Range Trees

Outline:

I. Construction of a 2D range Tree

II. Query with a 2D range Tree

III. High-dimensional range trees
## Improvement on a Kd-Tree

<table>
<thead>
<tr>
<th>Query time</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(\sqrt{n} + k)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

relatively high!
**Improvement on a Kd-Tree**

<table>
<thead>
<tr>
<th>Query time</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>relatively high! $O(\sqrt{n} + k)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

$O(\log^2 n + k)$
Improvement on a Kd-Tree

<table>
<thead>
<tr>
<th>Query time</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(\sqrt{n} + k)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>$O(\log^2 n + k)$</td>
<td>$O(n \log n)$</td>
</tr>
</tbody>
</table>
# Improvement on a Kd-Tree

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

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

**How?**
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

\[ P(v) : \text{leaves (points) of the subtree rooted at } v \]

\[ P(r) = P = \{ p_{i_1}, \ldots, p_{i_n} \} \]
Canonical Subsets

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

$P(r) = P = \{p_{i1}, ..., p_{in}\}$

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

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

\( P(\nu) \): leaves (points) of the subtree rooted at \( \nu \)

\[ P(r) = P = \{p_{i1}, \ldots, p_{in}\} \]

\[ P(\nu_1) \cup P(\nu_2) = P \]
\[ P(\nu_1) \cap P(\nu_2) = \emptyset \]

\[ P(\nu_3) \cup P(\nu_4) \cup P(\nu_5) \cup P(\nu_6) = P \]

Non-intersecting subsets
1. 2D Query

1. Find points with $x$-coordinate $\in [x, x']$
I. 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 $r \sim \mu$ when it makes a left turn, or
- left child of a node on the path $r \sim \mu'$ when it makes a right turn.
I. 2D Query

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

How to make this query fast?
2D Range Tree

Main tree

balanced BST on $x$-coordinate

$T$

$P(v)$
2D Range Tree

Main tree

$T$

$P(v)$

balanced BST on $x$-coordinate

$T_{assoc}(v)$

balanced BST on $y$-coordinate

$P(v)$
2D Range Tree

Main tree

 Associated structure (AS)
 (for every node \( v \) in the main tree to store \( P(v) \) )

\( T \)

\( T_{assoc}(v) \)

balanced BST on \( x \)-coordinate

balanced BST on \( y \)-coordinate

\( P(v) \)
Recursive Construction

\[ P = \{p_1, \ldots, p_n\} \text{ sorted on } x\text{-coordinate} \]

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

\[ P = \{p_1, \ldots, p_n\} \text{ sorted on } x\text{-coordinate} \]

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

\[ P = \{p_1, \ldots, p_n\} \text{ sorted on } x\text{-coordinate} \]

\[ \text{Build2DRangeTree}(P) \]

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
      \[ \nu \xrightarrow{p_1} T_{assoc} \]
4. else split \( P \) at the medium \( x\)-coordinate \((x_{mid})\) into \( P_{left} \) and \( P_{right} \)
5.
6.
Recursive Construction

\[ P = \{ p_1, \ldots, p_n \} \text{ sorted on } x\text{-coordinate} \]

\textbf{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 \[ \nu \quad \begin{array}{c} p_1 \end{array} \begin{array}{c} p_1 \end{array} \quad T_{assoc} \]
4. else split \( P \) at the medium \( x\)-coordinate \((x_{\text{mid}})\) into \( P_{\text{left}} \) and \( P_{\text{right}} \)
5. \[ \nu_{\text{left}} \leftarrow \text{Build2DRangeTree}(P_{\text{left}}) \]
6. \[ \nu_{\text{right}} \leftarrow \text{Build2DRangeTree}(P_{\text{right}}) \]
Combining Range Trees

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

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 $r \sim p$. 
How Often is a Point Stored?

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

At every depth $d$, $p$ is stored in exactly one AS.
At depth $d$ every point is stored at a leaf of *exactly one* AS.
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.
At depth $d$ every point is stored at a leaf of \textit{exactly one} AS.
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.
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 $O(#\text{leaves})$. 
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 $O(#\text{leaves})$.

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
Maintain $n$ points in two lists:

- sorted on $x$-coordinate
- sorted also on $y$-coordinate

(2, 10) $\xrightarrow{x}$-list $\xrightarrow{y}$ (4, 3) $\xrightarrow{y}$-list $\xrightarrow{x}$ (7, 5)
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 $[x, x']$. 

Diagram: 

- $r$
- $\mu$
- $\mu'$
- $\nu$
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

\[ = \sum_v O(\log n) + k \]
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

\[ = \sum_v O(\log n) + k \]

# 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

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

\[ = \sum_v O(\log n) + k \]

# 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

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

\[ = \sum_v O(\log n) + k \]

\[ = O(\log^2 n) + k \]

# report points

\[ = O(\log^2 n + k) \]
III. High-Dimensional Range Trees

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

$\nu$ on $x_1$-coord

$P(\nu)$
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.

\[
\vdots
\]

\[
\begin{array}{c}
\text{on } x_1\text{-coord} \\
\text{on } x_2\text{-coord} \\
\text{on } x_3\text{-coord} \\
\vdots \\
\text{on } x_d\text{-coord}
\end{array}
\]
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.

$d$-dimensional range tree

\[
\begin{align*}
\vdots \\
\end{align*}
\]
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.

\[
\begin{align*}
\vdots \\
\text{on } x_1\text{-coord} \\
\text{on } x_2\text{-coord} \\
\text{on } x_3\text{-coord} \\
\text{on } x_d\text{-coord}
\end{align*}
\]
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, $O(T_{d-1}(n))$.

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, \( O(T_{d-1}(n)) \).

\[ \downarrow \]

Time to build the AS of the root

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

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

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

\[
Q_d(n) = O(\log n) + O(\log n) \cdot Q_{d-1}(n)
\]

\[
Q_2(n) = O(\log^2 n)
\]
**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.

\[
\begin{align*}
Q_d(n) &= O(\log n) + O(\log n) \cdot Q_{d-1}(n) \\
Q_2(n) &= O(\log^2 n)
\end{align*}
\]

\[ Q_d(n) = O(\log^d n) \]
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.

\[
\begin{align*}
Q_d(n) &= O(\log n) + O(\log n) \cdot Q_{d-1}(n) \\
Q_2(n) &= O(\log^2 n)
\end{align*}
\]

\[ Q_d(n) = O(\log^d n) \]

- 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.

\[
\begin{align*}
Q_d(n) &= O(\log n) + O(\log n) \cdot Q_{d-1}(n) \\
Q_2(n) &= O(\log^2 n)
\end{align*}
\]

\[ Q_d(n) = O(\log^d n) \]

- 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 \( O(S_{d-1}(n)) \).

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

\[ S_1(n) = O(n) \]
$S_d(n)$: 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 $O(S_{d-1}(n))$.

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

$S_1(n) = O(n)$

\[
S_d(n) = O(n \log^{d-1} n)
\]
General Point Positions

Assumption:

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

Easy to remove!
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:

\[
(p_x, p_y) < (q_x, q_y) \text{ if } p_x < q_x \text{ or } (p_x = q_x \text{ and } p_y < q_y)
\]
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:

  \[(p_x, p_y) < (q_x, q_y) \text{ if } p_x < q_x \text{ or } (p_x = q_x \text{ and } p_y < q_y)\]

- Compare $y$-coordinate:

  \[(p_x, p_y) < (q_x, q_y) \text{ if } p_y < q_y \text{ or } (p_y = q_y \text{ and } p_x < q_x)\]