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

Model and Analysis

- roundtrips
- communicated bytes
- computation time per reconciliation session
- computation space per reconciliation session
- computation time per item
- computation space per item

P2P Reconciliation

Peer-to-peer systems:

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

P2P Reconciliation

Peer-to-peer systems:

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

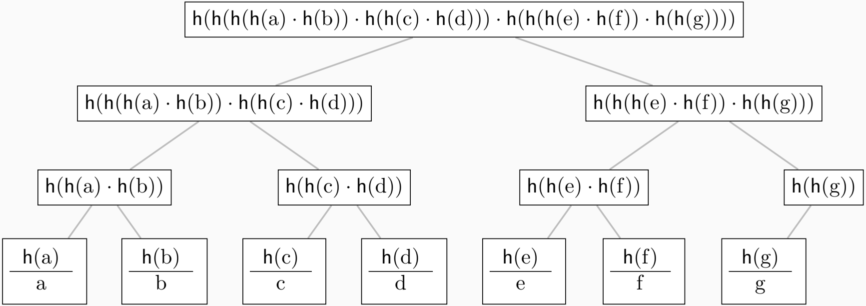
$\implies$ traditional approaches don't work

Reducing Computation Times

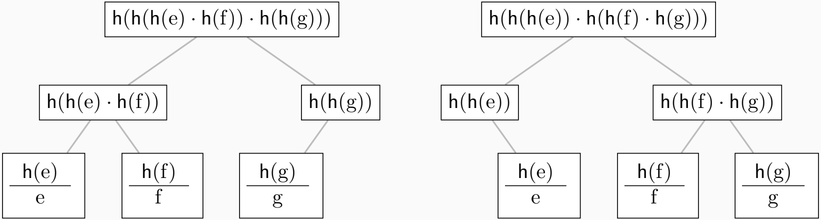
- Step 1: Put a Merkle tree on it
- Step 2: ???
- Step 3: Profit

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

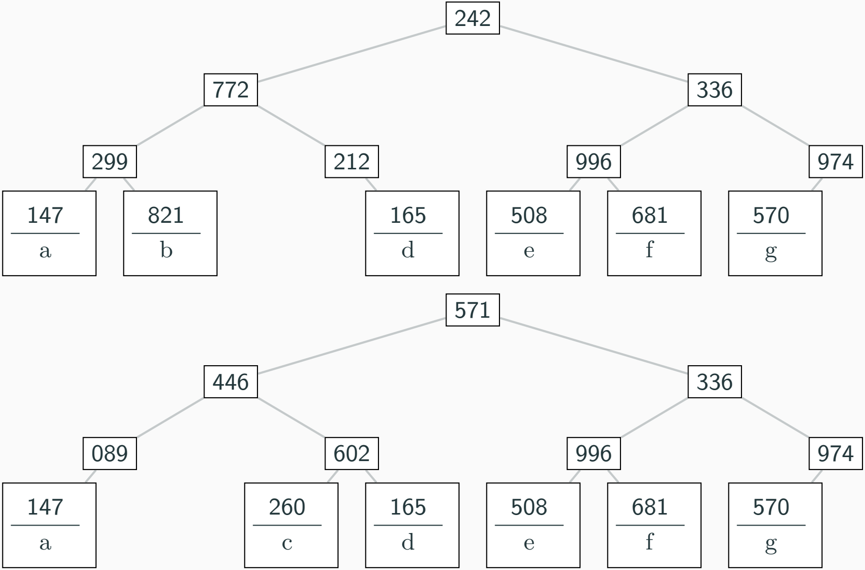
Merkle Trees



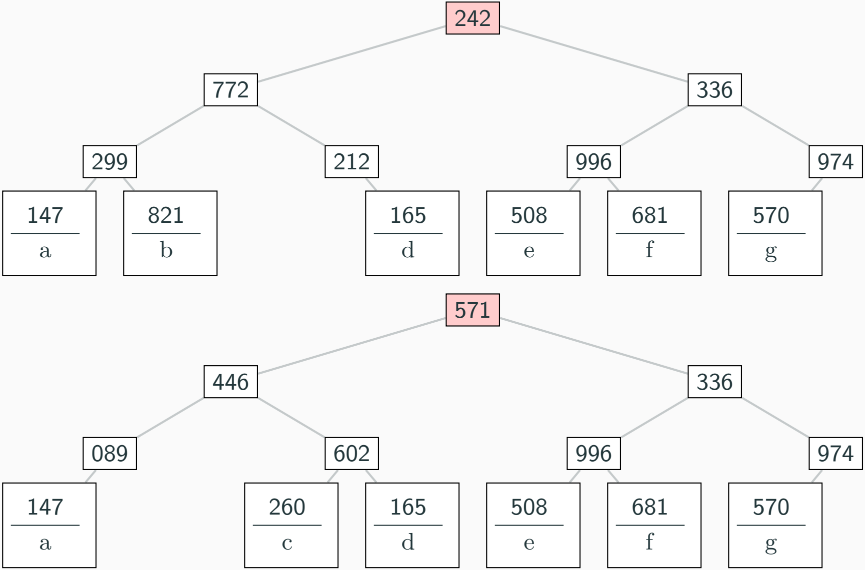
Merkle Trees



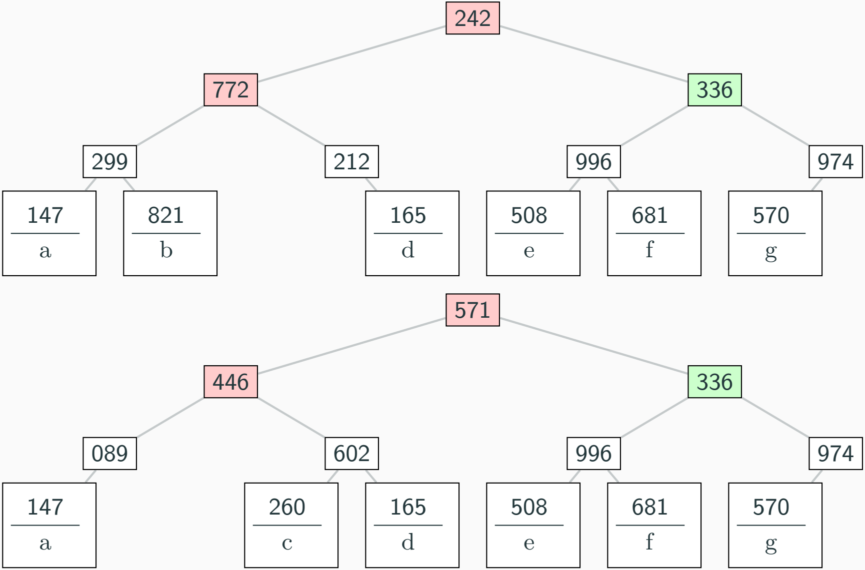
Merkle Tree Reconciliation



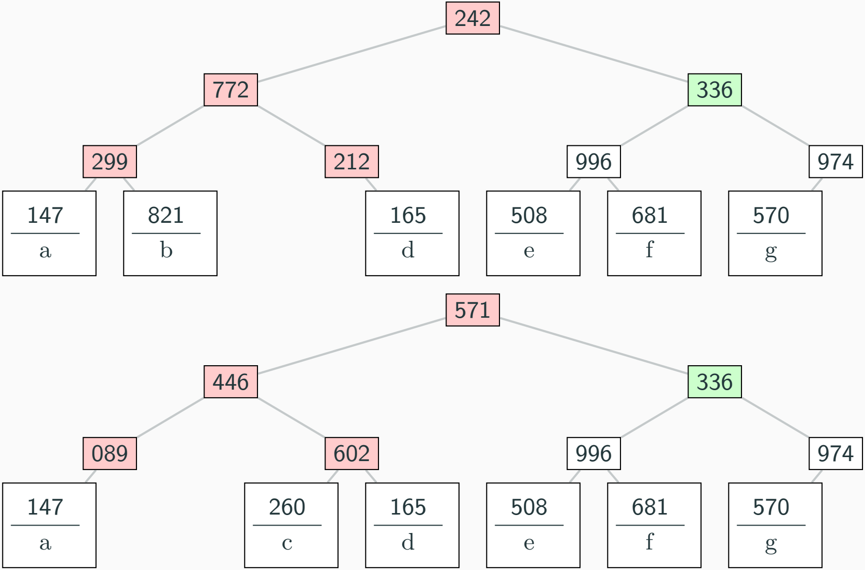
Merkle Tree Reconciliation



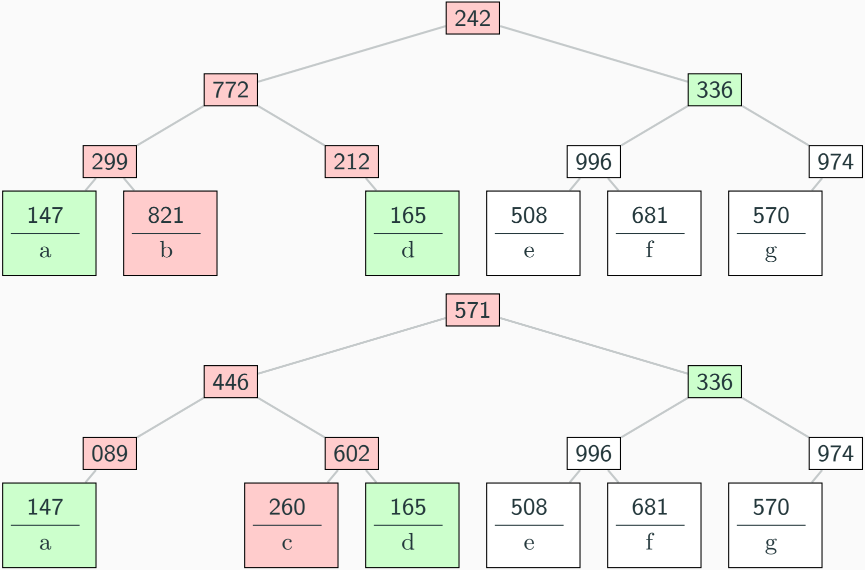
Merkle Tree Reconciliation



Merkle Tree Reconciliation



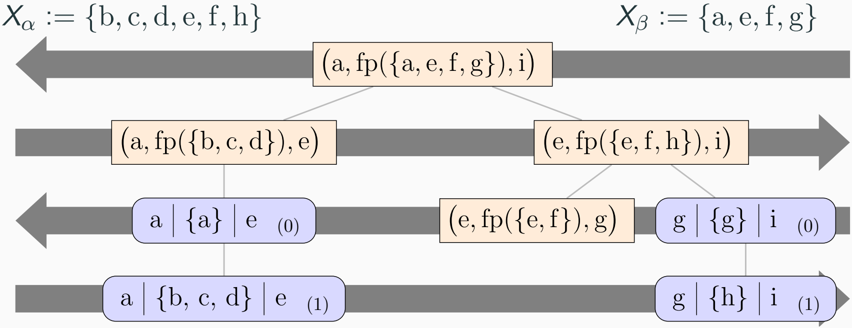
Merkle Tree Reconciliation



Merkle Tree Reconciliation

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

Range-Based Set Reconciliation



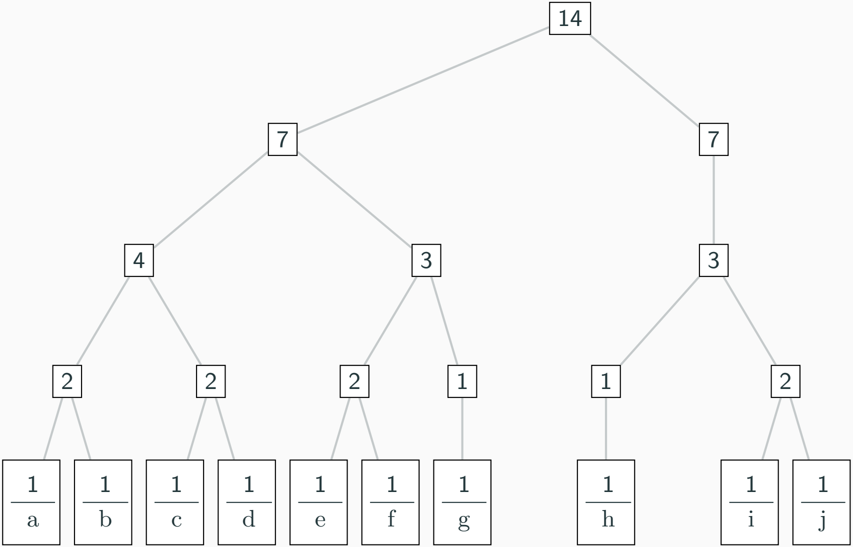
Some Nice Properties

- reasonably efficient: $\mathcal{O}(\min(n_\triangle \cdot \log(n), n))$ bytes communication, $\mathcal{O}(1)$ working memory
- can interpolate toward trivial
- arbitrary recursion anchor protocols
- arbitrary partition techniques

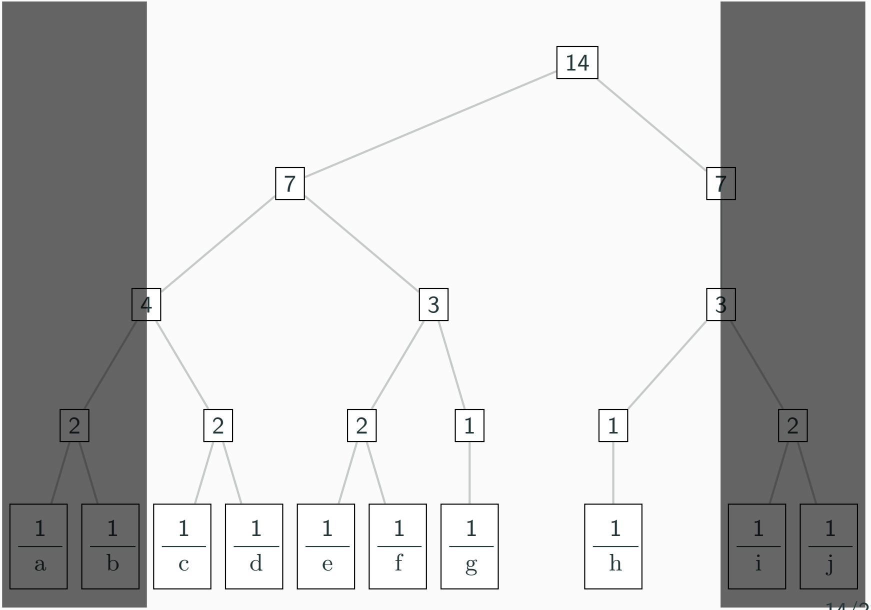
Some Nice Properties

- reasonably efficient: $\mathcal{O}(\min(n_{\triangle} \cdot \log(n), n))$ bytes communication, $\mathcal{O}(1)$ working memory
- can interpolate toward trivial
- arbitrary recursion anchor protocols
- arbitrary partition techniques
- but: linear computation times

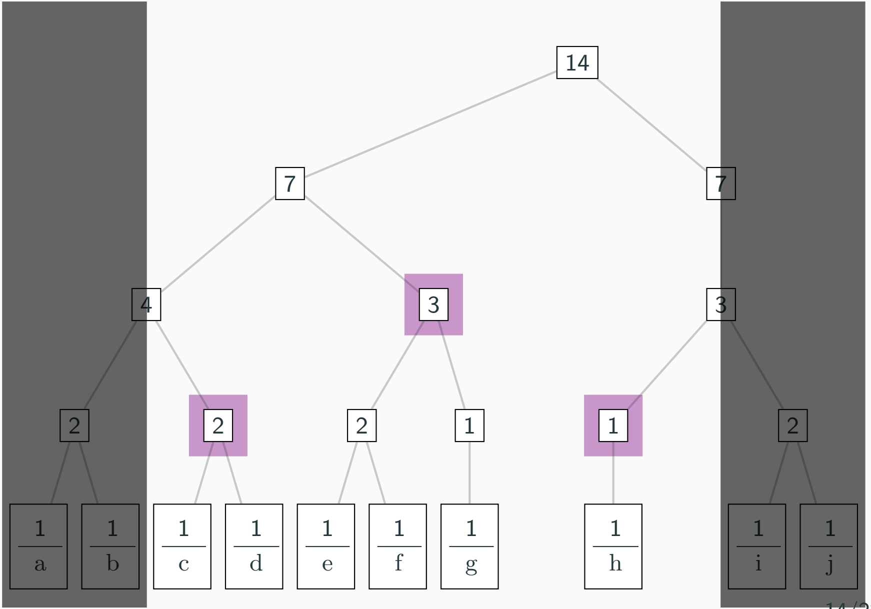
Order-Statistic Trees



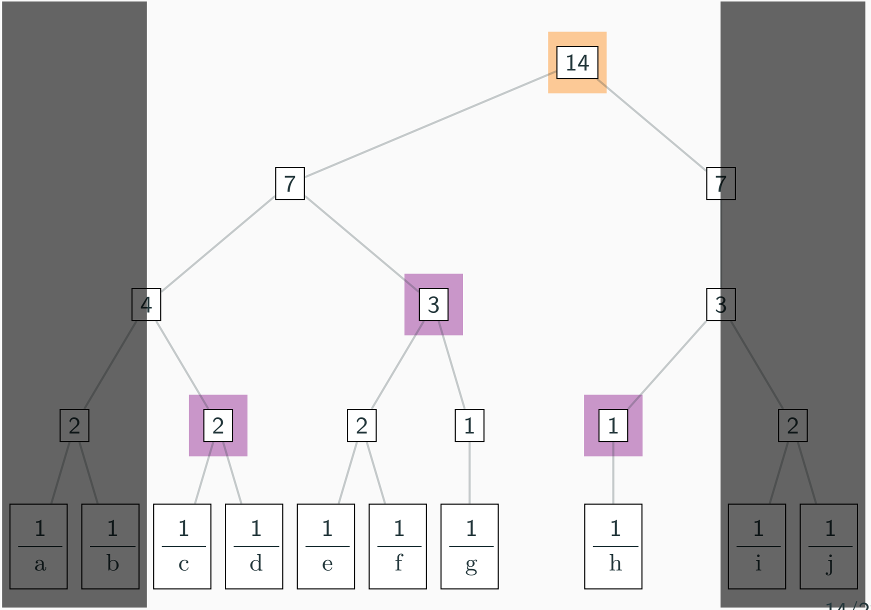
Order-Statistic Trees $[c, i)$



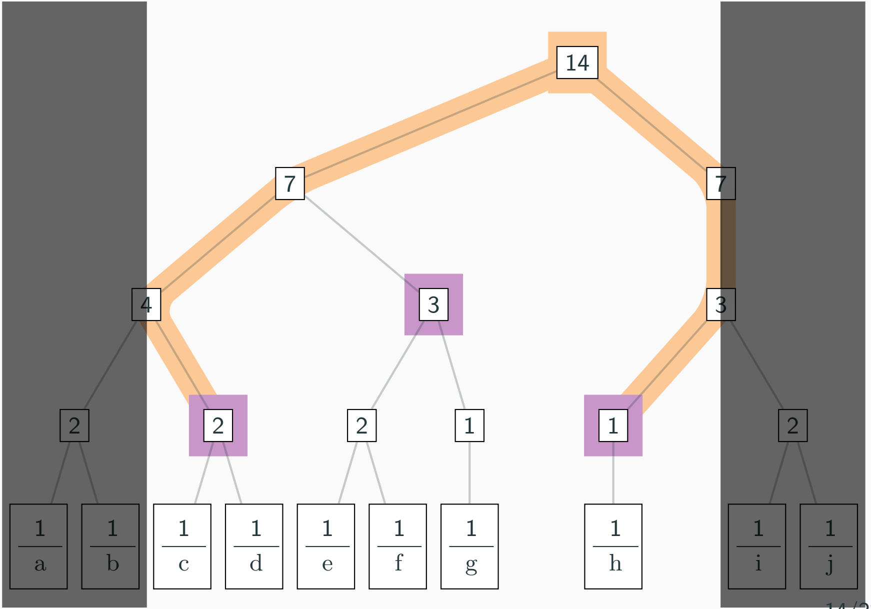
Order-Statistic Trees $[c, i)$



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

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

Monoid Trees

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

- set of labels: $\{n : 0 <= n < 2^{256} - 1\}$

- lifting into the monoid: sha256

Monoid Trees

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

- set of labels: $\{n : 0 <= n < 2^{256} - 1\}$
- binary associative function: xor
- neutral element: 0
- lifting into the monoid: sha256

Resulting Hash Functions

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

We *lift* f *to finite sets via* $\mathcal{M}$ to obtain $\mathsf{lift}_{\mathsf{f}}^{\mathcal{M}} : \mathcal{P}(U) \rightharpoonup M$ with:

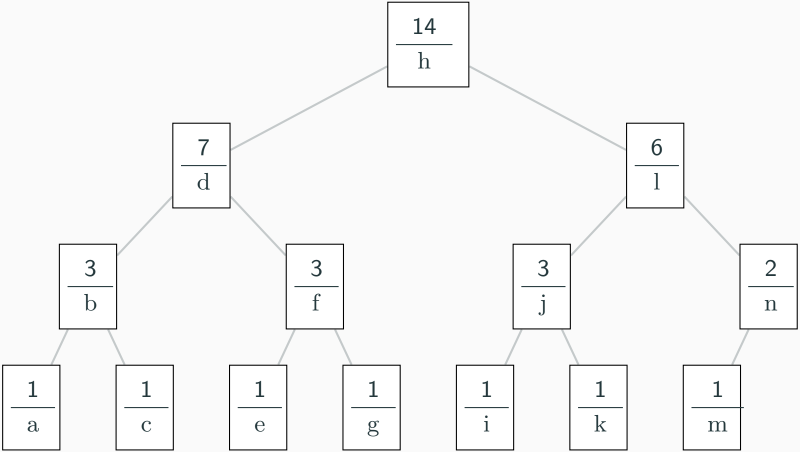
$$\begin{aligned} \mathsf{lift}_{\mathsf{f}}^{\mathcal{M}}(\emptyset) &:= \mathbb{0}, \\ \mathsf{lift}_{\mathsf{f}}^{\mathcal{M}}(S) &:= \mathsf{f}\big(\min(S)\big) \oplus \mathsf{lift}_{\mathsf{f}}^{\mathcal{M}}\big(S \setminus \{\min(S)\}\big). \end{aligned}$$

That is: $\mathsf{lift}_{\mathsf{f}}^{\mathcal{M}}(S) = \mathsf{f}(s_1) \oplus \mathsf{f}(s_2) \oplus \cdots \oplus \mathsf{f}(s_{|S|})$.

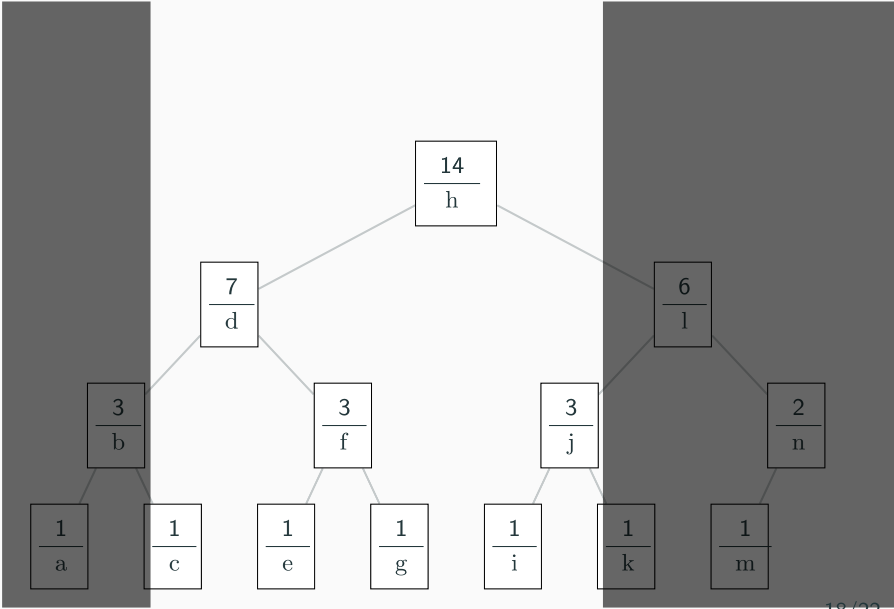
Advantages

- solid worst-case communication complexity
- implementation independence

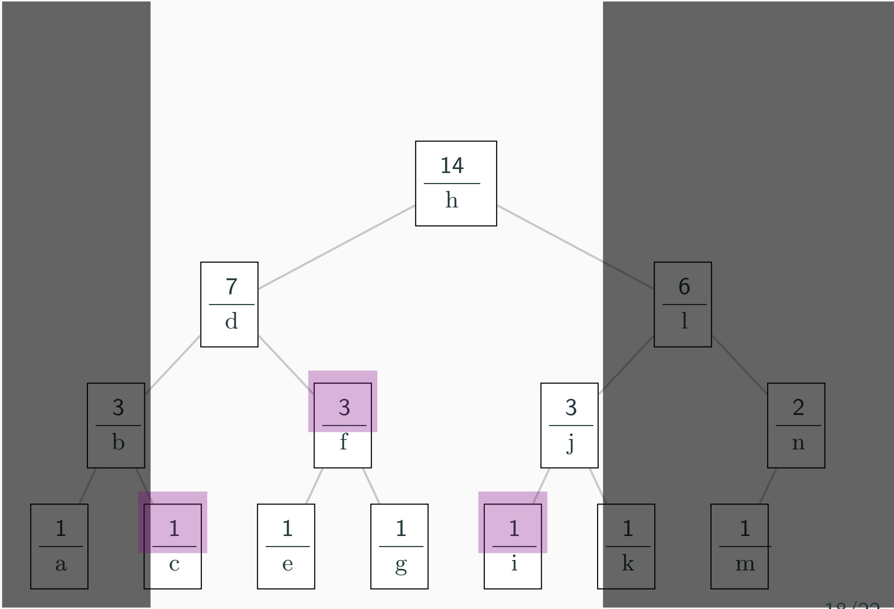
Alternative Datastructures



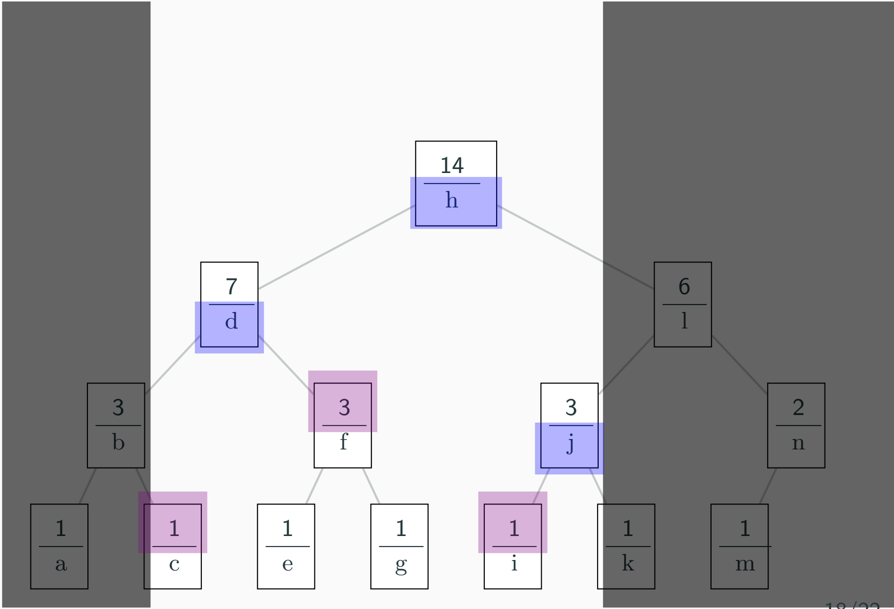
Alternative Datastructures



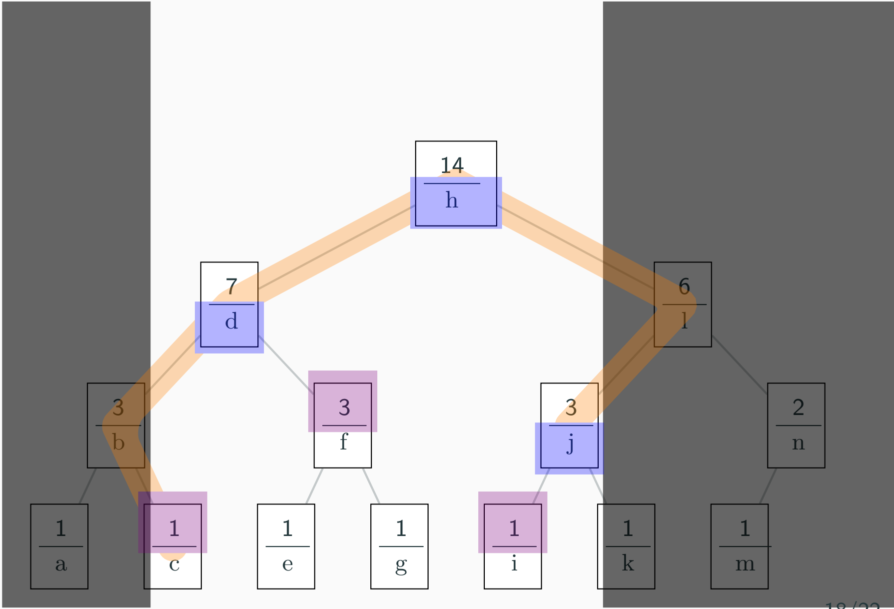
Alternative Datastructures



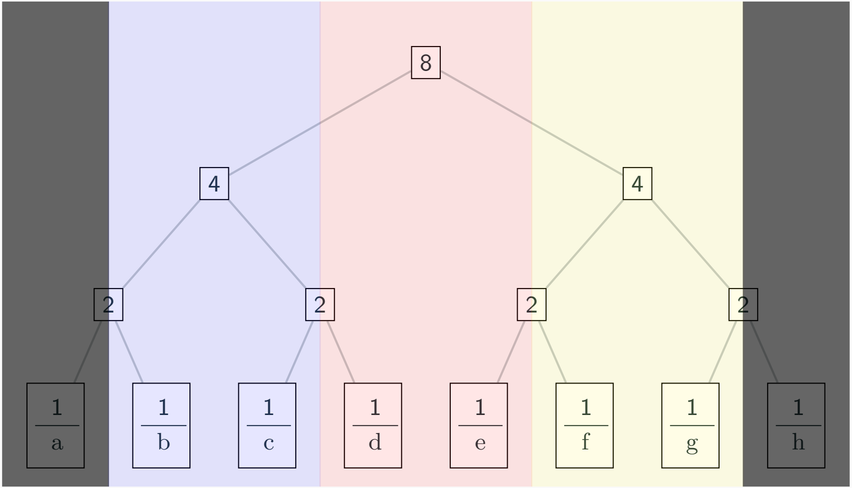
Alternative Datastructures



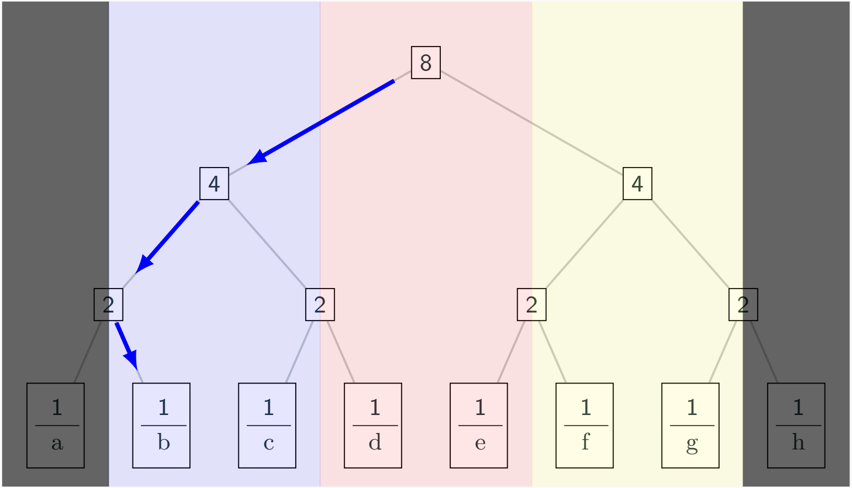
Alternative Datastructures



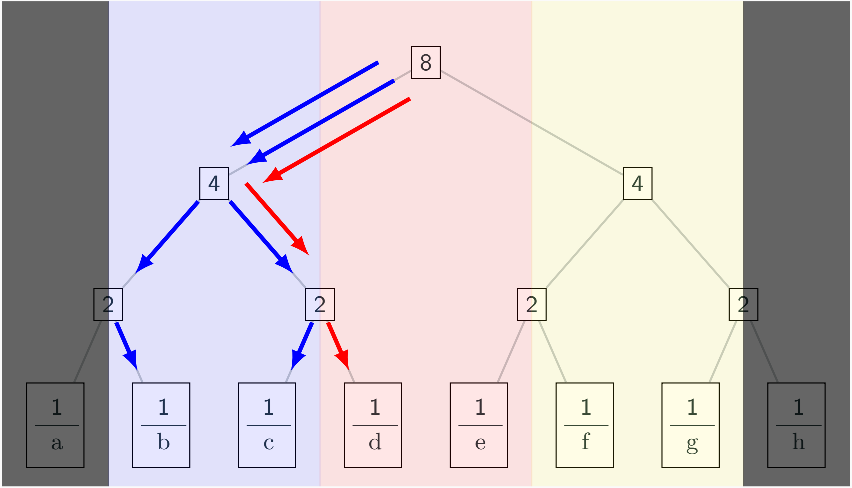
Successive Ranges



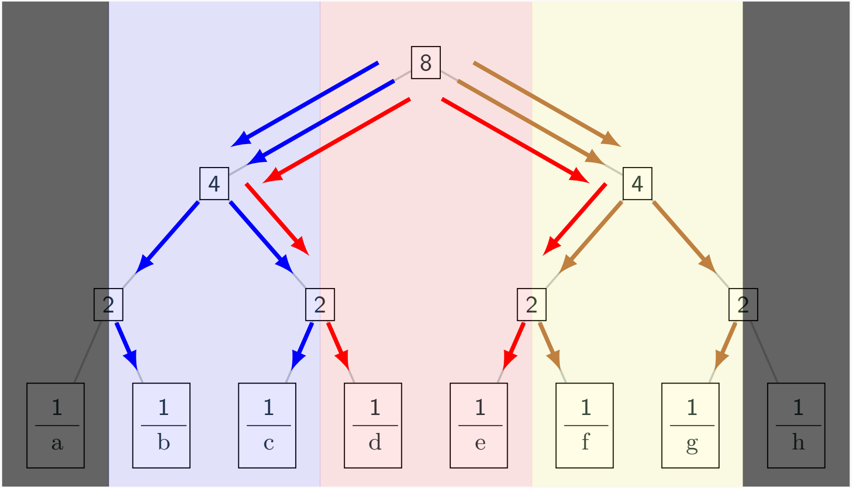
Successive Ranges



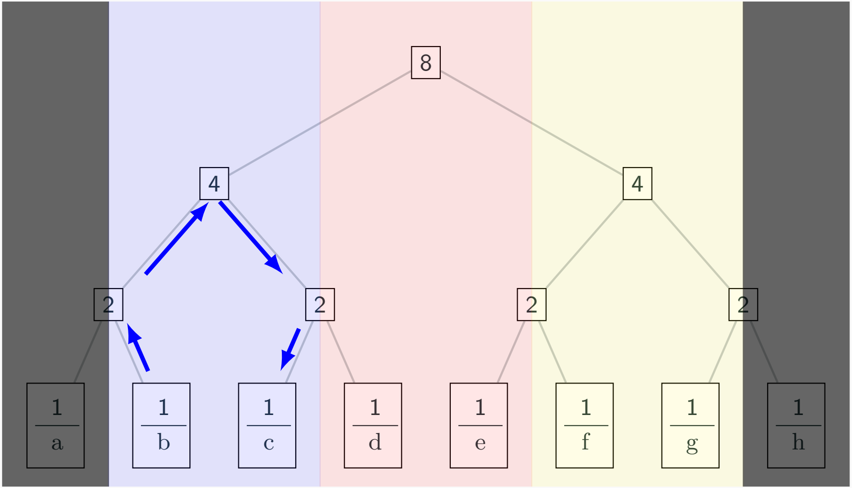
Successive Ranges



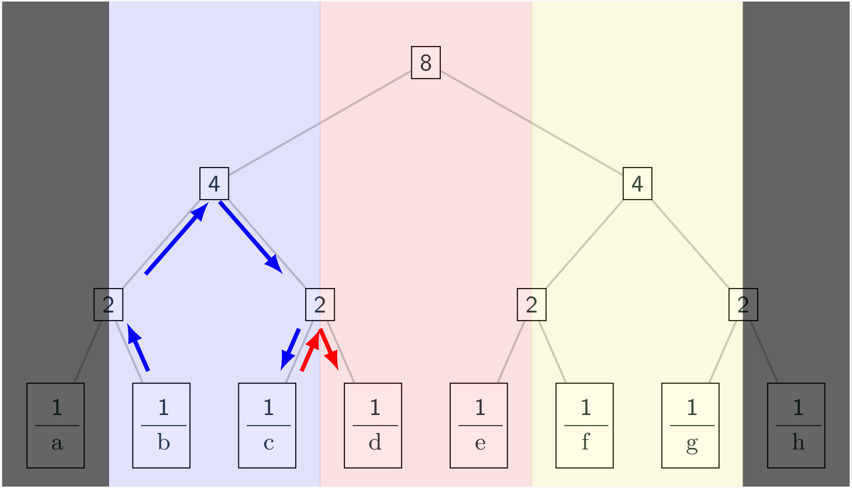
Successive Ranges



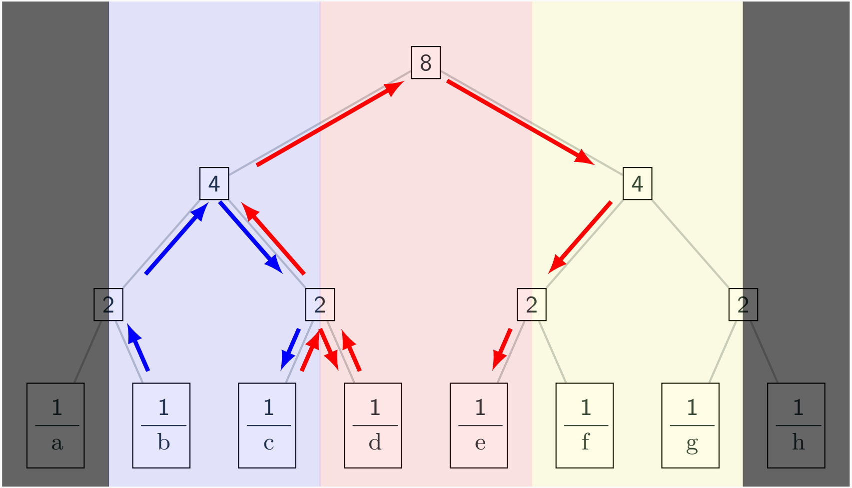
Successive Ranges



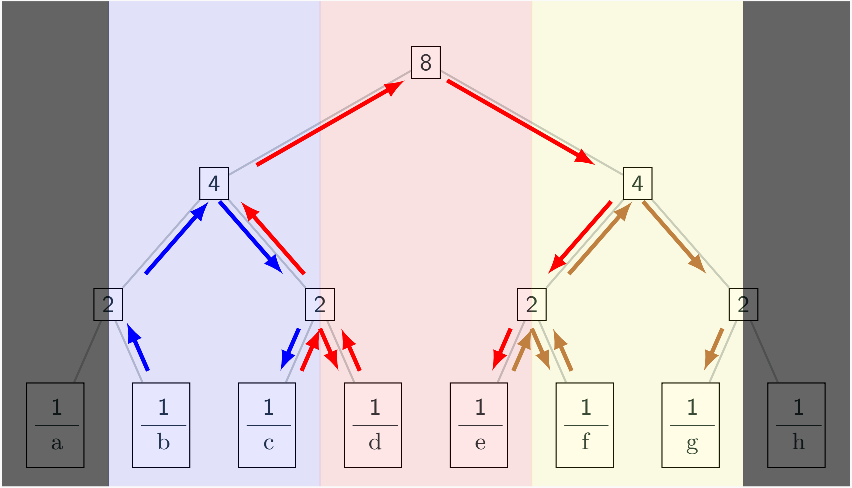
Successive Ranges



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 $\mathsf{f} : \mathcal{P}(U) \to M$.

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

Secure Hash Functions

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

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

XOR, addition, multiplication, lattices, RSA, eliptic curves.

Secure Hash Functions

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

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

XOR, addition, multiplication, lattices, RSA, eliptic curves.

Let further $\preceq$ a linear order on U.

f 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 $\mathsf{f}(S_0 \cup S_1) = \mathsf{f}(S_0) \oplus \mathsf{f}(S_1)$.

No commutativity: Cayley hash functions

Conclusion

Unlike conventional reconciliation techniques:

- robust
- simple