

Range-Based Set Reconciliation

Aljoscha Meyer

Set Reconciliation

- set union over a network
- between (exactly) two machines
- unstructured data
- no shared state or history

Trivial Reconciliation



Sam Gwilym

Model and Analysis

- Alfie and Betty talk over a network
- reliable communication, rounds of unit length, unlimited bandwidth
- probabilistic solutions

Model and Analysis

- Alfie and Betty talk over a network
- reliable communication, rounds of unit length, unlimited bandwidth
- probabilistic solutions
- n : size of the union
- n_{\triangle} : size of the symmetric difference

- roundtrips
- communicated bytes
- computation time
- computation space

Traditional Approaches

- obtain approximation of n_Δ
- compute message of size $\mathcal{O}(n_\Delta)$ by iterating over all n items
- exchange messages
- recover symmetric difference from those messages using at least $\mathcal{O}(n_\Delta)$ time and memory

Peer-to-peer systems:

- iterating over local set every time infeasible
- loading local set into memory infeasible
- some peers are out to get us

Peer-to-peer systems:

- iterating over local set every time infeasible
- loading local set into memory infeasible
- some peers are out to get us

⇒ traditional approaches don't work

Reducing Computational Complexities

- allow for logarithmic number of rounds

Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets

Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets
 - if equal, we are done

Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets
 - if equal, we are done
- divide-and-conquer

Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets
 - if equal, we are done
- divide-and-conquer
 - chose predicate for partitioning

Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets
 - if equal, we are done
- divide-and-conquer
 - chose predicate for partitioning
 - restrict to simple predicates (split into ranges)

Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets
 - if equal, we are done
- divide-and-conquer
 - chose predicate for partitioning
 - restrict to simple predicates (split into ranges)
 - peers alternatingly choose splits that are optimal for themselves

Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets
 - if equal, we are done
- divide-and-conquer
 - chose predicate for partitioning
 - restrict to simple predicates (split into ranges)
 - peers alternatingly choose splits that are optimal for themselves
- binary-search for differences

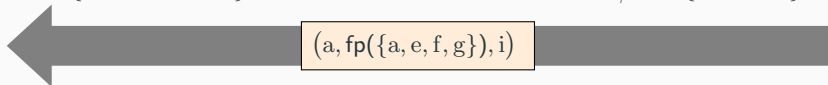
Reducing Computational Complexities

- allow for logarithmic number of rounds
- compare fingerprints of local sets
 - if equal, we are done
- divide-and-conquer
 - chose predicate for partitioning
 - restrict to simple predicates (split into ranges)
 - peers alternatingly choose splits that are optimal for themselves
- binary-search for differences
 - collaboratively
 - over the wire
 - in parallel
 - ok, the analogy is not perfect

Range-Based Set Reconciliation

$$X_\alpha := \{b, c, d, e, f, h\}$$

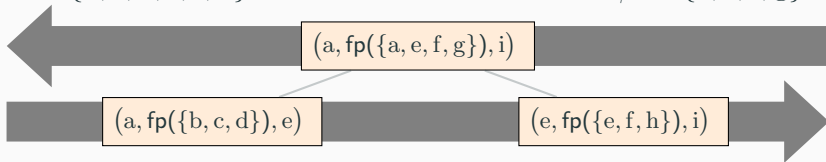
$$X_\beta := \{a, e, f, g\}$$



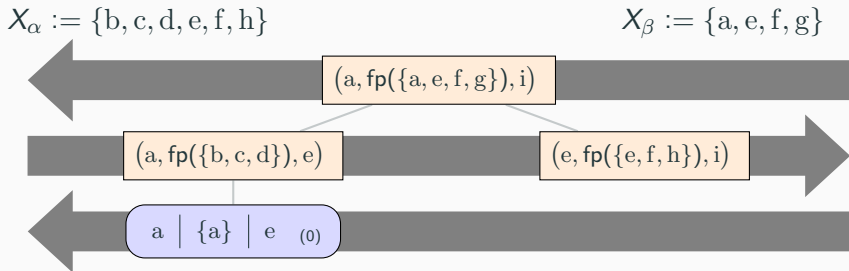
Range-Based Set Reconciliation

$X_\alpha := \{b, c, d, e, f, h\}$

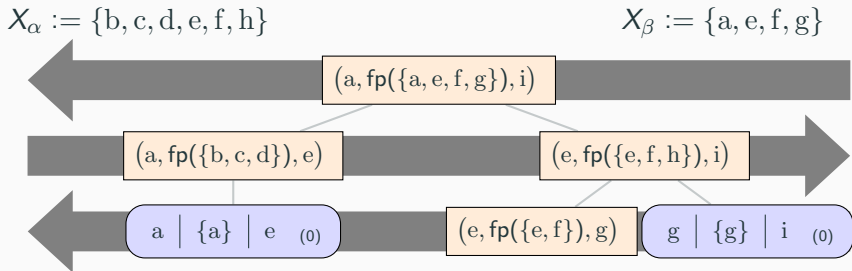
$X_\beta := \{a, e, f, g\}$



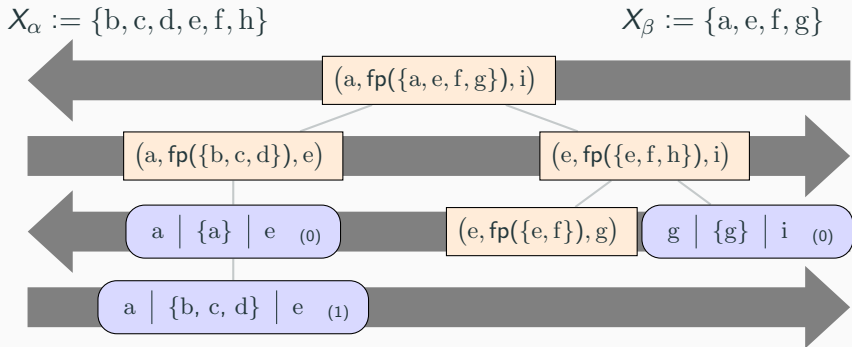
Range-Based Set Reconciliation



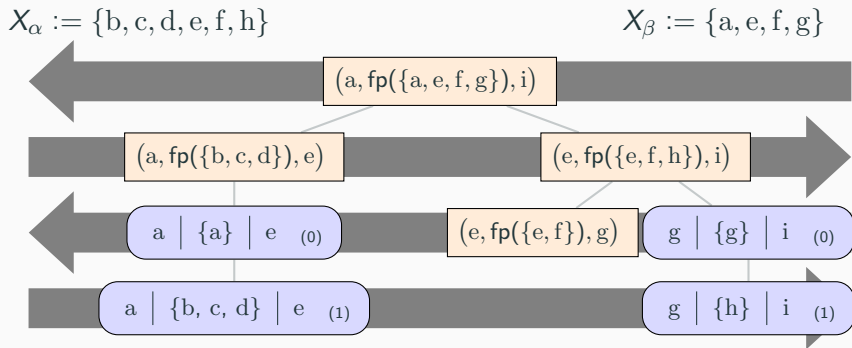
Range-Based Set Reconciliation



Range-Based Set Reconciliation



Range-Based Set Reconciliation



Some Nice Properties

- reasonably efficient: $\mathcal{O}(\min(n_{\Delta} \cdot \log(n), n))$ bytes communication, $\mathcal{O}(1)$ working memory
- can tune bandwidth vs roundtrip minimization
- arbitrary recursion anchor protocols
- arbitrary partition techniques

Some Nice Properties

- reasonably efficient: $\mathcal{O}(\min(n_{\Delta} \cdot \log(n), n))$ bytes communication, $\mathcal{O}(1)$ working memory
- can tune bandwidth vs roundtrip minimization
- arbitrary recursion anchor protocols
- arbitrary partition techniques
- but: linear computation times

Reducing Computation Times

- $\binom{n}{2}$ possible subranges \implies cannot precompute all fingerprints

Reducing Computation Times

- $\binom{n}{2}$ possible subranges \implies cannot precompute all fingerprints
- free space budget of $\mathcal{O}(n)$
- labeled trees!

Aurolat, Alex, and François Taïani. "Merkle search trees: Efficient state-based CRDTs in open networks." 2019 38th Symposium on Reliable Distributed Systems (SRDS). IEEE, 2019.

- define a unique tree shape for every set
- use that shape for a Merkle tree of your set
- only exchange fingerprints for ranges that correspond to subtrees

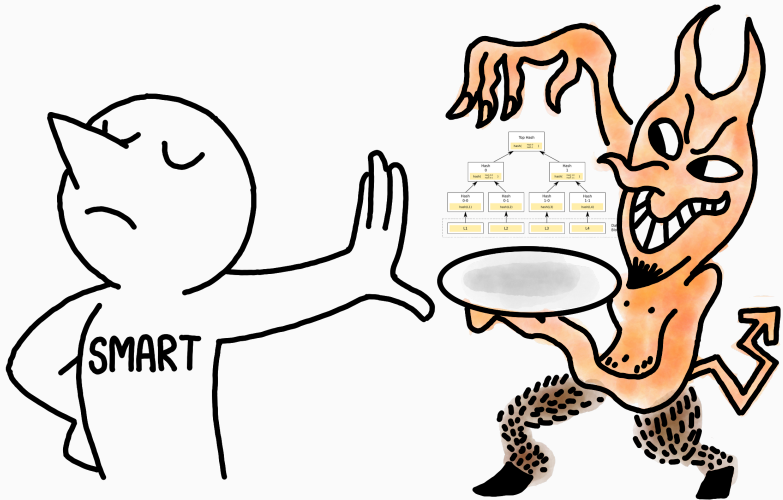
Aurolat, Alex, and François Taïani. "Merkle search trees: Efficient state-based CRDTs in open networks." 2019 38th Symposium on Reliable Distributed Systems (SRDS). IEEE, 2019.

- define a unique tree shape for every set
- use that shape for a Merkle tree of your set
- only exchange fingerprints for ranges that correspond to subtrees

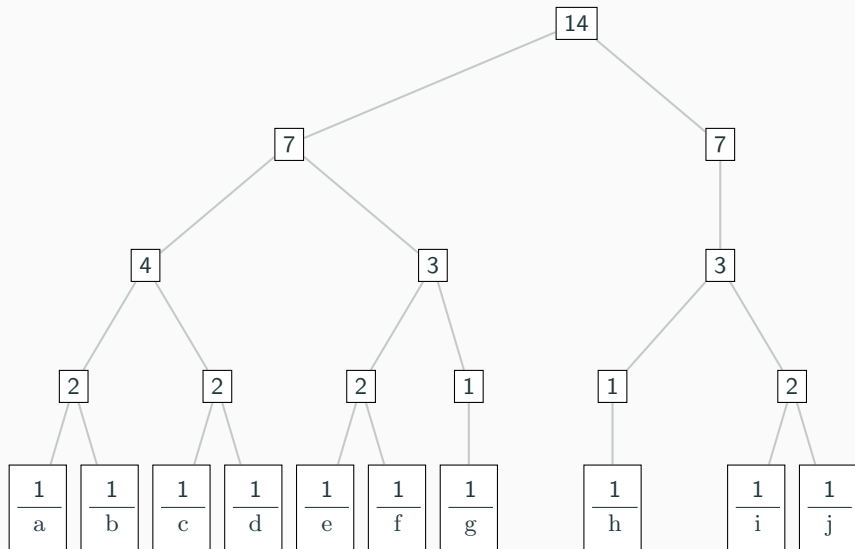
Problems:

- easily attacked with degenerate tree shapes
- peers cannot optimize data representation for their use-case

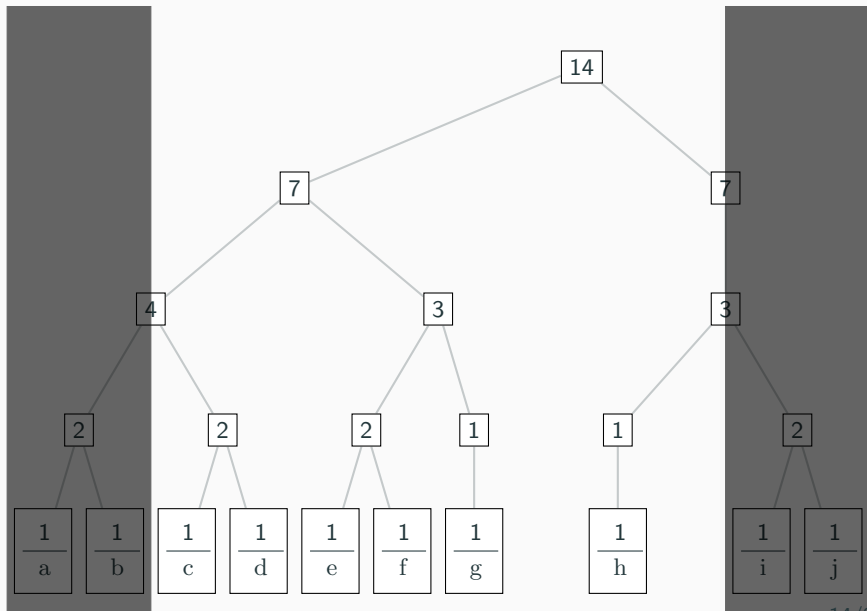
Say No to Merkle Trees



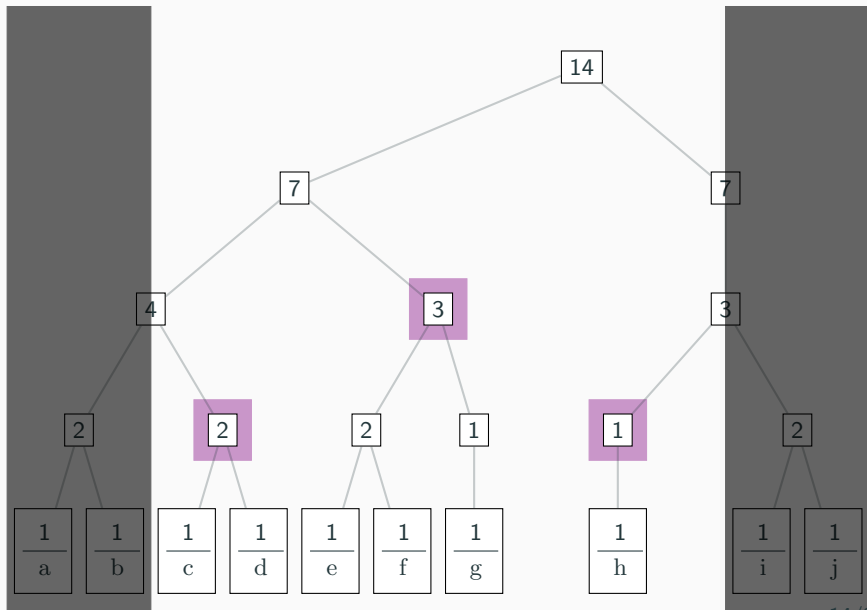
Order-Statistic Trees



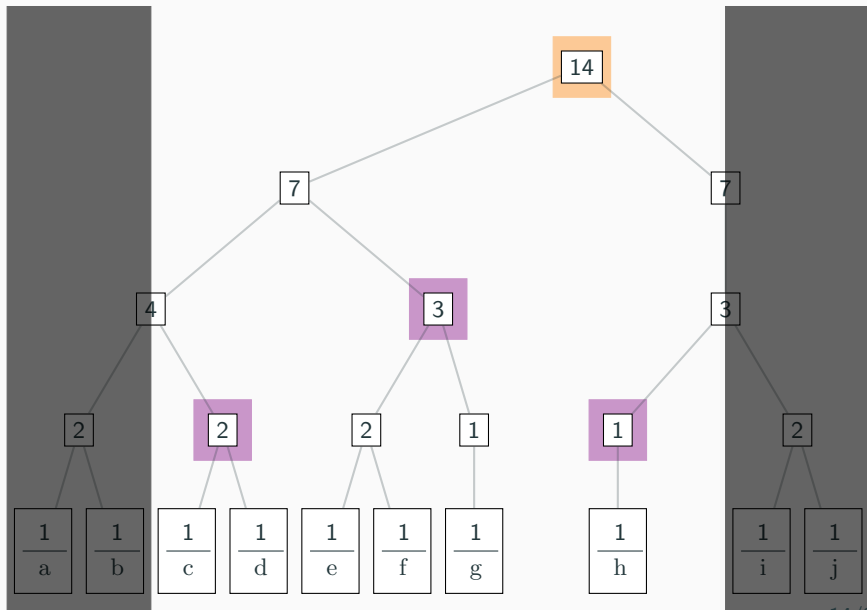
Order-Statistic Trees [c, i)



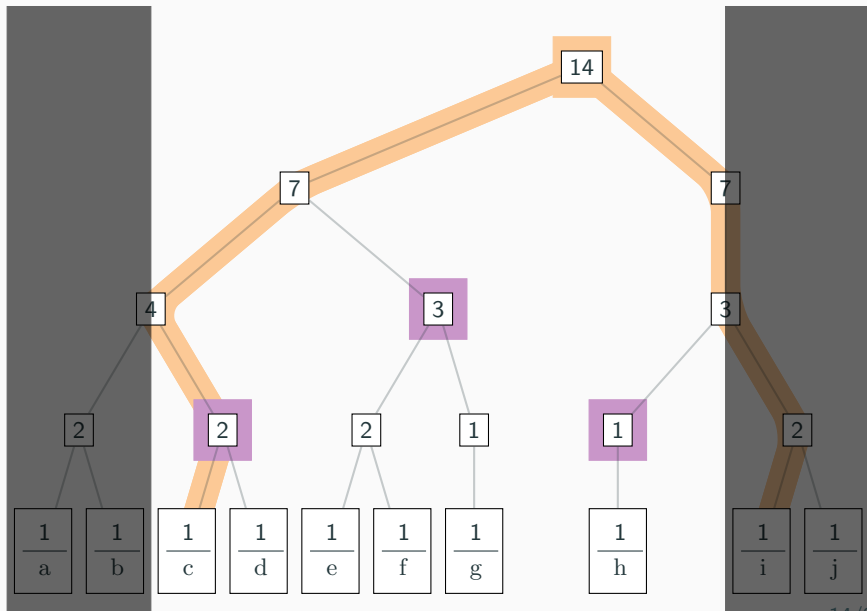
Order-Statistic Trees [c, i)



Order-Statistic Trees [c, i)



Order-Statistic Trees [c, i)



Monoid Trees

- set of labels: \mathbb{N}
- binary associative function: $+$
- neutral element: 0

Monoid Trees

Monoid:

- set of labels: \mathbb{N}
- binary associative function: $+$
- neutral element: 0

Monoid Trees

Monoid:

- set of labels: \mathbb{N}
- binary associative function: $+$
- neutral element: 0
- transforming into the monoid: $\lambda x.1$

Monoid Trees

Monoid:

- set of labels: \mathbb{N}
 - binary associative function: $+$
 - neutral element: 0
 - transforming into the monoid: $\lambda x.1$
-
- set of labels: $\{n : 0 \leq n < 2^{256} - 1\}$
 - transforming into the monoid: sha256

Monoid Trees

Monoid:

- set of labels: \mathbb{N}
 - binary associative function: $+$
 - neutral element: 0
 - transforming into the monoid: $\lambda x.1$
-
- set of labels: $\{n : 0 \leq n < 2^{256} - 1\}$
 - binary associative function: `xor`
 - neutral element: 0
 - transforming into the monoid: `sha256`

Resulting Functions

Let U be a set, \preceq a linear order on U , $\mathcal{M} = (M, \oplus, \mathbb{0})$ a monoid, and $h : U \rightarrow M$.

We *lift* h to finite sets via \mathcal{M} to obtain $\text{lift}_h^{\mathcal{M}} : \mathcal{P}(U) \rightarrow M$ with:

$$\text{lift}_h^{\mathcal{M}}(\emptyset) := \mathbb{0},$$

$$\text{lift}_h^{\mathcal{M}}(S) := h(\min(S)) \oplus \text{lift}_h^{\mathcal{M}}(S \setminus \{\min(S)\}).$$

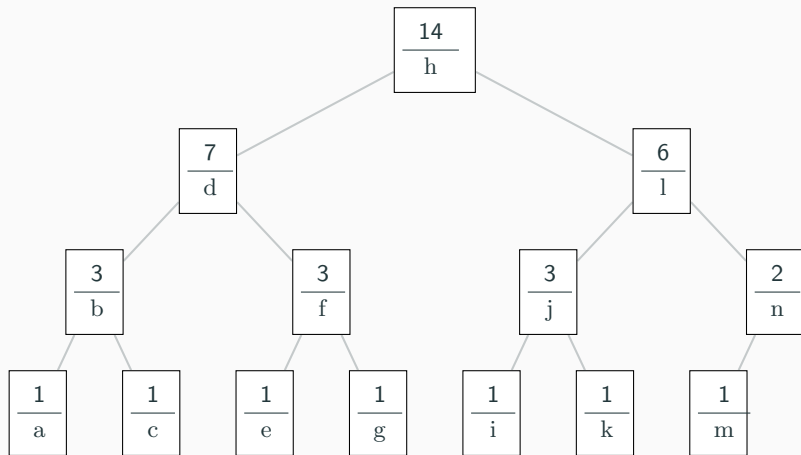
That is: $\text{lift}_h^{\mathcal{M}}(S) = h(s_1) \oplus h(s_2) \oplus \cdots \oplus h(s_{|S|})$.

Advantages

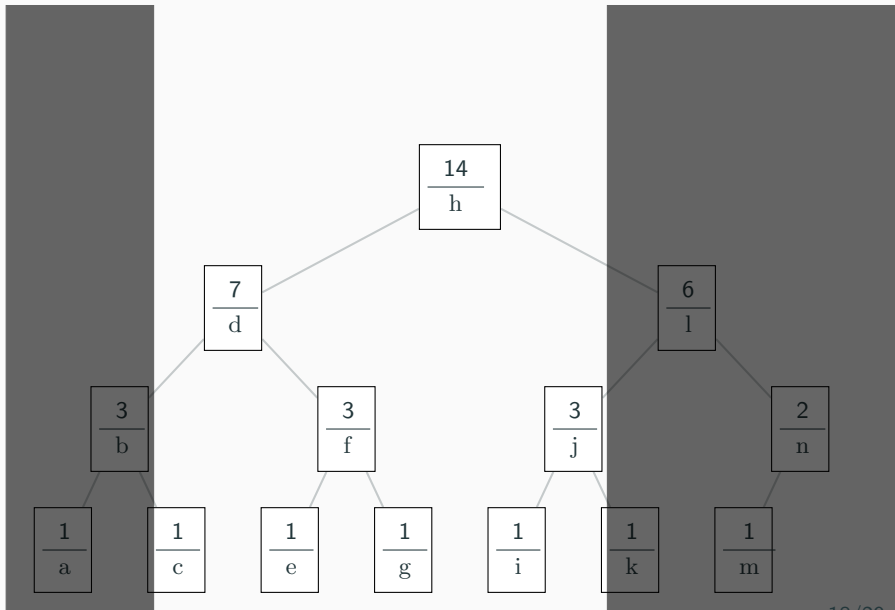
Range-based set reconciliation with such fingerprints achieves

- low computational overhead,
- solid worst-case communication complexity, and
- implementation independence.

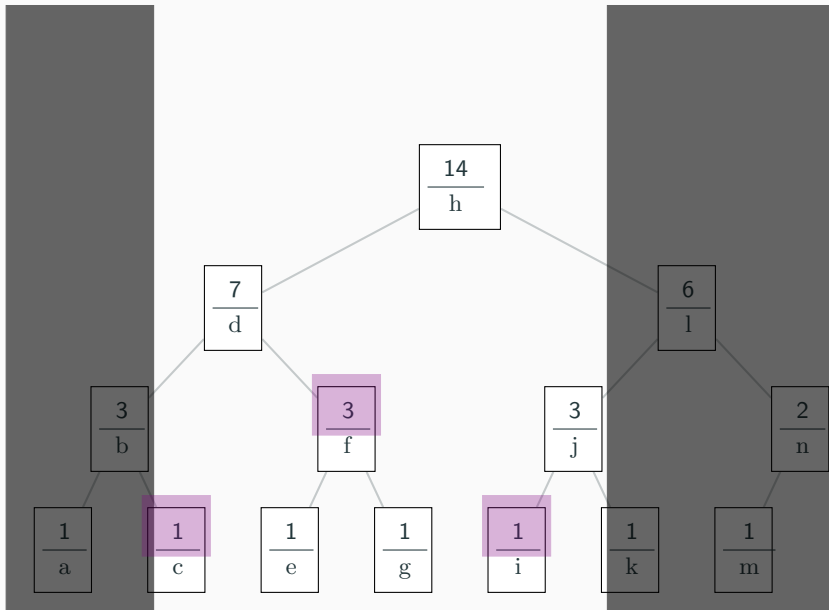
Alternative Datastructures



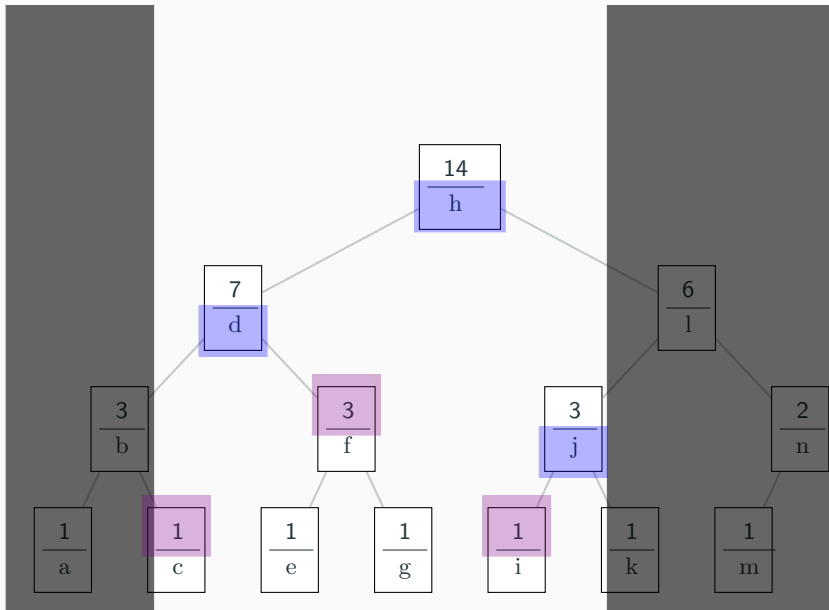
Alternative Datastructures



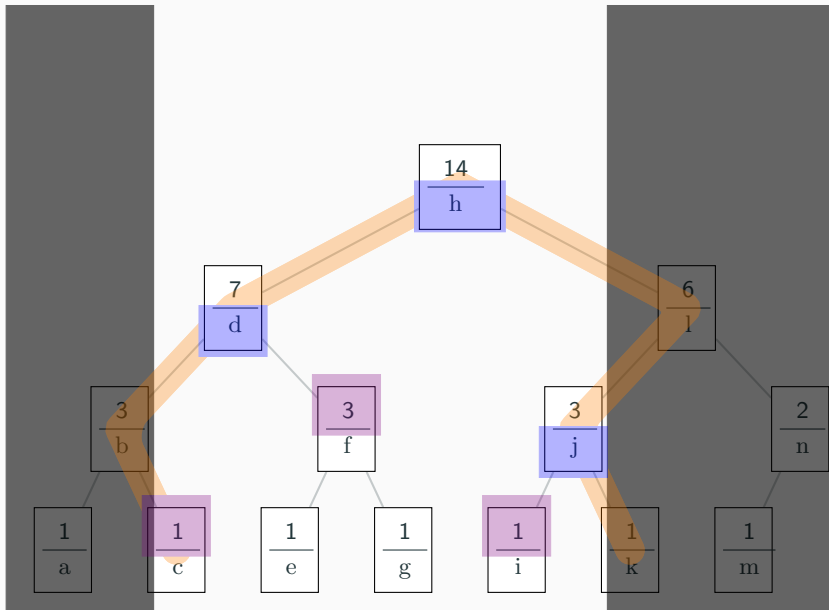
Alternative Datastructures



Alternative Datastructures



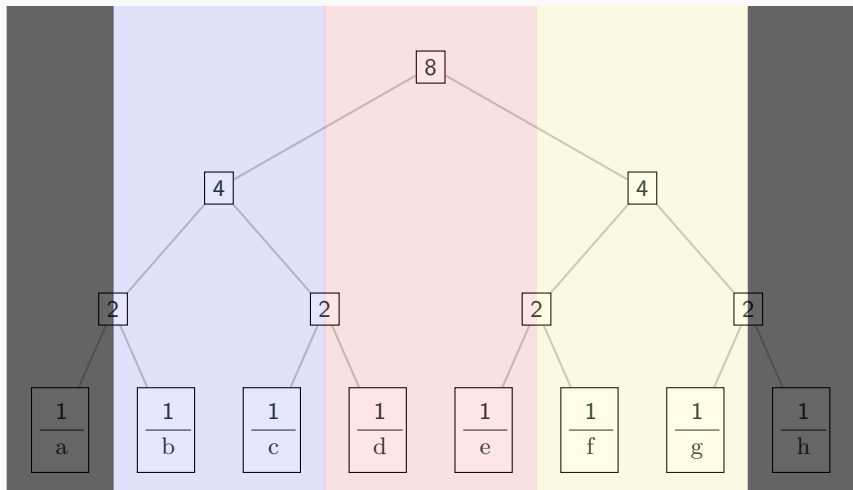
Alternative Datastructures



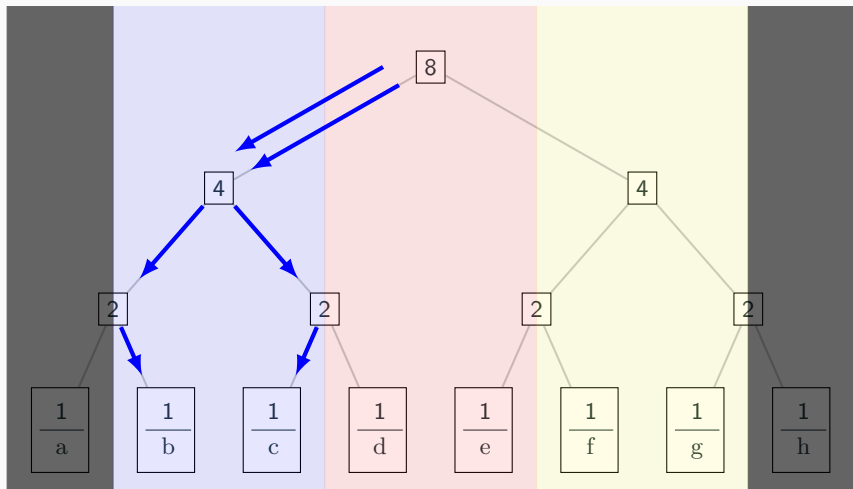
Alternative Datastructures

- monoid B-Tree
- monoid prefix tree
- monoid skip list
- monoid zip-tree
- no datastructure at all
- ...

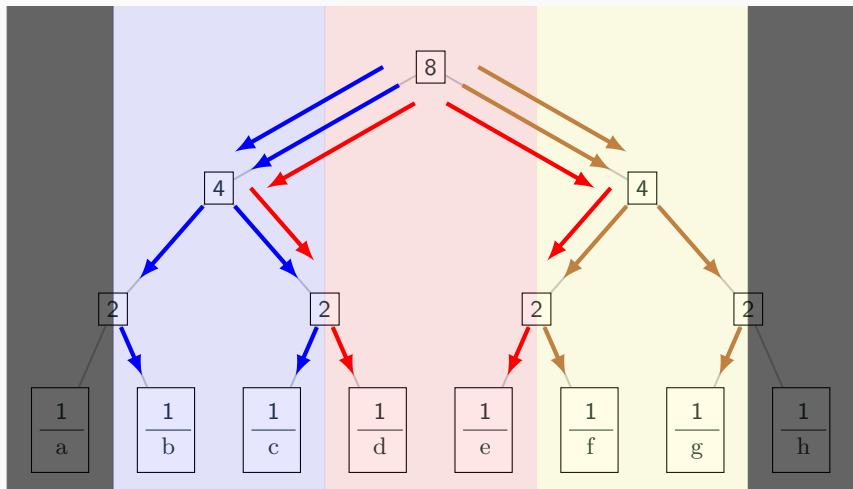
Successive Ranges



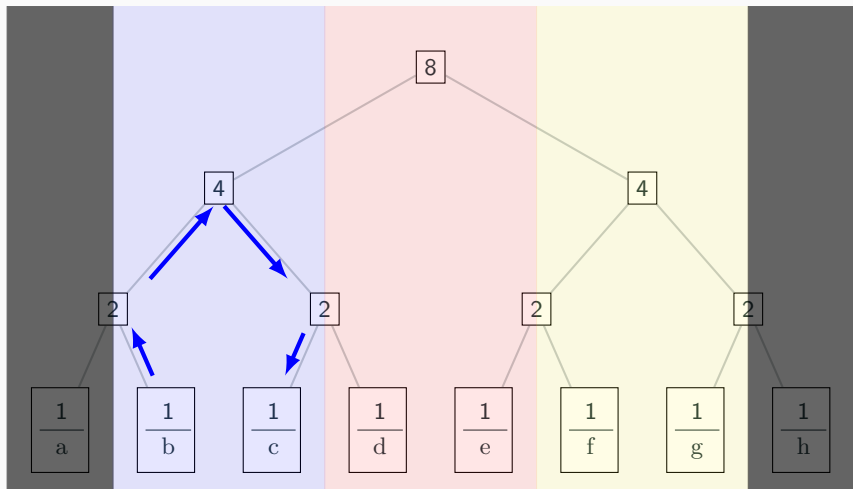
Successive Ranges



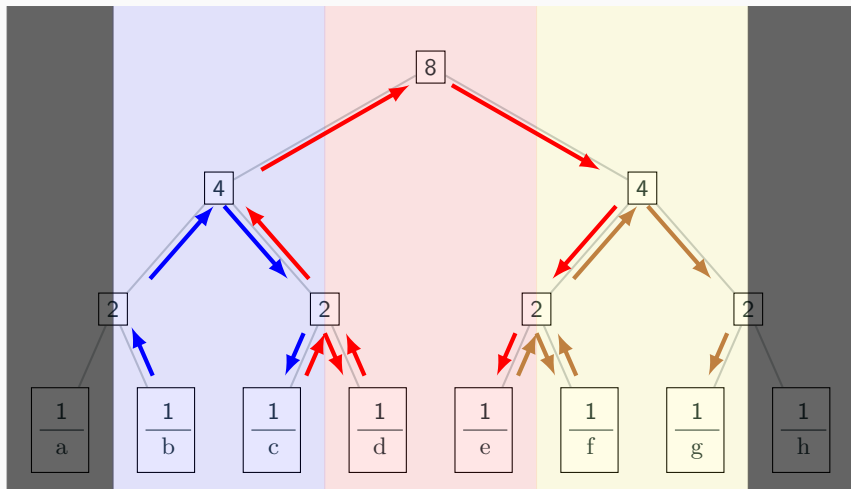
Successive Ranges



Successive Ranges



Successive Ranges



Some Peers Are Out To Get Us

- adversary must sabotage reconciliation

Some Peers Are Out To Get Us

- adversary must sabotage reconciliation
- active and passive adversaries

Some Peers Are Out To Get Us

- adversary must sabotage reconciliation
- active and passive adversaries
- randomized boundaries defeat individual collisions

Some Peers Are Out To Get Us

- adversary must sabotage reconciliation
- active and passive adversaries
- randomized boundaries defeat individual collisions
- better protection: secure hash functions

Secure Hash Functions

Let U be a set, $\mathcal{M} = (M, \oplus, \mathbb{0})$ a monoid, and $f : \mathcal{P}(U) \rightarrow M$.

f is *set-homomorphic* if $f(U_1 \cup U_2) = f(U_1) \oplus f(U_2)$.

Secure Hash Functions

Let U be a set, $\mathcal{M} = (M, \oplus, \mathbb{0})$ a monoid, and $f : \mathcal{P}(U) \rightarrow M$.

f is *set-homomorphic* if $f(U_1 \cup U_2) = f(U_1) \oplus f(U_2)$.

~~xor~~, ~~addition~~, multiplication, lattices, RSA, elliptic curves.

Secure Hash Functions

Let U be a set, $\mathcal{M} = (M, \oplus, \mathbb{0})$ a monoid, and $f : \mathcal{P}(U) \rightarrow M$.

f is *set-homomorphic* if $f(U_1 \cup U_2) = f(U_1) \oplus f(U_2)$.

~~xor~~, ~~addition~~, multiplication, lattices, RSA, elliptic curves.

Our functions: $\text{lift}_h^{\mathcal{M}}(S) = h(s_1) \oplus h(s_2) \oplus \cdots \oplus h(s_{|S|})$

Secure Hash Functions

Let U be a set, $\mathcal{M} = (M, \oplus, \mathbb{0})$ a monoid, and $f : \mathcal{P}(U) \rightarrow M$.

f is *set-homomorphic* if $f(U_1 \cup U_2) = f(U_1) \oplus f(U_2)$.

~~xor~~, ~~addition~~, multiplication, lattices, RSA, elliptic curves.

Our functions: $\text{lift}_h^{\mathcal{M}}(S) = h(s_1) \oplus h(s_2) \oplus \cdots \oplus h(s_{|S|})$

Let further \preceq a linear order on U .

h is a *tree-friendly function* if for $S_0, S_1 \in \mathcal{P}(U)$ with $\max(S_0) \prec \min(S_1)$, we have $h(S_0 \cup S_1) = h(S_0) \oplus h(S_1)$.

Secure Hash Functions

Let U be a set, $\mathcal{M} = (M, \oplus, \mathbb{0})$ a monoid, and $f : \mathcal{P}(U) \rightarrow M$.

f is *set-homomorphic* if $f(U_1 \cup U_2) = f(U_1) \oplus f(U_2)$.

~~xor~~, ~~addition~~, multiplication, lattices, RSA, elliptic curves.

Our functions: $\text{lift}_h^{\mathcal{M}}(S) = h(s_1) \oplus h(s_2) \oplus \cdots \oplus h(s_{|S|})$

Let further \preceq a linear order on U .

h is a *tree-friendly function* if for $S_0, S_1 \in \mathcal{P}(U)$ with $\max(S_0) \prec \min(S_1)$, we have $h(S_0 \cup S_1) = h(S_0) \oplus h(S_1)$.

No commutativity: Cayley hash functions

Summary

- divide-and-conquer to find differences by checking fingerprint equality
- tree-friendly functions allow for efficient and flexible implementation
- cryptographically secure tree-friendly functions exist
- pretty simple compared to traditional solutions

Summary

- divide-and-conquer to find differences by checking fingerprint equality
- tree-friendly functions allow for efficient and flexible implementation
- cryptographically secure tree-friendly functions exist
- pretty simple compared to traditional solutions
- remember to say no to Merkle trees

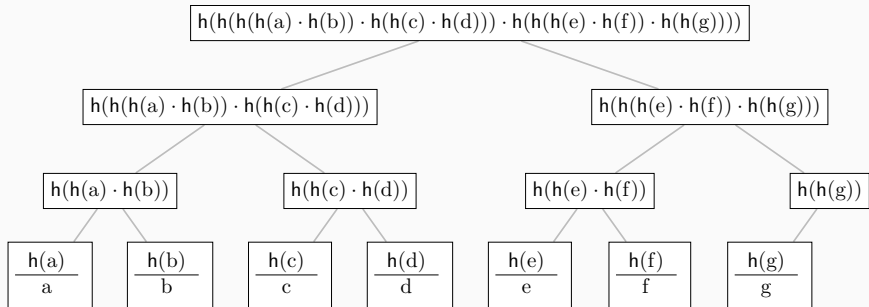
Bonus Slides!

Reducing Computation Times

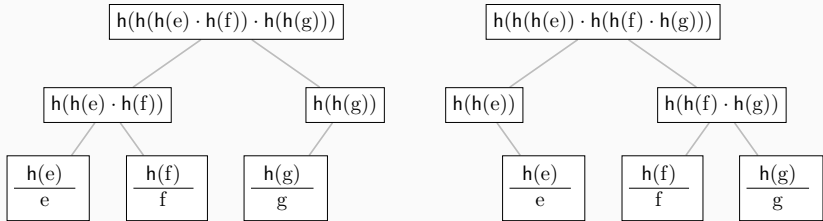
- Step 1: Put a Merkle tree on it
- Step 2: ???
- Step 3: Profit

Aurolat, Alex, and François Taïani. "Merkle search trees: Efficient state-based CRDTs in open networks." 2019 38th Symposium on Reliable Distributed Systems (SRDS). IEEE, 2019.

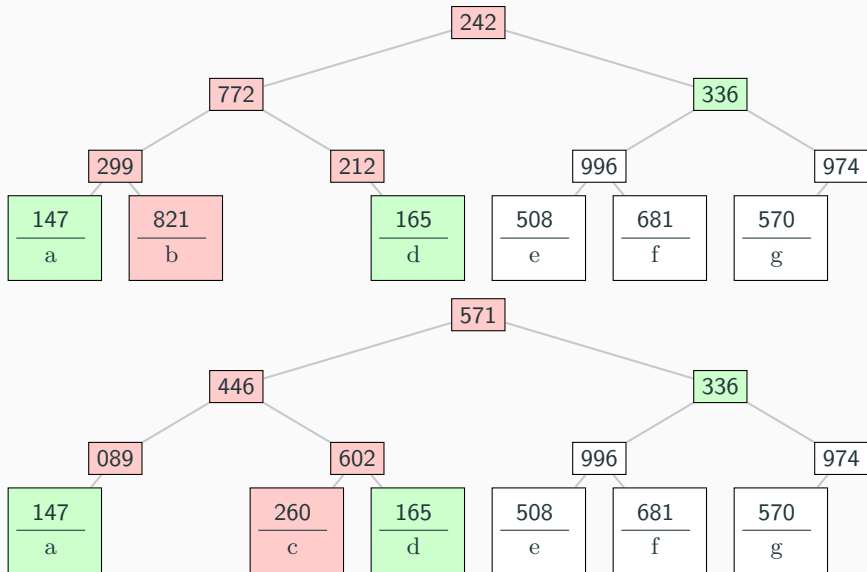
Merkle Trees



Merkle Trees



Merkle Tree Reconciliation



Merkle Tree Reconciliation

- inflexible data representation
- unacceptable worst-case complexity
 - remember, some peers are out to get us