Summary of Distance Browsing in Spatial Databases

By S. Xiao Fernández Marín

1 Summary

Distance scan & kNN
• Distance scan
– Scan the table/index in the order of distance from a given geometry
– Shortest to longest (ASC) or longest to shortest (DESC)
– Unlimited
• k nearest neighbours
– Find the k closest geometries to a geometry
– Always shortest to longest (ASC)
– Limited to k results

Alternatives not distance scan • Read the entire input, sort on distance
– Inefficient if k is small and table size is large
– Efficient if reading most of the data: Exact cut-off point will vary between implementations
• Read m > k rows and match against other criteria
– Must retry with larger m if selection filters out to less than k rows
• Incremental nearest neighbours/distance scan
– No predefined limit
– Less efficient if k is large
• Filter on other condition first and then sort on distance

R-tree - used as a base

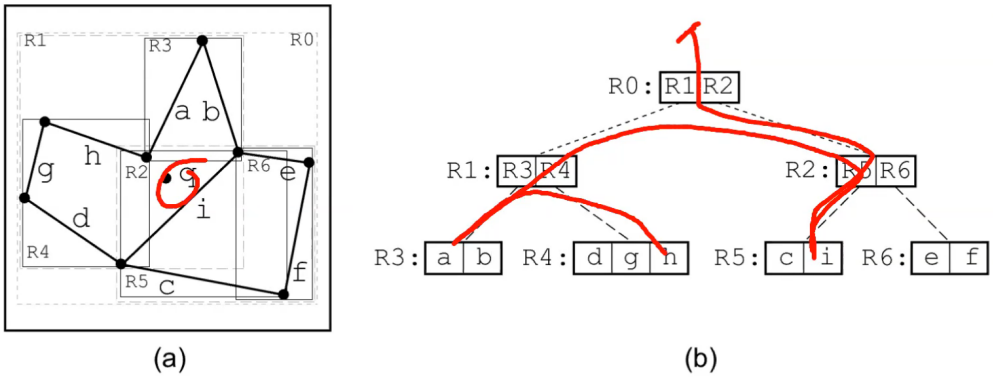

Fig. 1. An R-tree index for a set of nine line segments. (a) Spatial rendering of the line segments and bounding rectangles; (b) a tree access structure for (a). In the interest of clarity, the bounding rectangles for the individual line segments are omitted from (a) .

Figure 1: R-tree

Priority queue - combine it with an R-tree
• A sorted queue
• Elements are inserted based on a measure
• The element with the largest or smallest value is extracted first
• Doesn't need to maintain a total ordering of every element
• Sorting cost can be divided between insertion and extraction

Algorithm outline
• Insert root node into priority queue
• Extract closest element from priority queue
– If non-leaf node: Compute minimum distance to each immediate child and add them to the priority queue
– If leaf node: Compute minimum ditance to each geometry in leaf node and add them to the priority queue
– If geometry: Output geometry as next row
– Repeat until enough rows have been read
Note: This outline ignores important details around handling duplicates and nodes vs. geometries

General and R-Tree algorithm The general algorithm just finds where the spatial object (geometry) is, the R-Tree algorithm returns the bounding box or the geometry.

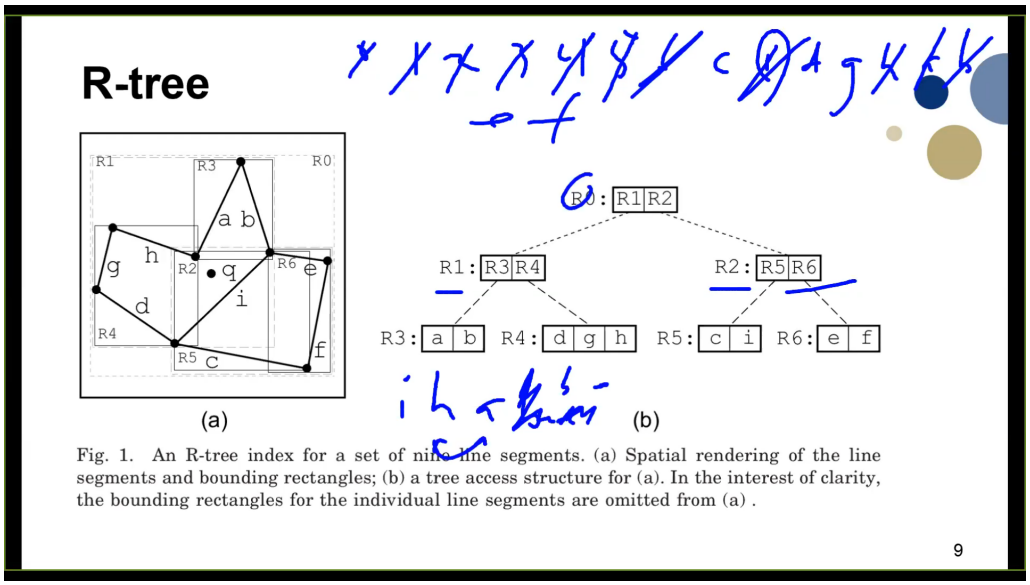

Fig. 1. An R-tree index for a set of nine line segments. (a) Spatial rendering of the line segments and bounding rectangles; (b) a tree access structure for (a). In the interest of clarity, the bounding rectangles for the individual line segments are omitted from (a) .

Figure 2: Algorithm

Notations: Pop up the bounding box and then get the actual shape of it.

Worst case scenario: Searching for a central point in a circle Optimization: **Comparable distance**
• A proxy for actual distance
• dist(A) > dist(B) <−> comp_dist(A) > comp_dist(B)
• Should be faster than actual distance
• E.g., Pythagoras without the square root;; Make tree algorithm cheaper, if square big, distance grater than others.

Priority queue size: increases with the n of elements in the DB, INN scales better

Planning for kNN
• How do we recognise the query?
– ORDER BY
– LIMIT
– GROUP BY? HAVING? -> ordering groups, not individual elements, we cant do it as distance scan on index, so this scan is harder
– WHERE? increase the number of elements to be on K (limit), it does not block us to do this optimiaztion but it will change K
• How do we combine distance scan with other optimizations?
Ignore any query that contains GROUP BY