Lecture Notes for *Introduction to Computational Geometry* (Com S 418/518)
Yan-Bin Jia, Iowa State University

# Range Trees

## Outline:

I. Construction of a 2D range Tree

II. Query with a 2D range Tree

III. High-dimensional range trees

Textbook: *Computational Geometry: Algorithms and Applications (*3rd ed.) by M. de Berg et al., Springer-Verlag, 2008.

# Improvement on a Kd-Tree

| | Query time | Storage |
|---|---|---|
| relatively high! | $O(\sqrt{n} + k)$ | $O(n)$ |

# Improvement on a Kd-Tree

| | Query time | Storage |
| --- | --- | --- |
| relatively high! | $O(\sqrt{n} + k)$ | $O(n)$ |
| | $\Downarrow$ | |
| | $O(\log^2 n + k)$ | |

# Improvement on a Kd-Tree

| | Query time | Storage |
|---|---|---|
| relatively high! | $O(\sqrt{n} + k)$ | $O(n)$ |
| | $\Downarrow$ | |
| | $O(\log^2 n + k)$ | $O(n \log n)$ |

# Improvement on a Kd-Tree

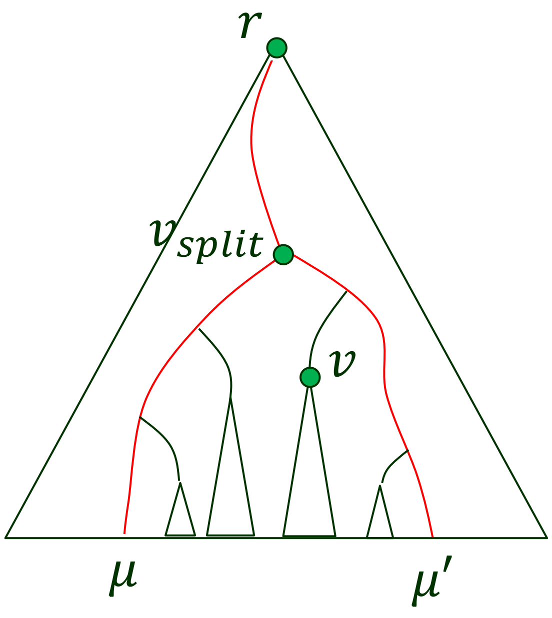
| | Query time | Storage |
|---|---|---|
| relatively high! | $O(\sqrt{n} + k)$ | $O(n)$ |
| | $\Downarrow$ | |
| | $O(\log^2 n + k)$ | $O(n \log n)$ |

How?

# I. 2D Query

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

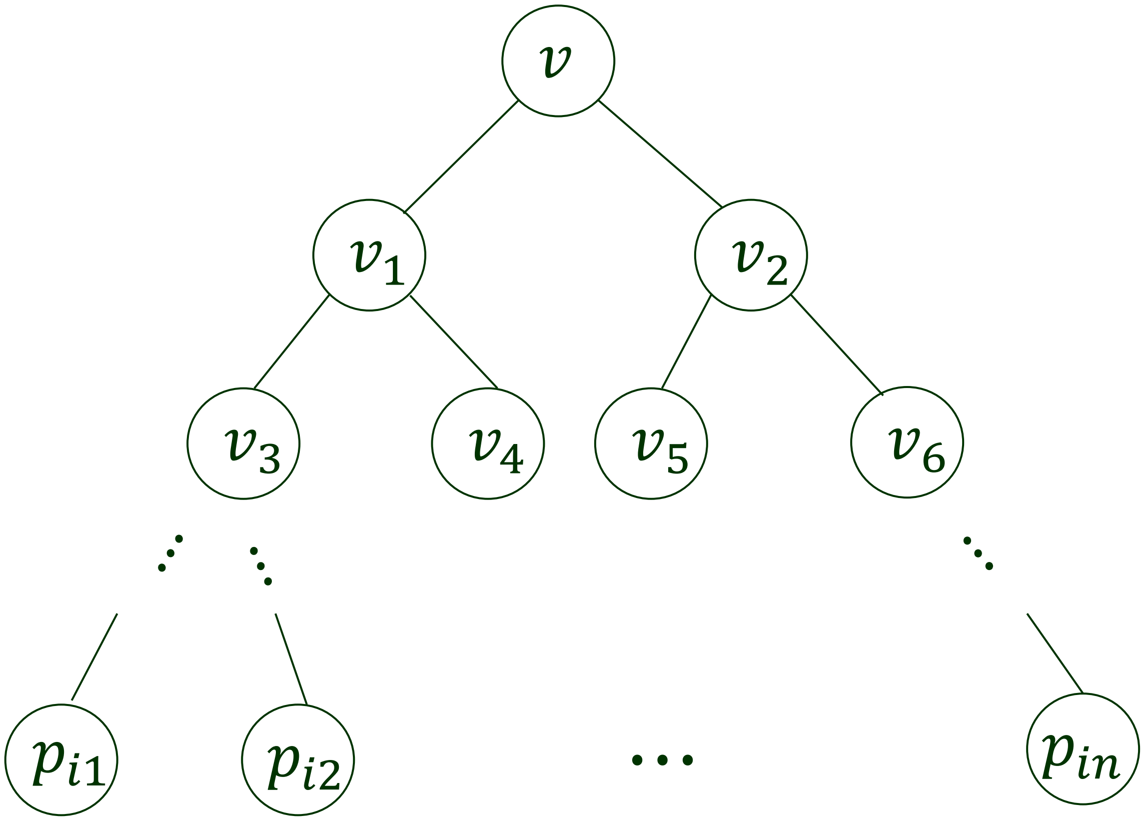
1. Find points with $x$-coordinate $\in [x, x']$




- Searches with $x$ and $x'$ in BST end at leaves $\mu$ and $\mu'$.
- Select a collection of subtrees which together contain exactly the points with $x$-coordinates $\in [x, x']$.

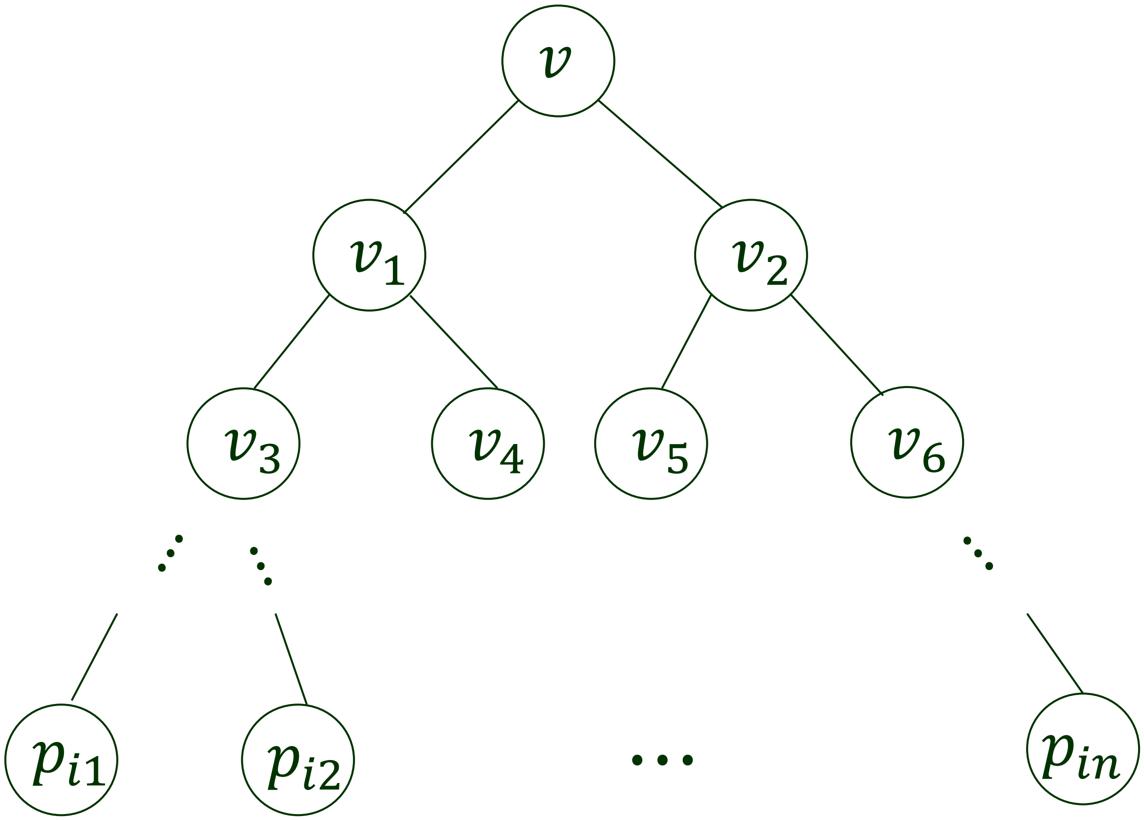
# Canonical Subsets

$P(v)$: leaves (points) of the subtree rooted at $v$



# Canonical Subsets

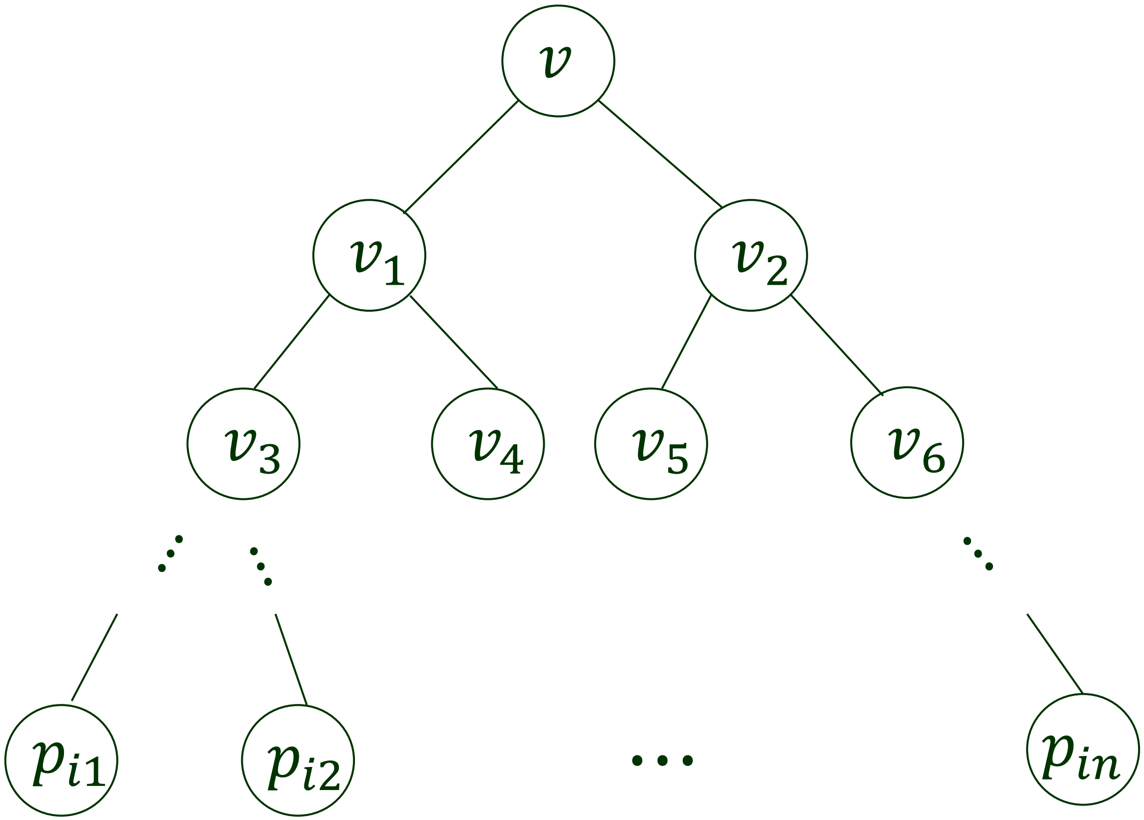
$P(v)$: leaves (points) of the subtree rooted at $v$




$P(v) = P = \{p_{i1}, \ldots, p_{in}\}$

# Canonical Subsets

$P(v)$: leaves (points) of the subtree rooted at $v$



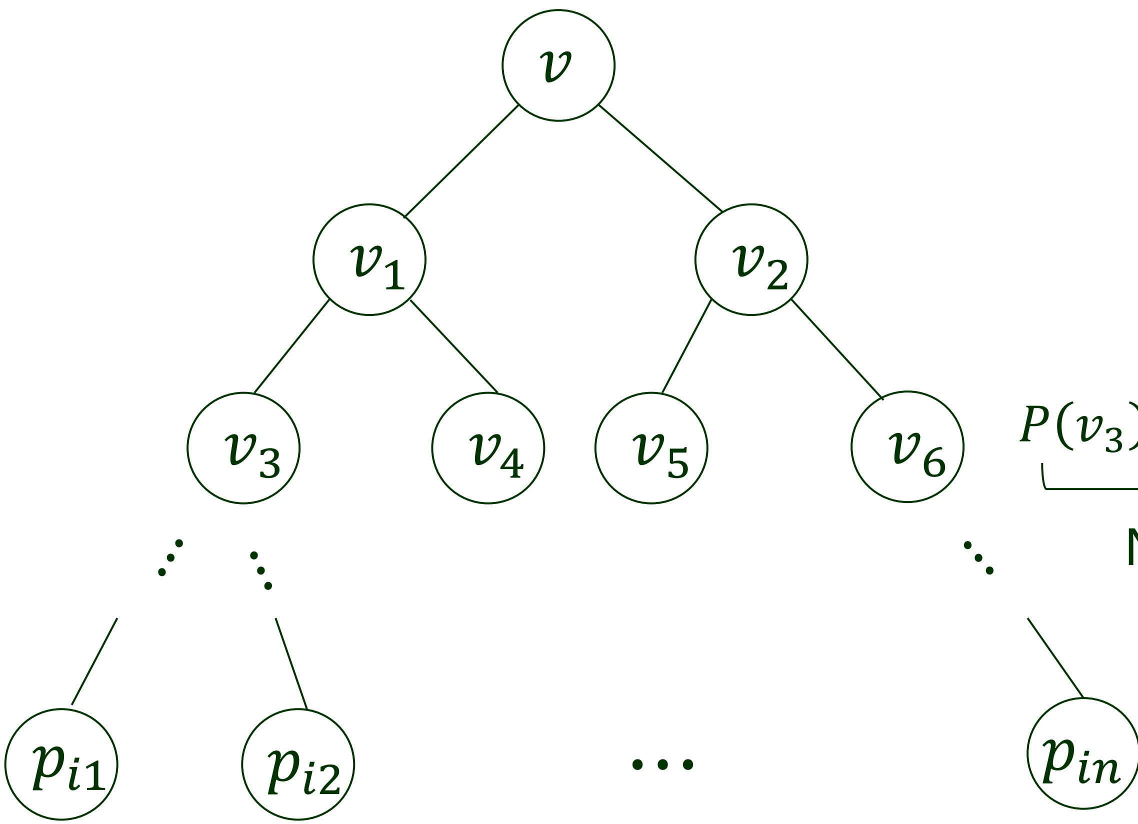

$P(v) = P = \{p_{i1}, \dots, p_{in}\}$

$P(v_1) \cup P(v_2) = P$

$P(v_1) \cap P(v_2) = \emptyset$

# Canonical Subsets

$P(v)$: leaves (points) of the subtree rooted at $v$



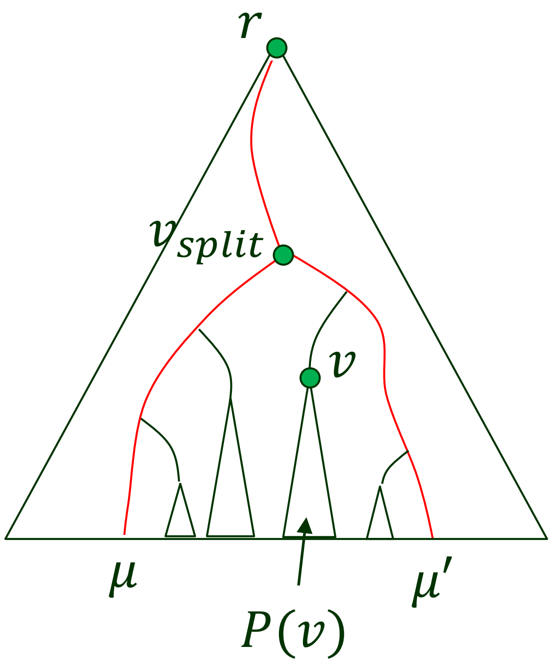

$$P(v) = P = \{p_{i1}, \dots, p_{in}\}$$

$$P(v_1) \cup P(v_2) = P$$
$$P(v_1) \cap P(v_2) = \emptyset$$

$$\underbrace{P(v_3) \cup P(v_4) \cup P(v_5) \cup P(v_6)}_{\text{Non-intersecting subsets}} = P$$

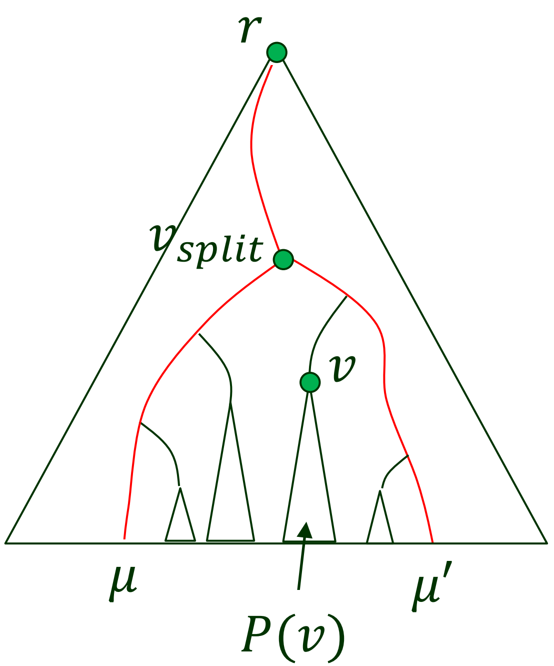
# Back to 2D Query

1. Find points with $x$-coordinate $\in [x, x']$



# Back to 2D Query

1. Find points with $x$-coordinate $\in [x, x']$




- Query $[x, x']$ yields $O(\log n)$ *disjoint* subsets:

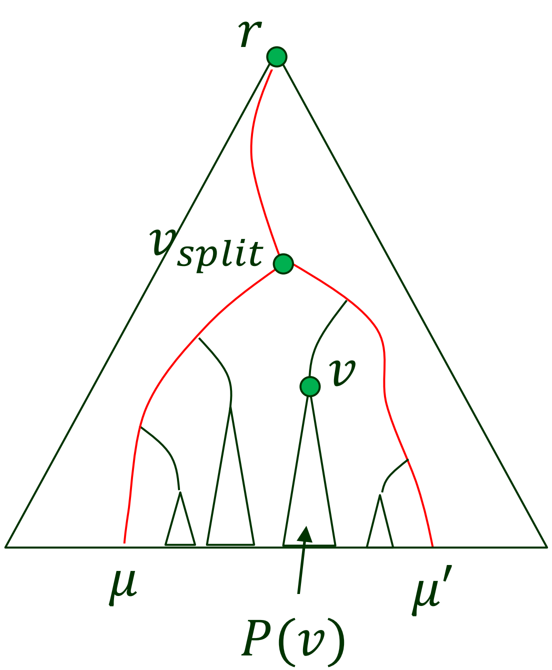
$$\bigcup_{v} P(v)$$

where $v$ is

- right child of a node on the path $r \rightsquigarrow \mu$ when it makes a left turn, or
- left child of a node on the path $r \rightsquigarrow \mu'$ when it makes a right turn.

# Back to 2D Query

1. Find points with $x$-coordinate $\in [x, x']$




♦ Query $[x, x']$ yields $O(\log n)$ *disjoint* subsets:

$$\bigcup_{v} P(v)$$

where $v$ is

- right child of a node on the path $r \rightsquigarrow \mu$ when it makes a left turn, or
- left child of a node on the path $r \rightsquigarrow \mu'$ when it makes a right turn.

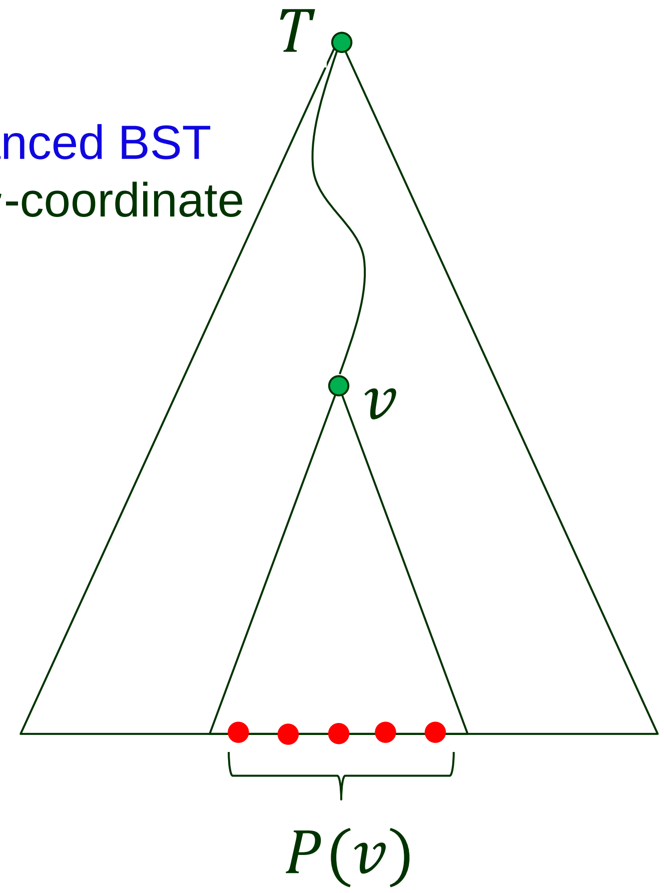
2. For each such $v$, query $[y, y']$ on $T(v)$.

How to make this query fast?

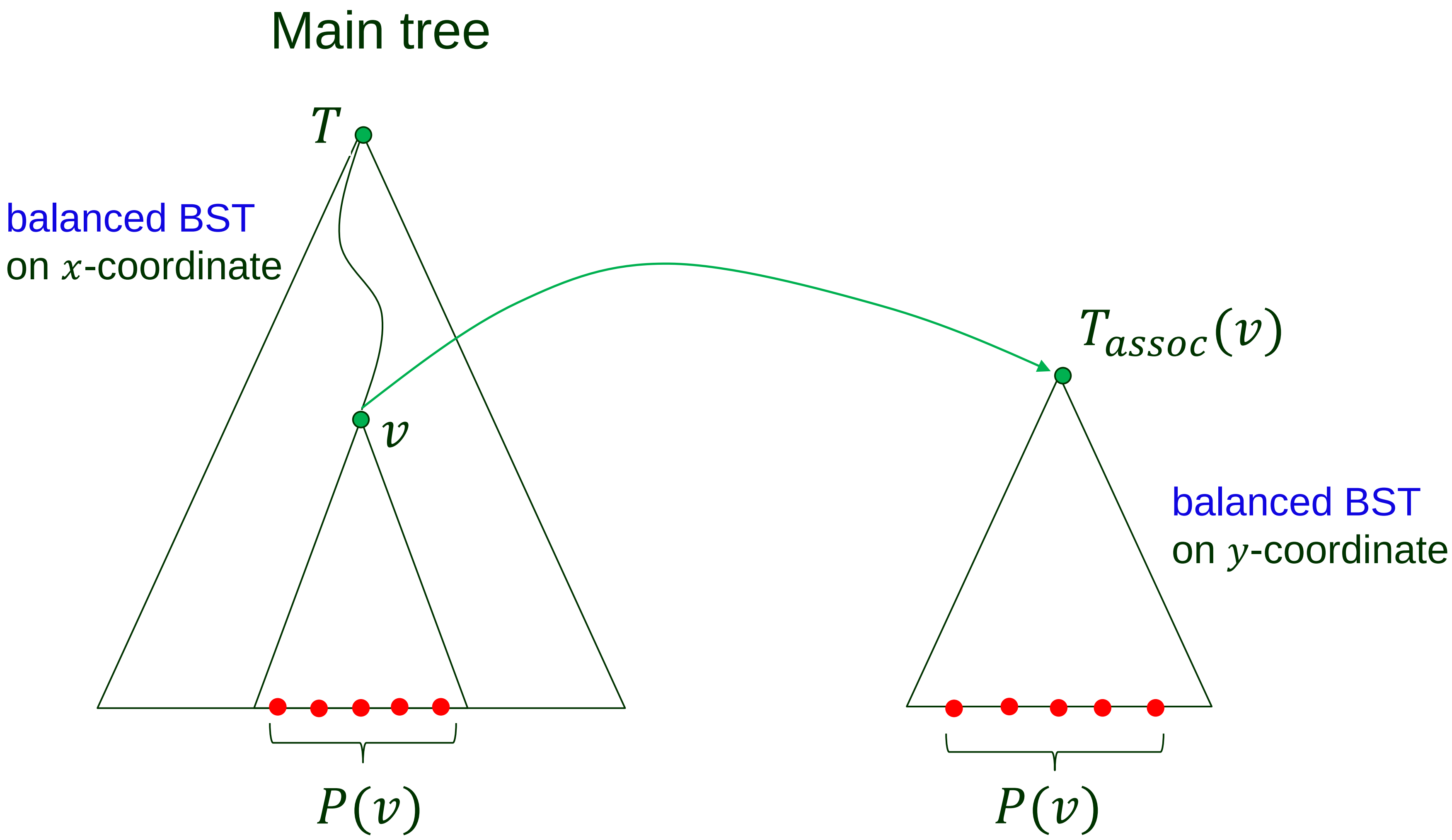
# 2D Range Tree

Main tree

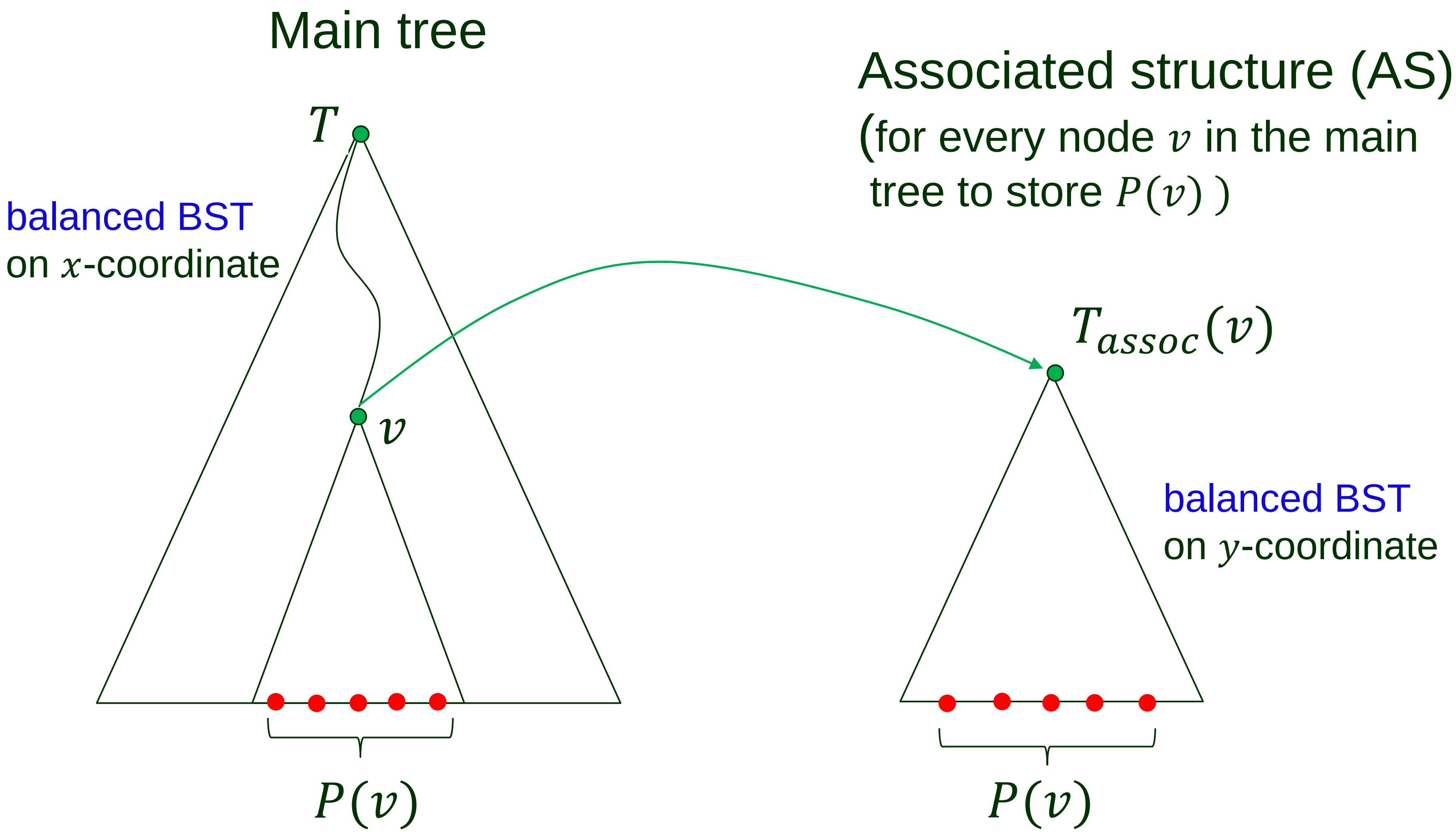
balanced BST
on $x$-coordinate



# 2D Range Tree



# 2D Range Tree



# Recursive Construction

$P = \{p_1, \cdots, p_n\}$ sorted on $x$-coordinate

Build2DRangeTree($P$)

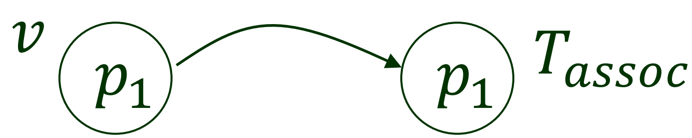
1. build $T_{assoc}$ on $P_y = \{y_i \mid (x_i, y_i) \in P\}$ // leaves storing points
2. if $n = 1$
3. then
4. else
5.
6.

# Recursive Construction

$P = \{p_1, \cdots, p_n\}$ sorted on $x$-coordinate

Build2DRangeTree($P$)

1. build $T_{assoc}$ on $P_y = \{y_i \mid (x_i, y_i) \in P\}$ // leaves storing points
2. if $n = 1$
3. then




4. else

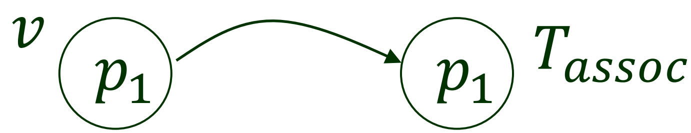
5.
6.

# Recursive Construction

$P = \{p_1, \cdots, p_n\}$ sorted on $x$-coordinate

Build2DRangeTree($P$)

1. build $T_{assoc}$ on $P_y = \{y_i \mid (x_i, y_i) \in P\}$ // leaves storing points
2. if $n = 1$
3. then




4. else split $P$ at the medium $x$-coordinate ($x_{mid}$)
   into $P_{left}$ and $P_{right}$

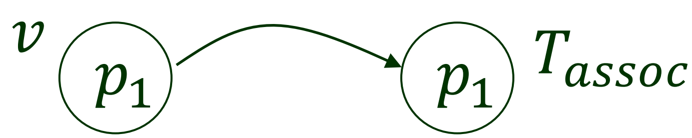
5.
6.

# Recursive Construction

$P = \{p_1, \cdots, p_n\}$ sorted on $x$-coordinate

Build2DRangeTree($P$)

1. build $T_{assoc}$ on $P_y = \{y_i \mid (x_i, y_i) \in P\}$ // leaves storing points
2. if $n = 1$
3. then




4. else split $P$ at the medium $x$-coordinate ($x_{mid}$)
   into $P_{left}$ and $P_{right}$

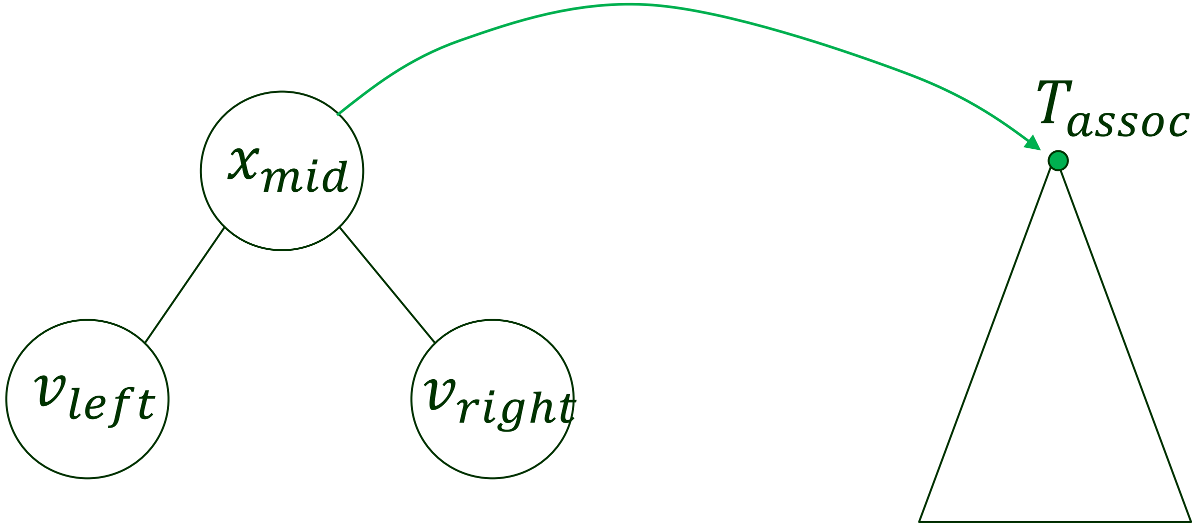
5. $v_{left} \leftarrow$ Build2DRangeTree($P_{left}$)
6. $v_{right} \leftarrow$ Build2DRangeTree($P_{right}$)

# Combining Range Trees

⋮

5. $v_{left} \leftarrow$ Build2DRangeTree($P_{left}$)
6. $v_{right} \leftarrow$ Build2DRangeTree($P_{right}$)

7.




8. return $v$

# Storage Analysis

Total size of the main tree &
all associated structures.

# Storage Analysis

Total size of the main tree &
all associated structures.

Every tree is a balanced BST.

# Storage Analysis

Total size of the main tree &
all associated structures.

⇓ Every tree is a balanced BST.

\# *leaves* in the main tree
& all associated structures.

# Storage Analysis

Total size of the main tree &
all associated structures.

Every tree is a balanced BST.

\# *leaves* in the main tree
& all associated structures.

Every leave stores a point.

# Storage Analysis

Total size of the main tree &
all associated structures.

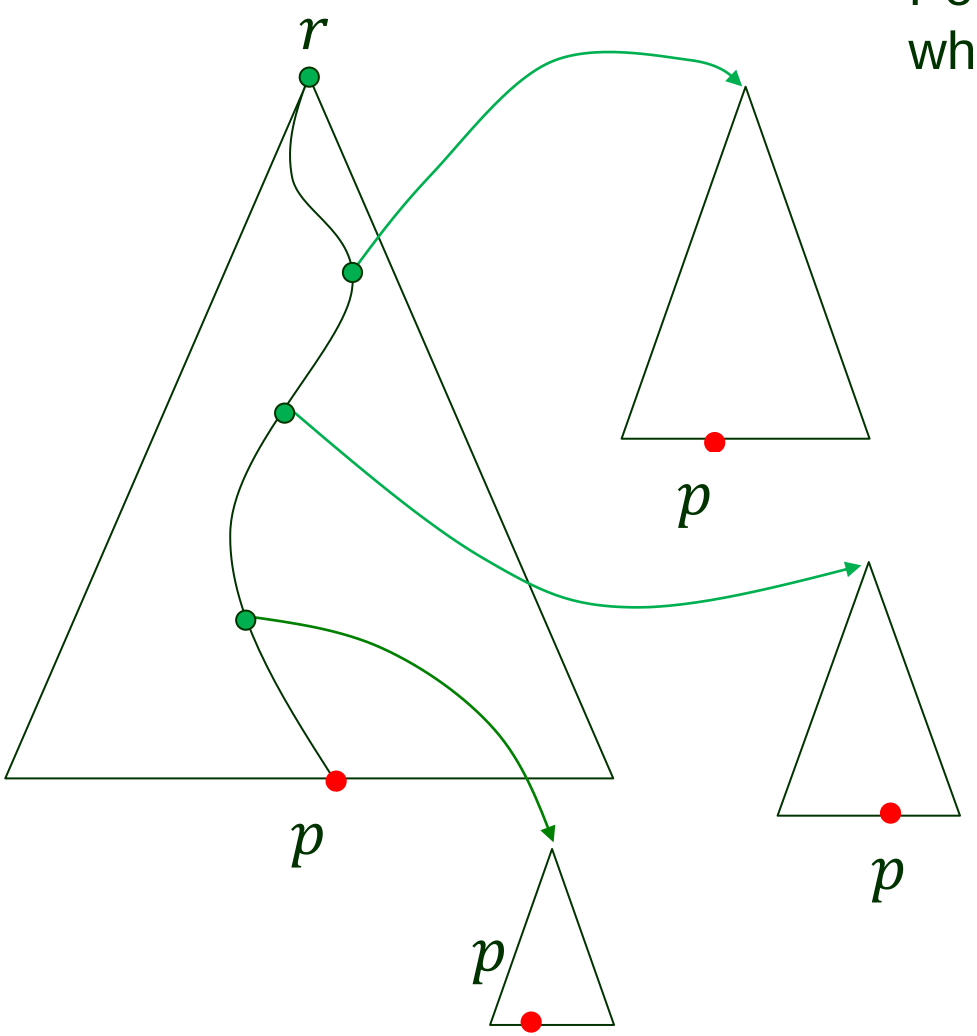
⇓ Every tree is a balanced BST.

\# *leaves* in the main tree
& all associated structures.

⇓ Every leave stores a point.

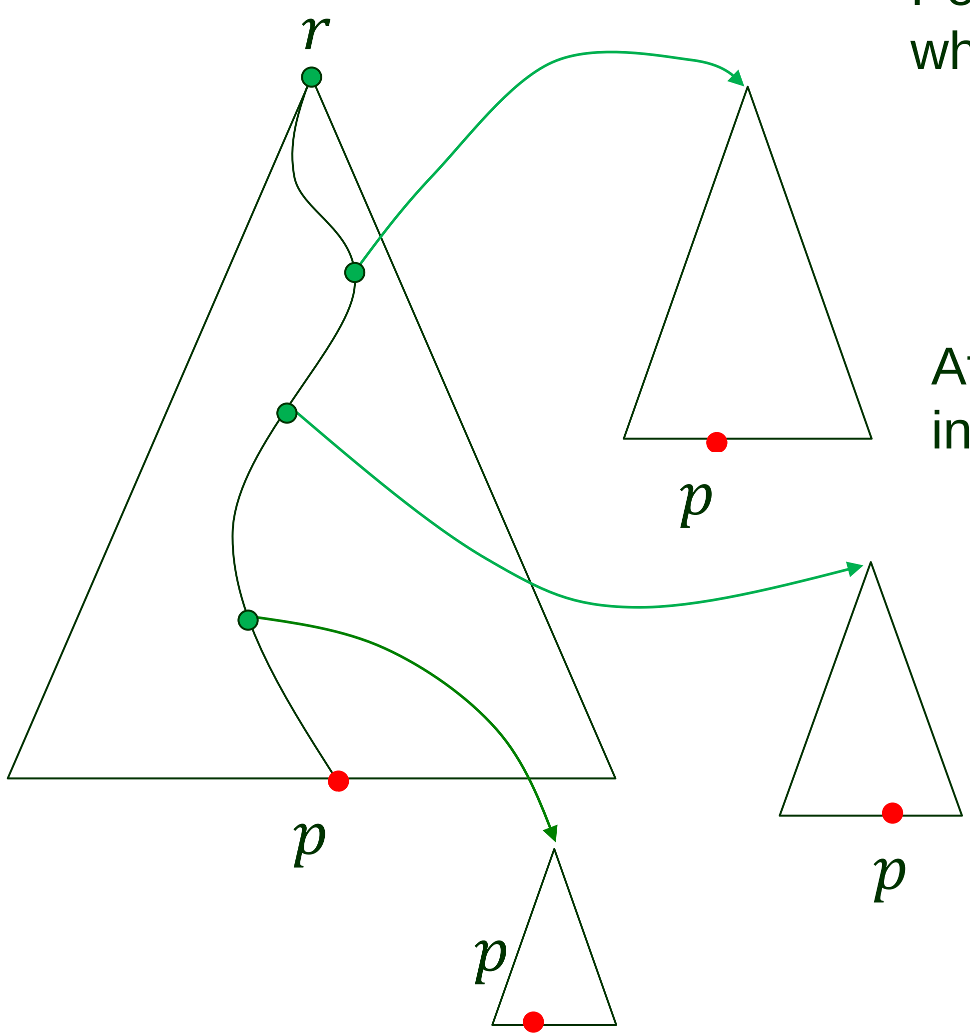
\# *times* the points are stored.

# How Often is a Point Stored?




Point $p$ is stored only in $T_{assoc}(v)$ where $v$ is on the path $r \rightsquigarrow p$.

# How Often is a Point Stored?



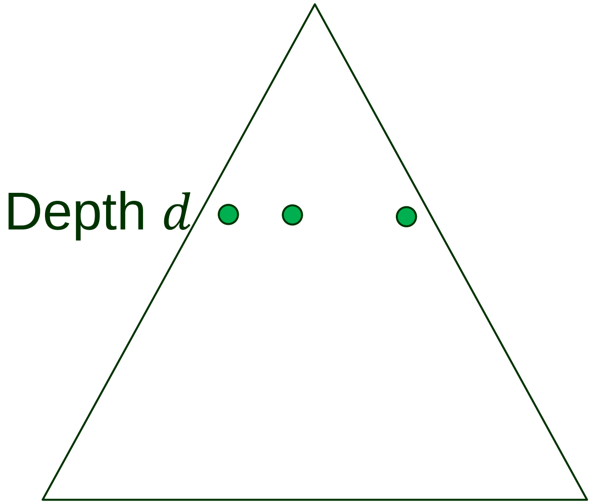

Point $p$ is stored only in $T_{assoc}(v)$ where $v$ is on the path $r \rightsquigarrow p$.

$\Downarrow$

At every depth $d$, $p$ is stored in exactly one AS.

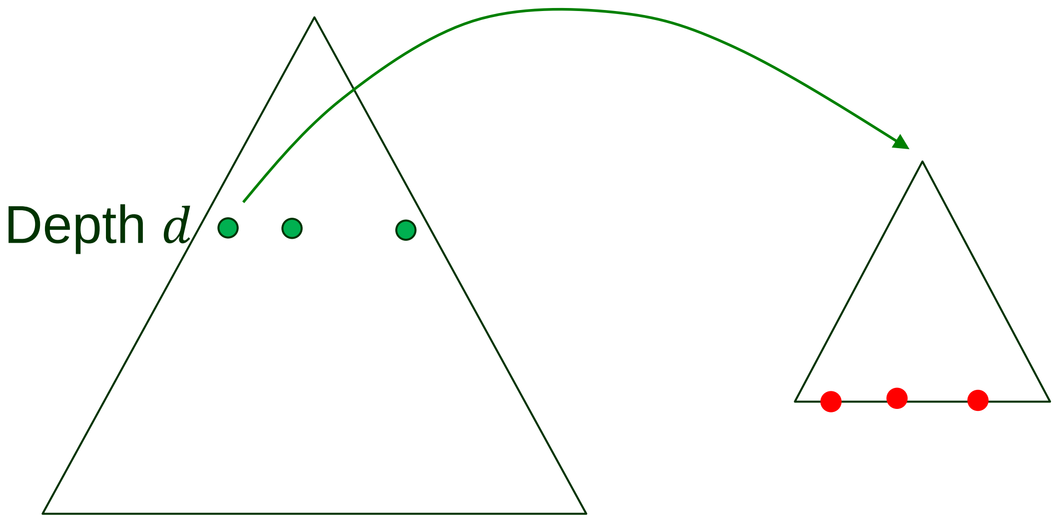
# Depth $d$

At depth $d$ every point is stored at a leaf of *exactly one* AS.



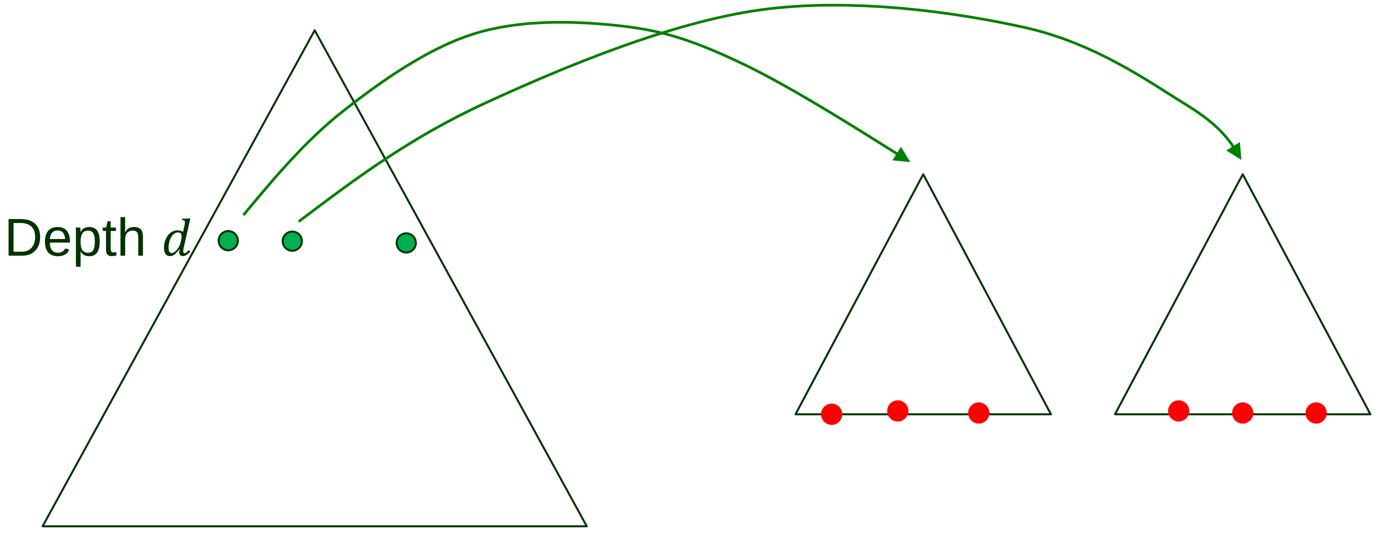
# Depth $d$

At depth $d$ every point is stored at a leaf of *exactly one* AS.



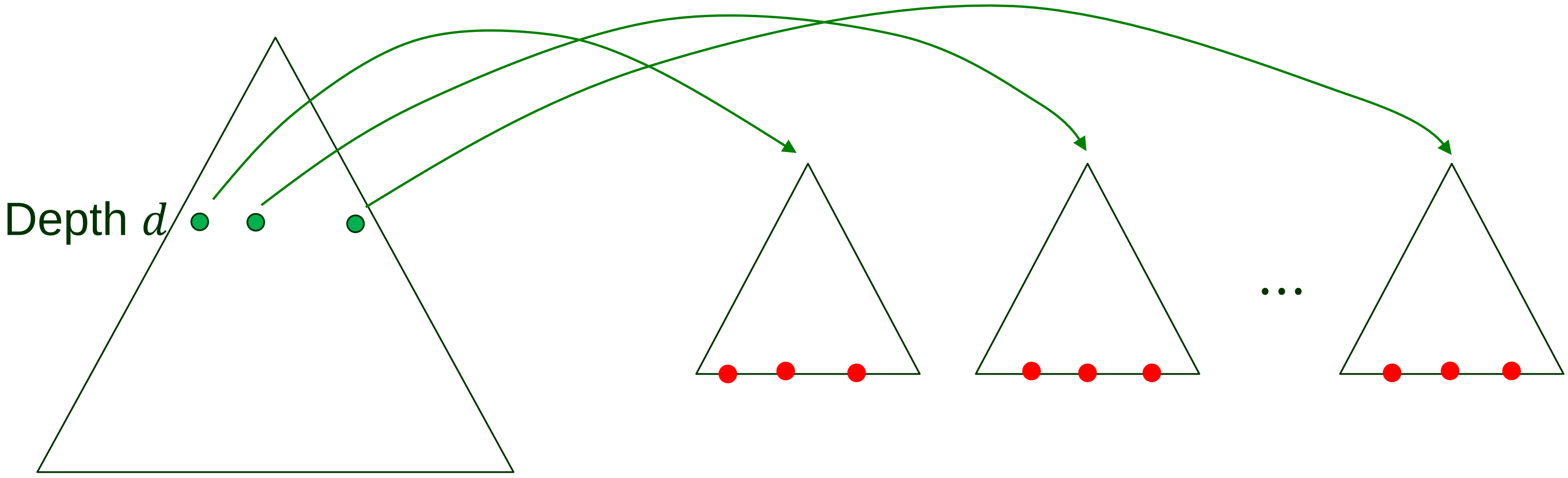
# Depth $d$

At depth $d$ every point is stored at a leaf of *exactly one* AS.



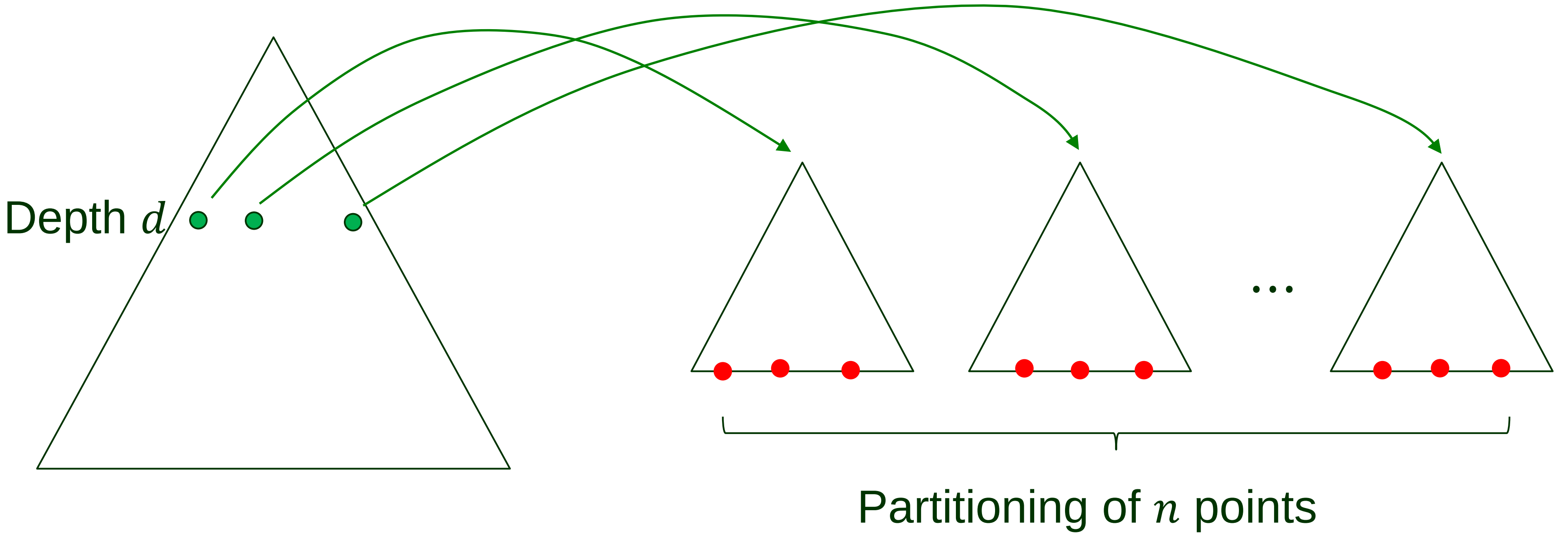
# Depth $d$

At depth $d$ every point is stored at a leaf of *exactly one* AS.



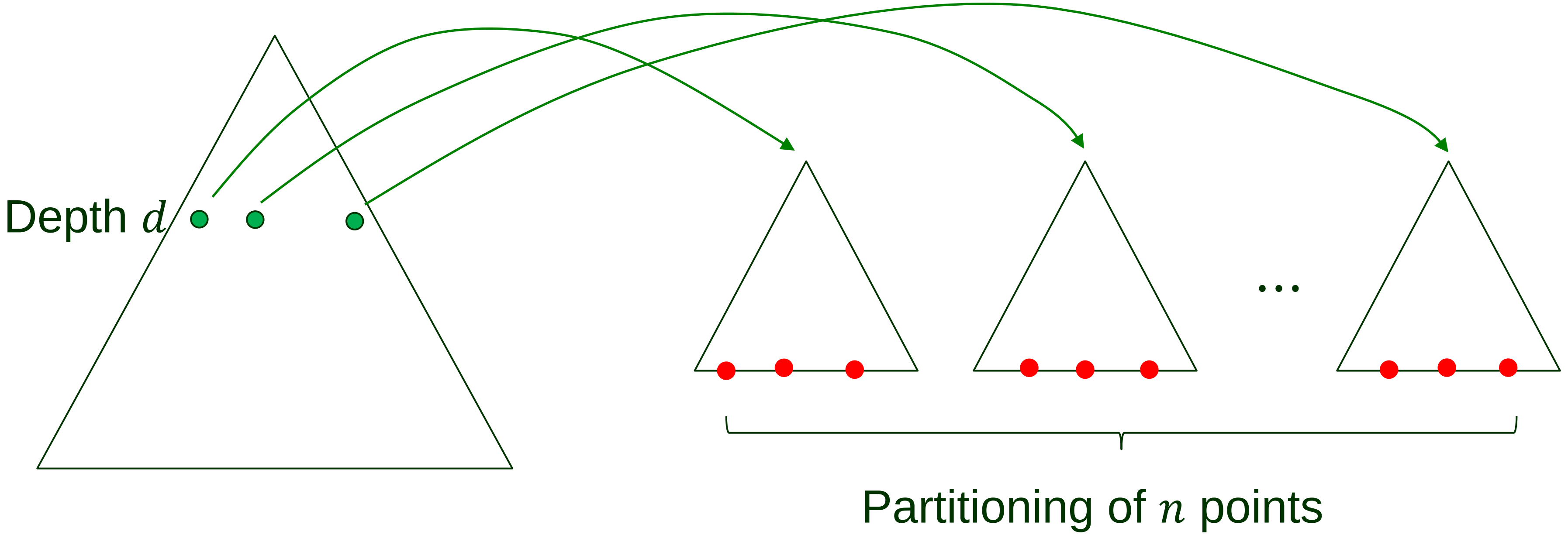
# Depth $d$

At depth $d$ every point is stored at a leaf of *exactly one* AS.



# Depth $d$

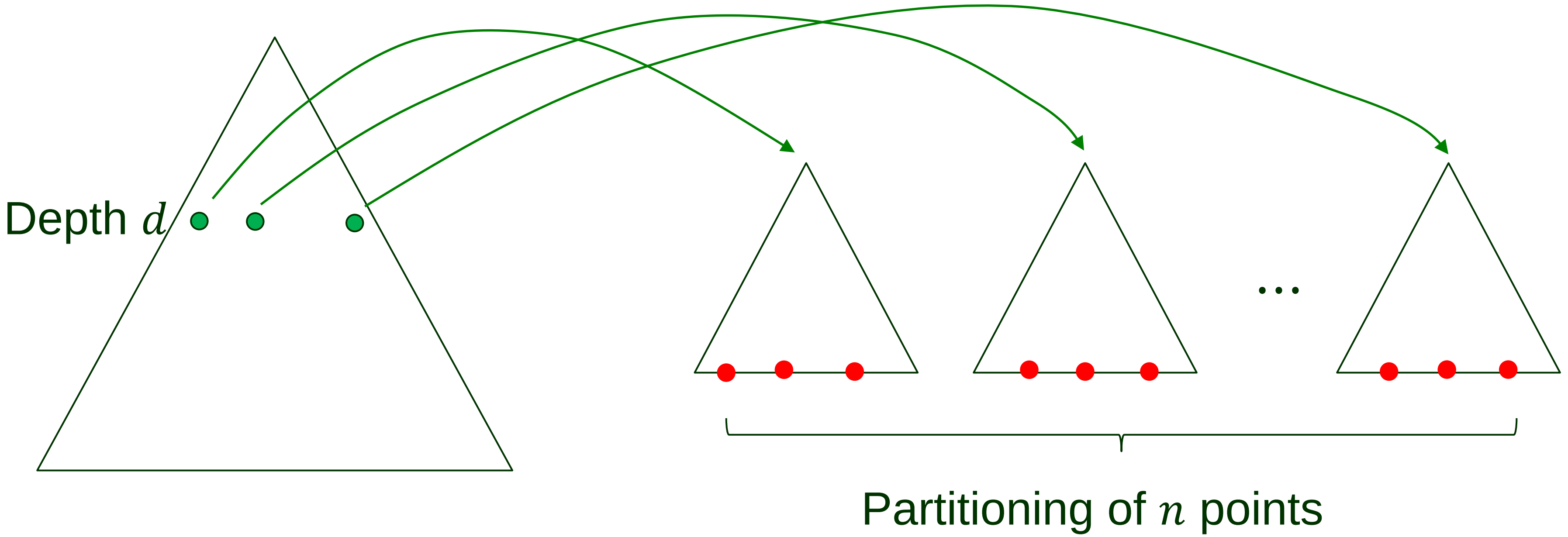
At depth $d$ every point is stored at a leaf of *exactly one* AS.




- The ASes at depth $d$ have exactly $n$ leaves (points) – no duplicates.

# Depth $d$

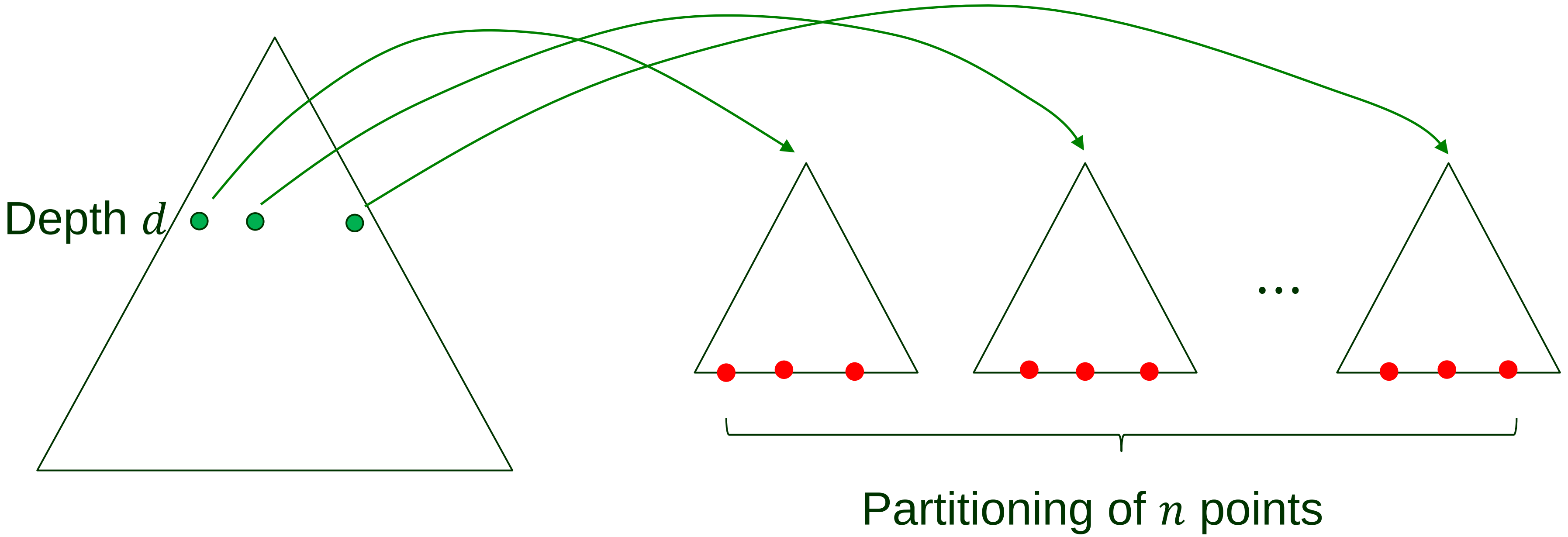
At depth $d$ every point is stored at a leaf of *exactly one* AS.




- The ASes at depth $d$ have exactly $n$ leaves (points) – no duplicates.
- Each AS as a balanced BST has size $O$(#leaves).

# Depth $d$

At depth $d$ every point is stored at a leaf of *exactly one* AS.




- The ASes at depth $d$ have exactly $n$ leaves (points) – no duplicates.
- Each AS as a balanced BST has size $O(\#\text{leaves})$.

$\Downarrow$

All the ASes at depth $d$ use $O(n)$ storage.

# Wrapping It Up

Each depth of $T$ requires $O(n)$ of total storage for ASes.

$T$ has height $O(\log n)$.

$T$ needs storage $O(n)$.

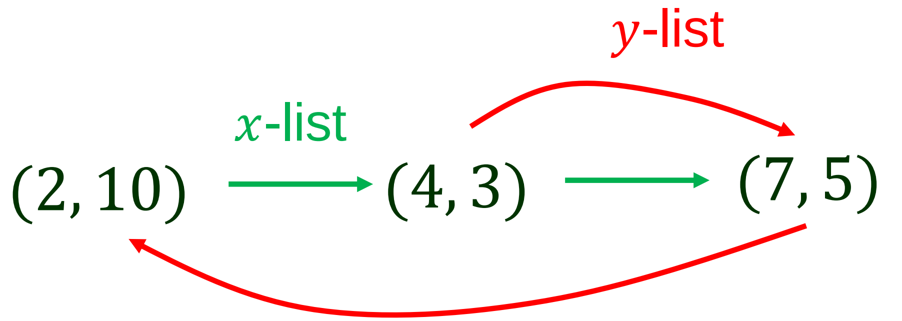
$O(n \log n)$ total storage

# Time of Construction

Maintain $n$ points in two lists:

- sorted on $x$-coordinate

- sorted also on $y$-coordinate

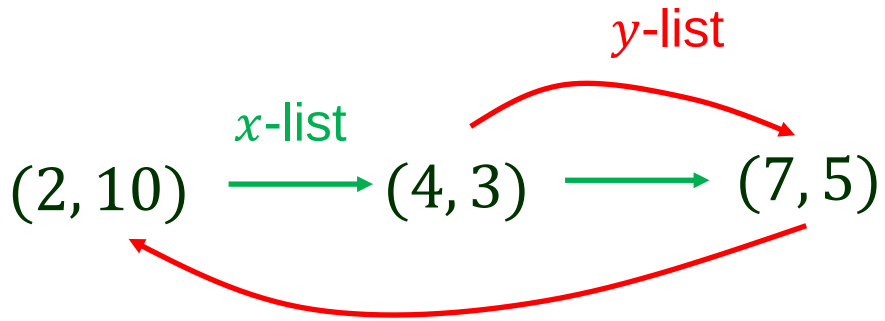


# Time of Construction

Maintain $n$ points in two lists:

- sorted on $x$-coordinate

- sorted also on $y$-coordinate



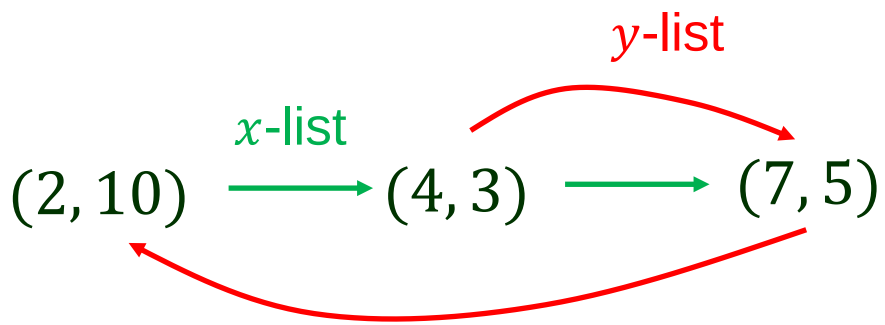

Construction time of each BST is linear to its size.

# Time of Construction

Maintain $n$ points in two lists:

- sorted on $x$-coordinate

- sorted also on $y$-coordinate

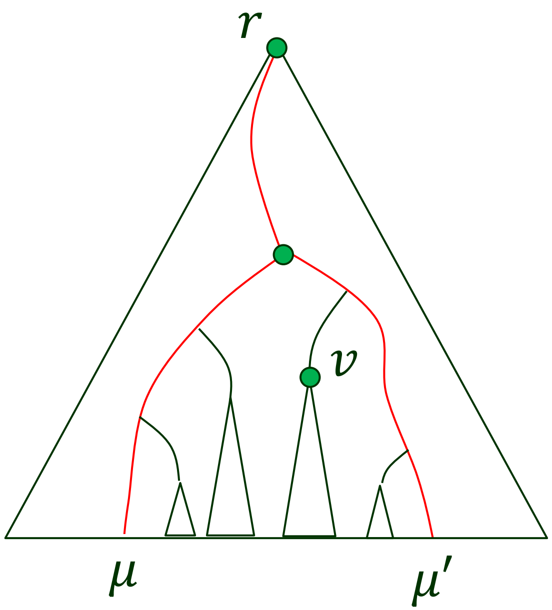



Construction time of each BST is linear to its size.

Total time: $O(n \log n)$

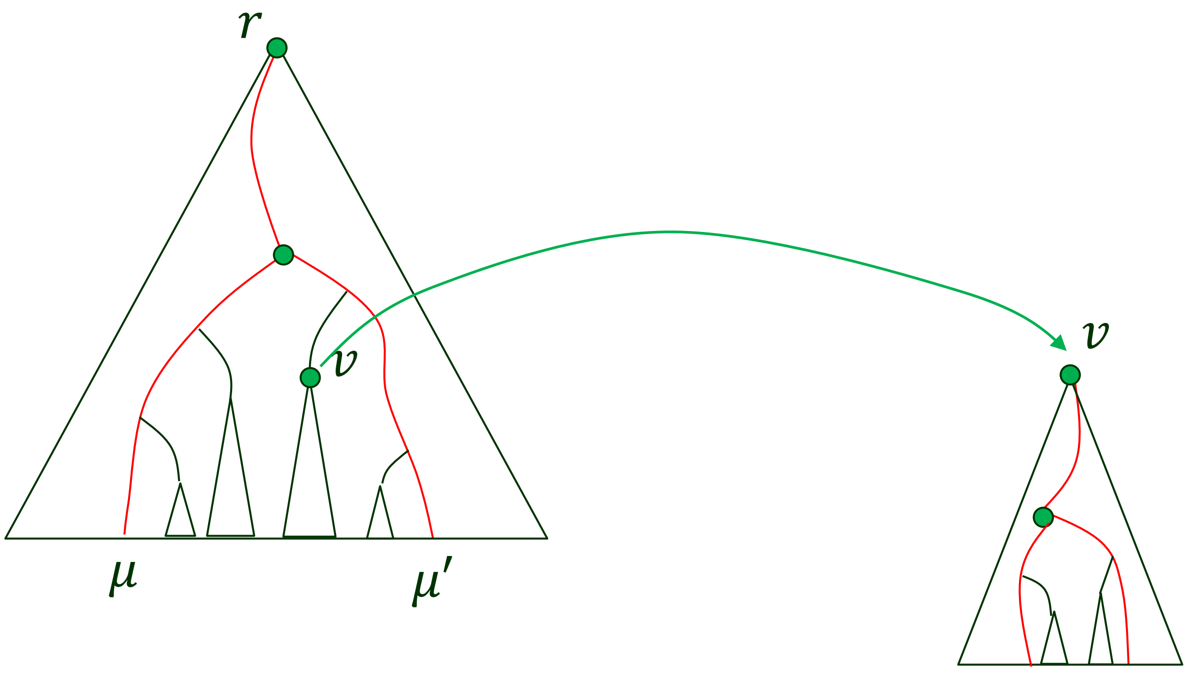
# II. Query

- Selects $O(\log n)$ canonical subsets which together contain points with $x$-coordinate in $[x, x']$.



# II. Query

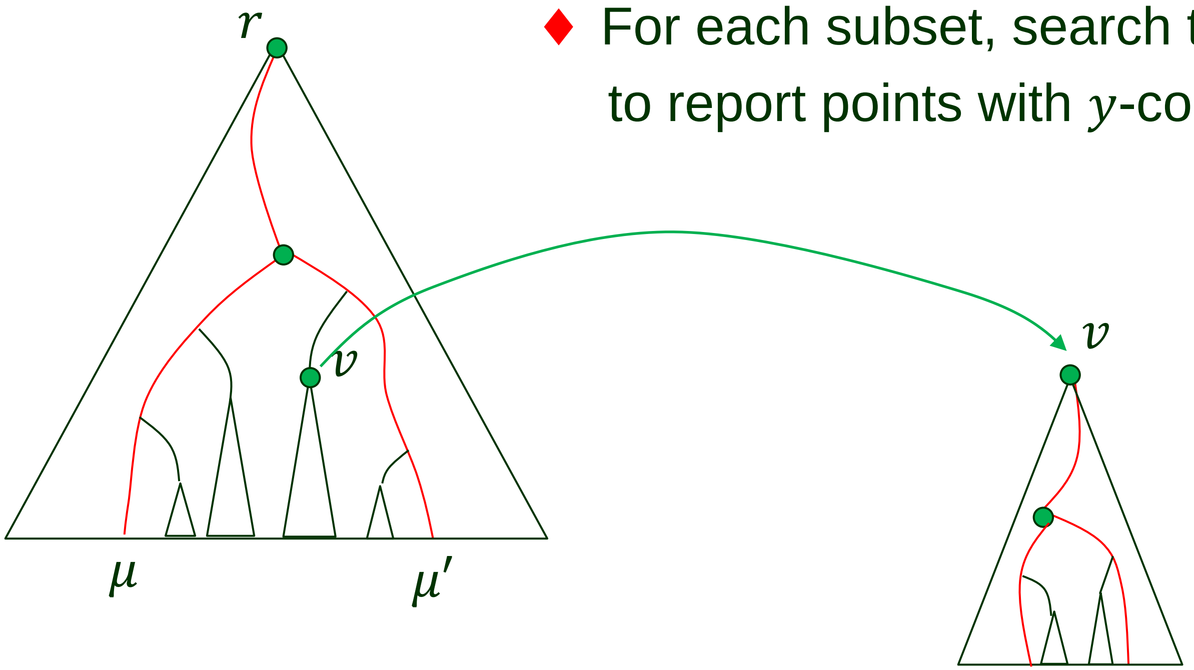
- Selects $O(\log n)$ canonical subsets which together contain points with $x$-coordinate in $[x, x']$.



# II. Query

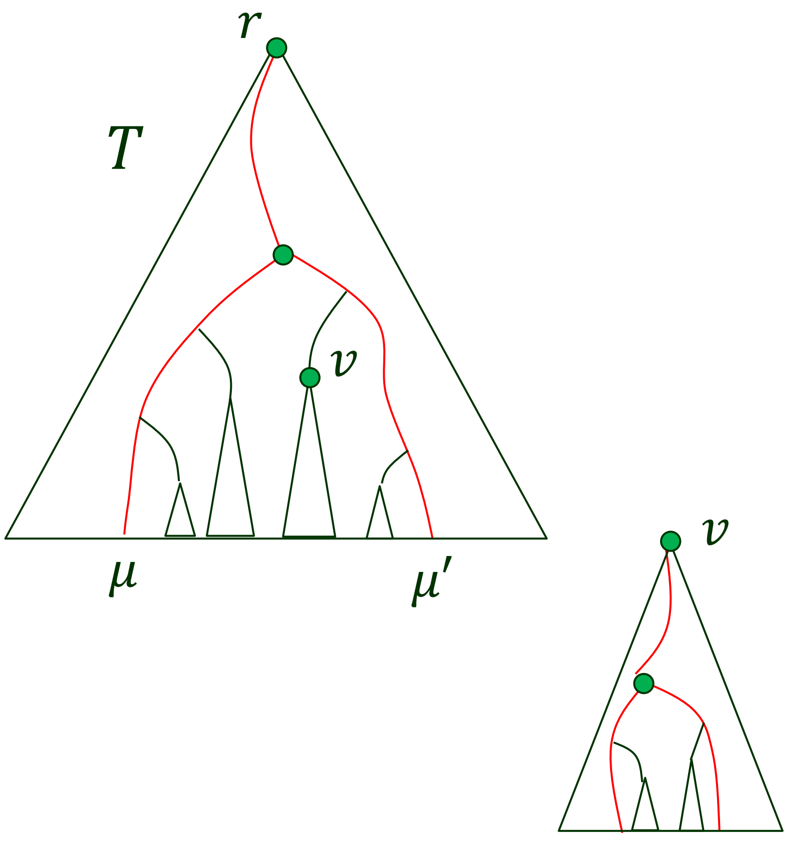
- Selects $O(\log n)$ canonical subsets which together contain points with $x$-coordinate in $[x, x']$.

- For each subset, search the associate structure to report points with $y$-coordinate in $[y, y']$.



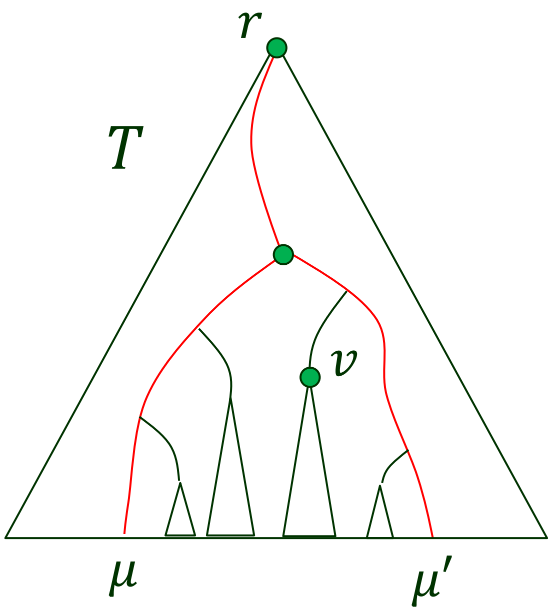
# Time of a Recursive Call

A recursive call happens at a node $v$ in the main tree $T$ *if $v$ is a child of some node on one of the two search paths.*



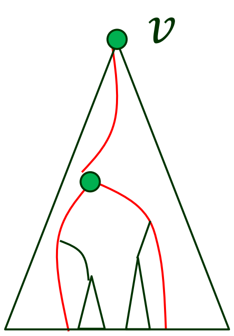
# Time of a Recursive Call

A recursive call happens at a node $v$ in the main tree $T$ *if $v$ is a child of some node on one of the two search paths.*



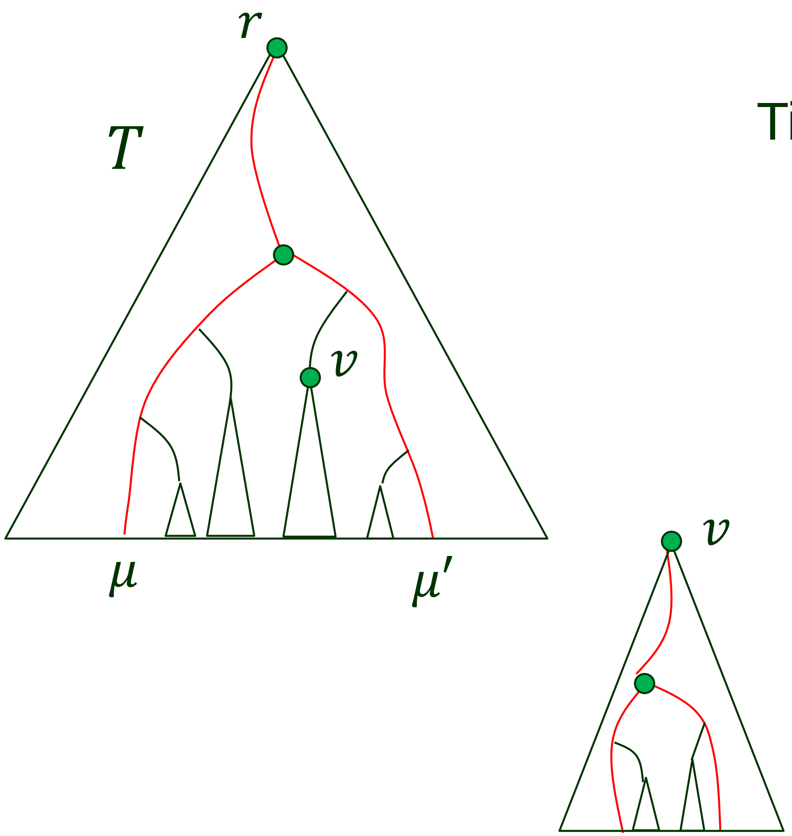

Time of this call:

$$O(\log n + k_v)$$



# Time of a Recursive Call

A recursive call happens at a node $v$ in the main tree $T$ *if $v$ is a child of some node on one of the two search paths.*




Time of this call:

$$O(\log n + k_v)$$

# points reported in this call

# Query Time

$$\sum_{v} O(\log n + k_v) = \sum_{v} O(\log n) + \sum_{v} k_v$$

child of some node on a
search path in the main tree

$$= \sum_{v} O(\log n) + k$$

# Query Time

$$\sum_v O(\log n + k_v) = \sum_v O(\log n) + \sum_v k_v$$

child of some node on a
search path in the main tree

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

# report points

# Query Time

$$\sum_v O(\log n + \ k_v) = \sum_v O(\log n) + \sum_v k_v$$

child of some node on a
search path in the main tree

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

Total number of
such $v$: $O(\log n)$

# report points

# Query Time

$$\sum_v O(\log n + k_v) = \sum_v O(\log n) + \sum_v k_v$$

child of some node on a
search path in the main tree

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

\# report points

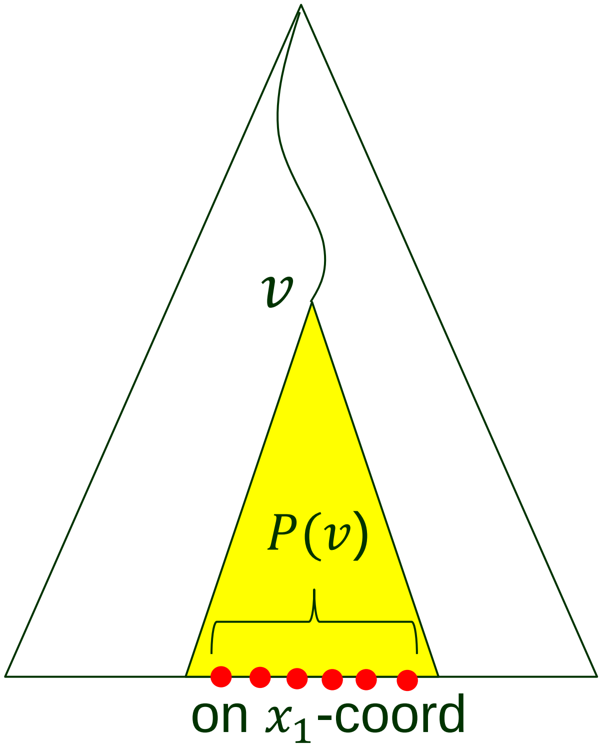
Total number of
such $v$: $O(\log n)$

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

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

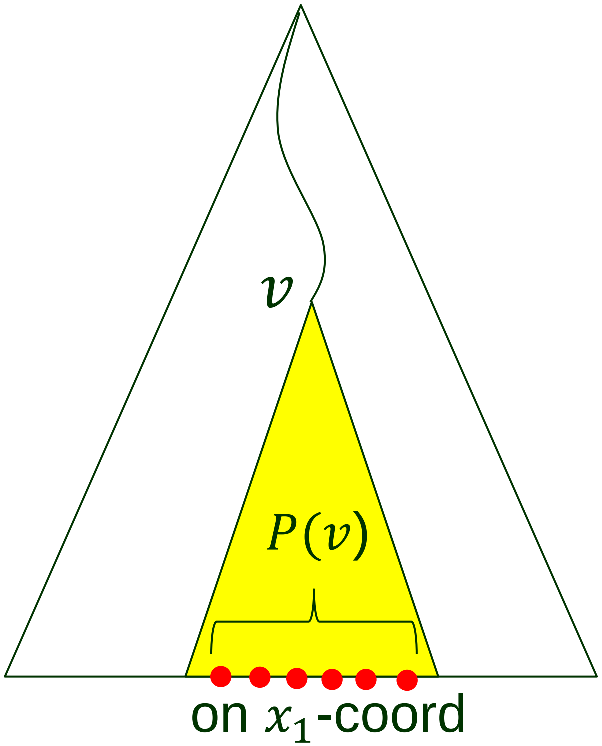
# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).



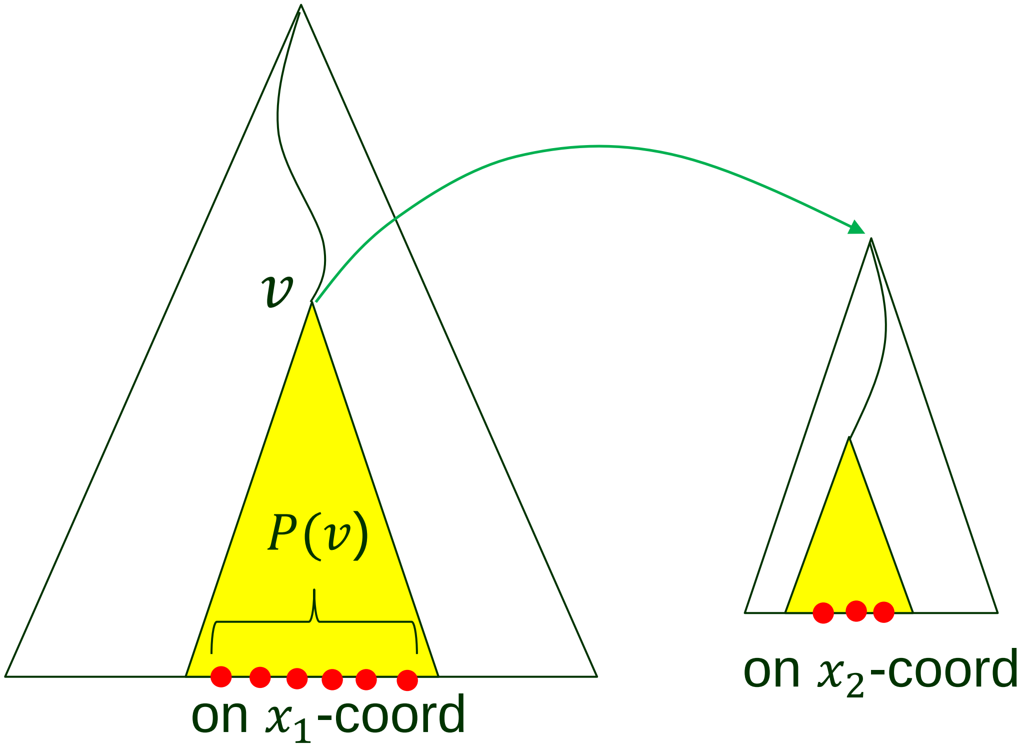
# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).
- For every node $v$, construct a BBST for $P(v)$ on $x_2$-coordinate.



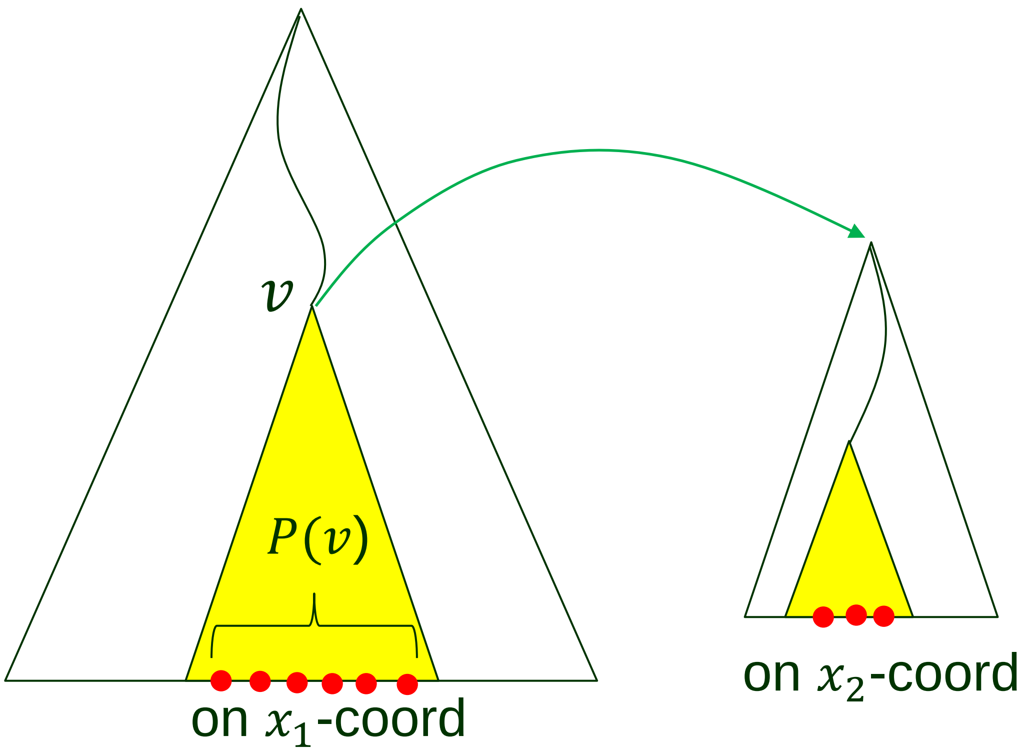
# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).
- For every node $v$, construct a BBST for $P(v)$ on $x_2$-coordinate.



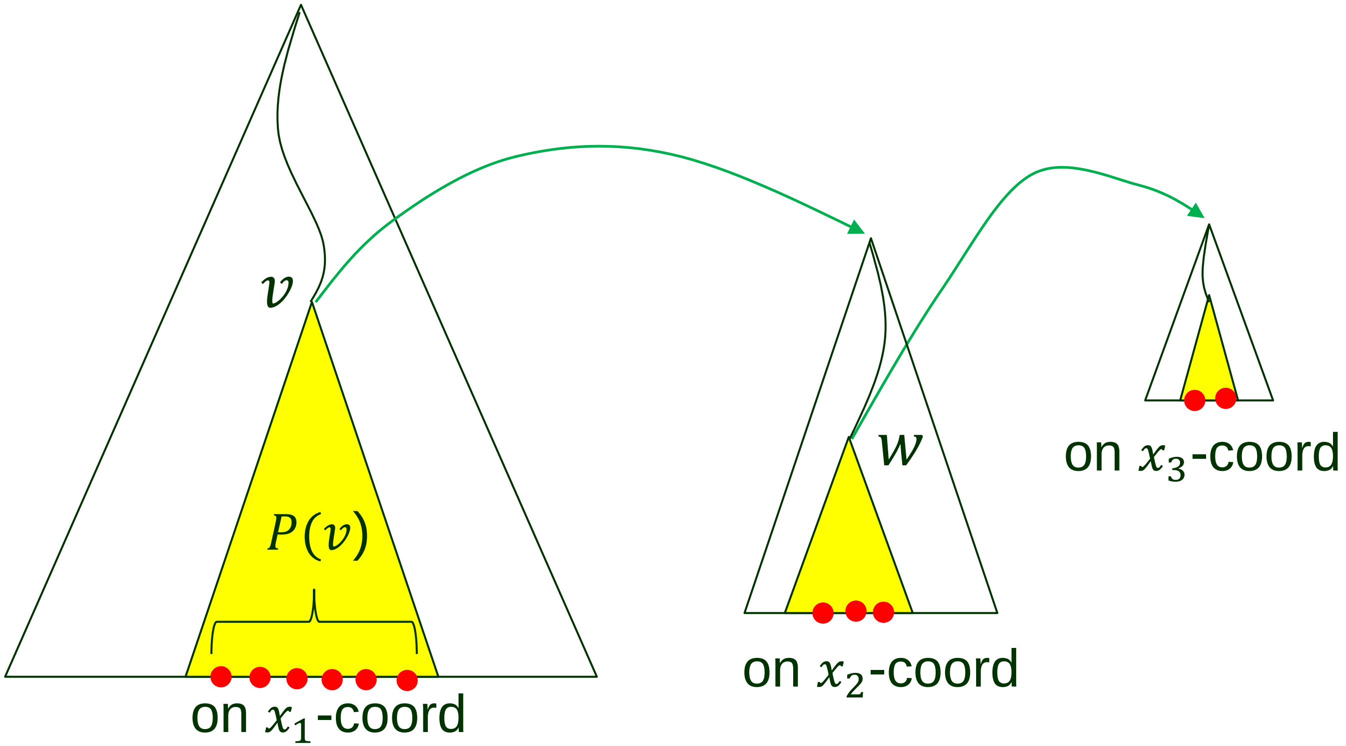
# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).
- For every node $v$, construct a BBST for $P(v)$ on $x_2$-coordinate.
- For every node $w$ in this second level BBST, construct a BBST for $P(w)$ on $x_3$-coordinate.



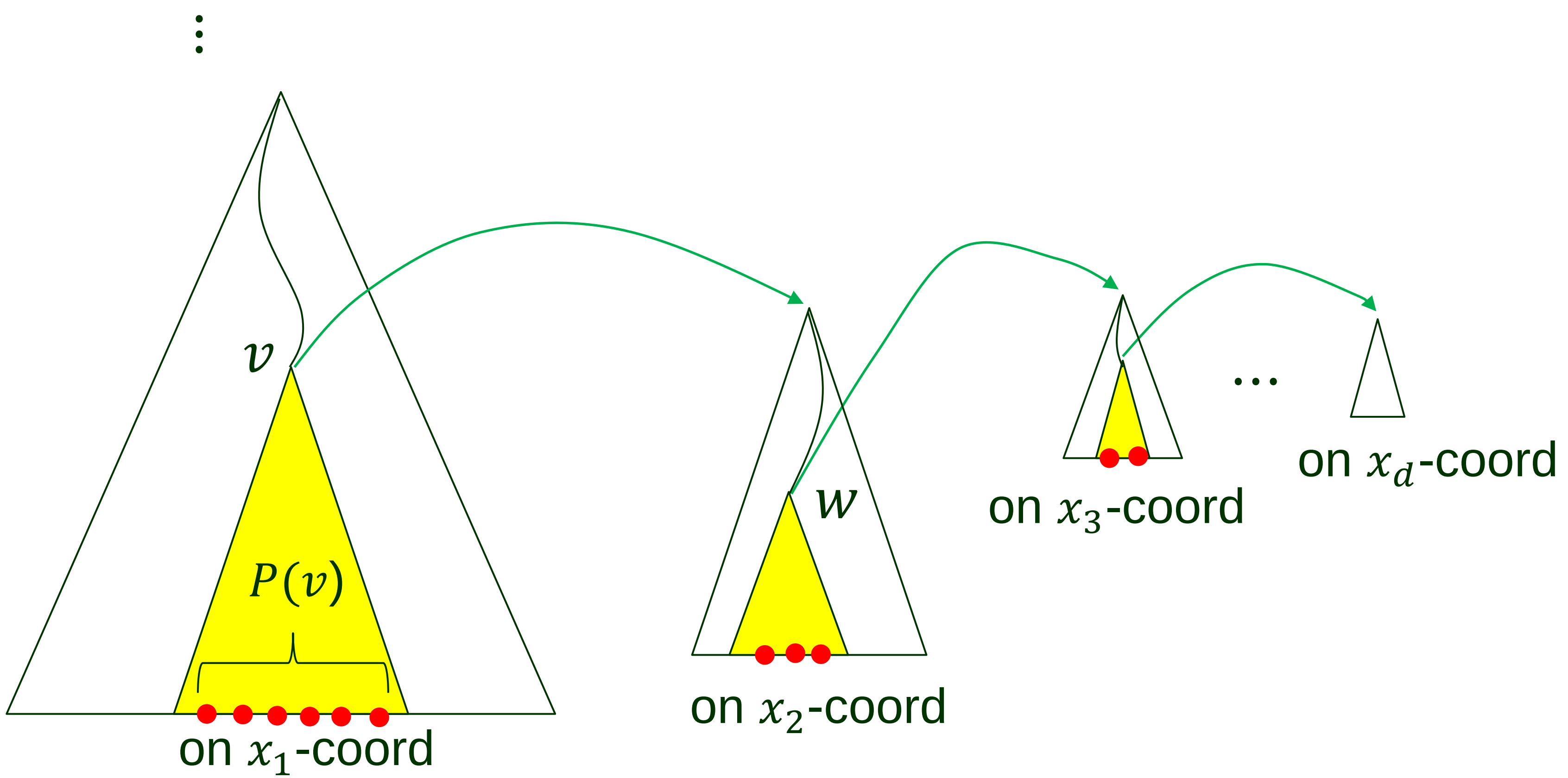
# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).
- For every node $v$, construct a BBST for $P(v)$ on $x_2$-coordinate.
- For every node $w$ in this second level BBST, construct a BBST for $P(w)$ on $x_3$-coordinate.



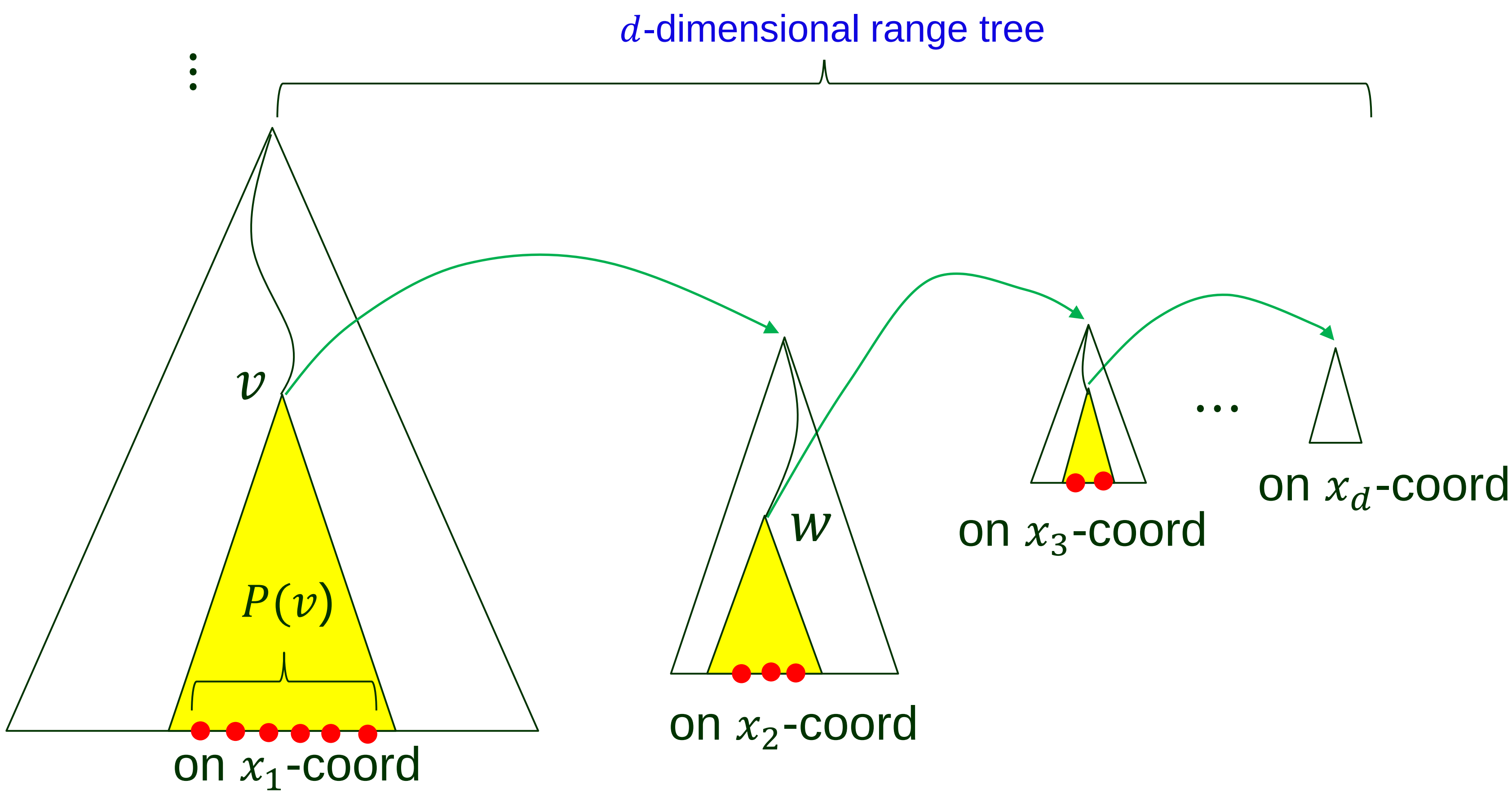
# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).
- For every node $v$, construct a BBST for $P(v)$ on $x_2$-coordinate.
- For every node $w$ in this second level BBST, construct a BBST for $P(w)$ on $x_3$-coordinate.



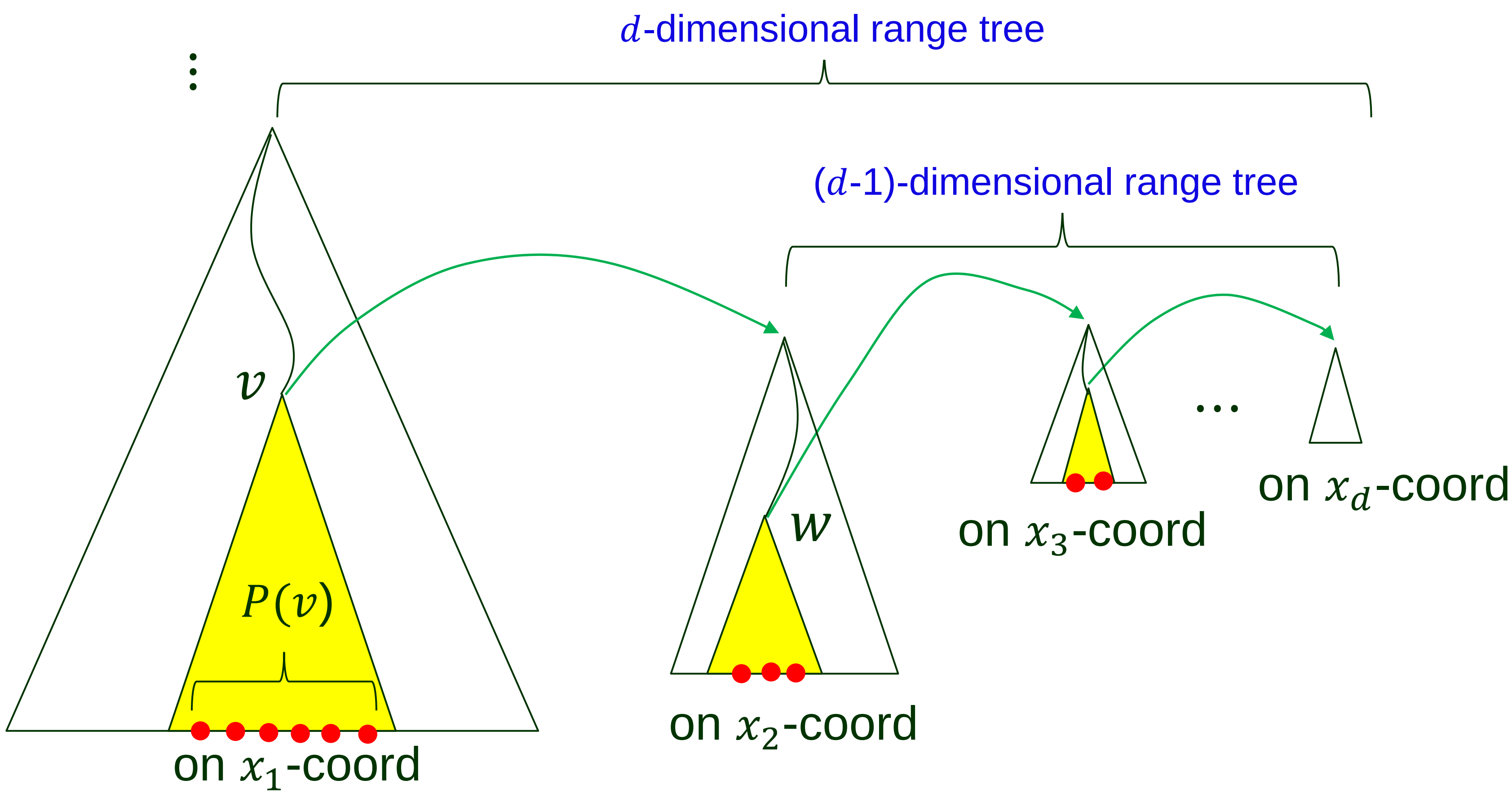
# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).
- For every node $v$, construct a BBST for $P(v)$ on $x_2$-coordinate.
- For every node $w$ in this second level BBST, construct a BBST for $P(w)$ on $x_3$-coordinate.



# III. High-Dimensional Range Trees

- Balanced BST (BBST) on $x_1$-coordinate (main tree).
- For every node $v$, construct a BBST for $P(v)$ on $x_2$-coordinate.
- For every node $w$ in this second level BBST, construct a BBST for $P(w)$ on $x_3$-coordinate.



# Construction Time

$T_d(n)$: construction time for a range tree on a set of $n$ points in $d$-dimensional space.

# Construction Time

$T_d(n)$: construction time for a range tree on a set of $n$ points in $d$-dimensional space.

- $O(n \log n)$ for BBST on $x_1$-coordinate.

# Construction Time

$T_d(n)$: construction time for a range tree on a set of $n$ points in $d$-dimensional space.

- $O(n \log n)$ for BBST on $x_1$-coordinate.
- At any depth of BBST, every point is stored in one AS.

# Construction Time

$T_d(n)$: construction time for a range tree on a set of $n$ points in $d$-dimensional space.

- $O(n \log n)$ for BBST on $x_1$-coordinate.
- At any depth of BBST, every point is stored in one AS.

$\Downarrow$

Time required to build all ASes at the depth is linear in their combined size, and thus, $\underbrace{O(T_{d-1}(n))}_{\text{Time to build the AS of the root}}$.

# Construction Time

$T_d(n)$: construction time for a range tree on a set of $n$ points in $d$-dimensional space.

- $O(n \log n)$ for BBST on $x_1$-coordinate.
- At any depth of BBST, every point is stored in one AS.

$\Downarrow$

Time required to build all ASes at the depth is linear in their combined size, and thus, $\underbrace{O(T_{d-1}(n))}_{\text{Time to build the AS of the root}}$.

$\Downarrow$

$$T_d(n) = O(n \log n) + O(\log n) \cdot T_{d-1}(n)$$

# Construction Time

$T_d(n)$: construction time for a range tree on a set of $n$ points in $d$-dimensional space.

- $O(n \log n)$ for BBST on $x_1$-coordinate.
- At any depth of BBST, every point is stored in one AS.

$$\Downarrow$$

Time required to build all ASes at the depth is linear in their combined size, and thus, $\underbrace{O(T_{d-1}(n))}_{\text{Time to build the AS of the root}}$.

$$\Downarrow$$

$$T_d(n) = O(n \log n) + O(\log n) \cdot T_{d-1}(n)$$

$$\Downarrow$$

$$T_d(n) = O(n \log^{d-1} n)$$

# Query Time

$Q_d(n)$: Time spent in querying a $d$-dimensional range tree on $n$ points.

# Query Time

$Q_d(n)$: Time spent in querying a $d$-dimensional range tree on $n$ points.

- $O(\log n)$ for search in the first-level tree.

# Query Time

$Q_d(n)$: Time spent in querying a $d$-dimensional range tree on $n$ points.

- $O(\log n)$ for search in the first-level tree.

- Querying of $O(\log n)$ $(d-1)$-dimensional range trees.

# Query Time

$Q_d(n)$: Time spent in querying a $d$-dimensional range tree on $n$ points.

- $O(\log n)$ for search in the first-level tree.
- Querying of $O(\log n)$ $(d-1)$-dimensional range trees.

$$\begin{cases} Q_d(n) = O(\log n) + O(\log n) \cdot Q_{d-1}(n) \\ Q_2(n) = O(\log^2 n) \end{cases}$$

# Query Time

$Q_d(n)$: Time spent in querying a $d$-dimensional range tree on $n$ points.

- $O(\log n)$ for search in the first-level tree.
- Querying of $O(\log n)$ $(d-1)$-dimensional range trees.

$$\begin{cases} Q_d(n) = O(\log n) + O(\log n) \cdot Q_{d-1}(n) \\ Q_2(n) = O(\log^2 n) \end{cases}$$

$$\Downarrow$$

$$Q_d(n) = O(\log^d n)$$

# Query Time

$Q_d(n)$: Time spent in querying a $d$-dimensional range tree on $n$ points.

- $O(\log n)$ for search in the first-level tree.
- Querying of $O(\log n)$ $(d-1)$-dimensional range trees.

$$\begin{cases} Q_d(n) = O(\log n) + O(\log n) \cdot Q_{d-1}(n) \\ Q_2(n) = O(\log^2 n) \end{cases}$$

$$\Downarrow$$

$$Q_d(n) = O(\log^d n)$$

- Add the time for reporting $k$ points.

# Query Time

$Q_d(n)$: Time spent in querying a $d$-dimensional range tree on $n$ points.

- $O(\log n)$ for search in the first-level tree.
- Querying of $O(\log n)$ $(d-1)$-dimensional range trees.

$$\begin{cases} Q_d(n) = O(\log n) + O(\log n) \cdot Q_{d-1}(n) \\ Q_2(n) = O(\log^2 n) \end{cases}$$

$$\Downarrow$$

$$Q_d(n) = O(\log^d n)$$

- Add the time for reporting $k$ points.

$O(\log^d n + k)$

# Storage

$S_d(n)$: Storage for $d$-dimensional range tree on $n$ points.

- At any depth of the main BBST, every point is stored in one AS.
- All the $(d-1)$-dimensional range trees generated from these ASes have combined size $O(S_{d-1}(n))$.

# Storage

$S_d(n)$: Storage for $d$-dimensional range tree on $n$ points.

- At any depth of the main BBST, every point is stored in one AS.
- All the $(d-1)$-dimensional range trees generated from these ASes have combined size $O(S_{d-1}(n))$.

$$S_d(n) = O(\log n) \cdot S_{d-1}(n)$$

# Storage

$S_d(n)$: Storage for $d$-dimensional range tree on $n$ points.

- At any depth of the main BBST, every point is stored in one AS.
- All the $(d-1)$-dimensional range trees generated from these ASes have combined size $O(S_{d-1}(n))$.

$$S_d(n) = O(\log n) \cdot S_{d-1}(n)$$

$$S_1(n) = O(n)$$

# Storage

$S_d(n)$: Storage for $d$-dimensional range tree on $n$ points.

- At any depth of the main BBST, every point is stored in one AS.
- All the $(d-1)$-dimensional range trees generated from these ASes have combined size $O(S_{d-1}(n))$.

$$S_d(n) = O(\log n) \cdot S_{d-1}(n)$$

$$S_1(n) = O(n)$$

$$\Downarrow$$

$$S_d(n) = O(n \log^{d-1} n)$$

# General Point Positions

Assumption:

No two points have the same $x$- or $y$-coordinate.

Easy to remove!

# General Point Positions

Assumption:

No two points have the same $x$- or $y$-coordinate.

Easy to remove!

Two distinct points $(p_x, p_y)$ and $(q_x, q_y)$.

# General Point Positions

Assumption:

No two points have the same $x$- or $y$-coordinate.

Easy to remove!

Two distinct points $(p_x, p_y)$ and $(q_x, q_y)$.

- Compare $x$-coordinate:

  $(p_x, p_y) < (q_x, q_y)$ if $p_x < q_x$ or ($p_x = q_x$ and $p_y < q_y$)

# General Point Positions

Assumption:

No two points have the same $x$- or $y$-coordinate.

Easy to remove!

Two distinct points $(p_x, p_y)$ and $(q_x, q_y)$.

- Compare $x$-coordinate:

  $(p_x, p_y) < (q_x, q_y)$ if $p_x < q_x$ or $(p_x = q_x$ and $p_y < q_y)$

- Compare $y$-coordinate:

  $(p_x, p_y) < (q_x, q_y)$ if $p_y < q_y$ or $(p_y = q_y$ and $p_x < q_x)$