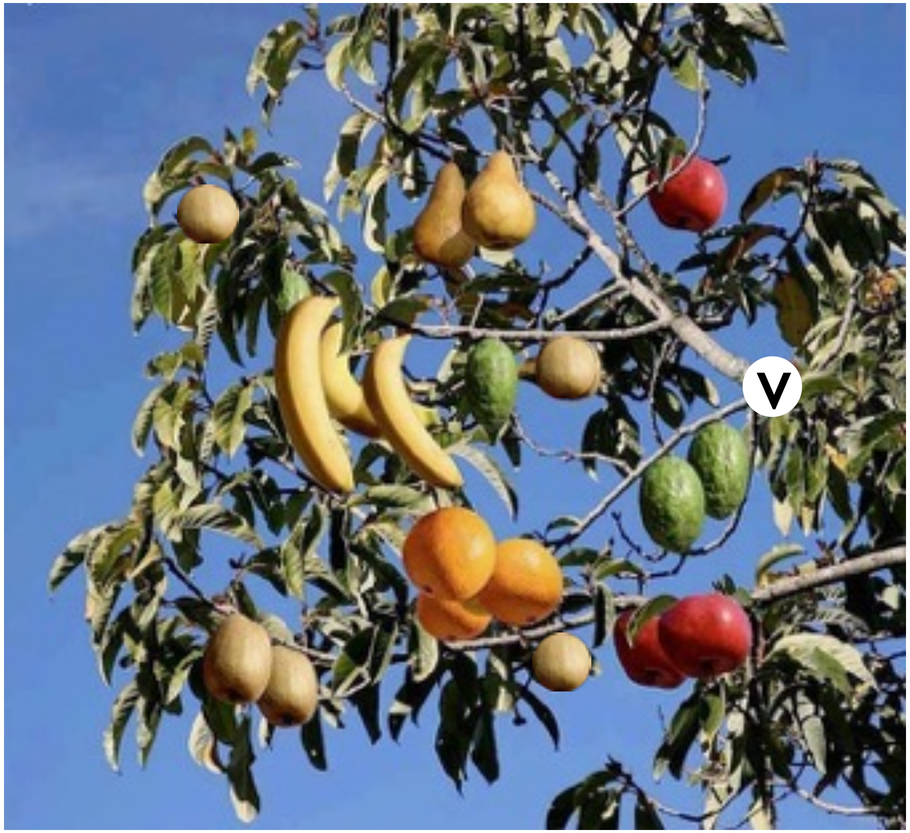
The Nearest Colored Node in a Tree

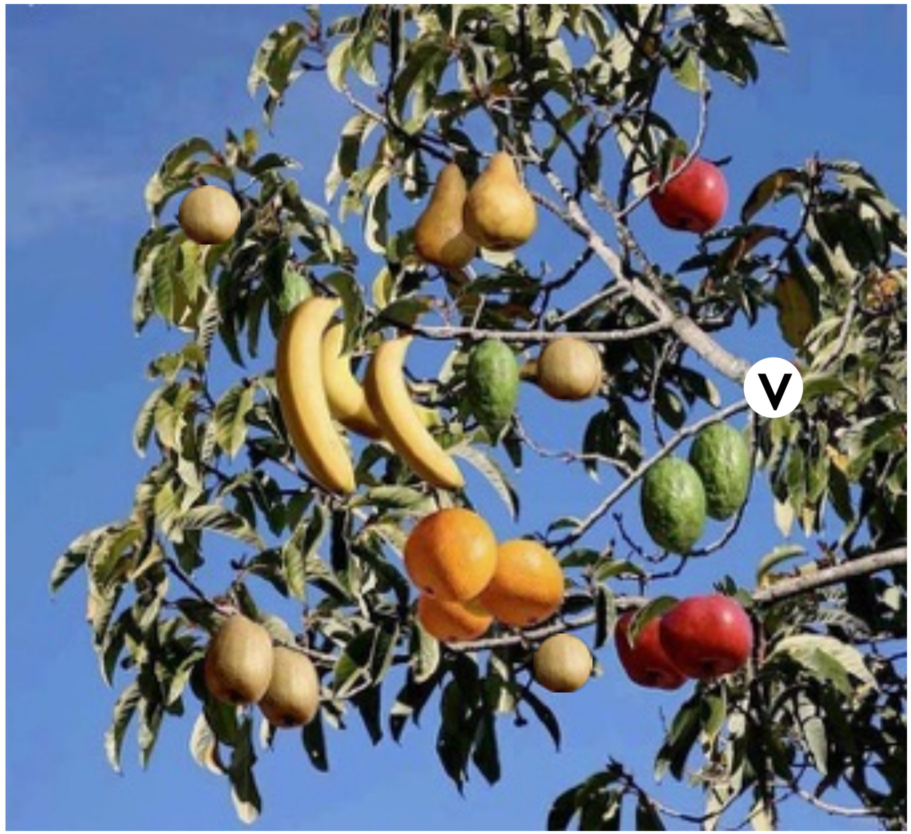
Paweł Gawrychowski, Gad M. Landau, Shay Mozes, **Oren Weimann**

The Nearest Colored Node in a Tree



Paweł Gawrychowski, Gad M. Landau, Shay Mozes, **Oren Weimann**

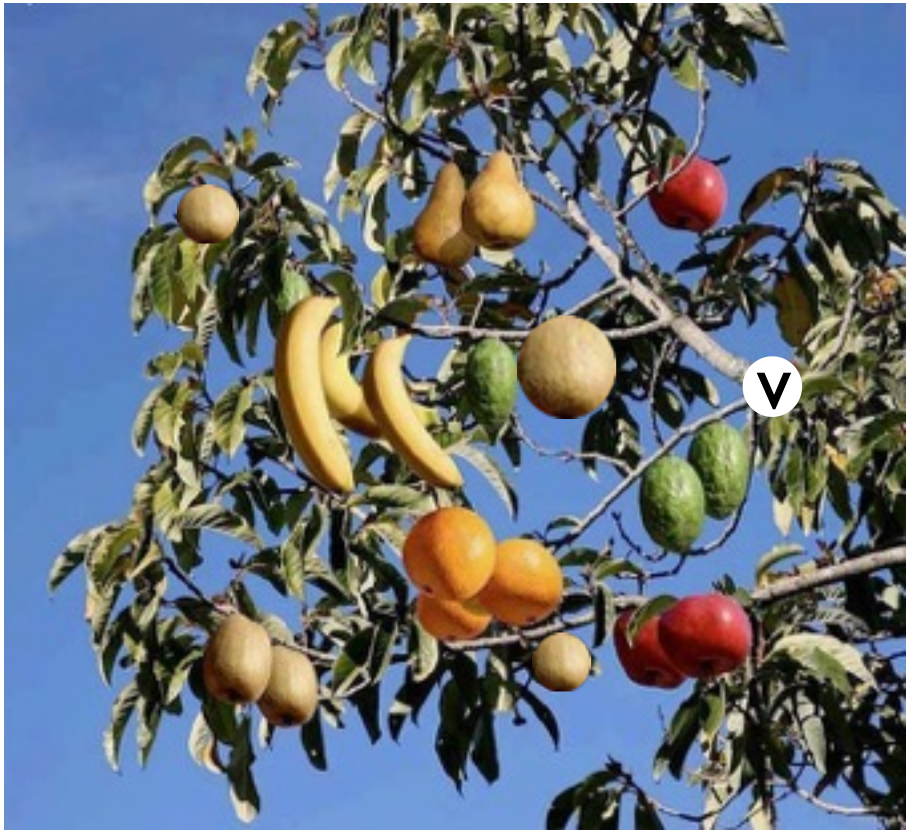
The Nearest Colored Node in a Tree



dist(v, )

Paweł Gawrychowski, Gad M. Landau, Shay Mozes, **Oren Weimann**

The Nearest Colored Node in a Tree

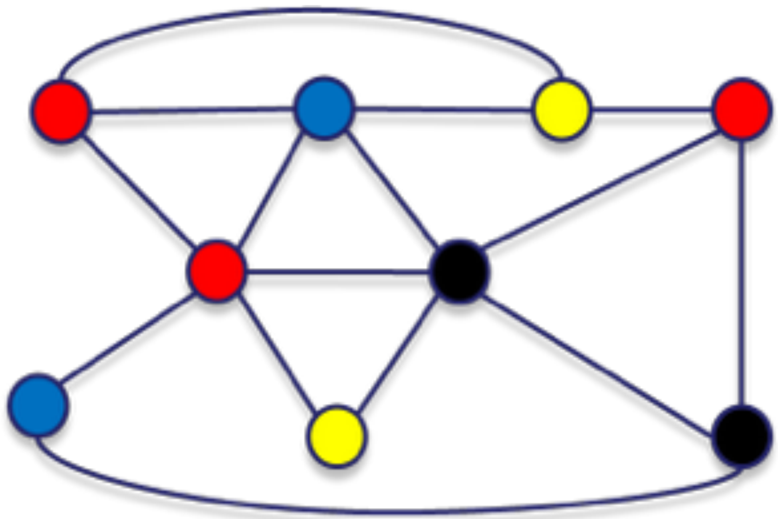


dist(v,)

Paweł Gawrychowski, Gad M. Landau, Shay Mozes, **Oren Weimann**

Vertex-Colored Network

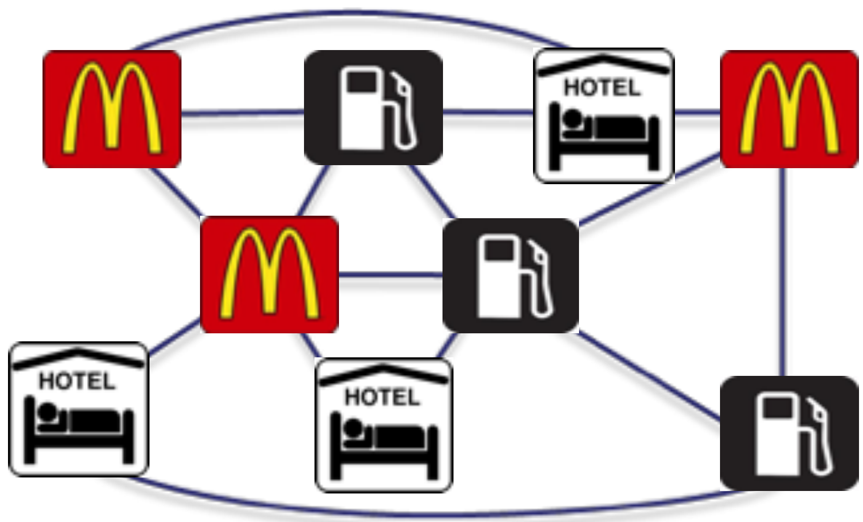
- Colors indicate functionality of a node.



Vertex-Colored Network

- Colors indicate functionality of a node.

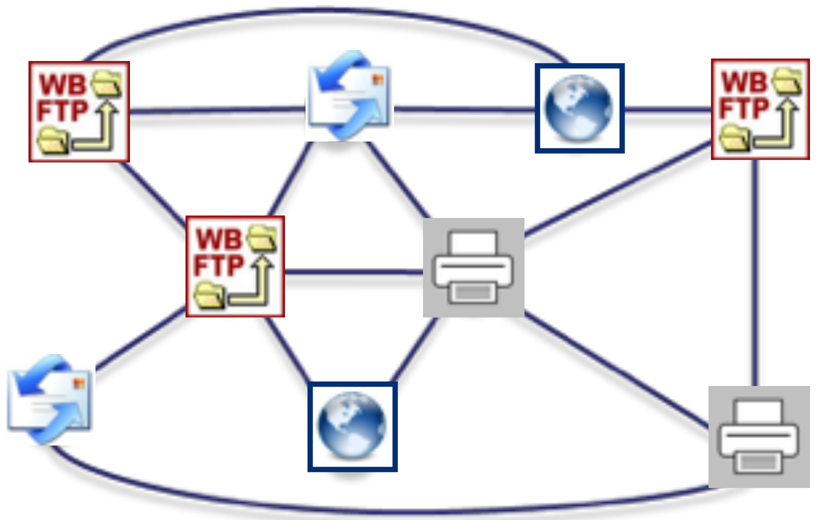
road
network



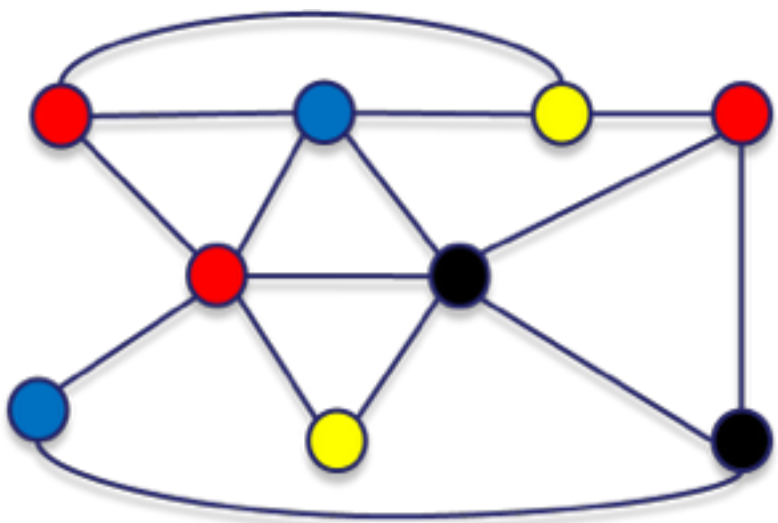
Vertex-Colored Network

- Colors indicate functionality of a node.

computer network



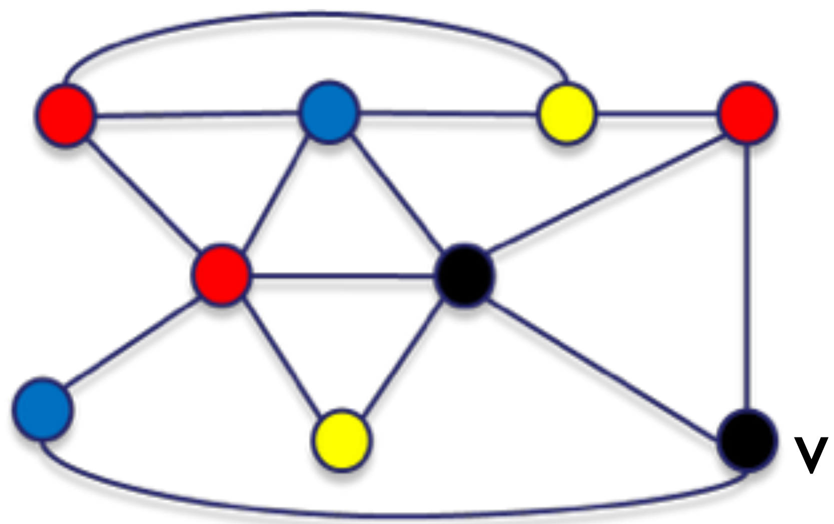
Vertex-Colored Distance Oracle



Vertex-Colored Distance Oracle

- Data Structure for queries:

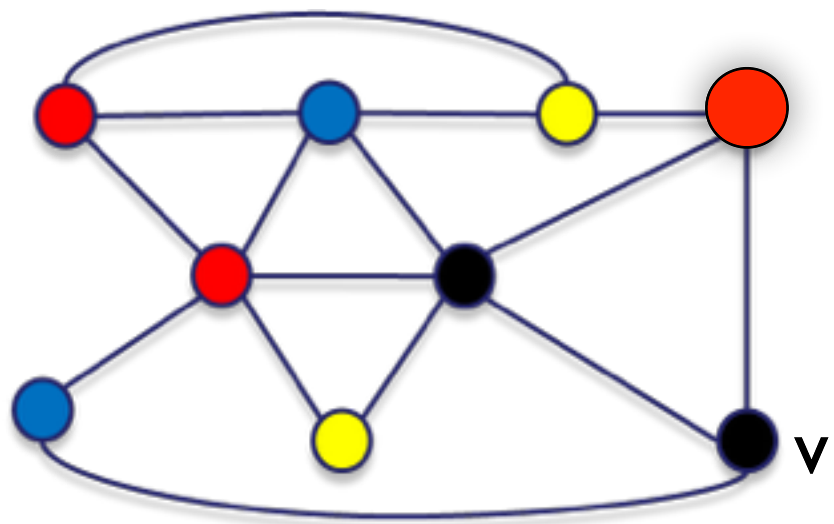
  "what's the closest red node to node v"



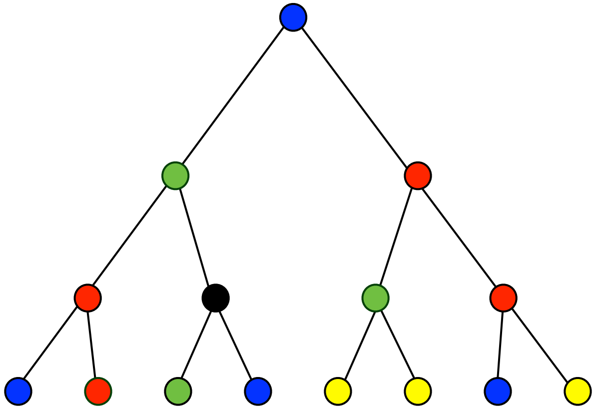
Vertex-Colored Distance Oracle

- Data Structure for queries:

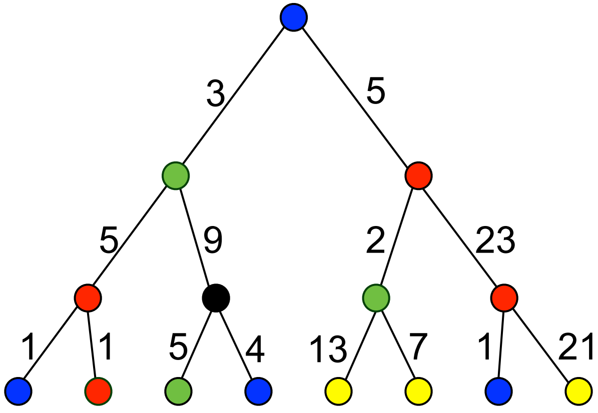
  “what’s the closest red node to node v”



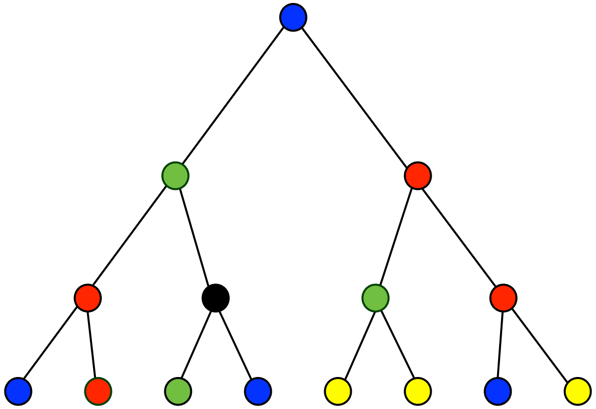
Trees



Trees

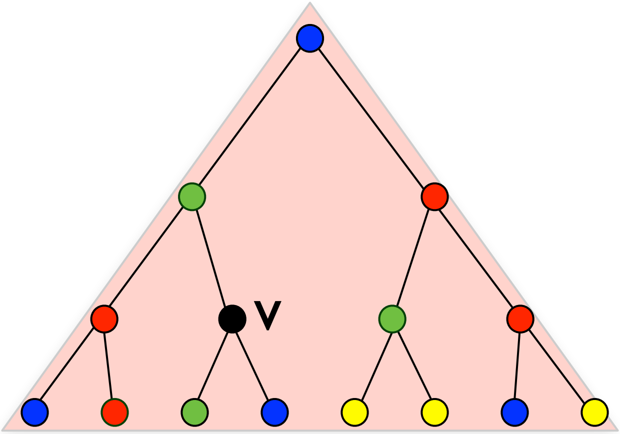


Trees



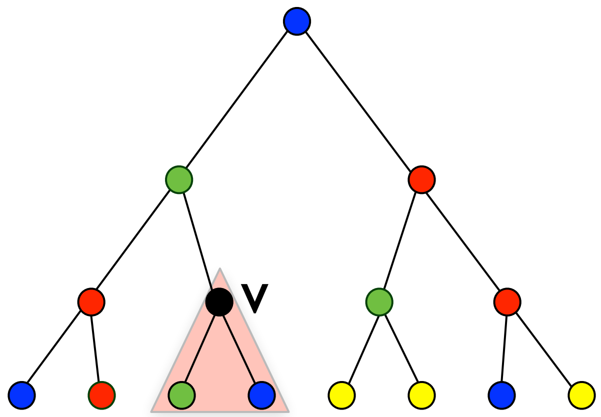
Trees

- Nearest colored node



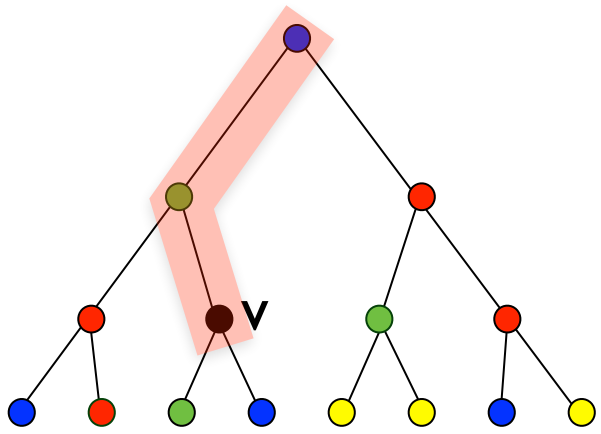
Trees

- Nearest colored descendant
- Nearest colored node



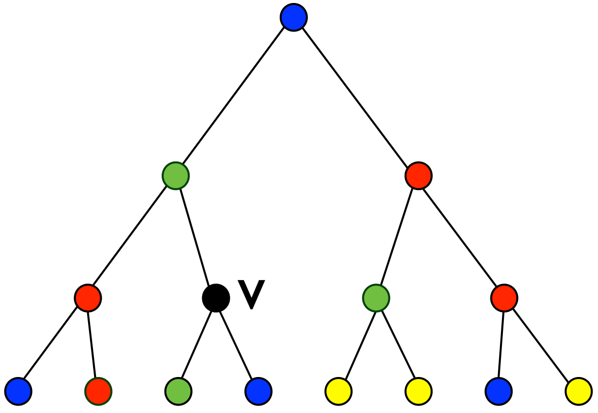
Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



Static Trees

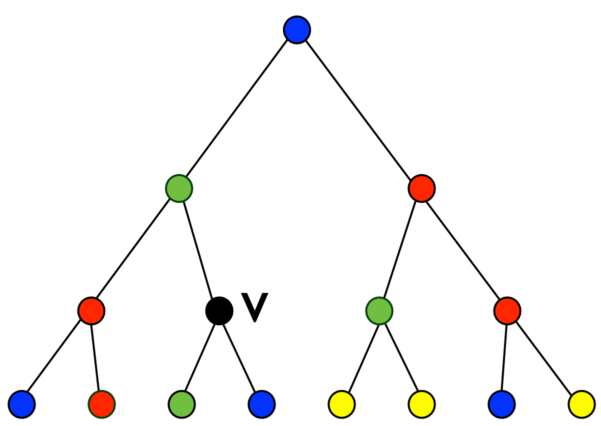
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

If number of colors is O(logn) there is an optimal O(n)-space O(1)-query solution
[Bille, Landau, Raman, Rao, Sadakane, W. 2011]

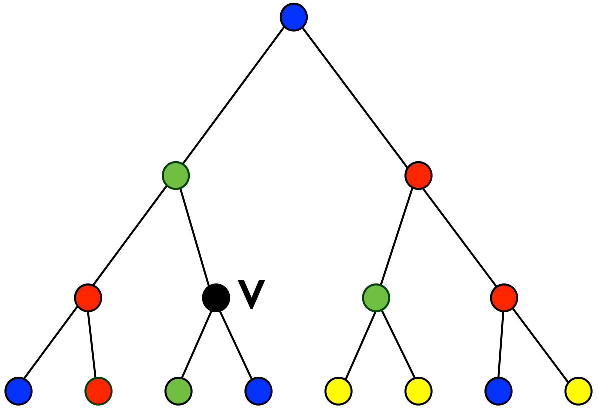


Static Trees

Implicit in [Belazzougui, Navarro 2010]:

Ω(log log n)-query for any
O(n polylog n)-space solution

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



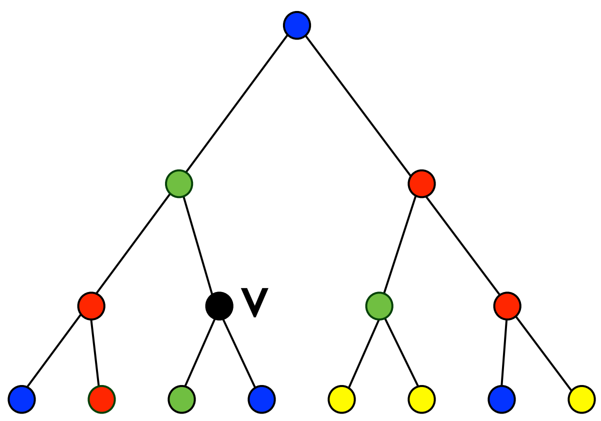
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Implicit in [Belazzougui, Navarro 2010]:

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

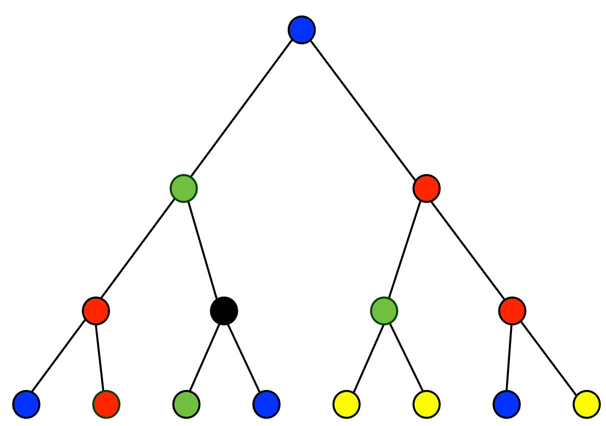


Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

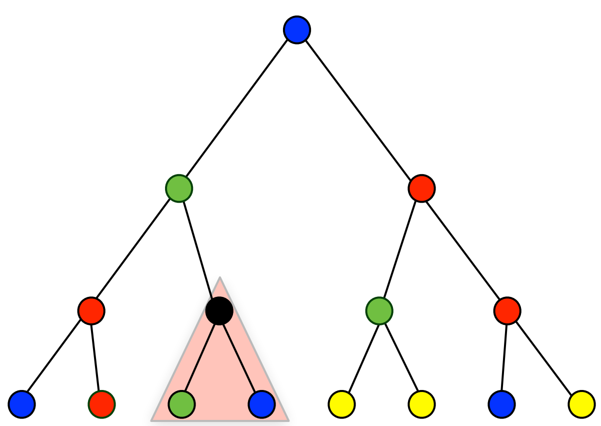


Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution



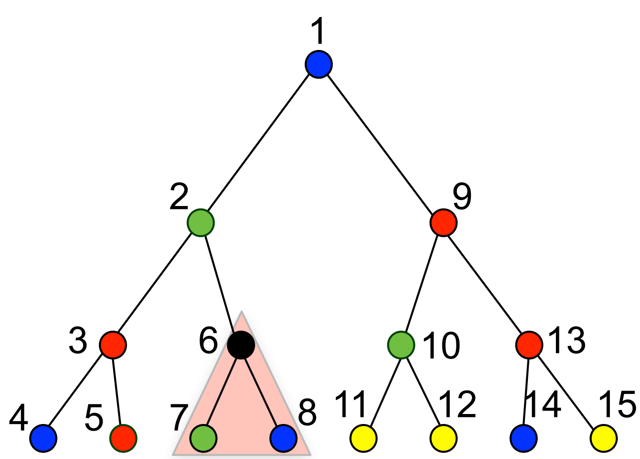
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. Assign preorder numbers



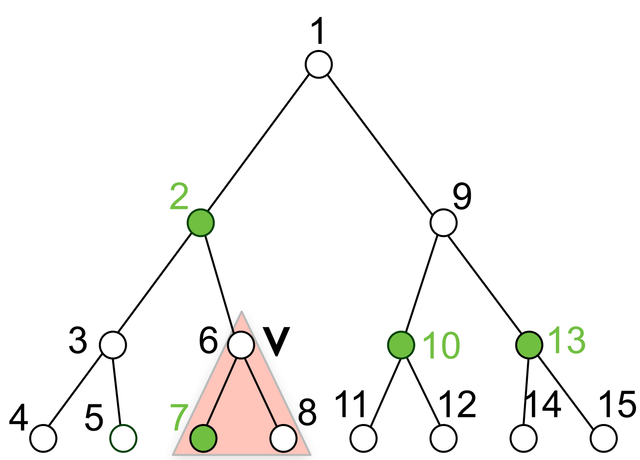
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. Assign preorder numbers
2. For every color: store these numbers in predecessor data structure
(given v find interval of all vertices in its subtree)



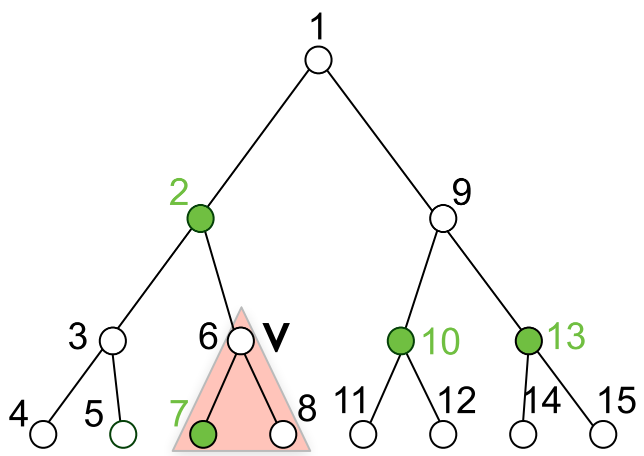
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. Assign preorder numbers
2. For every color: store these numbers in predecessor data structure
(given v find interval of all vertices in its subtree)
plus RMQ data structure (weight = dist. from root)

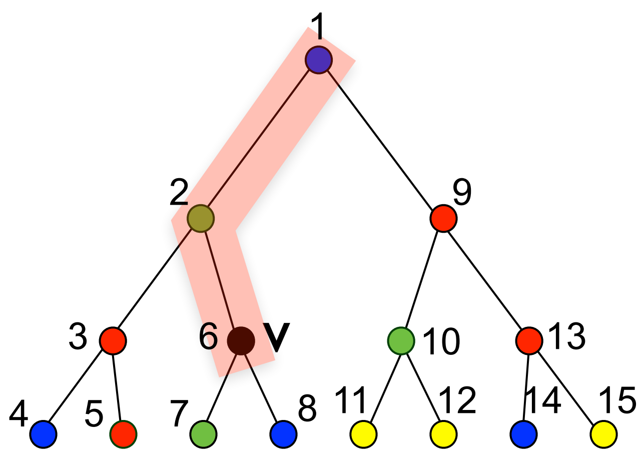


Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution



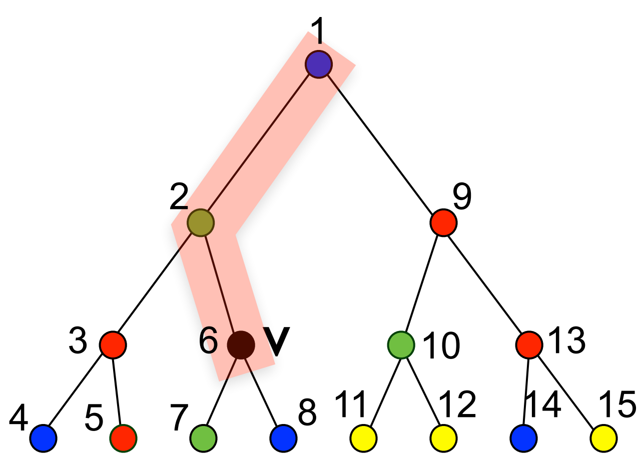
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any O(n polylog n)-space solution

O(log log n)-query O(n)-space solution

1. Assign preorder <u>and postorder</u> numbers



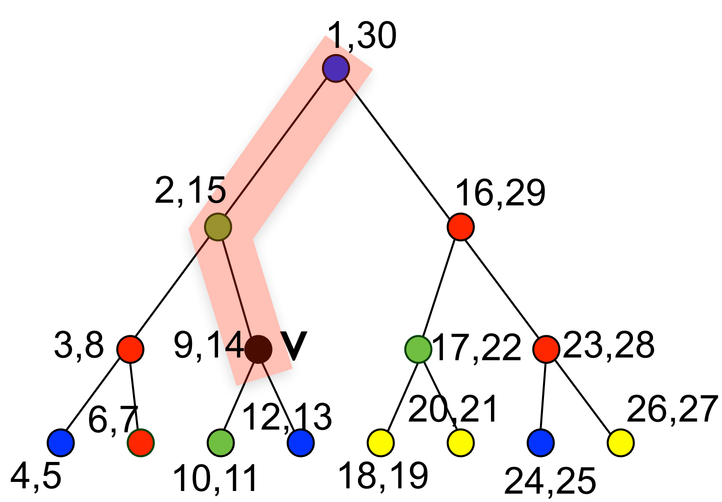
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. Assign preorder and postorder numbers



Static Trees

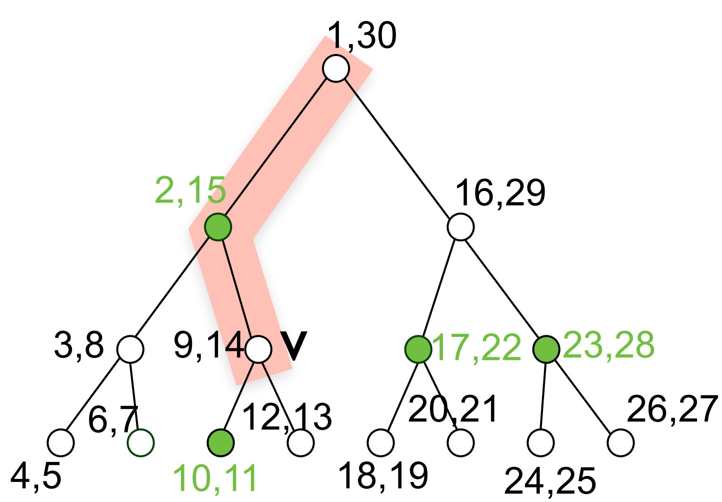
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. Assign preorder <u>and postorder</u> numbers

2. For every color: store these numbers in predecessor data structure



Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

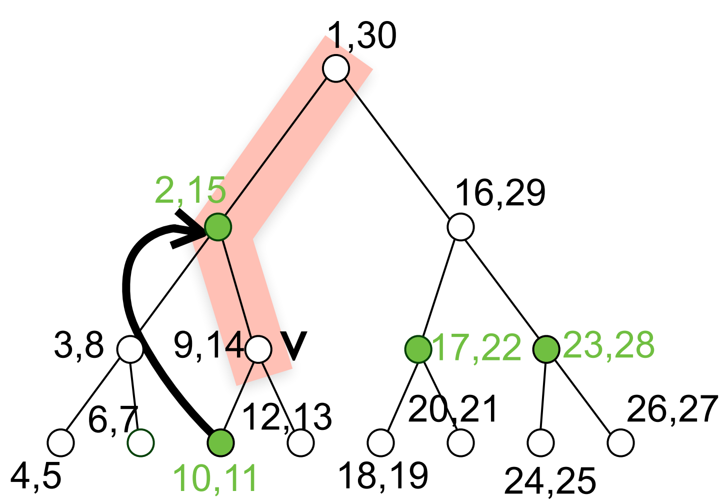
Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. Assign preorder and postorder numbers

2. For every color: store these numbers in predecessor data structure

plus nearest ancestor with the same color

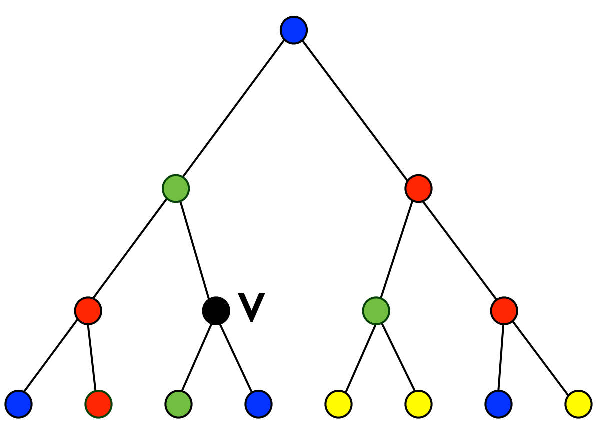


Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

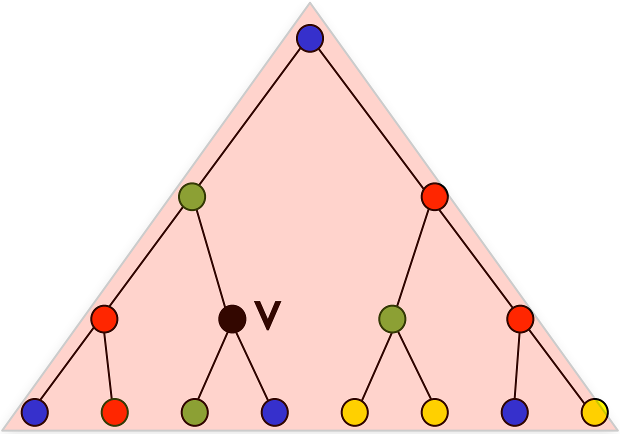


Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution



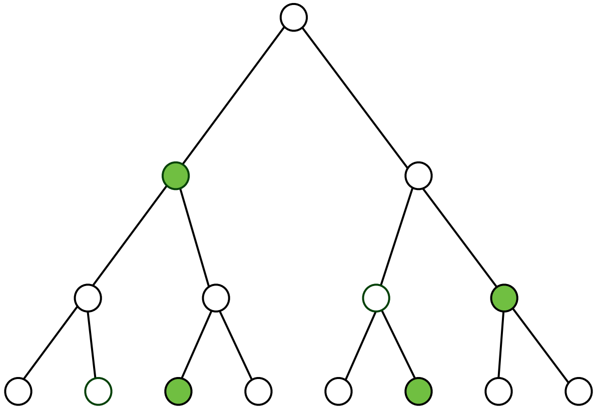
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. For every color: consider all nodes of this color



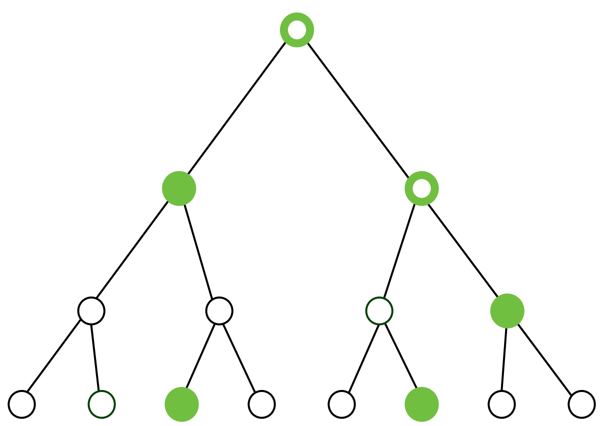
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$\Omega(\log \log n)$-query for any $O(n \text{ polylog } n)$-space solution

$O(\log \log n)$-query $O(n)$-space solution

1. For every color: consider all nodes of this color and their LCAs



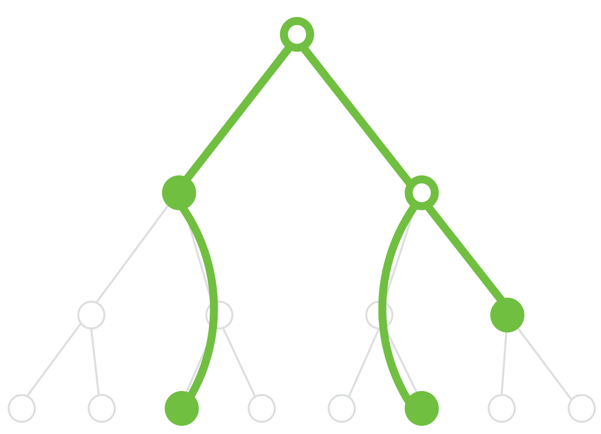
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. For every color: consider all nodes of this color and their LCAs and all edges (paths) between them



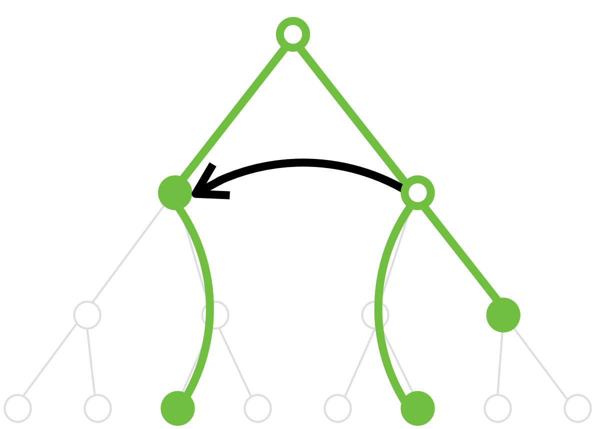
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$\Omega(\log\log n)$-query for any $O(n \text{ polylog } n)$-space solution

$O(\log\log n)$-query $O(n)$-space solution

1. For every color: consider all nodes of this color and their LCAs and all edges (paths) between them plus nearest node with this color



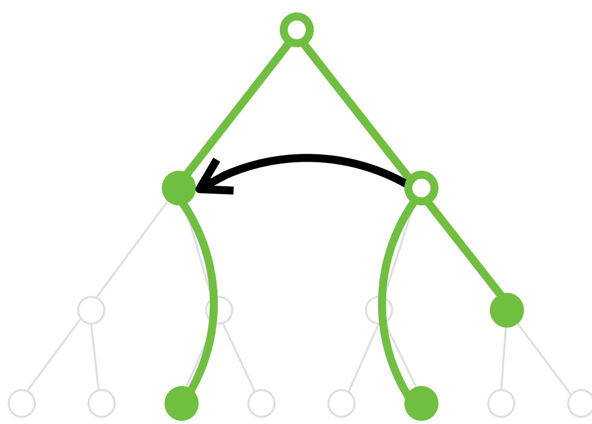
Static Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$\Omega(\log \log n)$-query for any
$O(n \text{ polylog } n)$-space solution

$O(\log \log n)$-query
$O(n)$-space solution

1. For every color: consider all nodes of this color and their LCAs
   and all edges (paths) between them
   plus nearest node with this color



Linear size in #●

Static Trees

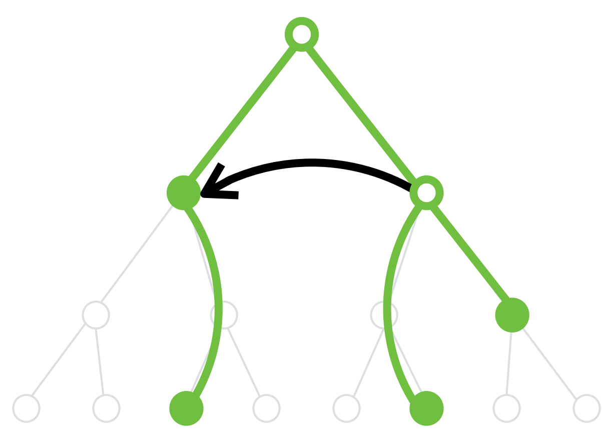
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. For every color: consider all nodes of this color and their LCAs
   and all edges (paths) between them
   plus nearest node with this color

2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in #●

Static Trees

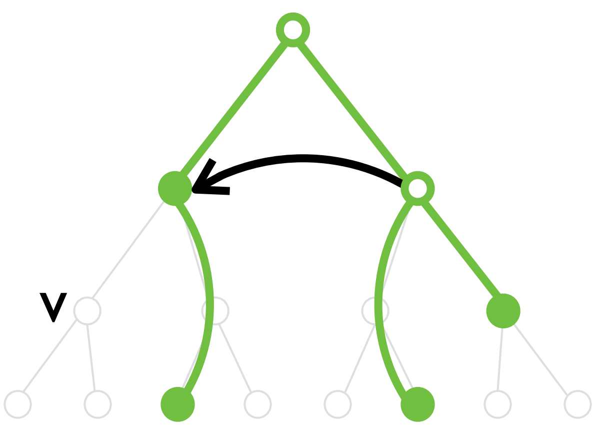
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. For every color: consider all nodes of this color and their LCAs
   and all edges (paths) between them
   plus nearest node with this color

2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in # ●

Static Trees

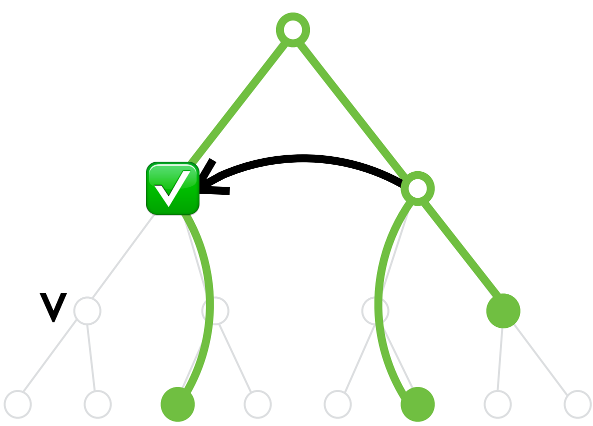
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$\Omega(\log \log n)$-query for any $O(n \text{ polylog } n)$-space solution

$O(\log \log n)$-query $O(n)$-space solution

1. For every color: consider all nodes of this color and their LCAs and all edges (paths) between them plus nearest node with this color

2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in #●

Static Trees

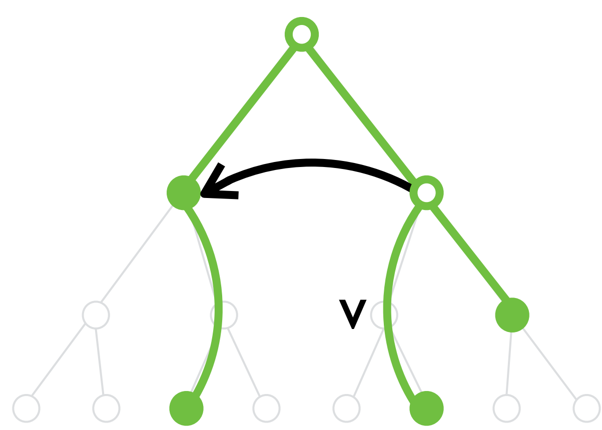
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$\Omega(\log \log n)$-query for any $O(n \text{ polylog } n)$-space solution

$O(\log \log n)$-query $O(n)$-space solution

1. For every color: consider all nodes of this color and their LCAs and all edges (paths) between them plus nearest node with this color

2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in #●

Static Trees

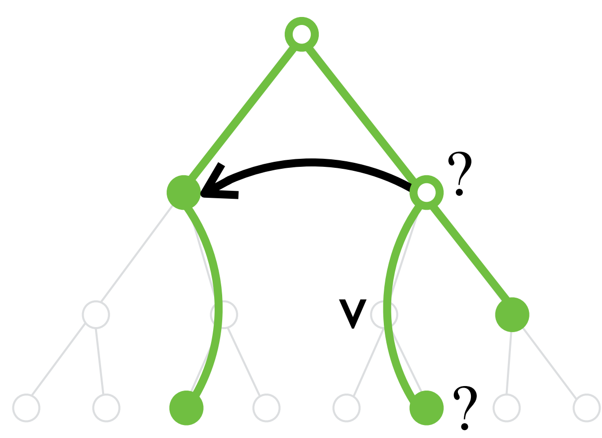
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. For every color: consider all nodes of this color and their LCAs
   and all edges (paths) between them
   plus nearest node with this color

2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in #●

Static Trees

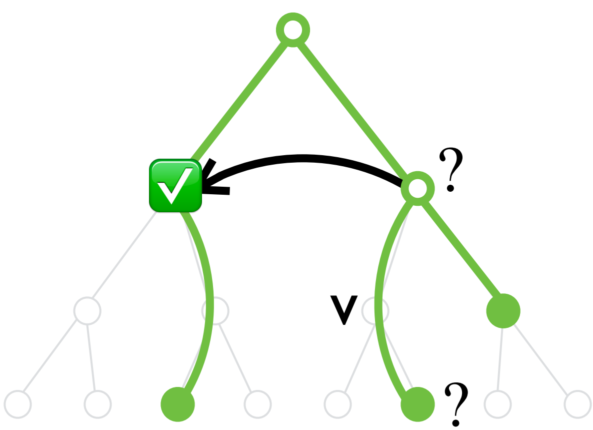
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$\Omega(\log\log n)$-query for any $O(n \operatorname{polylog} n)$-space solution

$O(\log\log n)$-query $O(n)$-space solution

1. For every color: consider all nodes of this color and their LCAs and all edges (paths) between them plus nearest node with this color

2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in #●

Static Trees

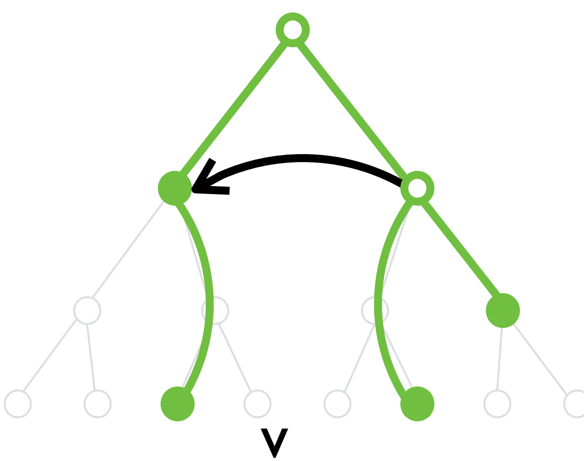
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any
O(n polylog n)-space solution

O(log log n)-query
O(n)-space solution

1. For every color: consider all nodes of this color and their LCAs
   and all edges (paths) between them
   plus nearest node with this color

2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in # ●

Static Trees

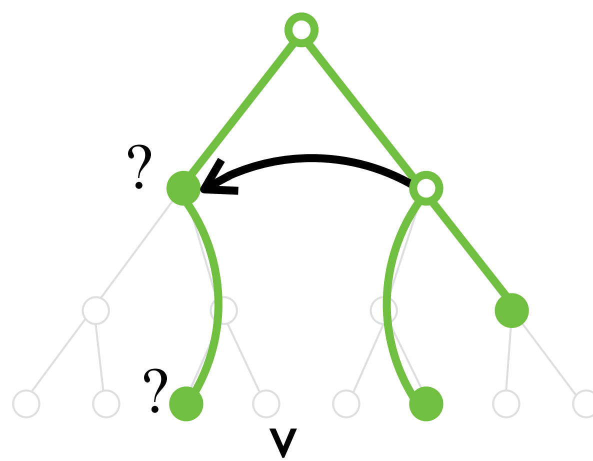
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

Ω(log log n)-query for any O(n polylog n)-space solution

O(log log n)-query O(n)-space solution

1. For every color: consider all nodes of this color and their LCAs and all edges (paths) between them plus nearest node with this color

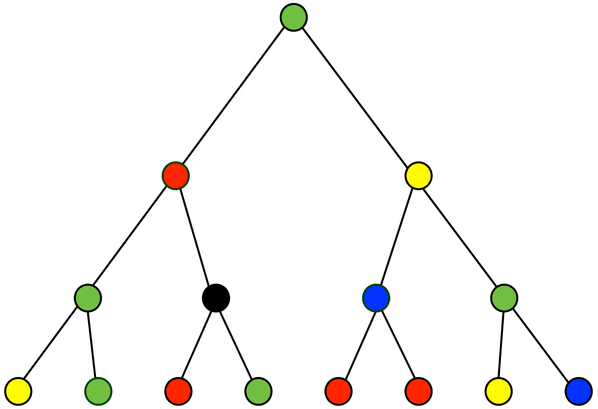
2. Now use the *nearest colored ancestor & descendant* solutions



Linear size in #●

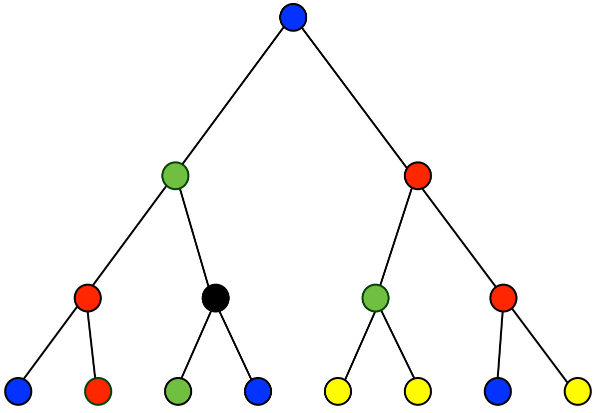
Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



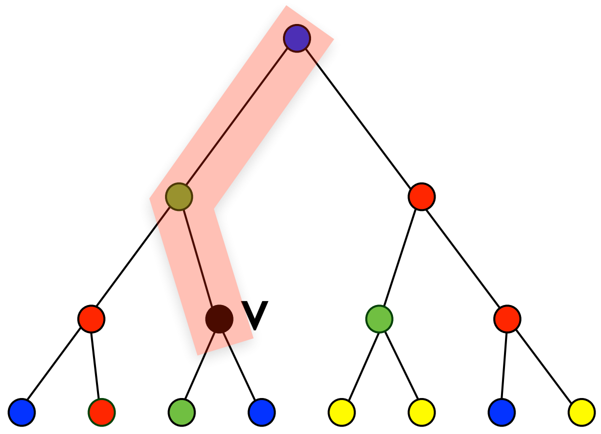
Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

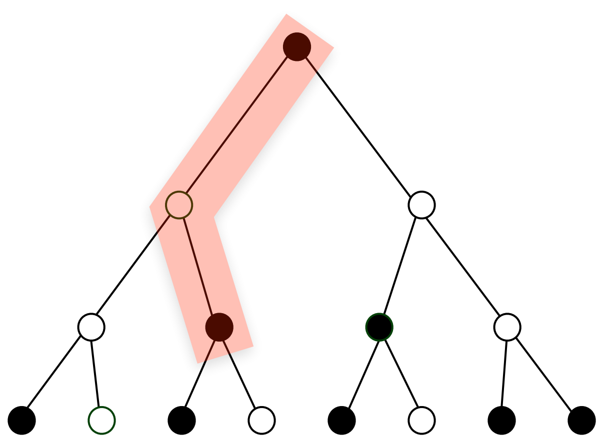


Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

[Alstrup, Husfeldt, Rauhe, 1998]:

O(polylog n)-update requires
Ω(logn/loglog n)-query



Dynamic Trees

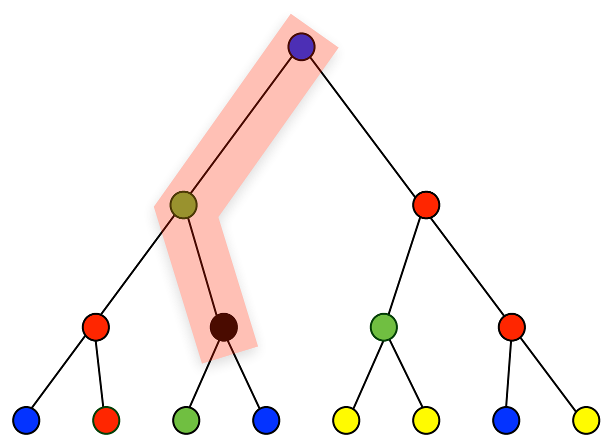
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

[Alstrup, Husfeldt, Rauhe, 1998]:

O(polylog n)-update requires
Ω(logn/loglog n)-query

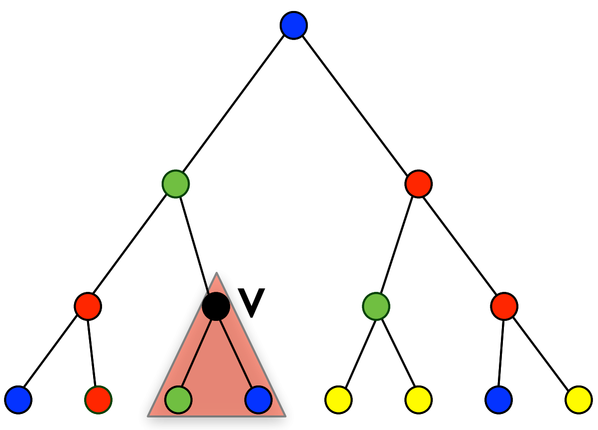
[Alstrup, Husfeldt, Rauhe, 1998]:

O(log log n)-update
O(logn/loglog n)-query
O(n)-space



Dynamic Trees

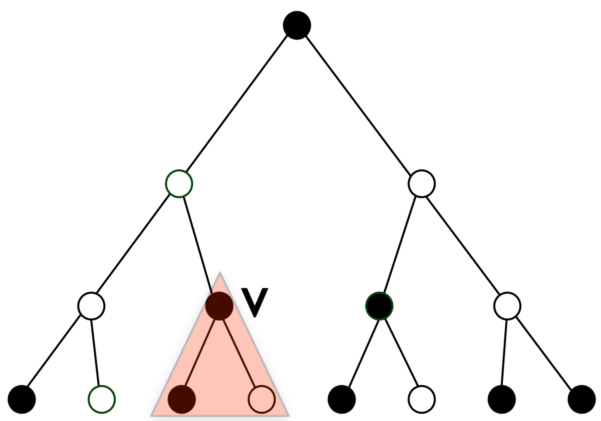
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

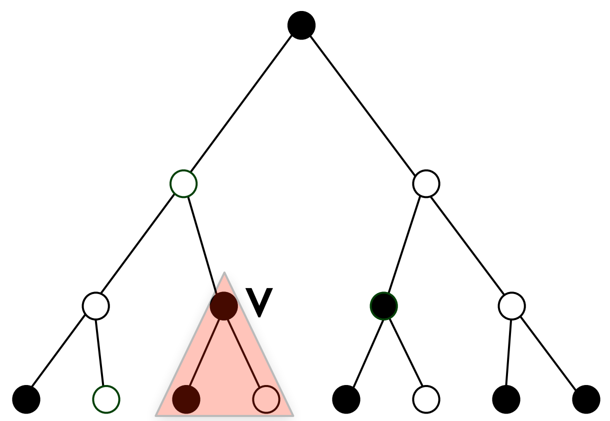


Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires Ω(logn/loglog n)-query

Lower bound via three simple reductions:



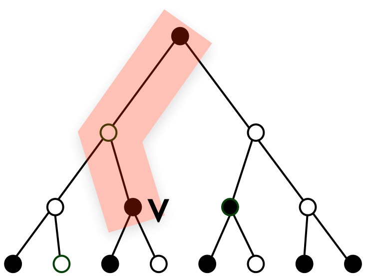
Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires Ω(logn/loglog n)-query

Lower bound via three simple reductions:

dynamic nearest
colored ancestor



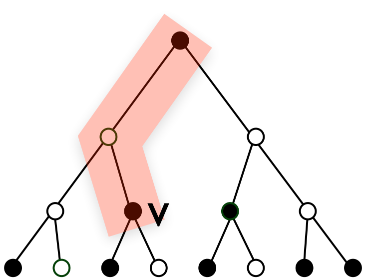
Dynamic Trees

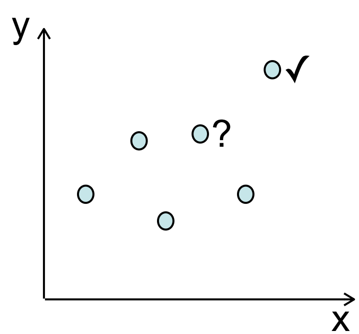
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

Lower bound via three simple reductions:

dynamic nearest colored ancestor → dynamic planar dominance emptiness





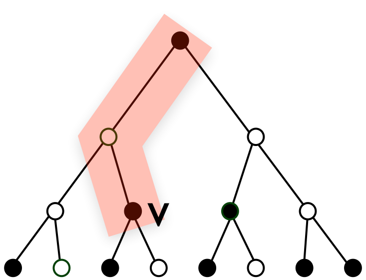
Dynamic Trees

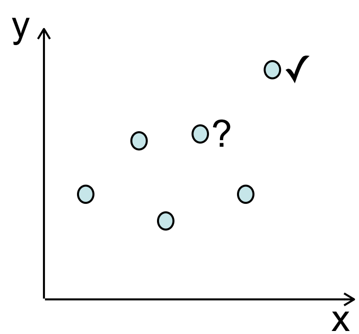
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

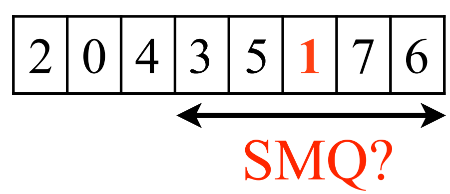
O(polylog n)-update requires Ω(logn/loglog n)-query

Lower bound via three simple reductions:

dynamic nearest colored ancestor → dynamic planar dominance emptiness → dynamic Suffix Minimum Queries







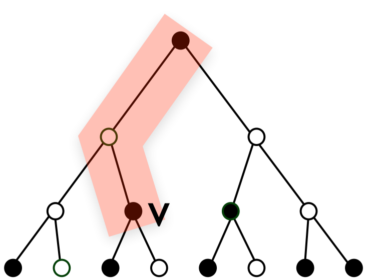
Dynamic Trees

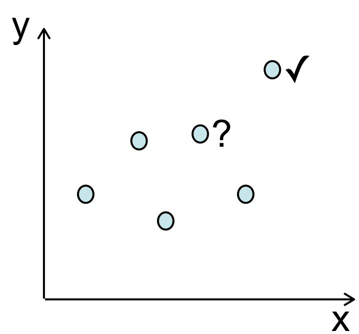
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

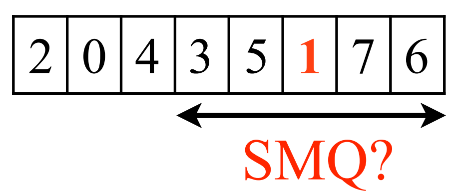
O(polylog n)-update requires Ω(logn/loglog n)-query

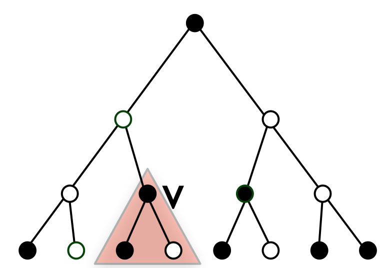
Lower bound via three simple reductions:

dynamic nearest colored ancestor → dynamic planar dominance emptiness → dynamic Suffix Minimum Queries → dynamic nearest colored descendant







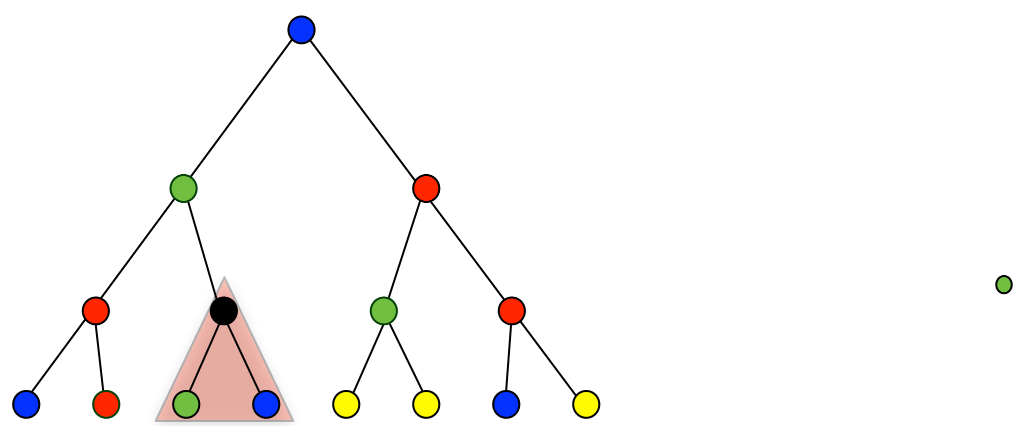


Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

O($\log^{2/3}$ n)-update
O(logn/loglog n)-query
O(n)-space



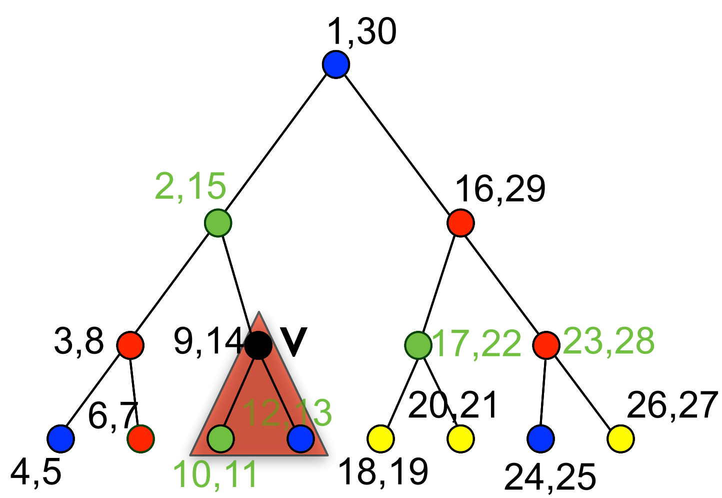
Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

O($\log^{2/3}$ n)-update
O(logn/loglog n)-query
O(n)-space

1. Assign preorder and postorder numbers



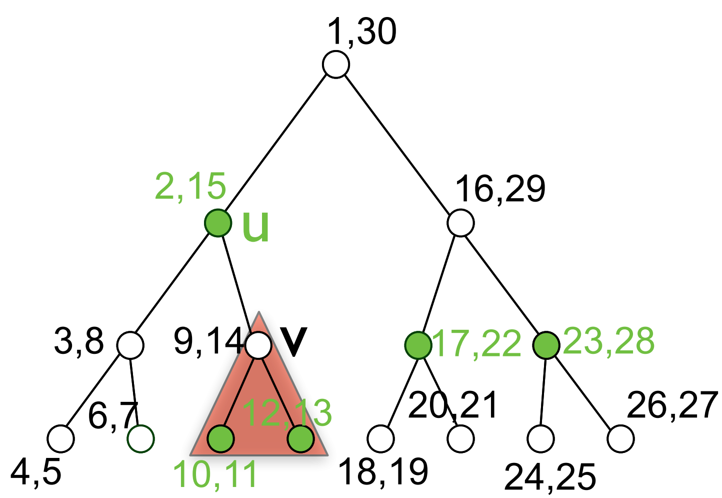
Dynamic Trees

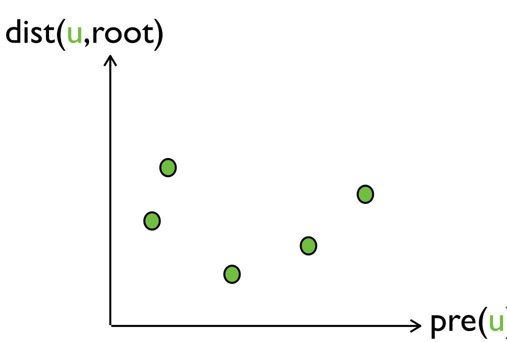
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

$O(\log^{2/3} n)$-update
O(logn/loglog n)-query
O(n)-space

1. Assign preorder and postorder numbers
2. For every color: store points (pre(u), dist(u,root)) in the 3-sided planar emptiness structure of [Wilkinson 2014]





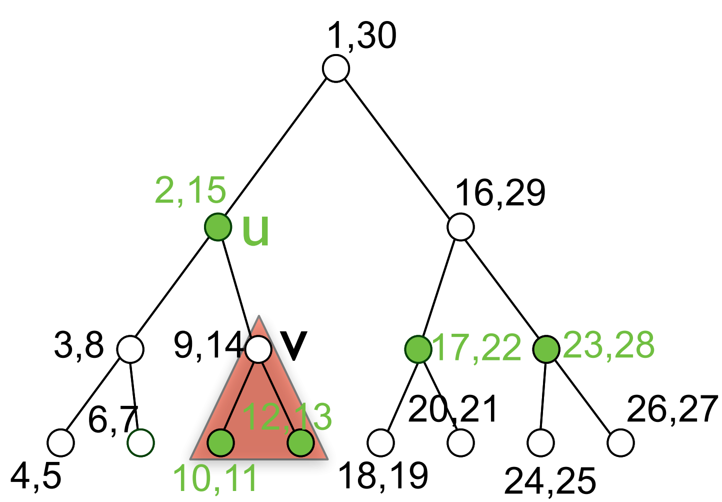
Dynamic Trees

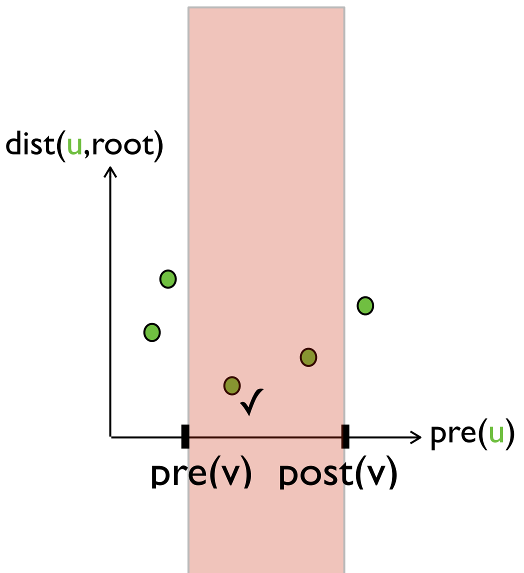
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

$O(\log^{2/3} n)$-update
O(logn/loglog n)-query
O(n)-space

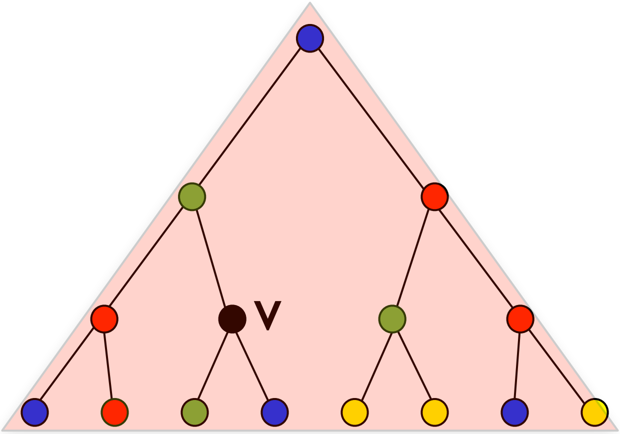
1. Assign preorder and postorder numbers
2. For every color: store points (pre(u), dist(u,root)) in the 3-sided planar emptiness structure of [Wilkinson 2014]





Dynamic Trees

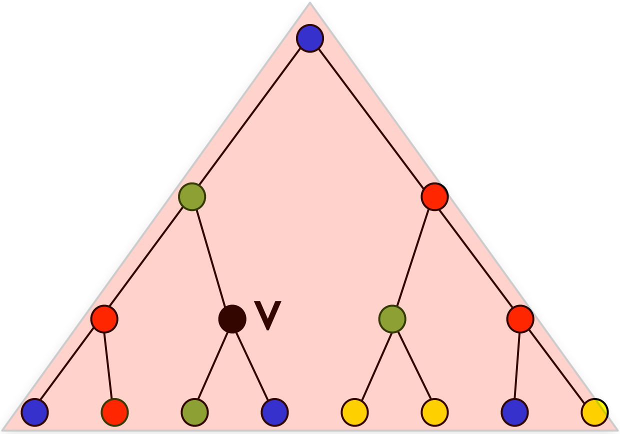
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query



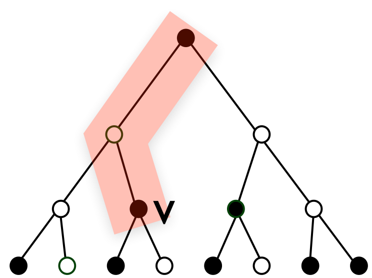
Dynamic Trees

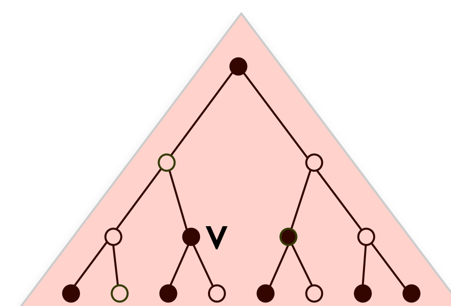
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

dynamic nearest
colored ancestor

dynamic nearest
colored node



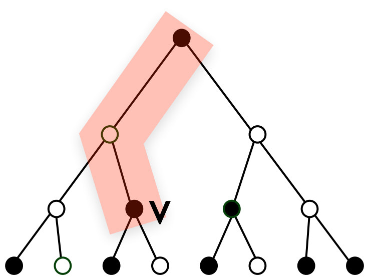


Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

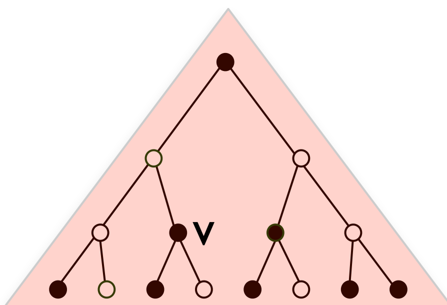
O(polylog n)-update requires Ω(logn/loglog n)-query

dynamic nearest colored ancestor



Blow up the lower bound tree weights exponentially so that nearest colored node is always an ancestor

dynamic nearest colored node

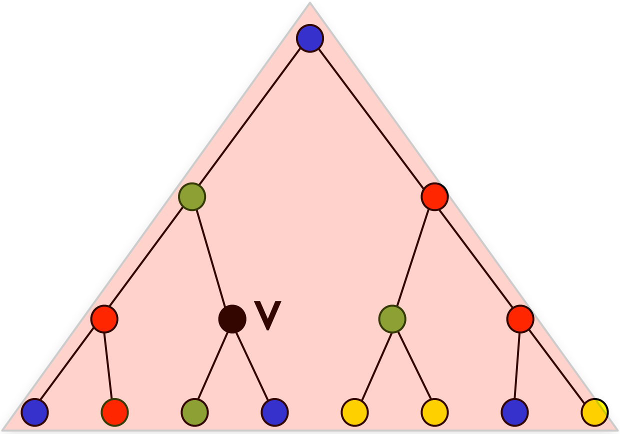


Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

O(log n)-update
O(log n)-query
O(n)-space



Dynamic Trees

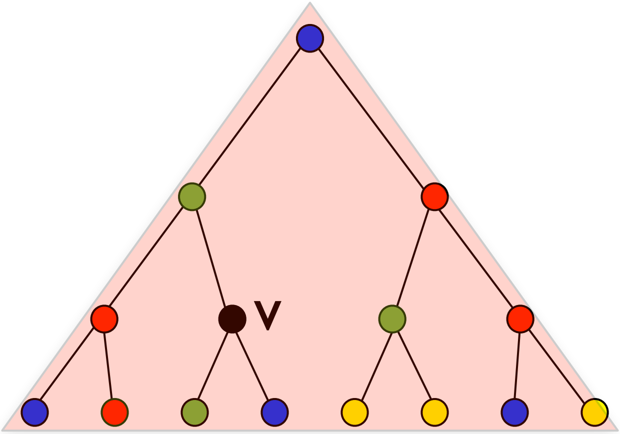
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(polylog n)-update requires
Ω(logn/loglog n)-query

O(log n)-update
O(log n)-query
O(n)-space

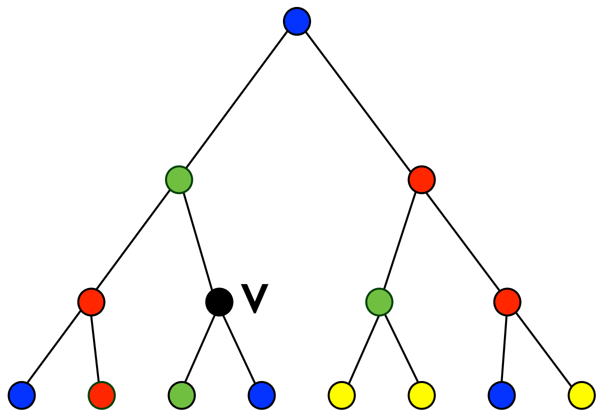
Most technical part of the paper:

Uses all of the above machinery plus a hybrid of Centroid decomposition and Top Trees, augmented with nearest colored centroid and with LCA…



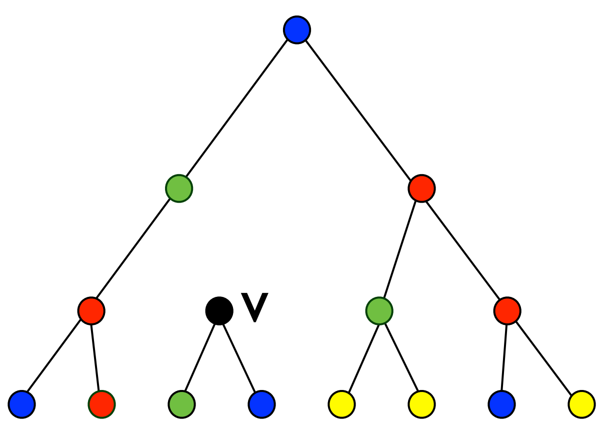
Fully Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



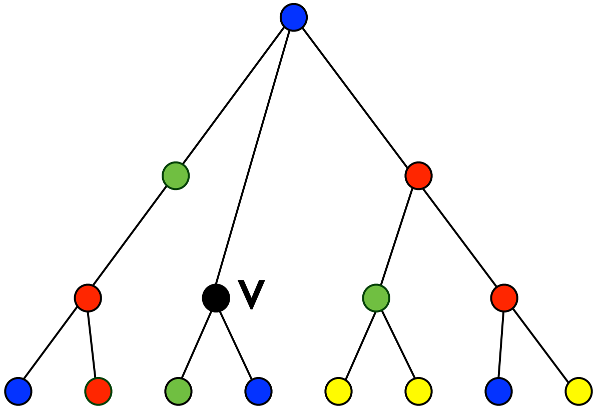
Fully Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node



Fully Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

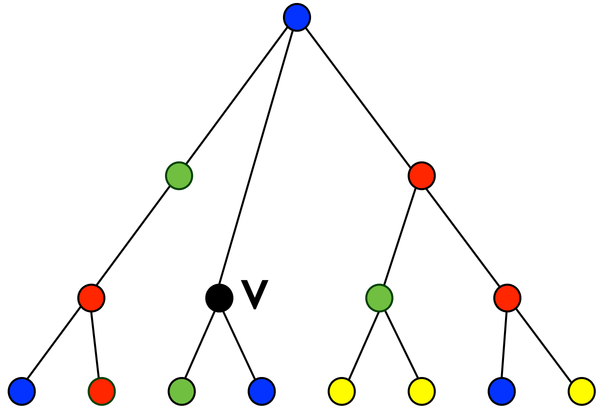


Fully Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space



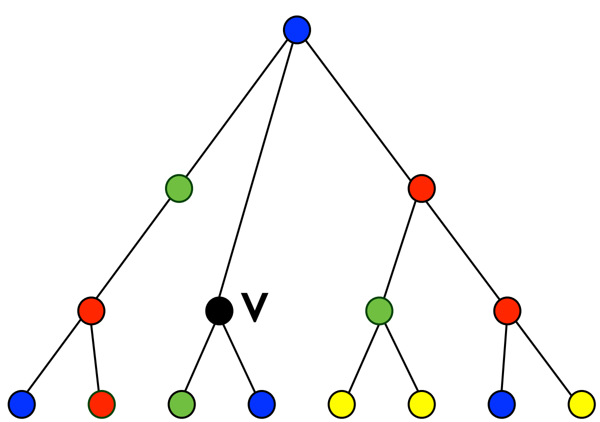
Fully Dynamic Trees

- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(#colors$^{1-\varepsilon}$) query and update implies an O(n$^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space



Fully Dynamic Trees

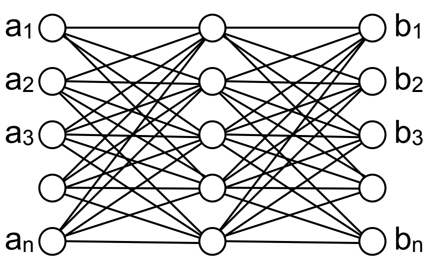
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space

APSP



Fully Dynamic Trees

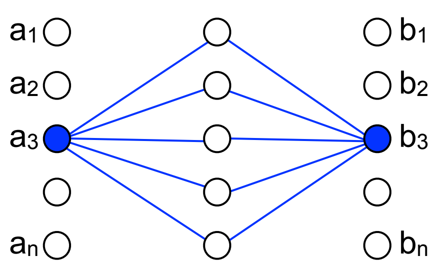
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

O(#colors$^{1-\varepsilon}$) query and update implies an O(n$^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space

APSP



Fully Dynamic Trees

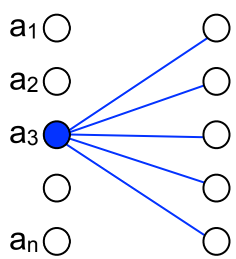
- Nearest colored ancestor
- Nearest colored descendant
- Nearest colored node

$O(\#\text{colors}^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#\text{colors} \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space

APSP

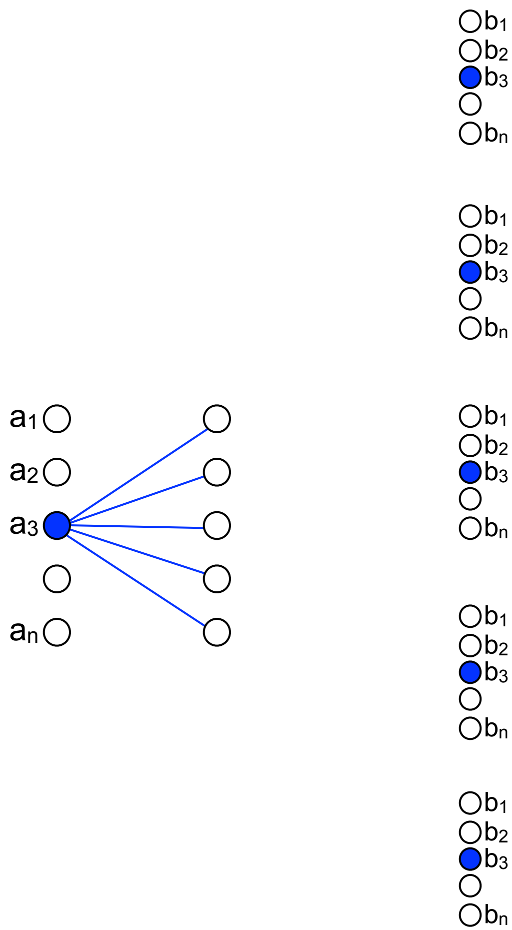


Fully Dynamic Trees

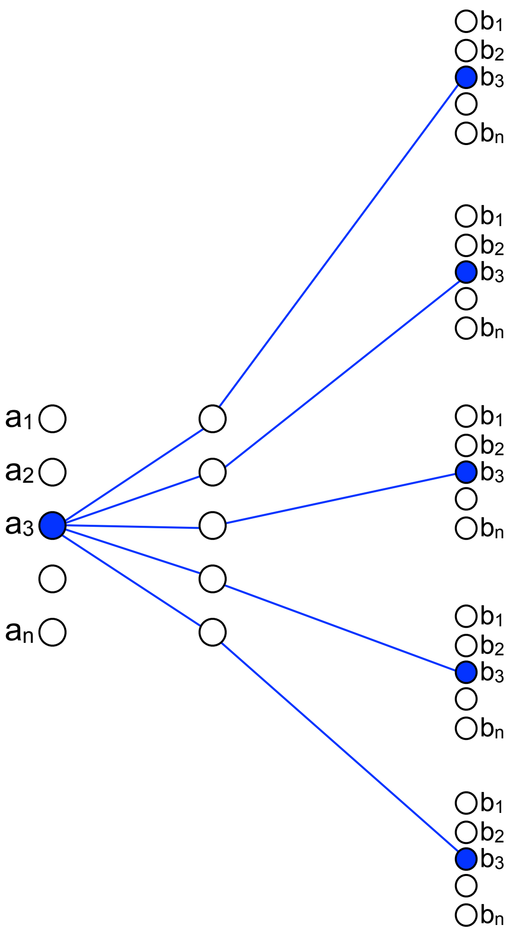
$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space



Fully Dynamic Trees

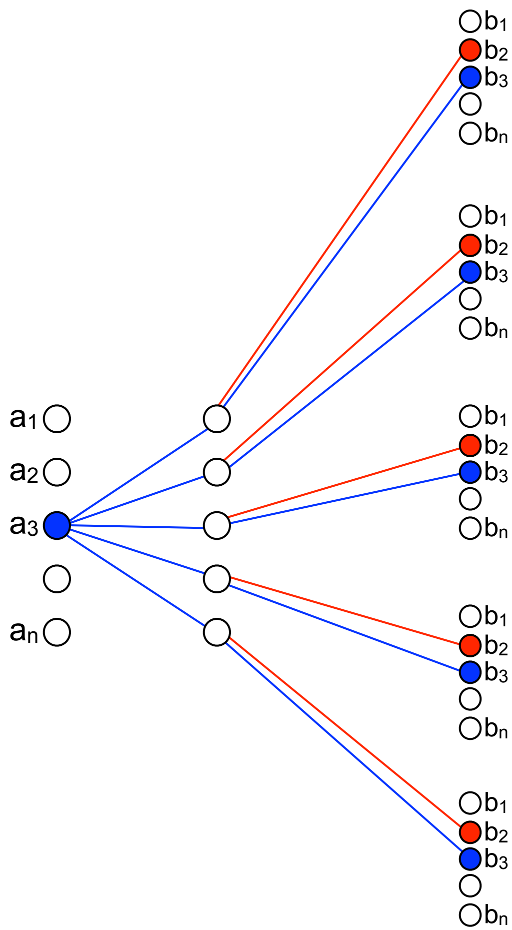


O(#colors$^{1-\varepsilon}$) query and update implies an O($n^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space

Fully Dynamic Trees



O(#colors$^{1-\varepsilon}$) query and update implies an O(n$^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

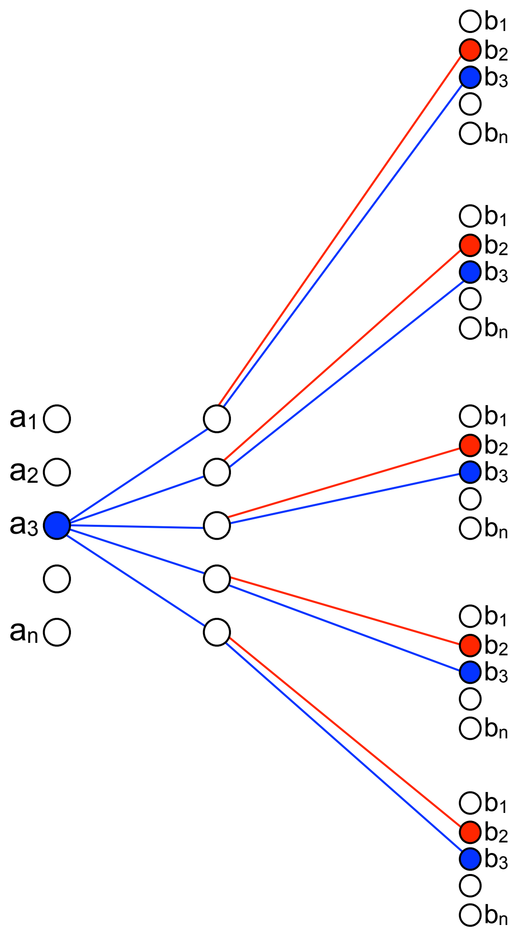
O(#colors · log n)-update
O(log n)-query
O(n)-space

Fully Dynamic Trees

$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space



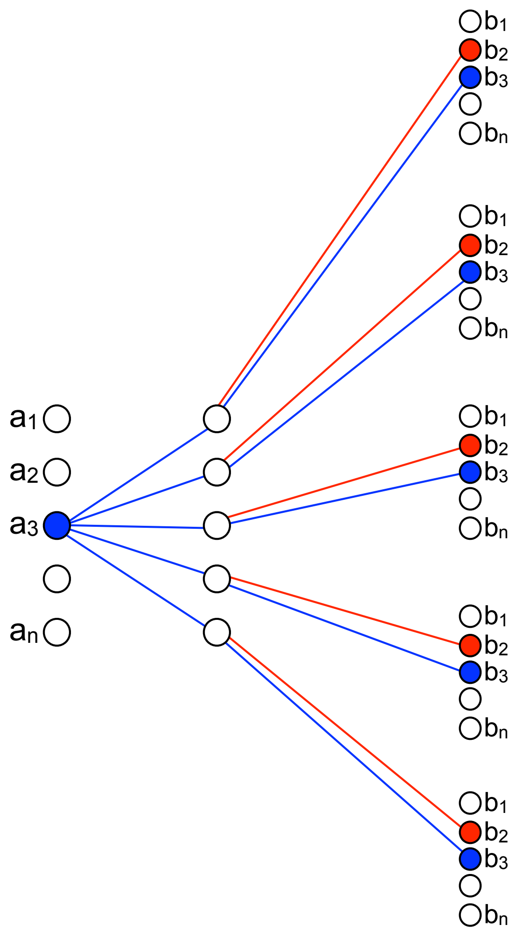
Tree with n^2 vertices and #colors = n

Fully Dynamic Trees

O(#colors$^{1-\varepsilon}$) query and update implies an O($n^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space



Tree with n^2 vertices and #colors = n

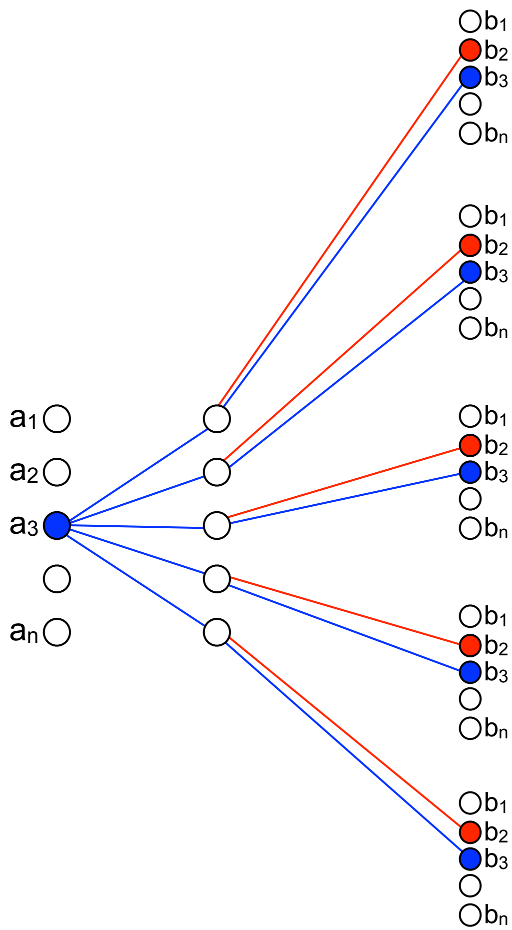
What's the closest blue to a_3 ?

Fully Dynamic Trees

$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space



Tree with n^2 vertices and #colors = n

What's the closest blue to a_3 ?

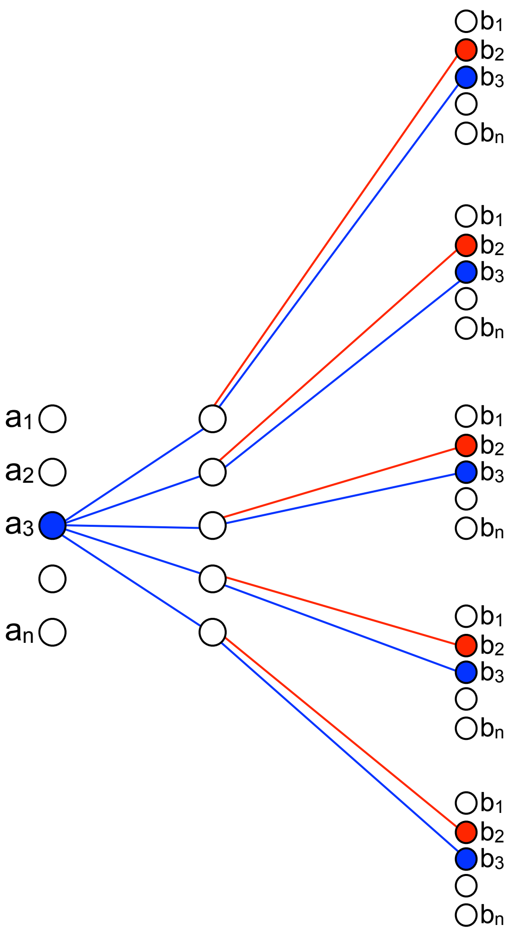
What's the closest red to a_3 ?

Fully Dynamic Trees

O(#colors$^{1-\varepsilon}$) query and update implies an O($n^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space



Tree with n^2 vertices and #colors = n

What's the closest blue to a_3 ?

What's the closest red to a_3 ?

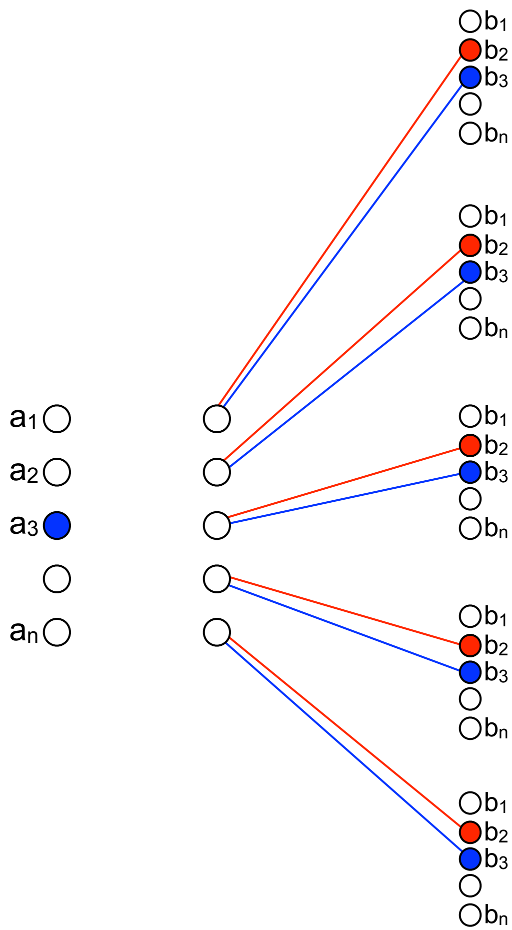
…

Fully Dynamic Trees

$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space



Tree with n^2 vertices and #colors = n

What's the closest blue to a_3 ?

What's the closest red to a_3 ?

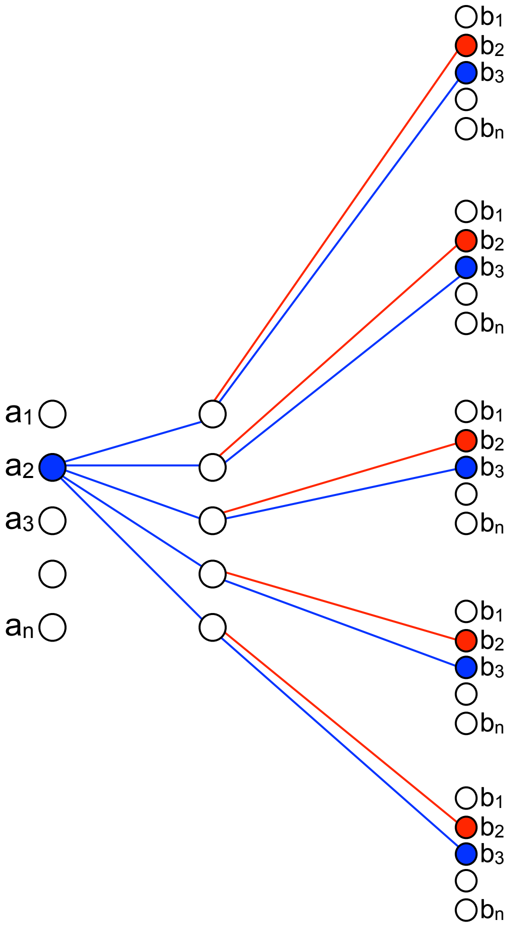
…

Fully Dynamic Trees

O(#colors$^{1-\varepsilon}$) query and update implies an O($n^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space



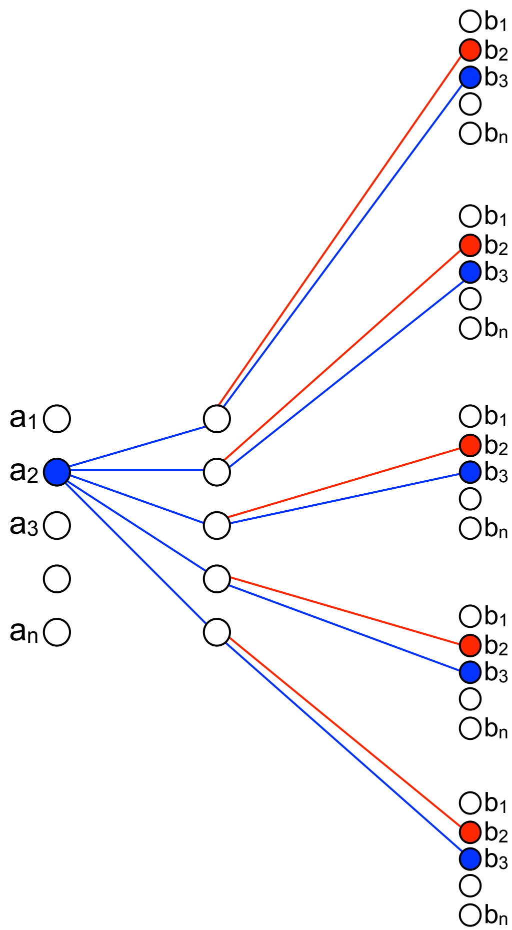
Tree with n^2 vertices and #colors = n

Fully Dynamic Trees

$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space



Tree with n^2 vertices and #colors = n

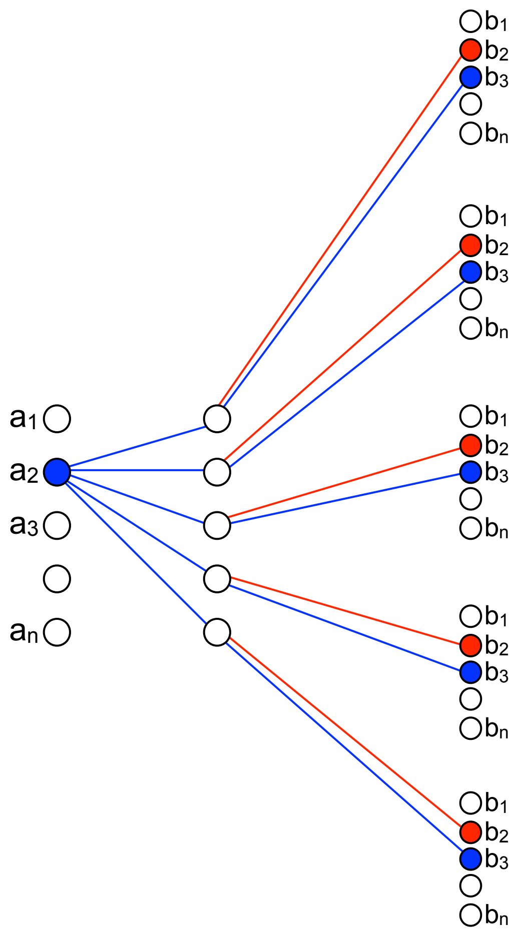
What's the closest blue to a_2 ?

Fully Dynamic Trees

O(#colors$^{1-\varepsilon}$) query and update implies an O(n$^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space



Tree with n^2 vertices and #colors = n

What's the closest blue to a_2 ?

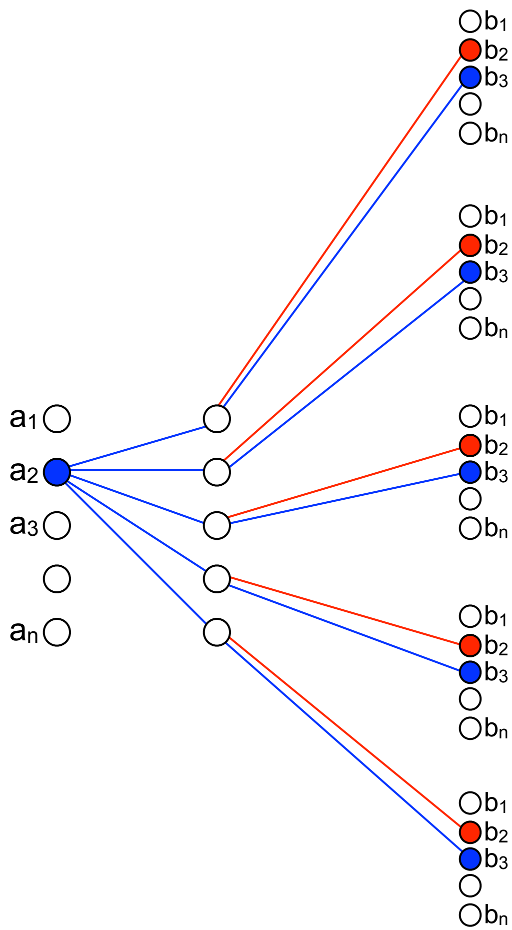
What's the closest red to a_2 ?

Fully Dynamic Trees

O(#colors$^{1-\varepsilon}$) query and update implies an O(n$^{3-\varepsilon}$) for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

O(#colors · log n)-update
O(log n)-query
O(n)-space



Tree with n^2 vertices and #colors = n

What's the closest blue to a_2 ?

What's the closest red to a_2 ?

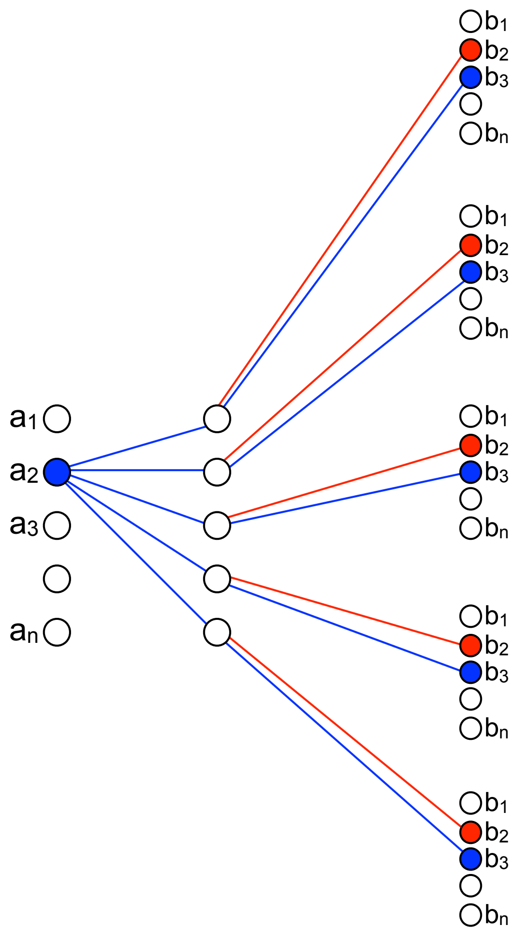
…

Fully Dynamic Trees

$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space



Tree with n^2 vertices and #colors = n

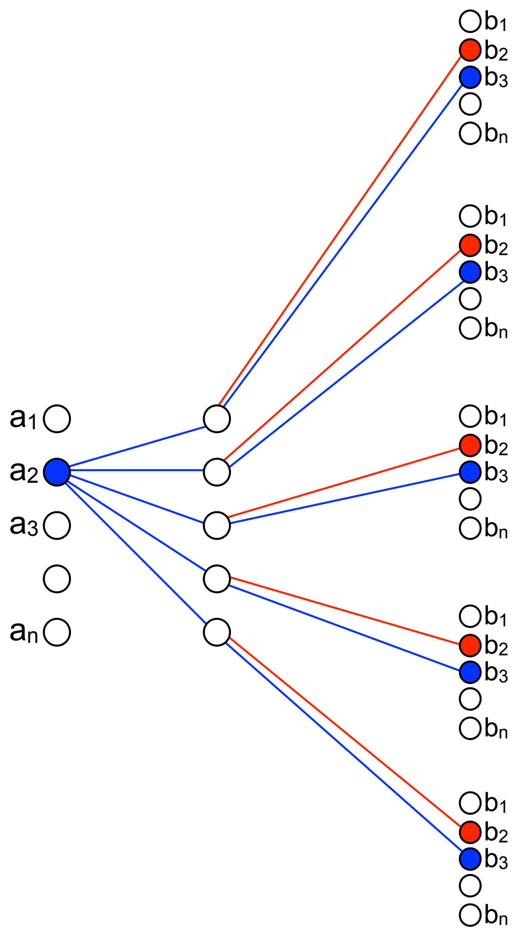
Overall:
n^2 updates and n^2 queries

Fully Dynamic Trees

$O(\#colors^{1-\varepsilon})$ query and update implies an $O(n^{3-\varepsilon})$ for APSP

[Alstrup, Holm, de-Lichtenberg, Thorup 2005]

$O(\#colors \cdot \log n)$-update
$O(\log n)$-query
$O(n)$-space



Tree with n^2 vertices and #colors = n

Overall:
n^2 updates and n^2 queries

If we could do each in less than $\#colors^{1-\varepsilon} = n^{1-\varepsilon}$ time then total is $n^{3-\varepsilon}$

Open Problems

- Nearest colored descendant

  $O(\log^{2/3} n)$-update
  $O(\log n / \log\log n)$-query
  $O(n)$-space

- Nearest colored node

  $O(\log n)$-update
  $O(\log n)$-query
  $O(n)$-space

Toh-dah !