Space-Efficient Data Structures for Lattices

J. Ian Munro – University of Waterloo

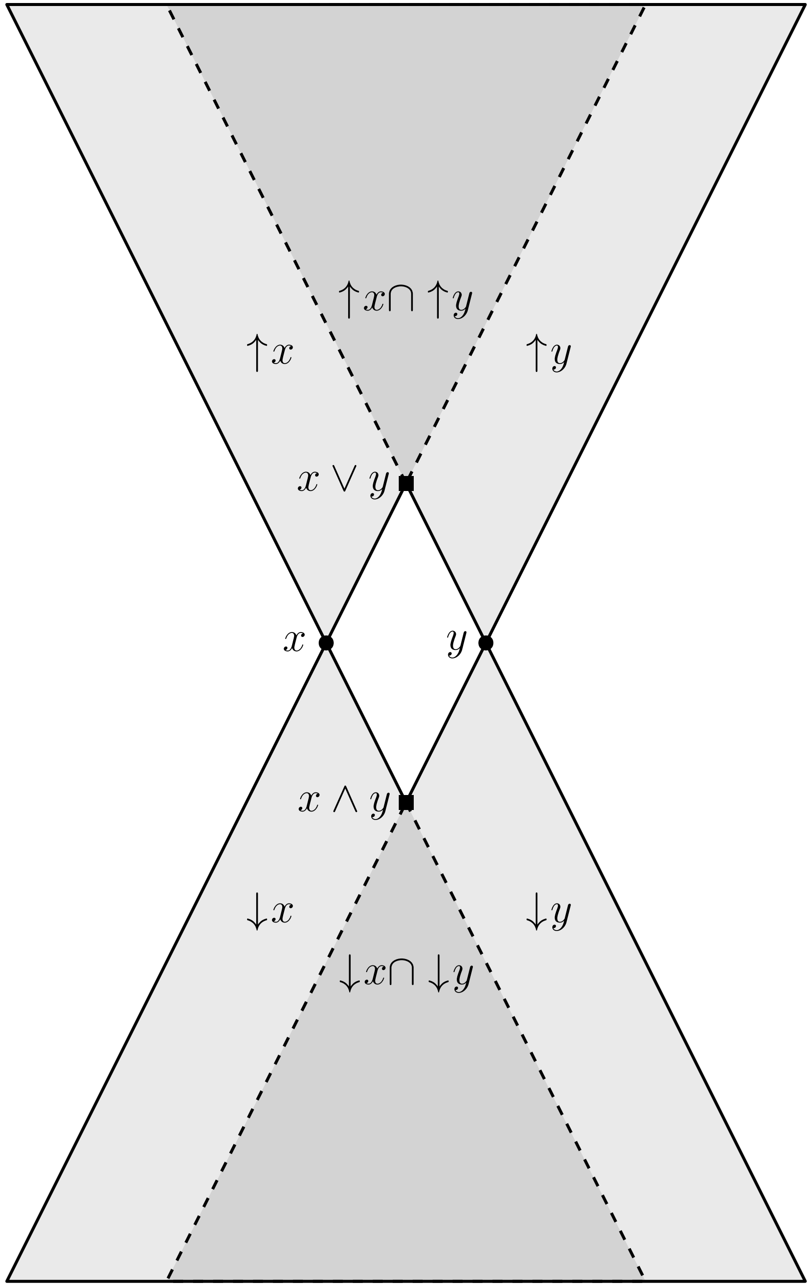
Bryce Sandlund – University of Waterloo

Corwin Sinnamon – Princeton University

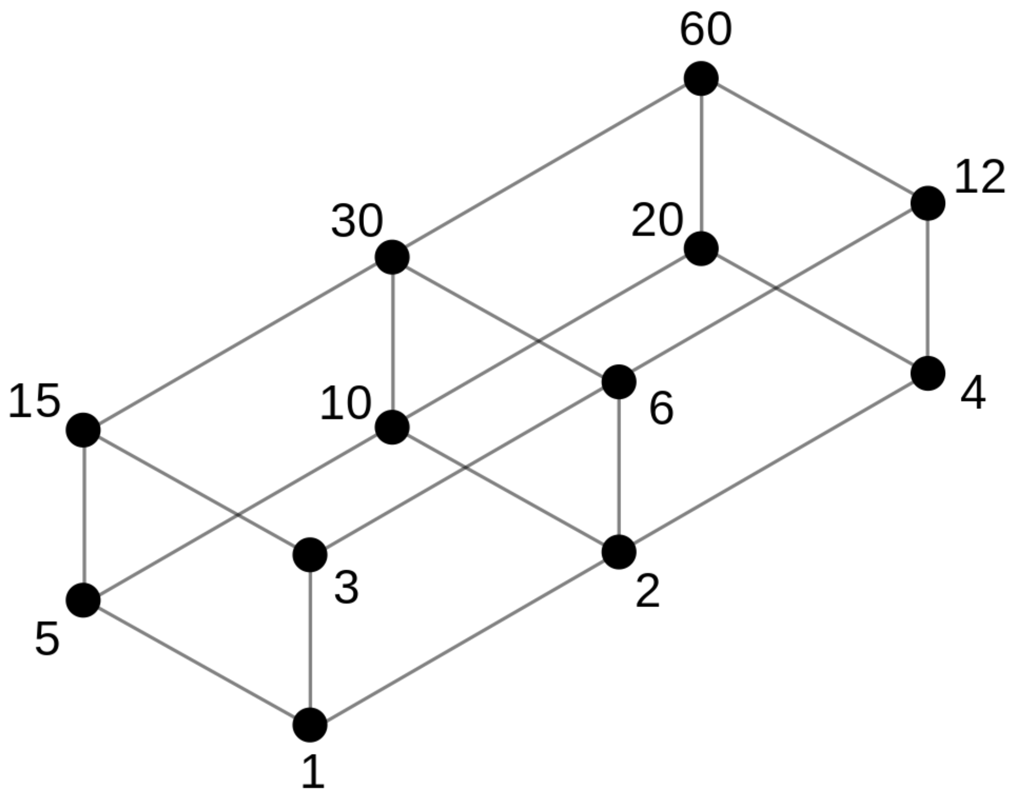
Definition

A lattice is a partial order L in which the set of elements larger than any $x, y \in L$ are all larger than (or equal to) an element $x \vee y$ known as the join of x and y.

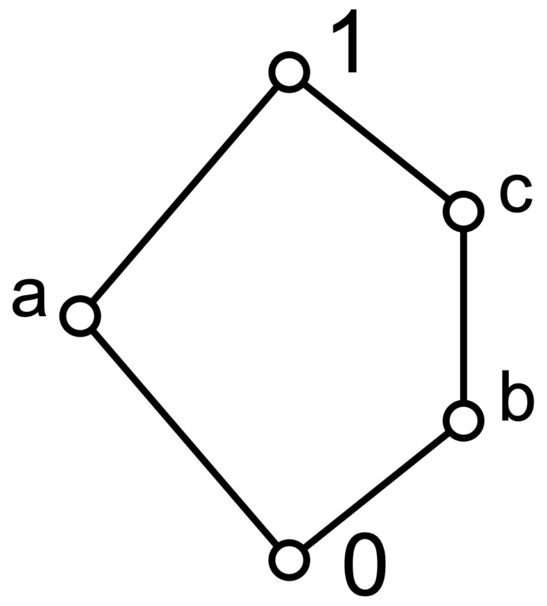
Similarly, the set of elements smaller than any $x, y \in L$ all are smaller than (or equal to) an element $x \wedge y$ known as the meet of x and y.



Examples



Left: Lattice of the integer divisors of 60, ordered by divisibility.



Right: The Lattice N_5.

The Lattice Problem

Design a space-efficient data structure over a lattice supporting the following operations:

- $TestOrder(x, y)$:= Return if $x \leq y$.
- $Meet(x, y)$:= Return the meet of x and y.
- $Join(x, y)$:= Return the join of x and y.

Lower bound: There are $2^{\theta(n\sqrt{n})}$ lattices on n elements, therefore any lattice data structure must take $\Omega(n\sqrt{n})$ bits.

Previous Work

Previous Work	Result	Citation
Store $O(\log n)$ bits for all pairs of elements	$O(n^2)$ words of space, $O(1)$ order testing and meet/join.	Naïve Attempt 1
Store the transitive reduction graph (TRG)	$O(n\sqrt{n})$ words of space, $O(n)$ order testing and meet/join.	Naïve Attempt 2
Efficient Implementation of Lattice Operations	Heuristic, $\Omega(n^2)$ space in worst-case.	Aït-Kaci et al., TOPLAS '89
An Efficient Data Structure for Lattice Operations	Claimed $O(n\sqrt{n})$ words of space, $O(1)$ order testing, $O(\sqrt{n})$ meet/join. Incorrect.	Talamo & Vocca, SICOMP '99
Time and Space Efficient Representations of Distributive Lattices	$O(n \log n)$ bits of space, $O(\log n)$ meet/join if the lattice is *distributive*.	Munro & Sinnamon, SODA '18

Our Results

1st Data Structure:

- $O(n\sqrt{n})$ words of space.
- $O(1)$ time $TestOrder(x, y)$.
- $O(n^{3/4})$ time $Meet(x, y)$ and $Join(x, y)$.

2nd Data Structure, for any $c \in \left[\frac{1}{2}, 1\right]$:

- $O(n^{1+c})$ words of space.
- $O(n^{1-c/2})$ time $Meet(x, y)$ and $Join(x, y)$.

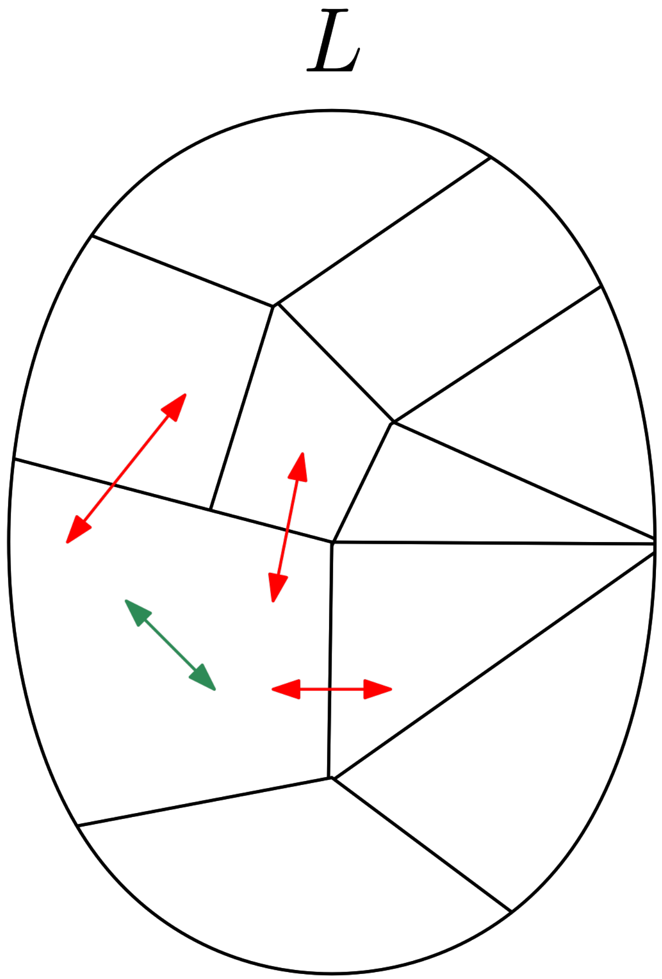
3rd Data Structure, where d is the maximum degree in the TRG of L:

- $O(n\sqrt{n})$ words of space.
- $O\left(\frac{d \log n}{\log d}\right)$ time $Meet(x, y)$ and $Join(x, y)$.

We will refer to space complexity in words for the rest of the talk

Intuition

- Break L into smaller blocks.
- Store information within each block and between a block and the rest of L.
- Goal: $\sqrt{n}$ blocks of $\sqrt{n}$ elements each.

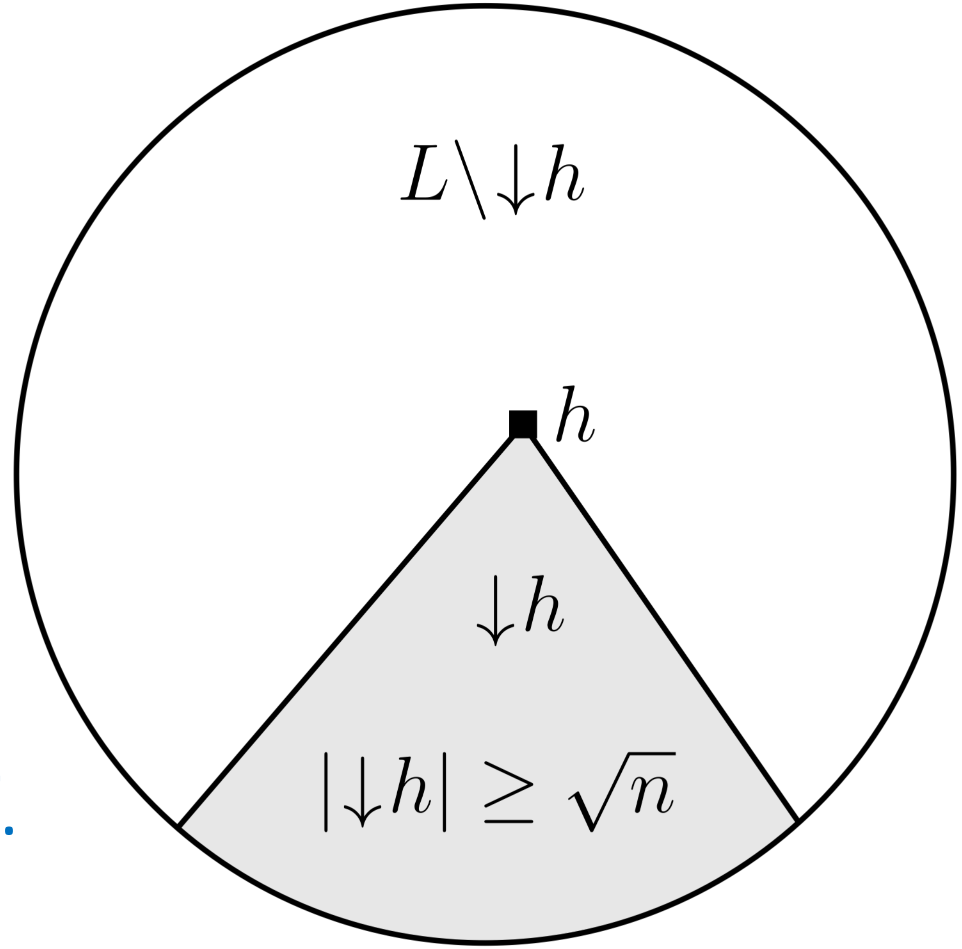


Block Decomposition

- Use the set of elements below an element, h, as a block B.
- Choose h so that $|\downarrow h|$ is smallest possible while satisfying $|\downarrow h| \geq \sqrt{n}$.

Implication 1:

All $x \in B$, $x \neq h$, satisfy $|\downarrow x \cap B| < \sqrt{n}$.

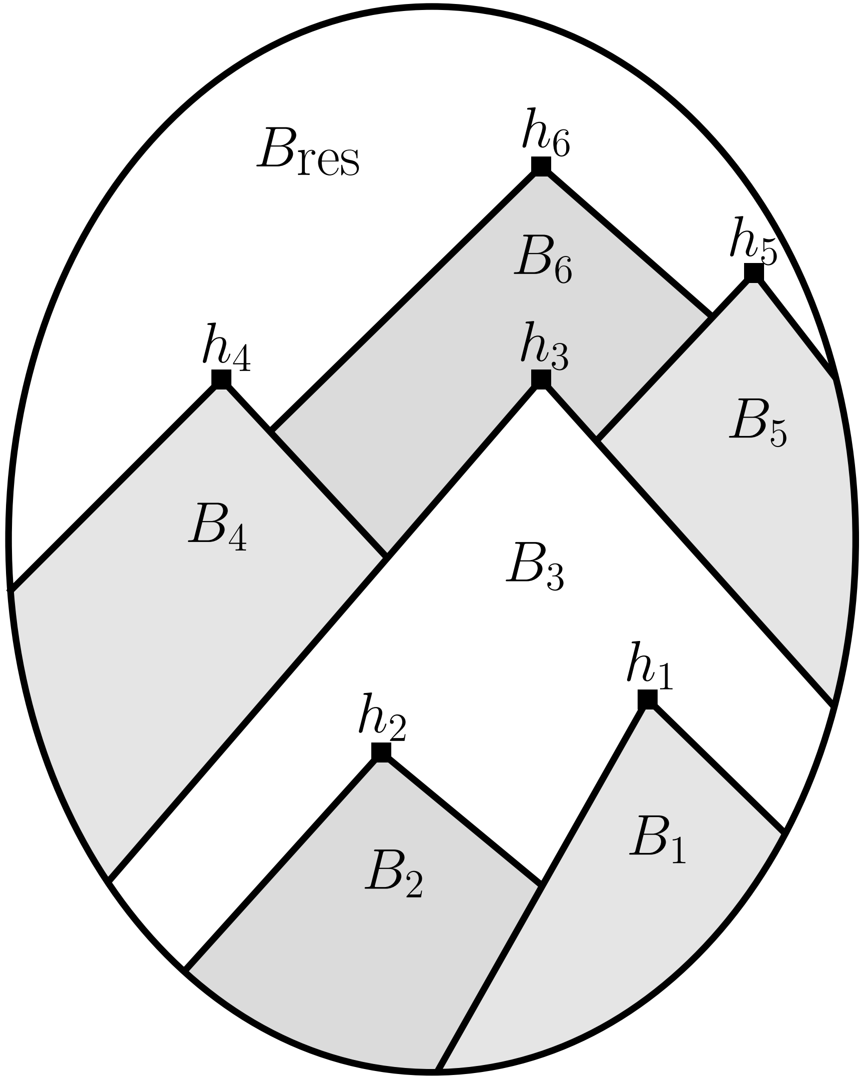


Block Decomposition

- Recurse on $L\setminus\downarrow$h to form blocks $B_1, B_2, \ldots, B_m$, and B_{res}.
- All blocks other than B_{res} have at least $\sqrt{n}$ elements.

Implication 2:

There are $O(\sqrt{n})$ blocks.



Information Stored – Order Testing

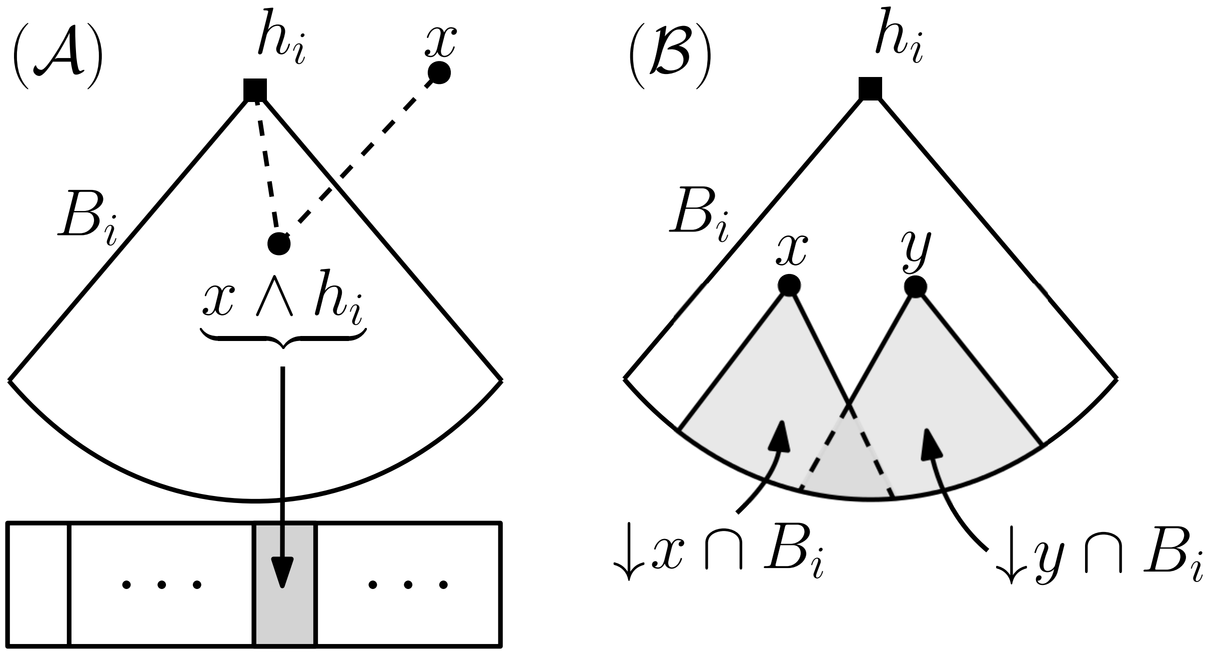
($\mathcal{A}$): For every element x, store $x \wedge h_i$, where $1 \leq i \leq m$.

($\mathcal{B}$): For an element $x \in B_i$, store $\downarrow x \cap B_i$ in a dictionary.

Space Complexity:

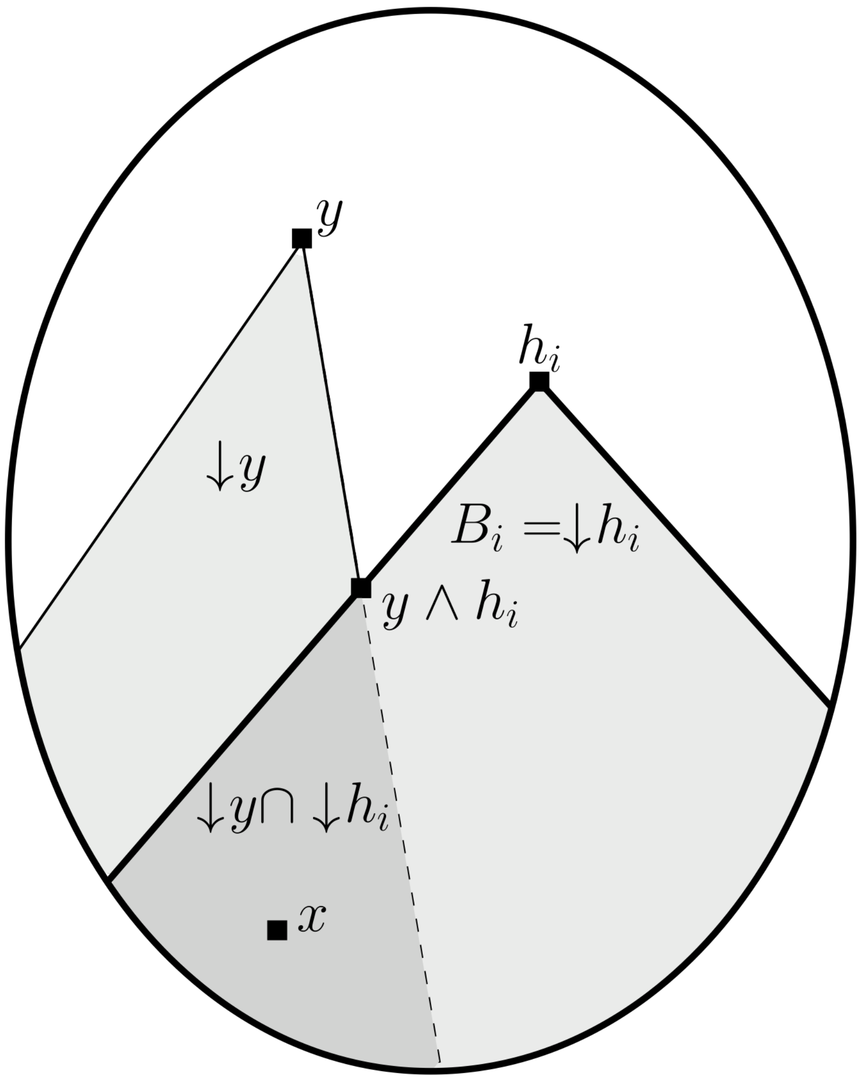
($\mathcal{A}$): $O(n\sqrt{n})$ by Implication 2.

($\mathcal{B}$): $O(n\sqrt{n})$ for $x \notin \{h_i\}$ by Implication 1; $O(n)$ for $\{h_i\}$.



Order Testing: $x \leq y$?

1. Suppose $x \in B_i$. Compute $y \wedge h_i$.
2. Report YES iff $x \in (\downarrow y \wedge h_i) \cap B_i$.

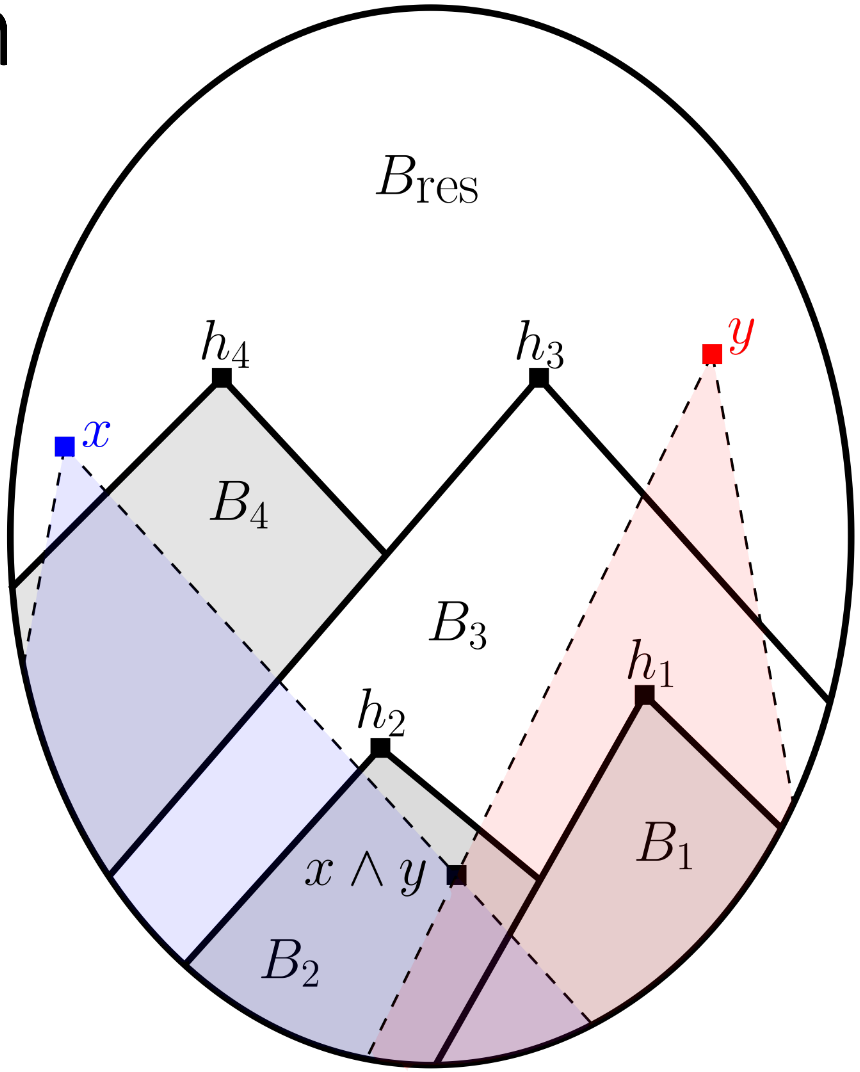


Time Complexity:

1. $O(1)$ via ($\mathcal{A}$).
2. $O(1)$ via ($\mathcal{B}$).

Meets and Joins – Intuition

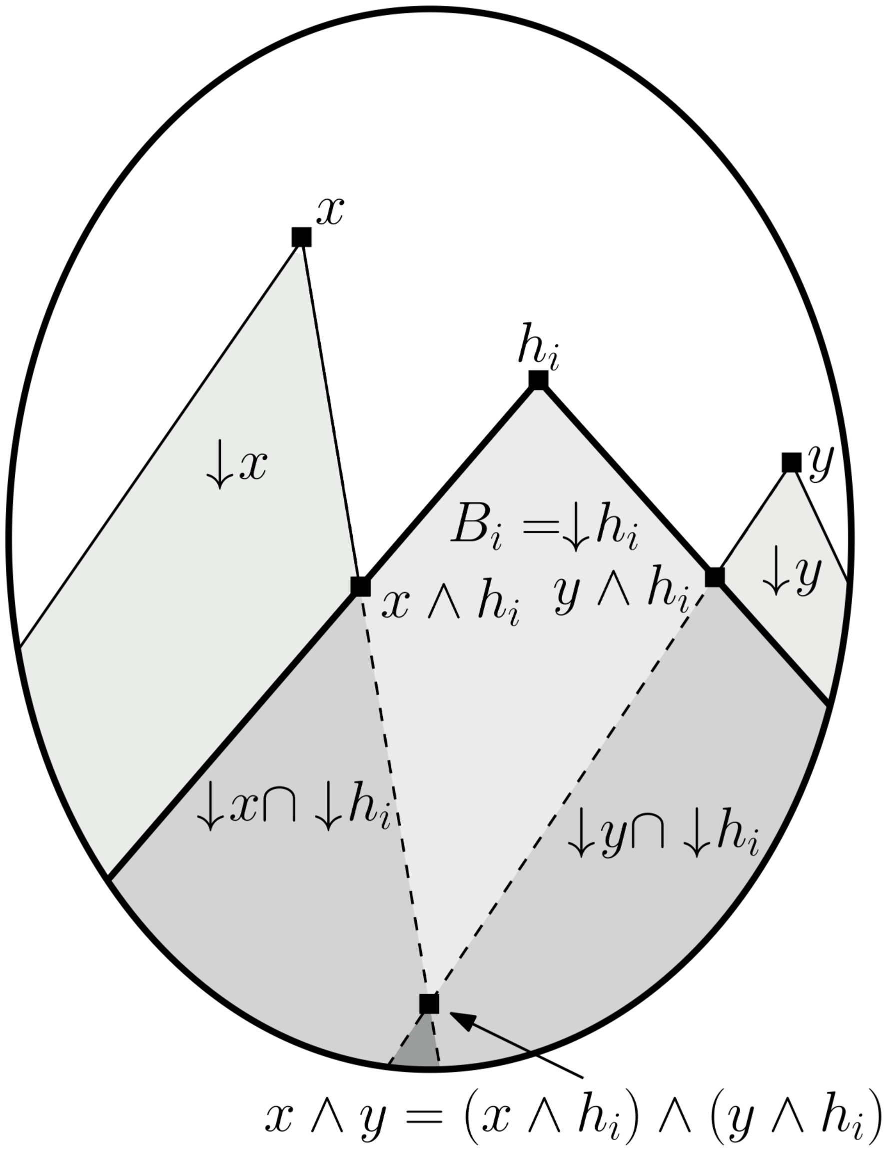
- Focus on meet.
- The meet can be in any of the $O(\sqrt{n})$ blocks. We must check them all.



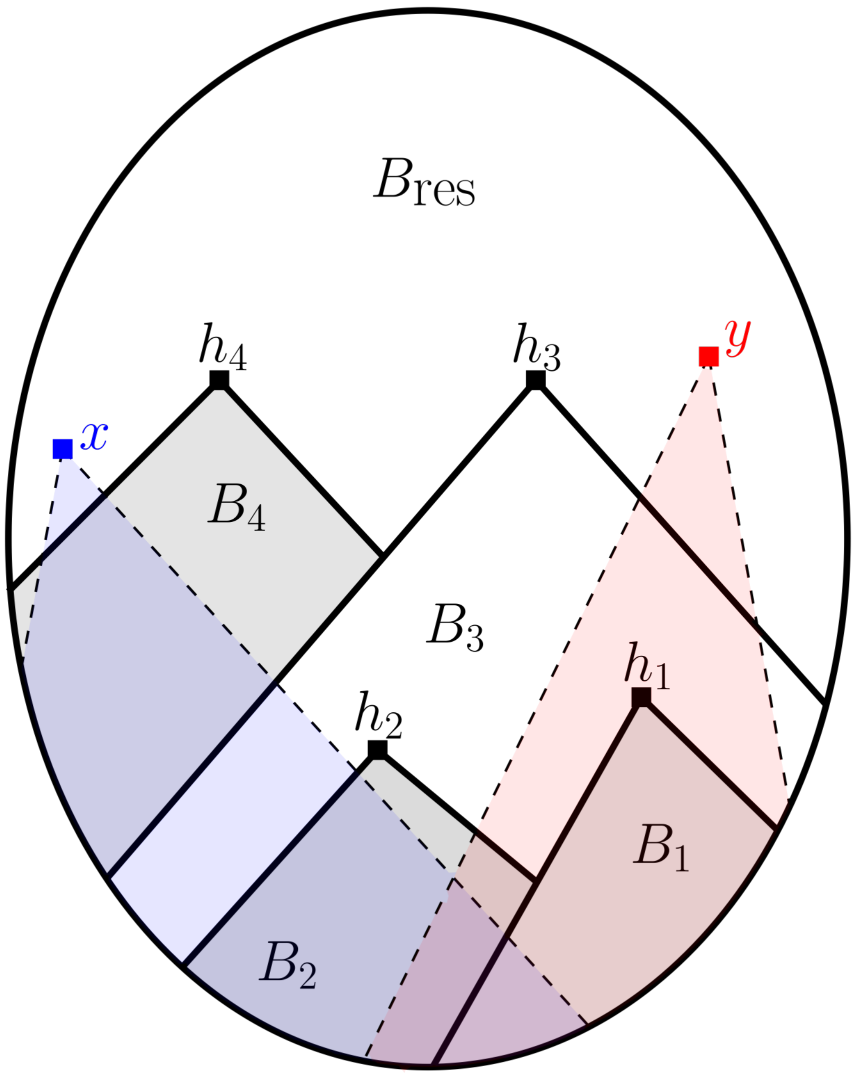
Meet

Suppose $x \wedge y \in B_i$.

Then $x \wedge y = (x \wedge h_i) \wedge (y \wedge h_i)$.

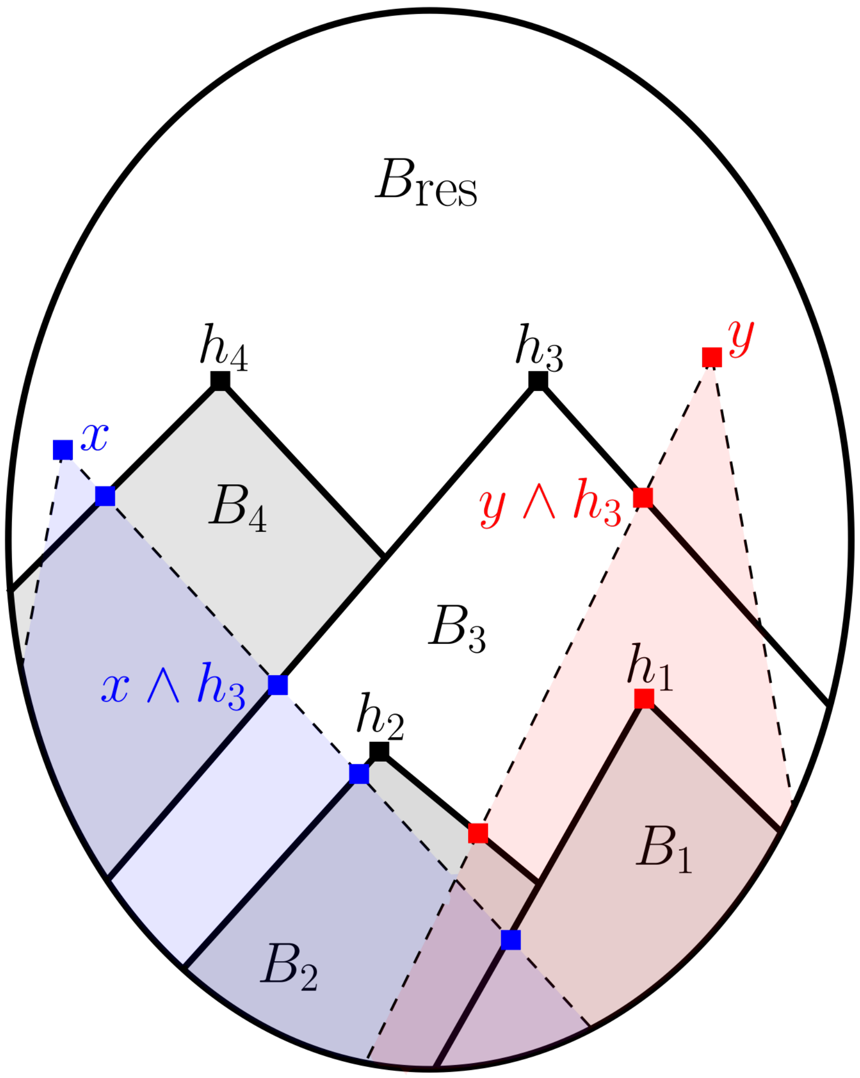


Meet Algorithm



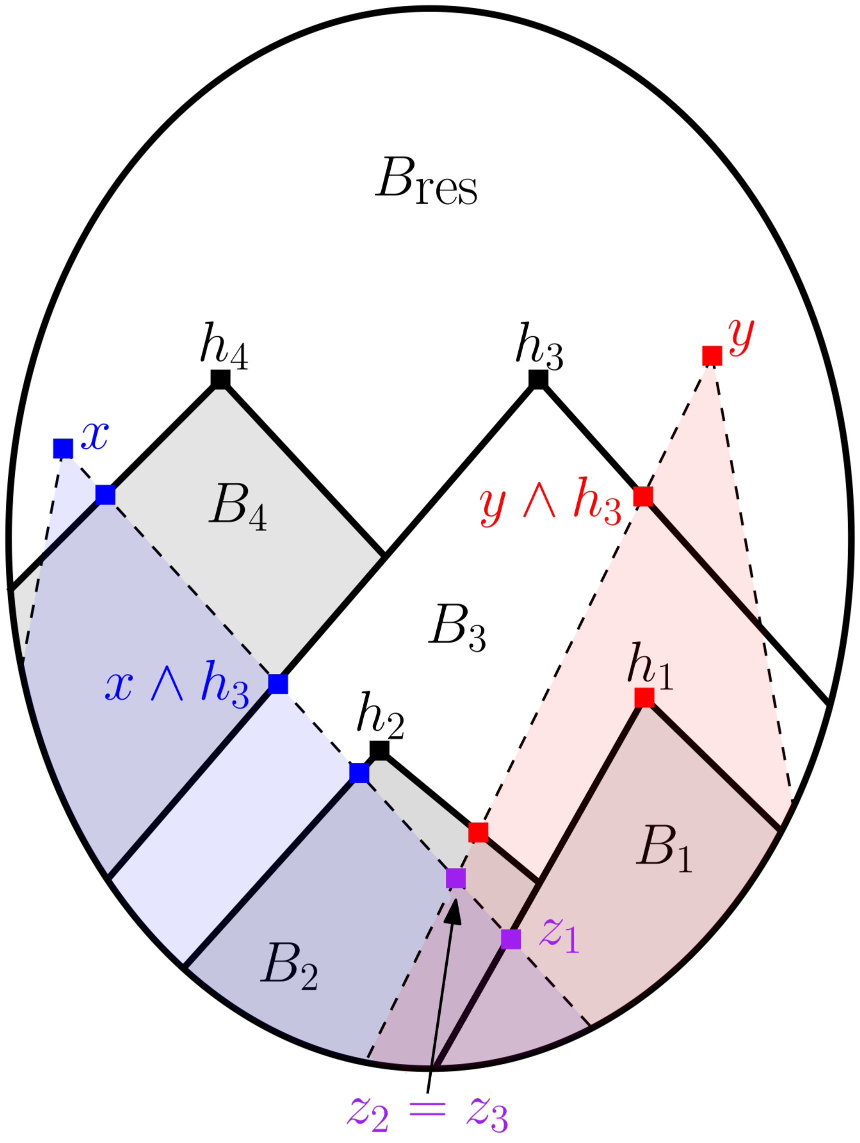
Meet Algorithm

1. Compute $x \wedge h_i$ and $y \wedge h_i$ for all i.



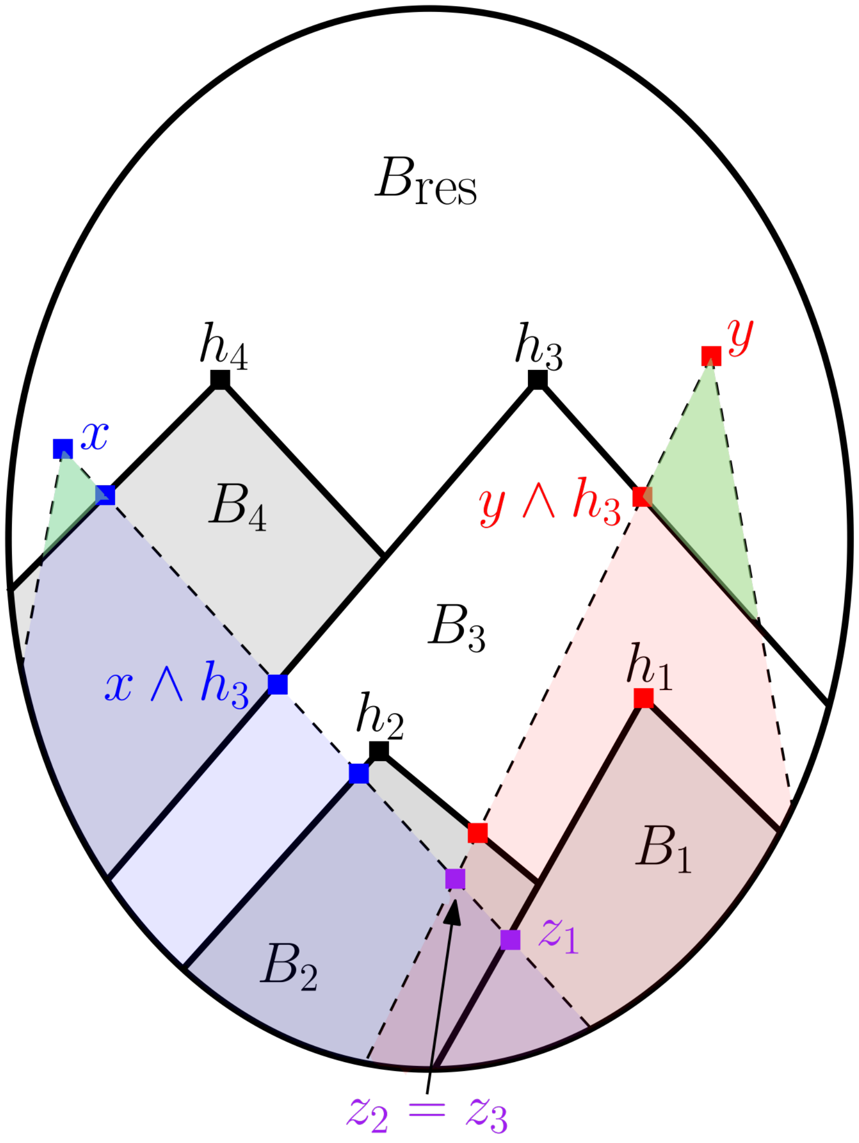
Meet Algorithm

1. Compute $x \wedge h_i$ and $y \wedge h_i$ for all i.
2. Compute $z_i = (x \wedge h_i) \wedge (y \wedge h_i)$ for all i.



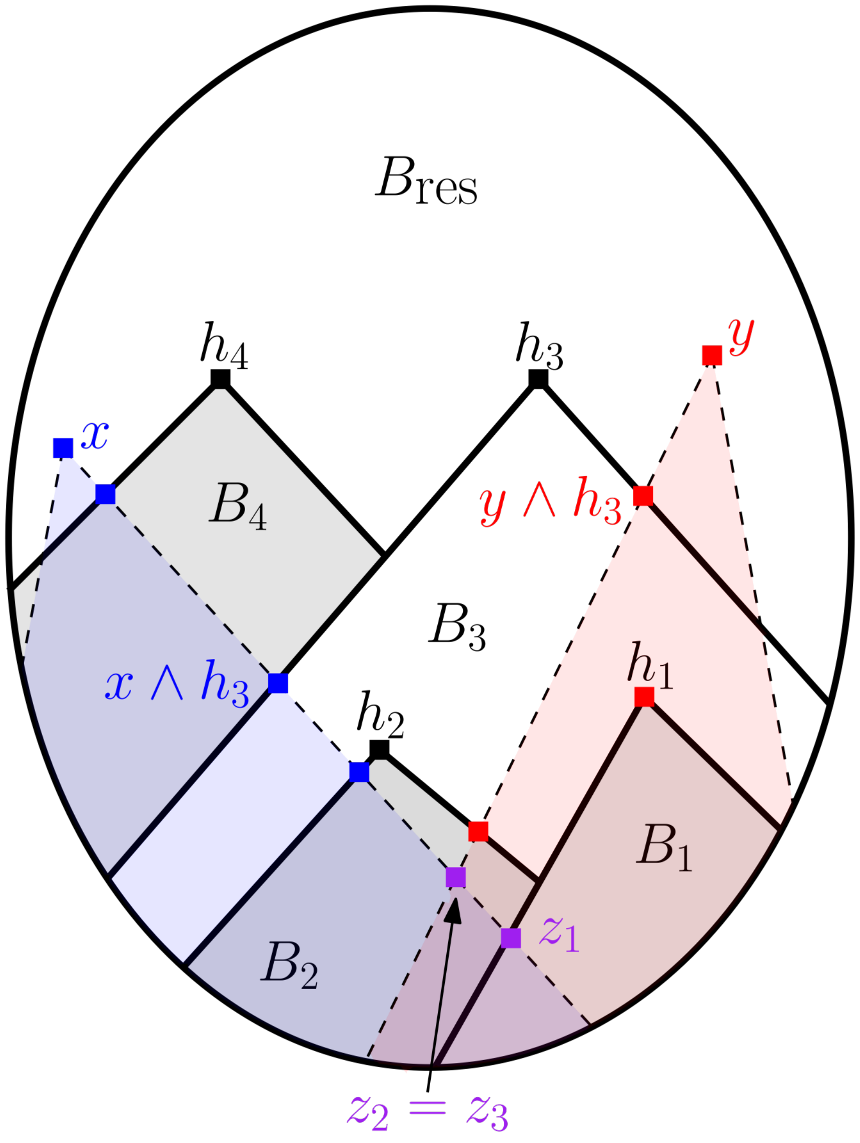
Meet Algorithm

1. Compute $x \wedge h_i$ and $y \wedge h_i$ for all i.
2. Compute $z_i = (x \wedge h_i) \wedge (y \wedge h_i)$ for all i.
3. If $x, y \in B_{res}$, find $\downarrow x \cap \downarrow y \cap B_{res}$.



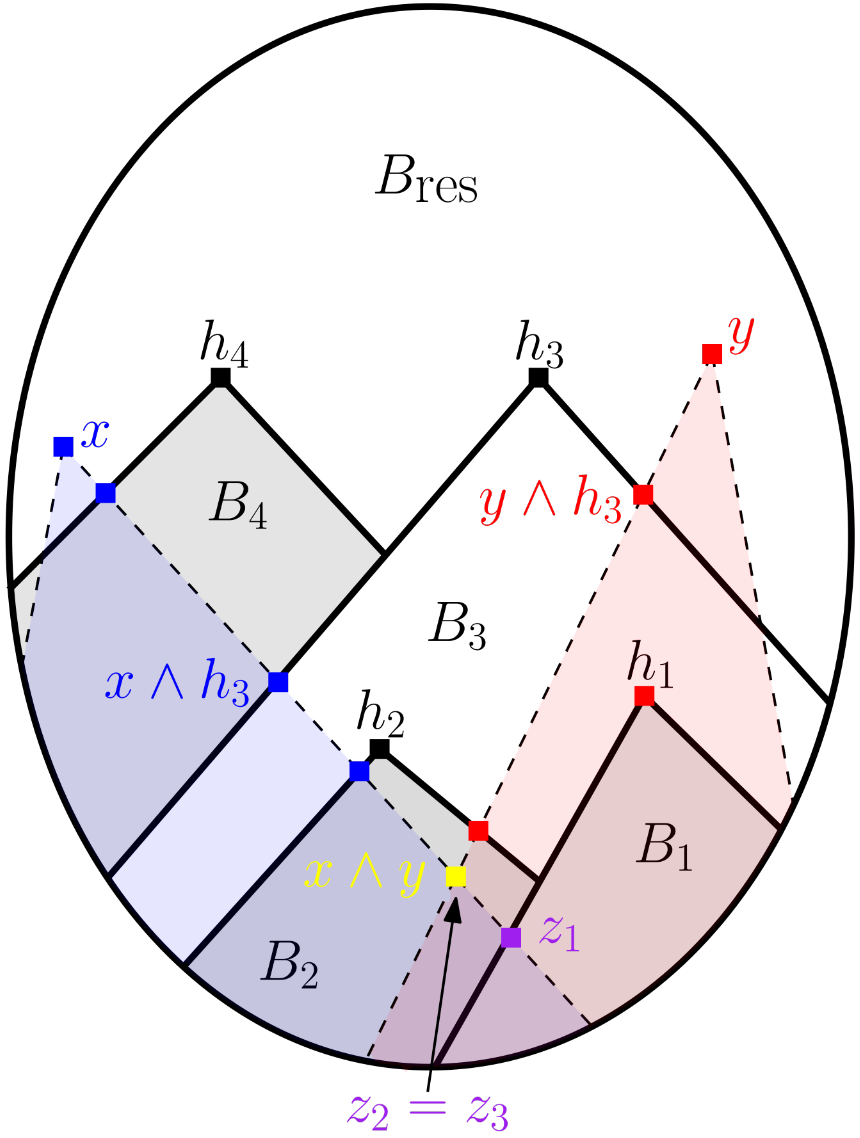
Meet Algorithm

1. Compute $x \wedge h_i$ and $y \wedge h_i$ for all i.
2. Compute $z_i = (x \wedge h_i) \wedge (y \wedge h_i)$ for all i.
3. If $x, y \in B_{res}$, find $\downarrow x \cap \downarrow y \cap B_{res}$.



Meet Algorithm

1. Compute $x \wedge h_i$ and $y \wedge h_i$ for all i.
2. Compute $z_i = (x \wedge h_i) \wedge (y \wedge h_i)$ for all i.
3. If $x, y \in B_{res}$, find $\downarrow x \cap \downarrow y \cap B_{res}$.
4. Return the largest element found in 2 or 3.



Time Complexity:

1. $O(\sqrt{n})$ by Implication 2 and $(\mathcal{A})$.
2. TBD.
3. $O(\sqrt{n})$ by Implication 1 and $(\mathcal{B})$.
4. $O(\sqrt{n})$.

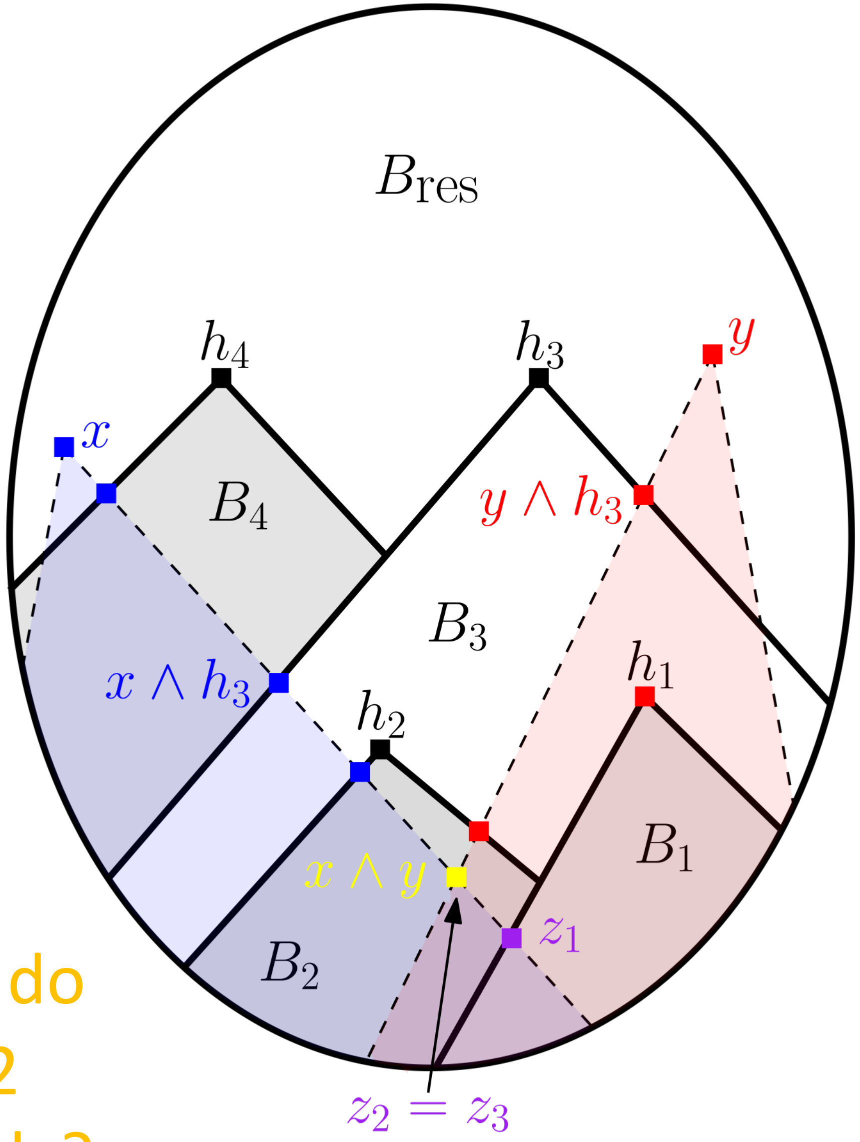
Meet Algorithm

1. Compute $x \wedge h_i$ and $y \wedge h_i$ for all i.
2. Compute $z_i = (x \wedge h_i) \wedge (y \wedge h_i)$ for all i.
3. If $x, y \in B_{res}$, find $\downarrow x \cap \downarrow y \cap B_{res}$.
4. Return the largest element found in 2 or 3.

Time Complexity:

1. $O(\sqrt{n})$ by Implication 2 and ($\mathcal{A}$).
2. TBD.
3. $O(\sqrt{n})$ by Implication 1 and ($\mathcal{B}$).
4. $O(\sqrt{n})$.

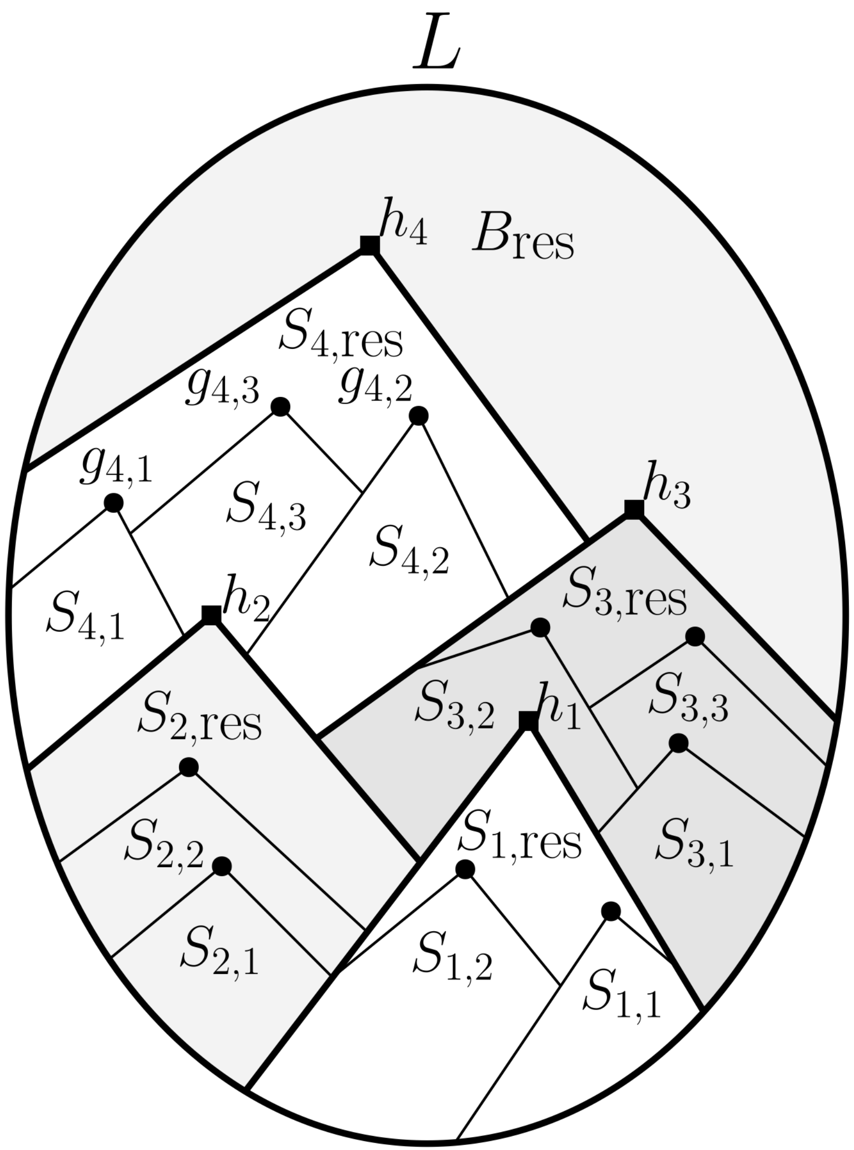
How to do step 2 efficiently?



Recurse!

- Subdivide each block B_i into subblocks $S_{i,1}, S_{i,2}, \ldots, S_{i,\ell_i}$ and residual subblock $S_{i,res}$.
- Choose subblock size $\sqrt{|B_i|}$.

($\mathcal{C}$): Store the meets of all pairs of elements in each $S_{i,j}$.



Space Complexity:

Recursion: $O(\sum_i |B_i|^{1.5}) = O(n\sqrt{n})$.

($\mathcal{C}$): $O\left(\sum_{i,j} |S_{i,j}|^2\right) = O(n\sqrt{n})$ by Implication 1.

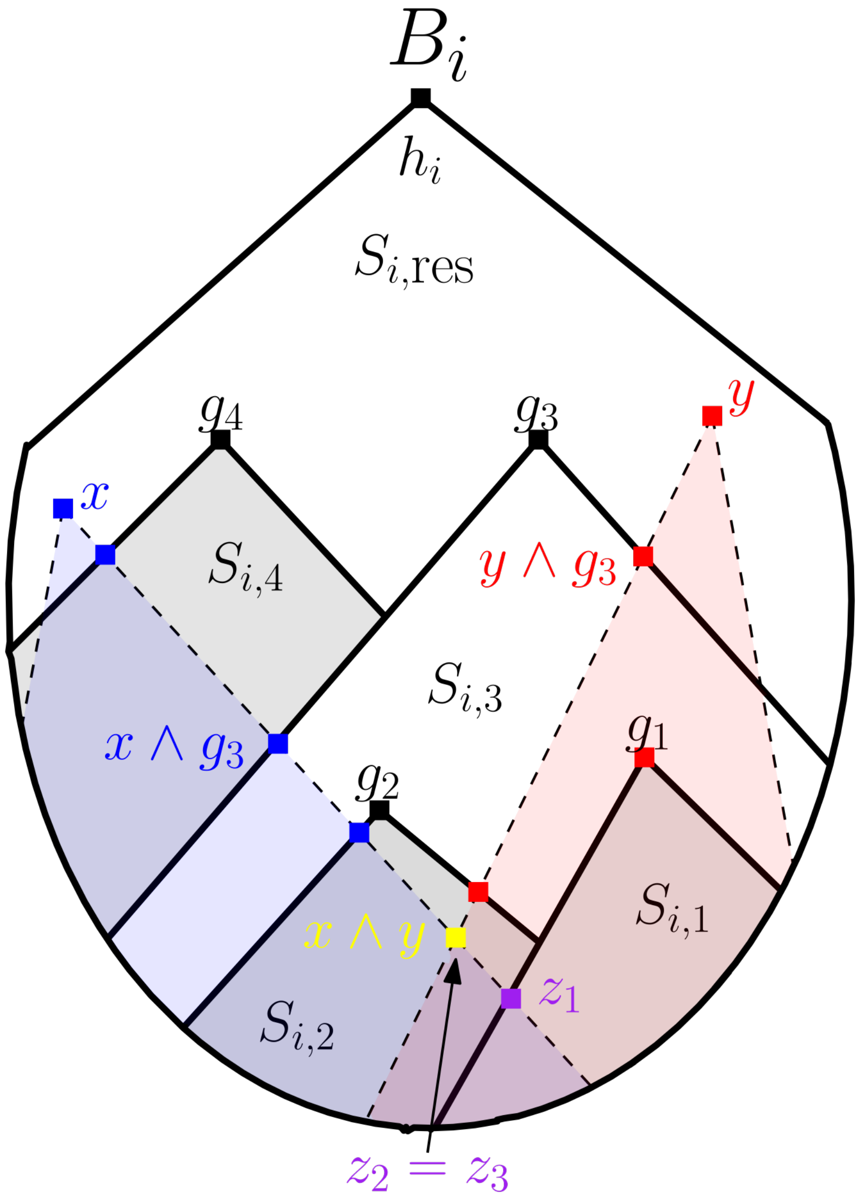
Meet Time Complexity

1. Compute $x \wedge g_i$ and $y \wedge g_i$ for all i.
2. Compute $z_i = (x \wedge g_i) \wedge (y \wedge g_i)$ for all i.
3. If $x, y \in S_{i,res}$, find $\downarrow x \cap \downarrow y \cap S_{i,res}$.
4. Return the largest element found in 2 or 3.

Meet in block B_i:

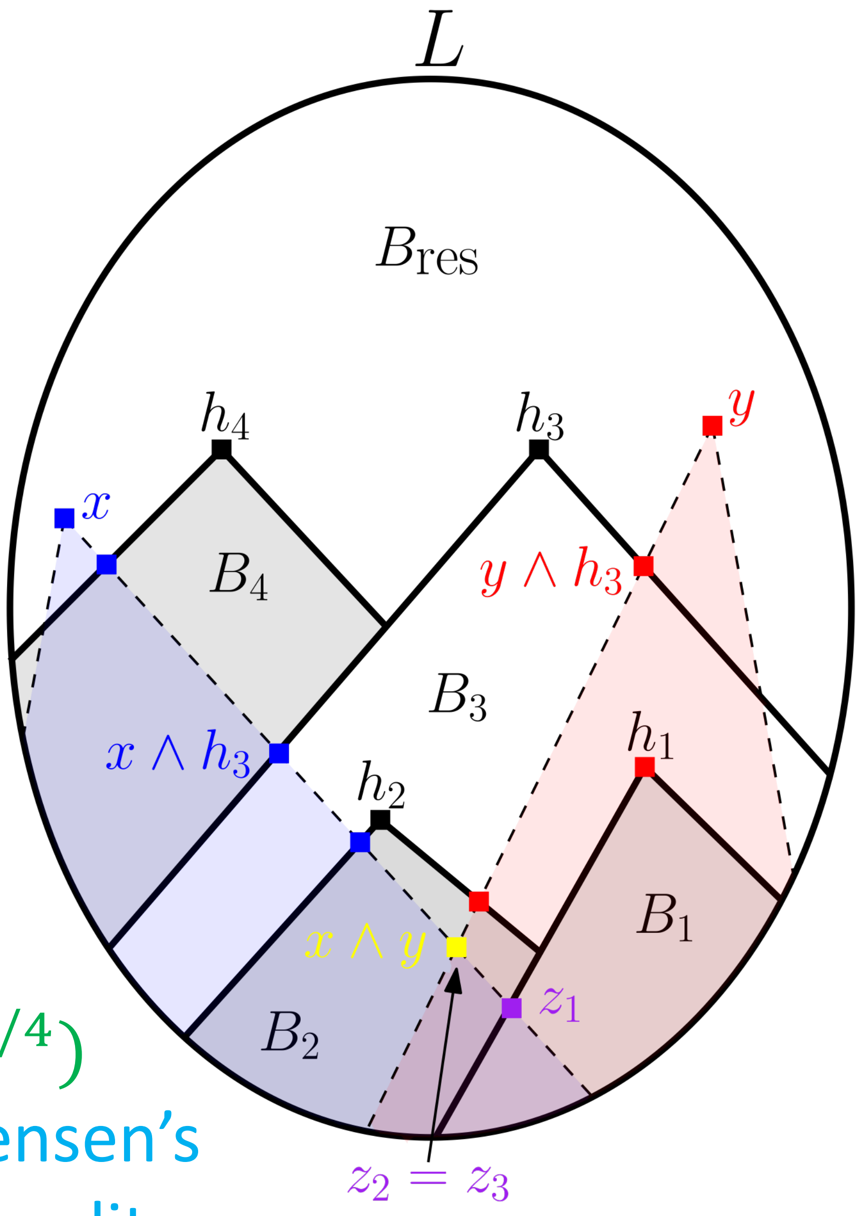
1. $O(\sqrt{|B_i|})$.
2. $O(\sqrt{|B_i|})$ via $(\mathcal{C})$.
3. $O(\sqrt{|B_i|})$.
4. $O(\sqrt{|B_i|})$.

Meet in L:



Meet Time Complexity

1. Compute $x \wedge h_i$ and $y \wedge h_i$ for all i.
2. Compute $z_i = (x \wedge h_i) \wedge (y \wedge h_i)$ for all i.
3. If $x, y \in B_{res}$, find $\downarrow x \cap \downarrow y \cap B_{res}$.
4. Return the largest element found in 2 or 3.



Meet in block B_i:

1. $O(\sqrt{|B_i|})$.
2. $O(\sqrt{|B_i|})$ via ($\mathcal{C}$).
3. $O(\sqrt{|B_i|})$.
4. $O(\sqrt{|B_i|})$.

Meet in L:

1. $O(\sqrt{n})$ via ($\mathcal{A}$).
2. $O\left(\sum_i \sqrt{|B_i|}\right) = O(n^{3/4})$ by Jensen's Inequality
3. $O(\sqrt{n})$ via ($\mathcal{B}$).
4. $O(\sqrt{n})$.

Thanks!

Questions?