Module 4: Index Structures

Lecture 15: Range Trees and Interval Trees

Prev topic | Next topic | Prev page | Next page

The Lecture Contains:

- Range tree
  - Multi-dimensional range trees
  - Analysis of range trees
- Windowing queries
- Interval tree
  - Example
  - Searching

Range tree

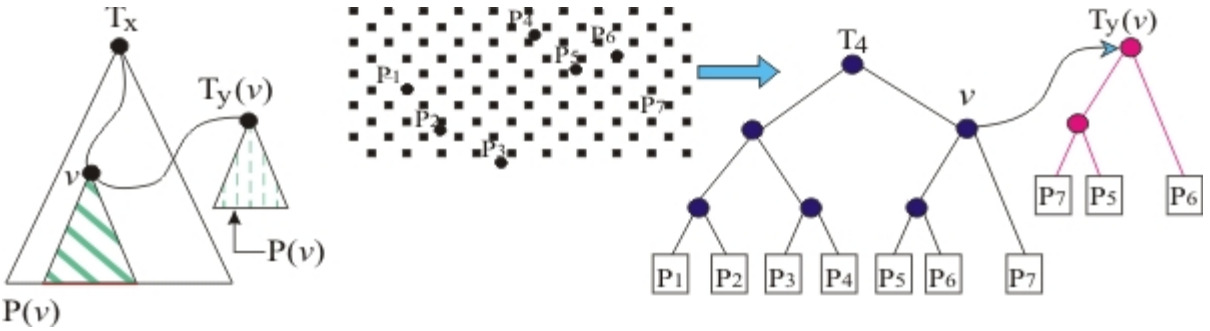
- In 1-dimension, similar to a balanced binary search tree
- Leaves are sorted and *doubly linked*

Range tree

- In 1-dimension, similar to a balanced binary search tree
- Leaves are sorted and *doubly linked*
- Range search proceeds by descending to the leaf <= to the minimum of the range
- Then, traverse forward pointers till maximum of the range is exceeded
- Time taken is $O(\lg N + T)$ where T is size of the answer set
- Space is $O(N)$

Multi-dimensional range trees

- BST of BSTs of BSTs of . . .
- 1-dimensional range tree of ($(d-1)$)-dimensional range trees
- Order dimensions as $x_1, x_2, \ldots, x_d$ and build accordingly
- For each canonical subset of points for a node built on x_1, there is a range tree of ($(d-1)$)-dimensionality associated with the node



Analysis of range trees

- Storage space is $O(N \lg^{d-1} N)$
- Time taken is $O(\lg^d N + T)$

Analysis of range trees

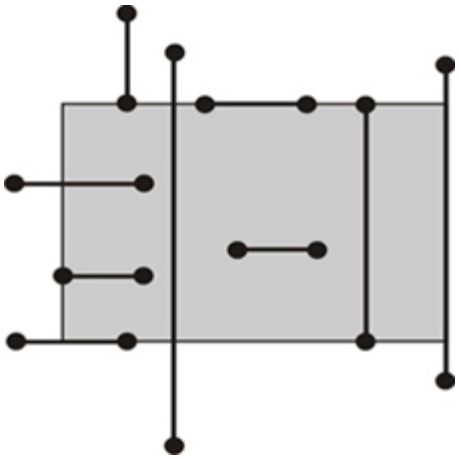
- Storage space is $O(N \lg^{d-1} N)$
- Time taken is $O(\lg^d N + T)$
- Search in x_1 and identify left-most and right-most leaf nodes L and R that cover x_1's range
- Find *least common ancestor* Q of L and R
- Perform range search for dimensions $x_2, \ldots, x_d$ for all range trees rooted at nodes between Q and L and Q and R

Analysis of range trees

- Storage space is $O(N \lg^{d-1} N)$
- Time taken is $O(\lg^d N + T)$
- Search in x_1 and identify left-most and right-most leaf nodes L and R that cover x_1's range
- Find *least common ancestor* Q of L and R
- Perform range search for dimensions $x_2, \ldots, x_d$ for all range trees rooted at nodes between Q and L and Q and R
- Fractional cascading improves search time in 2 dimensions to $O(\lg N + T)$

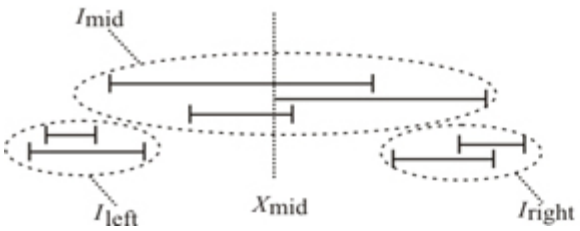
Windowing queries

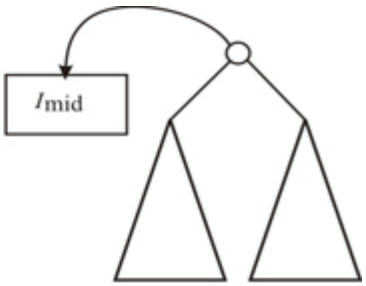
- Return all (axis-parallel) segments that are visible through the query window
- Segments which have one (or two) endpoints in the window can be searched using range trees
    - Bookkeeping to avoid duplication of segments with both end-points in the window
- Stabbing queries to nd out lines (equivalently points) that cut a particular arm $y = q_y$ of the window
    - Line $x = a$ is equivalent to the point (a, q_y)
    - Interval tree to search these



Interval tree

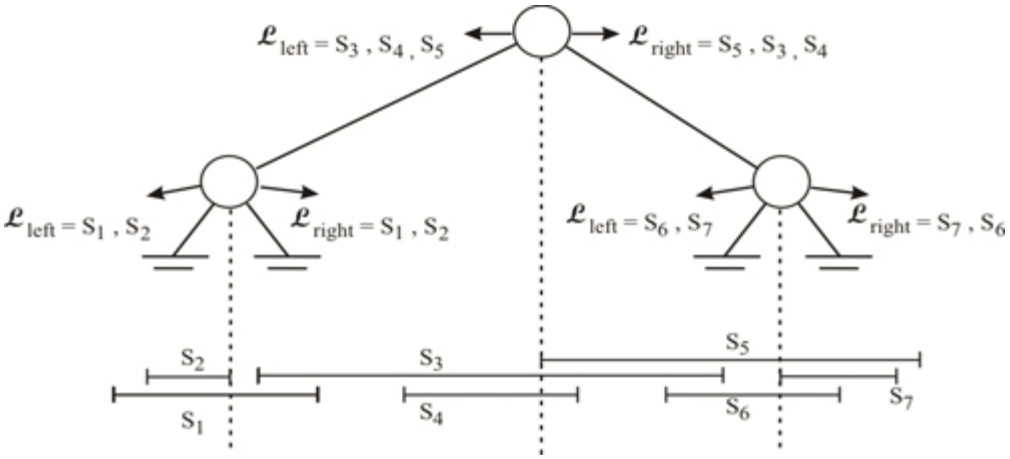
- Binary tree
- Each node storing 5 different pieces of information:
  - Bookkeeping to avoid duplication of segments with both end-points in the window
  - Median point x_{mid}
  - Left child pointer for intervals I_{left} ending before x_{mid}
  - Right child pointer for intervals I_{right} starting after x_{mid}
  - Lists storing all overlapping intervals I_{mid}
    - Left list L_{left} sorted in ascending order by left end-points
    - Right list L_{right} sorted in descending order by right end-points





Example

- $O(N)$ storage
- $O(N \lg N)$ construction time
- $O(\lg N + T)$ query time for *stabbing queries*
  - Find all intervals that intersect (stab) a given vertical line



Searching

- Find if query q_x is greater than max of left child
  - If not, recurse to left child
- Find if query q_x is lesser than min of right child
  - If not, recurse to right child
- Note that only one of the above cases can occur
  - Max of left child $< x_{mid} <$ of right child
- Find if q_x overlaps intervals in I_{mid}
  - If yes, traverse L_{left} and L_{right}

Searching

- Find if query q_x is greater than max of left child
  - If not, recurse to left child
- Find if query q_x is lesser than min of right child
  - If not, recurse to right child
- Note that only one of the above cases can occur
  - Max of left child $< x_{mid} <$ of right child
- Find if q_x overlaps intervals in I_{mid}
  - If yes, traverse L_{left} and L_{right}
- For interval queries and windowing queries, maintain L_{left} and L_{right} as range trees
  - Searches in y-dimension to nd intersecting segments
  - Searching time becomes $O(log^2 N + T)$