HILBERT

R-TREE

AN IMPROVED R-TREE USING FRACTALS

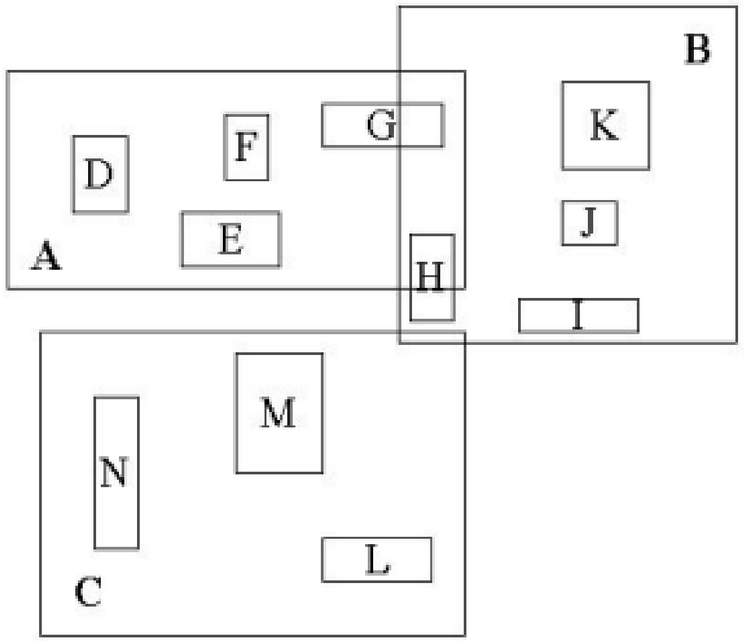
CONTENTS

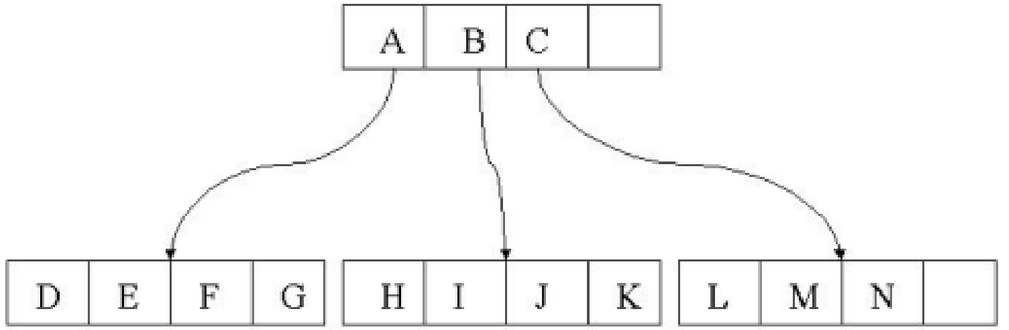
R-TREE

An R-tree is a data structure used for indexing spatial data, such as points or regions, in a two-dimensional space

The nodes in an R-tree are organized using the minimum bounding rectangle (MBR) of their child nodes. The MBR is the smallest rectangle that contains all the data points in the node.

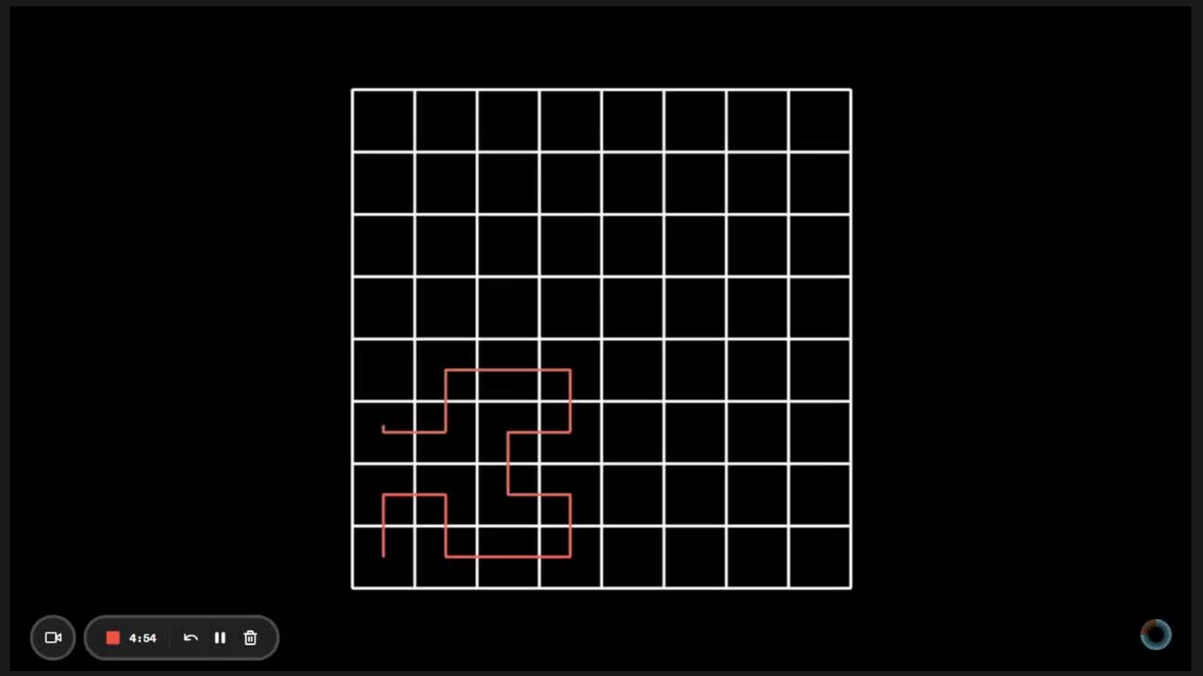
The nodes are organized in a tree structure where the root node contains all the data points, and each child node represents a subset of the data points in its parent node.





HILBERT R-TREE

A Hilbert R-tree is a spatial index structure that uses a space-filling curve called the Hilbert curve to impose a linear ordering on the data rectangles



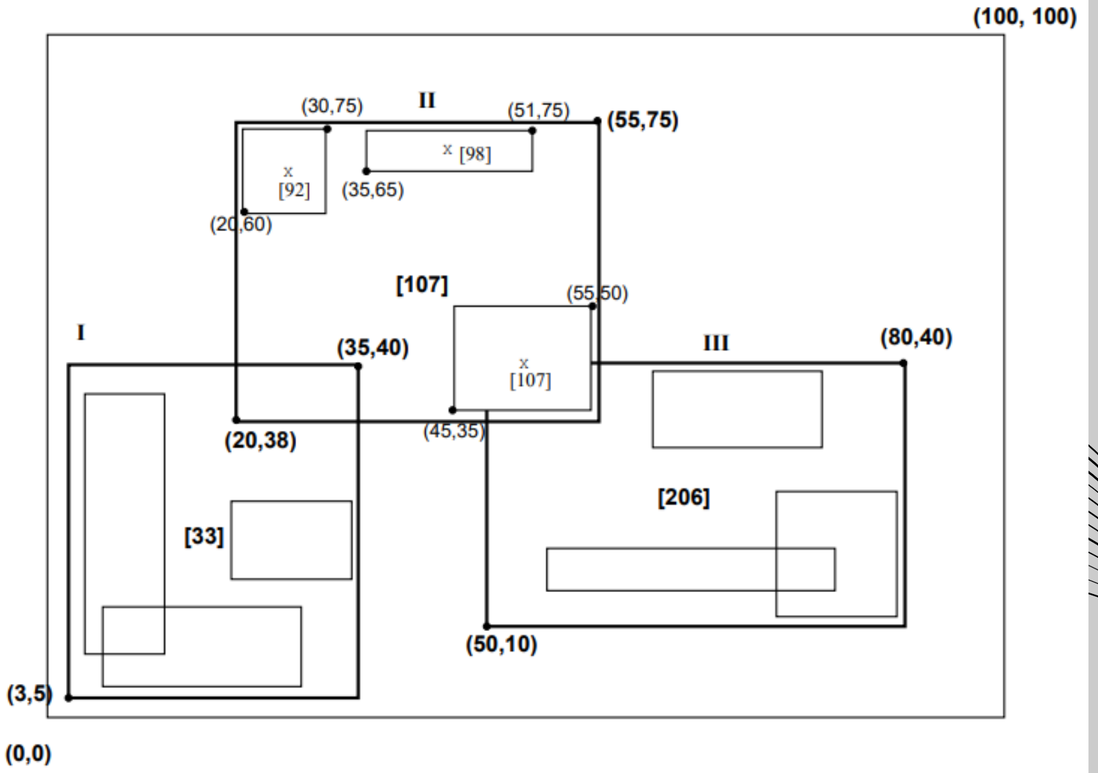
By mapping the 2D space onto a 1D space using the Hilbert curve, the Hilbert R-tree can perform range queries in the multidimensional space as if they were one-dimensional queries, using the linear ordering of the Hilbert values.

This is achieved by dividing the Hilbert curve into segments and clustering data rectangles that fall within the same segment.

STRUCTURE OF A HILBERT R-TREE

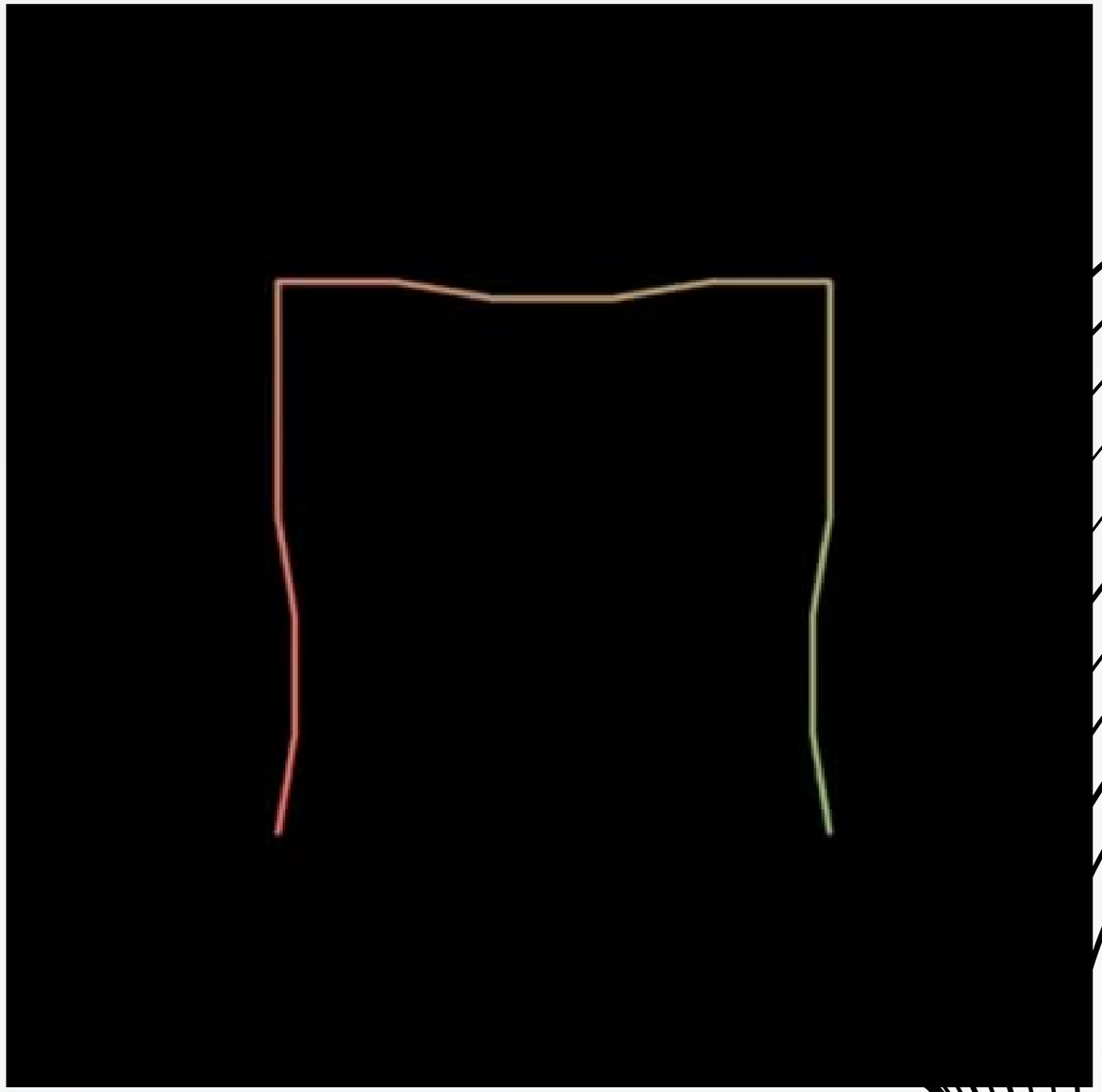
The Hilbert R-tree is organized as a hierarchical tree structure, where each node in the tree represents a bounding rectangle that contains a set of child nodes or data rectangles.

The root node represents the entire data set, and each leaf node represents a single data rectangle



Hilbert Node Structure - It stores a pointer to the parent node, a boolean value is_leaf, the address of a point if is_leaf is true, rectangle which defines the MBR for that node, number of children nodes, LHV among the children, and an array which stores the pointers to the children nodes.

Hilbert R Tree Structure - It stores the pointer to a root node and the order of the hilbert curve used to construct the tree



HILBERT CURVE

Hilbert Curve achieves the best clustering among all methods

COMPARISON BETWEEN R-TREES & HILBERT R-TREES

R- TREES

- R-trees use rectangles to group data points together. Each rectangle is called a "minimum bounding rectangle" (MBR)
- The R-tree is organized like a tree structure, with nodes representing groups of rectangles that contain data points.

HILBERT R-TREES

- Hilbert R-trees use a special curve called the Hilbert curve to group data points together.
- Hilbert R-trees divide the space into squares and assign each data point to its corresponding square.
- More efficient for searching spatial data that are clustered together.

SEARCHING ALGORITHM

- The hilbert_rtree_search function takes two arguments: a pointer to the root node of a Hilbert R-tree and a point object p.
- If the root node is NULL, the function returns NULL.
- If the root node is not NULL, the function checks if it is a leaf node.

- If the root is not a leaf node, the function uses the hilbert_cmp function to traverse the tree.
- The given function takes two pointers to "point" objects and returns -1, 0, or 1 depending on the hilbert curve order of the first point and the second point.

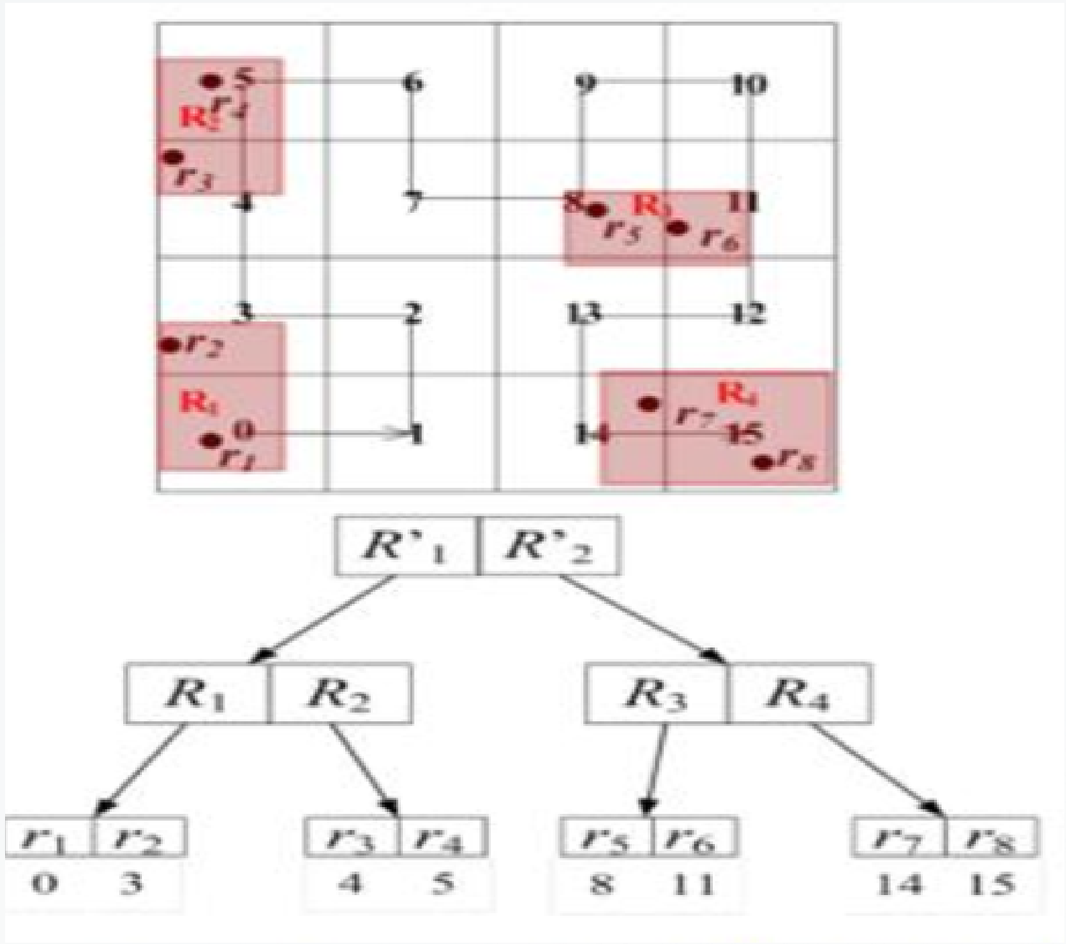
- If the root node is a leaf node, the function iterates over the **point** objects stored in the node and checks if any of them match **p**.
- If a match is found, the function returns a pointer to the leaf node.
- If no match is found, the function returns NULL.

- It returns -1, 0, 1 given the hilbert curve value of the first point is less than, equal to, or greater than the second point.
- The function iterates over the entries until it finds the index **i** of the child node whose entry has a LHV greater than or equal to **p**.

SEARCHING ALGORITHM

- The child node corresponding to index **i** is the smallest child node that can contain **p** in its subtree.
- Then we recursively call **hilbert_rtree_search** on the child node , by passing the pointer to the child node and the same point p.

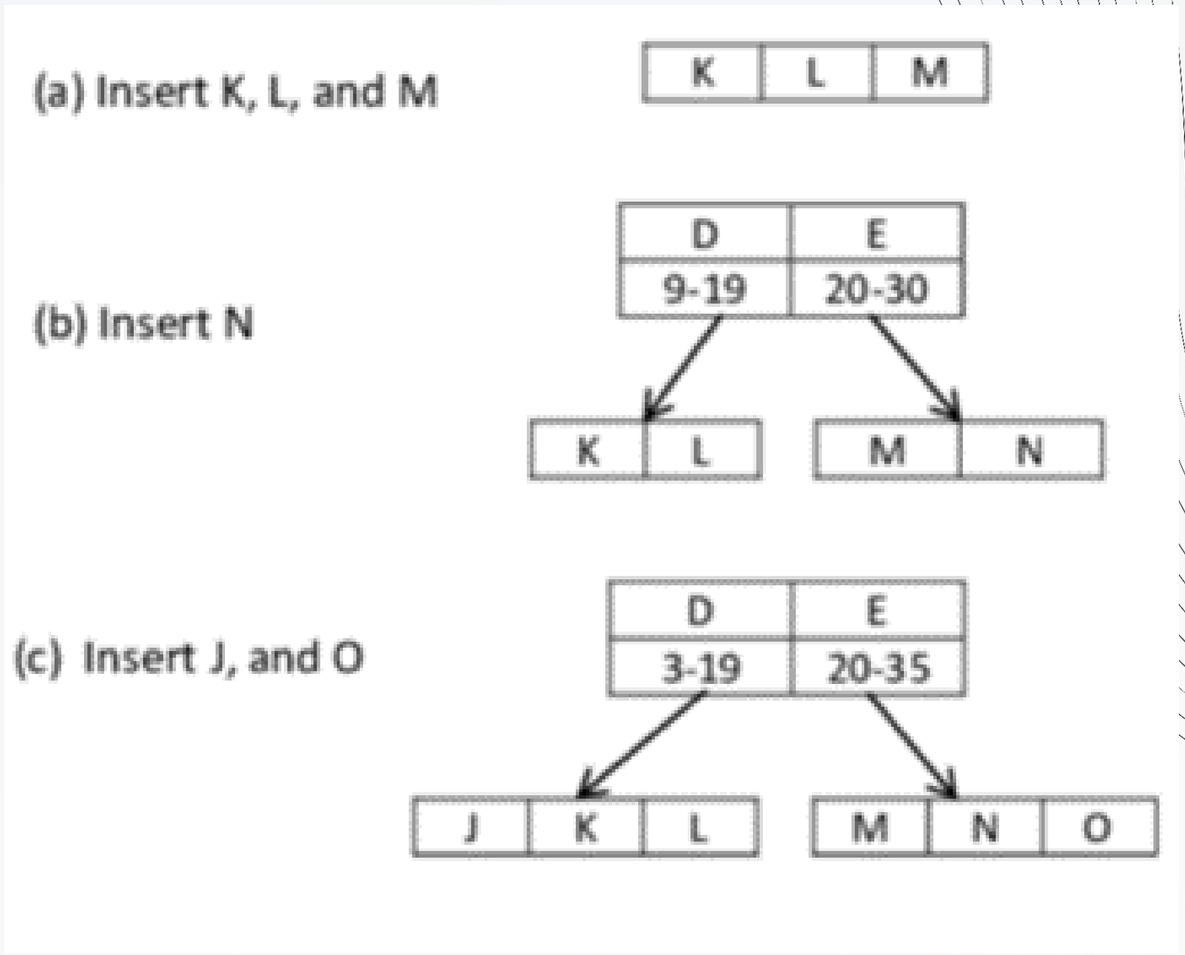
- The result is stored in a pointer variable **result**.
- If **result** is not NULL, the function immediately returns **result.**
- If **result** is NULL, it means that the point was not found in this subtree, then the function iterates over the remaining entries.



INSERTION

- The code begins by including the header file "insert.h" which contains the declarations for the data types and functions used in the code.
- The function **insert_into_leaf** takes a leaf node and a point as input and inserts the point into the leaf node's entry list. The entries in the list are sorted using the **hilbert_cmp** function which compares the Hilbert values of two points.

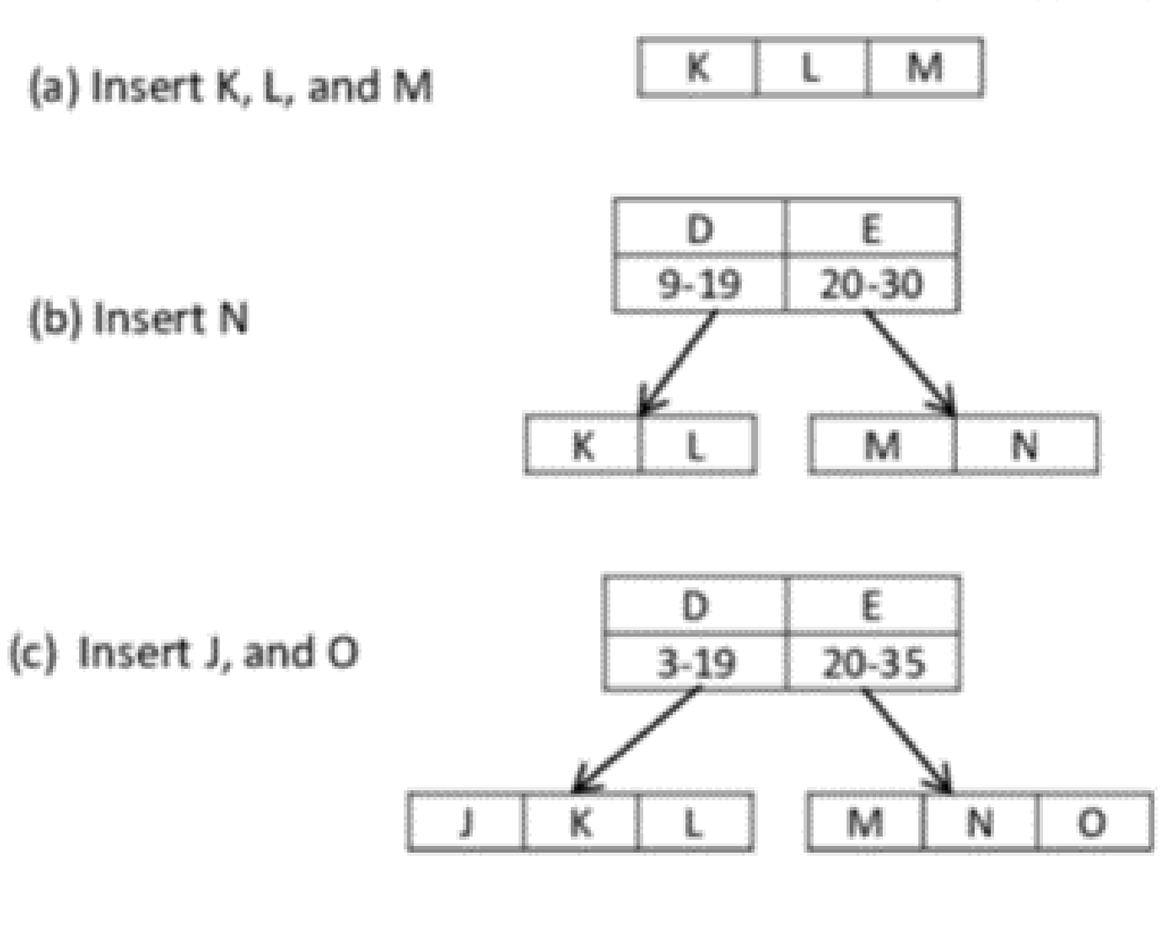
INSERTION EXAMPLE:



INSERTION

- The function **choose_node** takes the root node of the tree and a point as input and traverses the tree to find the leaf node that the point should be inserted into. At each non-leaf node encountered during traversal, the child node with the closest entry to the input point is chosen.

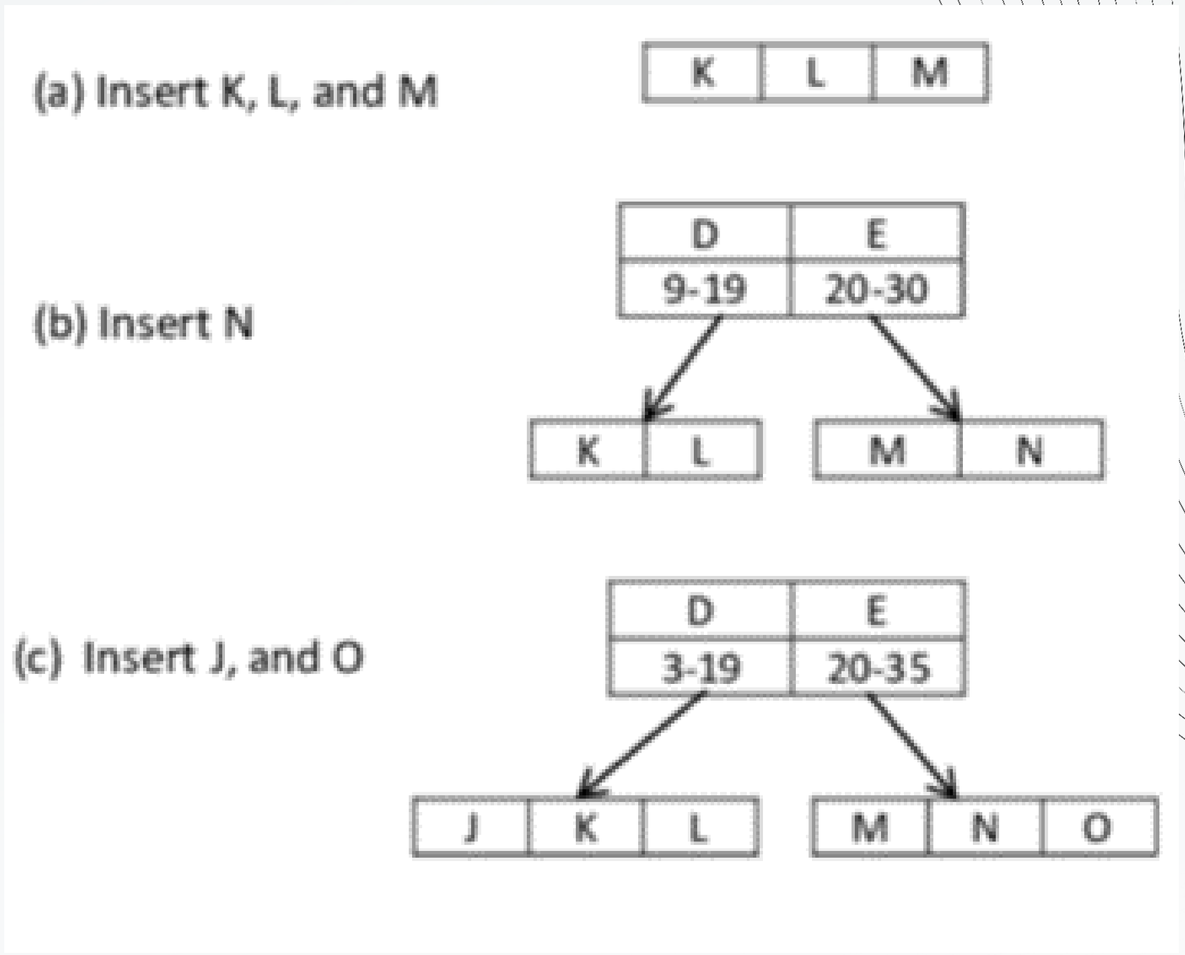
INSERTION EXAMPLE:



INSERTION

- The function **insert** is the main entry point for inserting a point into the tree. It takes a pointer to the root node and a point as input. If the root node is null, it creates a new root node and inserts the point into it. If the root node is a leaf node and has room for the new point, the point is inserted into the leaf node. Otherwise, the leaf node is split into two, and the input point is inserted into one of the two leaf nodes. If the leaf node that was split was the root node, a new root node is created, and the two leaf nodes become its children. Otherwise, the insertion process continues recursively from the node where the split occurred.

INSERTION EXAMPLE:



MBR

This function mbr_union() is used to compute the minimum bounding rectangle (MBR) that encloses two given MBRs in a 2-dimensional space. The function takes two MBRs a and b as input and returns a new MBR that encloses both the input MBRs.

It first compares the x and y coordinates of the bottom-left and top-right corners of the two MBRs separately, and assigns the minimum value to the bottom-left corner and the maximum value to the top-right corner of the resulting MBR.

mbr_union()

This function updates the MBR (minimum bounding rectangle) of a given node in a tree structure. It initializes the MBR of the node using the first entry in the node. Then it iterates through the remaining entries in the node, and adjusts the MBR values based on the location of each entry's x and y coordinates. The resulting MBR represents the smallest rectangle that encompasses all the entries in the node.

update_mbr(node *n)

This is a recursive function that updates the MBR of each node in the tree. If the current node is a leaf, it simply updates its MBR using the update_mbr() function. If the current node is an internal node, it recursively updates the MBR of its children, and then computes its own MBR by taking the union of its children's MBRs using the mbr_union() function. The function iterates over all the children of the node and updates their MBRs recursively. The updated MBRs are then used to update the MBR of the current node.

update_mbr_recursive(node *n)

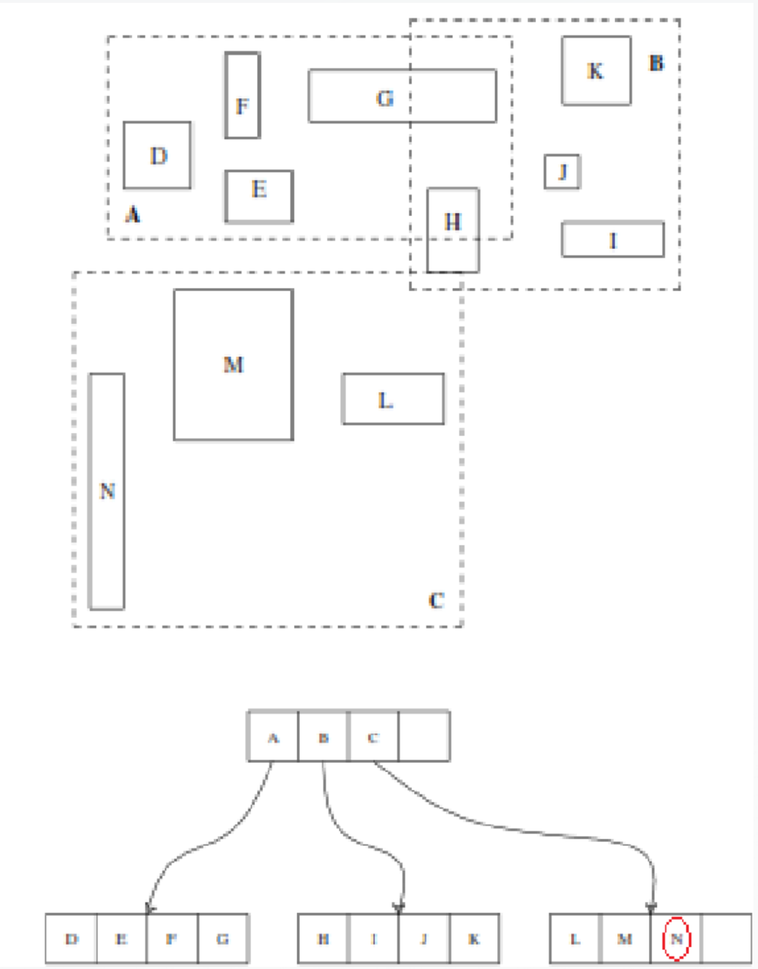
OVERFLOW HANDELING ALGORITHM

- The function **split_leaf** takes a leaf node, a pointer to a new leaf node, and a point as input. It splits the leaf node into two by creating a new leaf node and moving half of the entries from the original leaf node to the new leaf node. The input point is then inserted into one of the two leaf nodes, depending on its Hilbert value.

PRE-ORDER ALGORITHM

- The **print_preorder** function takes as input a pointer to a node in the Hilbert R-tree and prints information about the node and its children in preorder traversal order.
- The function first checks if the input node is NULL.

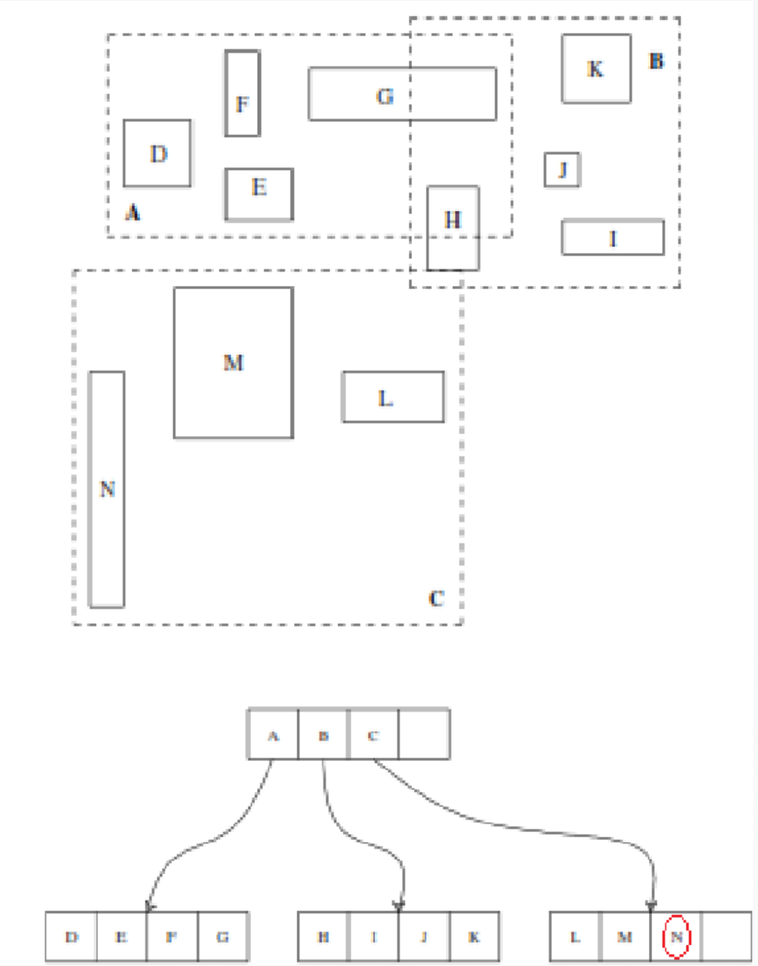
- If it is, the function simply returns without doing anything.
- If the node is not NULL, the function first prints the number of entries in the node, whether the node is a leaf or not, and the minimum bounding rectangles (MBRs) of the entries in the node.



PRE-ORDER ALGORITHM

- The function then iterates over the entries in the node. For each entry, if the node is a leaf, the function prints the coordinates of the entry's point.
- If the node is not a leaf, the function recursively calls it on the child node.

- After iterating over the entries in the node, if the node is not a leaf, the function recursively calls itself on the child node corresponding to the MBR of all entries in the node.
- The output of the function provides information about the number of entries in each node, the coordinates of each entry, and the MBRs of each node.



CONCLUSION

Hilbert R- Tree outperforms all other R-tree methods for the following reasons-

- It introduces good ordering among rectangles.
- Similar rectangles are grouped together, thus the R-tree has nodes with small MBR leading to faster response times
- Better packing of rectangles results in a shallower tree with a bigger fanout.