Discovering Descriptive Tile Trees

Nikolaj Tatti, Jilles Vreeken

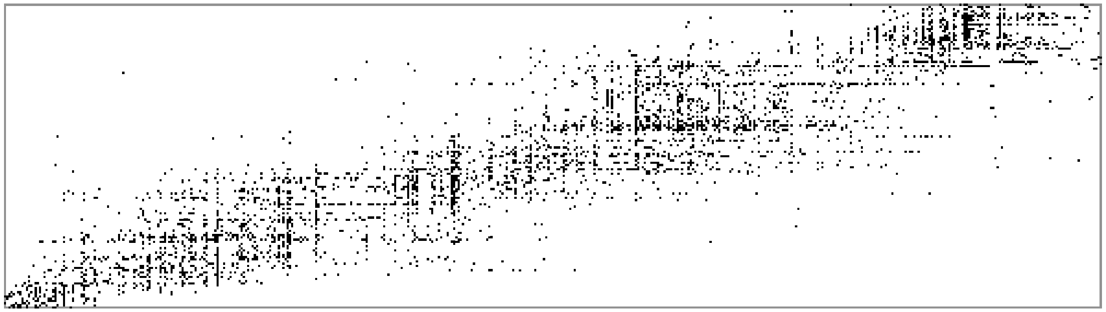
nikolaj.tatti@ua.ac.be, jilles.vreeken@ua.ac.be

ADReM, University of Antwerp, Belgium

What do we want

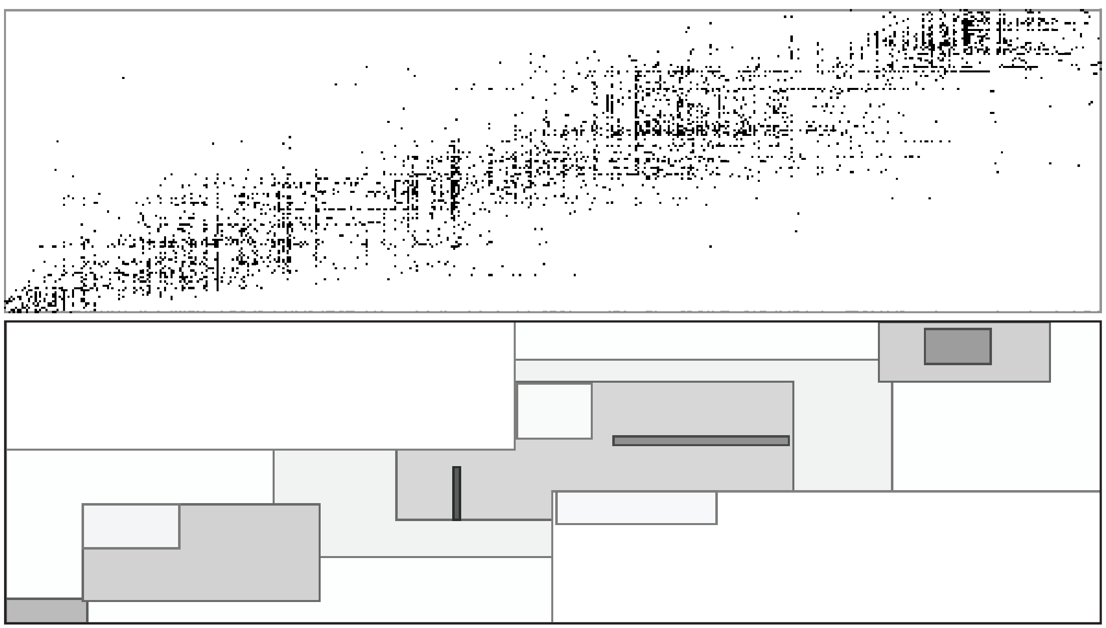
Example

- binary data of fossils
- 139 genera, 501 sites
- $D(x, y)$ is 1 if fossil x was found at site y
- sites and genera has a natural order, time



What do we want

- we want to describe where are the 1s / 0s
- we describe the distribution with tiles
- we allow exception tiles within tiles (and exceptions of exceptions..)



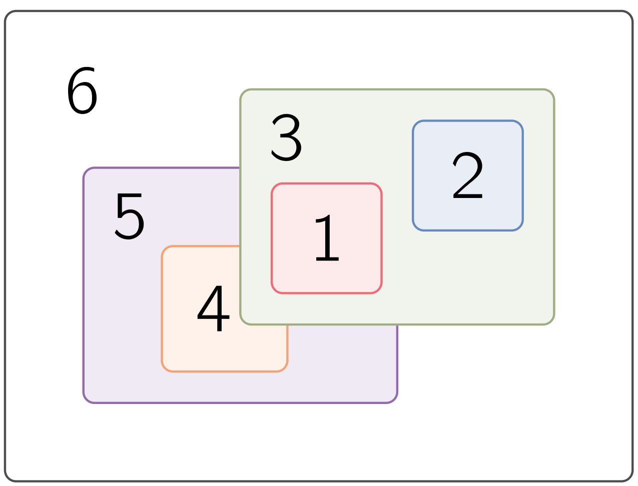
What's the catch

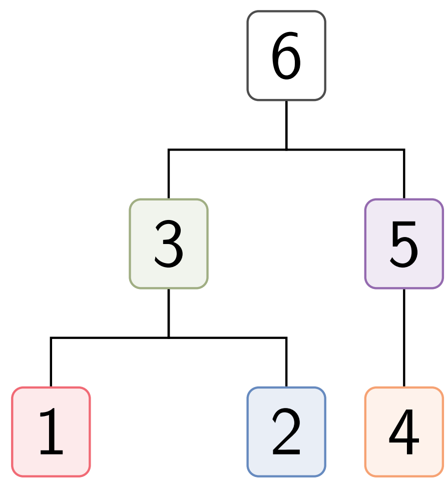
- data needs to be ordered, both features and data points
- not all data have meaningful orders...
  - ...but if does, it is silly not to use it
- if order is not known, we can find it with spectral techniques
- in return we get
  - polynomial algorithms
  - visualization techniques

Tile trees

Tile trees

- a tile is a consecutive submatrix of data
- tile tree is a tree of tiles (children are ordered)
- a child is a subtile of its parent
- root contains the whole data
- children and earlier siblings cover the data first





Scoring tile trees

- each tile is represented by a Bernoulli random variable *(Gionis et al. 2004)*
- negative log-likelihood + MDL to control overfitting
- prefers
  - sparse or dense tiles
  - tiles that are significantly different from their parent tiles

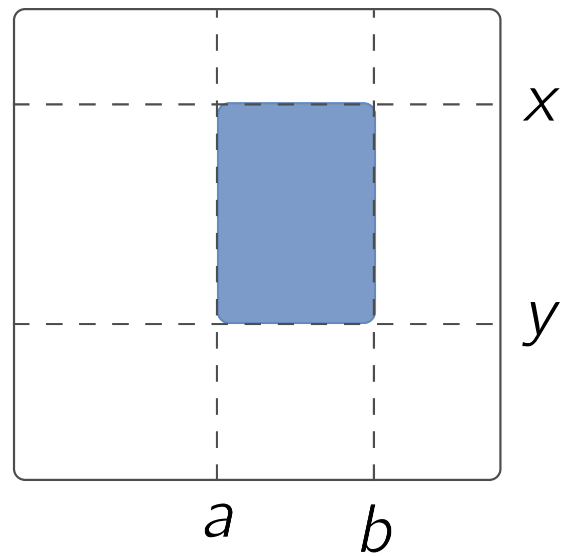
Finding a good tree

- the search space contains all possible tile trees
- we cannot enumerate all trees
- doesn't seem to have any structure that we can exploit
- we resort to greedy heuristics
  - find the optimal subtile
  - recurse until score cannot improve

Finding optimal subtile

Naive approach

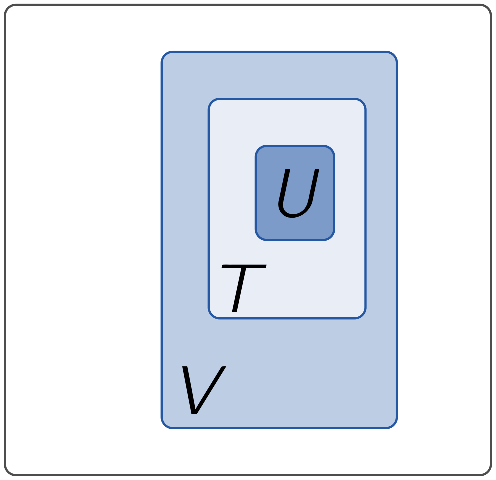
```
foreach b = 1, ..., N do
    foreach a = 1, ..., b do
        foreach y = 1, ..., M do
            foreach x = 1, ..., y do
                test tile (x, y) × (a, b);
```



- needs $\Theta(M^2 N^2)$ time
- we can improve this to $\Theta(NM \min(N, M))$ time by replacing the last for-loop

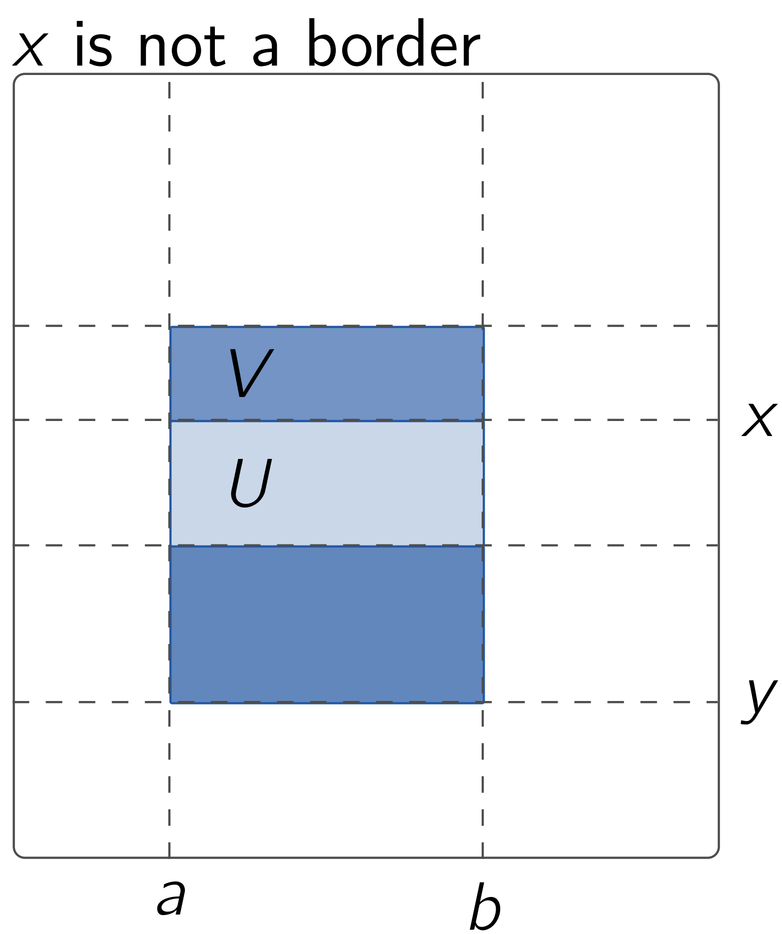
Key Theorem

- let T be a dense tile
- let $U \subset T$ be a subtile
- let $T \subset V$ be a supertile
- if $dens(V \setminus T) \geq dens(T \setminus U)$
  → either U or V is as good as T
  → ignore T

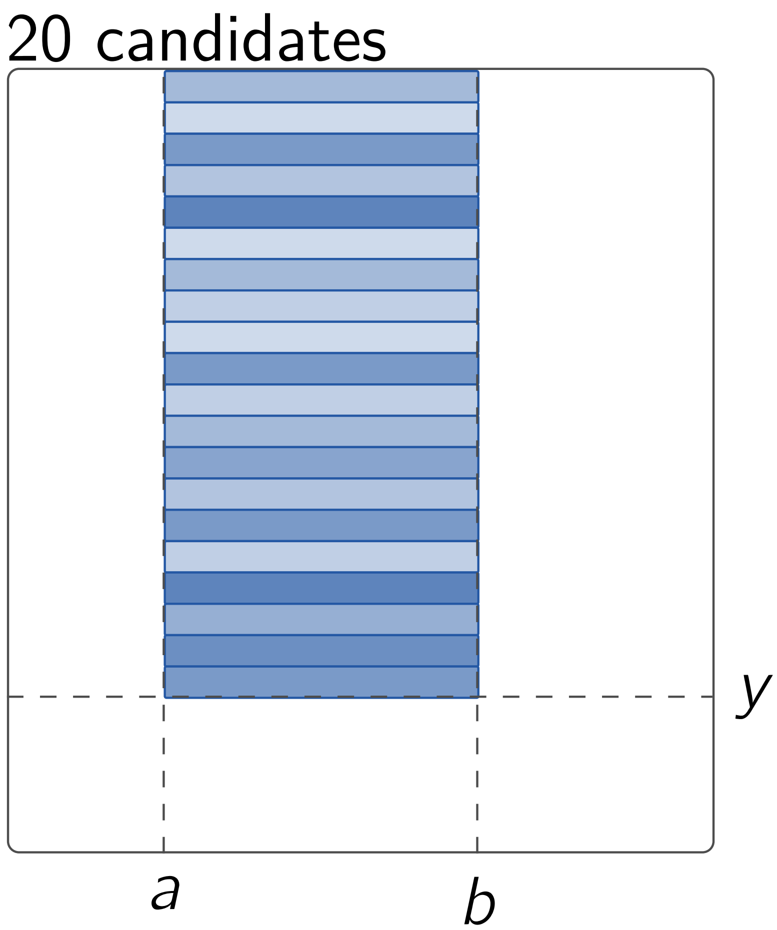


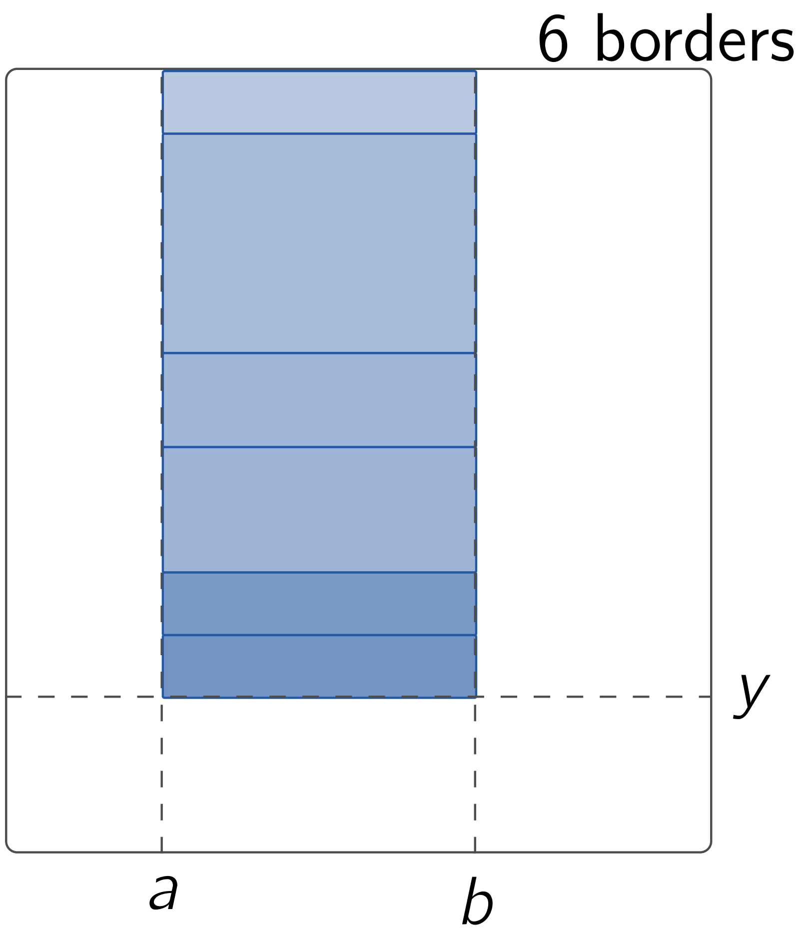
Borders

- fix a, b
- x is not a border of y if $dens(V) \geq dens(U)$
- we only need to check $x \in borders(y)$



Borders



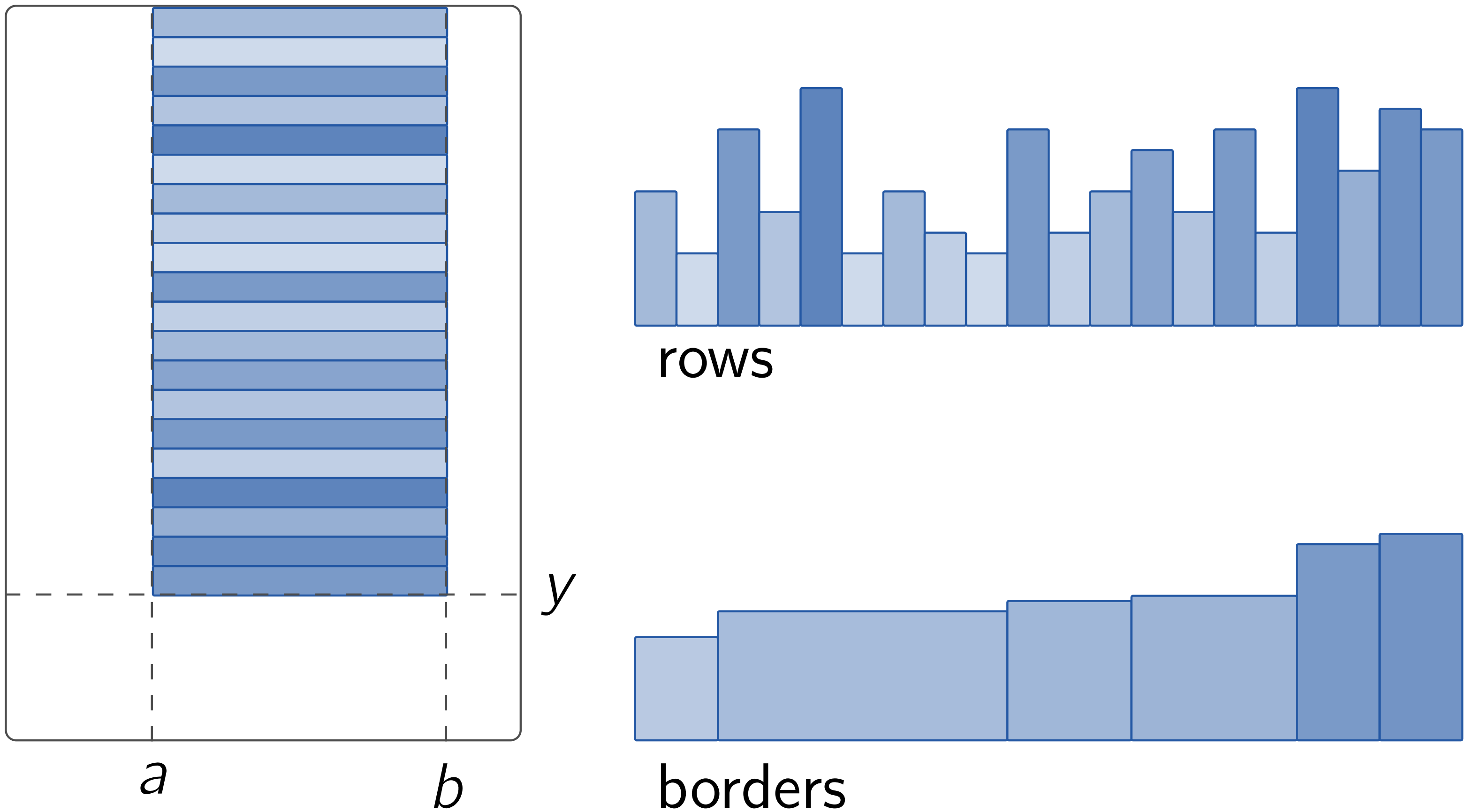


Updating borders

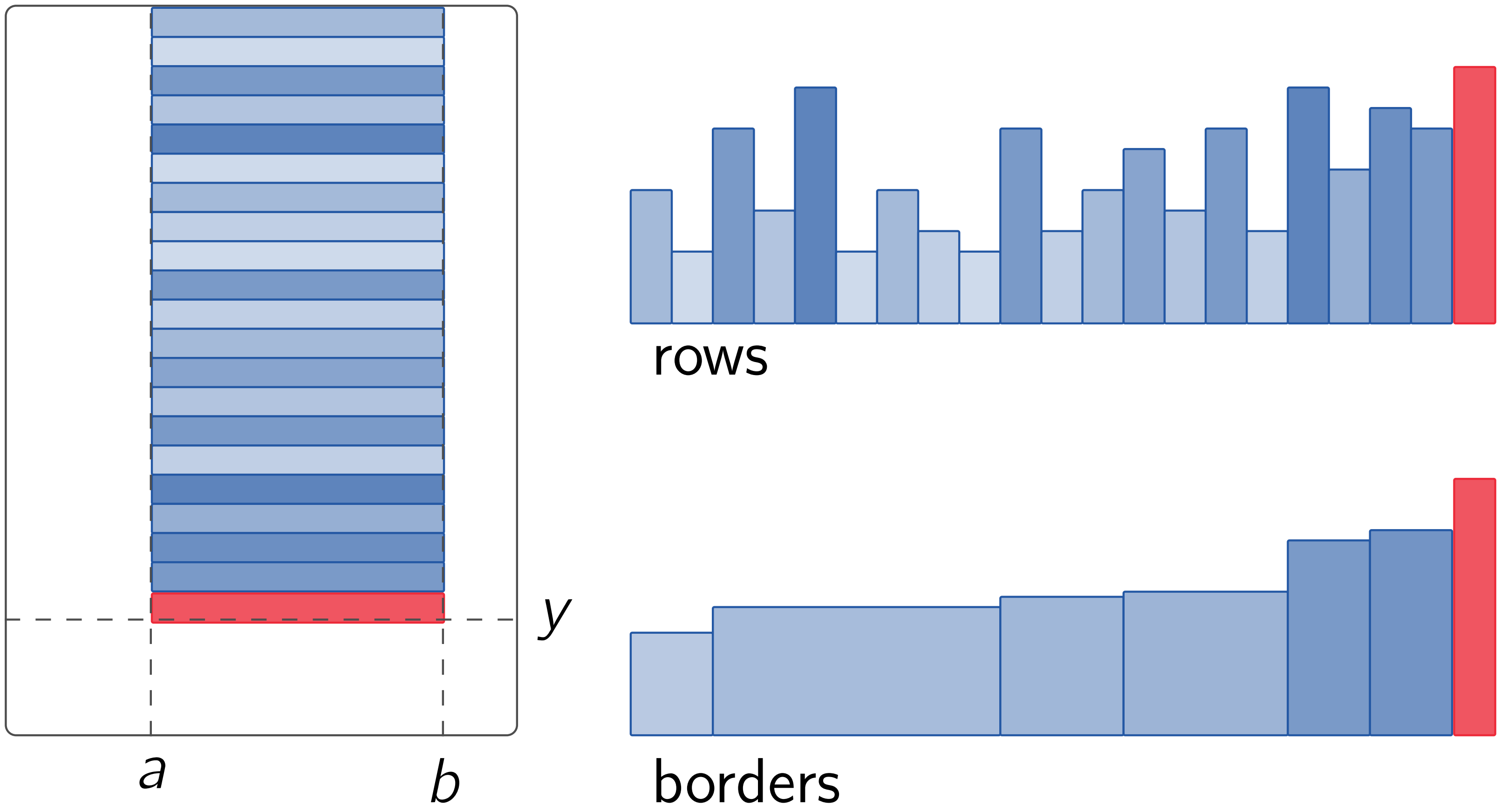
- to compute *borders*(y) from *borders*$(y-1)$ *(Calders et al. 2007)*

add yth row of to the borders;
while $\mathit{dens}(\mathit{last\ tile}) \leq \mathit{dens}(\mathit{2nd\ last\ tile})$ **do**
  join last and 2nd last tiles;

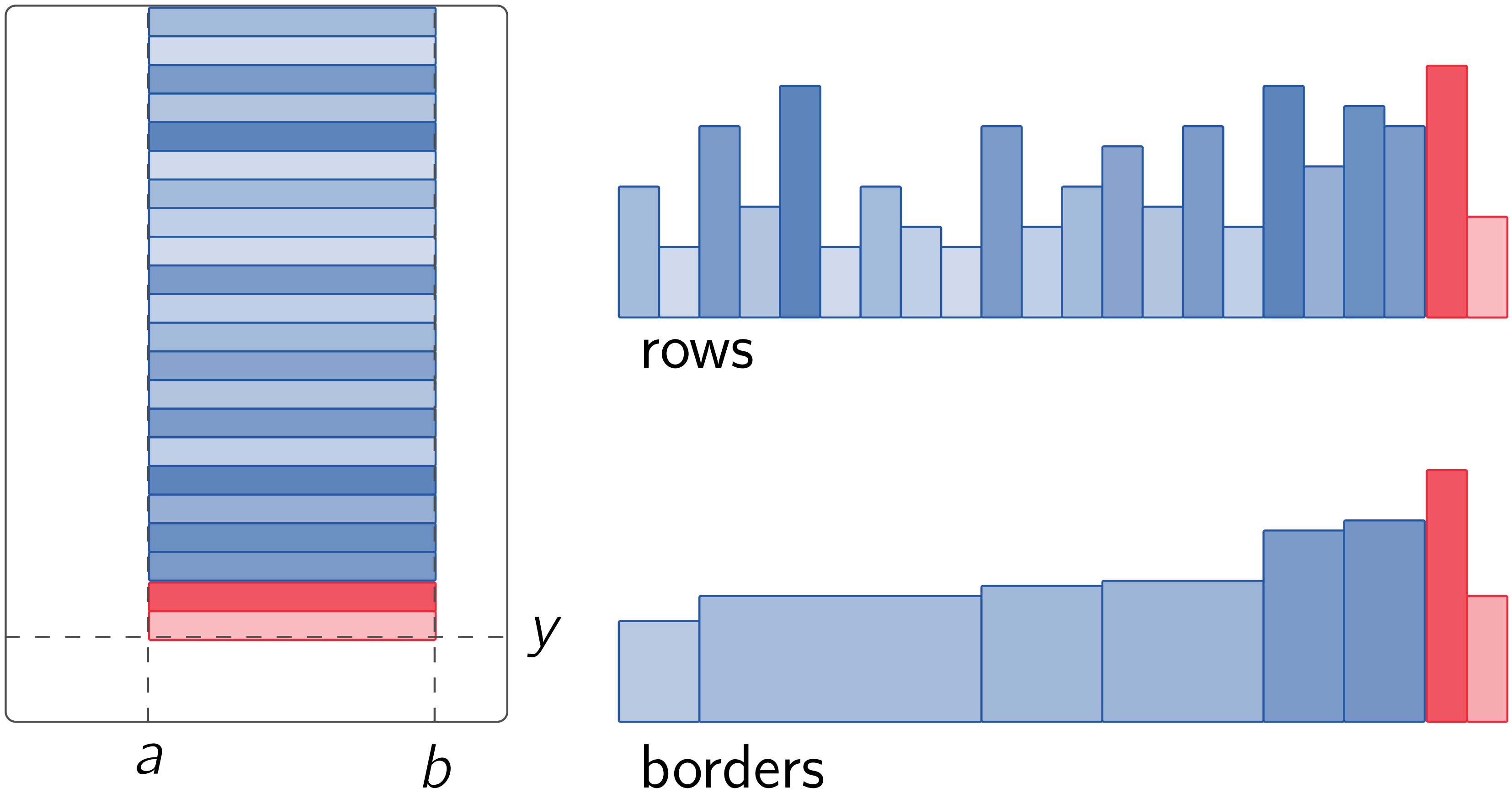
Updating borders



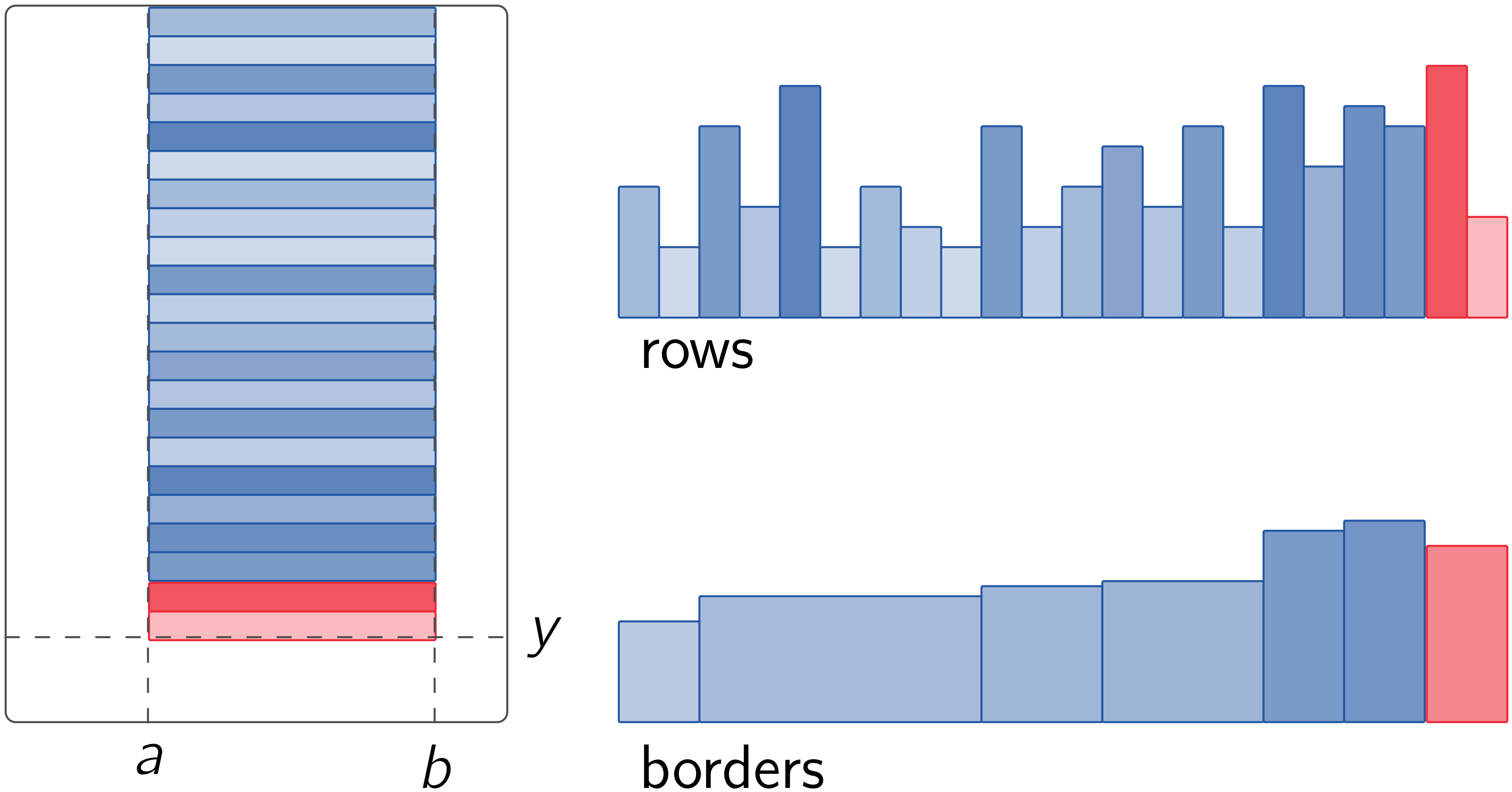
Updating borders



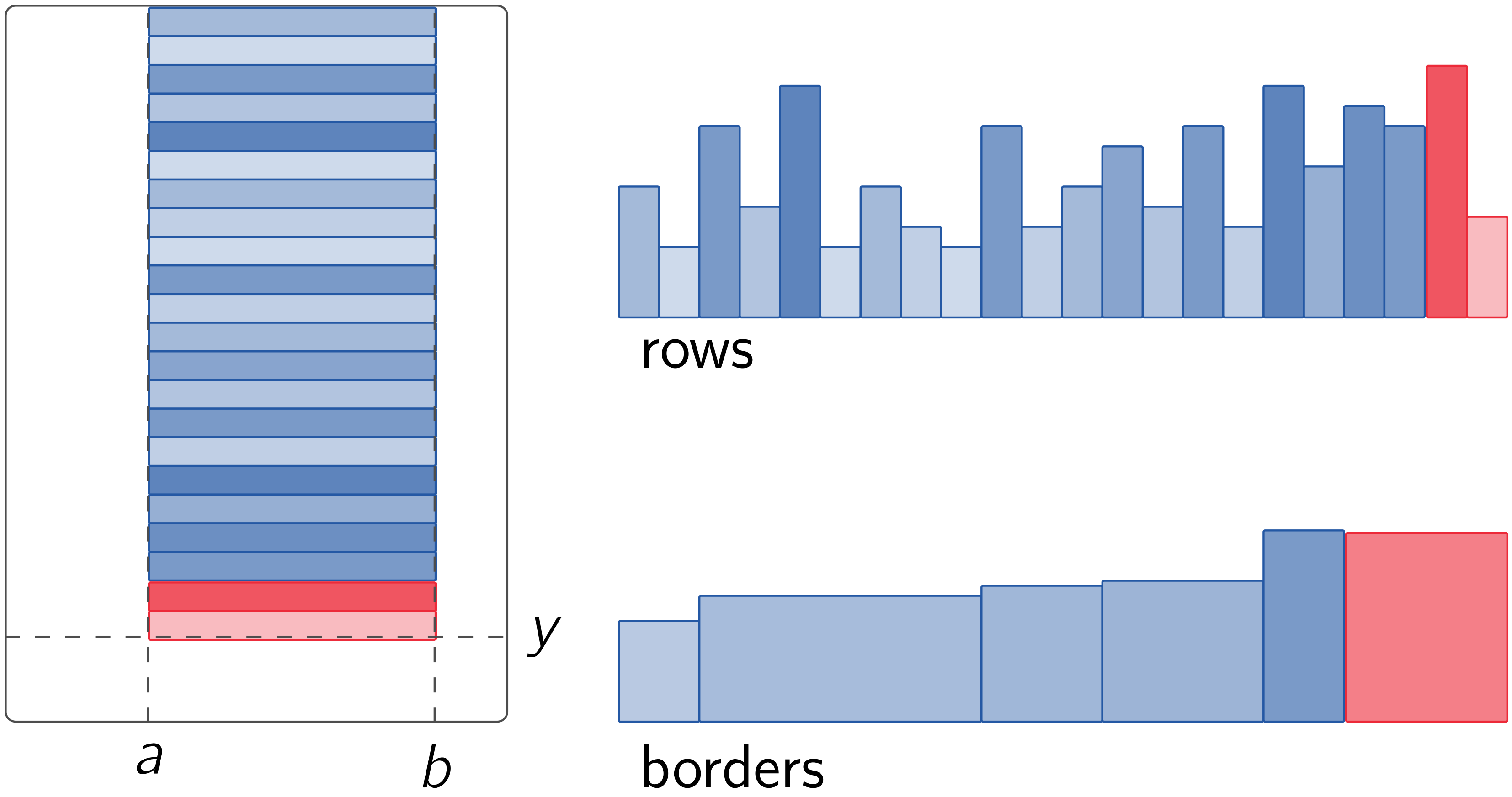
Updating borders



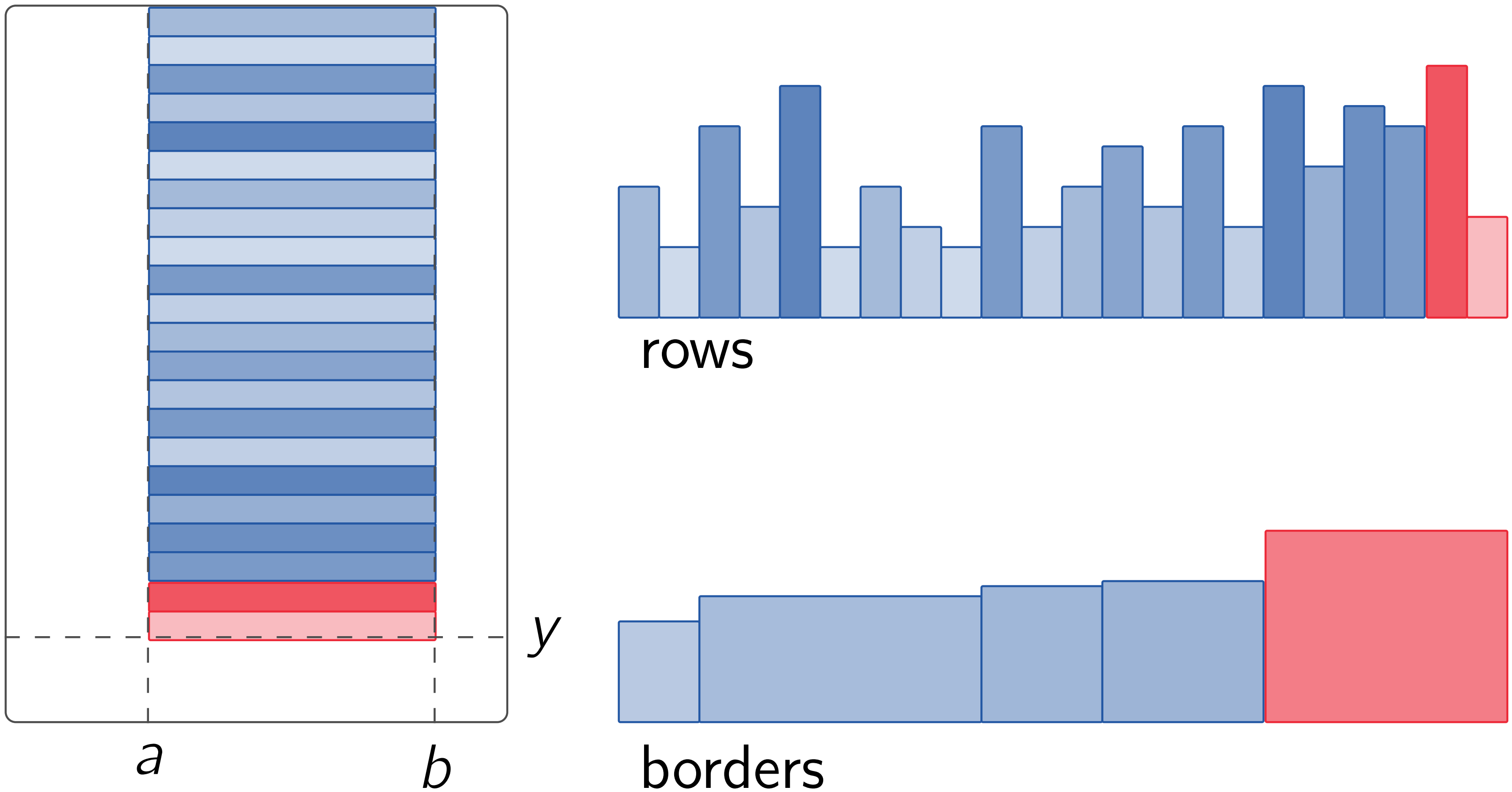
Updating borders



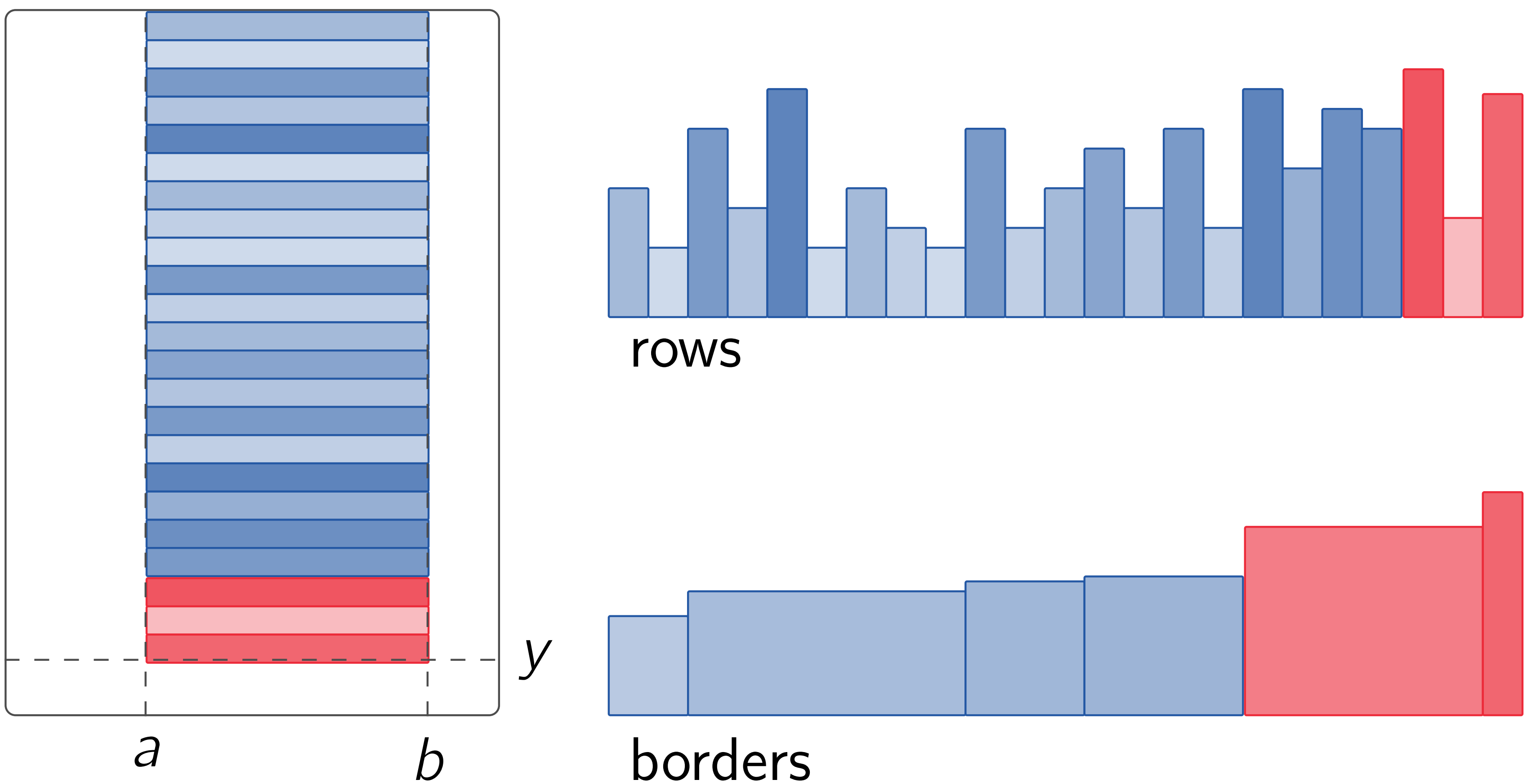
Updating borders



Updating borders



Updating borders



Computational Complexity

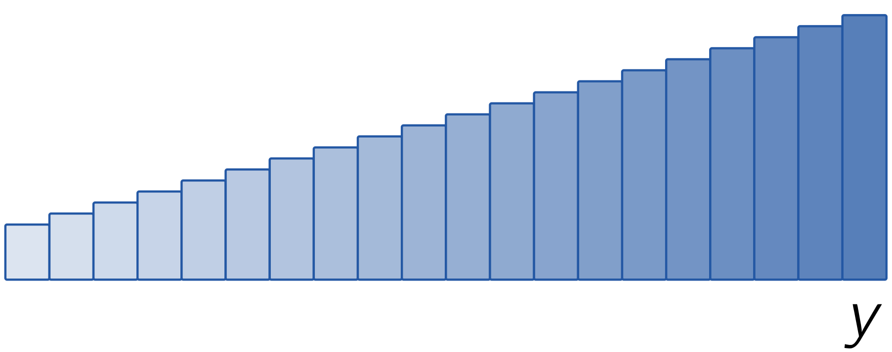
- updating $borders(y)$ takes K_y iterations
- update deletes K_y indices
- once index is deleted it will never appear again,

$$\sum_{y=1}^{M} K_y \leq M$$

- total computational complexity is $\Theta(M)$
- amortized computational complexity is $\Theta(1)$

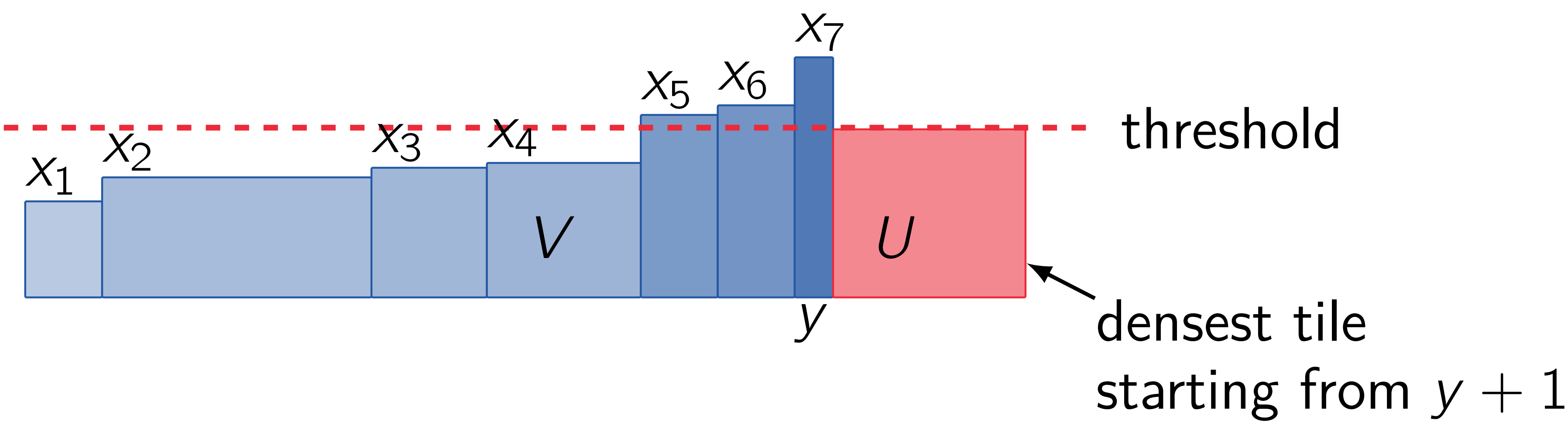
Borders are not enough

- updating borders can be done in constant time
- we can have $|borders(y)| = y$
- checking $(x, y) \times (a, b)$ for every $x \in borders(y)$ is not enough



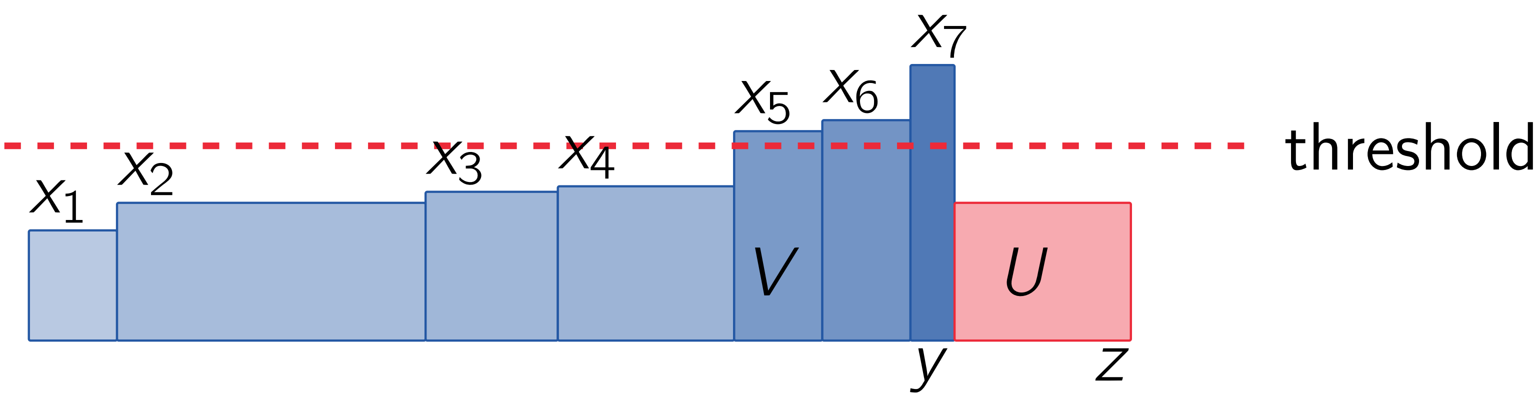
Pruning borders

- $dens(U) > dens(V)$, we can ignore x_4
- monotonicity implies that we can ignore $x_1 - x_4$



Pruning borders further

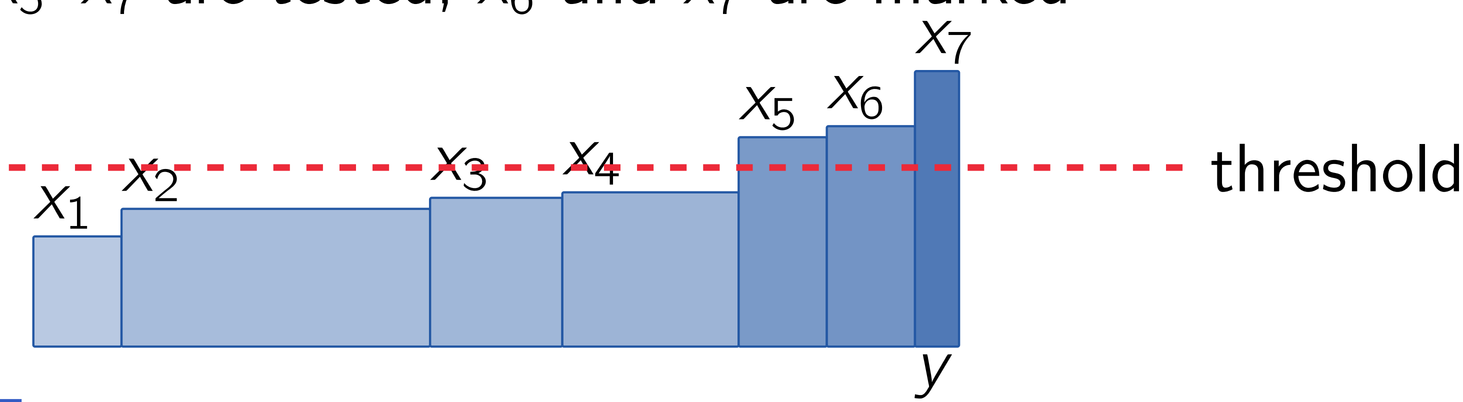
- for any U, $dens(U) < dens(V)$
- ignore x_6 (and x_7) after y



Test algorithm

```
foreach unmarked x ∈ borders(y) do
    if if the block x is too sparse then
        Break;
    test (x, y) × (a, b);
mark all but last tested borders;
```

x_5–x_7 are tested, x_6 and x_7 are marked



Computational Complexity

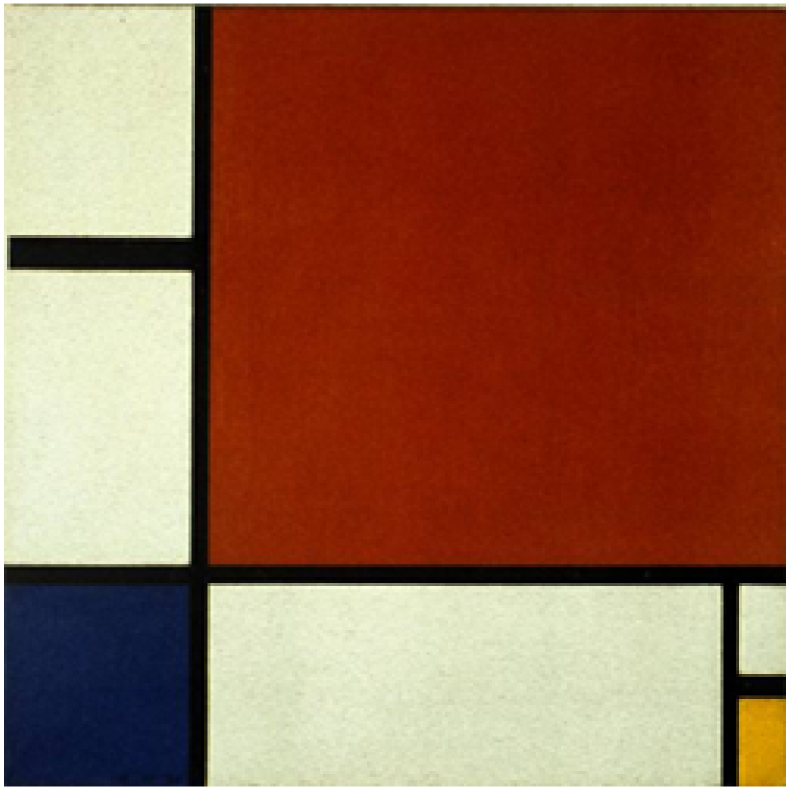
- we test L_y tiles for each y
- during this we mark $L_y - 1$ borders
- once border is marked, it will not be tested,

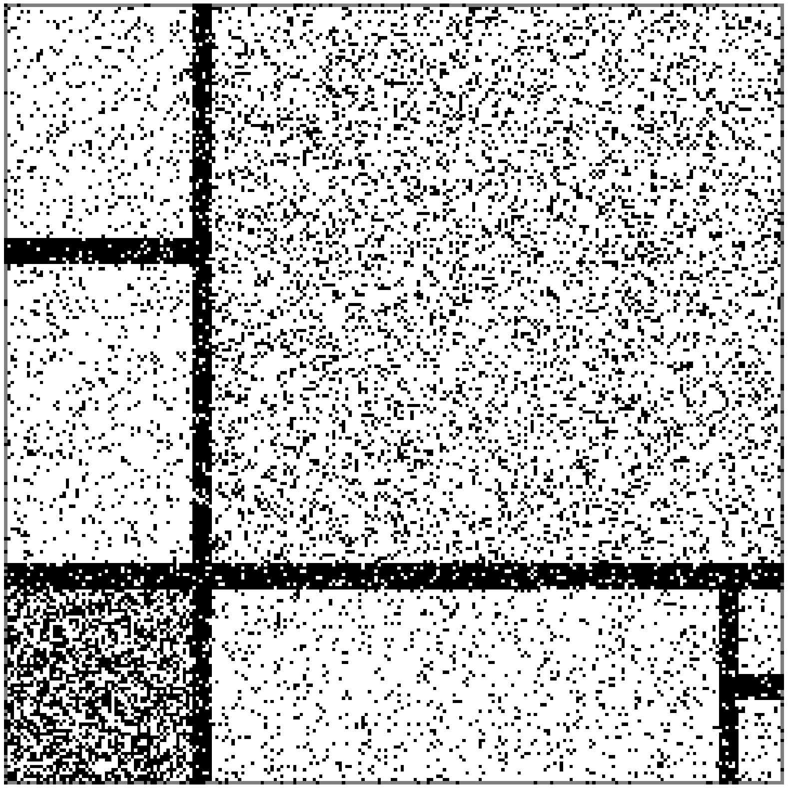
$$\sum_{y=1}^{M} L_y = M + \sum_{y=1}^{M} L_y - 1 \leq M + M$$

- we do $\Theta(M)$ test in total
- we do $\Theta(1)$ tests per each y

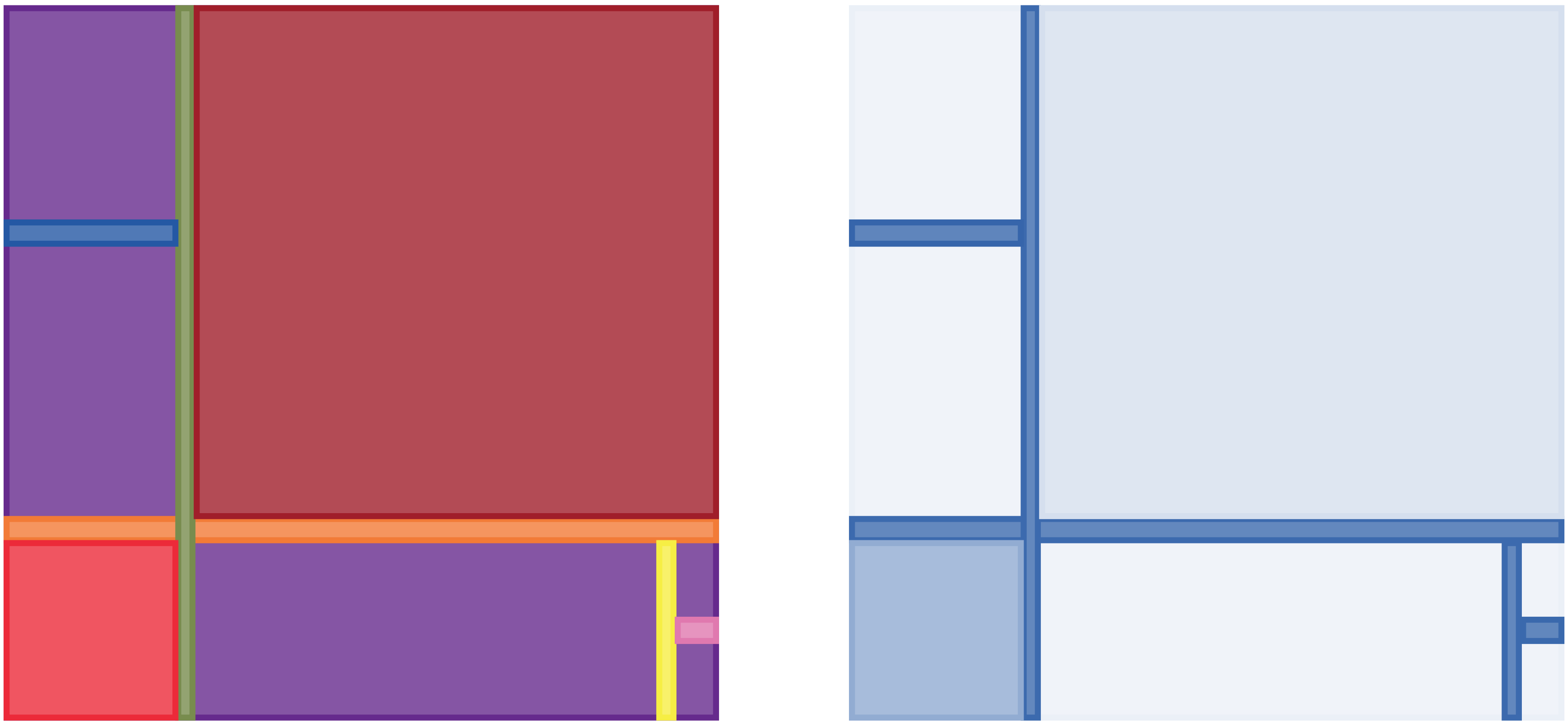
Experiments

Composition II





Composition II



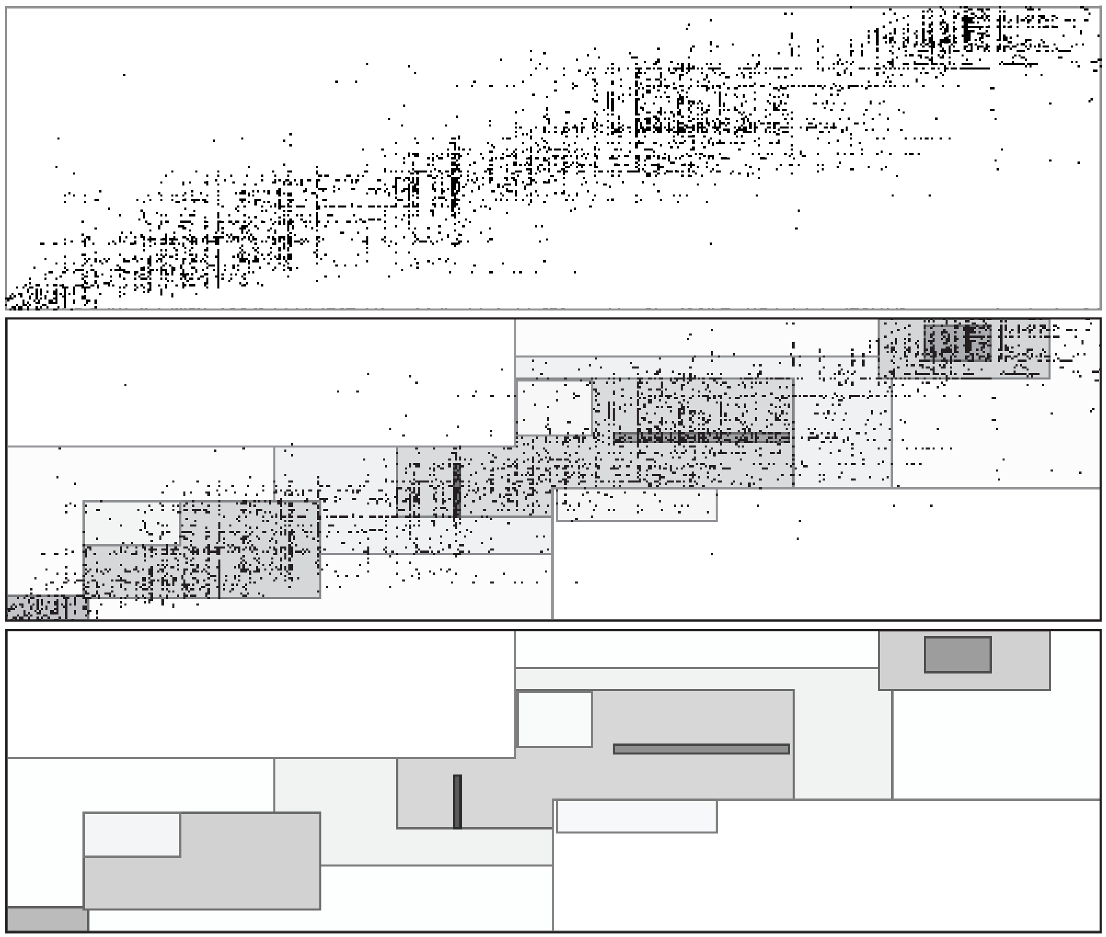
Experiments

				Overlap		
Dataset	M	N	%1s	L%	$\lvert\mathcal{T}\rvert$	*time*
Composition	240	240	23.2	81.58	7	1m23s
Abstracts	859	541	6.6	89.54	14	27m54s
DNA Amp.	4 590	391	1.5	61.61	446	625m
Mammals	2 183	121	20.5	54.62	50	3m06s
Paleo	501	139	5.1	79.07	13	1m22s

Experiments

				Disjoint		
Dataset	M	N	%1s	L%	$\lvert\mathcal{T}\rvert$	*time*
Composition	240	240	23.2	81.72	8	57s
Abstracts	859	541	6.6	89.59	14	16m03s
DNA Amp.	4 590	391	1.5	61.91	466	334m
Mammals	2 183	121	20.5	54.69	55	1m37s
Paleo	501	139	5.1	80.23	14	39s

Paleontological data



Conclusions

- how to discover tile trees from ordered binary data
- score a tree via a statistical model
- apply MDL principle to control model selection
- greedy heuristic for finding the tree
- finding optimal subtile in cubic time, instead of quadrupled time

That's it