Innovative Assignment 1

Gahan Saraiya (18MCEC10), Rushi Trivedi (18MCEC08), Raj Kothari (18MCEC07)

18mcec10@nirmauni.ac.in, 18mcec08@nirmauni.ac.in, 18mcec07@nirmauni.ac.in

I. Introduction

Aim of this assignment is to produce feature matrix for various multidimensional indexes.

II. Feature Matrix

The merits and demerits of below listed indexes are compared:

- Hash Based
  - Grid File
  - Partitioned Hash
- Tree Based
  - Multi-key
  - kd-Tree
  - Quad Tree
  - R Tree

I. Need for Index

All the queries on the database are not always simpler and as a growing size of data and requirements of analytics on it various type of queries needs to be executed. Time taken by such queries to execute grows with the size of data and hence it is necessity to optimize it.

To optimize such query first we need to determine trade off between various indexes which are used in modern database and conclude a feature matrix describing which index is suitable for which type of query.

III. Spatial search problems

Spatial data has two primary query types:

- nearest neighbors
- range queries

I. K-Nearest Neighbor

Problem Statement: Given millions of points, such as locations of city or area, how do we determine the closest points to a given query point?

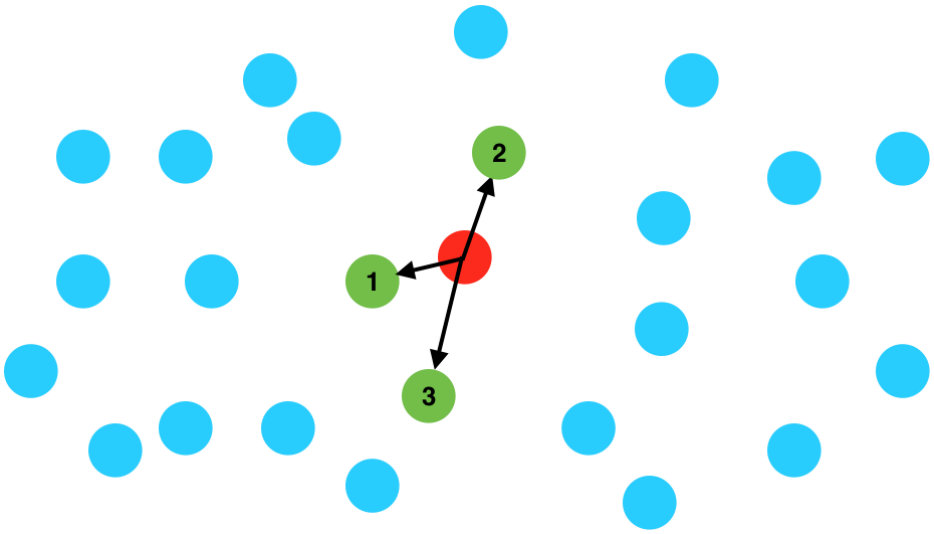

Figure 1: three nearest neighbors (points marked with green) of point (marked with red)

II. Range and radius queries

Problem Statement: How do we retrieve all points inside a rectangle (range query) or circle (radius query) or any polygon.

III. Solution - spatial trees

- Data changes are usually much less frequent than queries, so incurring an initial cost of processing data into an index is a fair price to pay for instant searches afterwards.
- Almost all spatial data structures share the same principle to enable efficient search: branch and bound. It means arranging data in a tree-like structure that allows discarding branches at once if they do not fit our search criteria.

IV. R-Tree

IV.1 Trade off

✓ Quadtrees require fine-tuning by choosing appropriate tiling level in order to optimize performance. No specific tuning is required for R-Trees.

✓ It's used by all modern spatial databases and many game engines.

✓ R-trees are much faster than Quadtree for nearest neighbours queries.

✓ R-trees are much faster than Quadtree for nearest neighbours queries.

✗ Quadtree indexes are created faster than R-tree.

✗ Quadtree can be implemented on top of existing B-tree. R-Tree -cannot

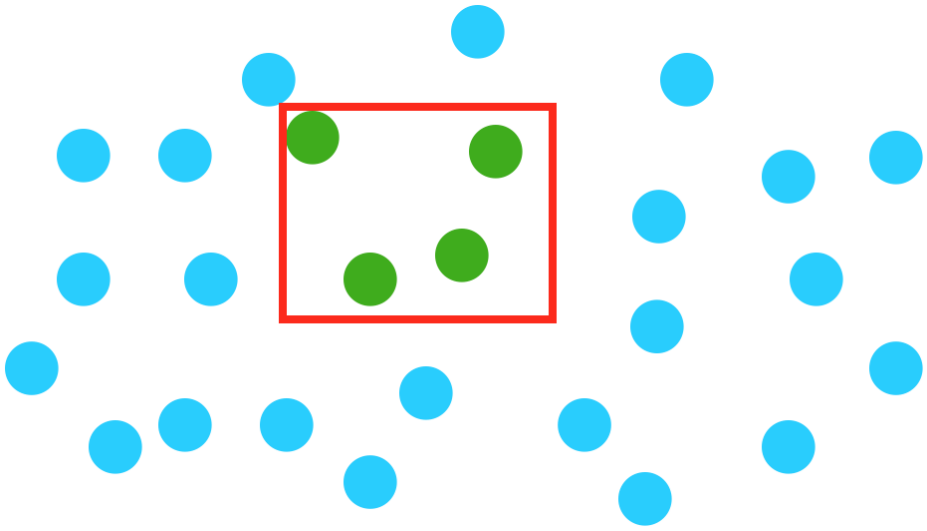

Figure 2: three nearest neighbors (points marked with green) of point (marked with red)

V. kd tree

kd tree is similar to R-tree, but instead of sorting the points into several boxes at each tree level. We sort them into two halves (around a median point) either left and right, or top and bottom, alternating between x and y split on each level

✗ Compared to R-tree, K-d tree can usually only contain points (not rectangles), and doesn't handle adding and removing points.

✓ It's much easier to implement, and it's very fast.

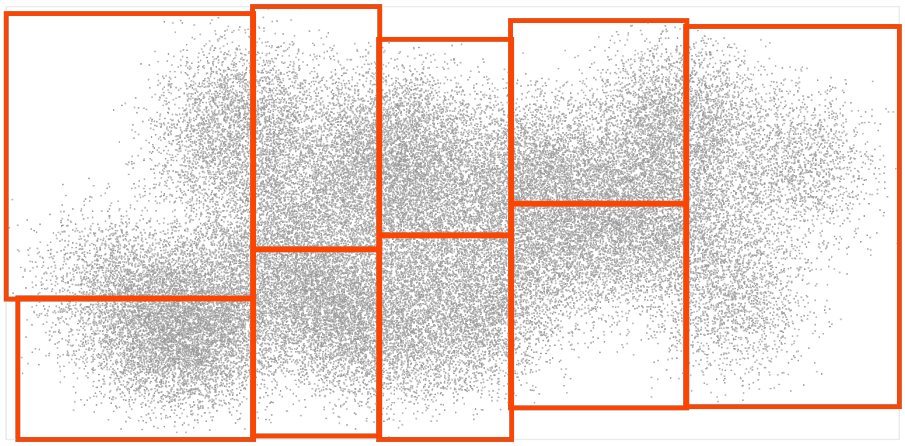

Figure 3: sorted bunch of points in to rectangular boxes

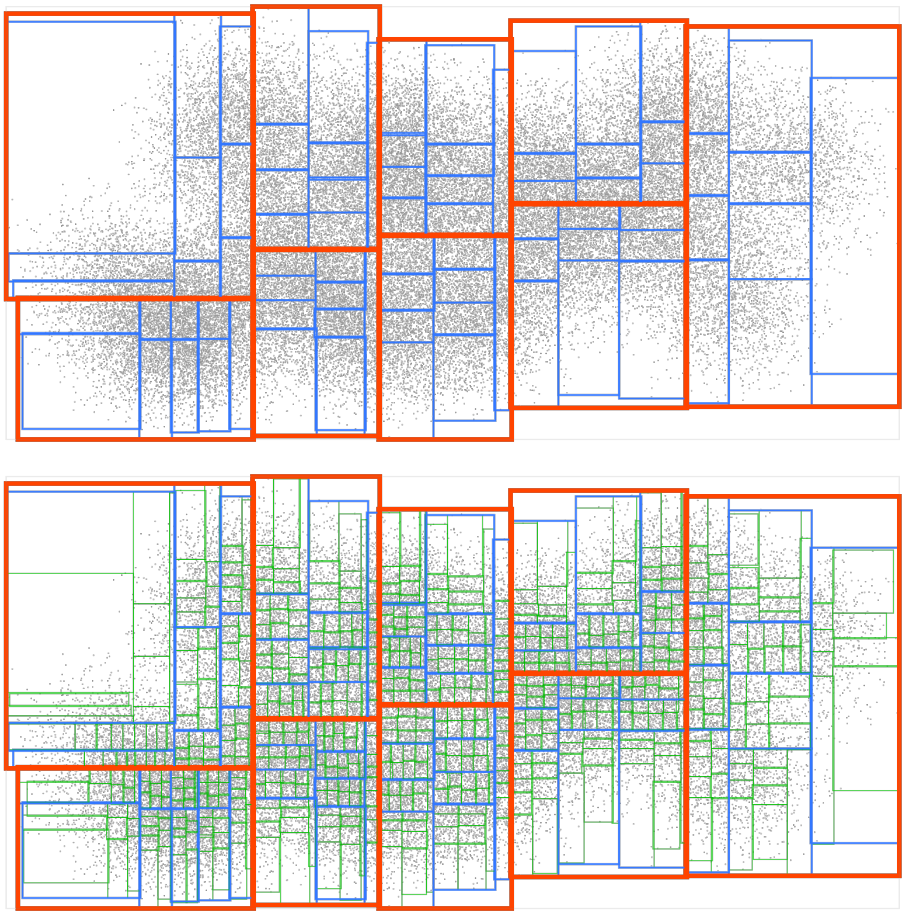

Figure 4: further sorted bunch of points in to rectangular boxes from each rectangular box

Figure 5: R

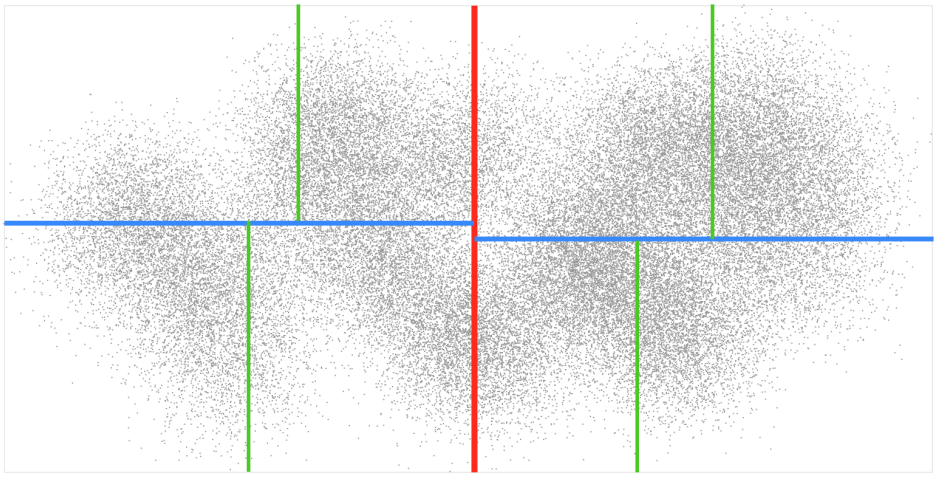

Figure 6: first three levels of kd tree

Figure 7: R

Table 1: Feature Matrix for Multidimensional Indexed

Query Type	**Hash Based**		**Tree Based**			
	Grid	Partitioned Hash	MultiKey	kd-Tree	Quad Tree	R Tree
Exact Match	✔	✔	✔	✔	✔	Reasonable
Partial Match	✔	✔	works only for first key	✔	✔	✔
Range	✔	✗	✗	✔	✔	✔
Nearest Neighbour	✔	✗	✗	Reasonable	✔	Reasonable
Where am I	N/A	N/A	N/A	N/A	N/A	✔
Balanced Tree	N/A	N/A	✔	✗	✗	✔
# of empty nodes or buckets	High (if large data file)	–	–	–	High [Sol: keep only Not-NULL pointer only]	N/A
Splitting	Easy	Hard	N/A	N/A	N/A	N/A
Splitting Point	Distribute Data	N/A	N/A	any point that distribute data	centre point always	N/A
uniformly distributed data with a high change frequency					✔	✗