Orthogonal Range Searching

Sándor Kisfaludi-Bak

Computaional Geometry
Summer semester 2020

max planck institut
informatik

Overview

- Intro and problem definition

Overview

- Intro and problem definition

- Kd trees

Overview

- Intro and problem definition

- Kd trees

- Range trees

Overview

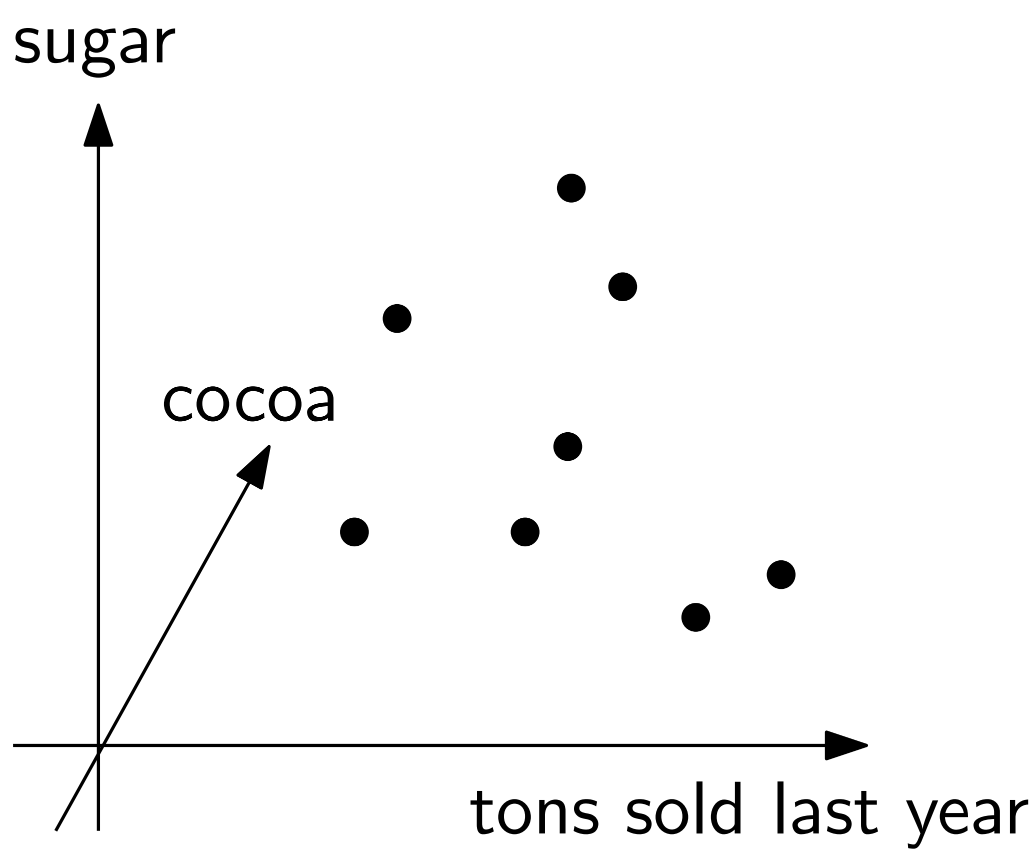
- Intro and problem definition
- Kd trees
- Range trees
- Fractional cascading

Overview

- Intro and problem definition
- Kd trees
- Range trees
- Fractional cascading
- Priority search trees

The obvious applciation

Database of cakes.

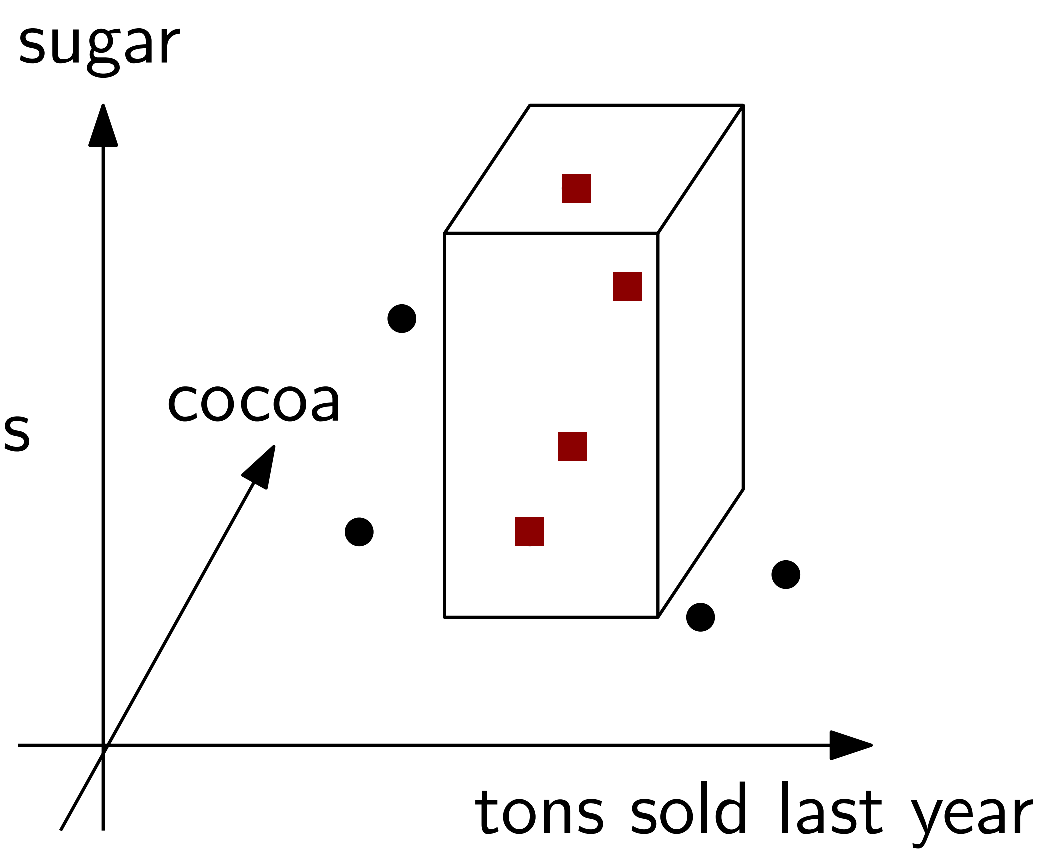


The obvious applciation

Database of cakes.

Which cakes have

- sugar content [0.12,0.17]
- cocoa content [0.05,0.1]
- and sold btw. 3 and 4 tons last year?



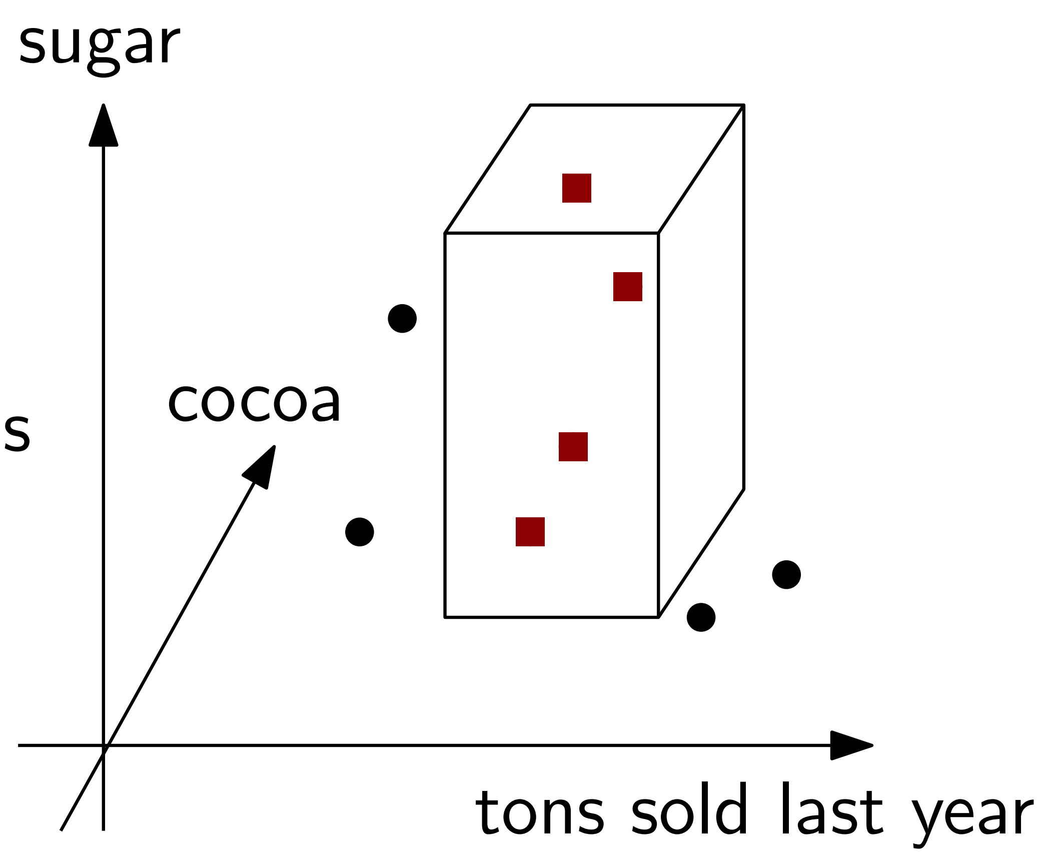
The obvious applciation

Database of cakes.

Which cakes have

- sugar content [0.12,0.17]
- cocoa content [0.05,0.1]
- and sold btw. 3 and 4 tons last year?

Orthogonal range query

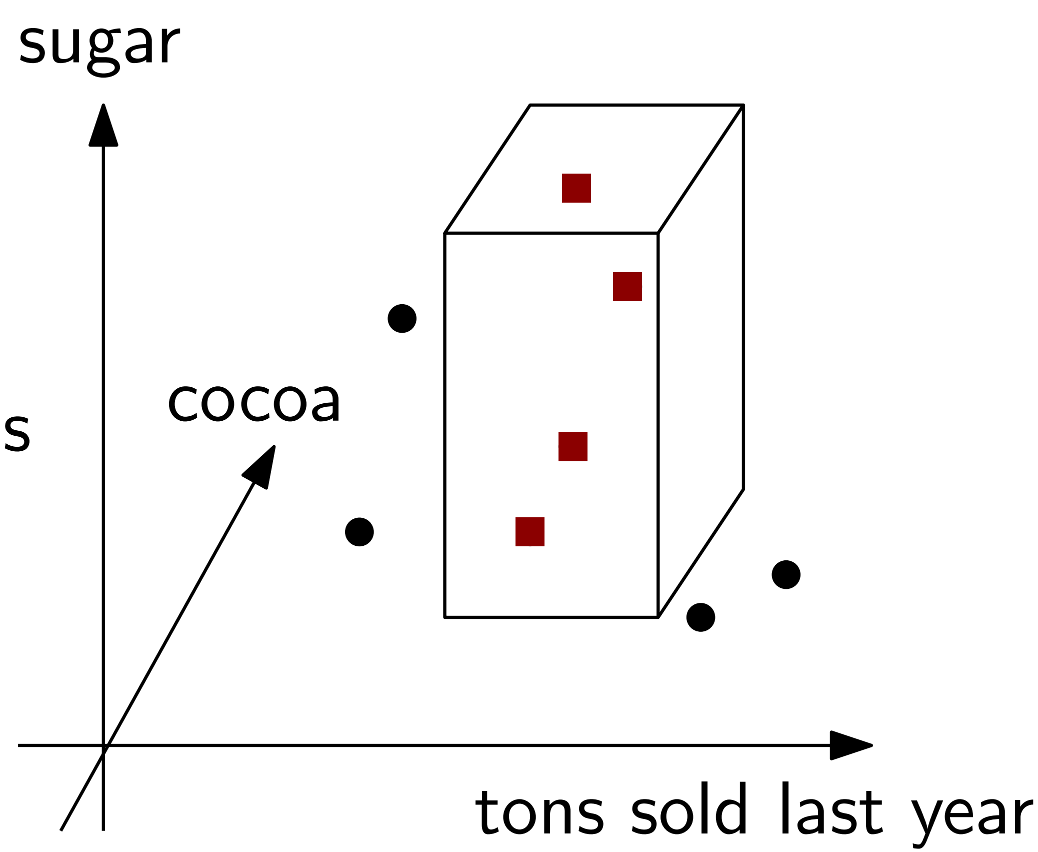


The obvious applciation

Database of cakes.

Which cakes have

- sugar content [0.12,0.17]
- cocoa content [0.05,0.1]
- and sold btw. 3 and 4 tons last year?



Orthogonal range query

Task: support such queries efficiently

Problem definition

Given n points in $\mathbb{Z}^d$ or $\mathbb{Q}^d$,

1. Preprocess them in $\widetilde{O}(n)$ time and space to
2. support orthogonal range queries in $\mathrm{poly}(\log n) + O(k)$
3. on a Word RAM.

Problem definition

Given n points in $\mathbb{Z}^d$ or $\mathbb{Q}^d$,

1. Preprocess them in $\widetilde{O}(n)$ time and space to
2. support orthogonal range queries in $\mathrm{poly}(\log n) + O(k)$
3. on a Word RAM.

output size

Problem definition

Given n points in $\mathbb{Z}^d$ or $\mathbb{Q}^d$,

1. Preprocess them in $\widetilde{O}(n)$ time and space to
2. support orthogonal range queries in $\mathrm{poly}(\log n) + \underbrace{O(k)}_{\text{output size}}$
3. on a Word RAM.

Static: preprocess and answer queries

$\updownarrow$

Dynamic: update insertions and deletions in $\mathrm{poly}(\log(n))$

The 1-dimensional problem

Query: $[x, x']$

The 1-dimensional problem

Query: $[x, x']$

Option 1. Use sorted array:

Binary search for x

Report next until exceeds x'

The 1-dimensional problem

Query: $[x, x']$

Option 1. Use sorted array:

Binary search for x

Report next until exceeds x'

Static only

The 1-dimensional problem

Query: $[x, x']$

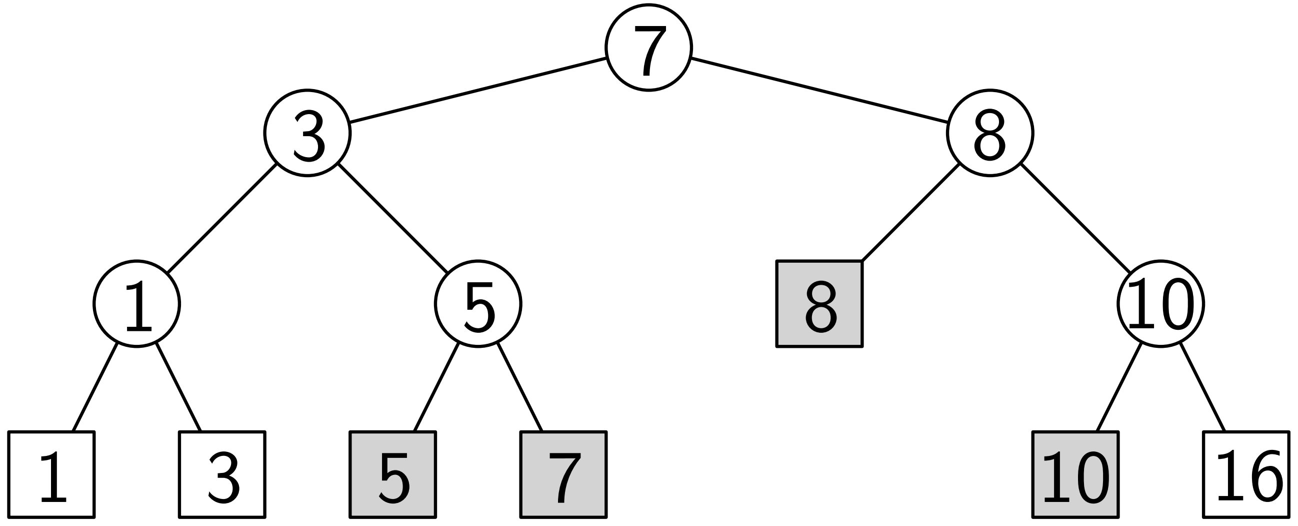
Option 1. Use sorted array:

Binary search for x

Report next until exceeds x'

Static only

Option 2. Binary search tree:



The 1-dimensional problem

Query: $[x, x']$

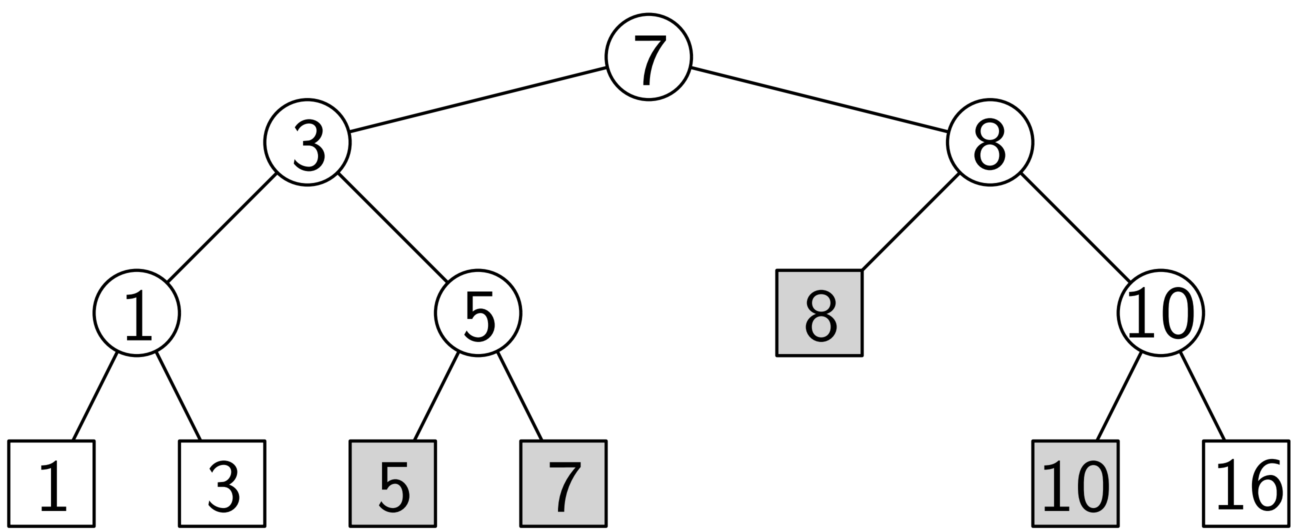
Option 1. Use sorted array:

Binary search for x

Report next until exceeds x'

Static only

Option 2. Binary search tree:



P in the leaves

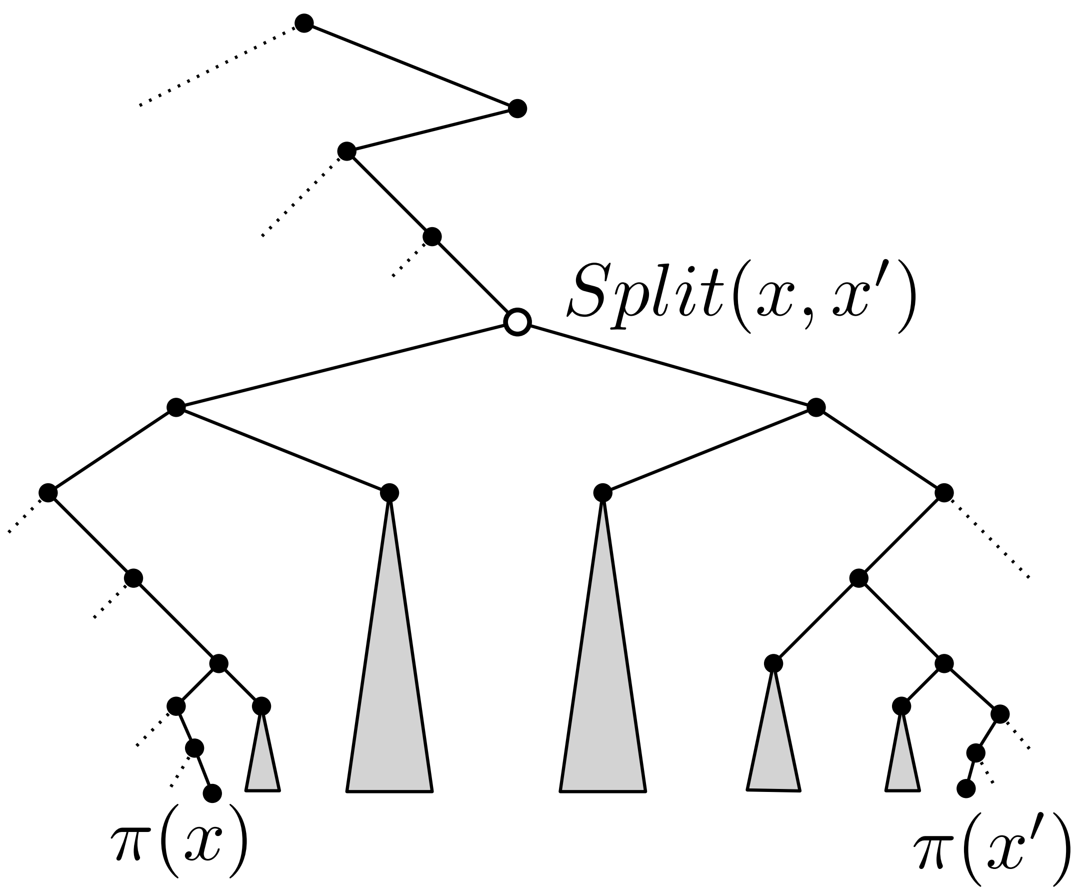
Query: $[4, 11]$

inner vertex $=$ largest value in left child's subtree

Dynamic solution in $\mathbb{R}^1$

Answering a query

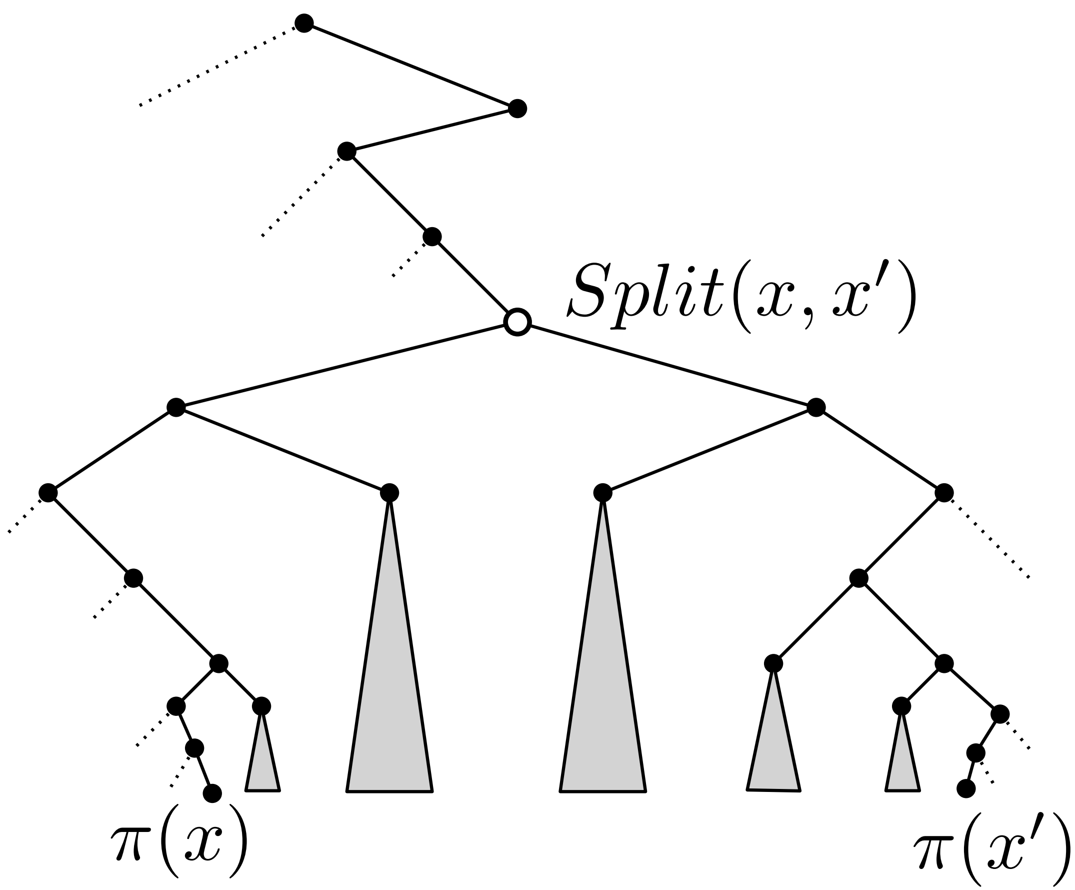
- Binary search for $Split(x, x')$
- Search for x, reporting right child subtrees
- Search for x', reporting left child subtrees



Dynamic solution in $\mathbb{R}^1$

Answering a query

Binary search for $Split(x, x')$
Search for x, reporting right child subtrees
Search for x', reporting left child subtrees



Space $= O(n)$

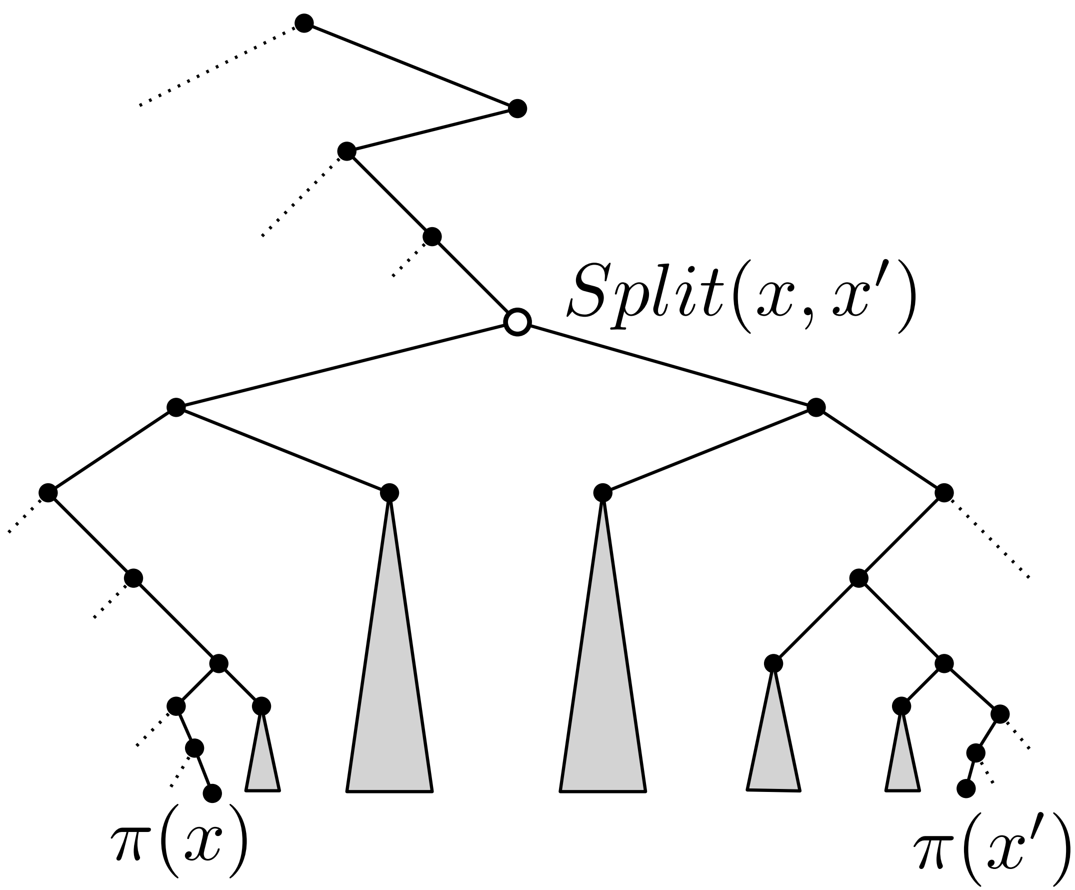
Dynamic solution in $\mathbb{R}^1$

Answering a query

Binary search for $Split(x, x')$

Search for x, reporting right child subtrees

Search for x', reporting left child subtrees



Space $= O(n)$

Preprocess $= O(n \log n)$

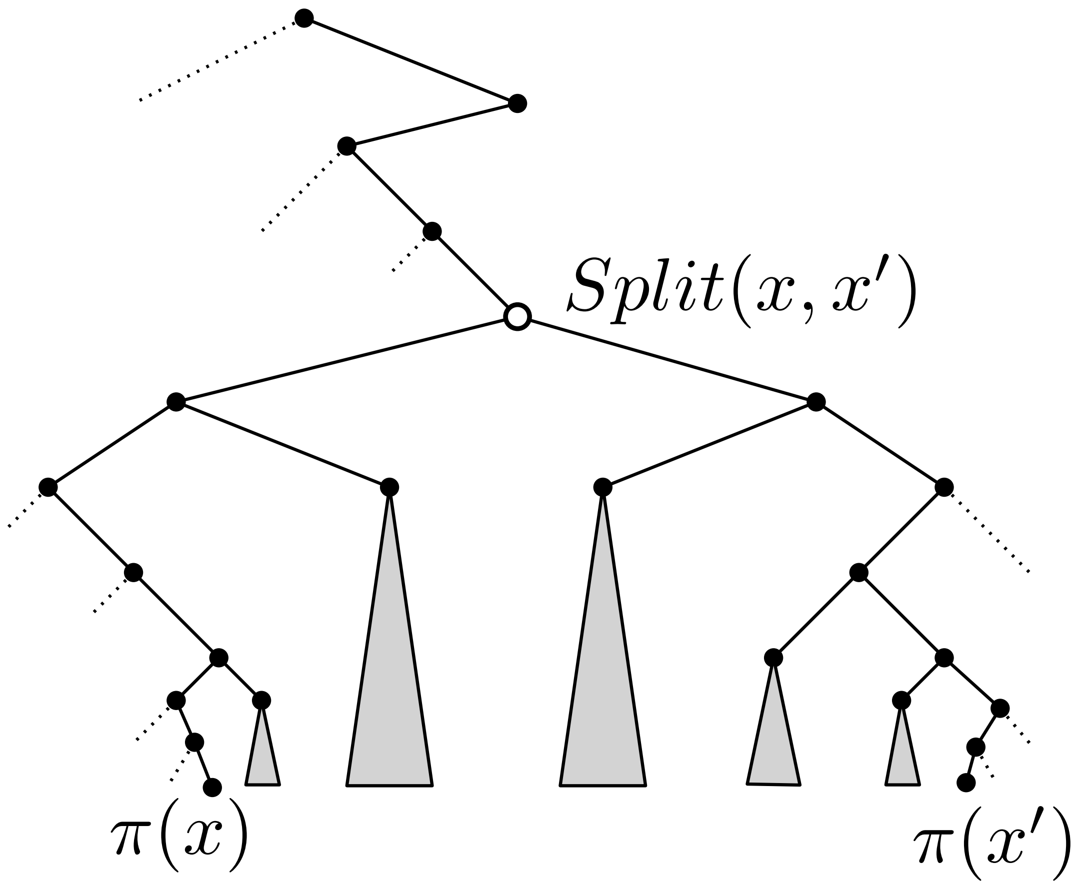
Dynamic solution in $\mathbb{R}^1$

Answering a query

Binary search for $Split(x, x')$

Search for x, reporting right child subtrees

Search for x', reporting left child subtrees



Space $= O(n)$

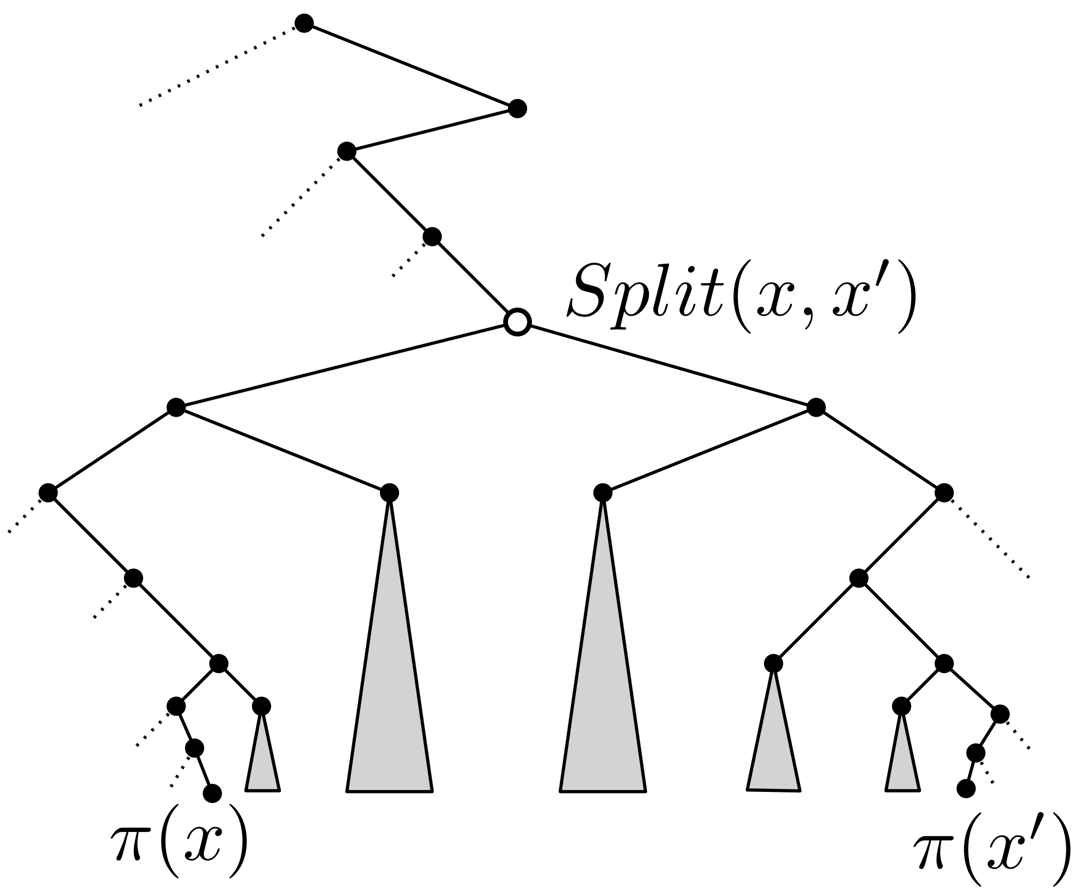
Preprocess $= O(n \log n)$

Update $= O(\log n)$

Dynamic solution in $\mathbb{R}^1$

Answering a query

- Binary search for $Split(x, x')$
- Search for x, reporting right child subtrees
- Search for x', reporting left child subtrees



Space $= O(n)$

Preprocess $= O(n \log n)$

Update $= O(\log n)$

Query $= O(\log n + k)$

Kd trees
Bentley 1975

Kd trees in $\mathbb{R}^2$

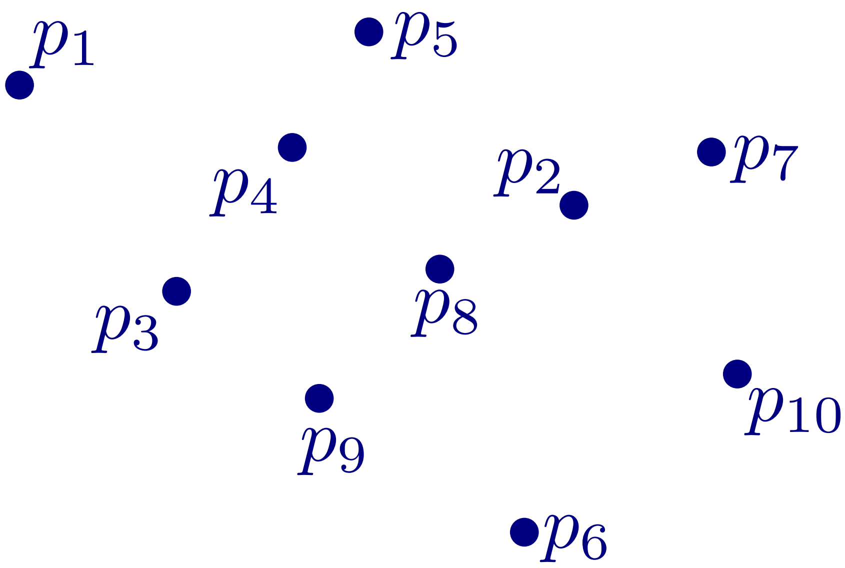
Assume distinct x- and y-coords

Idea:

Kd trees in $\mathbb{R}^2$

Assume distinct x- and y-coords

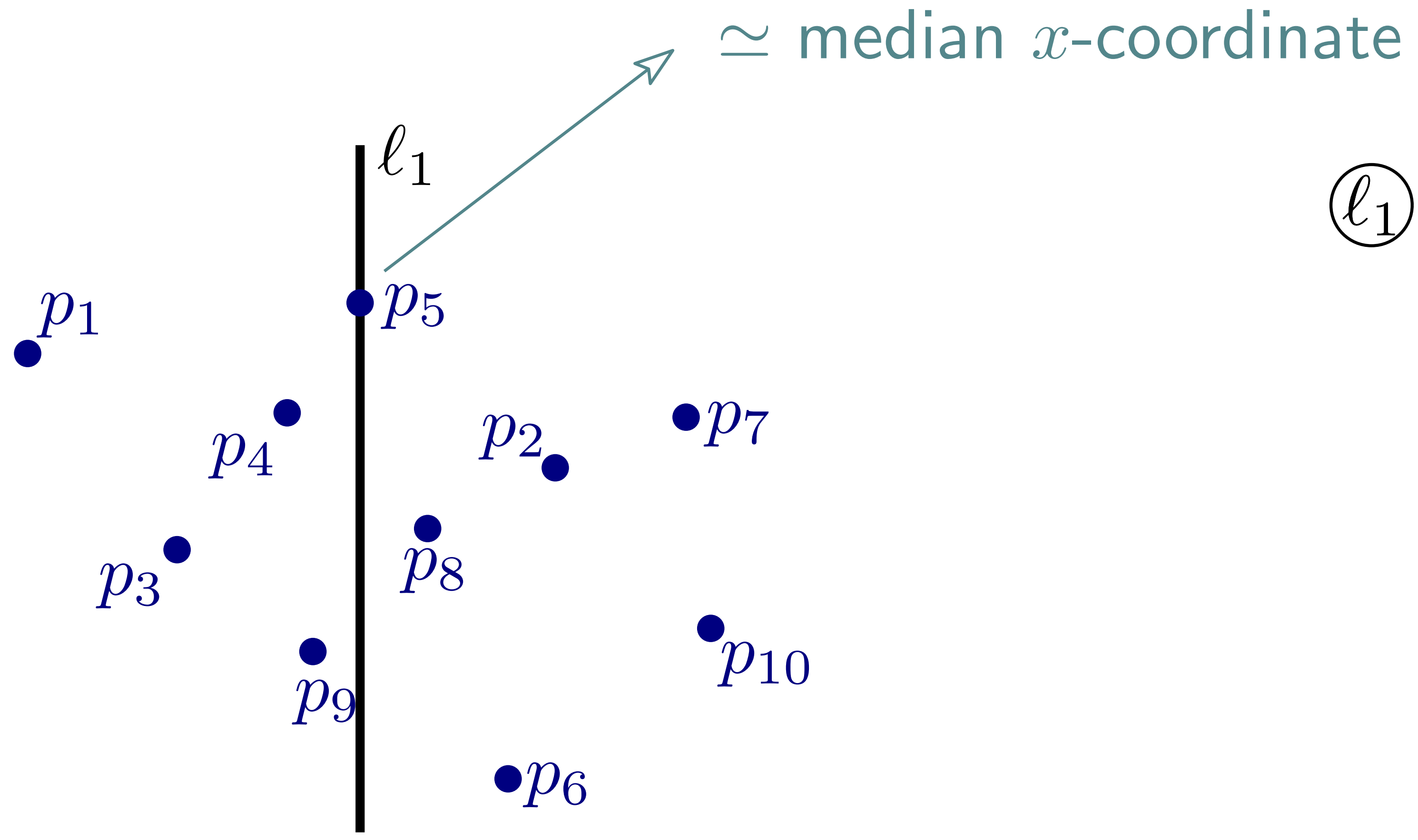
Idea:



Kd trees in $\mathbb{R}^2$

Assume distinct x- and y-coords

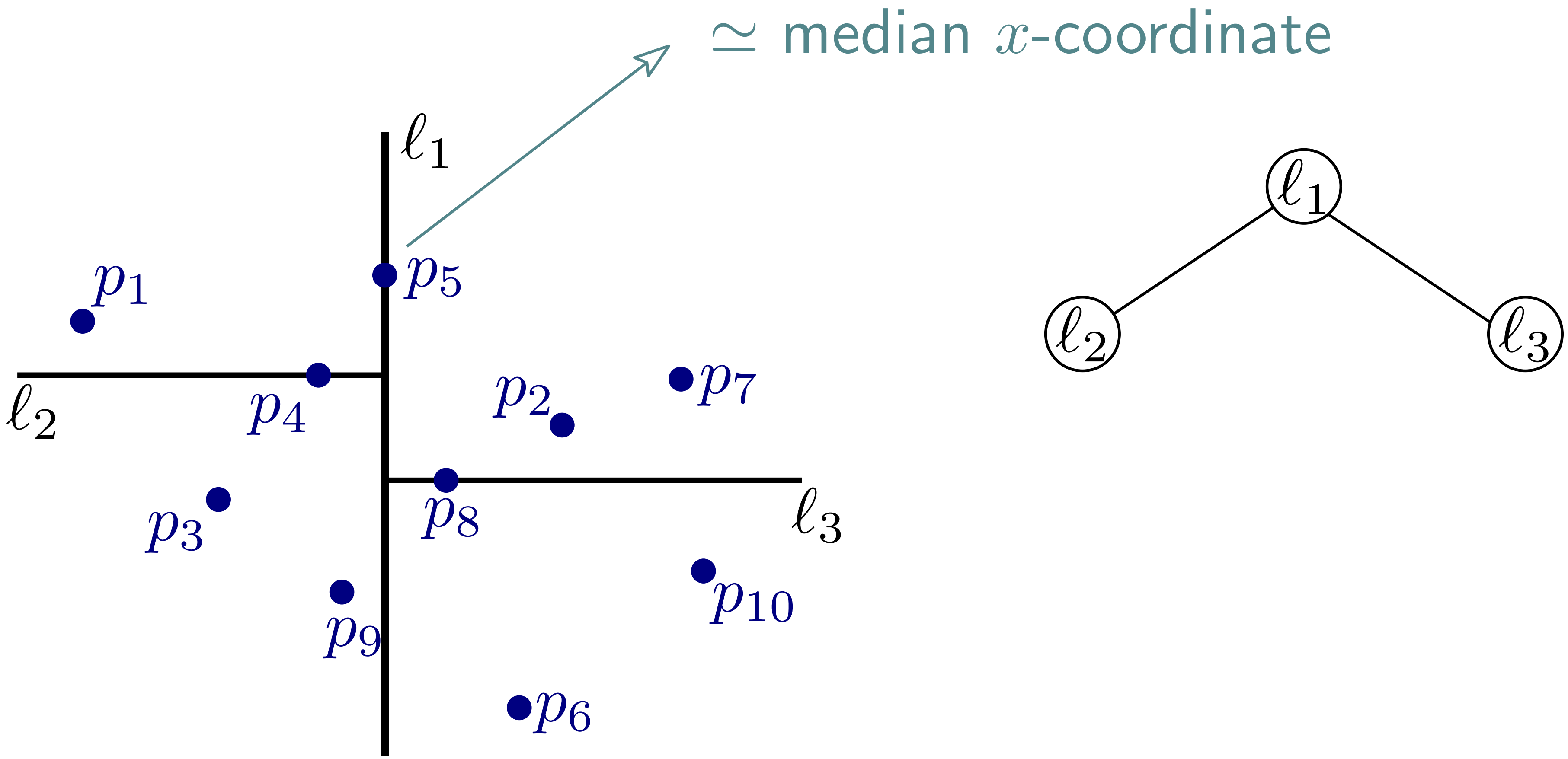
Idea:



Kd trees in $\mathbb{R}^2$

Assume distinct x- and y-coords

Idea:

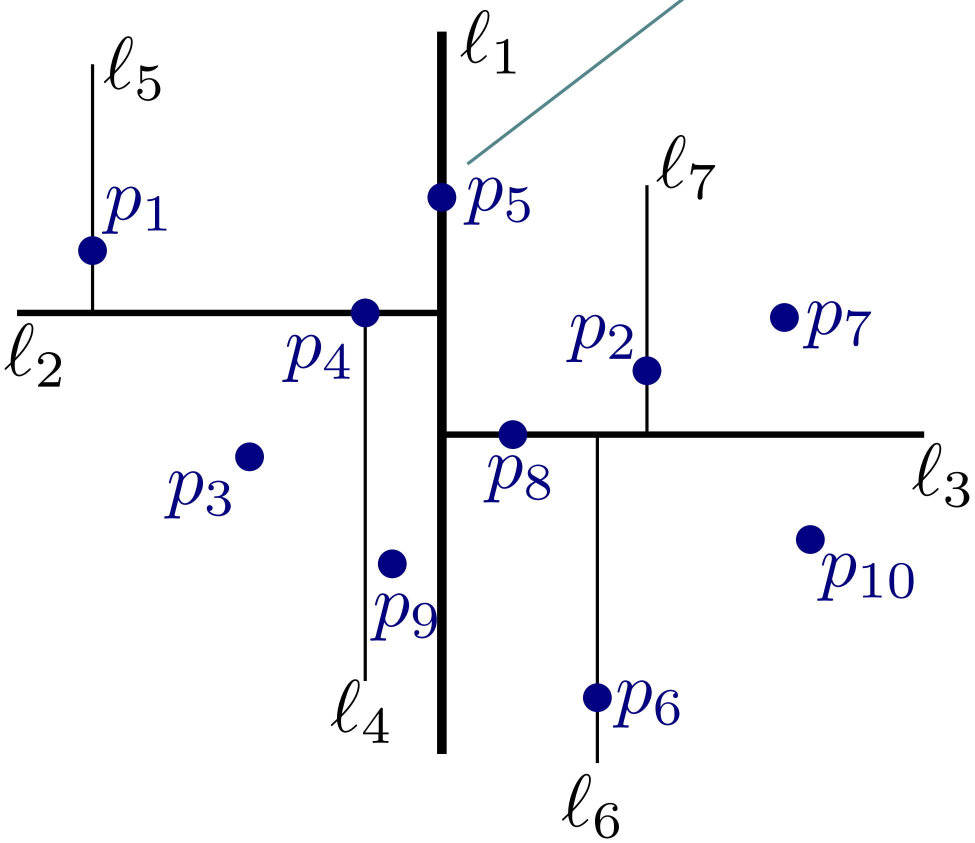


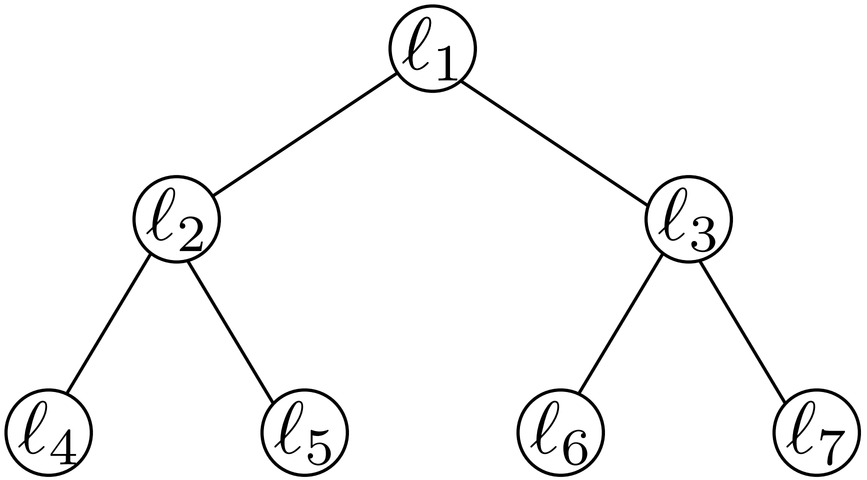
Kd trees in $\mathbb{R}^2$

Assume distinct x- and y-coords

Idea:

$\simeq$ median x-coordinate



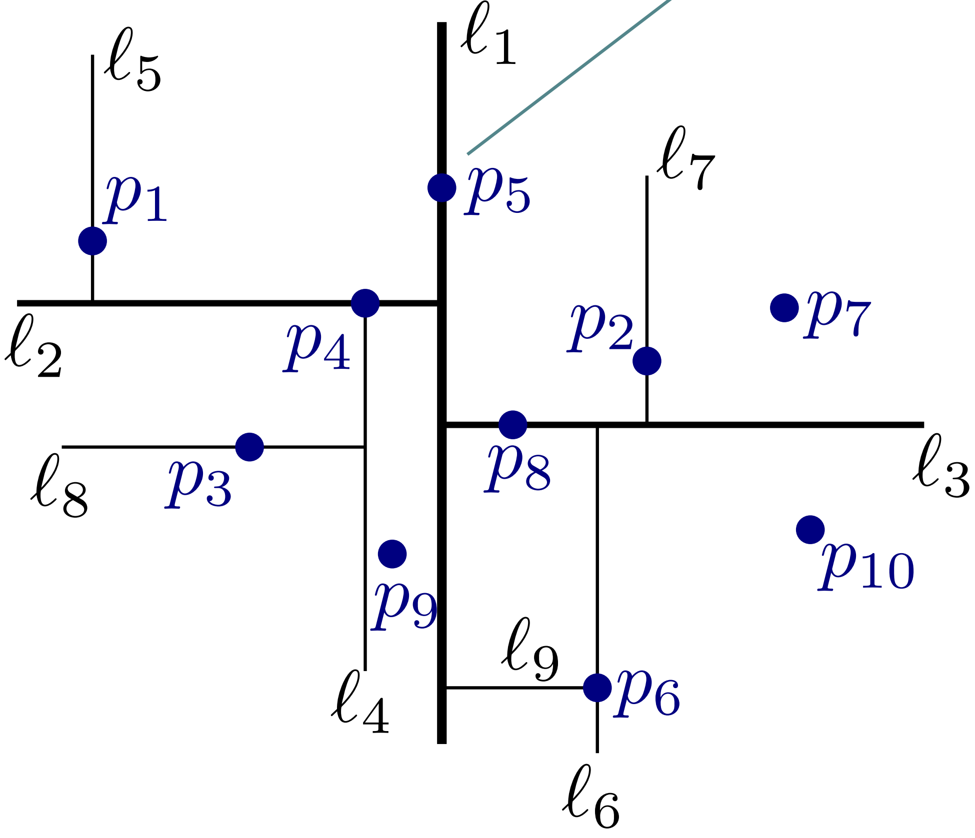


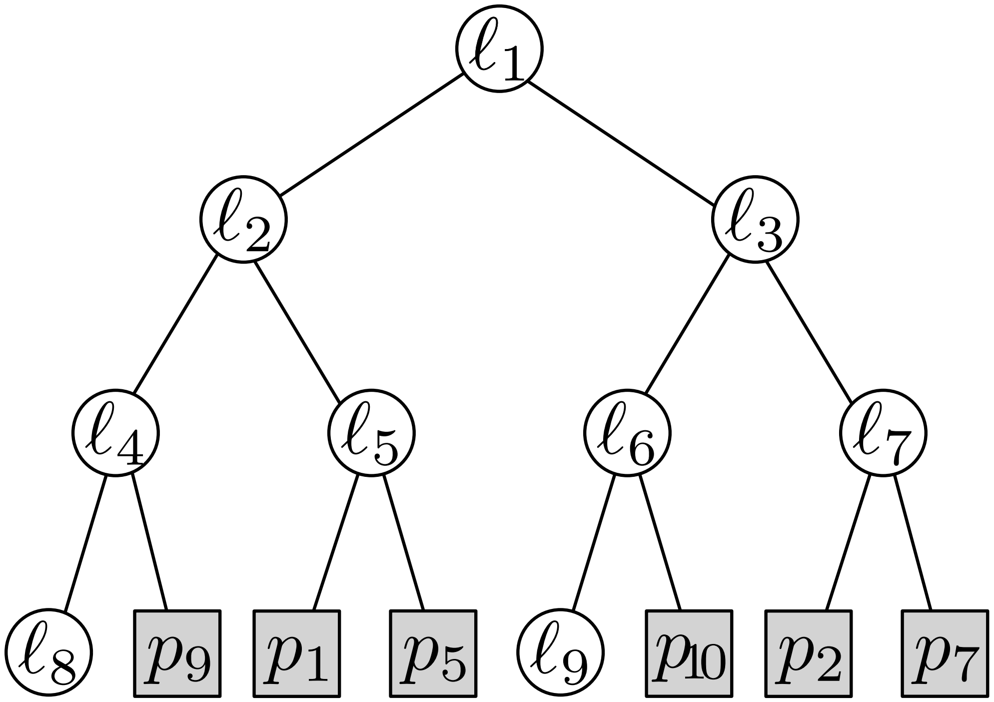
Kd trees in $\mathbb{R}^2$

Assume distinct x- and y-coords

Idea:

$\simeq$ median x-coordinate

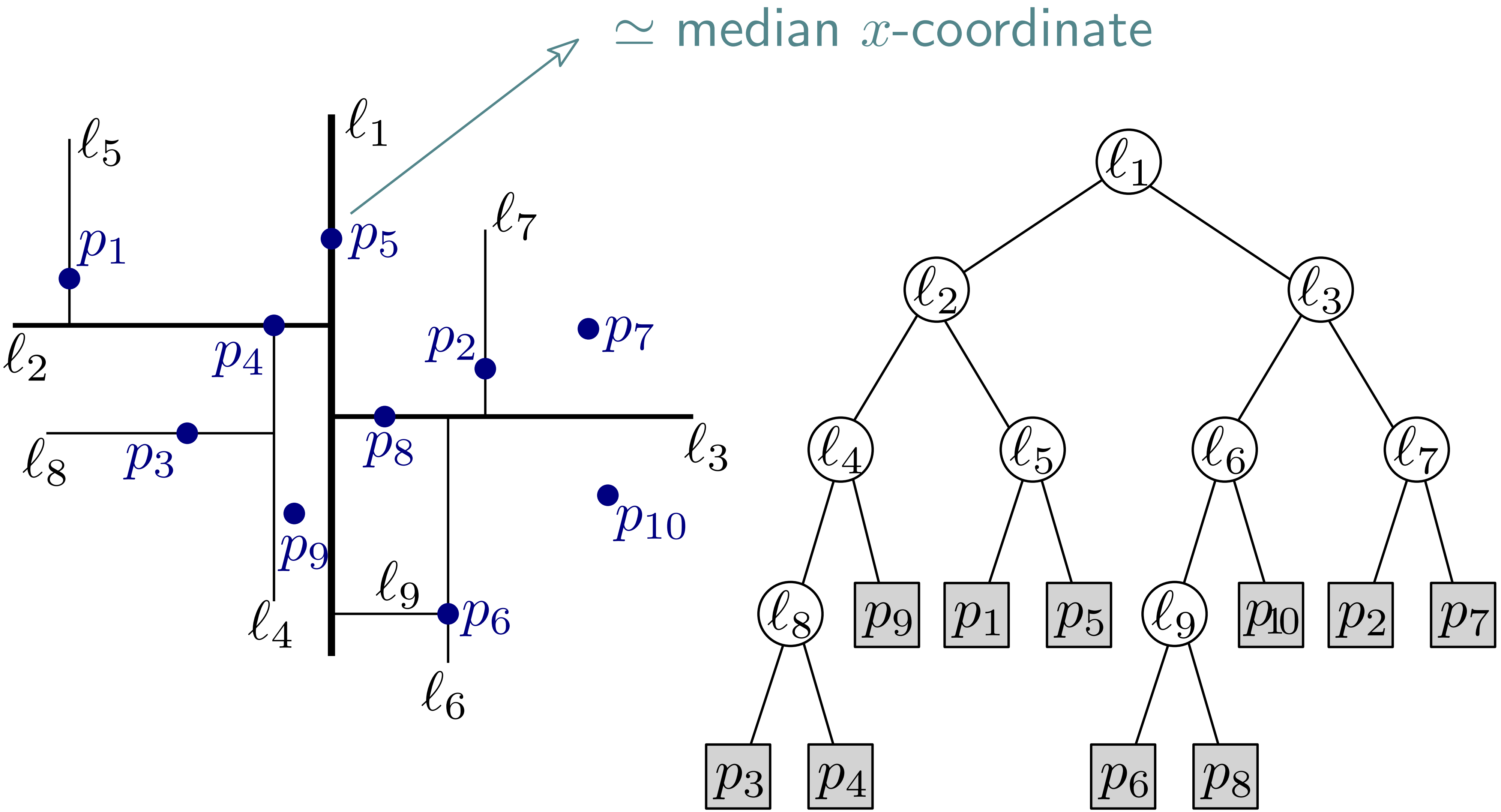




Kd trees in $\mathbb{R}^2$

Assume distinct x- and y-coords

Idea:



Kd tree anatomy

Space: $O(n)$
Tree has $O(\log n)$ depth.

Preprocessing: use linear time median:

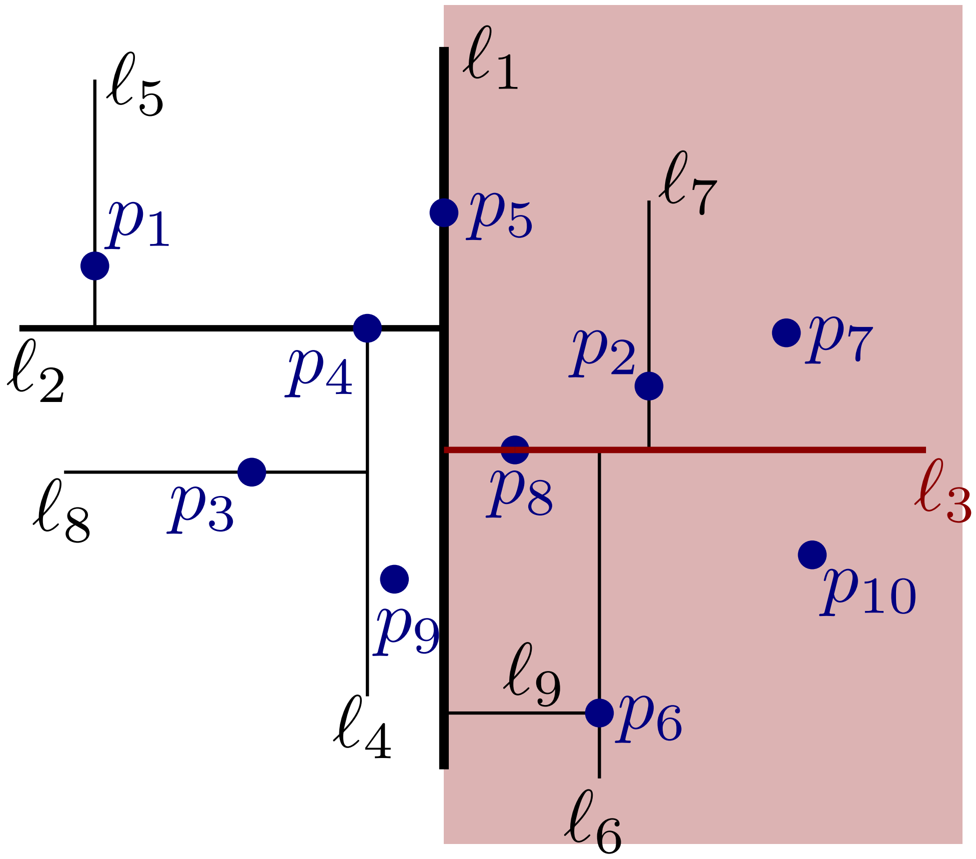
$$T(n) = 2T(n/2) + O(n) \ \Rightarrow O(n \log n)$$

Kd tree anatomy

Space: $O(n)$
Tree has $O(\log n)$ depth.

Preprocessing: use linear time median:

$$T(n) = 2T(n/2) + O(n) \ \Rightarrow O(n \log n)$$



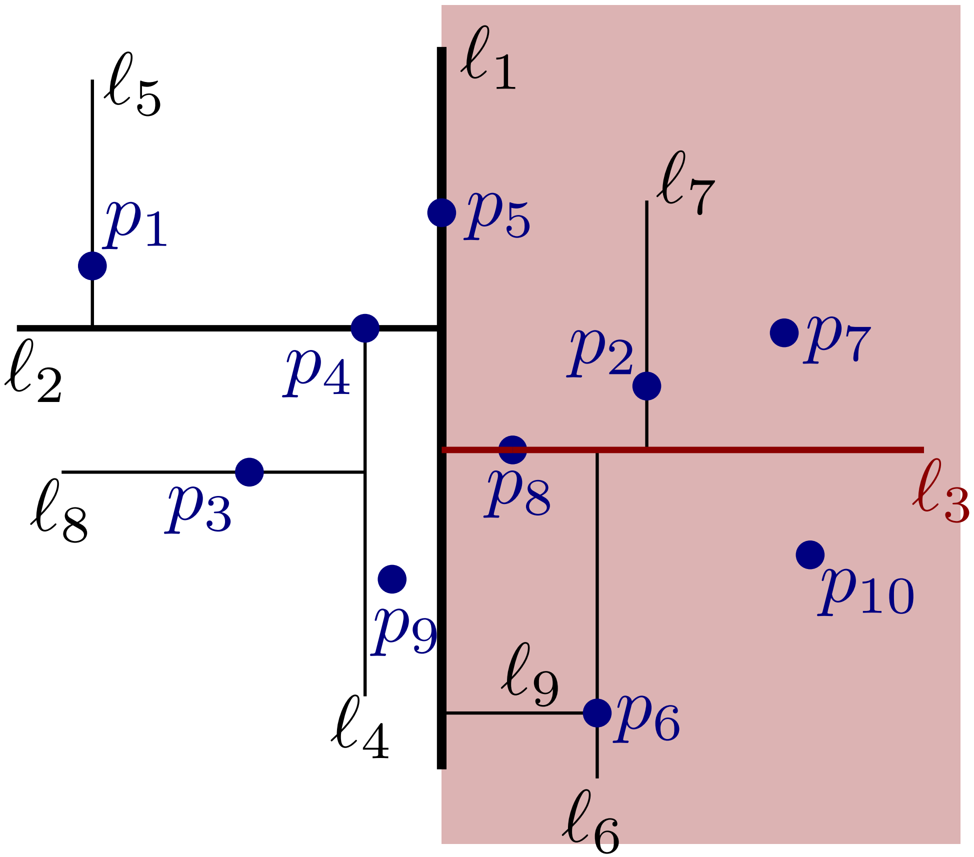
Each ℓ has a rectangular region

Kd tree anatomy

Space: $O(n)$
Tree has $O(\log n)$ depth.

Preprocessing: use linear time median:

$$T(n) = 2T(n/2) + O(n) \;\Rightarrow O(n \log n)$$

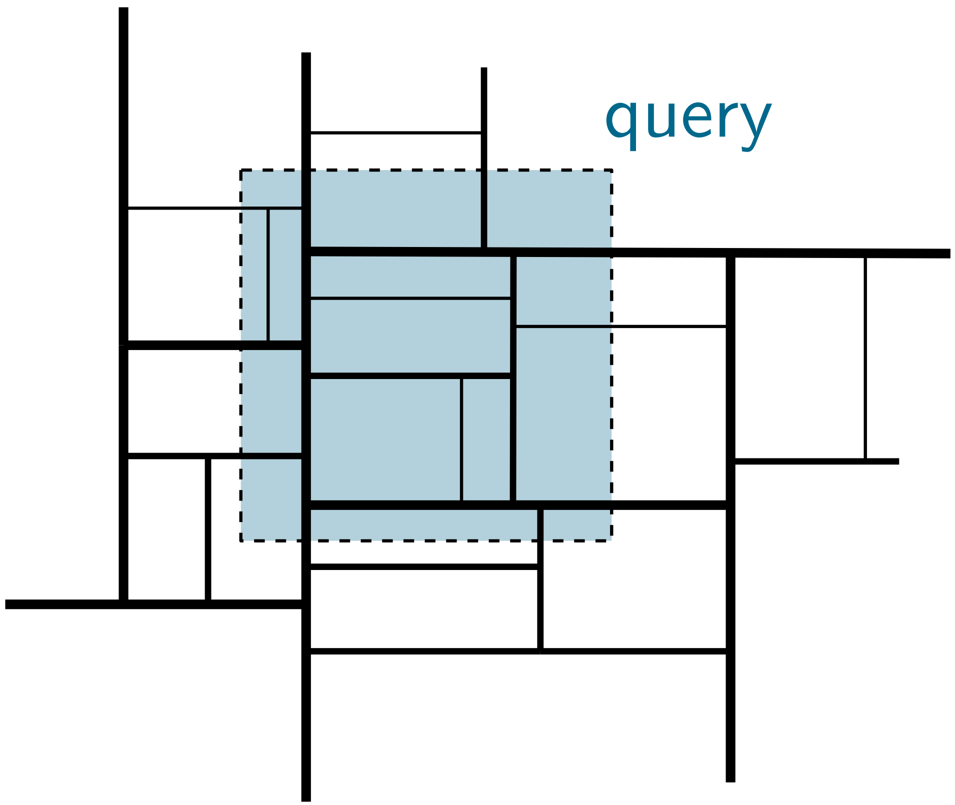


Each ℓ has a rectangular region

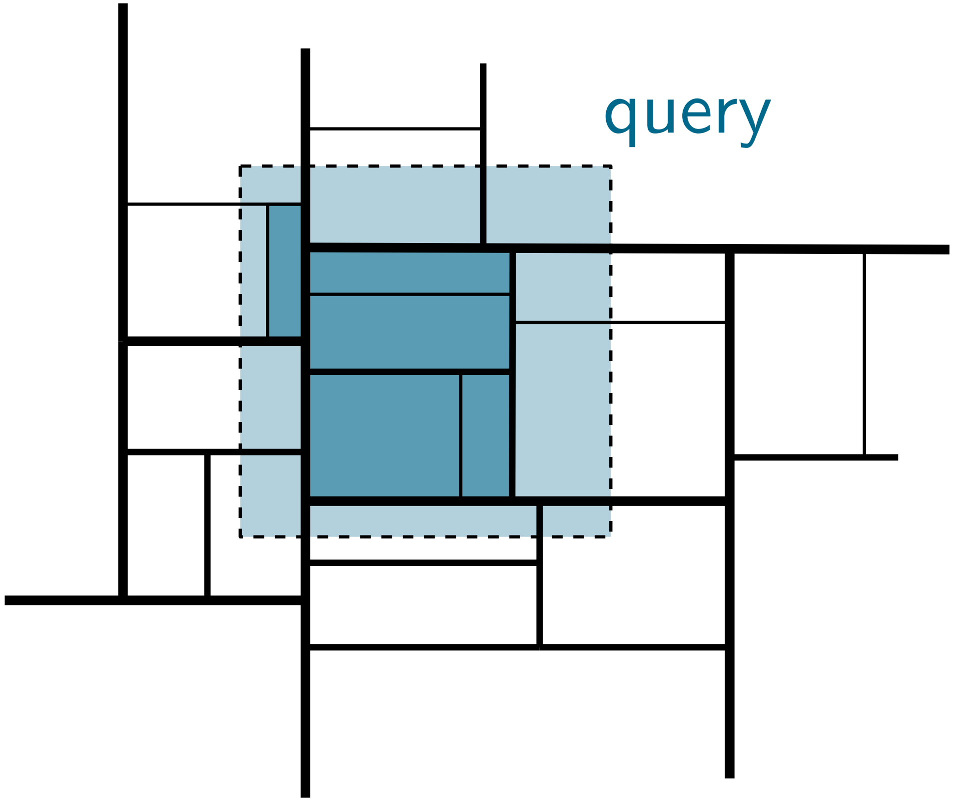
$$reg(\ell_1) = \mathbb{R}^2$$

Child regions of ℓ:
separated by ℓ

Querying a kd tree

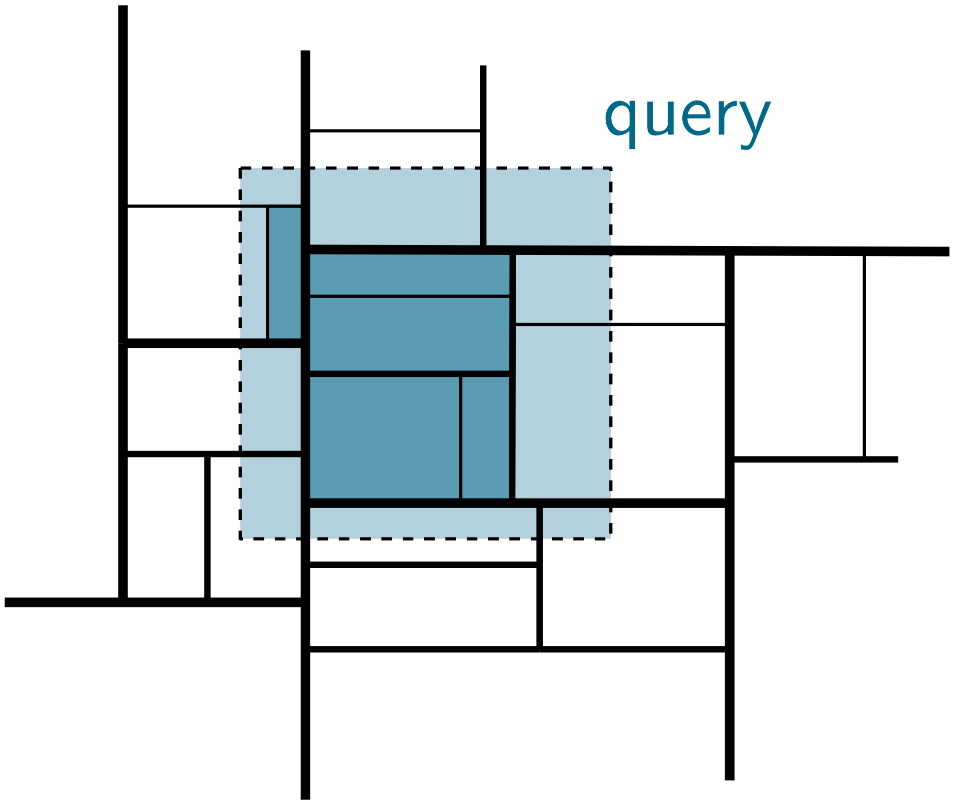


Querying a kd tree



Report subtree if $reg(\ell) \subseteq Q$

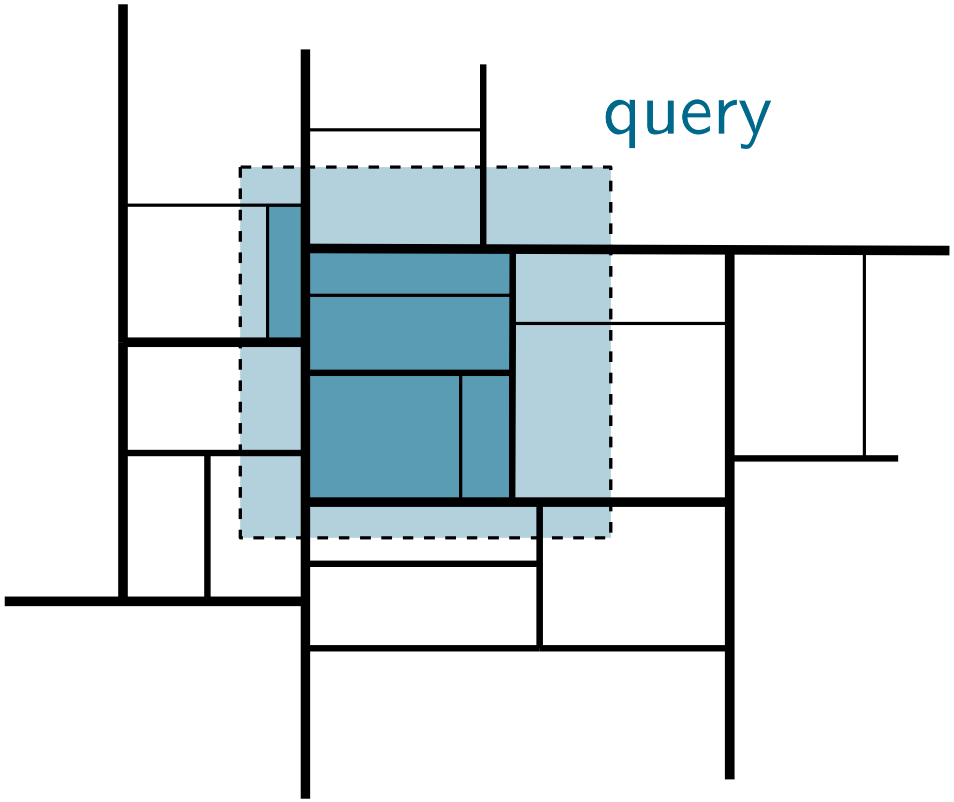
Querying a kd tree



Report subtree if $reg(\ell) \subseteq Q$

If $reg(\ell)$ intersects ∂Q: check!

Querying a kd tree

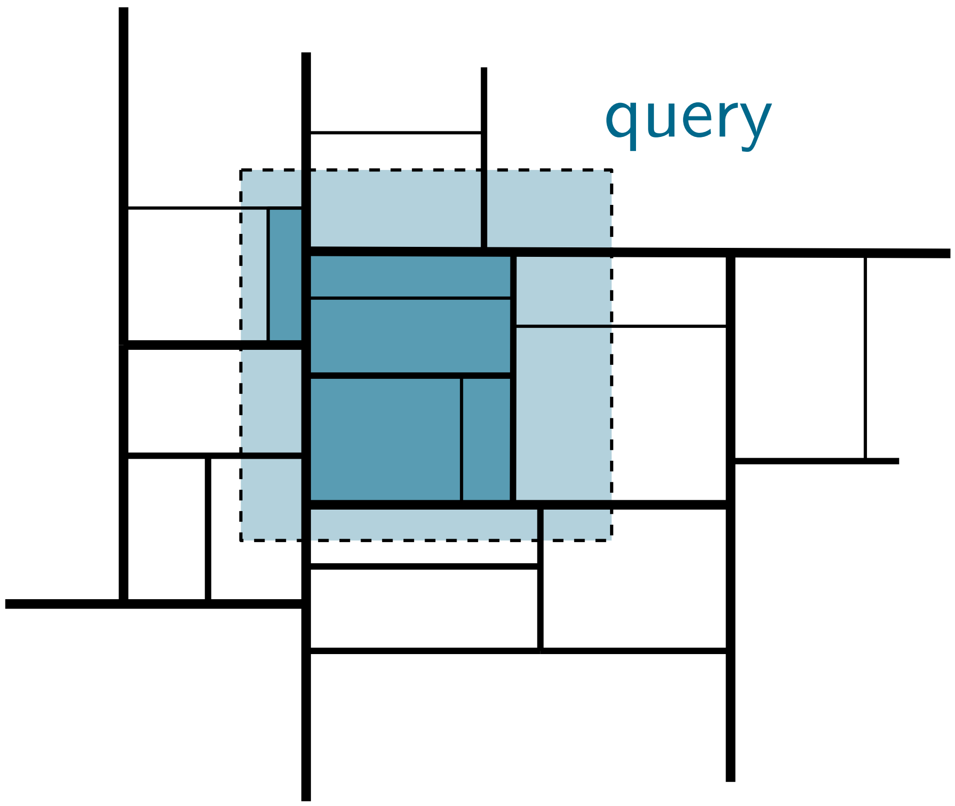


Report subtree if $reg(\ell) \subseteq Q$

If $reg(\ell)$ intersects ∂Q: check!

Searching, reporting covered regions: $O(\log n + k)$

Querying a kd tree



Report subtree if $reg(\ell) \subseteq Q$

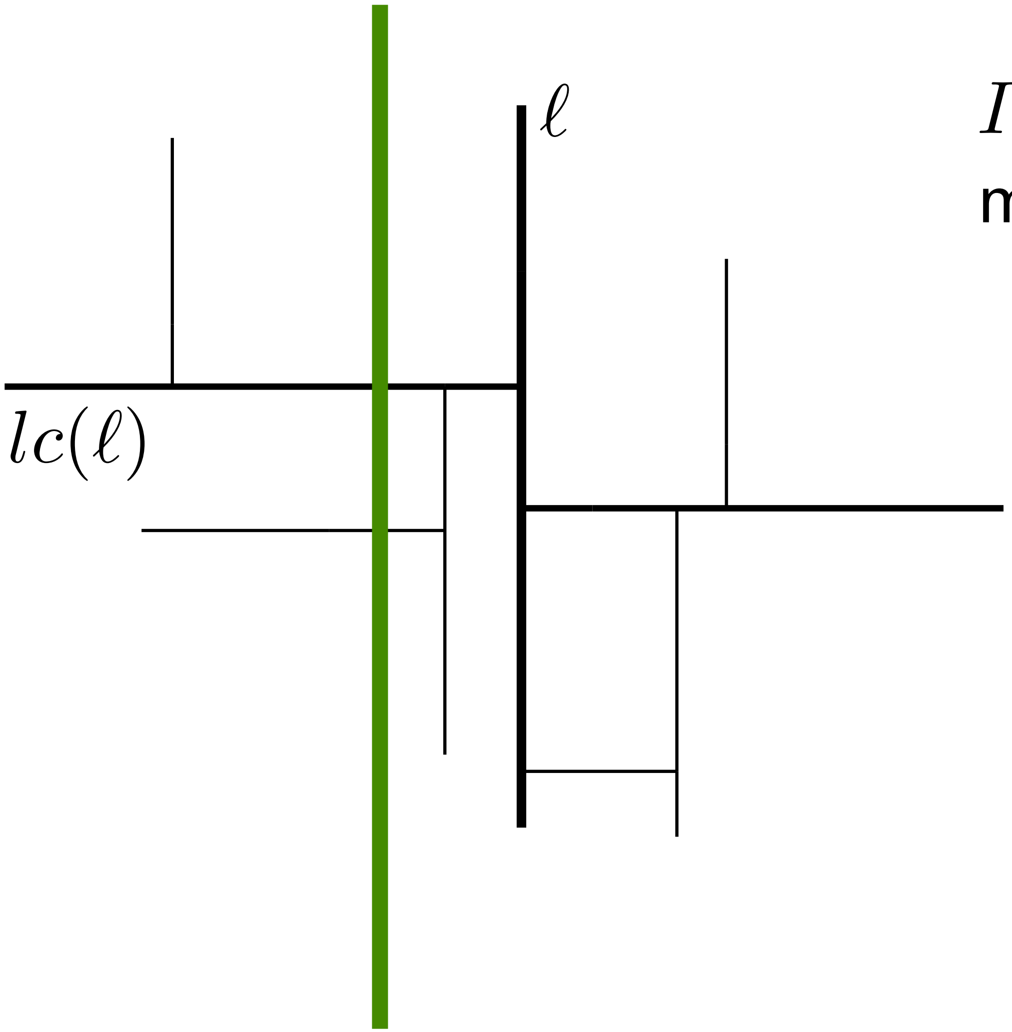
If $reg(\ell)$ intersects ∂Q: check!

Searching, reporting covered regions: $O(\log n + k)$

How many regions can intersect ∂Q?

Regions interseted by the boundary

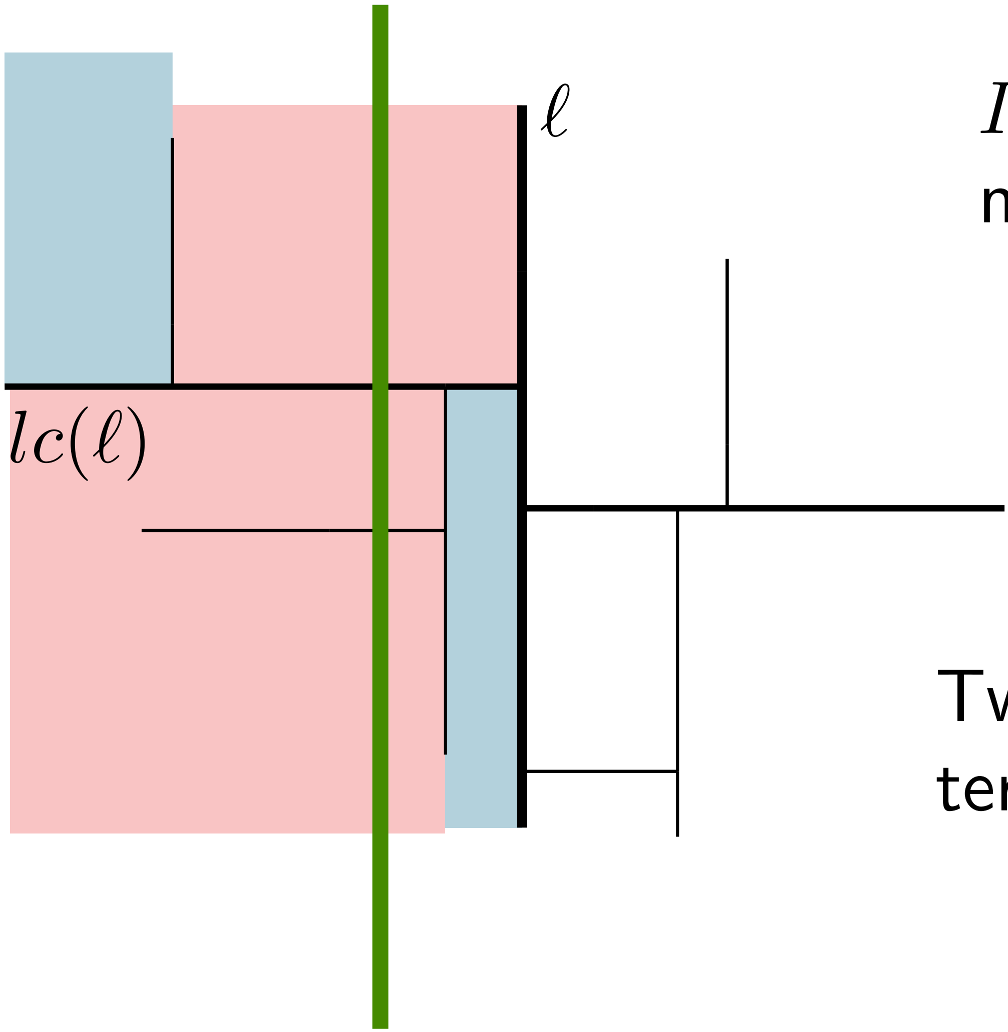
Claim. A vertical line intersects at most $O(\sqrt{n})$ node regions.



$I(n)$: vert. line intersects this many regions in kd tree of size n

Regions interseted by the boundary

Claim. A vertical line intersects at most $O(\sqrt{n})$ node regions.

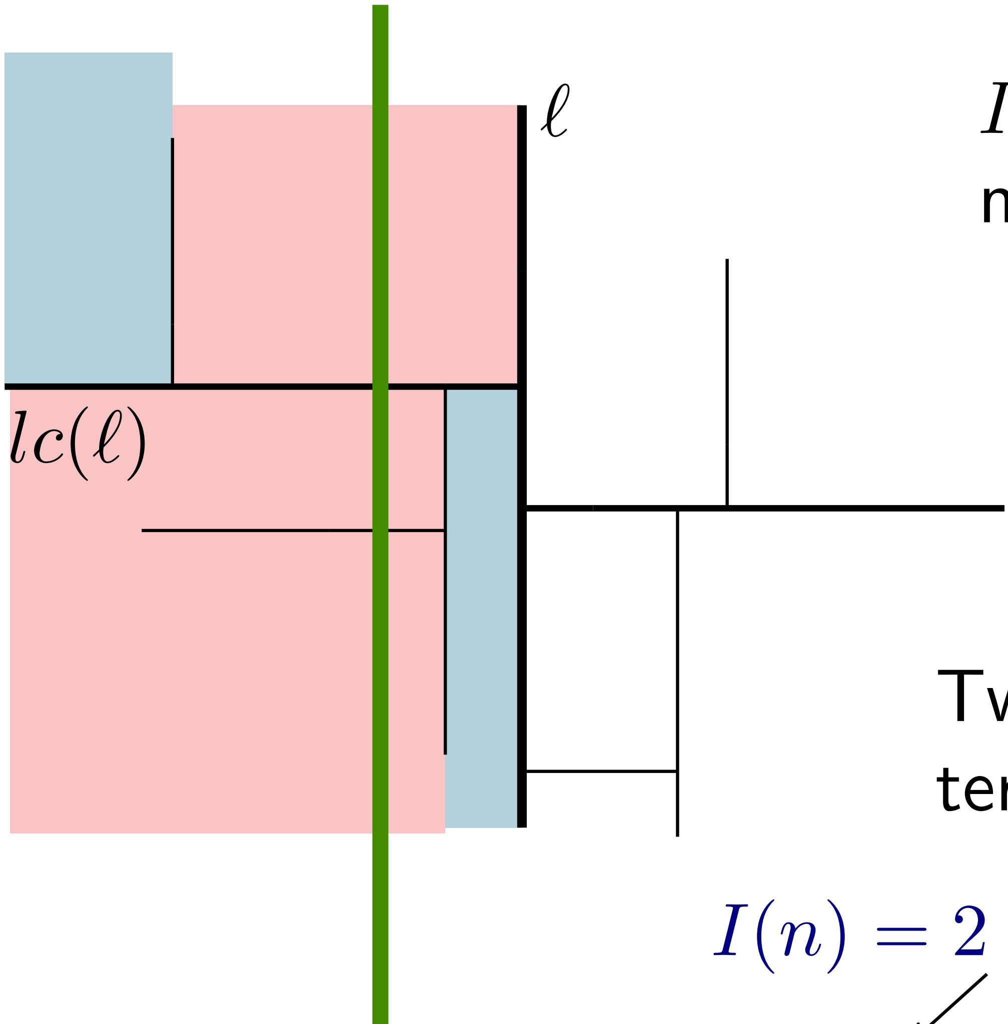


$I(n)$: vert. line intersects this many regions in kd tree of size n

Two grandchild regions are not intersected

Regions interseted by the boundary

Claim. A vertical line intersects at most $O(\sqrt{n})$ node regions.



$I(n)$: vert. line intersects this many regions in kd tree of size n

Two grandchild regions are not intersected

$$I(n) = 2 + 2I(n/4)$$

$reg(\ell)$ and $reg(\text{leftchild}(\ell))$

Intersections in red subtrees

Kd tree query time

$$I(n) = 2 + 2I(n/4) = 2 + 2 \cdot (2 + 2 \cdot (\dots)) = 2 + 4 + \dots + 2^{\log_4 n}$$

Kd tree query time

$$I(n) = 2 + 2I(n/4) = 2 + 2 \cdot (2 + 2 \cdot (\dots)) = 2 + 4 + \cdots + 2^{\log_4 n}$$

$$\Rightarrow I(n) = O(\sqrt{n}) \qquad \square$$

Kd tree query time

$$I(n) = 2 + 2I(n/4) = 2 + 2\cdot(2 + 2\cdot(\dots)) = 2 + 4 + \cdots + 2^{\log_4 n}$$

$$\Rightarrow I(n) = O(\sqrt{n}) \qquad \square$$

Q has ≤ 2 vertical and ≤ 2 horizontal sides
$\Rightarrow \partial Q$ intersects $O(\sqrt{n})$ regions

Kd tree query time

$$I(n) = 2 + 2I(n/4) = 2 + 2 \cdot (2 + 2 \cdot (\dots)) = 2 + 4 + \dots + 2^{\log_4 n}$$

$$\Rightarrow I(n) = O(\sqrt{n}) \qquad \square$$

Q has ≤ 2 vertical and ≤ 2 horizontal sides
$\Rightarrow \partial Q$ intersects $O(\sqrt{n})$ regions

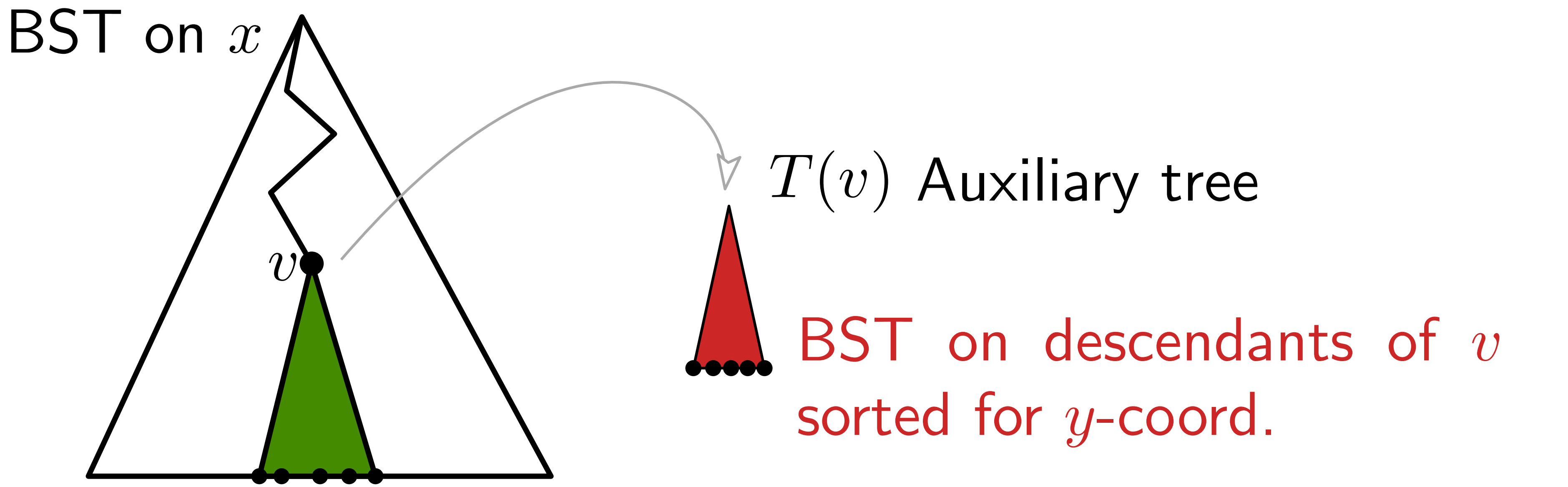
Orthogonal range queries of a 2-dim kd tree take $O(\sqrt{n} + k)$ time.

Range trees

Range tree in $\mathbb{R}^2$

Binary search tree on x-xoordinates

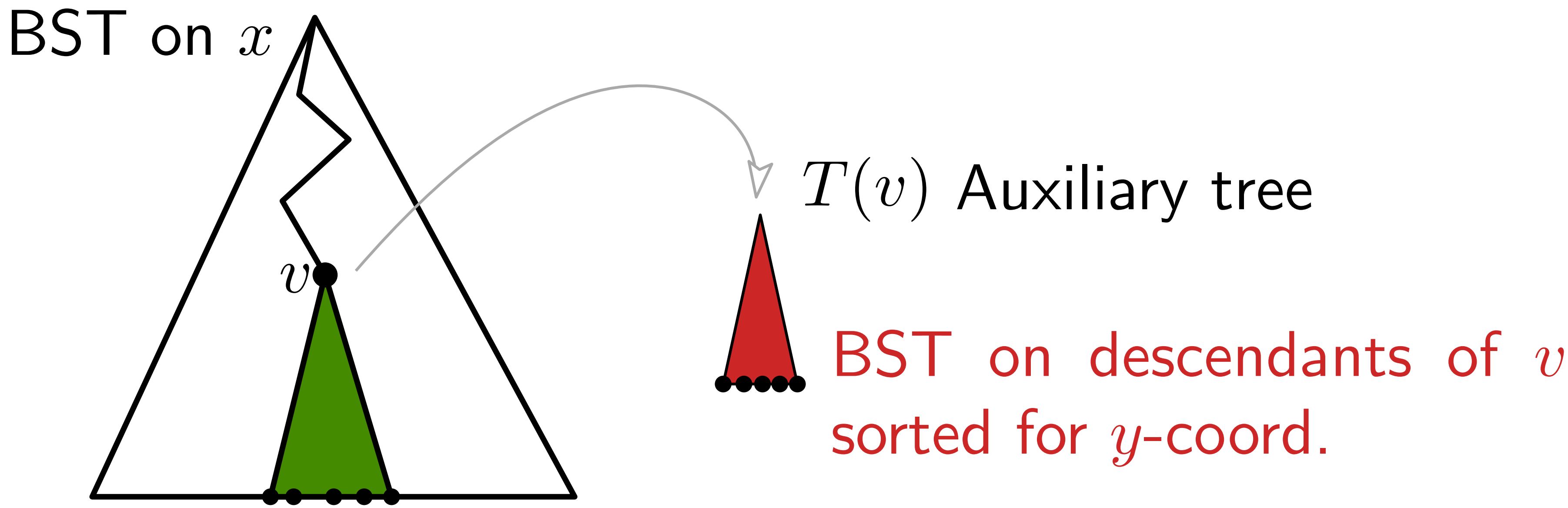
Each node has a new BST for y-coords of descendant leaves



Range tree in $\mathbb{R}^2$

Binary search tree on x-xoordinates

Each node has a new BST for y-coords of descendant leaves



Construction:

- Presort on y-coord to array A

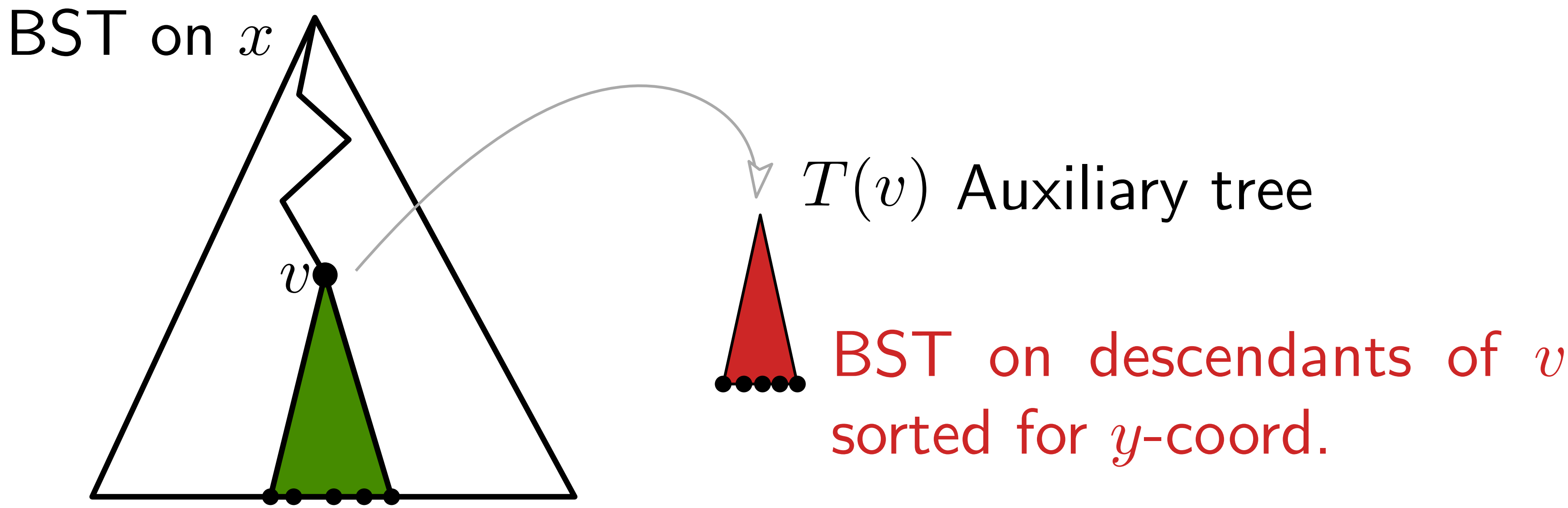
Construct(v, A):

- Construct $T(v)$ using A
- Find median x, split A to A_{left}, A_{right}
- Construct($lc(v), A_{left}$), Construct($rc(v), A_{right}$)

Range tree in $\mathbb{R}^2$

Binary search tree on x-xoordinates

Each node has a new BST for y-coords of descendant leaves



Construction:

- Presort on y-coord to array A $O(n \log n)$

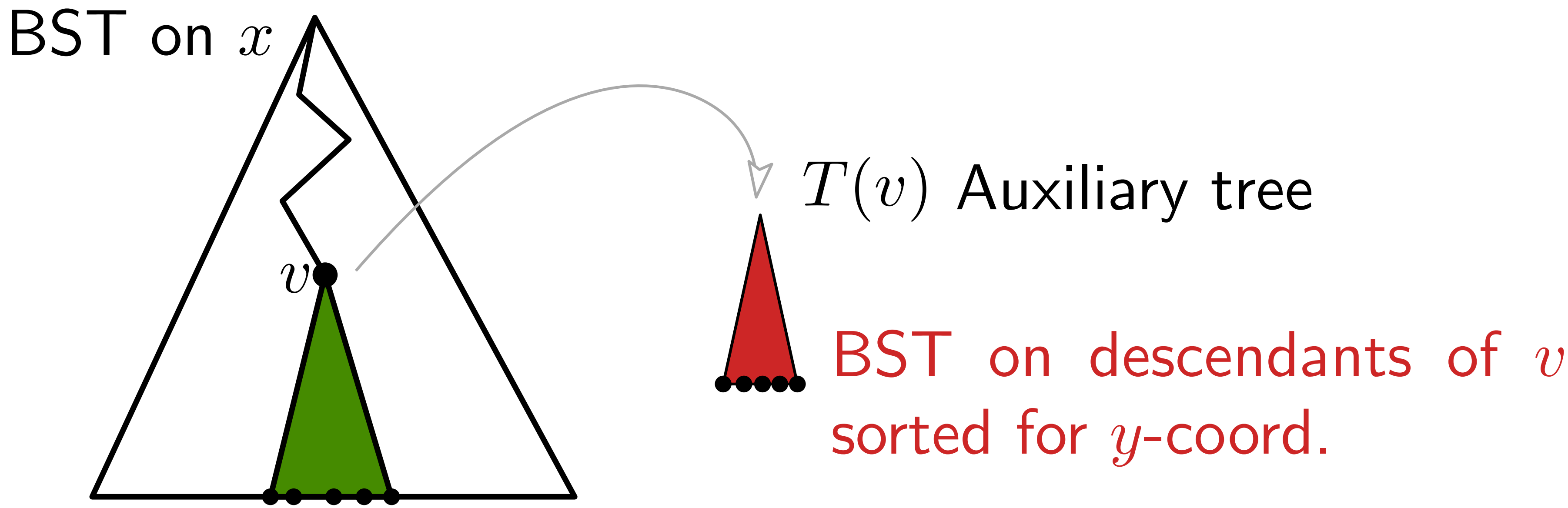
Construct(v, A):

- Construct $T(v)$ using A $O(n)$
- Find median x, split A to A_{left}, A_{right} $O(n)$
- Construct($lc(v), A_{left}$), Construct($rc(v), A_{right}$)

Range tree in $\mathbb{R}^2$

Binary search tree on x-xoordinates

Each node has a new BST for y-coords of descendant leaves



Construction:

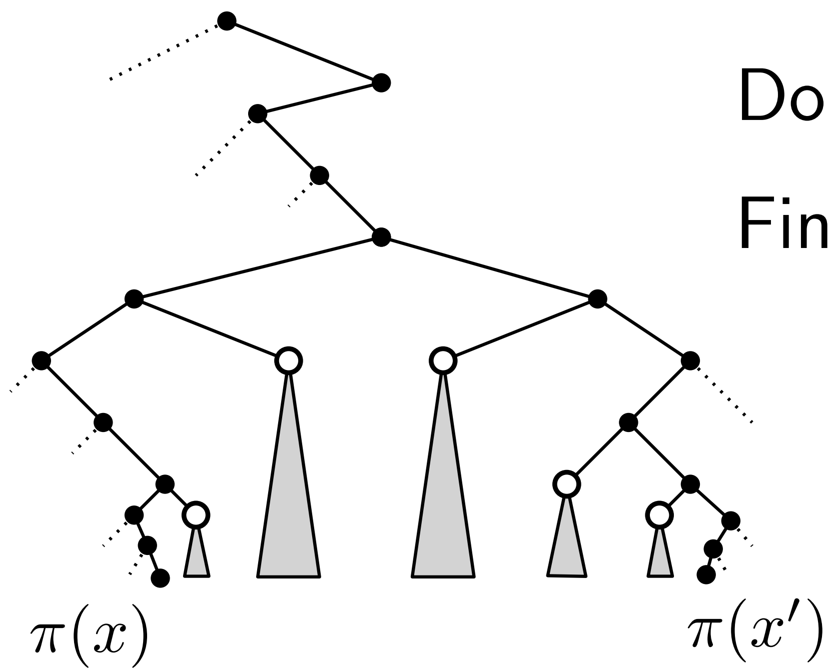
- Presort on y-coord to array A $\quad O(n \log n)$

Construct(v, A):

- Construct $T(v)$ using A $\quad O(n)$
- Find median x, split A to A_{left}, A_{right} $\quad O(n)$
- Construct($lc(v), A_{left}$), Construct($rc(v), A_{right}$)

$O(n \log n)$

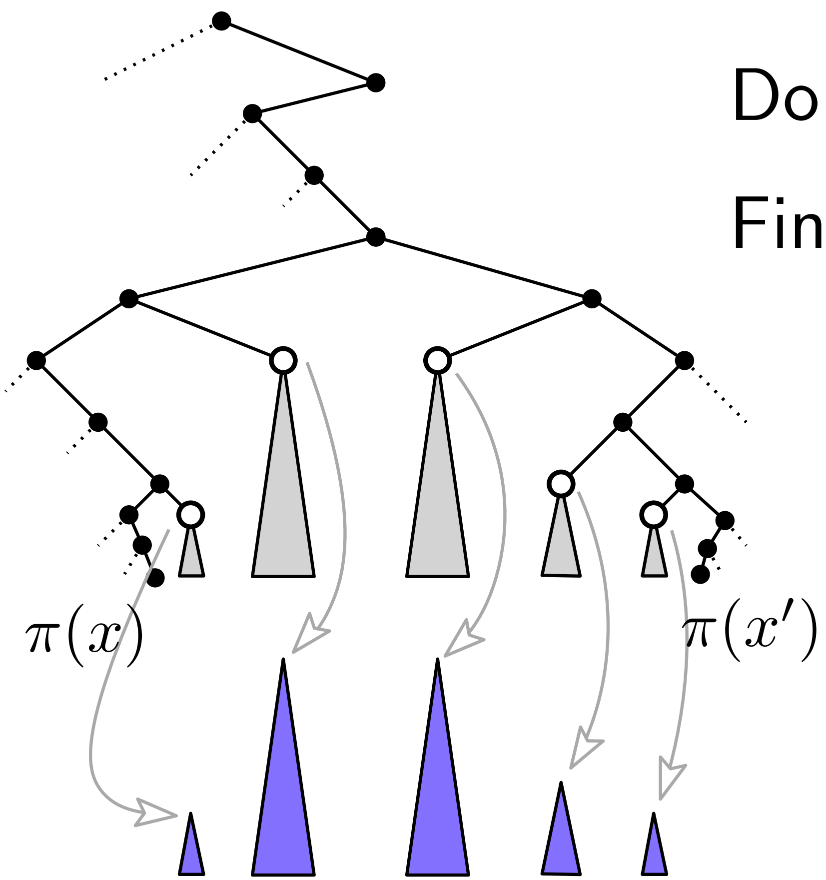
Querying a 2-dim range tree



Do 1-dim range query on x-coordiantes

Find $O(\log n)$ reported subtree roots

Querying a 2-dim range tree



Do 1-dim range query on x-coordiantes

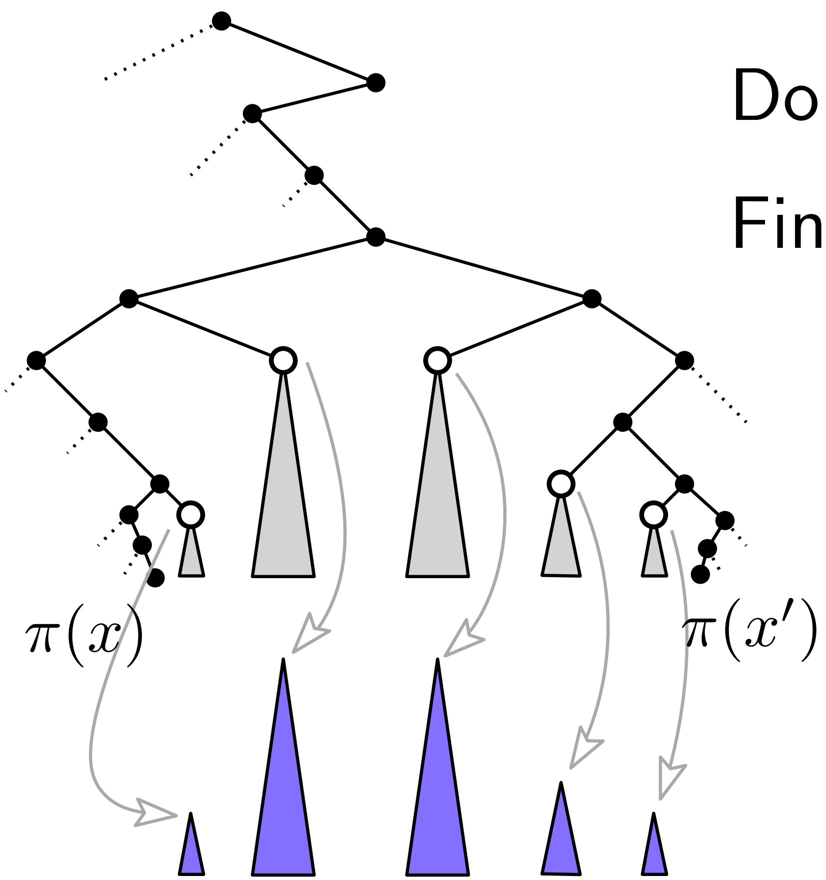
Find $O(\log n)$ reported subtree roots

Do 1-dim query on each of the $O(\log n)$ selected y-trees

Querying a 2-dim range tree

Do 1-dim range query on x-coordiantes

Find $O(\log n)$ reported subtree roots

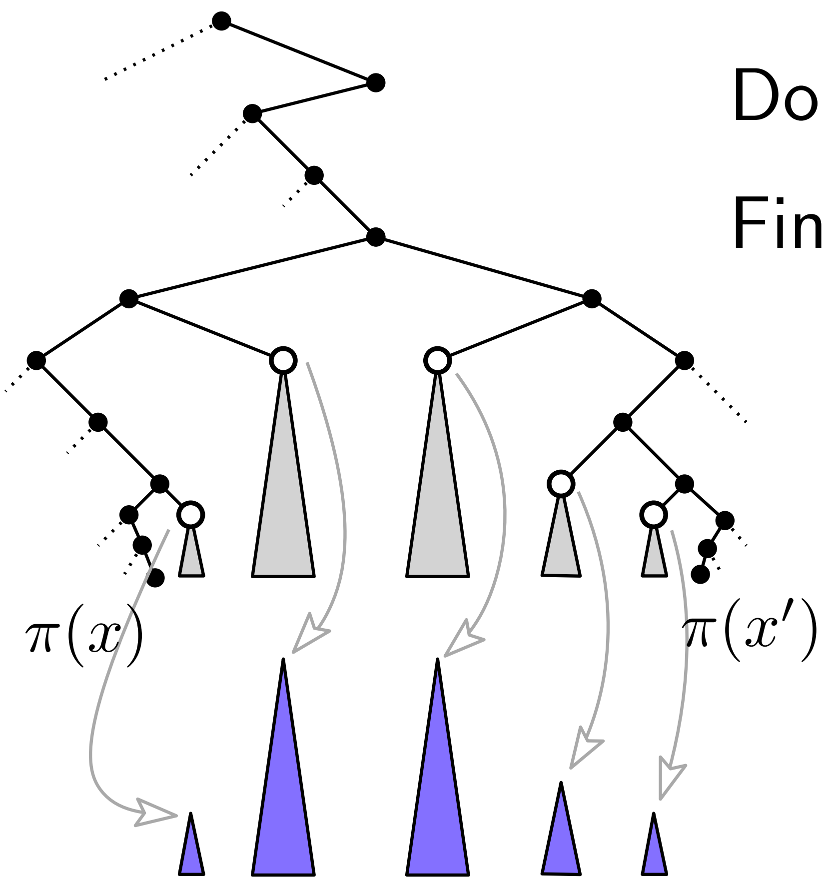


Do 1-dim query on each of the $O(\log n)$ selected y-trees

Total query time:

$$\sum_v O(\log n + k_v) = O(\log^2 n + k)$$

Querying a 2-dim range tree



Do 1-dim range query on x-coordiantes

Find $O(\log n)$ reported subtree roots

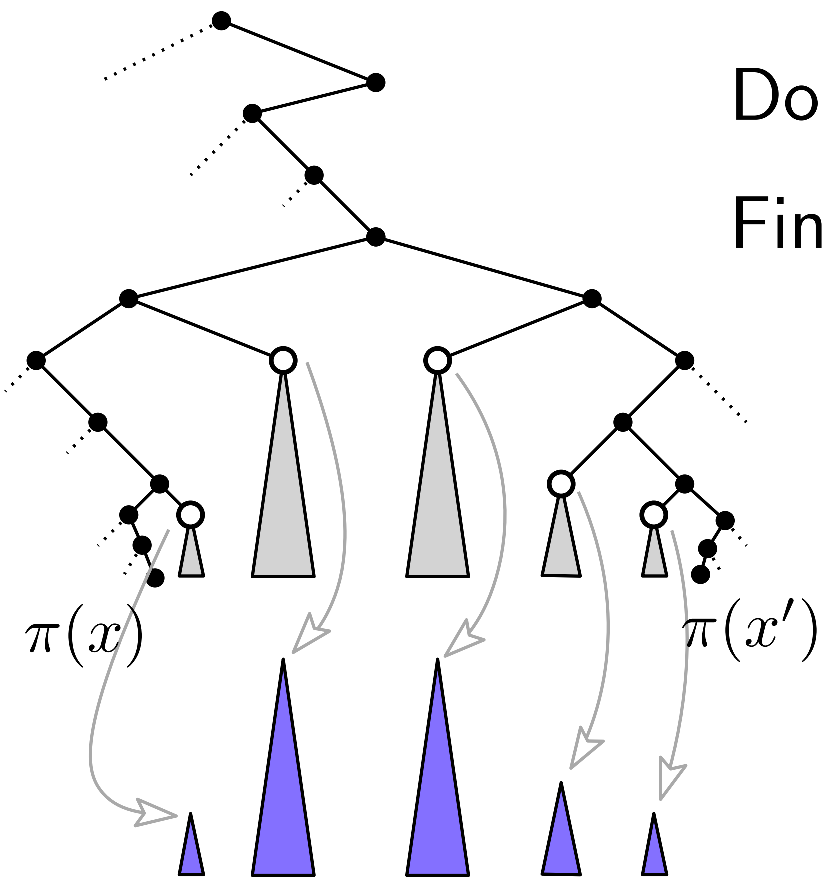
Do 1-dim query on each of the $O(\log n)$ selected y-trees

Total query time:

$$\sum_v O(\log n + k_v) = O(\log^2 n + k)$$

In $\mathbb{R}^d$, it gives $O(\log^d n + k)$

Querying a 2-dim range tree



Do 1-dim range query on x-coordiantes

Find $O(\log n)$ reported subtree roots

Do 1-dim query on each of the $O(\log n)$ selected y-trees

Each level stores chunks of a $(d-1)$-dim range tree

Total query time:

$$\sum_v O(\log n + k_v) = O(\log^2 n + k)$$

In $\mathbb{R}^d$, it gives $O(\log^d n + k)$ $\boxed{\text{But! space is up to } \Theta(n \log^{d-1} n)}$

Handling degeneracies

Composite numbers: $a, b \in \mathbb{R} \rightarrow (a|b)$
Sorted lexicographically.

Handling degeneracies

Composite numbers: $a, b \in \mathbb{R} \to (a|b)$
Sorted lexicographically.

repalce $p = (x, y)$ with $p^* = ((x|y), (y|x))$ for each $p \in P$

$\Rightarrow$ all first and second coords are distinct

Handling degeneracies

Composite numbers: $a, b \in \mathbb{R} \to (a|b)$
Sorted lexicographically.

repalce $p = (x, y)$ with $p^* = ((x|y), (y|x))$ for each $p \in P$

$\Rightarrow$ all first and second coords are distinct

Given query $Q = [x, x'] \times [y, y']$, replace by

$$Q^* = \left[(x| -\infty),\ (x'|\infty)\right] \times \left[(y| -\infty),\ (y'|\infty)\right]$$

Handling degeneracies

Composite numbers: $a, b \in \mathbb{R} \rightarrow (a|b)$
Sorted lexicographically.

repalce $p = (x, y)$ with $p^* = ((x|y), (y|x))$ for each $p \in P$

$\Rightarrow$ all first and second coords are distinct

Given query $Q = [x, x'] \times [y, y']$, replace by

$$Q^* = \big[(x| -\infty),\ (x'|\infty)\big] \times \big[(y| -\infty),\ (y'|\infty)\big]$$

Observation.

$$p \in Q \Leftrightarrow p^* \in Q^*$$

(Special) Fractional cascading

Lueker 1978, Willard 1978
Chazelle and Guibas 1986

Speeding up binary searches on subsets

Array and subarray, query $[4, 45]$
$\Rightarrow$ two binary searhces?

A	2	3	5	8	13	21	34	55	89	92	95	99

B	2	8	13	34	55	92	95

Speeding up binary searches on subsets

Array and subarray, query $[4, 45]$
$\Rightarrow$ two binary searhces?

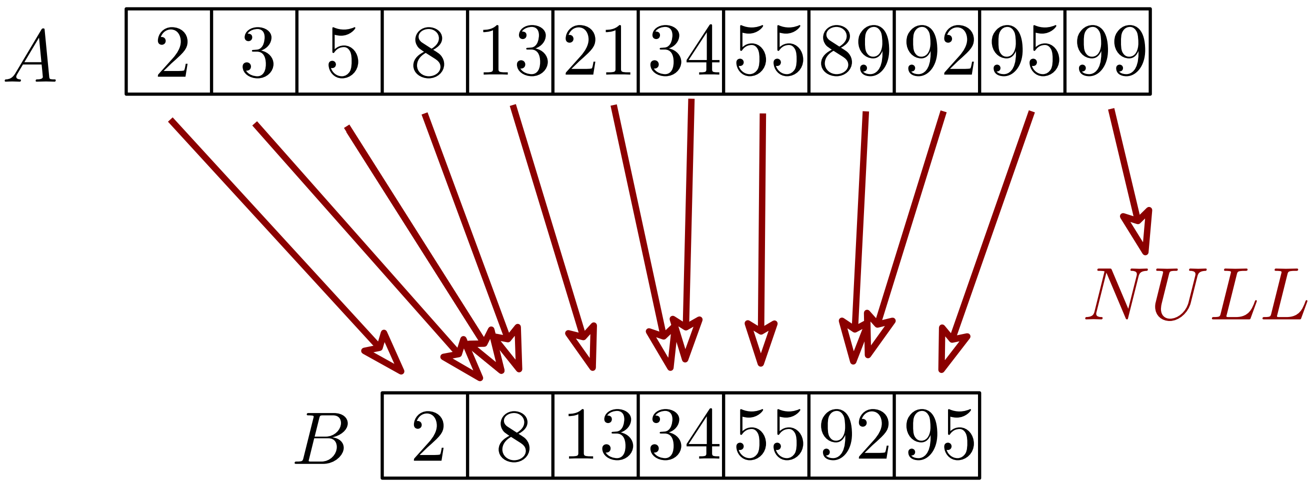
A	2	3	5	8	13	21	34	55	89	92	95	99

B	2	8	13	34	55	92	95

Add pointer from $A[x]$ to the smallest in B larger than $A[x]$

Speeding up binary searches on subsets

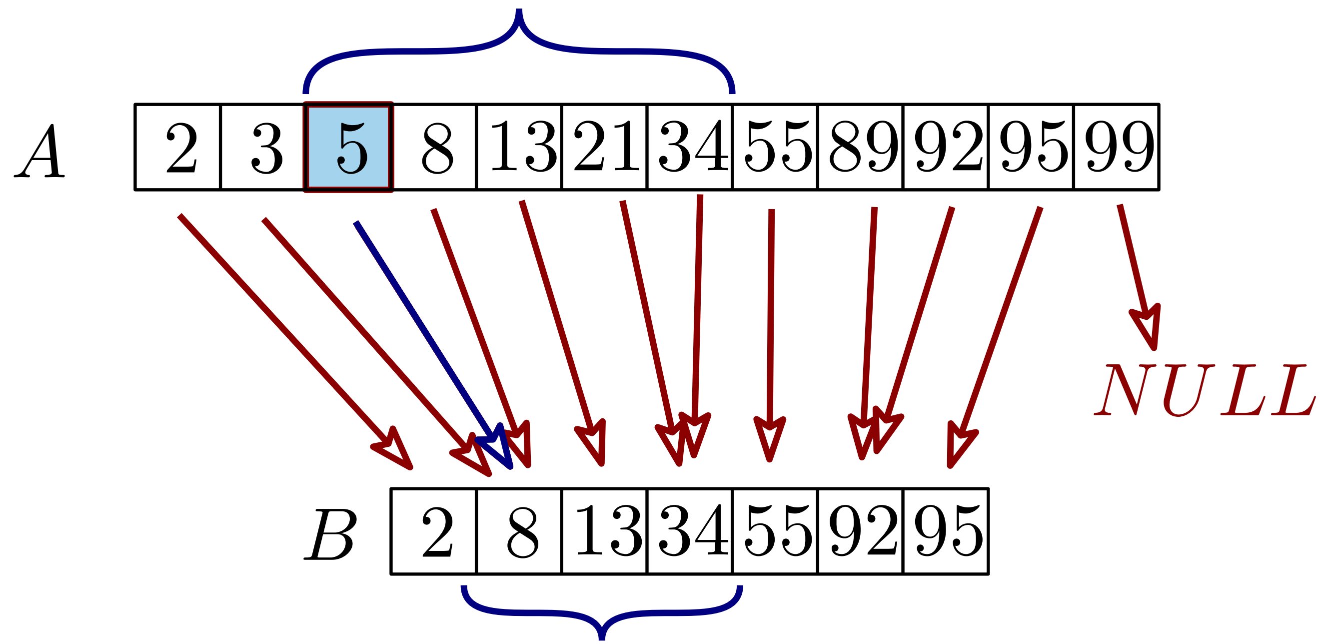
Array and subarray, query $[4, 45]$
$\Rightarrow$ two binary searhces?



Add pointer from $A[x]$ to the smallest in B larger than $A[x]$

Speeding up binary searches on subsets

Array and subarray, query $[4, 45]$
$\Rightarrow$ two binary searhces?



Add pointer from $A[x]$ to the smallest in B larger than $A[x]$

Binary search A, use pointer of smallest reported

Layered range trees in $\mathbb{R}^2$

Idea:

Replace 2nd level BSTs for y-coords w/ sorted arrays A_v
$+$ 2 pointers per entry

At node v, array A_v has:

Layered range trees in $\mathbb{R}^2$

Idea:
Replace 2nd level BSTs for y-coords w/ sorted arrays A_v
$+$ 2 pointers per entry

At node v, array A_v has:

- Points in decendant leafs of v sorted by y-coords

Layered range trees in $\mathbb{R}^2$

Idea:

Replace 2nd level BSTs for y-coords w/ sorted arrays A_v
+ 2 pointers per entry

At node v, array A_v has:

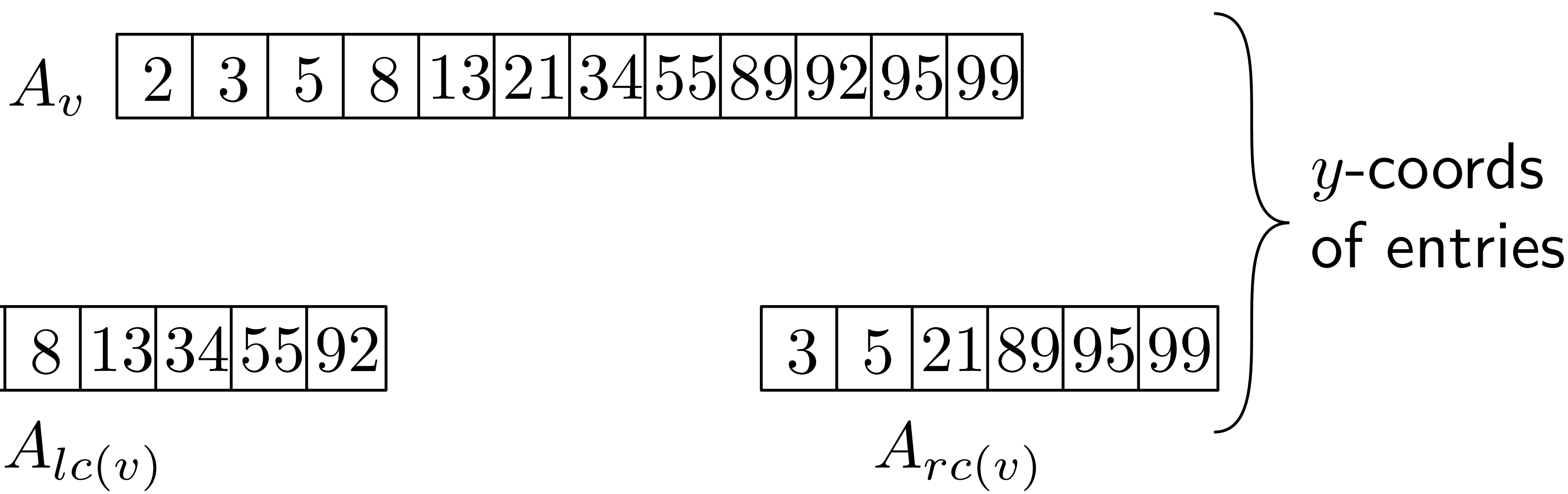
- Points in decendant leafs of v sorted by y-coords
- Each entry $(a, b) \in A_v$ has:
  - Pointer to entry in $A_{lc(v)}$ with smallest y-coord $\geq b$
  - Pointer to entry in $A_{rc(v)}$ with smallest y-coord $\geq b$

Layered range trees in $\mathbb{R}^2$

Idea:
Replace 2nd level BSTs for y-coords w/ sorted arrays A_v
$+$ 2 pointers per entry

At node v, array A_v has:

- Points in decendant leafs of v sorted by y-coords
- Each entry $(a, b) \in A_v$ has:
  - Pointer to entry in $A_{lc(v)}$ with smallest y-coord $\geq b$
  - Pointer to entry in $A_{rc(v)}$ with smallest y-coord $\geq b$

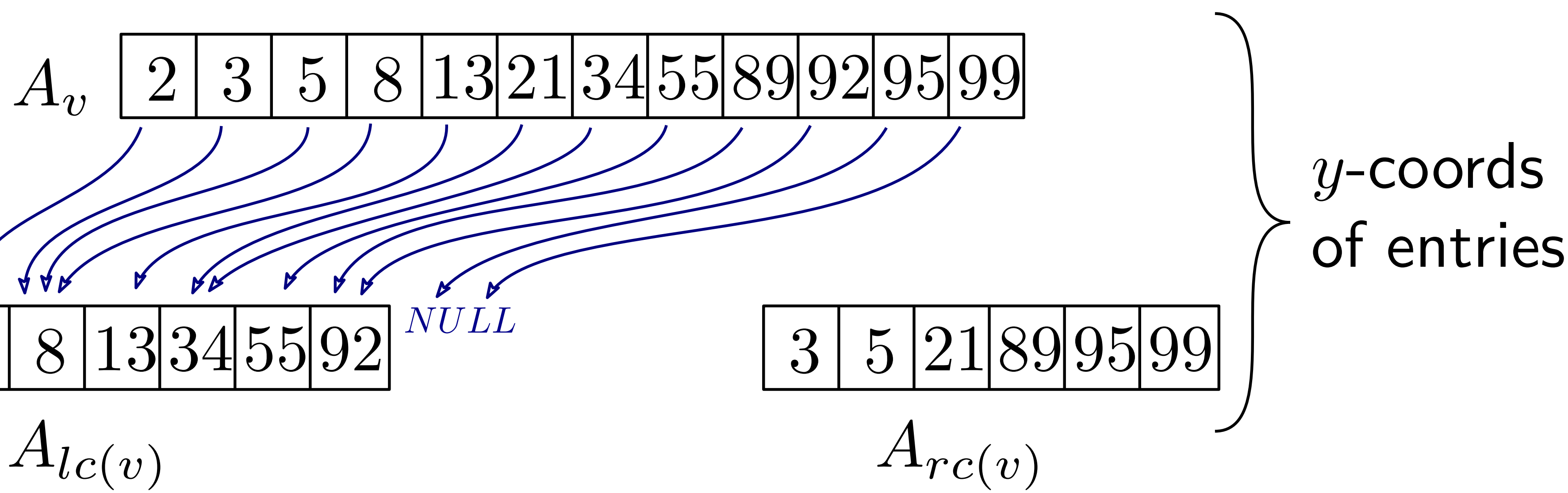


Layered range trees in $\mathbb{R}^2$

Idea:
Replace 2nd level BSTs for y-coords w/ sorted arrays A_v
+ 2 pointers per entry

At node v, array A_v has:

- Points in decendant leafs of v sorted by y-coords
- Each entry $(a, b) \in A_v$ has:
  - Pointer to entry in $A_{lc(v)}$ with smallest y-coord $\geq b$
  - Pointer to entry in $A_{rc(v)}$ with smallest y-coord $\geq b$



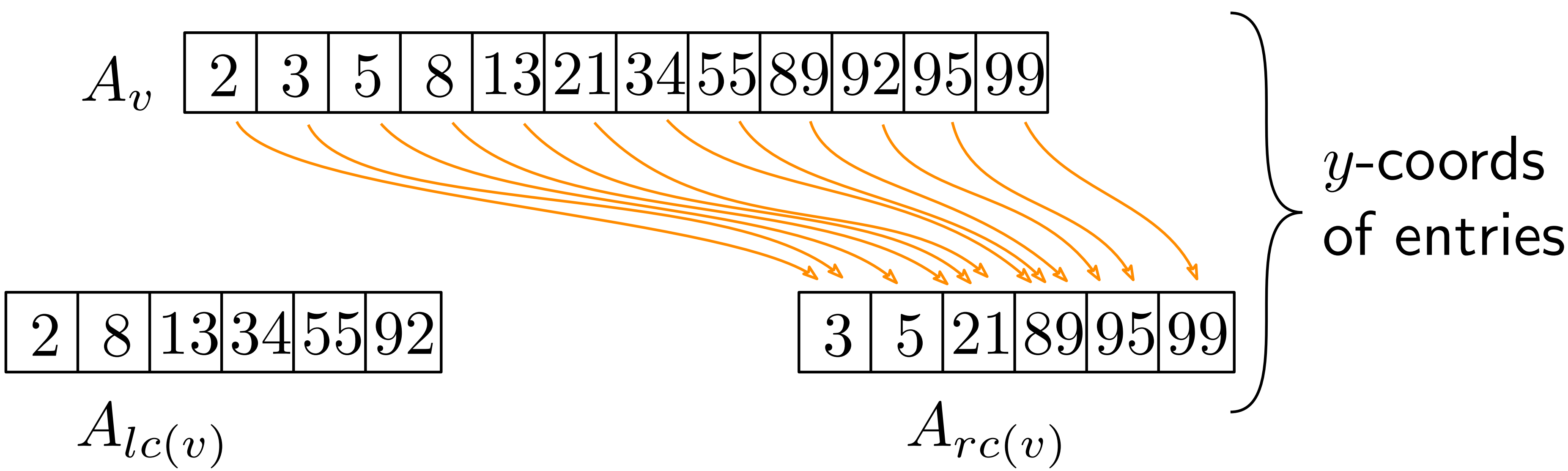
Layered range trees in $\mathbb{R}^2$

Idea:

Replace 2nd level BSTs for y-coords w/ sorted arrays A_v
+ 2 pointers per entry

At node v, array A_v has:

- Points in decendant leafs of v sorted by y-coords
- Each entry $(a, b) \in A_v$ has:
  - Pointer to entry in $A_{lc(v)}$ with smallest y-coord $\geq b$
  - Pointer to entry in $A_{rc(v)}$ with smallest y-coord $\geq b$



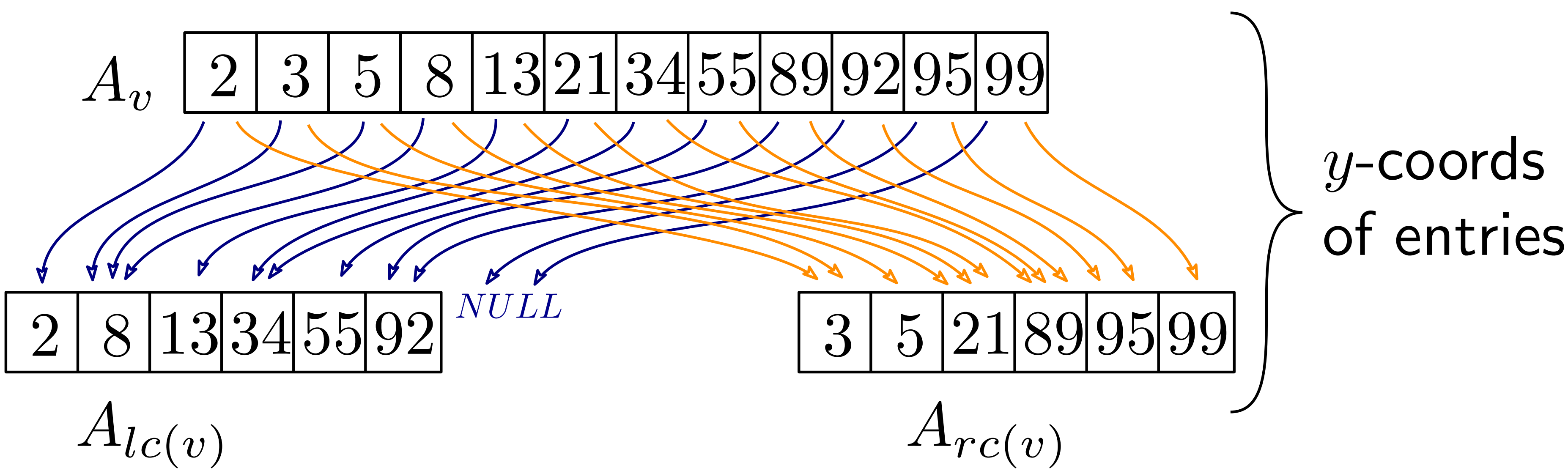
Layered range trees in $\mathbb{R}^2$

Idea:

Replace 2nd level BSTs for y-coords w/ sorted arrays A_v
+ 2 pointers per entry

At node v, array A_v has:

- Points in decendant leafs of v sorted by y-coords
- Each entry $(a, b) \in A_v$ has:
  - Pointer to entry in $A_{lc(v)}$ with smallest y-coord $\geq b$
  - Pointer to entry in $A_{rc(v)}$ with smallest y-coord $\geq b$



Querying a layered range tree

Query: $[x, x'] \times [y, y']$

Search top (x-)tree for x and x'.
At $v = Split(x, x')$, binary search for y in A_v $\quad \rightarrow A_v.find(y)$

Querying a layered range tree

Query: $[x, x'] \times [y, y']$

Search top (x-)tree for x and x'.
At $v = Split(x, x')$, binary search for y in A_v $\quad \rightarrow A_v.find(y)$

Stepping to $lc(v)$, follow left pointer from $A_v.find(y)$

Querying a layered range tree

Query: $[x, x'] \times [y, y']$

Search top (x-)tree for x and x'.
At $v = Split(x, x')$, binary search for y in A_v $\quad \rightarrow A_v.find(y)$

Stepping to $lc(v)$, follow left pointer from $A_v.find(y)$

$\Rightarrow$ Maintaining position in $A_{lc(v)}$ at each step takes $O(1)$ time!

If w is root of a selected subtree:

Reporting from $T(w)$ can be done from A_w
in $O(1 + k_w)$ time.

Querying a layered range tree

Query: $[x, x'] \times [y, y']$

Search top (x-)tree for x and x'.
At $v = Split(x, x')$, binary search for y in A_v $\quad \rightarrow A_v.find(y)$

Stepping to $lc(v)$, follow left pointer from $A_v.find(y)$

$\Rightarrow$ Maintaining position in $A_{lc(v)}$ at each step takes $O(1)$ time!

If w is root of a selected subtree:
Reporting from $T(w)$ can be done from A_w
in $O(1 + k_w)$ time.

Total query time: $O(\log n + k)$

Priority search trees
McCreight 1985

Priority search tree

Priority queue (insert, pop max)

$+$

Binary search tree

Priority search tree

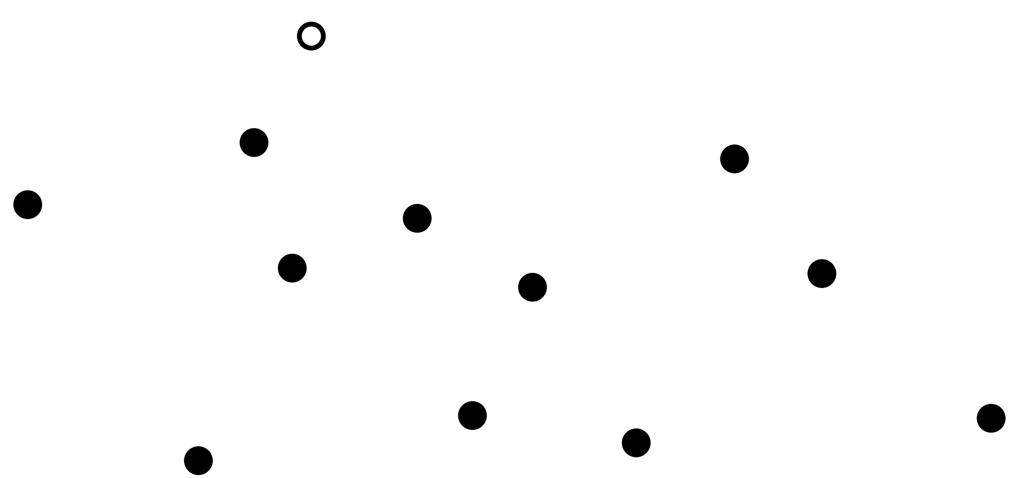
Priority queue (insert, pop max)

$+$

Binary search tree

Construction:

- root r stores point with max y-coord
  $med_x(root) :=$ median x-coordinate of $P \setminus \{r\}$



Priority search tree

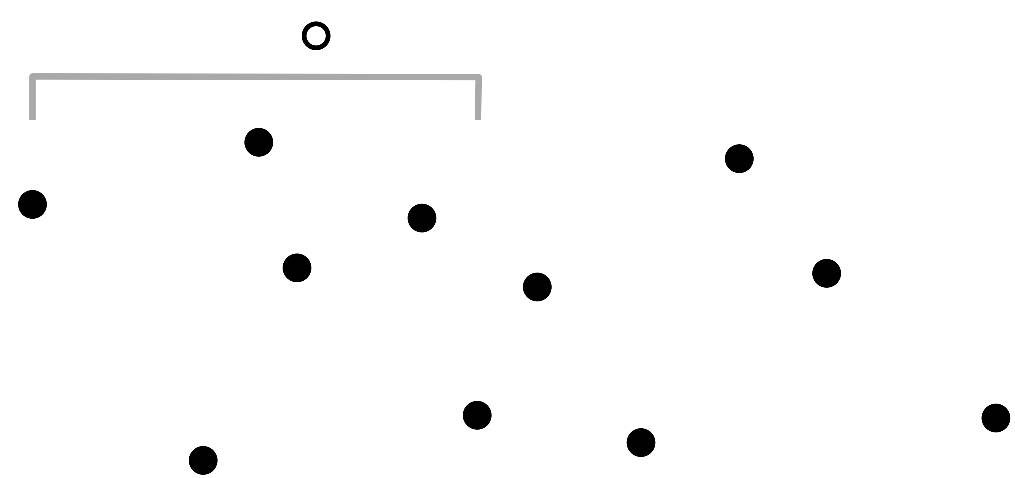
Priority queue (insert, pop max)

$+$

Binary search tree

Construction:

- root r stores point with max y-coord
  $med_x(root) :=$ median x-coordinate of $P \setminus \{r\}$
- left subtree: points $p \in P \setminus \{r\}$ with $p_x \leq med_x(r)$



Priority search tree

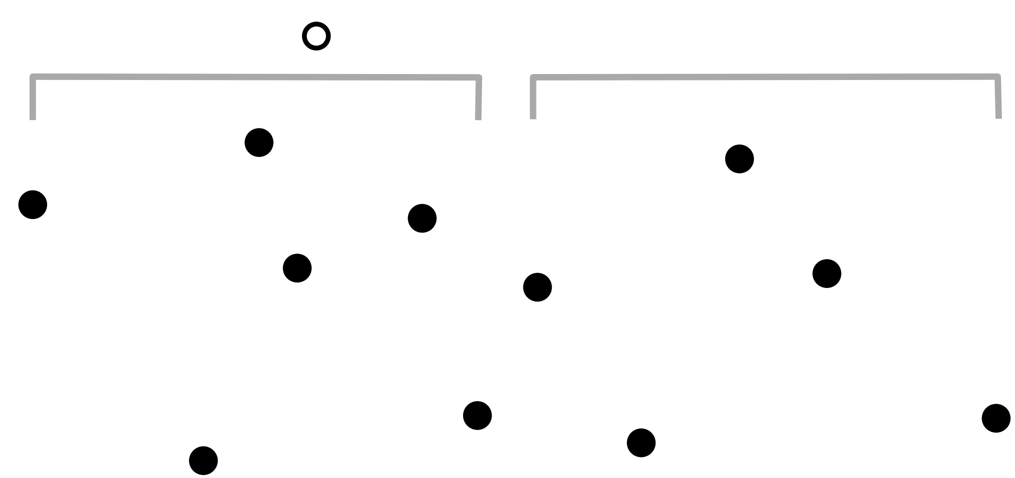
Priority queue (insert, pop max)

$+$

Binary search tree

Construction:

- root r stores point with max y-coord
  $med_x(root) :=$ median x-coordinate of $P \setminus \{r\}$
- left subtree: points $p \in P \setminus \{r\}$ with $p_x \leq med_x(r)$
- right subtree: points $p \in P \setminus \{r\}$ with $p_x > med_x(r)$



Priority search tree

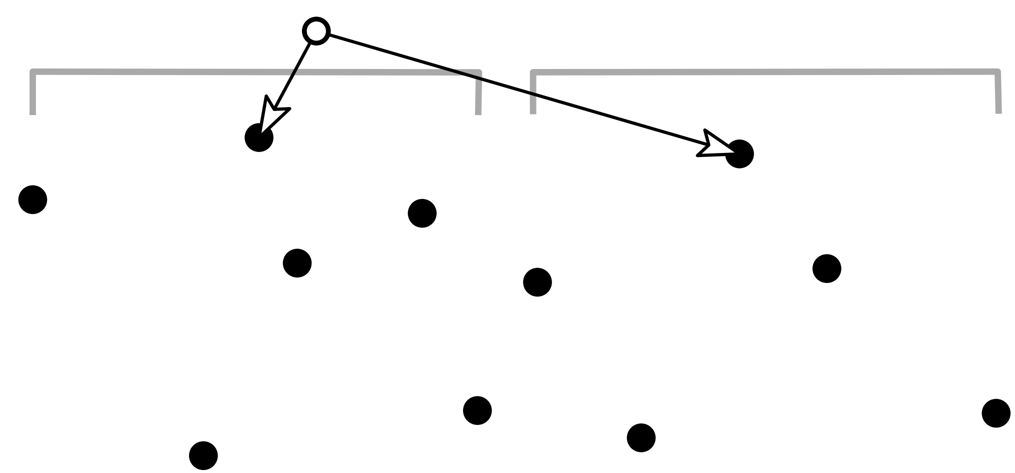
Priority queue (insert, pop max)

$+$

Binary search tree

Construction:

- root r stores point with max y-coord
  $med_x(root) :=$ median x-coordinate of $P \setminus \{r\}$
- left subtree: points $p \in P \setminus \{r\}$ with $p_x \leq med_x(r)$
- right subtree: points $p \in P \setminus \{r\}$ with $p_x > med_x(r)$



Priority search tree

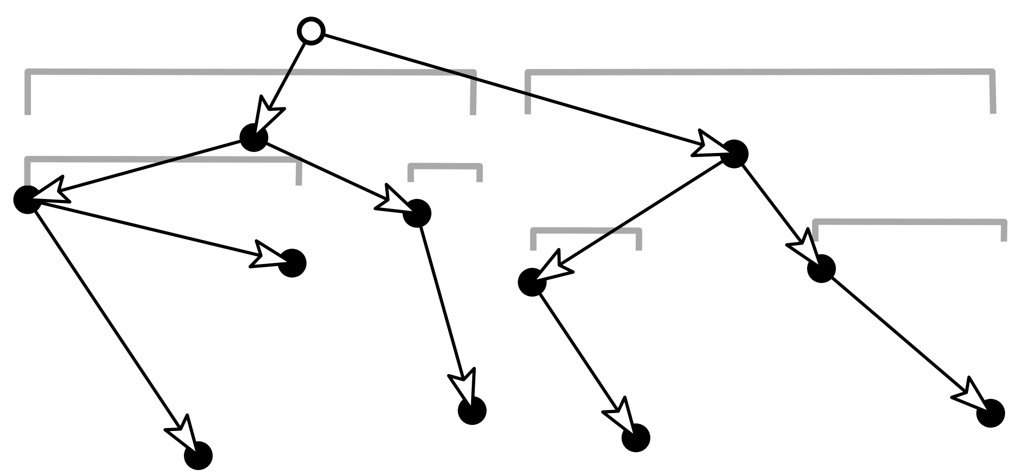
Priority queue (insert, pop max)

$+$

Binary search tree

Construction:

- root r stores point with max y-coord
  $med_x(root) :=$ median x-coordinate of $P \setminus \{r\}$
- left subtree: points $p \in P \setminus \{r\}$ with $p_x \leq med_x(r)$
- right subtree: points $p \in P \setminus \{r\}$ with $p_x > med_x(r)$



Priority search tree

Priority queue (insert, pop max)

$+$

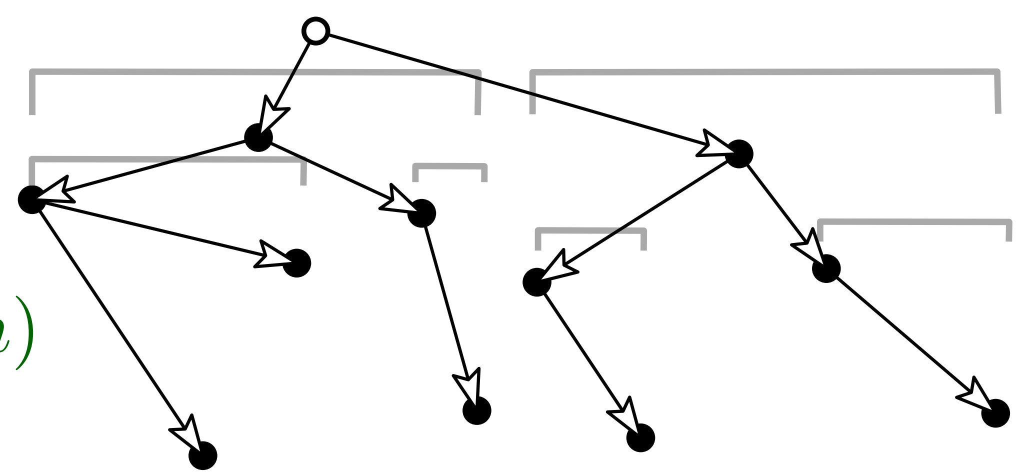
Binary search tree

Construction:

- root r stores point with max y-coord
  $med_x(root) :=$ median x-coordinate of $P \setminus \{r\}$
- left subtree: points $p \in P \setminus \{r\}$ with $p_x \leq med_x(r)$
- right subtree: points $p \in P \setminus \{r\}$ with $p_x > med_x(r)$

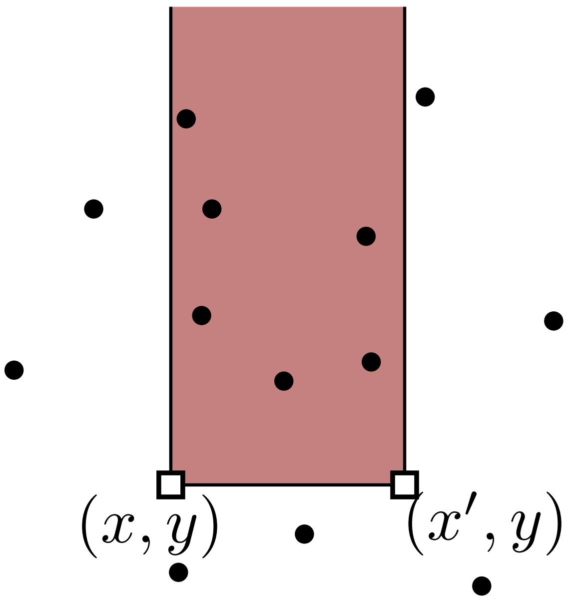
Space: $O(n)$

Construction: $O(n \log n)$



Answering 3-sided queries

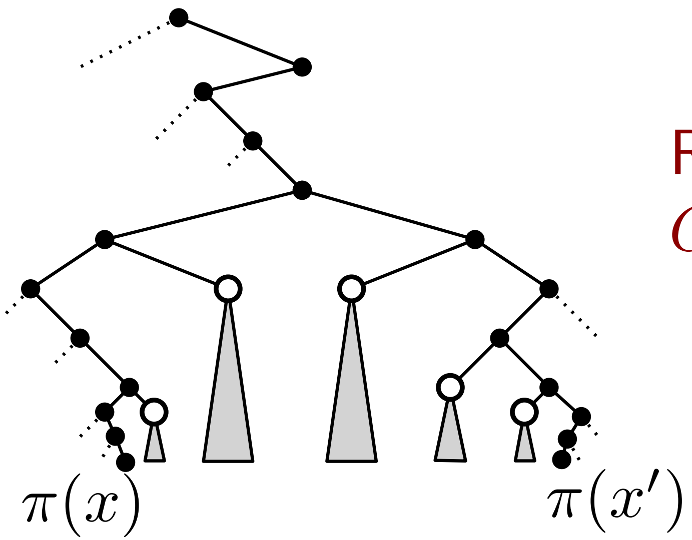
Wlog queries of the type $[x, x'] \times [y, \infty]$



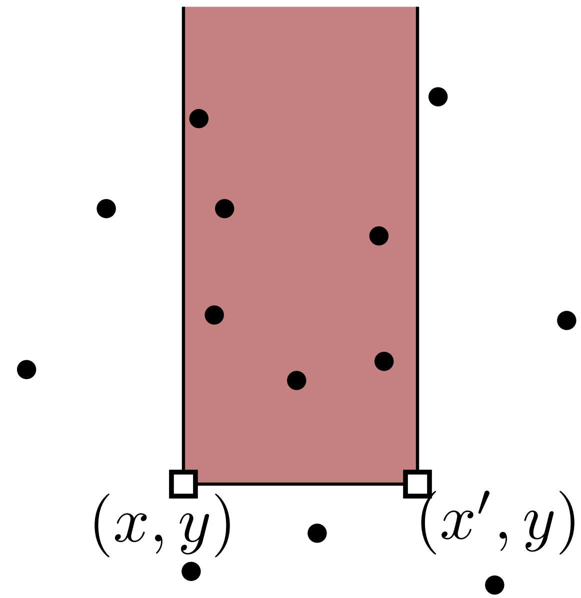
Answering 3-sided queries

Wlog queries of the type $[x, x'] \times [y, \infty]$

To answer the query from root $= v$:



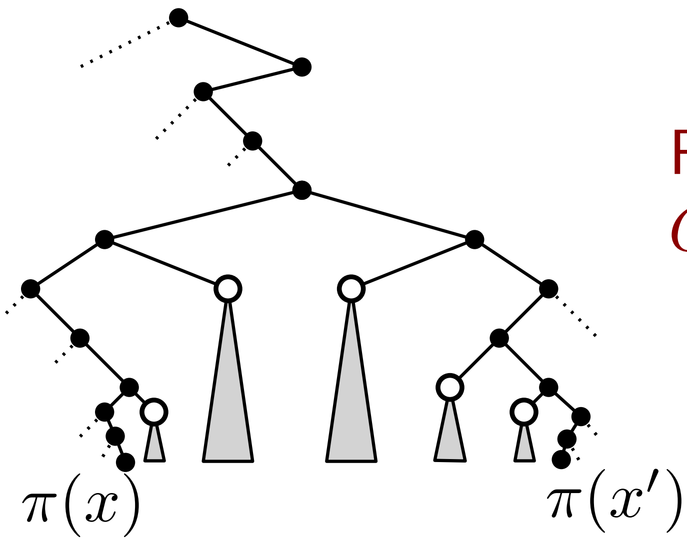
Report subtree of v:
$O(1 + k_v)$

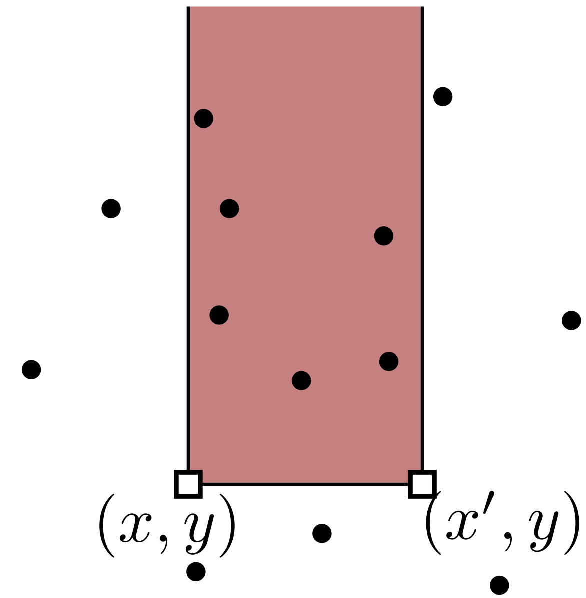


Answering 3-sided queries

Wlog queries of the type $[x, x'] \times [y, \infty]$

To answer the query from root $= v$:





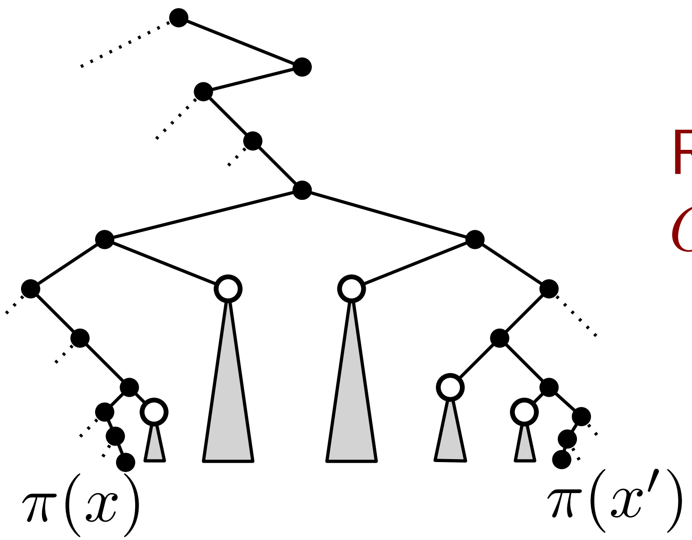
Report subtree of v:
$O(1 + k_v)$

$+$ check each node stored
on the search paths!

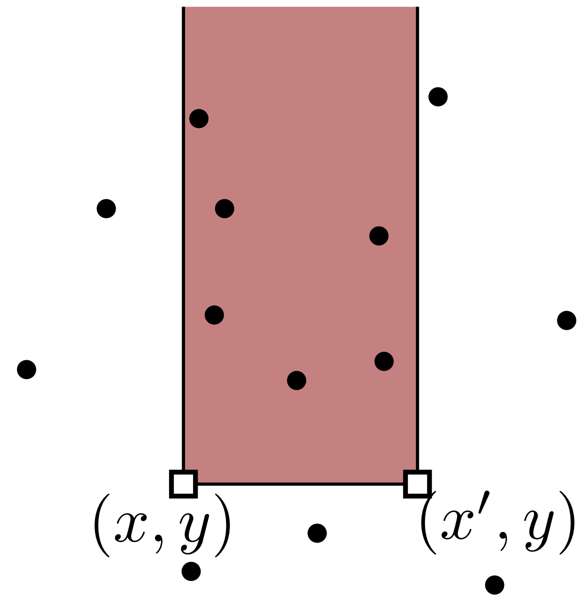
Answering 3-sided queries

Wlog queries of the type $[x, x'] \times [y, \infty]$

To answer the query from root $= v$:



Report subtree of v:
$O(1 + k_v)$



$+$ check each node stored
on the search paths!

Query time: $O(\log n + k)$

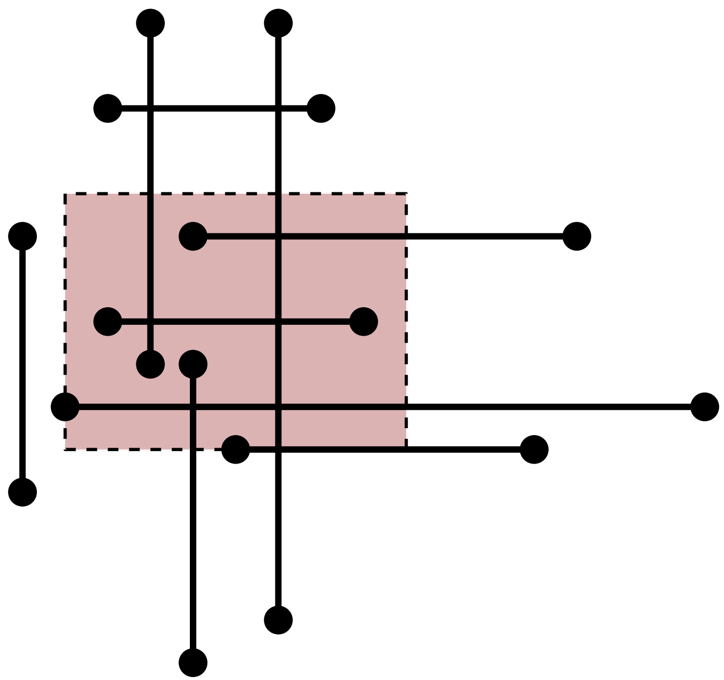
Current best data structures

Assume coordinates fit in machine words on w bits.
Multiply all by $\log w \simeq \log\log n$.

amortized

Static	Dynamic
s:$O(n)$ q:$O(\log^{\varepsilon} n + k \log^{\varepsilon} n)$ or s:$O(n \log\log n)$, q:$O(\log\log n + k \log\log n)$ or s:$O(n \log^{\varepsilon} n)$, q:$O(\log\log n + k)$	s:$O(n \log^{2/3+o(1)} n)$ q:$O(\frac{\log n}{\log\log n} + k)$ u:$O(\log^{2/3+o(1)} n)$
Chan–Larsen–Pătrașcu	Chan–Tsakalidis

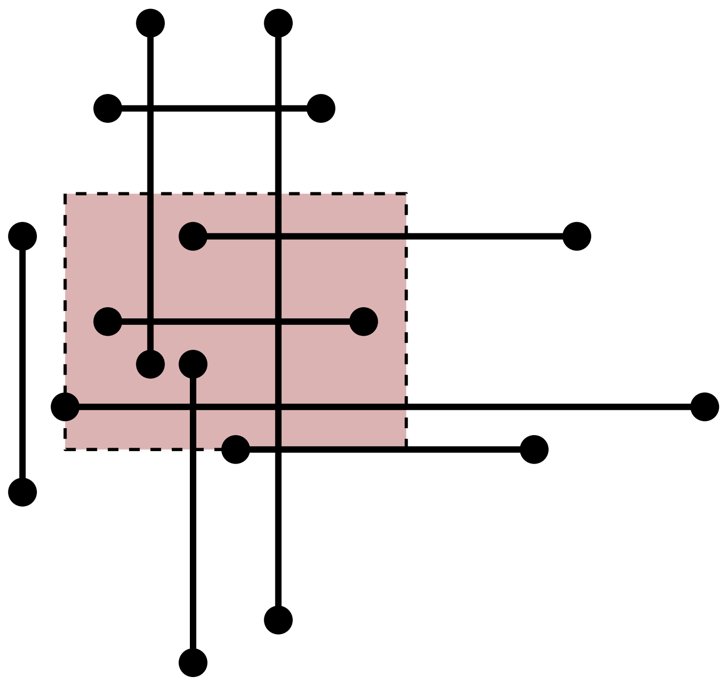
Range searching axis-parallel intervals



$\rightarrow$ intervals ending in Q:
range searching on endpoints

$\rightarrow$ intervals "crossing" Q?

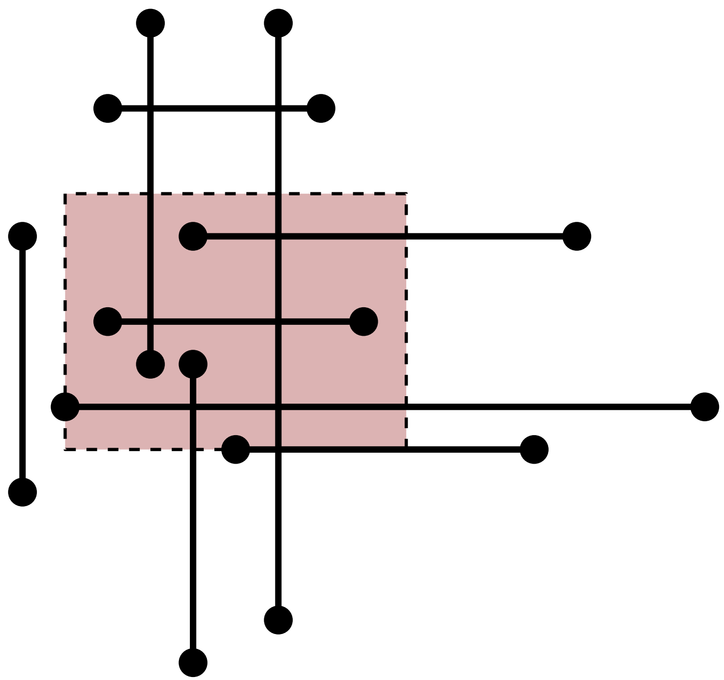
Range searching axis-parallel intervals



$\rightarrow$ intervals ending in Q:
range searching on endpoints

$\rightarrow$ intervals "crossing" Q?
if horizontal, they cross left boundary of Q

Range searching axis-parallel intervals

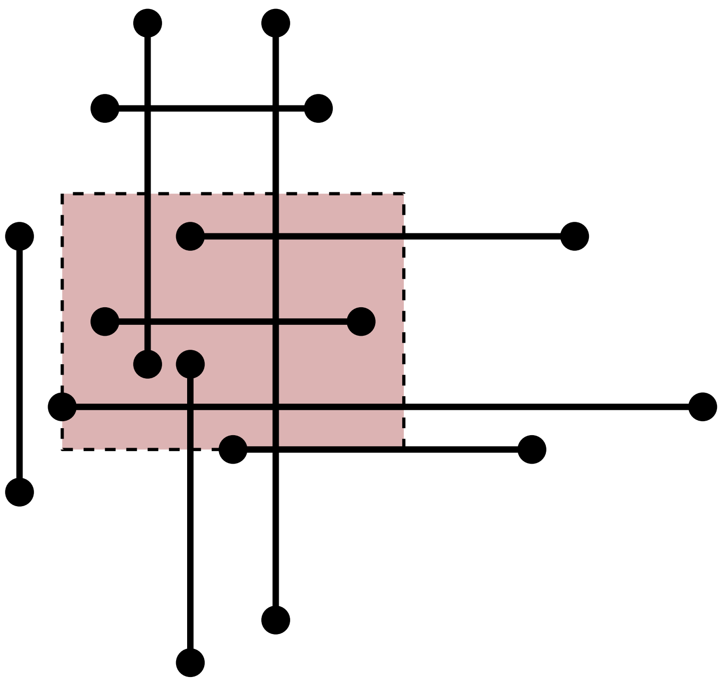


$\rightarrow$ intervals ending in Q:
range searching on endpoints

$\rightarrow$ intervals "crossing" Q?
if horizontal, they cross left boundary of Q

Given a vertical query segment, report all horizontal segments it is crossing.

Range searching axis-parallel intervals

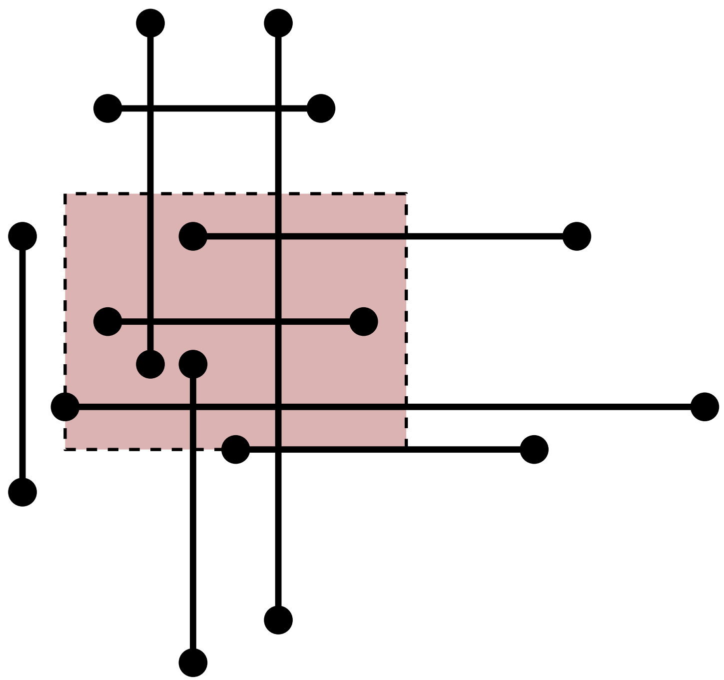


→ intervals ending in Q:
range searching on endpoints

→ intervals "crossing" Q?
if horizontal, they cross left boundary of Q

Given a vertical query ~~segment~~ line, report all horizontal segments it is crossing.

Range searching axis-parallel intervals



$\rightarrow$ intervals ending in Q:
range searching on endpoints

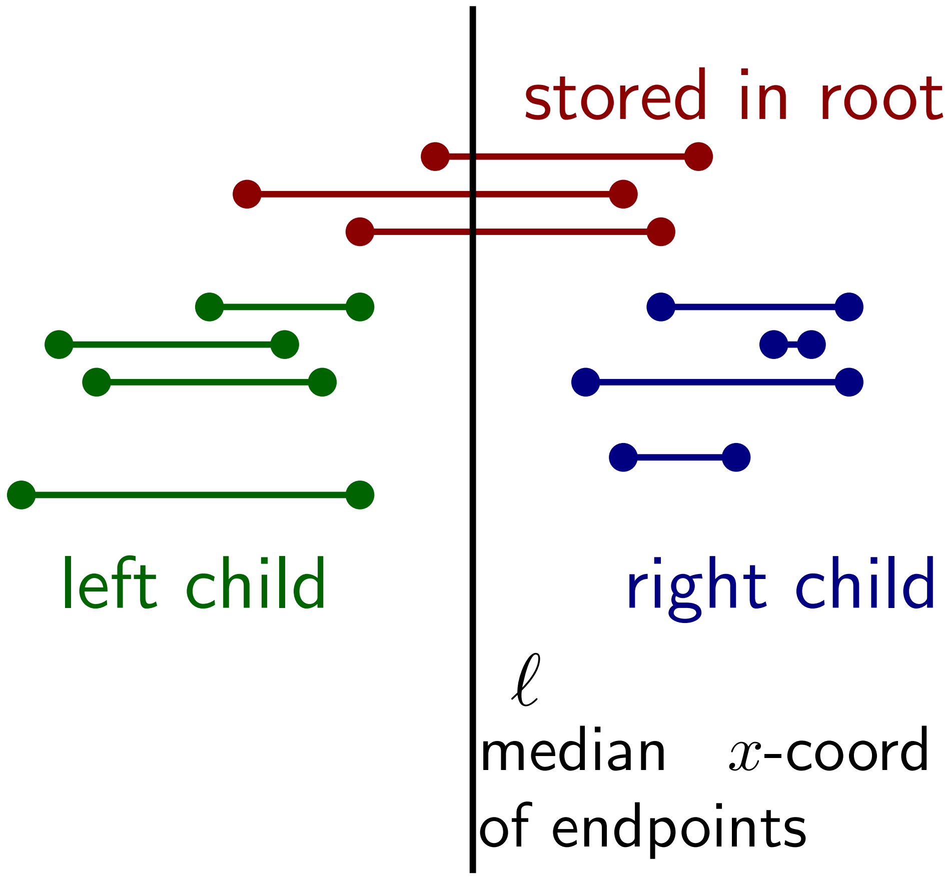
$\rightarrow$ intervals "crossing" Q?
if horizontal, they cross left boundary of Q

Given a vertical query ~~segment~~ line, report all horizontal segments it is crossing.

$\Leftrightarrow$ in $\mathbb{R}^1$, which intervals contain query point q?

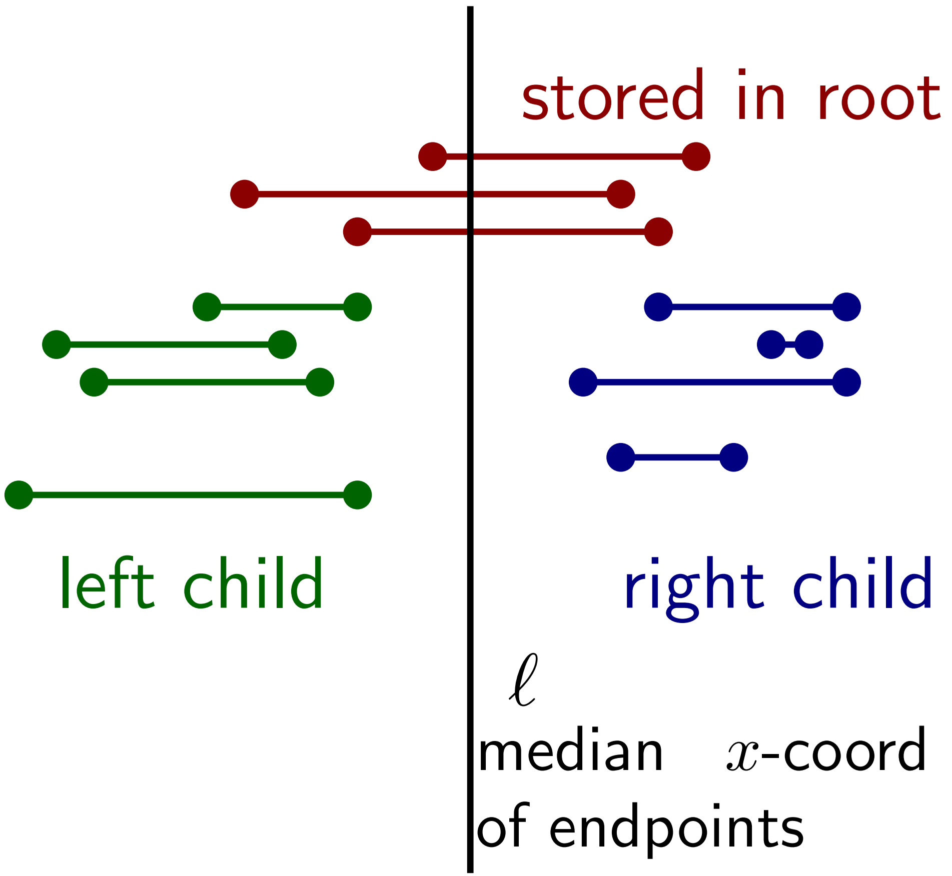
Interval trees (Edelsbrunner 1980 McCreight 1980)

root has intervals intersected by ℓ sorted for left endpoints in a list (same for right)



Interval trees (Edelsbrunner 1980 McCreight 1980)

root has intervals intersected by ℓ
sorted for left endpoints in a list
(same for right)



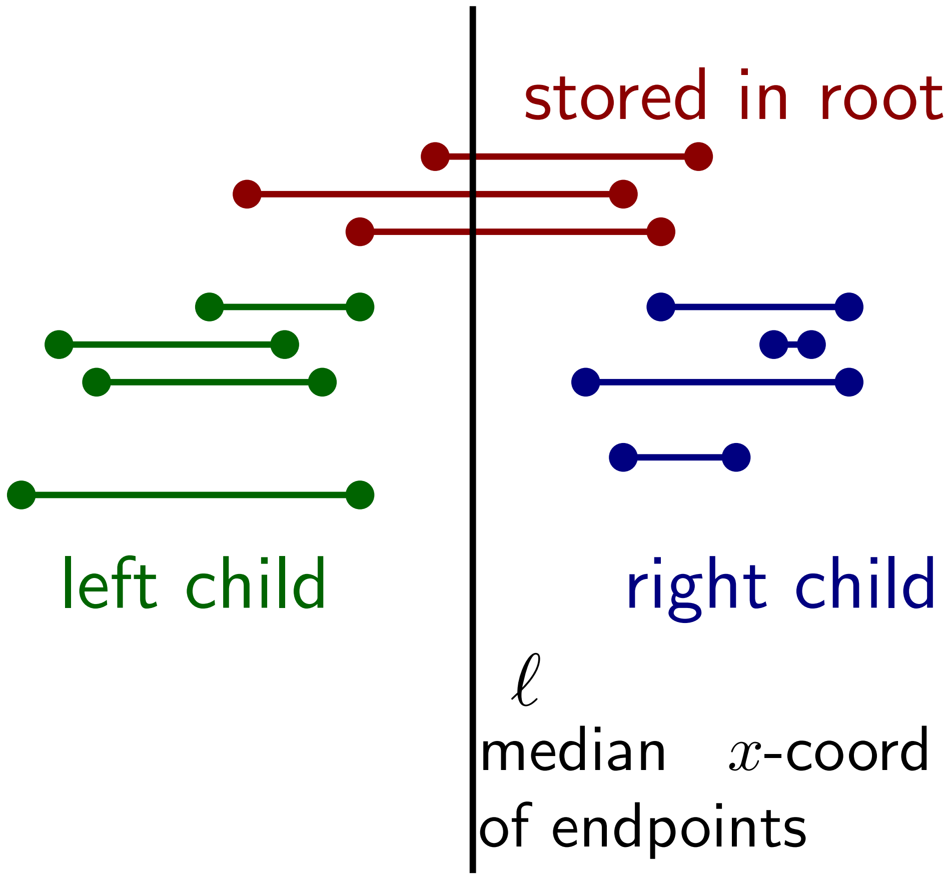
Preprocess: $O(n \log n)$

Space: $O(n)$

Query ($\mathbb{R}^1$): $O(\log n + k)$

Interval trees (Edelsbrunner 1980 McCreight 1980)

root has intervals intersected by ℓ
sorted for left endpoints in a list
(same for right)



Preprocess: $O(n \log n)$

Space: $O(n)$

Query ($\mathbb{R}^1$): $O(\log n + k)$

If segment query:
use priority search tree instead of list