, G) = \min \begin{cases} \delta(F - r_F, G) + c_{del}(r_F) & \text{if } F \neq \emptyset \\ \delta(F, G - r_G) + c_{del}(r_G) & \text{if } G \neq \emptyset \\ \delta(R_F, R_G) + \delta(F - R_F, G - R_G) + c_{match}(r_F, r_G) & \text{if } F, G \neq \emptyset \end{cases}$$

Don't choose always the right side!

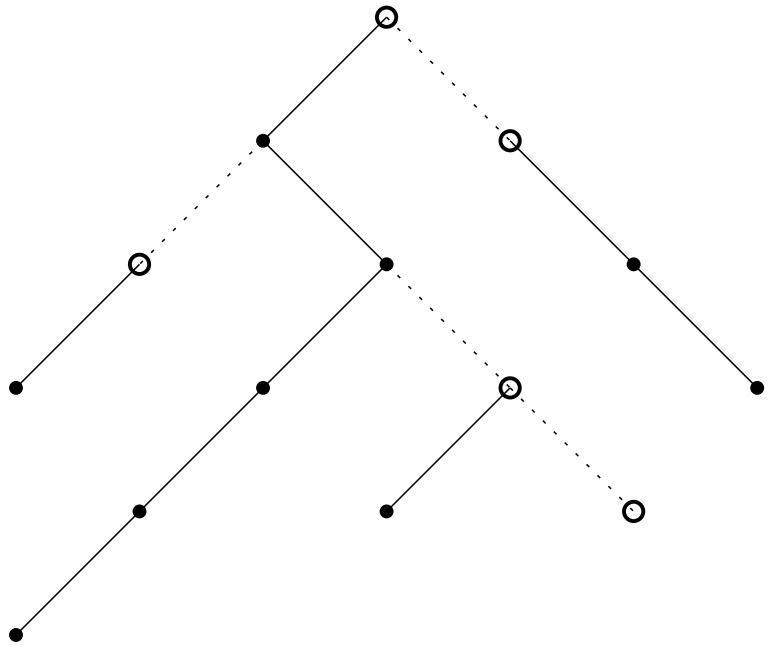
How to improve Zhang & Shasha

Recall the DP:

$$\delta(F, G) = \min \begin{cases} \delta(F - r_F, G) + c_{del}(r_F) & \text{if } F \neq \emptyset \\ \delta(F, G - r_G) + c_{del}(r_G) & \text{if } G \neq \emptyset \\ \delta(R_F, R_G) + \delta(F - R_F, G - R_G) + c_{match}(r_F, r_G) & \text{if } F, G \neq \emptyset \end{cases}$$

Don't choose always the right side!

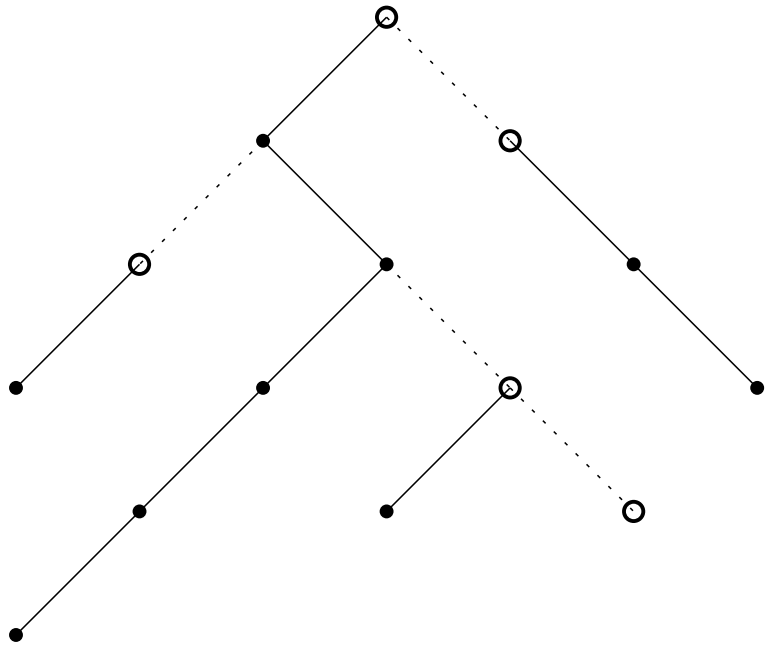
Use heavy-light decomposition



Avoid heavy child:

- (Klein) – always in T_1 : $\mathcal{O}(n^3 \log n)$,
- (as in this paper) – first in T_1, then in T_2 : $\mathcal{O}(n^3 \log \log n)$,
- (Demaine et al.) $\approx$ in the bigger of F and G: $\mathcal{O}(n^3)$.

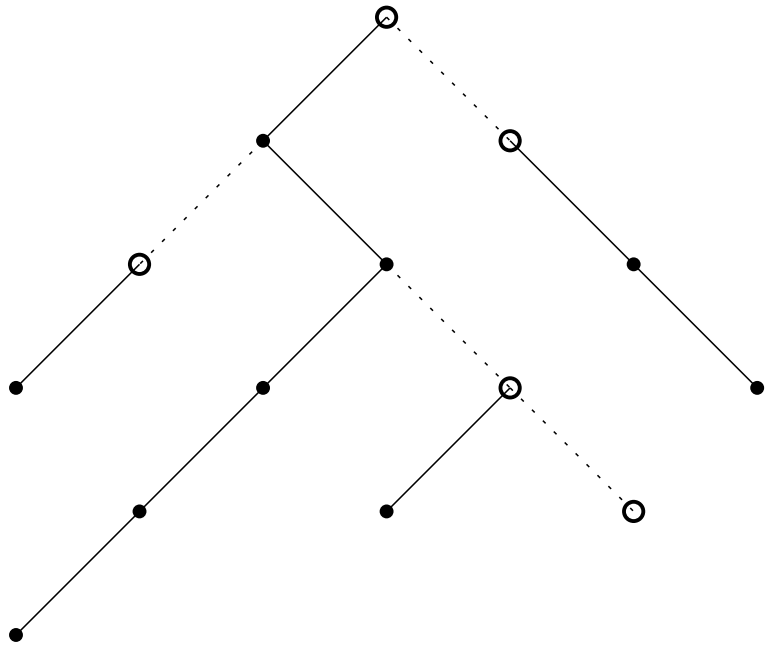
Use heavy-light decomposition



Avoid heavy child:

- (Klein) – always in T_1 : $\mathcal{O}(n^3 \log n)$,
- (as in this paper) – first in T_1, then in T_2 : $\mathcal{O}(n^3 \log \log n)$,
- (Demaine et al.) $\approx$ in the bigger of F and G: $\mathcal{O}(n^3)$.

Use heavy-light decomposition

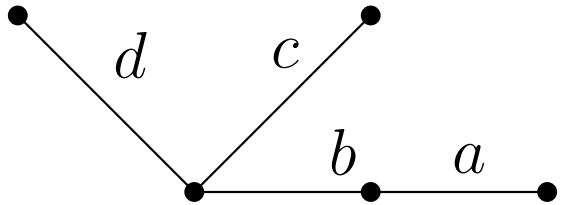


Avoid heavy child:

- (Klein) – always in T_1 : $\mathcal{O}(n^3 \log n)$,
- (as in this paper) – first in T_1, then in T_2 : $\mathcal{O}(n^3 \log \log n)$,
- (Demaine et al.) $\approx$ in the bigger of F and G: $\mathcal{O}(n^3)$.

Why the unrooted case is more difficult?

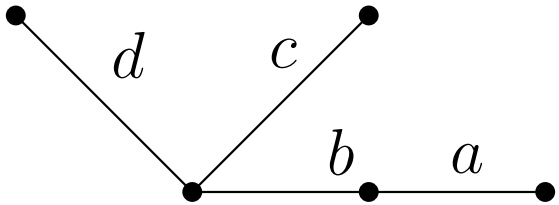
One tree:



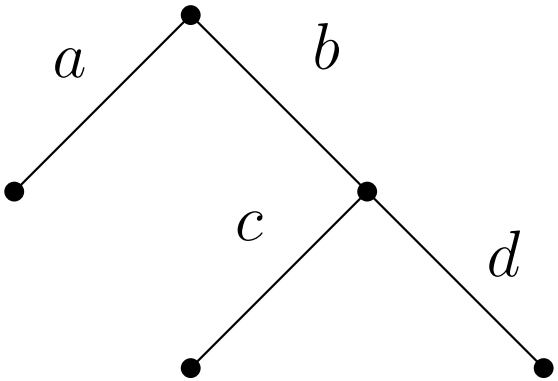
in different rootings:

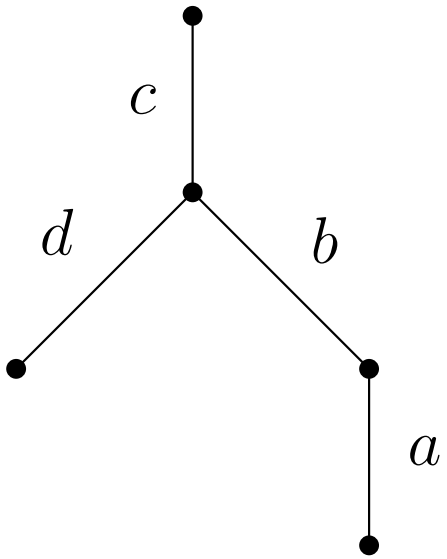
Why the unrooted case is more difficult?

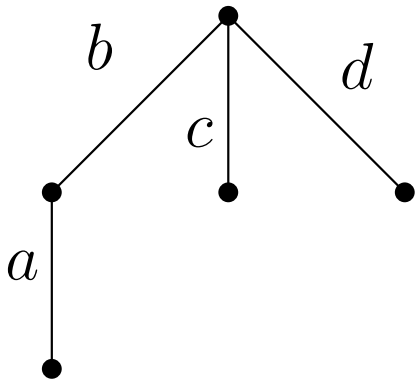
One tree:



in different rootings:

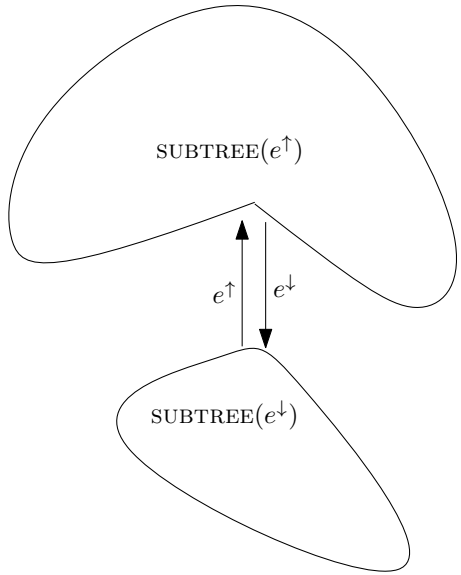






How to approach the unrooted case

Root both trees and replace each edge with two darts:

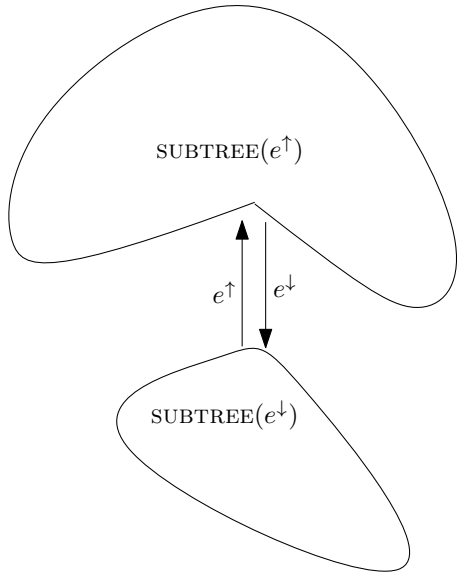


Need a strategy for subtrees of darts in T_2:

- down the tree (use algorithm of Demaine et al.)
- up the tree along a heavy edge (avoid parent)
- up the tree along a light edge (?)

How to approach the unrooted case

Root both trees and replace each edge with two darts:

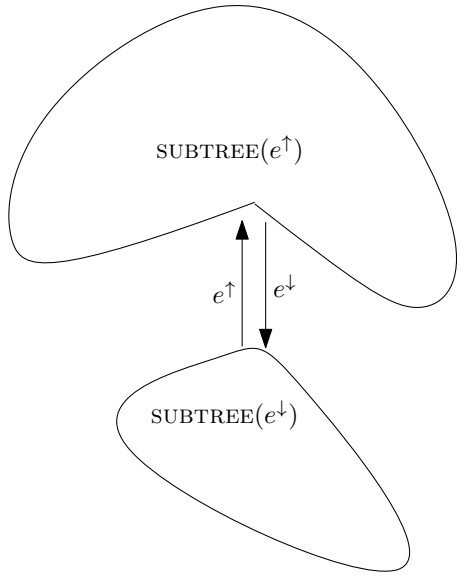


Need a strategy for subtrees of darts in T_2:

- down the tree (use algorithm of Demaine et al.)
- up the tree along a heavy edge (avoid parent)
- up the tree along a light edge (?)

How to approach the unrooted case

Root both trees and replace each edge with two darts:

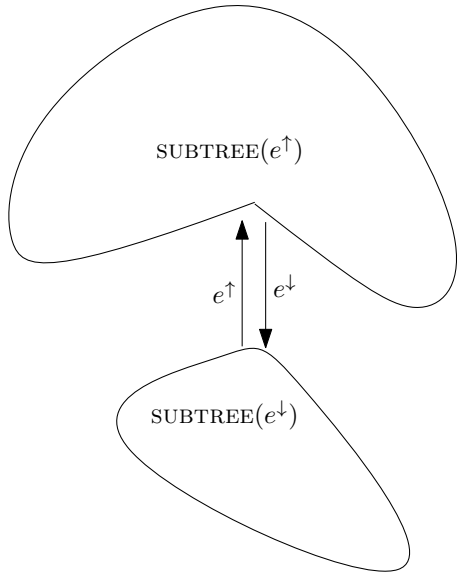


Need a strategy for subtrees of darts in T_2:

- down the tree (use algorithm of Demaine et al.)
- up the tree along a heavy edge (avoid parent)
- up the tree along a light edge (?)

How to approach the unrooted case

Root both trees and replace each edge with two darts:

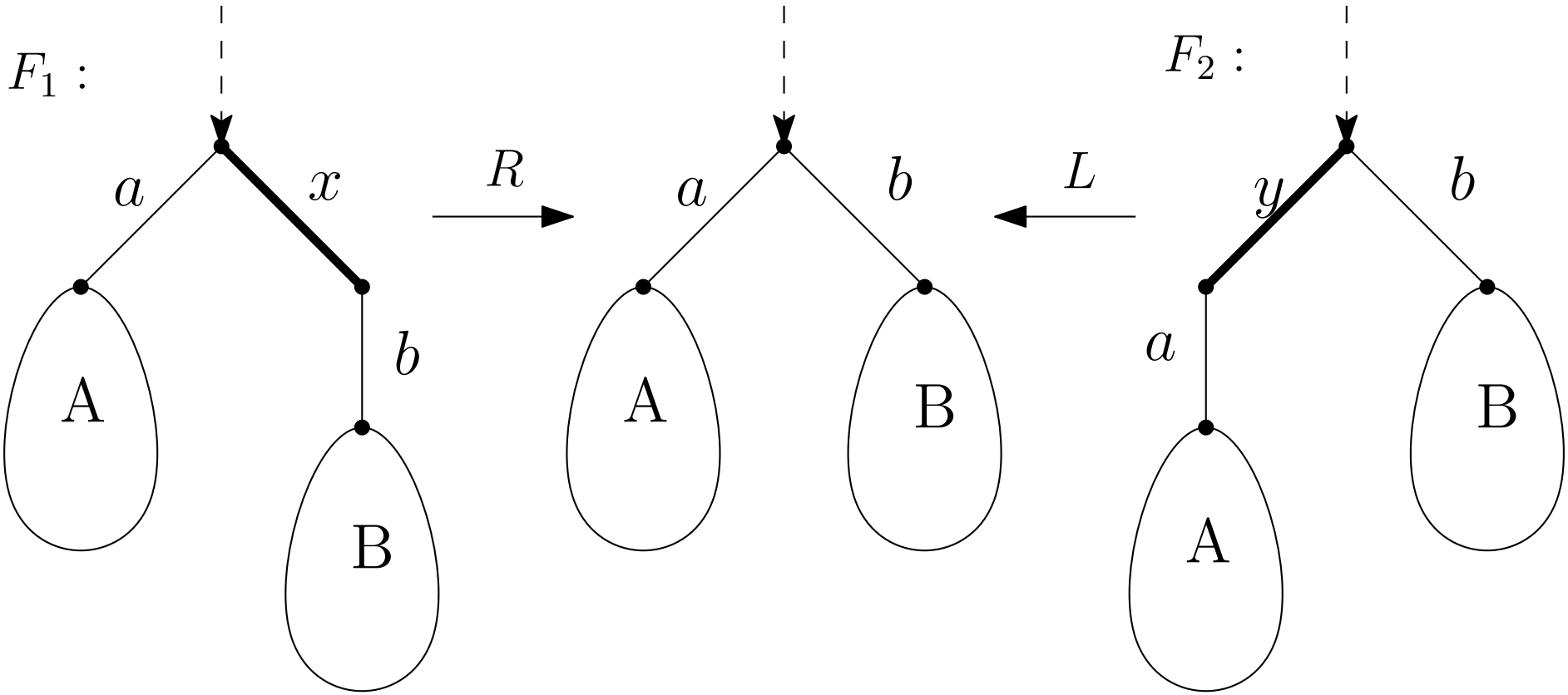


Need a strategy for subtrees of darts in T_2:

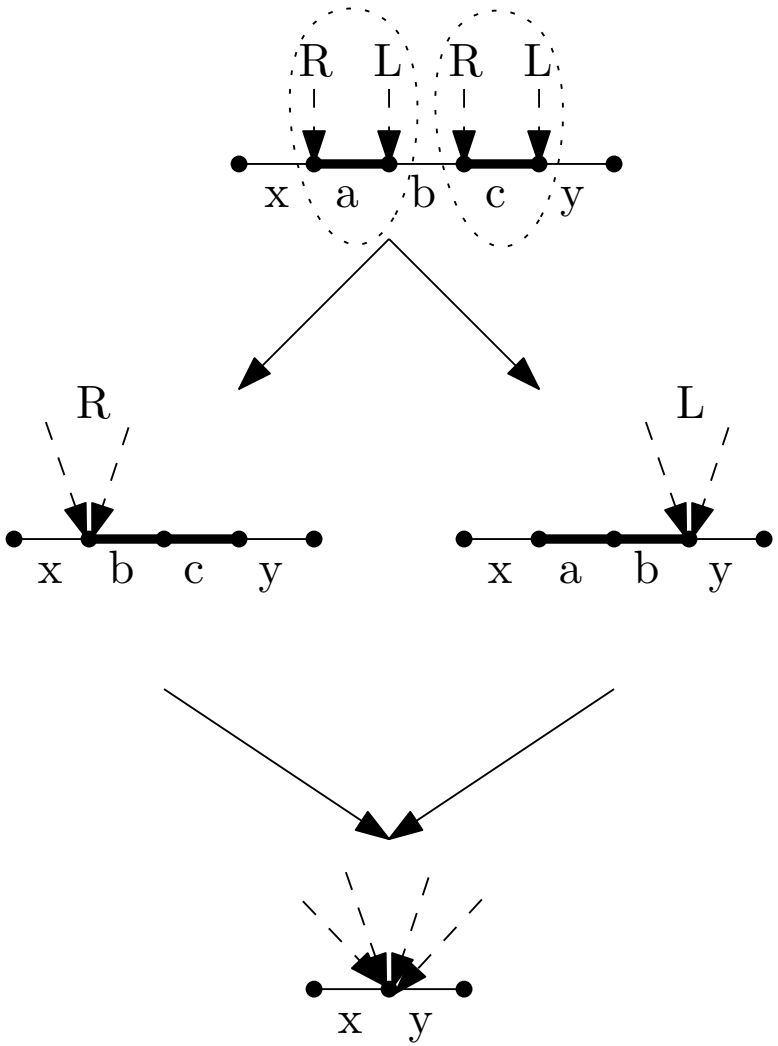
- down the tree (use algorithm of Demaine et al.)
- up the tree along a heavy edge (avoid parent)
- up the tree along a light edge (?)

Main idea

Try to reduce subproblems for two different trees to one common tree:



Global Divide & Conquer strategy



$\Longrightarrow$ $\mathcal{O}(n^3 \log\log n)$ algorithm

What else do we need to get $\mathcal{O}(n^3)$ time

1. some nodes are more important than the other
2. analyze and make use of the recurrence:

$$t(m) \leq m^2 \left(1 + \ln \frac{n}{m}\right) + \sum_i t(m_i) \quad \text{for } m_i \leq \frac{m}{2}, \quad \sum_i m_i \leq m,$$

3. conquer dividing not always evenly

Theorem (full version of this paper)

Edit distance between unrooted trees of size n and m, where $m \leq n$, can be computed in $\mathcal{O}(nm^2(1 + \log \frac{n}{m})) = \mathcal{O}(n^3)$ time.

What else do we need to get $\mathcal{O}(n^3)$ time

1. some nodes are more important than the other
2. analyze and make use of the recurrence:

$$t(m) \leq m^2 \left(1 + \ln \frac{n}{m}\right) + \sum_i t(m_i) \quad \text{for } m_i \leq \frac{m}{2}, \quad \sum_i m_i \leq m,$$

3. conquer dividing not always evenly

Theorem (full version of this paper)

Edit distance between unrooted trees of size n and m, where $m \leq n$, can be computed in $\mathcal{O}(nm^2(1 + \log \frac{n}{m})) = \mathcal{O}(n^3)$ time.

What else do we need to get $\mathcal{O}(n^3)$ time

1. some nodes are more important than the other
2. analyze and make use of the recurrence:

$$t(m) \leq m^2 \left(1 + \ln \frac{n}{m}\right) + \sum_i t(m_i) \quad \text{for } m_i \leq \frac{m}{2}, \quad \sum_i m_i \leq m,$$

3. conquer dividing not always evenly

Theorem (full version of this paper)

Edit distance between unrooted trees of size n and m, where $m \leq n$, can be computed in $\mathcal{O}(nm^2(1 + \log \frac{n}{m})) = \mathcal{O}(n^3)$ time.

What else do we need to get $\mathcal{O}(n^3)$ time

1. some nodes are more important than the other
2. analyze and make use of the recurrence:

$$t(m) \leq m^2 \left(1 + \ln \frac{n}{m}\right) + \sum_i t(m_i) \quad \text{for } m_i \leq \frac{m}{2}, \quad \sum_i m_i \leq m,$$

3. conquer dividing not always evenly

Theorem (full version of this paper)

Edit distance between unrooted trees of size n and m, where $m \leq n$, can be computed in $\mathcal{O}(nm^2(1 + \log \frac{n}{m})) = \mathcal{O}(n^3)$ time.

What else do we need to get $\mathcal{O}(n^3)$ time

1. some nodes are more important than the other
2. analyze and make use of the recurrence:

$$t(m) \leq m^2 \left(1 + \ln \frac{n}{m}\right) + \sum_i t(m_i) \quad \text{for } m_i \leq \frac{m}{2}, \quad \sum_i m_i \leq m,$$

3. conquer dividing not always evenly

Theorem (full version of this paper)

Edit distance between unrooted trees of size n and m, where $m \leq n$, can be computed in $\mathcal{O}(nm^2(1 + \log \frac{n}{m})) = \mathcal{O}(n^3)$ time.

Open questions

1. Shave logarithmic factors
2. Approximation
3. ... ?

Thank you!

Open questions

1. Shave logarithmic factors
2. Approximation
3. ... ?

Thank you!

Open questions

1. Shave logarithmic factors
2. Approximation
3. ... ?

Thank you!

Open questions

1. Shave logarithmic factors
2. Approximation
3. ... ?

Thank you!