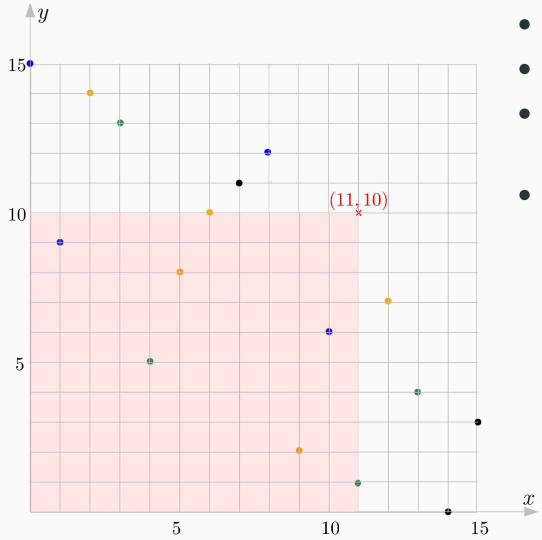
Adaptive Data Structures for Colored 2D Dominance Range Counting

Younan Gao

Dalhousie University

Define the Problems



- Word-RAM: $w = \Theta(\lg n)$ bits,
- on an $n \times n$ grid,
- colors drawn from $[0..C-1]$, where $C \leq n$,
- and colored dominance range counting: $k = 3$.

Related Work and Our Result

Adaptive 2D orthogonal range counting		
Space	Query Time	Remark
$O(n \lg\lg n)$	$O(\lg\lg n + \log_w k)$	TALG'2016

- The α-capped version of the problem
- Nested shallow cuttings

Related Work and Our Result

Adaptive 2D orthogonal range counting		
Space	Query Time	Remark
$O(n \lg \lg n)$	$O(\lg \lg n + \log_w k)$	TALG'2016
Adaptive colored 1D range counting		
$O(n)$	$O(1 + \log_w k)$	TODS'2014

Related Work and Our Result

Adaptive 2D orthogonal range counting		
Space	Query Time	Remark
$O(n \lg \lg n)$	$O(\lg \lg n + \log_w k)$	TALG'2016
Adaptive colored 1D range counting		
$O(n)$	$O(1 + \log_w k)$	TODS'2014
Colored 2D dominance range counting		
$O(n)$	$O(\log_w n)$	Known

- colored 2D dominance range counting $\rightarrow$ 2D stabbing counting;
- 2D stabbing counting $\rightarrow$ 2D dominance range counting.
  - $k = C - \overline{k}$.

Related Work and Our Result

Adaptive 2D orthogonal range counting		
Space	Query Time	Remark
$O(n \lg\lg n)$	$O(\lg\lg n + \log_w k)$	TALG'2016
Adaptive colored 1D range counting		
$O(n)$	$O(1 + \log_w k)$	TODS'2014
Colored 2D dominance range counting		
$O(n)$	$O(\log_w n)$	Known
Adaptive colored 2D dominance range counting		
$O(n)$	$O(1 + \log_w k)$	New

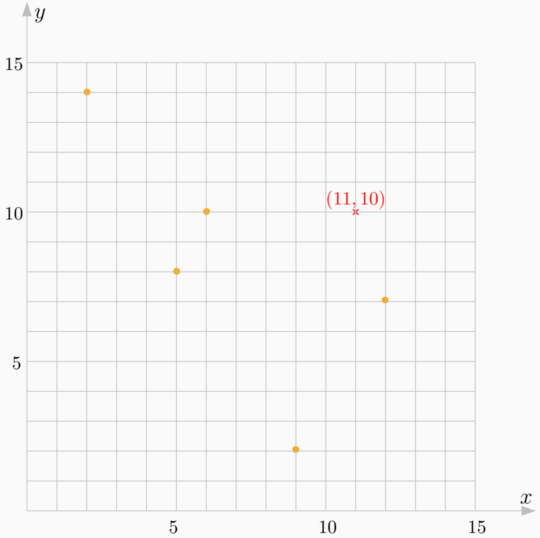
- Colored 2D dominance range counting $\rightarrow$ 2D stabbing counting
- Adaptive 2D 3-sided stabbing counting
  - The α-capped version of 2D 3-sided stabbing counting
  - Nested shallow cuttings

Related Work and Our Result

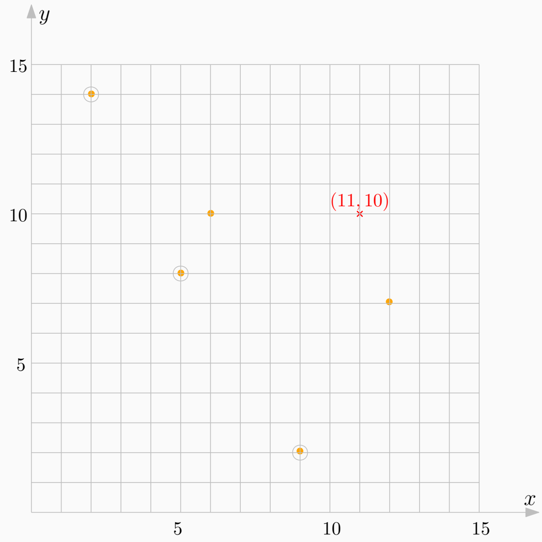
Adaptive 2D orthogonal range counting		
Space	Query Time	Remark
$O(n \lg\lg n)$	$O(\lg\lg n + \log_w k)$	TALG'2016
Adaptive colored 1D range counting		
$O(n)$	$O(1 + \log_w k)$	TODS'2014
Colored 2D dominance range counting		
$O(n)$	$O(\log_w n)$	Known
Adaptive colored 2D dominance range counting		
$O(n)$	$O(1 + \log_w k)$	New

- Colored 2D dominance range counting $\rightarrow$ 2D stabbing counting ✓
- Adaptive 2D 3-sided stabbing counting ✓
  - The α-capped version of 2D 3-sided stabbing counting
  - Nested shallow cuttings

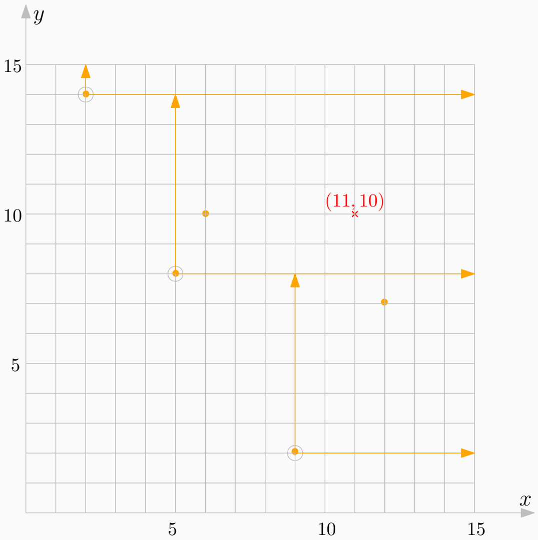
Reducing to 2D 3-Sided Stabbing Counting



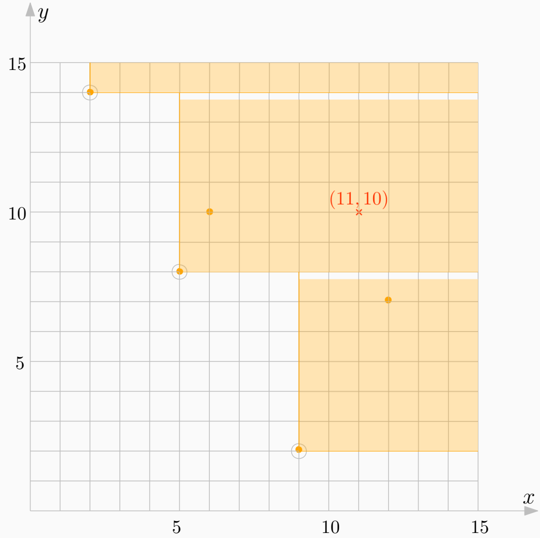
Reducing to 2D 3-Sided Stabbing Counting



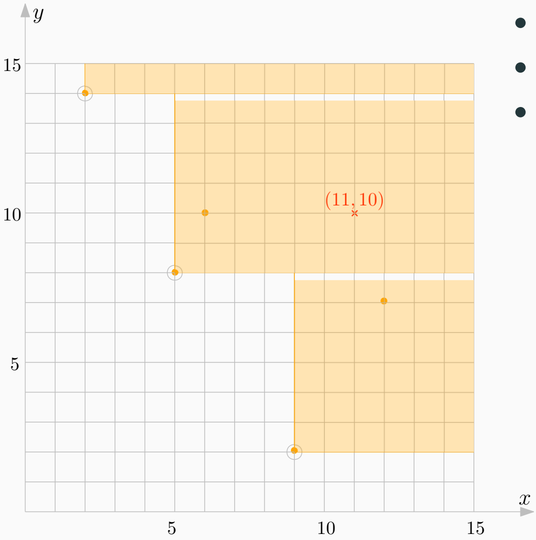
Reducing to 2D 3-Sided Stabbing Counting



Reducing to 2D 3-Sided Stabbing Counting

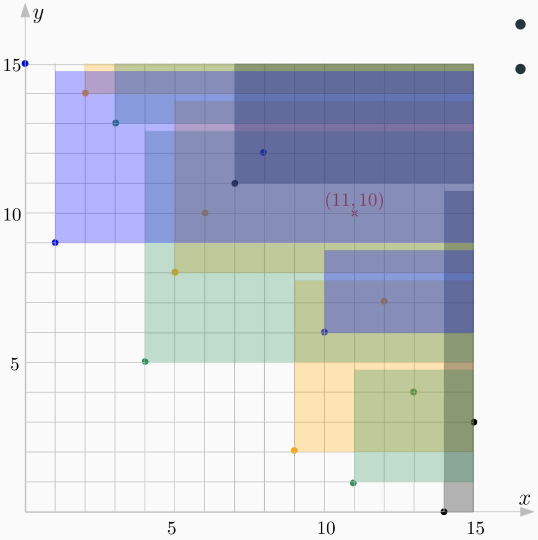


Reducing to 2D 3-Sided Stabbing Counting



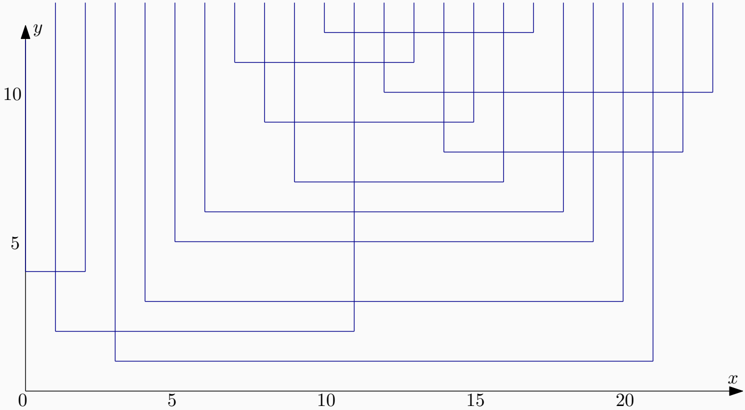
- All rectangles are disjoint.
- Each rectangle has ≤ 3 sides.
- An orange point is dominated by the query point iff an orange rectangle contains the query point.

Reducing to 2D 3-Sided Stabbing Counting



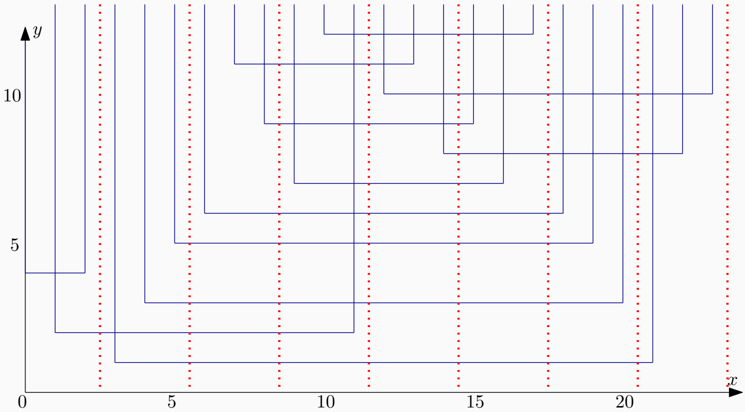
- At most $\sum_c |P_c| = n$ rectangles
- # of the rectangles that contain the query point = # of the distinct colors dominated by the query point.

A t-level shallow cutting.



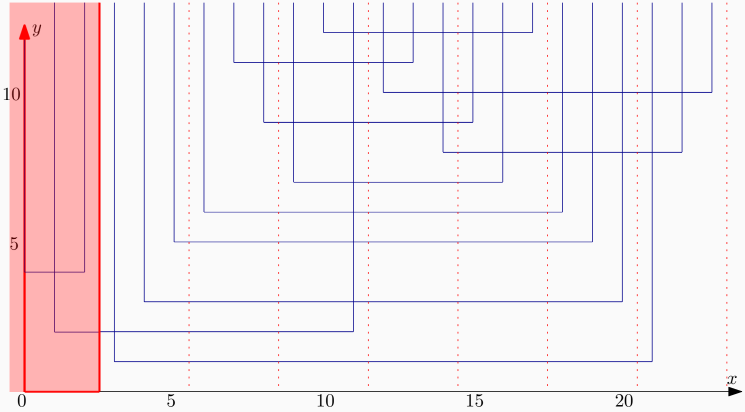
- Assume, w.l.o.g, each rect is of the form $[x_1, x_2] \times [y_1, +\infty)$.

A t-level shallow cutting.



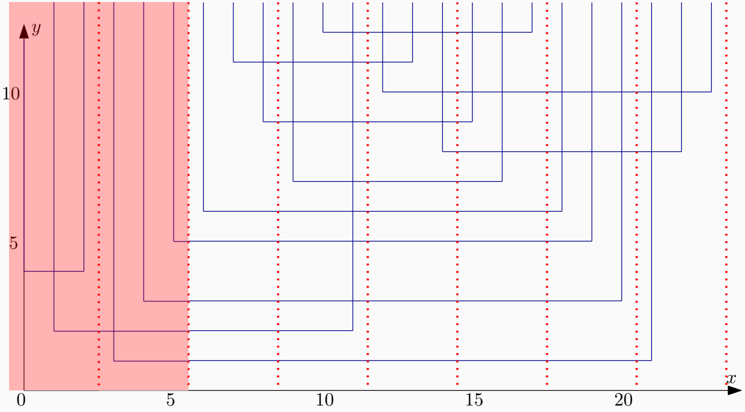
- Divide the vertical edges into slabs of size t.

A t-level shallow cutting.

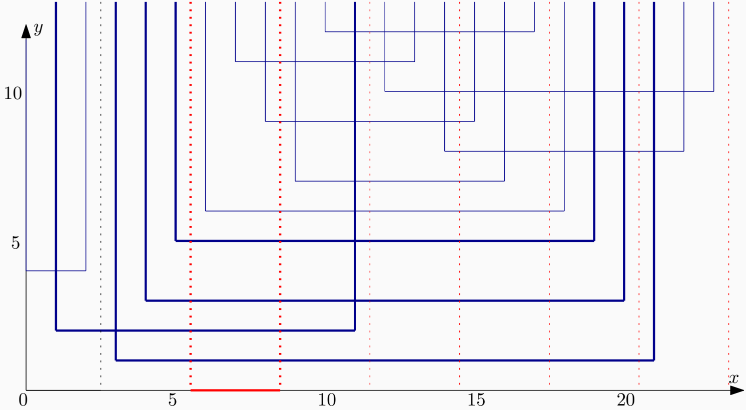


- Each cell is of the form $[x_1, x_2] \times (-\infty, y_2]$.

A t-level shallow cutting.

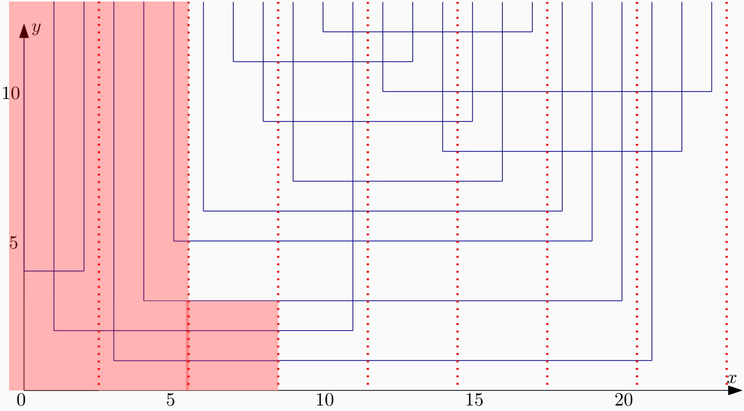


A t-level shallow cutting.

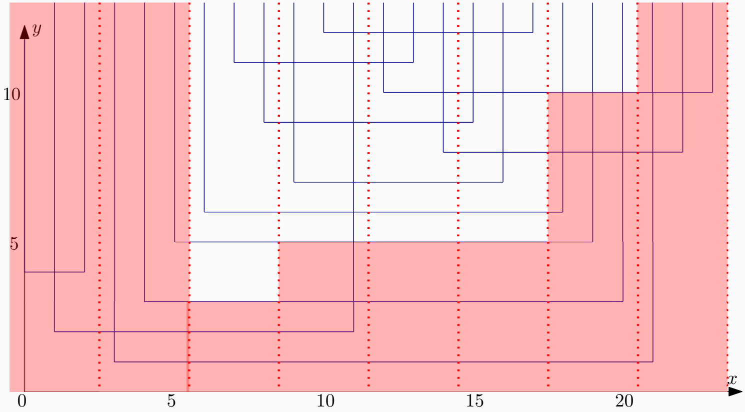


- 4 **rectangles** span the third slab.

A t-level shallow cutting.

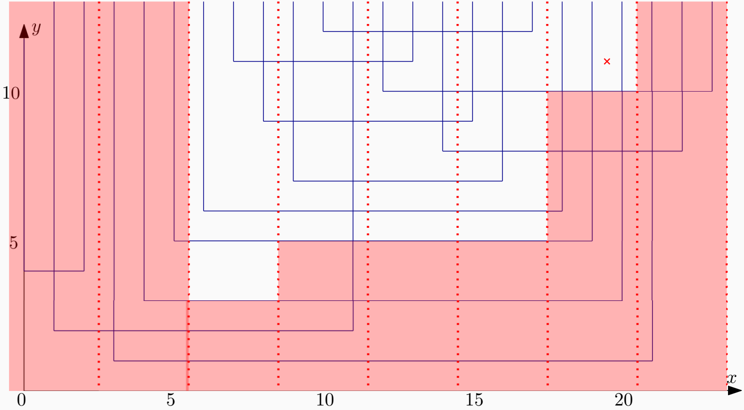


A t-level shallow cutting.



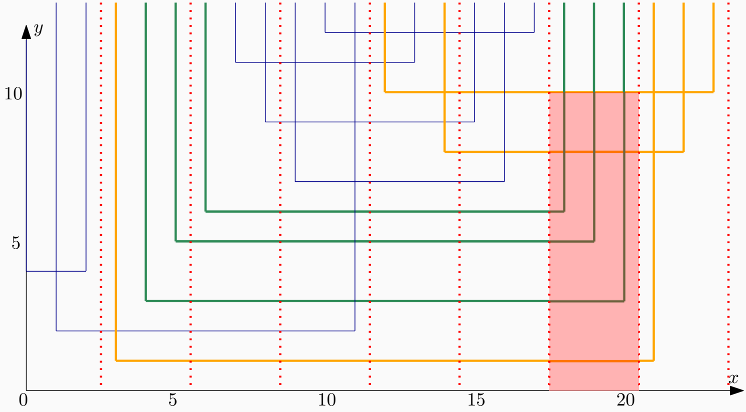
- Overall, $2n/t$ cells are created.

A t-level shallow cutting.



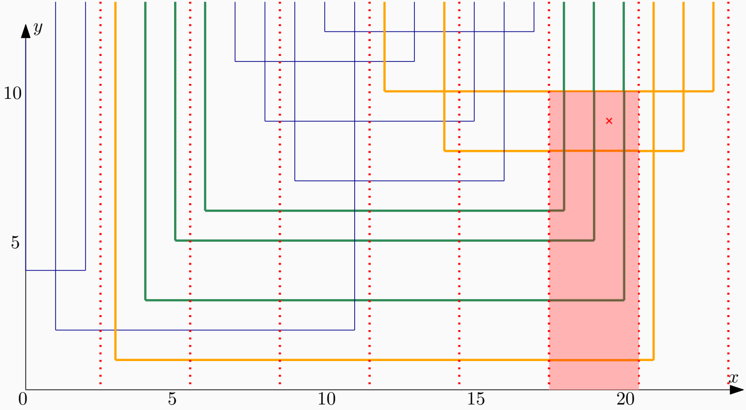
- If q is not contained in any cells, then $\geq t$ rects contain q.
- α-capped version of stabbing counting

A t-level shallow cutting.



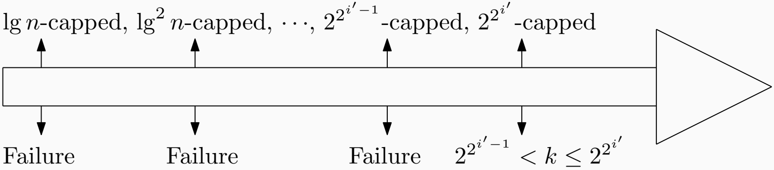
- Each cell intersects with $\leq 2t$ rectangles:
  - $\leq t$ rectangles of type-1
  - $\leq t$ rectangles of type-2.

A t-level shallow cutting.



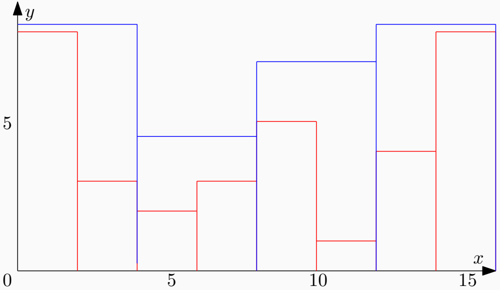
- In α-capped version, the query time is bounded by $O(\log_w \alpha)$, instead of $O(\log_w n)$.

An $O(n \lg \lg n)$-word solution



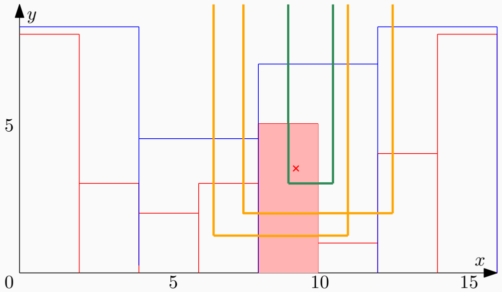
- For each $\lg \lg \lg n \leq i \leq \lg \lg n$,
- construct 2^{2^i}-capped data structure.
- Return k in $O(\lg \lg k + \log_w k)$ time.

Nested Shallow Cuttings: An Observation



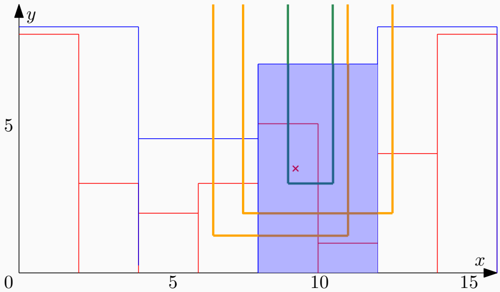
- **Observation:** All rects that intersect a smaller cell of t-level cutting intersect the parent cell in t^2-level cutting.

Nested Shallow Cuttings: An Observation



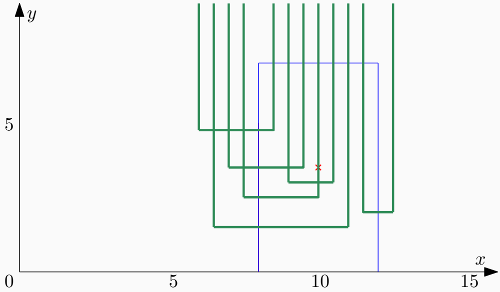
- **Observation:** All rects that intersect a smaller cell of t-level cutting intersect the parent cell in t^2-level cutting.
- Before, we looked for the smallest cell that contains the query point

Nested Shallow Cuttings: An Observation



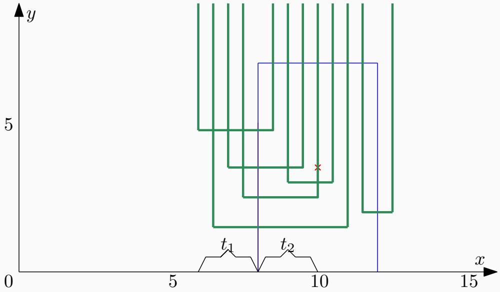
- **Observation:** All rects that intersect a smaller cell of t-level cutting intersect the parent cell in t^2-level cutting.
- Before, we looked for the smallest cell that contains the query point

Handling Type-1 Rectangles



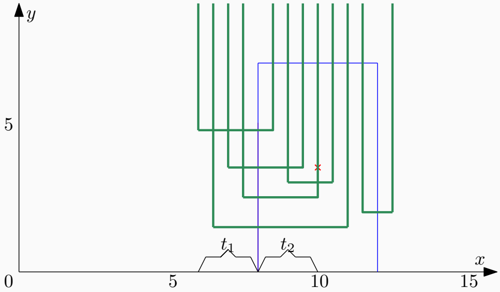
- Saving space from $O(\alpha \lg n)$ to $O(\alpha \lg \alpha)$ bits by rank reduction;

Handling Type-1 Rectangles



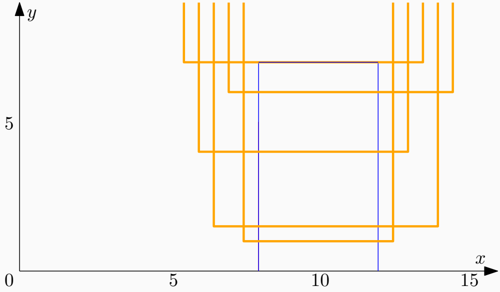
- Saving space from $O(\alpha \lg n)$ to $O(\alpha \lg \alpha)$ bits by rank reduction;
- x-rank of q: $t_1 + t_2$, where
  - t_1 is pre-stored, using $O(\lg \alpha)$ bits and
  - t_2 is $q.x \mod \alpha$

Handling Type-1 Rectangles



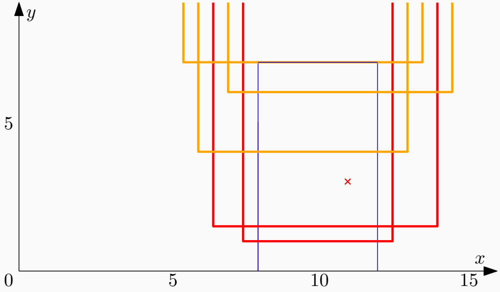
- Saving space from $O(\alpha \lg n)$ to $O(\alpha \lg \alpha)$ bits by rank reduction;
- x-rank of q: $t_1 + t_2$, where
  - t_1 is pre-stored, using $O(\lg \alpha)$ bits and
  - t_2 is $q.x \mod \alpha$
- y-rank of q: Use $O(\alpha(\lg n / \log_w \alpha + \lg \alpha))$ bits, plus additional $O(n)$ words, and return in $O(\log_w \alpha)$ time.

Handling Type-2 Rectangles



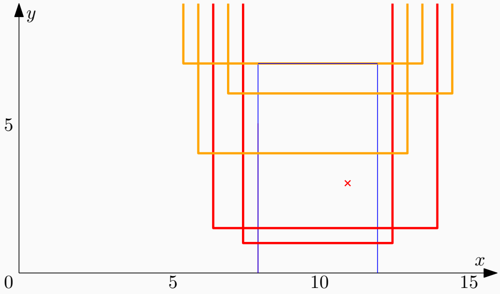
- Recall that a cell intersects $\leq \alpha$ type-2 rects.

Handling Type-2 Rectangles



- Recall that a cell intersects $\leq \alpha$ type-2 rects.

Handling Type-2 Rectangles



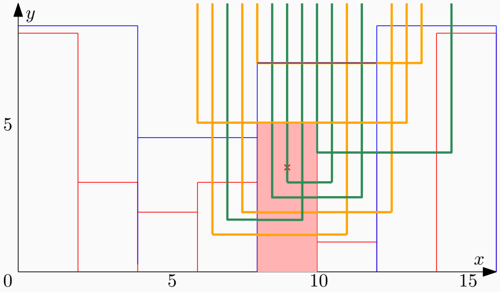
- Recall that a cell intersects $\leq \alpha$ type-2 rects.
- Now, build the data structure for $\sqrt{\alpha}$ lowest ones:
  - A predecessor structure implemented by Fusion Trees
  - using $O(\sqrt{\alpha} \lg n)$ bits of space.

Wrap-Up: Space Costs

- Total space cost in bits:

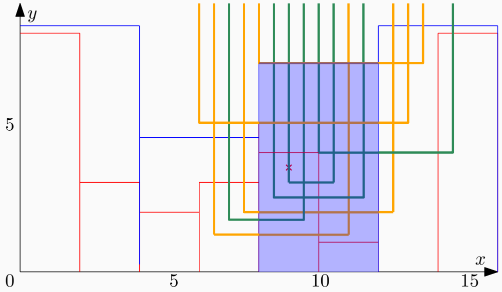
$$O(n \lg n) + \sum_{\alpha} O(\frac{n}{\alpha} \cdot (\sqrt{\alpha} \lg n + \alpha(\frac{\lg n}{\log_w \alpha} + \lg \alpha))) = O(n \lg n)$$

Wrap-Up: The Query Algorithm



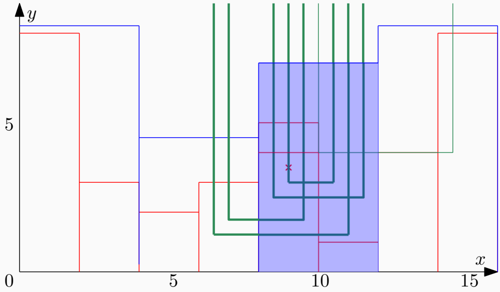
- Finding the smallest cell that contains q in constant time, e.g., $2^{2^{i'-1}} \leq k \leq 2^{2^{i'}}$.

Wrap-Up: The Query Algorithm



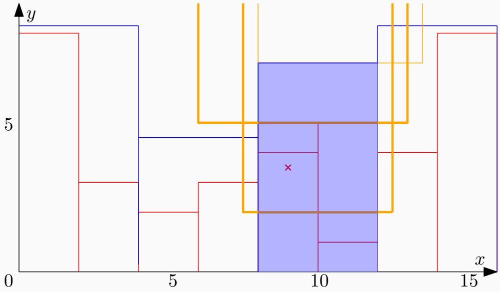
- Finding the smallest cell that contains q in constant time, e.g., $2^{2^{i'-1}} \leq k \leq 2^{2^{i'}}$.
- Finding the parent of the smallest cell.

Wrap-Up: The Query Algorithm



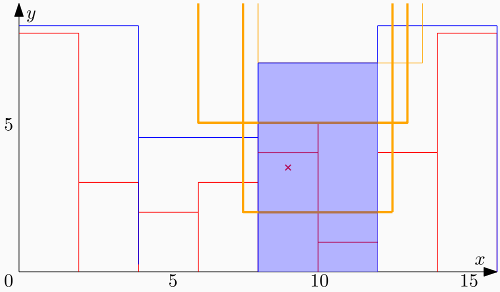
- Finding the smallest cell that contains q in constant time, e.g., $2^{2^{i'-1}} \leq k \leq 2^{2^{i'}}$.
- Finding the parent of the smallest cell.
- Search for k_1 among type-1 rects in $O(\log_w 2^{2^{i'+1}}) = O(\log_w k)$ time.

Wrap-Up: The Query Algorithm



- Finding the smallest cell that contains q in constant time, e.g., $2^{2^{i'-1}} \leq k \leq 2^{2^{i'}}$.
- Finding the parent of the smallest cell.
- Search for k_1 among type-1 rects in $O(\log_w 2^{2^{i'+1}}) = O(\log_w k)$ time.
- Search for k_2 among type-2 rects in $O(\log_w \sqrt{2^{2^{i'+1}}}) = O(\log_w k)$.

Wrap-Up: The Query Algorithm



- Finding the smallest cell that contains q in constant time, e.g., $2^{2^{i'-1}} \leq k \leq 2^{2^{i'}}$.
- Finding the parent of the smallest cell.
- Search for k_1 among type-1 rects in $O(\log_w 2^{2^{i'+1}}) = O(\log_w k)$ time.
- Search for k_2 among type-2 rects in $O(\log_w \sqrt{2^{2^{i'+1}}}) = O(\log_w k)$.
- return $k_1 + k_2$ as k.

Open Problems

Colored 3D dominance range counting:

- $O(n \lg n / \lg \lg n)$ words of space and $O((1 + \log_w k)^2)$ query time?

Thanks!