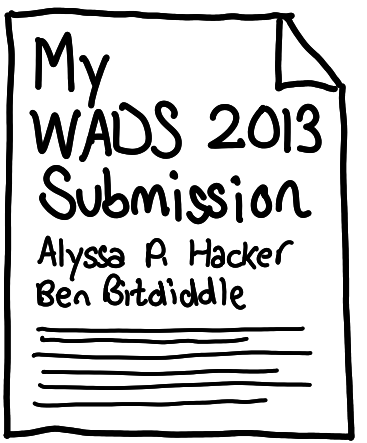
Blame Trees

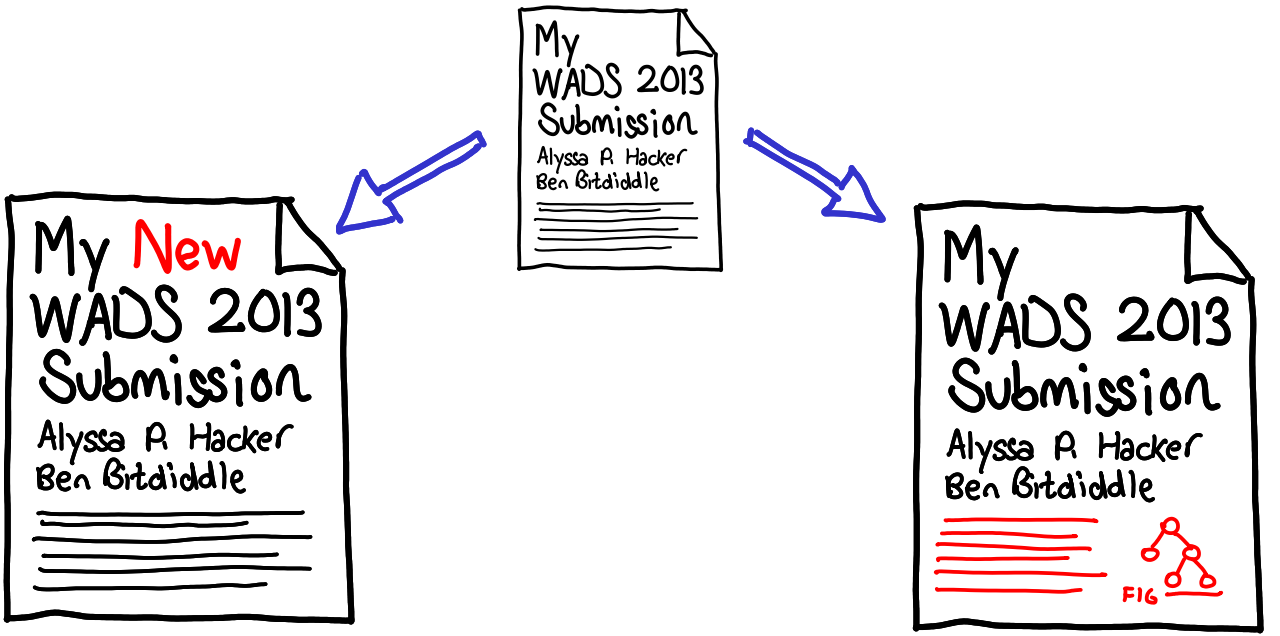
Erik D. Demaine (MIT)
Pavel Panchekha (MIT)
David A. Wilson (MIT)
Edward Z. Yang (Stanford)

WADS 2013

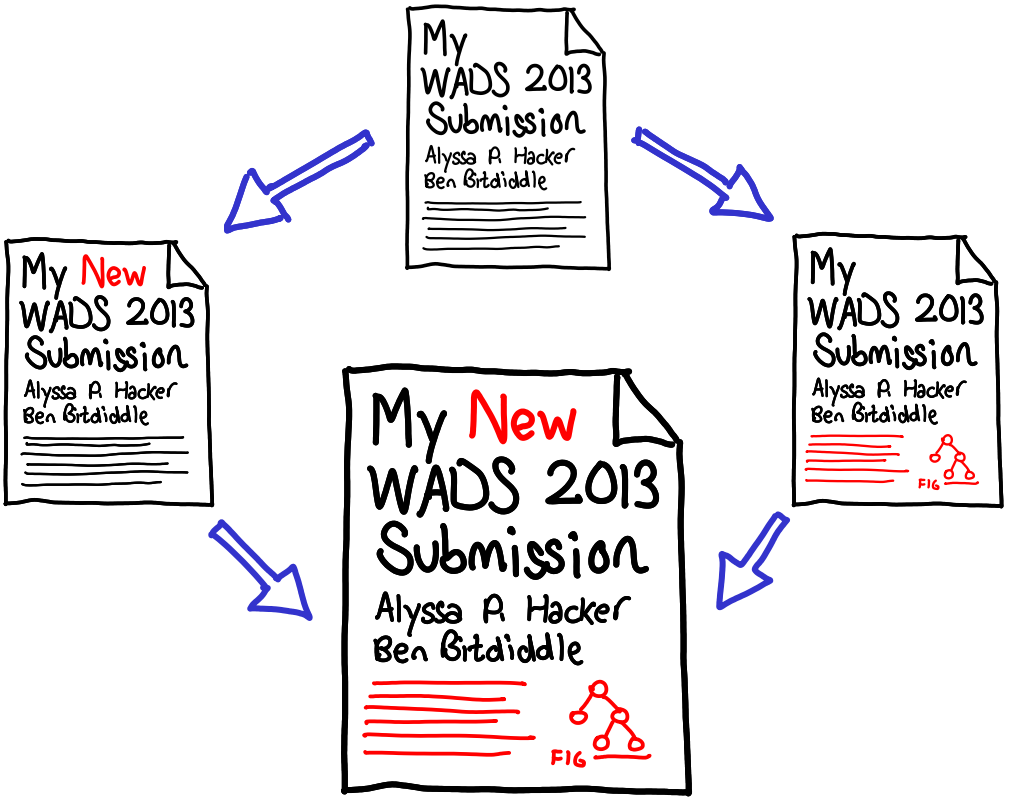
Document Merge
in Version Control Systems



Document Merge
in Version Control Systems



Document Merge
in Version Control Systems



Document Merge
in Version Control Systems

Blame trees are a functional* data structure which can be merged efficiently.

* implies confluently persistent

Document Merge
in Version Control Systems

Blame trees are a functional* data structure which can be merged efficiently.

?

* implies confluently persistent

Document Merge
in Version Control Systems

Blame trees are a functional* data structure which can be merged efficiently.

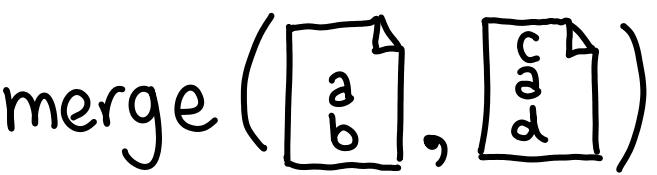
?

Git, Mercurial, SVN, Perforce, Darcs, Bazaar, CVS, RCS, Bitkeeper, Monotone, Fossil, Veracity ...

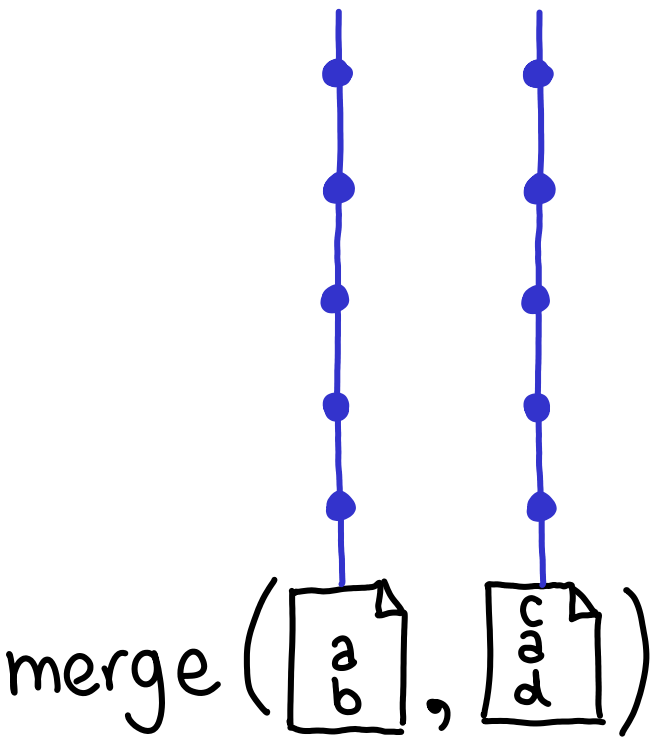
* implies confluently persistent

Let's look at this from an algorithms & data structures perspective

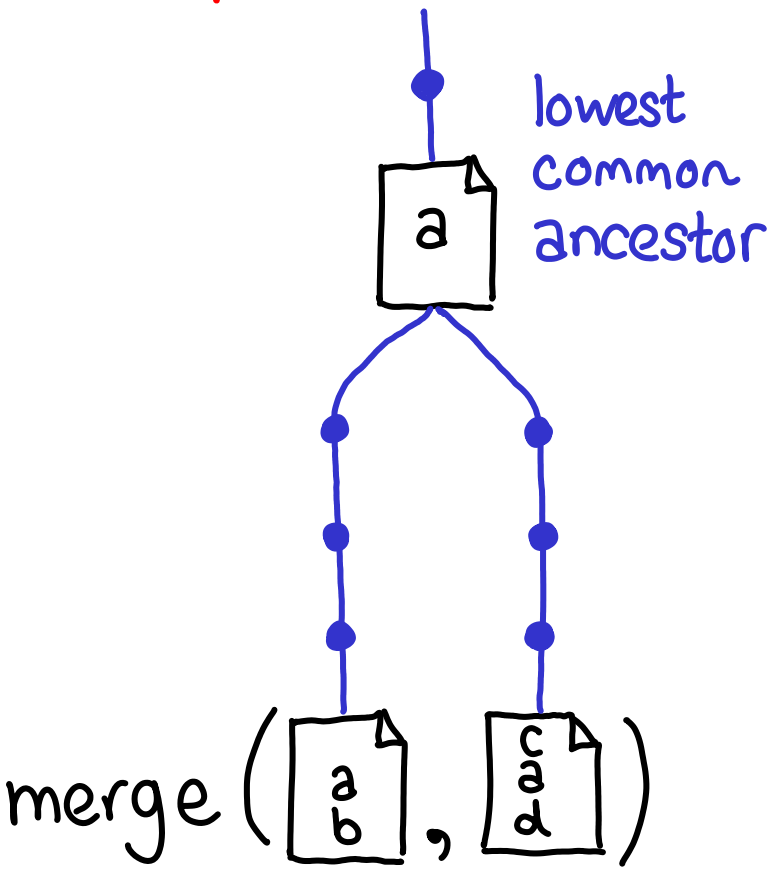
three-way recursive merge [Git]



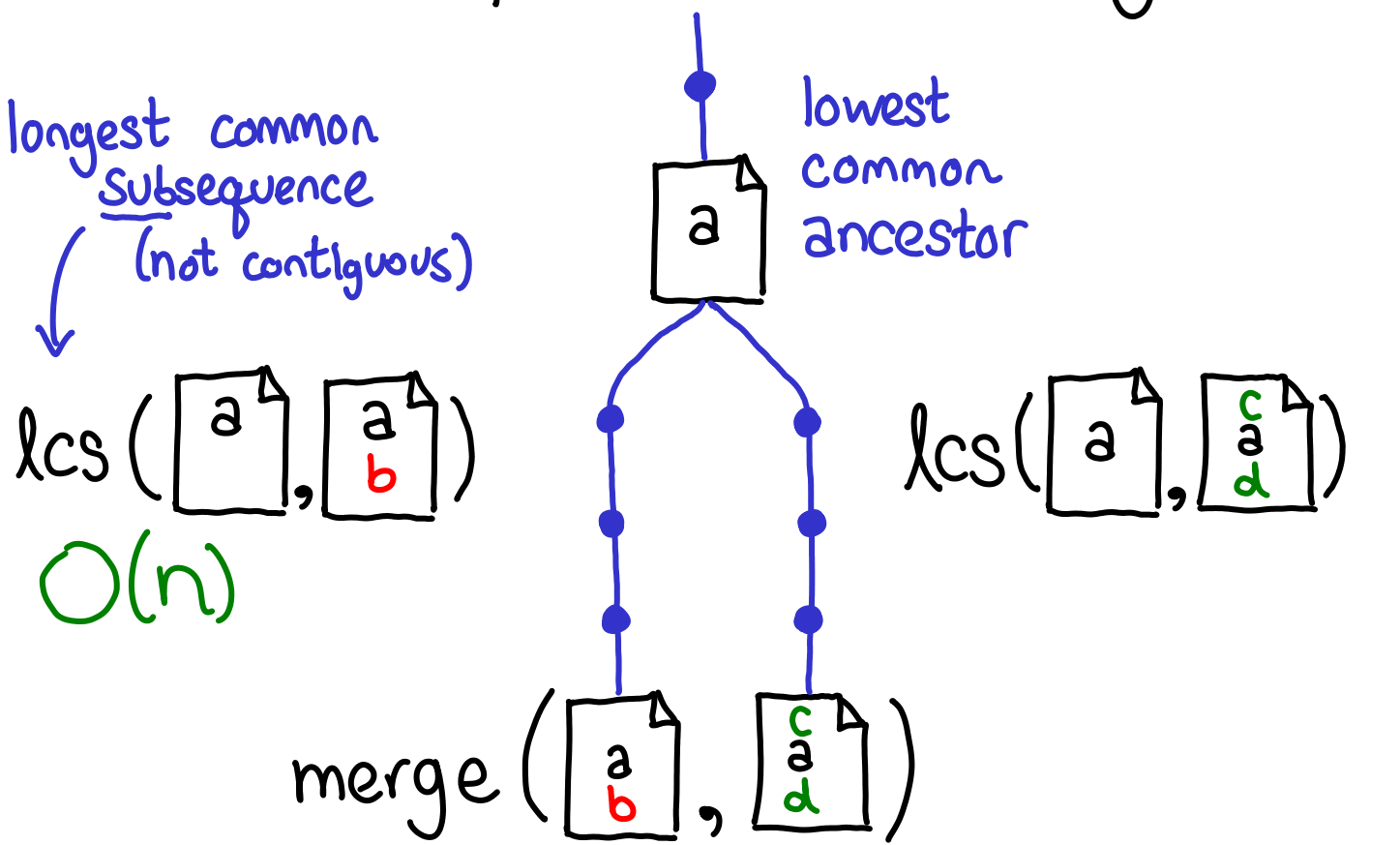
three-way recursive merge [Git]



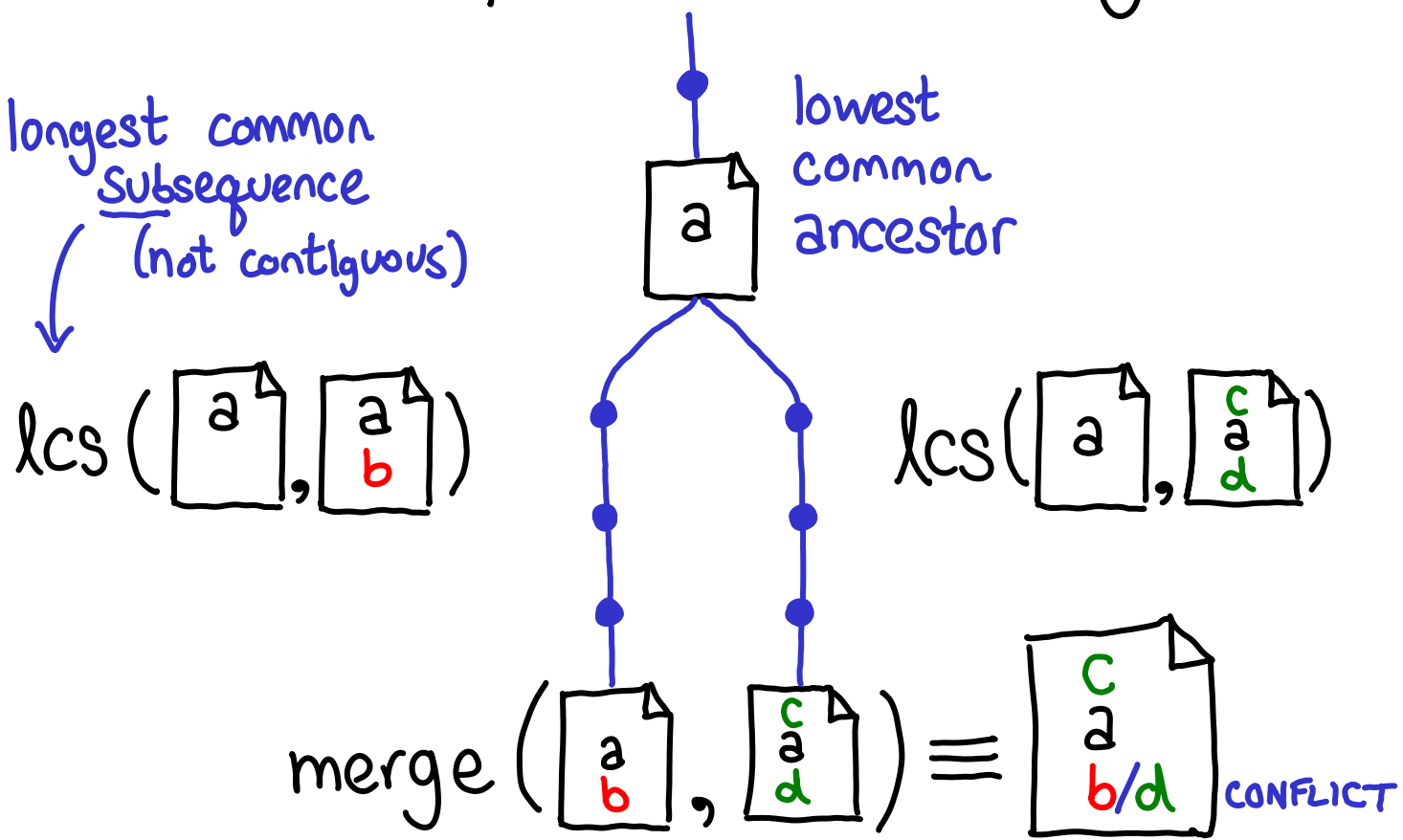
three-way recursive merge [Git]



three-way recursive merge [Git]

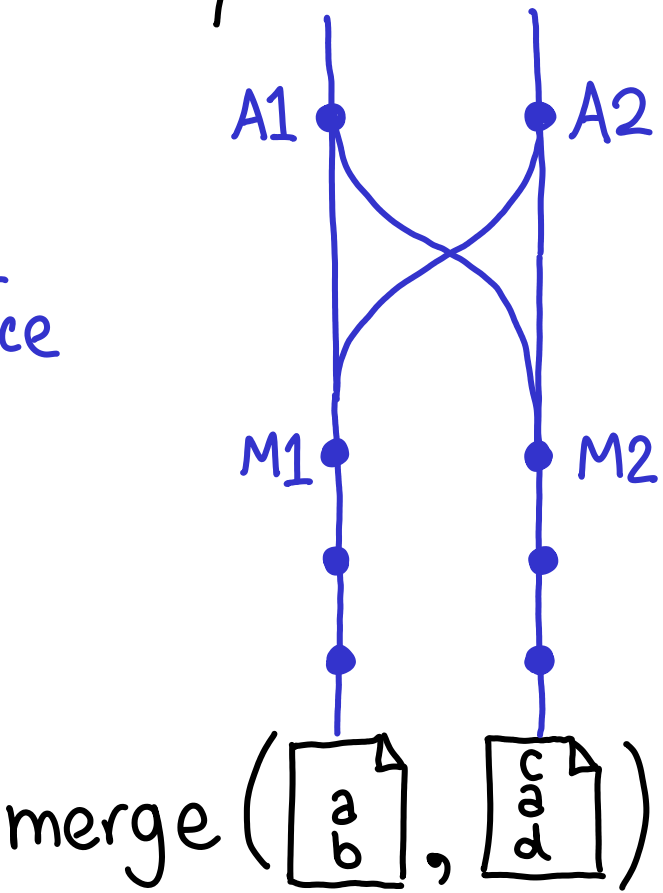


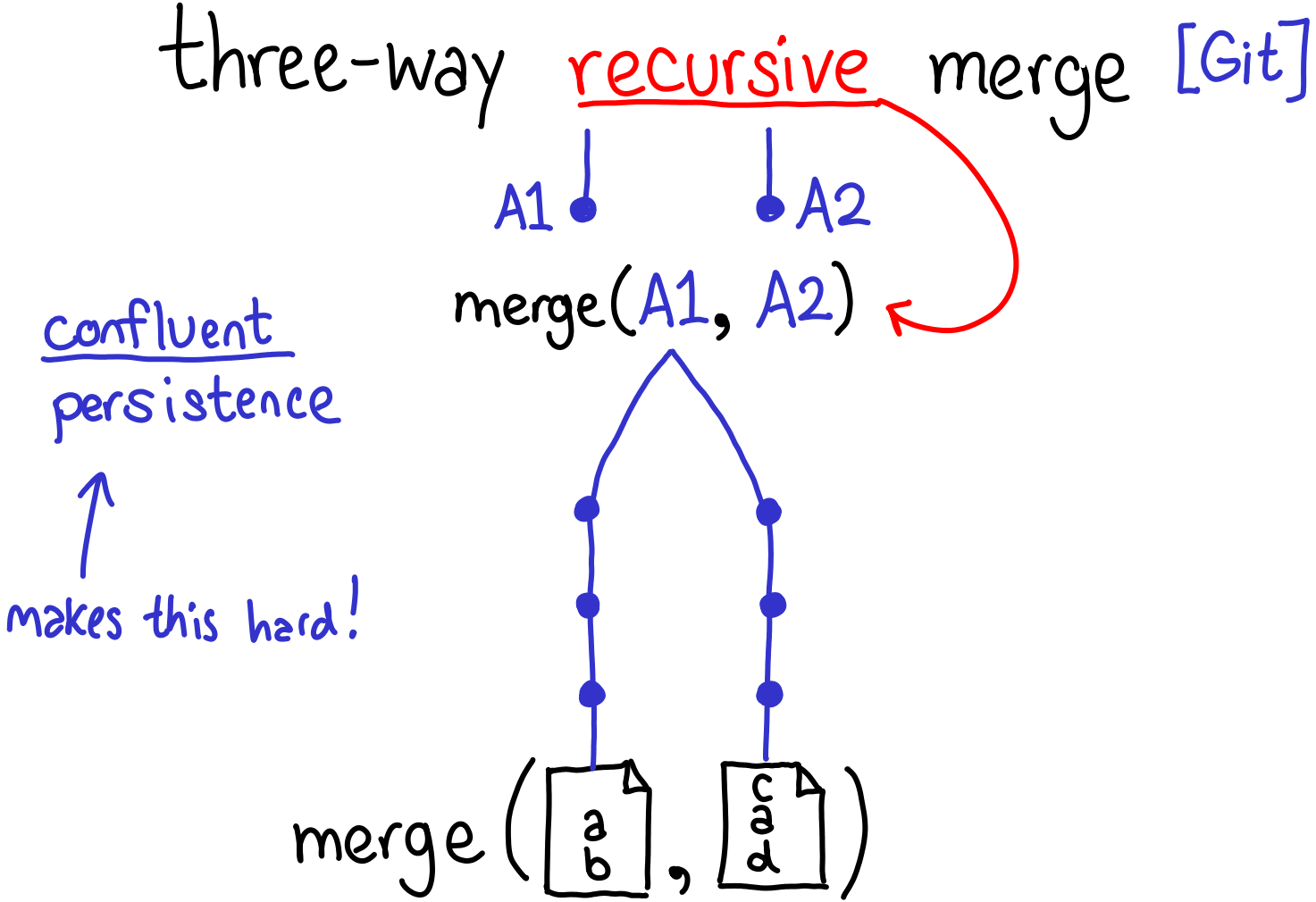
three-way recursive merge [Git]



three-way recursive merge [Git]

confluent persistence




three-way recursive merge [Git]
A1
A2
merge(A1, A2)
confluent persistence
makes this hard!
merge
a
b
c
a
d

three-way recursive merge [Git]

requires $O(nk)$ time and the entire history of a document

(where k is the number of recursive calls — usually $k=1$)

three-way recursive merge [Git]

requires $O(nk)$ time and the entire history of a document

(where k is the number of recursive calls — usually $k=1$)

not too bad in practice, but...

three-way recursive merge [Git]

- it's not asymptotically optimal
  should be cheaper when documents similar
- can be a problem with large documents
  both merge & history is expensive
- may be inflexible
  my way (lcs) or the high way!

blame trees

merge requires $O(\sum_{S \in \text{shared regions}} \log|S| + \text{conflicts})$ time

using a structured document representation

(with $O(\log n)$ update/delete/insert)

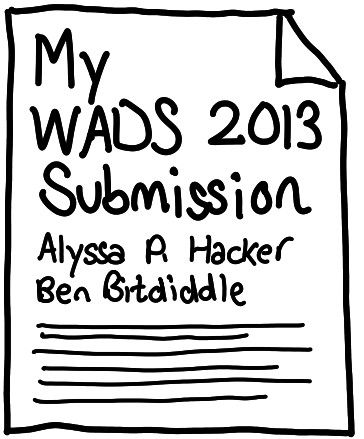
blame trees

merge requires $O(\sum_{S \in \text{shared regions}} \log|S| + \text{conflicts})$ time

using a structured document representation

(with $O(\log n)$ update/delete/insert)

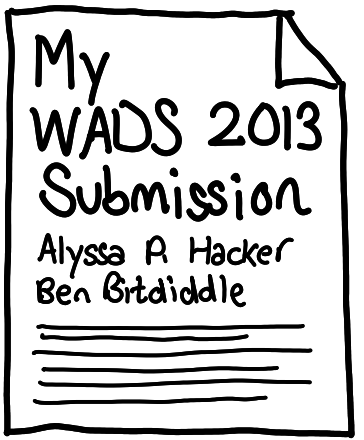
Abstract the Problem

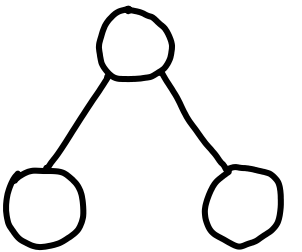


sequence of characters

model of VCS merge

Abstract the Problem





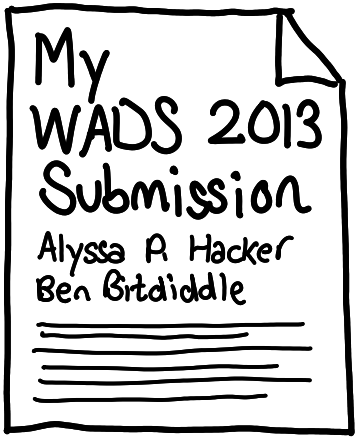
sequence of characters

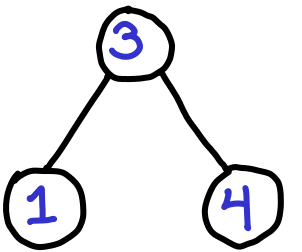
model of VCS merge

sorted associative map

Abstract the Problem

[Dietz '87]



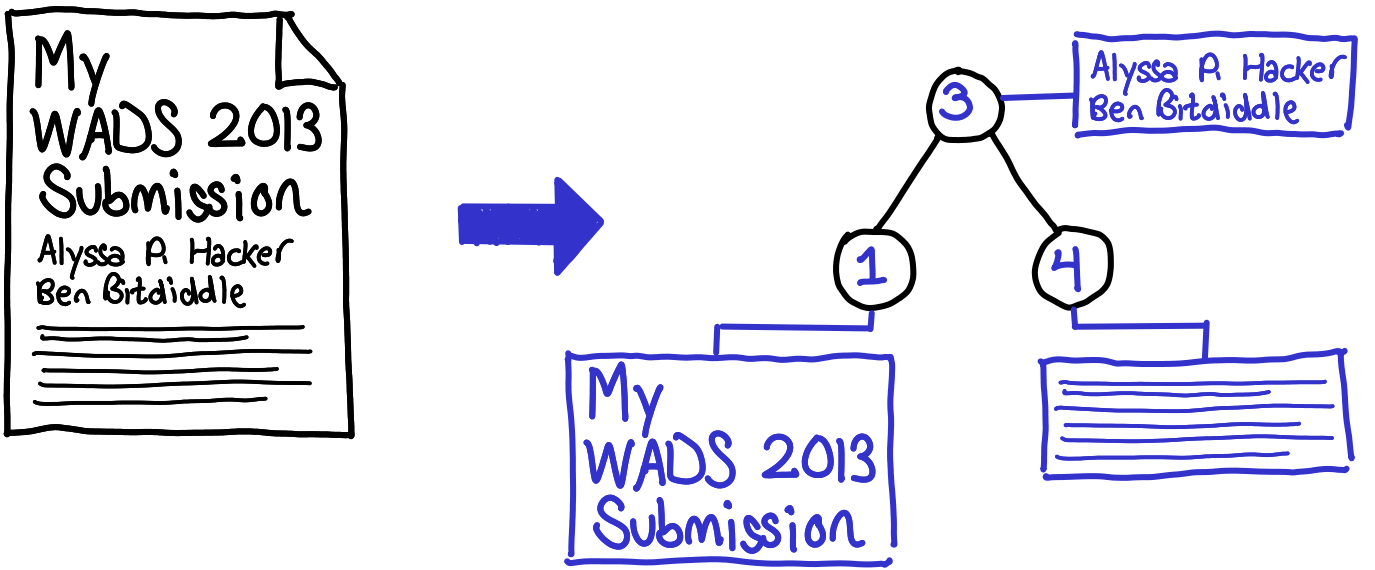


$$between(k_1, k_2) = k_{12} \text{ s.t. } k_1 < k_{12} \wedge k_{12} < k_2$$

sequence of characters
model of VCS merge

sorted associative map
keys: line "number"

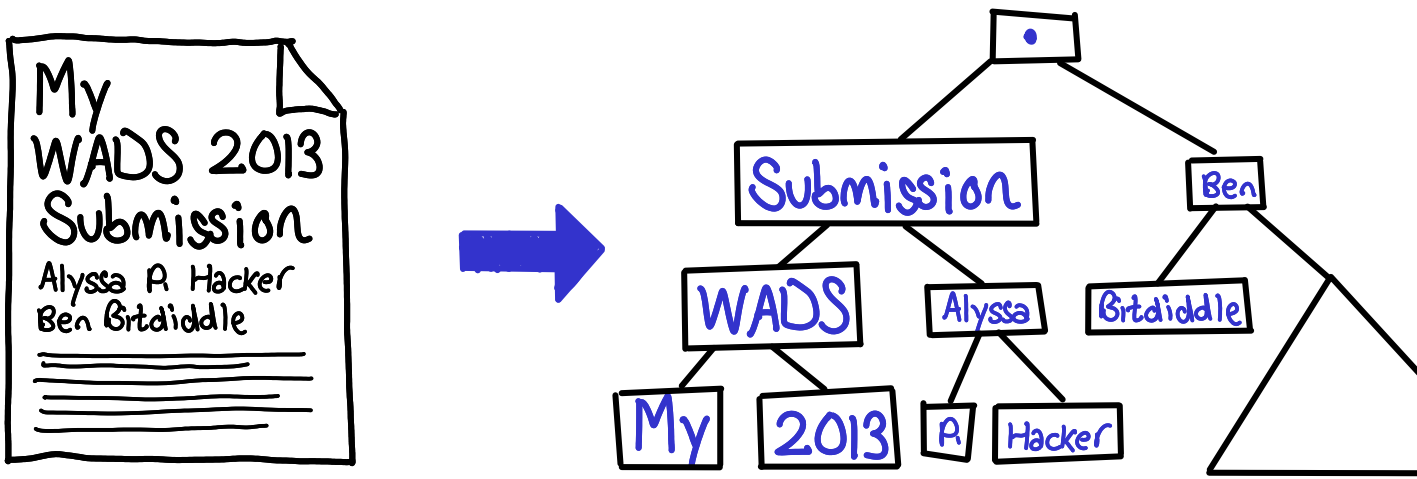
Abstract the Problem



sequence of characters
model of VCS merge

sorted associative map
keys: line "number"
values: fragments of document

Abstract the Problem



sequence of characters
model of VCS merge

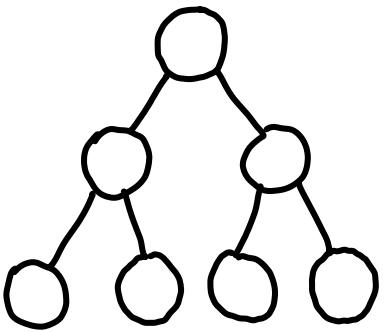
sorted associative map
keys: word "number"
values: fragments of document

Abstract the Problem

functional
data structure

⇓

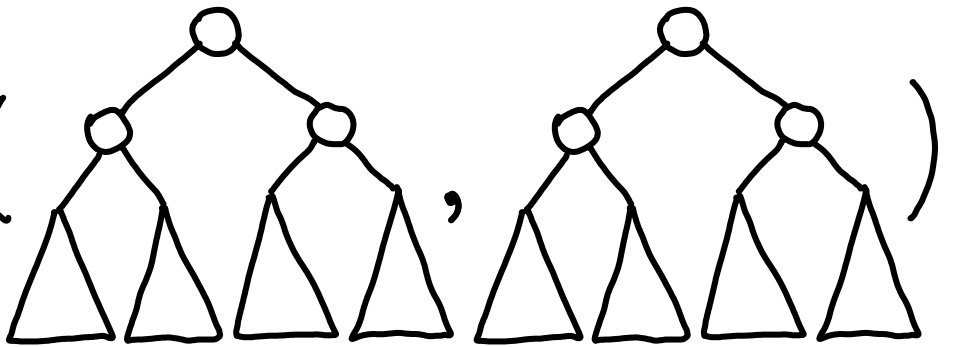
confluently
persistent



```
data Tree = Node  key
                  value
                  left
                  right

          | Leaf
```

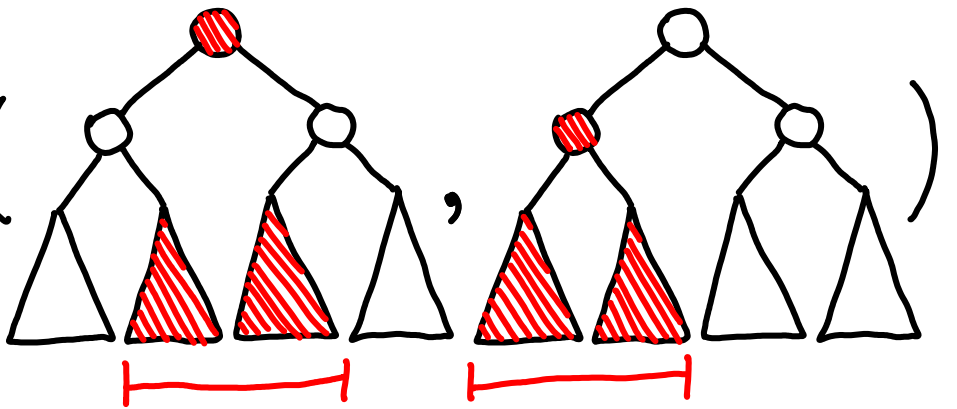
Our Result

merge()

$$O\left(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|\right)$$

shared

Our Result

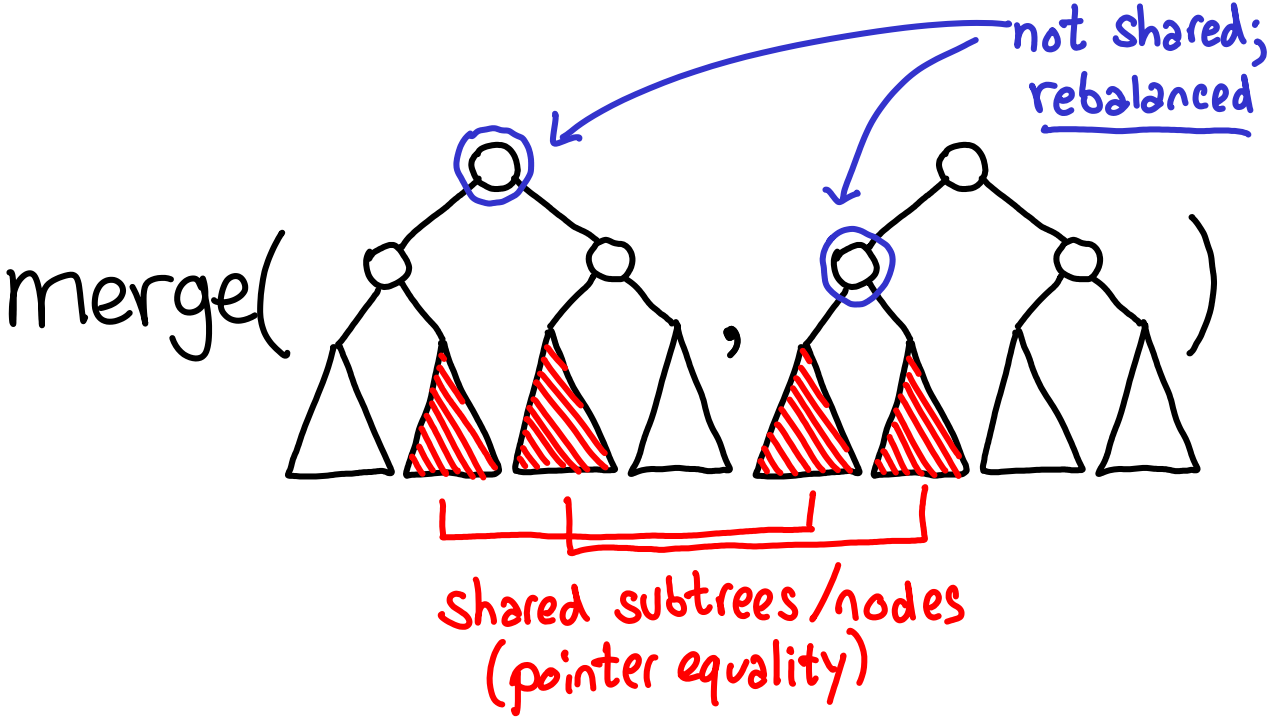
merge()

shared regions
(same keys and values)

$$O\left(R\left(\text{non-shared nodes}\right) + \sum_{S \in \text{shared regions}} \log |S|\right)$$

shared

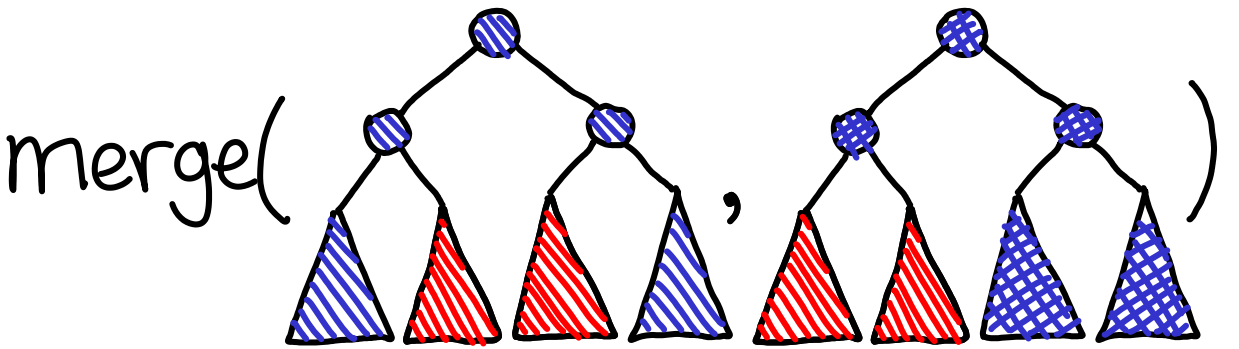
Our Result



$$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|)$$

shared
unshared

Our Result

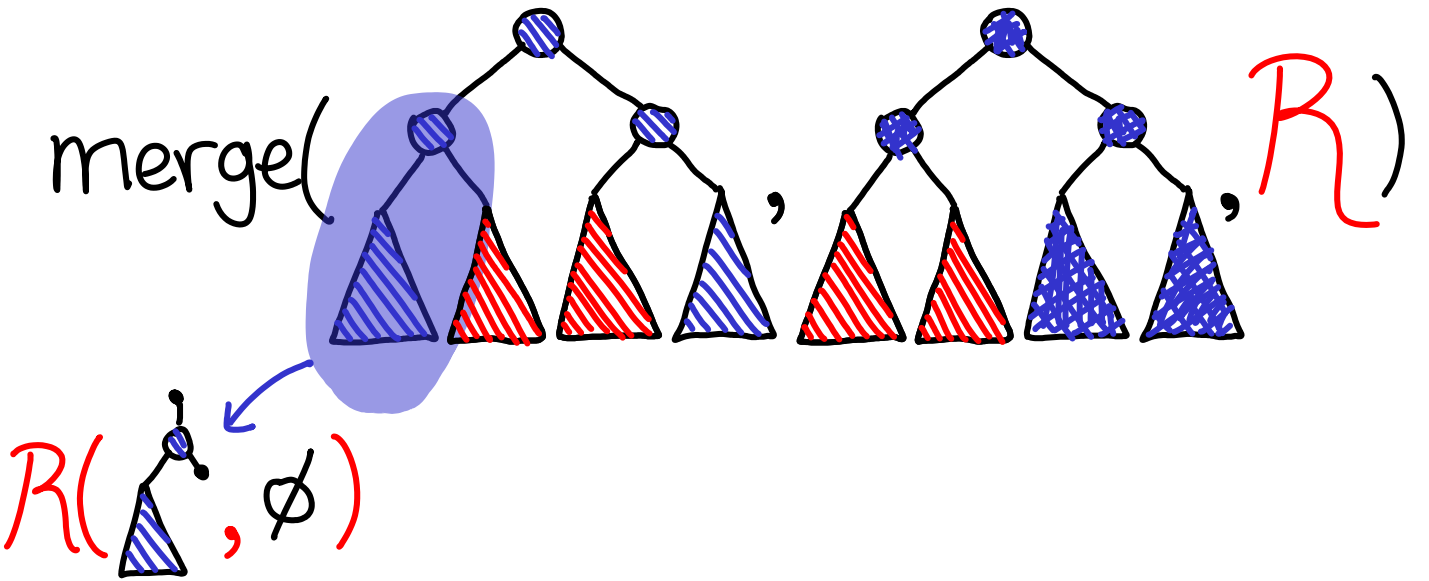


non-shared subtrees/nodes

$$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|)$$

shared
unshared

Our Result

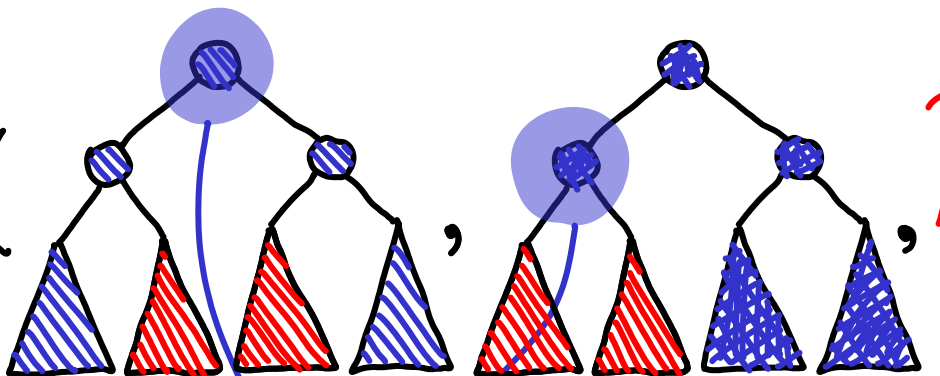


$$O\left(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|\right)$$

shared
unshared

Our Result

merge(, , R)



$R(\ ,\emptyset)$

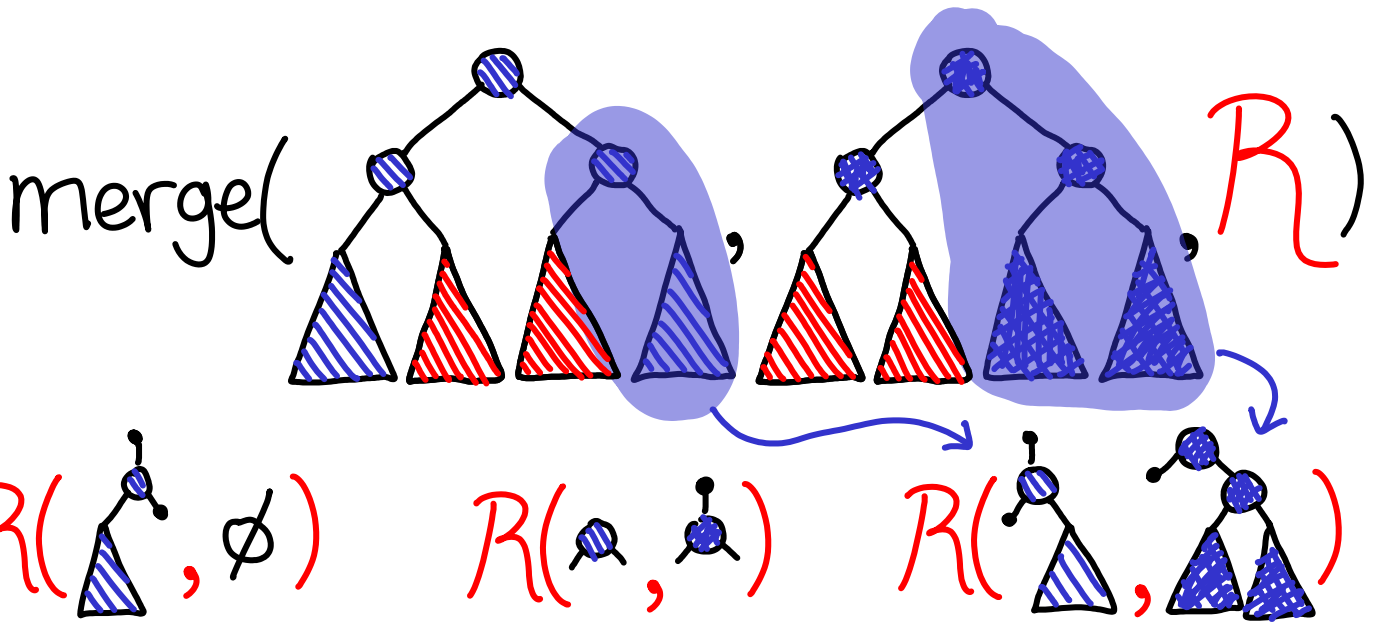
$R(\ ,\ )$

(can optimize)

$$O\left(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|\right)$$

shared
unshared

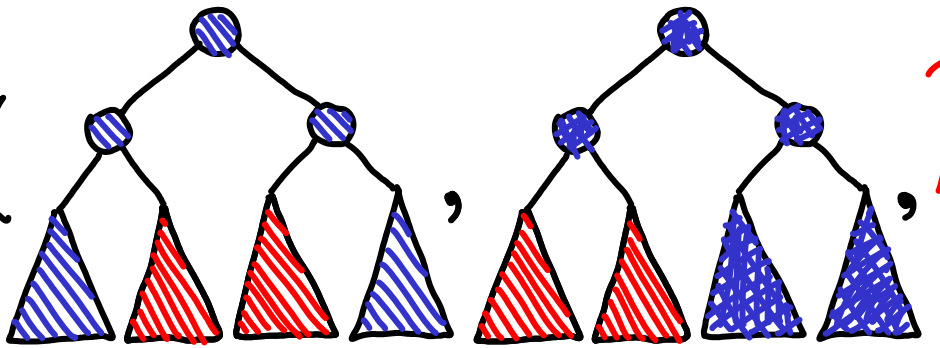
Our Result



$$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|)$$

shared
unshared

Our Result

merge( , R)

Ω(non-shared nodes)

$$O\left(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|\right)$$

blame trees

require $O(\sum_{S \in \text{shared regions}} \log|S| + \text{conflicts})$ time

using a structured document representation

how?

The key idea is to annotate all subtrees with their last update revision (blame). The primary technical difficulty is efficiently identifying shared subtrees.

Blame

```
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   1) \documentcl
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   2)
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   3) \title{Blam
0178feba (Edward Z. Yang  2013-05-03 14:39:26 -0700   4) \author{Eri
fd01474f (Erik Demaine    2013-02-24 21:04:55 -0500   5) \institute{
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   6)
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   7) \usepackage
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   8) \usepackage
eb146b8a (Edward Z. Yang  2013-01-15 20:50:55 -0800   9) \usepackage
4e1a0491 (Erik Demaine    2013-02-24 21:22:45 -0500  10) \usepackage
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800  11)
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800  12) \newcommand
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800  13)
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  14) %\newcomman
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  15) %\algloopde
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  16)
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  17) % Put figur
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  18) \def\textfr
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  19) \def\topfra
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  20) \def\dbltop
```

Blame

content

```
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   1) \documentcl
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   2)
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   3) \title{Blam
0178feba (Edward Z. Yang  2013-05-03 14:39:26 -0700   4) \author{Eri
fd01474f (Erik Demaine    2013-02-24 21:04:55 -0500   5) \institute{
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   6)
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   7) \usepackage
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   8) \usepackage
eb146b8a (Edward Z. Yang  2013-01-15 20:50:55 -0800   9) \usepackage
4e1a0491 (Erik Demaine    2013-02-24 21:22:45 -0500  10) \usepackage
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800  11)
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800  12) \newcommand
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800  13)
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  14) %\newcomman
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  15) %\algloopde
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  16)
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  17) % Put figur
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  18) \def\textfr
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  19) \def\topfra
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  20) \def\dbltop
```

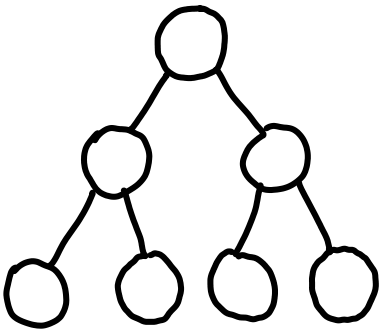
revision annotation

Blame

```
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   1) \documentcl
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   2)
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   3) \title{Blam
0178feba (Edward Z. Yang  2013-05-03 14:39:26 -0700   4) \author{Eri
fd01474f (Erik Demaine    2013-02-24 21:04:55 -0500   5) \institute{
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   6)
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800   7) \usepackage
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800   8) \usepackage
eb146b8a (Edward Z. Yang  2013-01-15 20:50:55 -0800   9) \usepackage
4e1a0491 (Erik Demaine    2013-02-24 21:22:45 -0500  10) \usepackage
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800  11)
d1a89f24 (Edward Z. Yang  2012-11-30 12:16:35 -0800  12) \newcommand
^1e00c69 (Edward Z. Yang  2012-11-28 10:41:07 -0800  13)
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  14) %\newcomman
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  15) %\algloopde
7b24b512 (Edward Z. Yang  2012-12-13 13:04:13 -0800  16)
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  17) % Put figur
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  18) \def\textfr
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  19) \def\topfra
38ec8c67 (Erik Demaine    2013-02-24 22:35:40 -0500  20) \def\dbltop
```

Idea: Record Blame

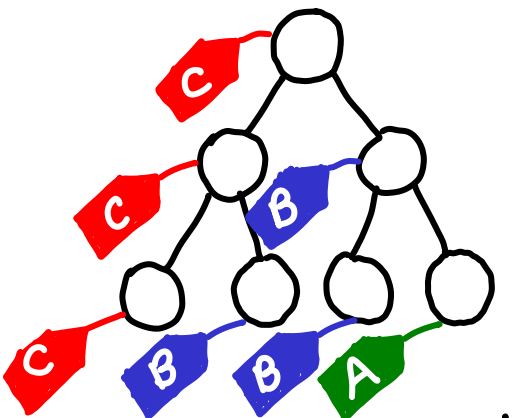
(don't compute it!)



```
data Tree = Node  key
                  value
                  left
                  right
          | Leaf
```

Idea: Record Blame

(don't compute it!)



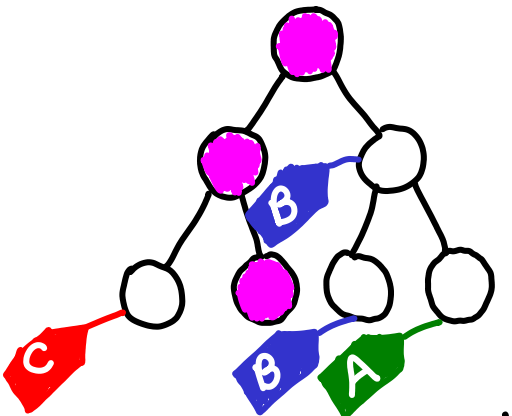
data Tree = Node key
value
revision
left
right
| Leaf

Idea: Record Blame

(don't compute it!)

path copying

[Oka'99]



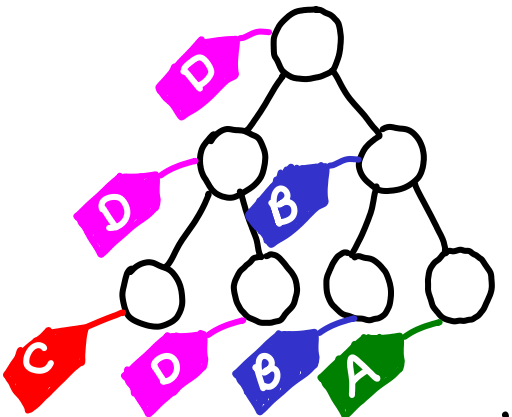
data Tree = Node key value revision left right | Leaf

Idea: Record Blame

(don't compute it!)

path copying

[Oka'99]



data Tree = Node key value revision left right | Leaf

Key invariant

$$key_1 = key_2 \land rev_1 = rev_2 \iff tree_1 = tree_2$$

Algorithm

1. Traverse tree, identifying shared regions.
2. Split tree at region boundaries.
3. Perform conflict resolution on unshared regions.
4. Merge trees back together.

Algorithm

1. Traverse tree, identifying shared regions.
2. Split tree at region boundaries.
3. Perform conflict resolution on unshared regions.
4. Merge trees back together.

$\sum_{\text{seregions}} \log|s|$

$\sum_{\text{seregions}} \log|s|$

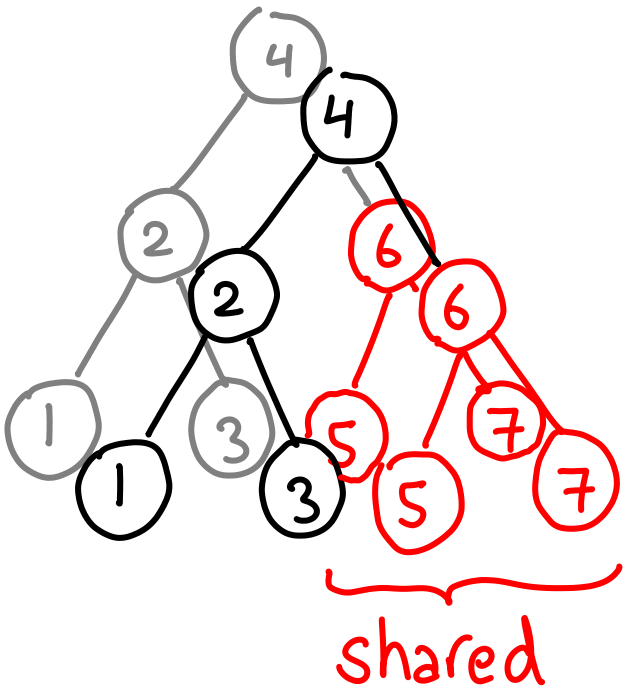
Algorithm

1. Traverse tree, identifying shared regions.
2. Split tree at region boundaries. $\sum_{\text{seregions}} \log|s|$
3. Perform conflict resolution on unshared regions. $\mathcal{R}(\text{non-shared})$
4. Merge trees back together. $\sum_{\text{seregions}} \log|s|$

Algorithm

1. Traverse tree, identifying shared regions.	???
2. Split tree at region boundaries.	$\sum_{\text{seregions}} \log\|s\|$
3. Perform conflict resolution on unshared regions.	$\mathcal{R}(\text{non-shared})$
4. Merge trees back together.	$\sum_{\text{seregions}} \log\|s\|$

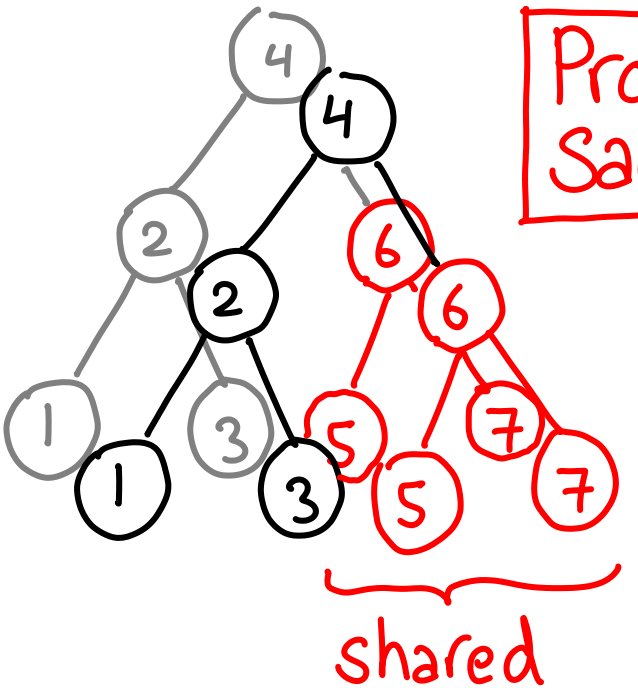
Traversal: Solution 1 (naïve)



XXXXGGG
YYYYGGG

$$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|)$$

Traversal: Solution 1 (naïve)



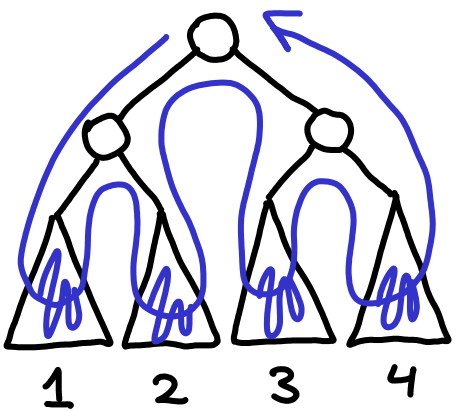
Problem: Requires Same Structure

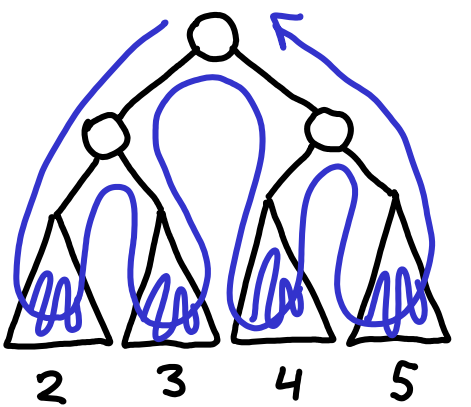
XXXXGGG
YYYYGGG

$$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log |S|)$$

Traversal: Solution 2 (in-order)

f(left)
visit
f(right)



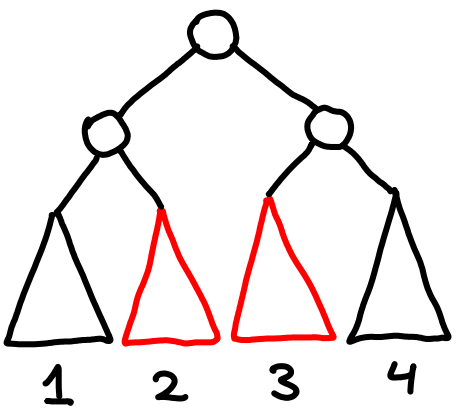


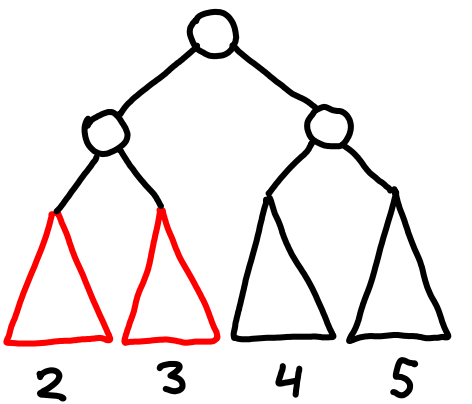
$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} |S|)$

Traversal: Solution 2 (in-order)

Short-circuited

```
f(left)
visit
if(!shared)
    f(right)
```

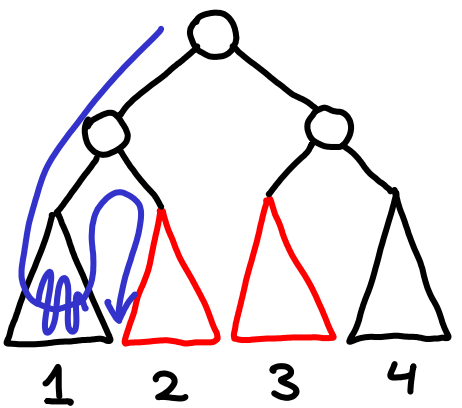


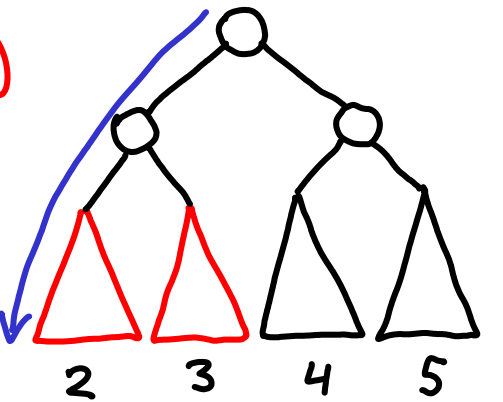


Traversal: Solution 2 (in-order)

Short-circuited

```
f(left)
visit
if(!shared)
    f(right)
```

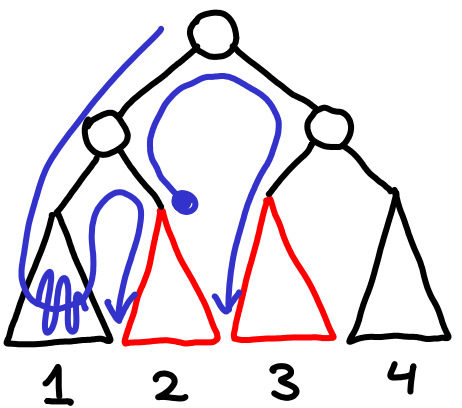


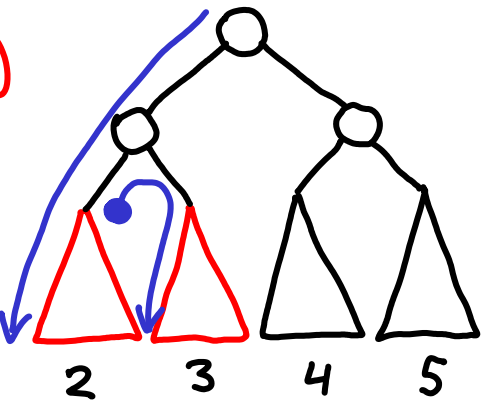


Traversal: Solution 2 (in-order)

Short-circuited

```
f(left)
visit
if(!shared)
    f(right)
```

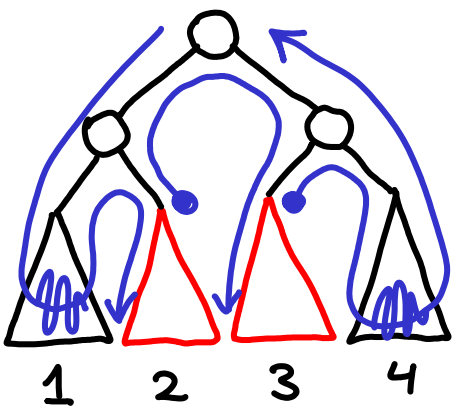


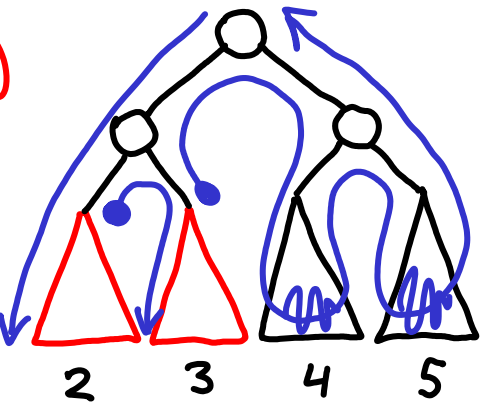


Traversal: Solution 2 (in-order)

Short-circuited

```
f(left)
visit
if(!shared)
    f(right)
```

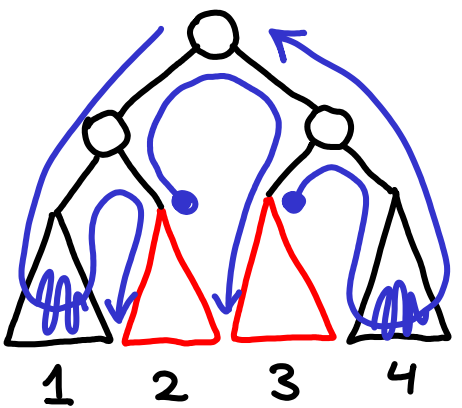


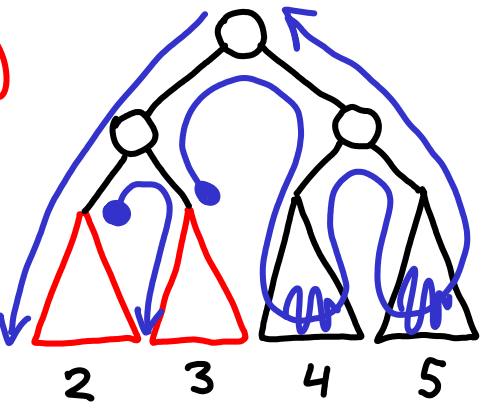


Traversal: Solution 2 (in-order)

Short-circuited

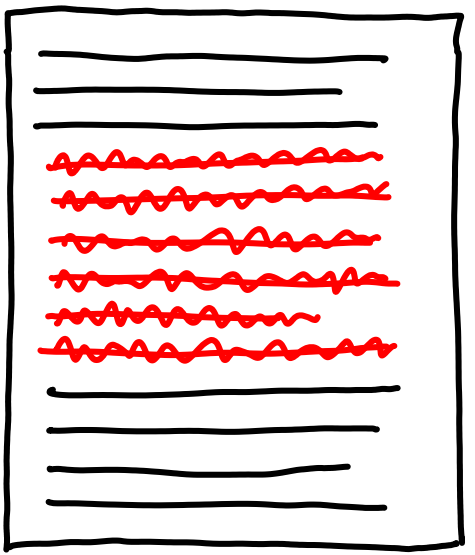
```
f(left)
visit
if(!shared)
    f(right)
```

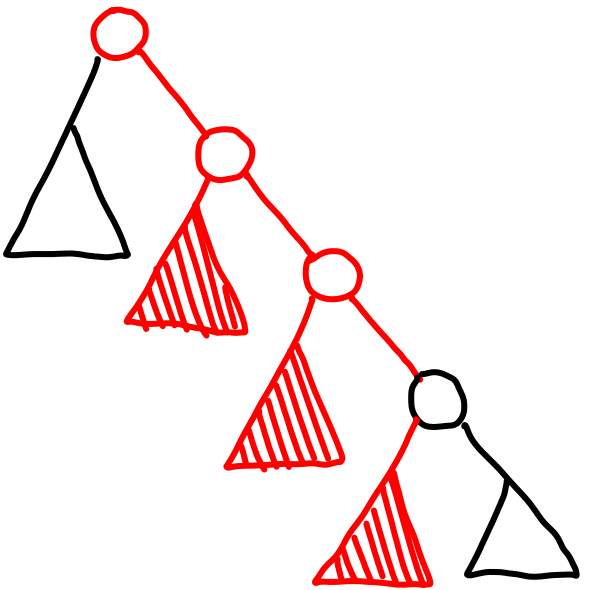




$$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared regions}} \log^2 |S|)$$

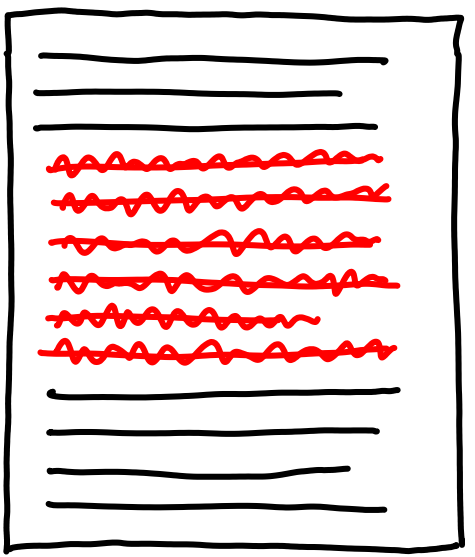
Why $\log^2(|S|)$?

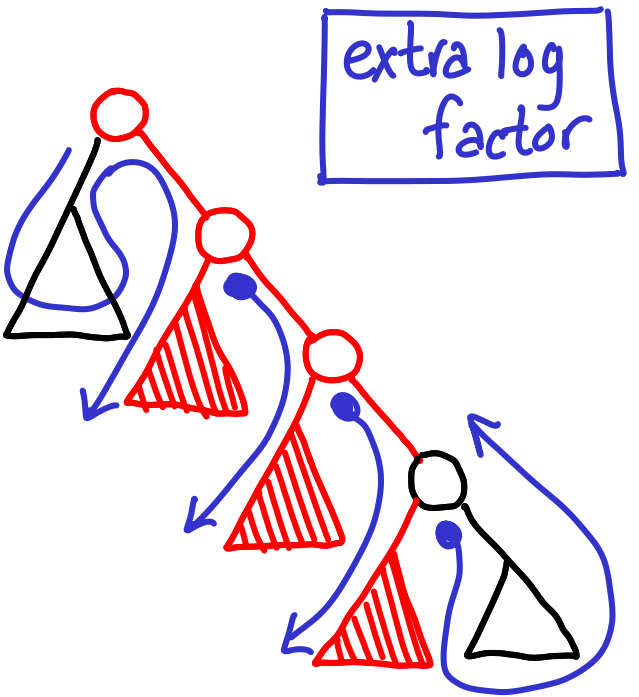




shared subregion $\neq$ shared subtree

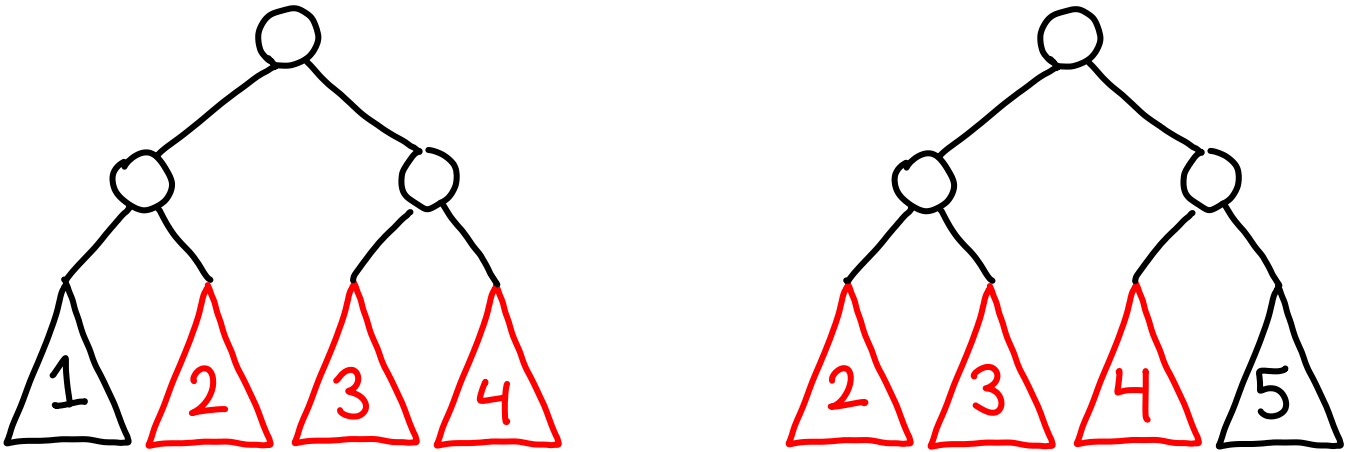
Why $\log^2(|S|)$?





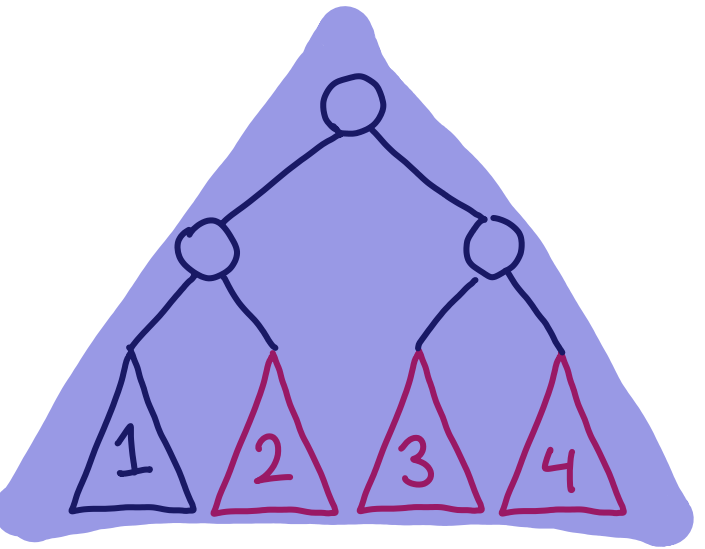
shared subregion $\neq$ shared subtree

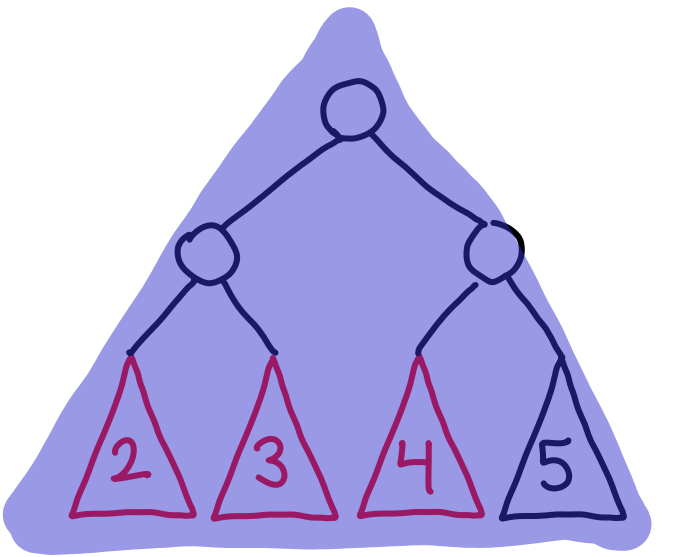
Traversal: Solution 3 (level-order)

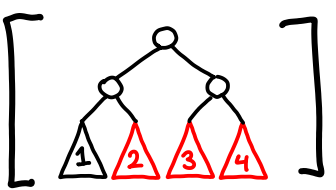


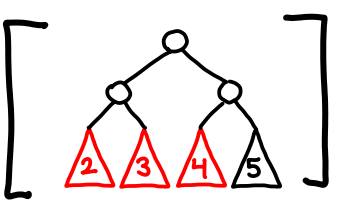
traverse by level

Traversal: Solution 3 (level-order)

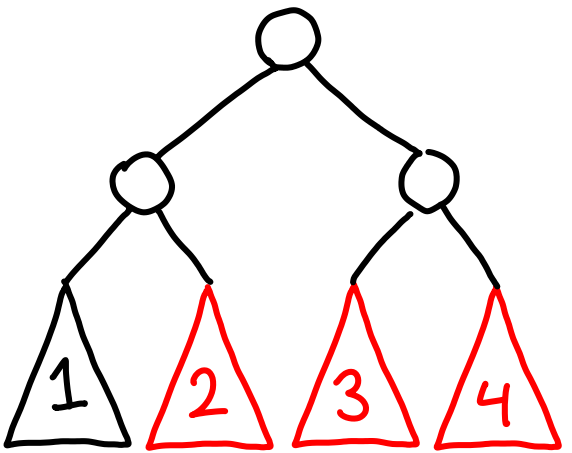


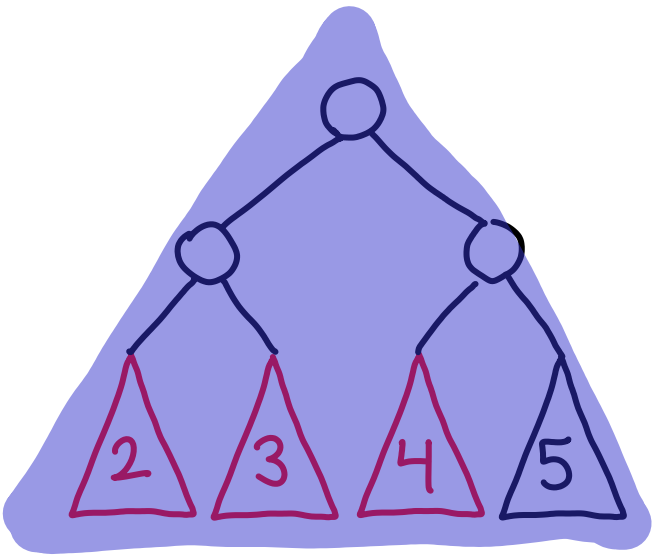


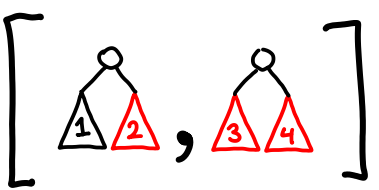


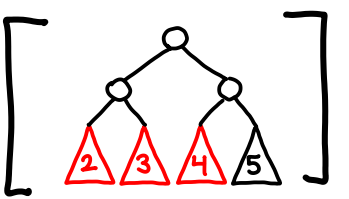


Traversal: Solution 3 (level-order)

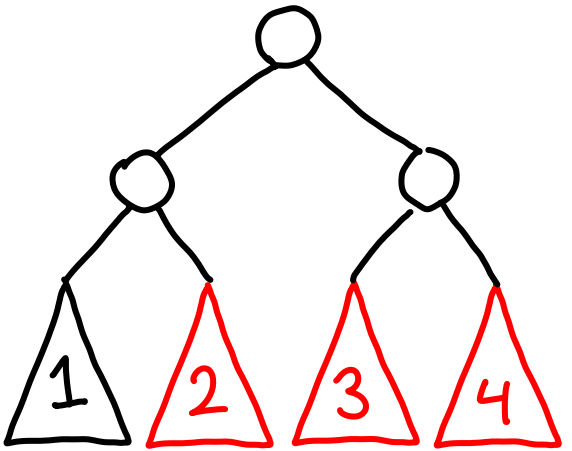


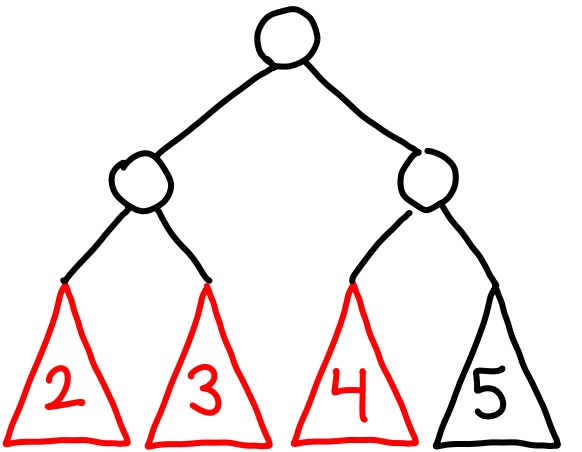


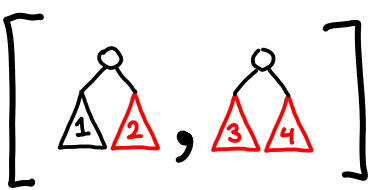


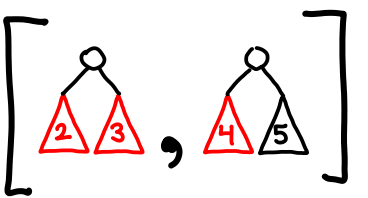


Traversal: Solution 3 (level-order)

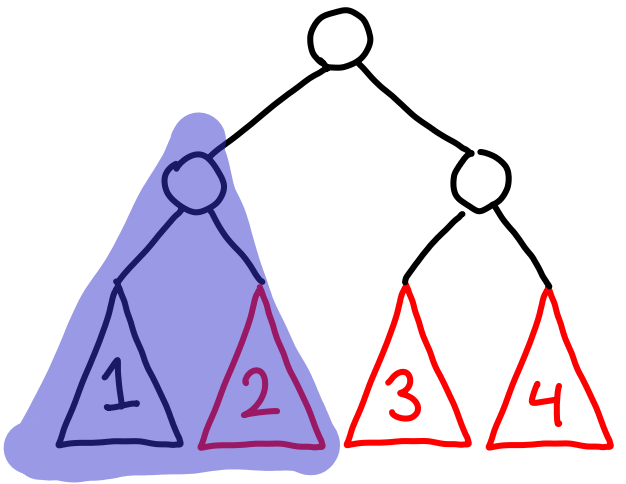


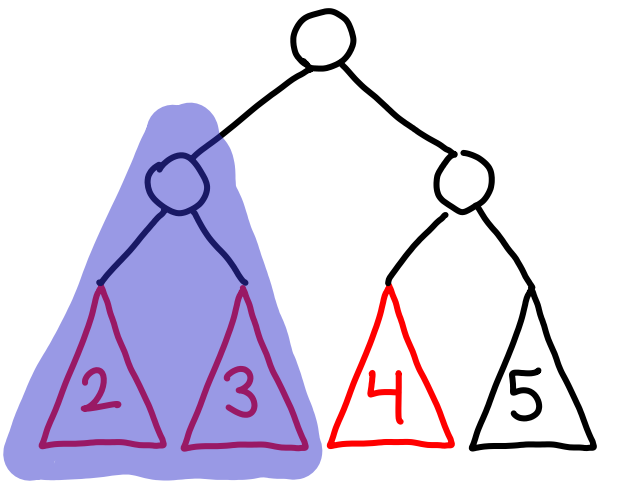


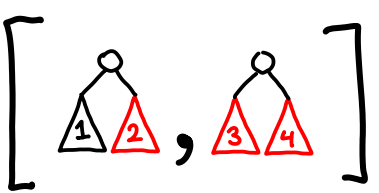


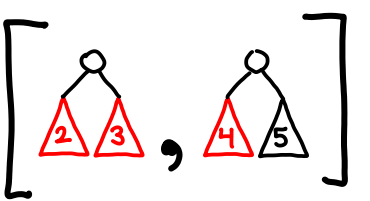


Traversal: Solution 3 (level-order)

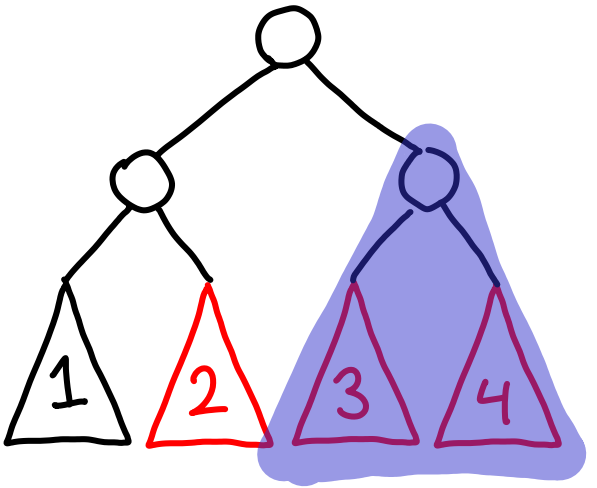


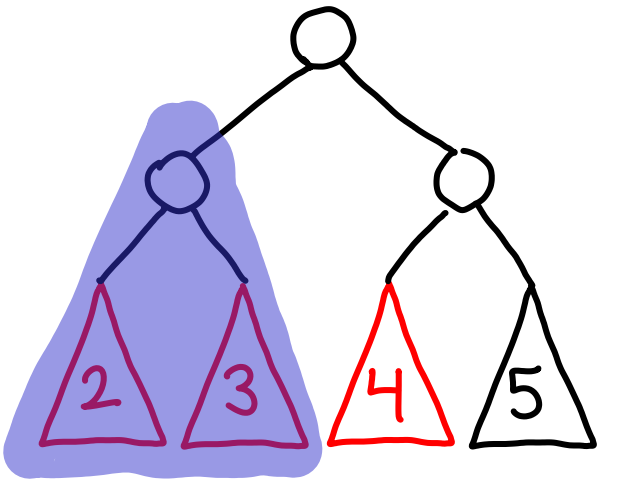


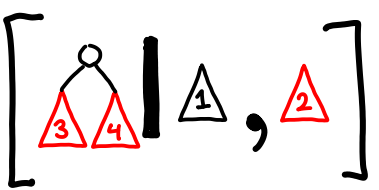


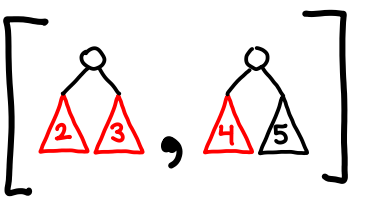


Traversal: Solution 3 (level-order)

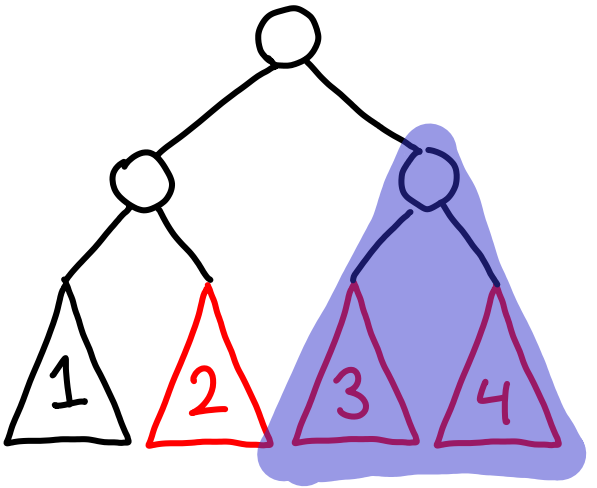


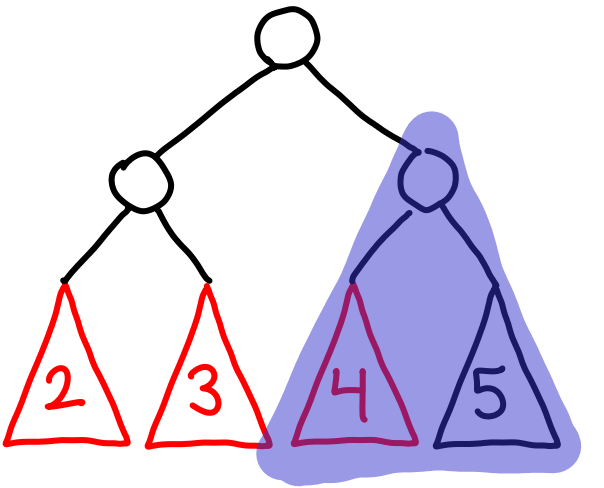


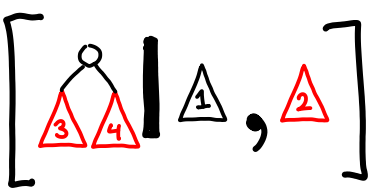


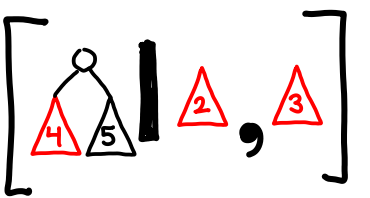


Traversal: Solution 3 (level-order)

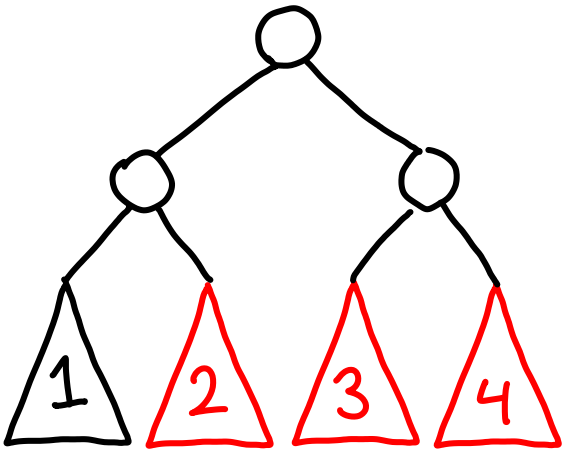


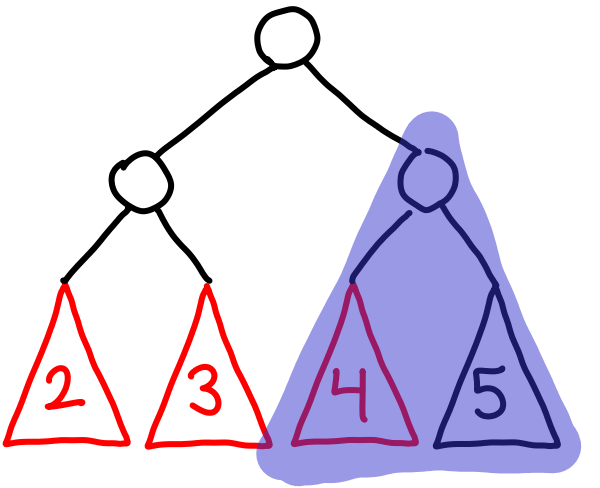


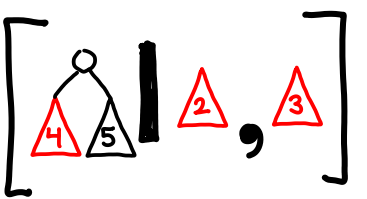




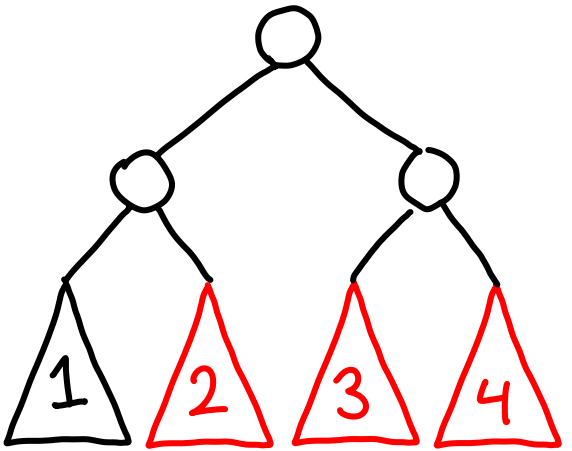
Traversal: Solution 3 (level-order)

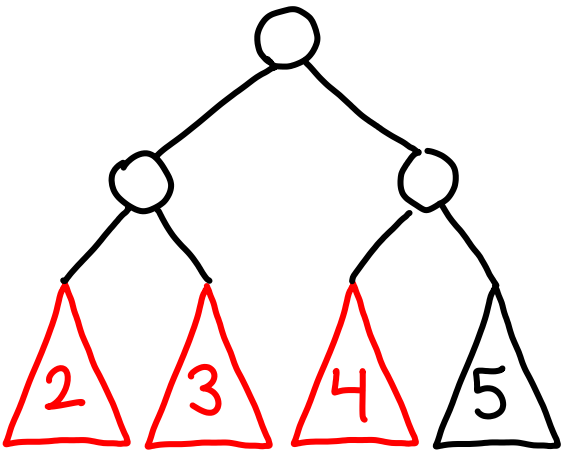


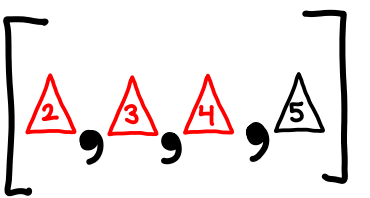




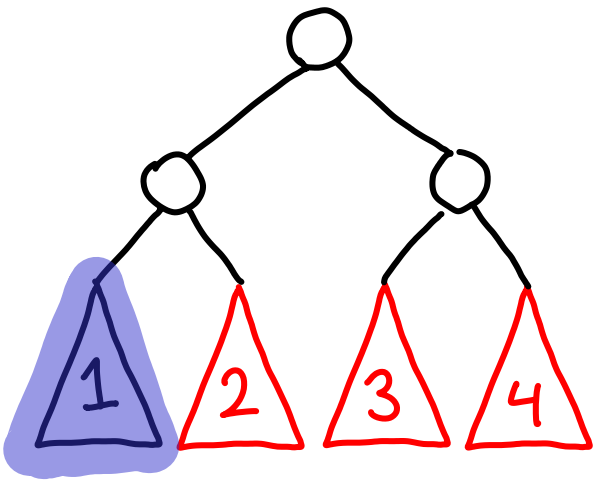
Traversal: Solution 3 (level-order)

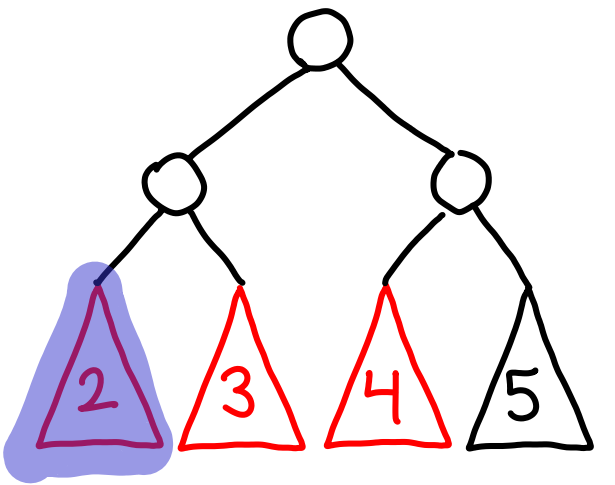


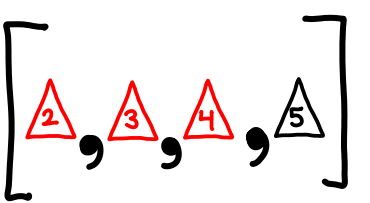




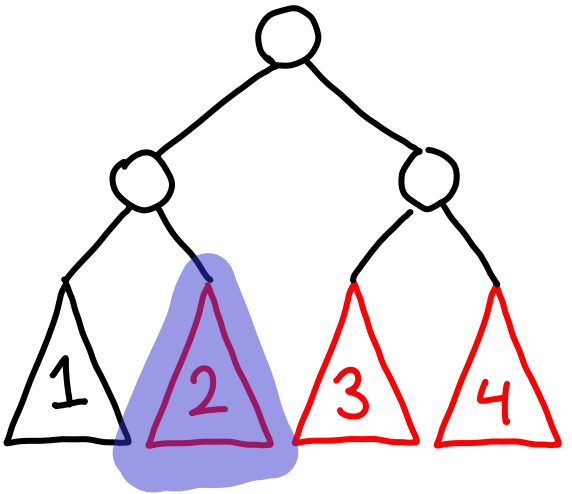
Traversal: Solution 3 (level-order)

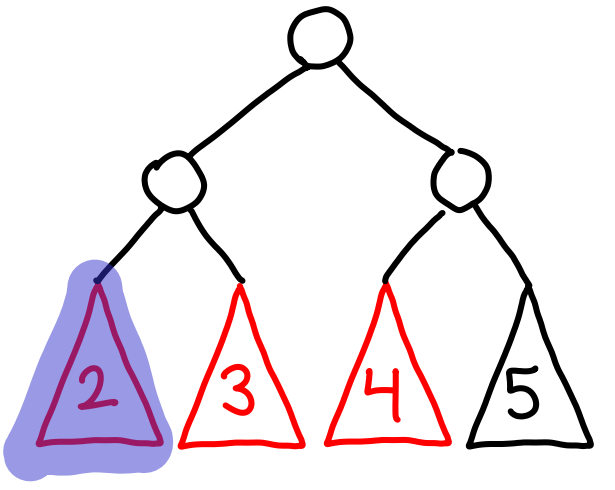


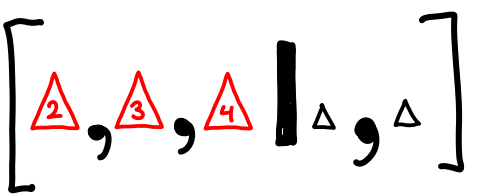


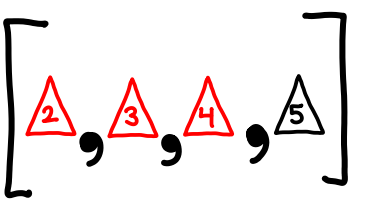


Traversal: Solution 3 (level-order)

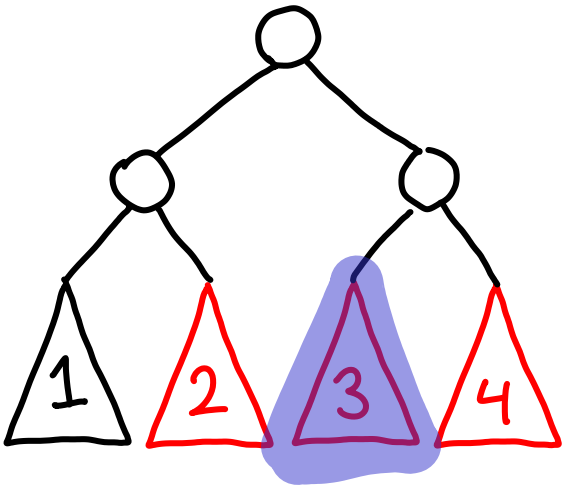


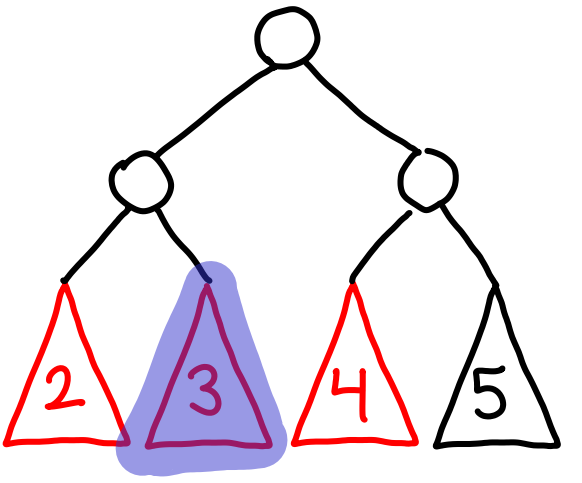


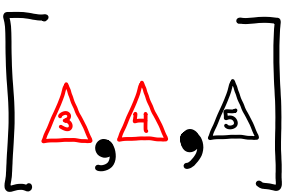




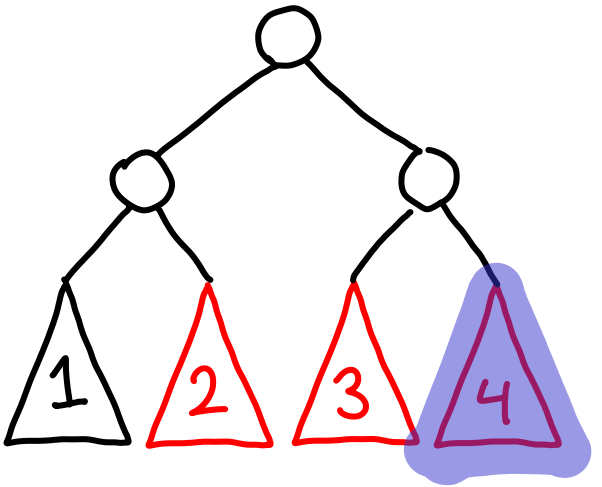
Traversal: Solution 3 (level-order)

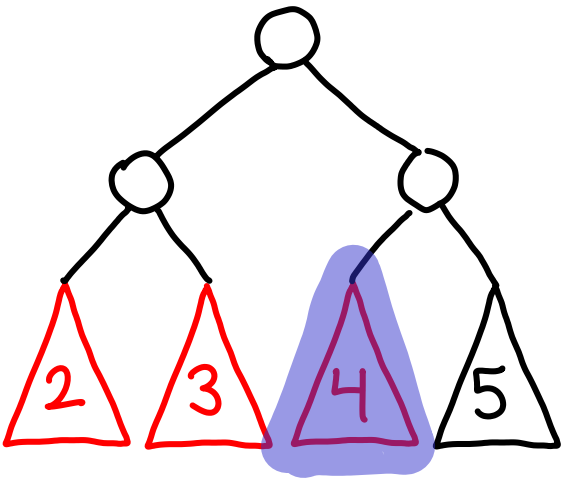






Traversal: Solution 3 (level-order)

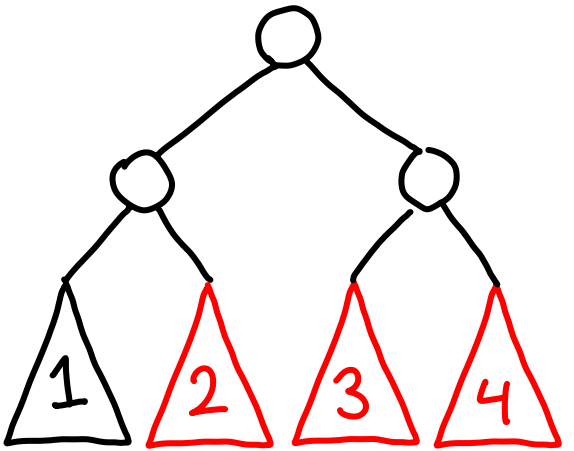


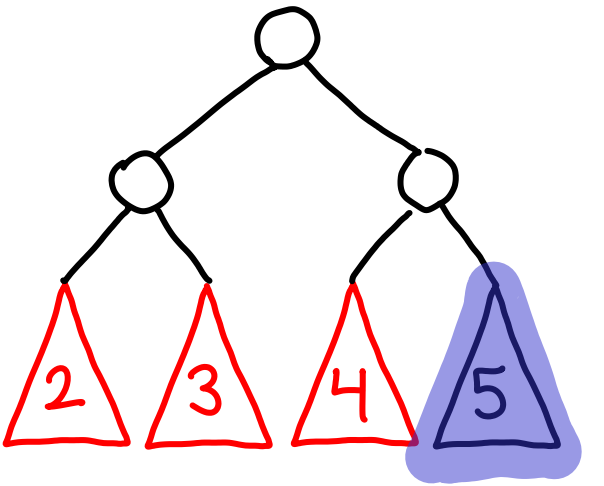


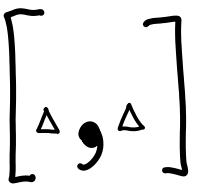
[4 | Δ , Δ]

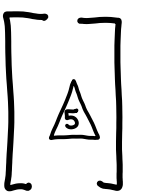
[4 , 5]

Traversal: Solution 3 (level-order)

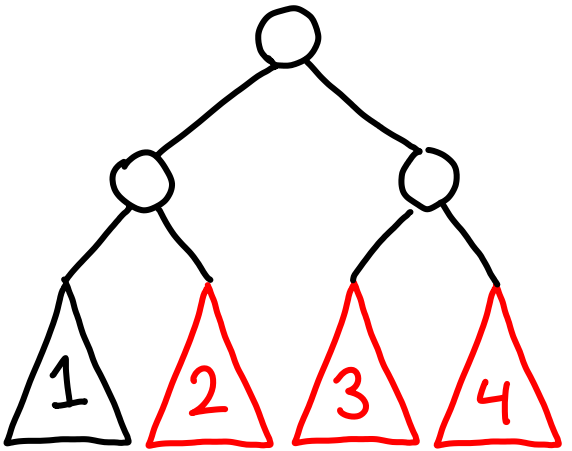


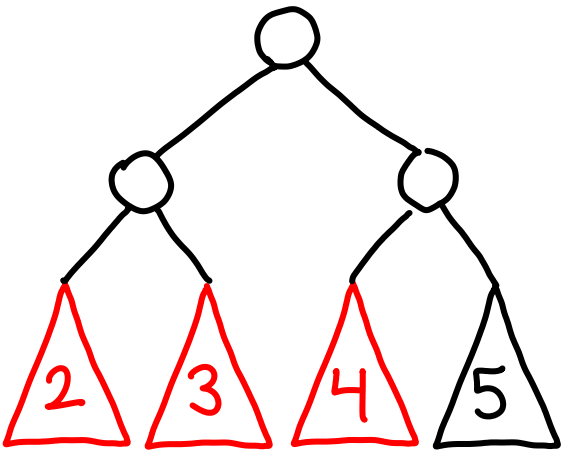


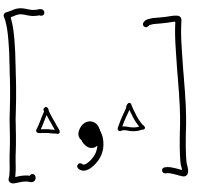


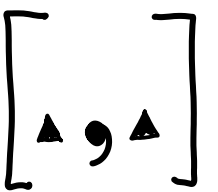


Traversal: Solution 3 (level-order)

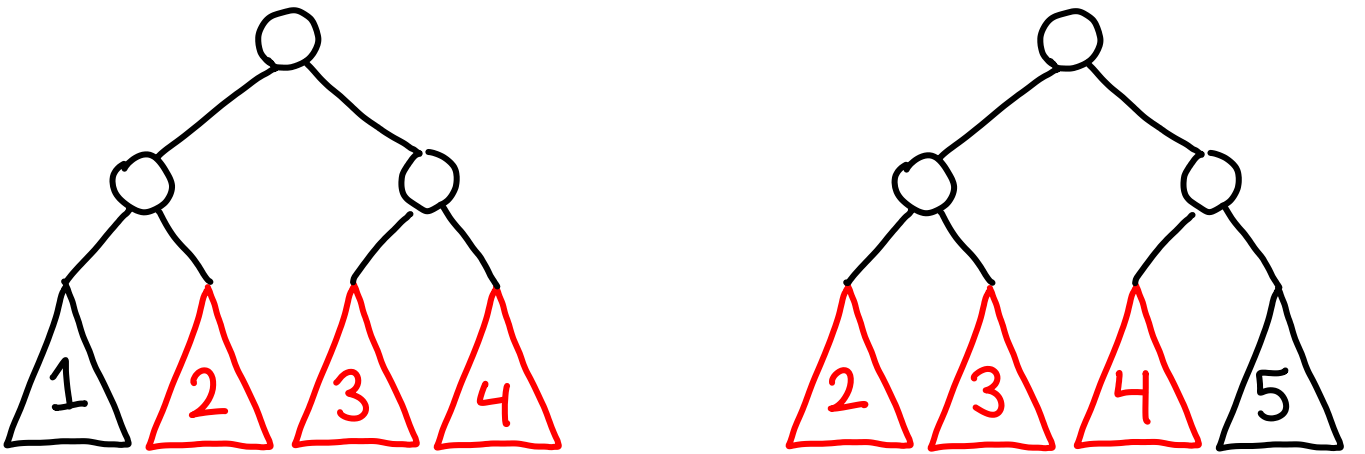








Traversal: Solution 3 (level-order)



$O(R(\text{non-shared nodes}) + \sum_{S \in \text{shared subtrees}} \log |S|)$ ✓

Conclusion

- Level-order works for red-black trees
- Bound matches optimal bound set by adaptive merging algorithms [CLP'93, DLM'00]
- Pesky constant factors.