Range searching with Range Trees

Given a set of points, preprocess them into a data structure to support fast range queries.

**1D**
- **BST**
  - Build: $O(n \log n)$
  - Space: $O(n)$
  - Range queries: $O(p n + k)$

**2D**
- **2D Range Queries**
  - Denote query $[x_1, x_2] \times [y_1, y_2]$
  - Idea:
    - Find all points with $x$-coordinates in $[x_1, x_2]$
    - Of all these points, find all points with $y$-coordinate in $[y_1, y_2]$

**Towards 2D Range Trees**
- Store points in a BST by $x$-coordinate
- Use BST to find all points with the $x$-coordinates in $[x_1, x_2]$
- Of all these points, find all points with $y$-coordinate in $[y_1, y_2]$

**Example Diagrams**
- 2D Range Queries
- Towards 2D Range Trees
Towards 2D Range Trees

- Store points in a BBST by x-coordinate
- Use BBST to find all points with the x-coordinates in \([x_1, x_2]\)
- Of all these points, find all points with y-coordinates in \([y_1, y_2]\)

They are sitting in \(O(lg n)\) subtrees

The \(k\) points in the range are in \(O(lg n)\) subtrees

For each subtree we need all points in \([y_1, y_2]\).

A 1-dimensional range query with \([25, 90]\)

General 1D range query
Towards 2D Range Trees

• Store points in a BBST by x-coordinate
• Use BBST to find all points with the x-coordinates in \([x_1, x_2]\)
• Of all these points, find all points with y-coordinates \([y_1, y_2]\)

They are sitting in \(O(\log n)\) subtrees

For each subtree we need all points in \([y_1, y_2]\)

What is a good data structure for range search on y?

The 2D Range Tree

\(P\) is a set of points
\(\text{RangeTree}(P)\) is
• A BBST \(T\) of \(P\) on x-coordinates
• Any node \(v\) in \(T\) stores a BBST \(T_{\text{assoc}}(v)\) of \(P(v)\), by y-coordinates

\(P(v)\): all points in subtree rooted at \(v\)

Class work
• Show the BBST with all data in leaves for \(P = \{1,2,3,4,5,6,7,8,9,10\}\)
• Write pseudocode for the algorithm to build BBST(P)

BuildBBST(P)
The 2D Range Tree

P = set of points
RangeTree(P) is
• A BBST T of P on x-coord
• Any node v in T stores a BBST T_{assoc} of P(v), by y-coord

Class work
• Let P = [(1,4), (5,8), (4,1), (7,3), (3,2), (2,6), (8,7)].
  Show the range tree of P.

Questions
• How do you build it and how fast?
• How much space does it take?
• How do you answer range queries and how fast?
Let \( P = \{p_1, p_2, \ldots, p_n\} \). Assume \( P \) sorted by x-coord.

Algorithm \text{Build2DRT}(P):

1. Construct the associated structure: build a BBST \( T_{assoc} \) on the set of y-coordinates of \( P \).
2. If \( P \) contains only one point:
   - create a leaf \( v \) storing this point, create its \( T_{assoc} \) and return \( v \).
3. Else:
   1. Partition \( P \) into 2 sets w.r.t. the median coordinate \( x_{middle} \):
      \[ P_{left} = \{p \in P. p_x \leq x_{middle}\}, \quad P_{right}\ldots \]
   2. \( v_{left} = \text{Build2DRT}(P_{left}) \)
   3. \( v_{right} = \text{Build2DRT}(P_{right}) \)
   4. Create a node \( v \) storing \( x_{middle} \), make \( v_{left} \) its left child, make \( v_{right} \) its right child, make \( T_{assoc} \) its associate structure.
   5. return \( v \).

Building a 2D Range Tree

How fast?

- Let \( T(n) \) be the time of \( \text{Build2DRT}(P) \) of \( n \) points.
- Constructing a BBST on an unsorted set of keys takes \( O(n \lg n) \).
- Then,
  \[ T(n) = 2T(n/2) + O(n \lg n) \]
- This solves to \( O(n \lg^2 n) \).

Building a 2D Range Tree

- Common trick: pre-sort \( P \) on y-coord and pass it along as argument

\[ (P_x, P_y) \text{ is set of points sorted by x-coord, } P \text{ is set of points sorted by y-coord} \]

\[ \text{Build2DRT}(P_x, P_y) \]

- Maintain the sorted sets through recursion

\( P_1 \text{ sorted by x, } P \text{ sorted by y} \)

\( P_2 \text{ sorted by x, } P_2 \text{ sorted by y} \)

- If keys are in order, a BBST can be built in \( O(n) \).
- We have
  \[ T(n) = 2T(n/2) + O(n) \text{ which solves to } O(n \lg n) \]

The 2D Range Tree

- How much space does a range tree use?

Two arguments can be made:

- At each level in the tree, each point is stored exactly once (in the associated structure of precisely one node). So every level stores all points and uses \( O(n) \) space: \( \implies O(n \lg n) \) space.
- Each point \( p \) is stored in the associated structures of all nodes on the path from root to \( p \). So one point is stored \( O(\lg n) \) times: \( \implies O(n \lg^2 n) \) space.
Range queries with the 2D Range Tree

- Find the split node $x_{\text{split}}$ where the search paths for $x_1$ and $x_2$ split.
- Follow path root to $x_{\text{split}}$: for each node $v$ to the right of the path, query its associated structure $T_{\text{assoc}}(v)$ with $[y_1,y_2]$.
- Follow path root to $x_{\text{split}}$: for each node $v$ to the left of the path, query its associated structure $T_{\text{assoc}}(v)$ with $[y_1,y_2]$.

How long does this take?

Range queries with the 2D Range Tree

- How long does a range query take?
- There are $O(\log n)$ subtrees in between the paths.
- We query each one of them using its associated structure.
- Querying its $T_{\text{assoc}}$ takes $O(\log n + k')$.
- Overall it takes $\sum O(\log n + k') = O(\log^2 n + k)$.

Comparison

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\log n$</th>
<th>$\log^2 n$</th>
<th>$\sqrt{n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>4</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>64</td>
<td>6</td>
<td>36</td>
<td>8</td>
</tr>
<tr>
<td>256</td>
<td>8</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>1024</td>
<td>10</td>
<td>100</td>
<td>32</td>
</tr>
<tr>
<td>4096</td>
<td>12</td>
<td>144</td>
<td>64</td>
</tr>
<tr>
<td>16384</td>
<td>14</td>
<td>196</td>
<td>128</td>
</tr>
<tr>
<td>65536</td>
<td>16</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>1M</td>
<td>20</td>
<td>450</td>
<td>1K</td>
</tr>
<tr>
<td>16M</td>
<td>24</td>
<td>576</td>
<td>4K</td>
</tr>
</tbody>
</table>

3D Range Trees

- $P =$ set of points in 3D
- $\text{3DRangeTree}(P)$
  - Construct a BBST on x-coord
  - Each node $v$ will have an associated structure that's a 2D range tree for $P(v)$ on the remaining coords.
3D Range Trees

Size:
- An associated structure for $n$ points uses $O(n \log n)$ space. Each point is stored in all associated structures of all its ancestors $\Rightarrow O(n \log^2 n)$

Let's try this recursively
- Let $S(n)$ be the size of a 3D Range Tree of $n$ points
- Find a recurrence for $S(n)$
- Think about how you build it: you build an associated structure for $P$ that's a 2D range tree; then you build recursively a 3D range tree for the left and right half of the points
- $S(n) = 2S(n/2) + S(n)$
- This solves to $O(n \log^2 n)$

3D Range Trees

Build time:
- Think recursively
- Let $B(n)$ be the time to build a 3D Range Tree of $n$ points
- Find a recurrence for $B(n)$
- Think about how you build it: you build an associated structure for $P$ that's a 2D range tree; then you build recursively a 3D range tree for the left and right half of the points
- $B(n) = 2B(n/2) + B(n)$
- This solves to $O(n \log n)$

3D Range Trees

Query:
- Query BBST on x-coord to find $O(\log n)$ nodes
- Then perform a 2D range query in each node

Time?
- Let $Q(n)$ be the time to answer a 3D range query
- Find a recurrence for $Q(n)$
- $Q(n) = O(\log n) + O(\log n) - Q(n)$
- This solves to $O(\log^3 n + k)$

Comparison RangeTree and kdtree

<table>
<thead>
<tr>
<th>n</th>
<th>$\log n$</th>
<th>$\log^2 n$</th>
<th>$n^{7/4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>10</td>
<td>10,000</td>
<td>181</td>
</tr>
<tr>
<td>65,536</td>
<td>16</td>
<td>65,536</td>
<td>4096</td>
</tr>
<tr>
<td>1M</td>
<td>20</td>
<td>160,000</td>
<td>32,768</td>
</tr>
<tr>
<td>1G</td>
<td>30</td>
<td>810,000</td>
<td>5,931,641</td>
</tr>
<tr>
<td>1T</td>
<td>40</td>
<td>2,560,000</td>
<td>1G</td>
</tr>
</tbody>
</table>