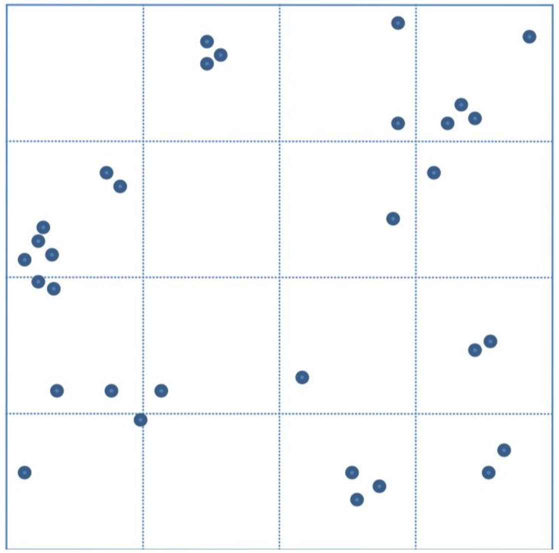
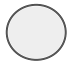
Quadtree

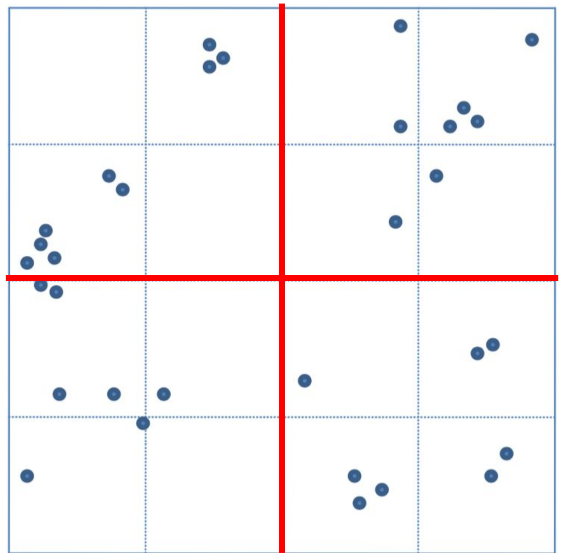
Split each partition that contains more objects than the page limit into four equal quadrants

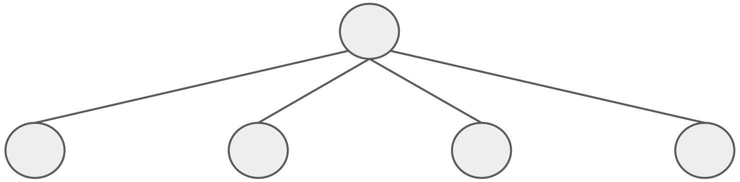
Quadtree (page size 2)



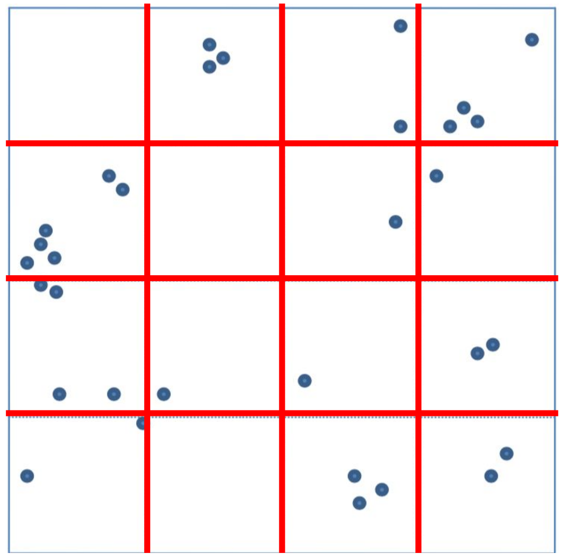


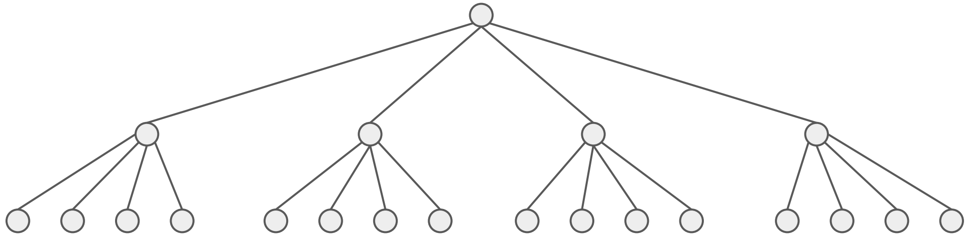
Quadtree (page size 2)
1st Split



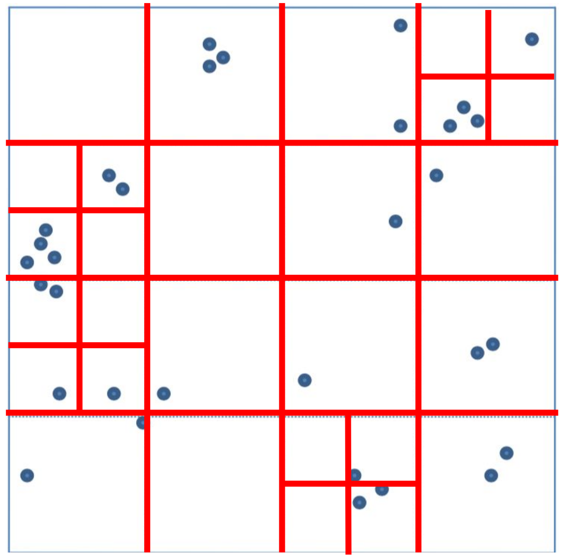


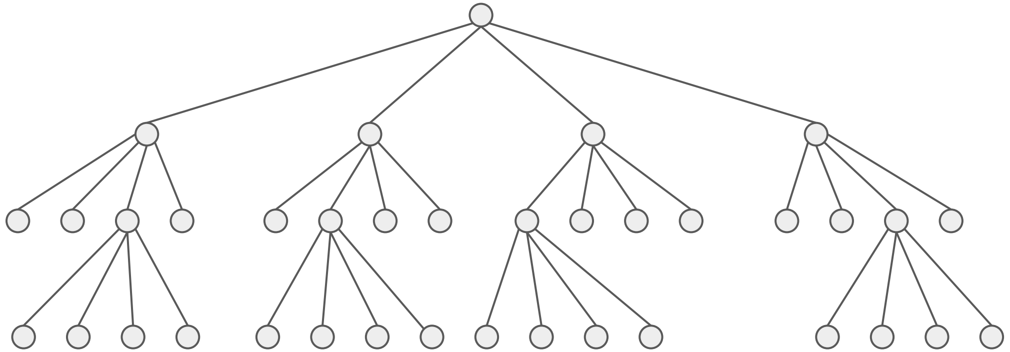
Quadtree (page size 2)
2nd Split



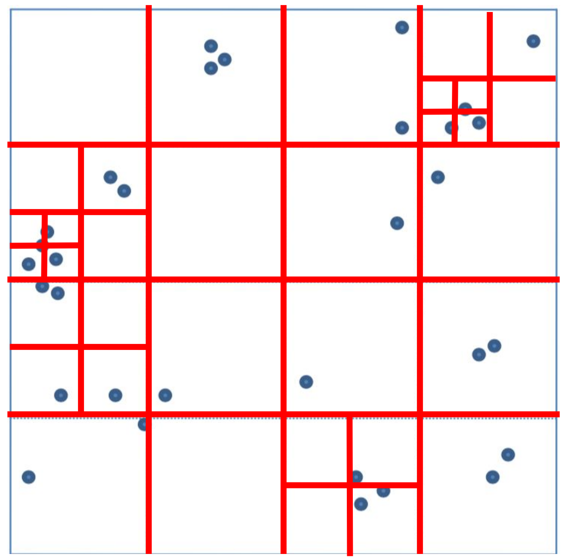


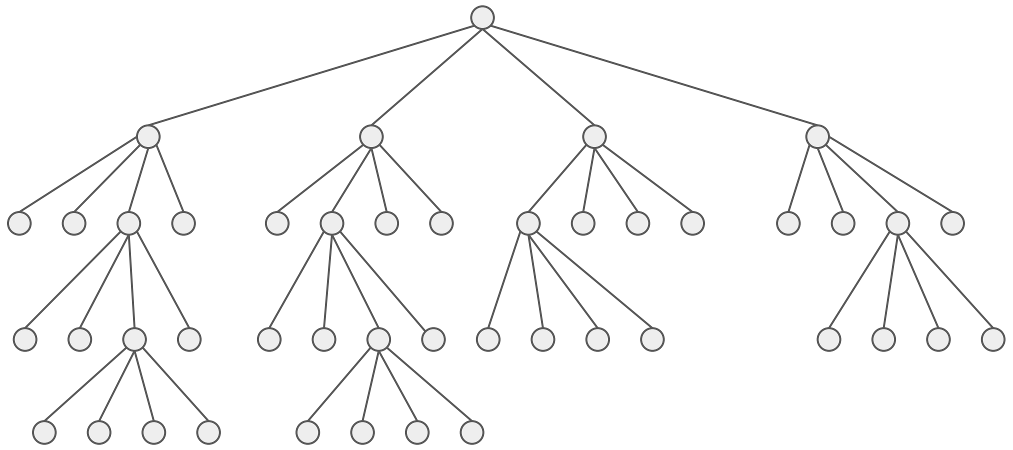
Quadtree (page size 2)
3rd Split





Quadtree (page size 2)
4th Split





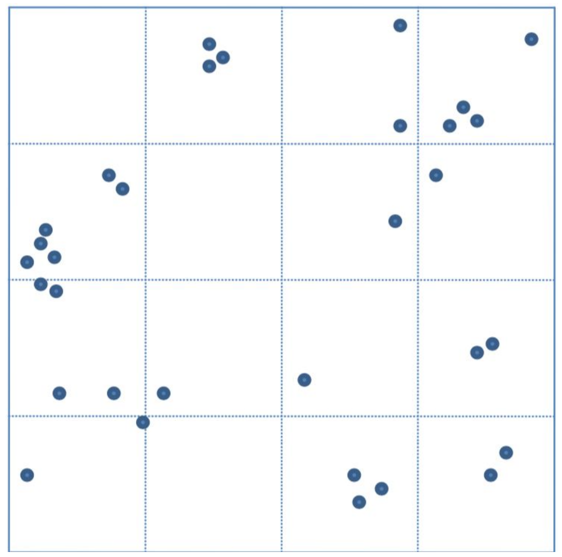
Each partition contains at most two objects → stopping criteria is reached

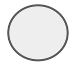
Quadtrees are unbalanced and each node has exactly four children

kD-Tree

Split each partition that contains more objects than the page limit into two partitions along the corresponding axis. The splitting hyperplane is chosen wrt the data distribution.

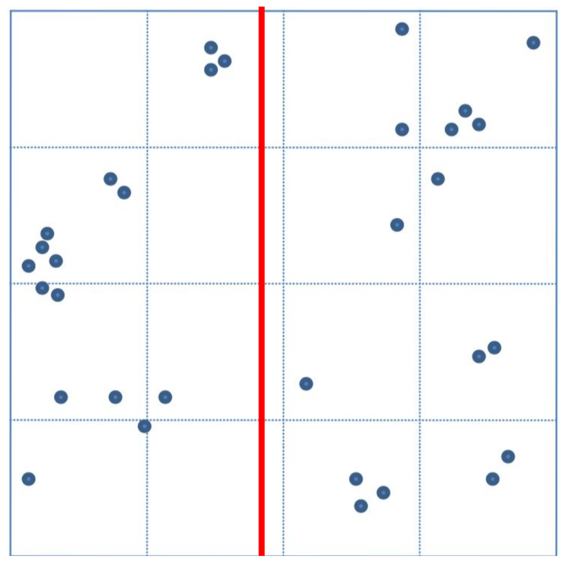
kD-Tree (page size 4)

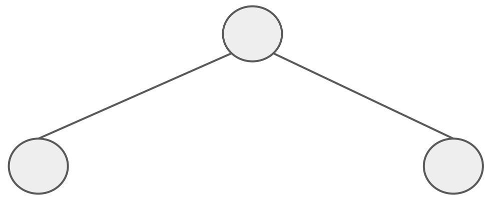




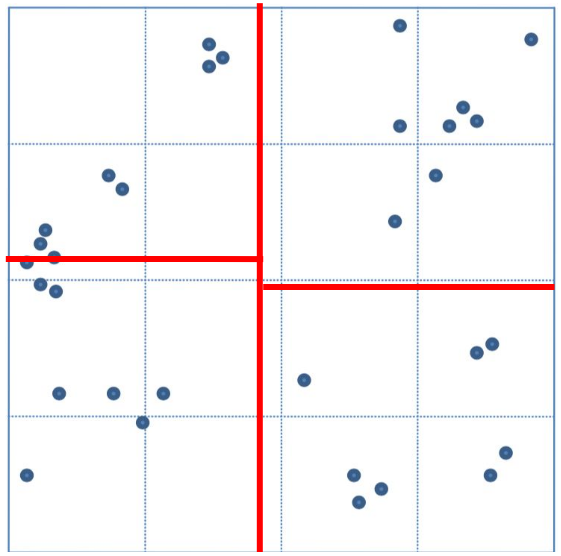
kD-Tree (page size 4)

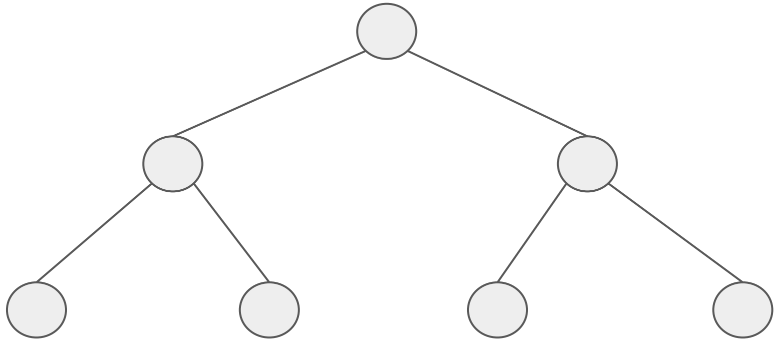
1st Split (along the x-axis)



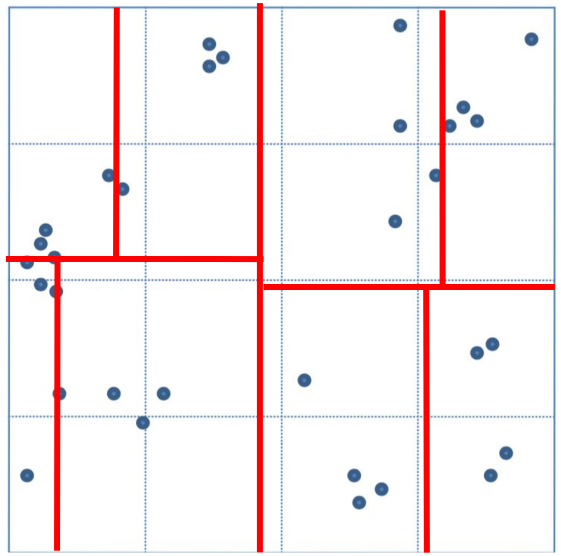


kD-Tree (page size 4)
2nd Split (along the y-axis)

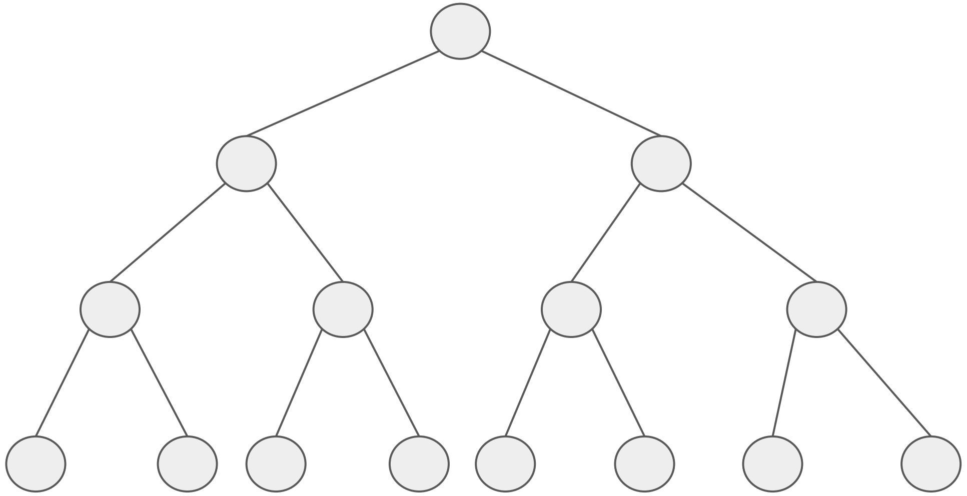




kD-Tree (page size 4)
3rd Split (along the x-axis)



Each partition contains at most four objects → stopping criteria is reached

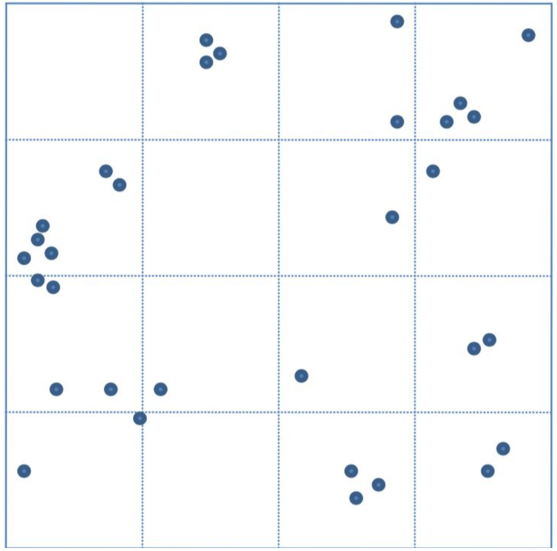


kD-Trees are balanced binary space partitioning trees

R-Tree

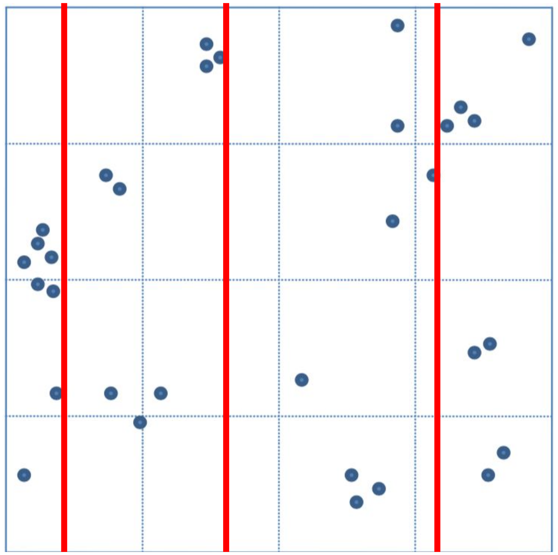
Aggregate objects by approximating them with minimum bounding rectangles (MBRs). Unlike the Quadtree or the kD-Tree, the R-Tree is built bottom-up (instead of top-down).

R-Tree (page size 2)



I omit drawing the tree structure here. Due to having a page size of 2, this would correspond to building a binary tree bottom up, i.e., starting with 32 objects on the leaf-node level, the next level would contain 16 nodes (since we aggregate two objects within one MBR), then the next level would have 8 nodes (again, we approximate 2 MBR objects by one larger MBR) and so on.

R-Tree (page size 2)
1st Iteration: number of objects n = 32, page limit M = 2



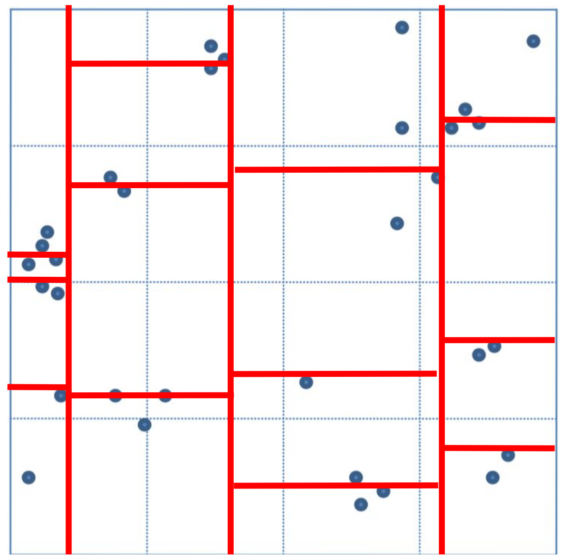
Using the sort-tile algorithm, we first need to determine how many partitions (resp. quantiles) we need per dimension:

$$q = \lceil \sqrt{\frac{n}{M}} \rceil = \lceil \sqrt{\frac{32}{2}} \rceil = 4$$

Now, that we know how many partitions we need, we can derive the maximum number of objects that should be in one partition (for the splits along the x-axis):

$$m = q \cdot M = 4 \cdot 2 = 8$$

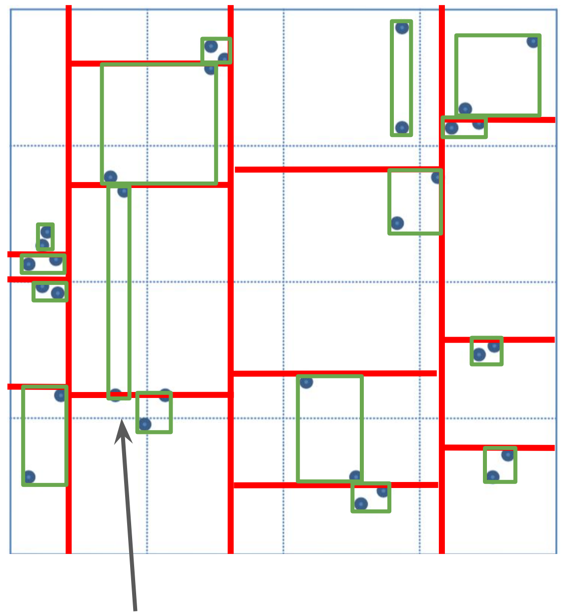
R-Tree (page size 2)
1st Iteration: number of objects n = 32, page limit M = 2



Next we need to partition along the y-axis, and due to the page limit being 2, we ensure that each partition finally contains 2 elements.

Note: The last partition could possibly contain less than two objects

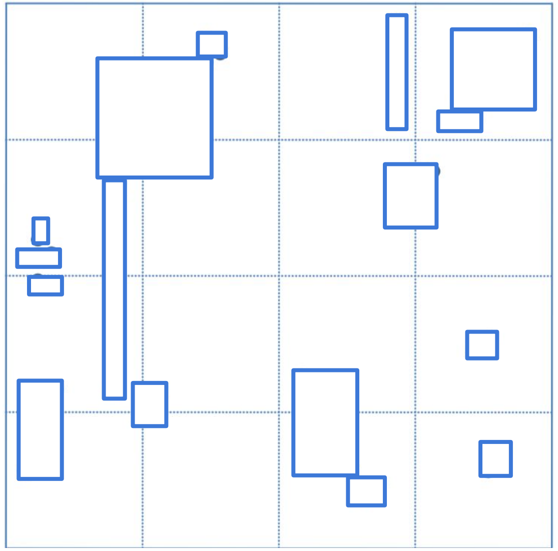
R-Tree (page size 2)
1st Iteration: number of objects n = 32, page limit M = 2



Given our partitioning, we can derive the minimum bounding rectangles that approximate the data objects within the partitions.

Note: Here, we have the situation where two data objects have the same y-value. If the algorithm hasn’t assigned data objects to partitions ultimately, we can do this assignment at the step where we create the approximations (resp. MBRs). This is usually done by following certain heuristics, e.g., minimizing the empty space covered by MBRs.

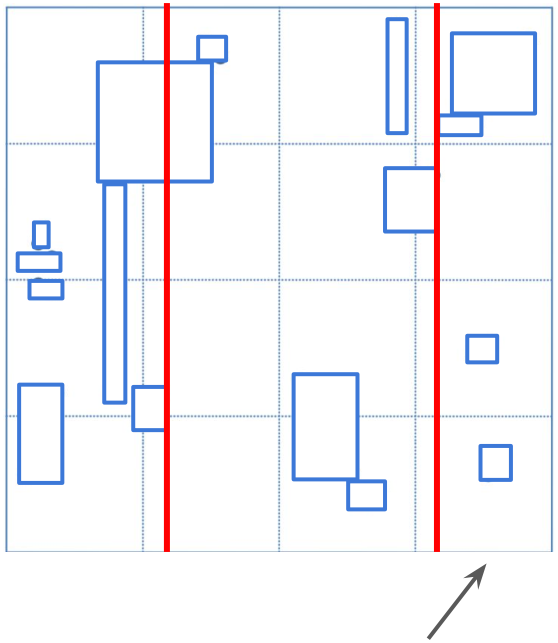
R-Tree (page size 2)
1st Iteration: number of objects n = 32, page limit M = 2



At the end of the first iteration we finally end up with a database consisting of 16 MBR objects.

In terms of tree structure, we are one level above the leaf node level now.

R-Tree (page size 2)
2nd Iteration: number of objects n = 16, page limit M = 2



We only have 4 objects in this partition, but this is fine

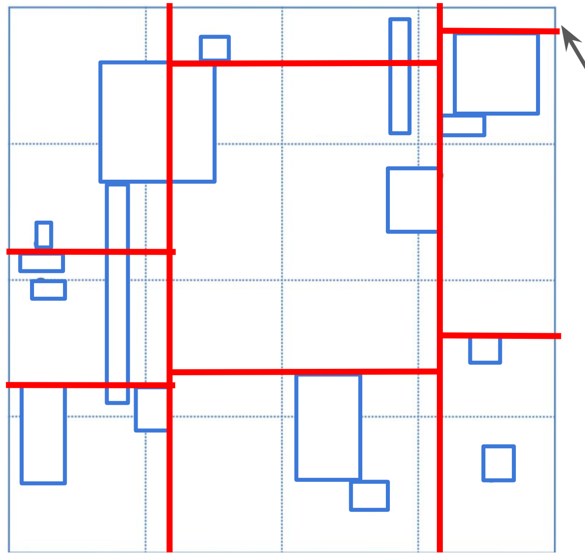
At the beginning of iteration 2, we have to compute the number of partitions per dimension and the number of objects per partition again:

$$q = \lceil \sqrt{\frac{n}{M}} \rceil = \lceil \sqrt{\frac{16}{2}} \rceil = 3$$

$$m = q \cdot M = 3 \cdot 2 = 6$$

Having these, we again start partitioning along the x-axis. Note that we use the upper right corner of each MBR to determine into which partition an MBR falls.

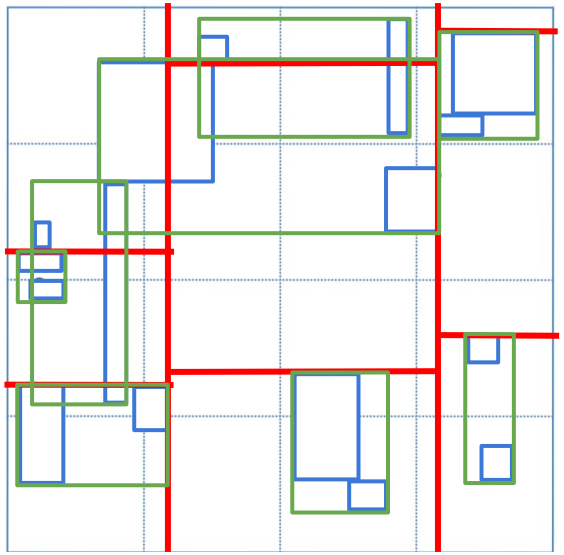
R-Tree (page size 2)
2nd Iteration: number of objects n = 16, page limit M = 2



Next, we split along the y-axis and make sure that we have two objects (bc M=2) in each partition.

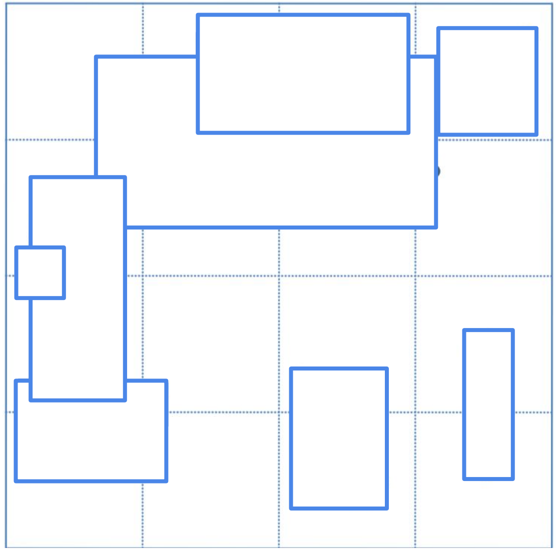
In this case, we have zero objects in the last partition

R-Tree (page size 2)
2nd Iteration: number of objects n = 16, page limit M = 2



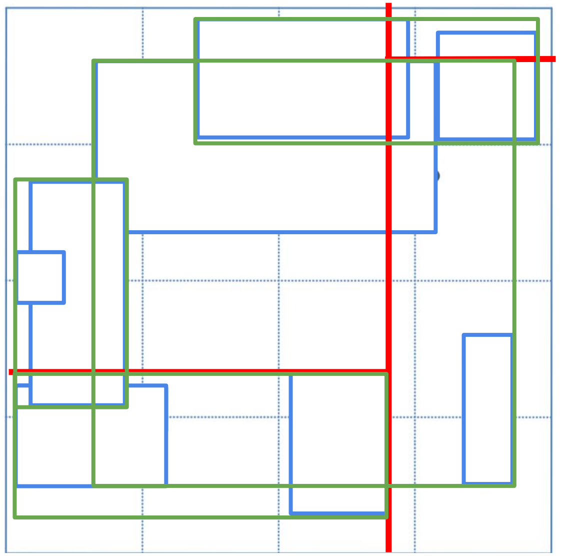
We approximate the objects partition-wise by MBRs...

R-Tree (page size 2)
2nd Iteration: number of objects n = 16, page limit M = 2



… and finally end up with a total number of 8 MBR objects at the end of the second iteration.

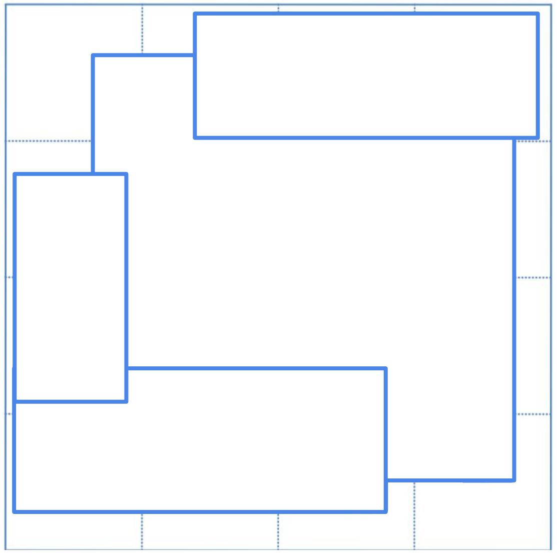
R-Tree (page size 2)
3rd Iteration: number of objects n = 8, page limit M = 2



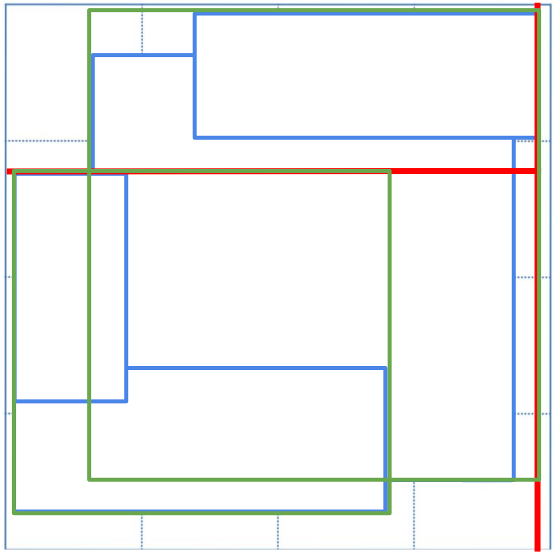
$$q = \lceil\sqrt{\frac{n}{M}}\rceil = \lceil\sqrt{\frac{8}{2}}\rceil = 2$$

$$m = q \cdot M = 2 \cdot 2 = 4$$

After doing the same steps as in the iterations before



R-Tree (page size 2)
4th Iteration: number of objects n = 4, page limit M = 2

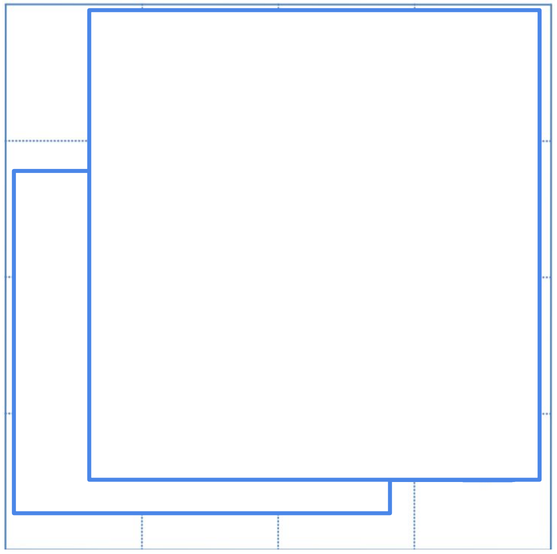


$$q = \lceil\sqrt{\frac{n}{M}}\rceil = \lceil\sqrt{\frac{4}{2}}\rceil = 2$$

$$m = q \cdot M = 2 \cdot 2 = 4$$

$$\xrightarrow{\quad}$$

After doing the same steps as in the iterations before



If we re-evaluate the quantile value (which can either be done at the end of each iteration or the beginning of a new iteration), we get $q = \lceil\sqrt{\frac{n}{M}}\rceil = \lceil\sqrt{\frac{2}{2}}\rceil = 1$. Now, q < 2 and this is the stopping criteria, i.e., we reached the root node.