A LaTeX'ed solution to 2D Spatial Indexing.

Mitchell Scott (mtscot4)

October 9, 2024

2D Spatial Indexing Examples

Assignment 4-1: Quad Trees

Assignment 4-2: kD-Trees

Assignment 4-4: R Trees

Table of Contents

Assignment 4-1: Quad Trees

Assignment 4-2: kD-Trees

Assignment 4-4: R Trees

Quad-Tree Set-up

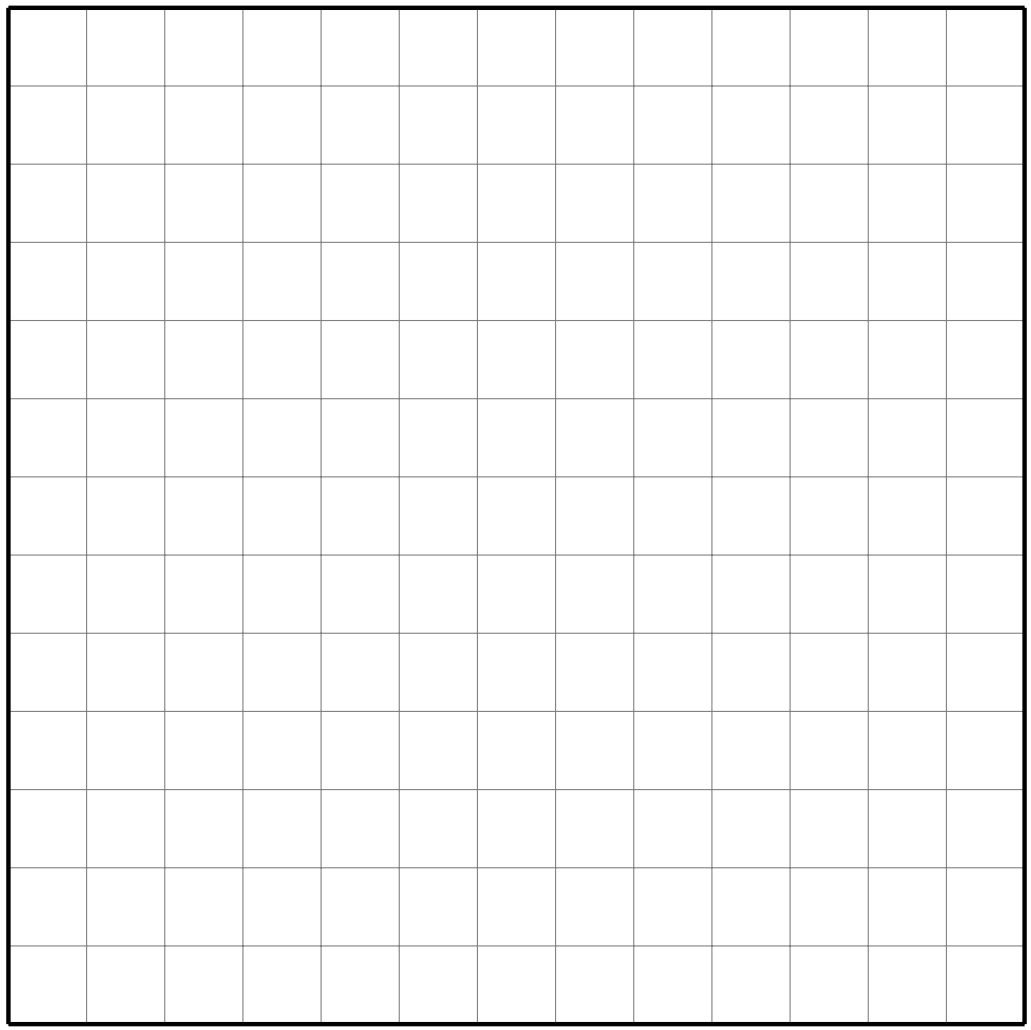
Insert the following points (in this order) into an initially empy Quad-Tree:

$$A = (0, 0), B = (10, 10), C = (8, 2), D = (9, 3), E = (2, 2),$$
$$F = (6, 2), G = (2, 10), H = (7, 3), I = (5, 5), J = (7, 4)$$

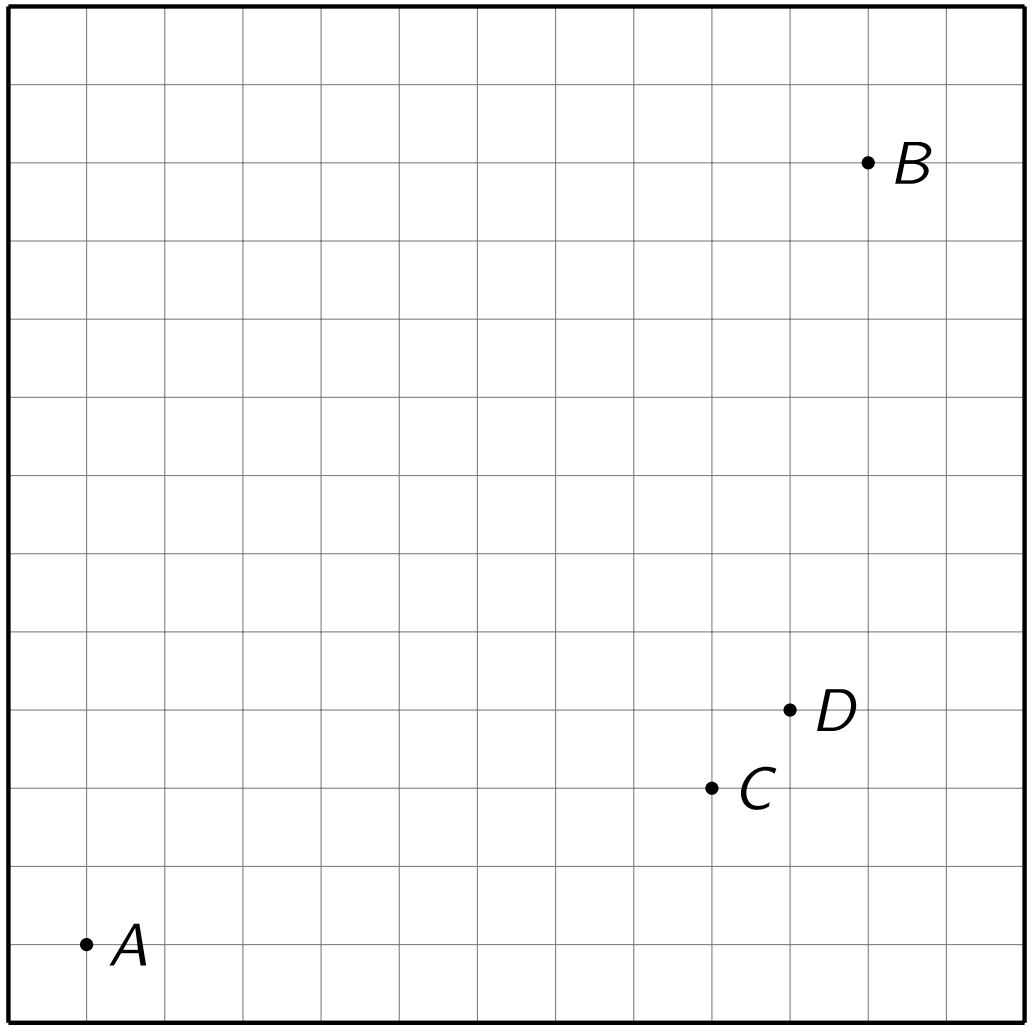
The maximum page capacity of this Quad-Tree is 4.

- Draw the current Quad-tree after each split.

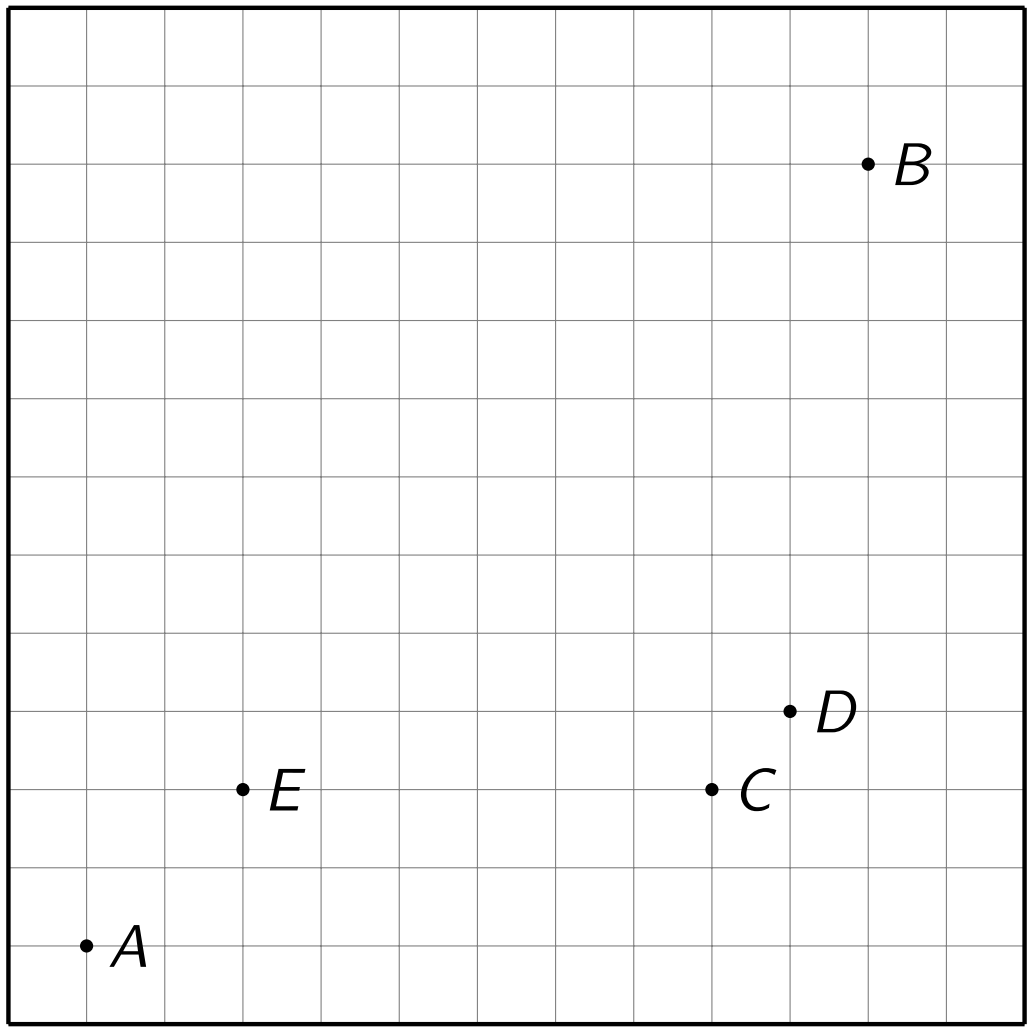
The initially empty Quad-tree has one entry that covers the entire data space, by definition. We decided to make it $[-1, 12] \times [-1, 12]$.



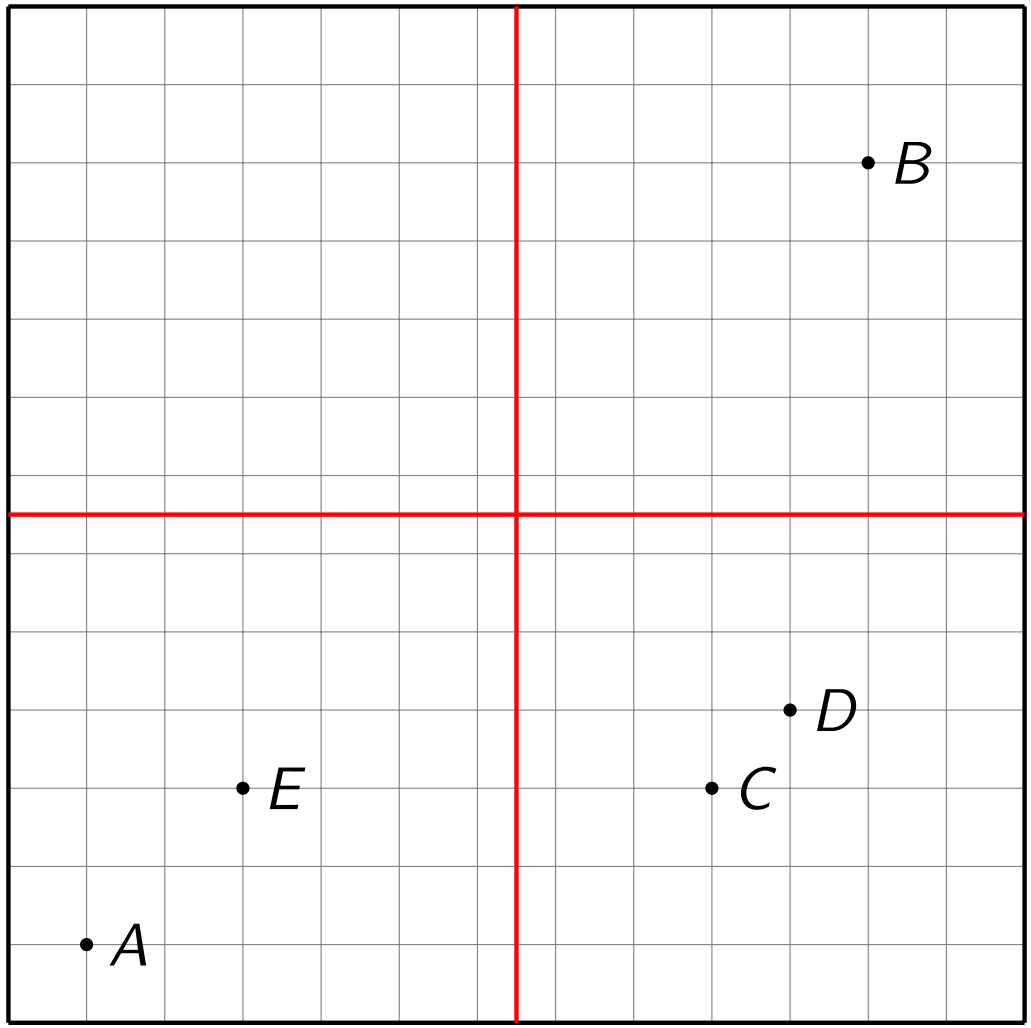
Since the page capacity is 4, we are able to add A, B, C, and D with no hiccups.



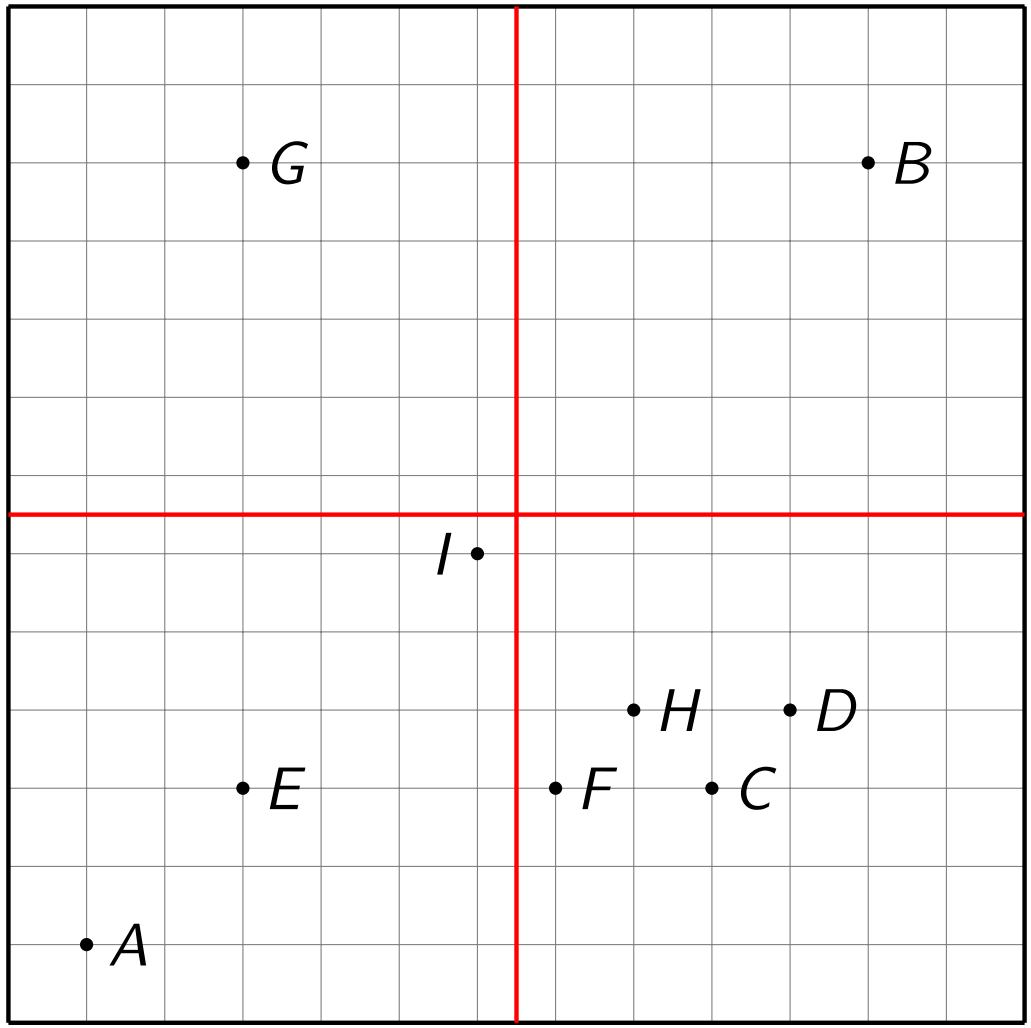
Overflow! Adding Point E, means that we will have 5 points in the data space, which is higher than 4. Therefore we have to do a quad-split.



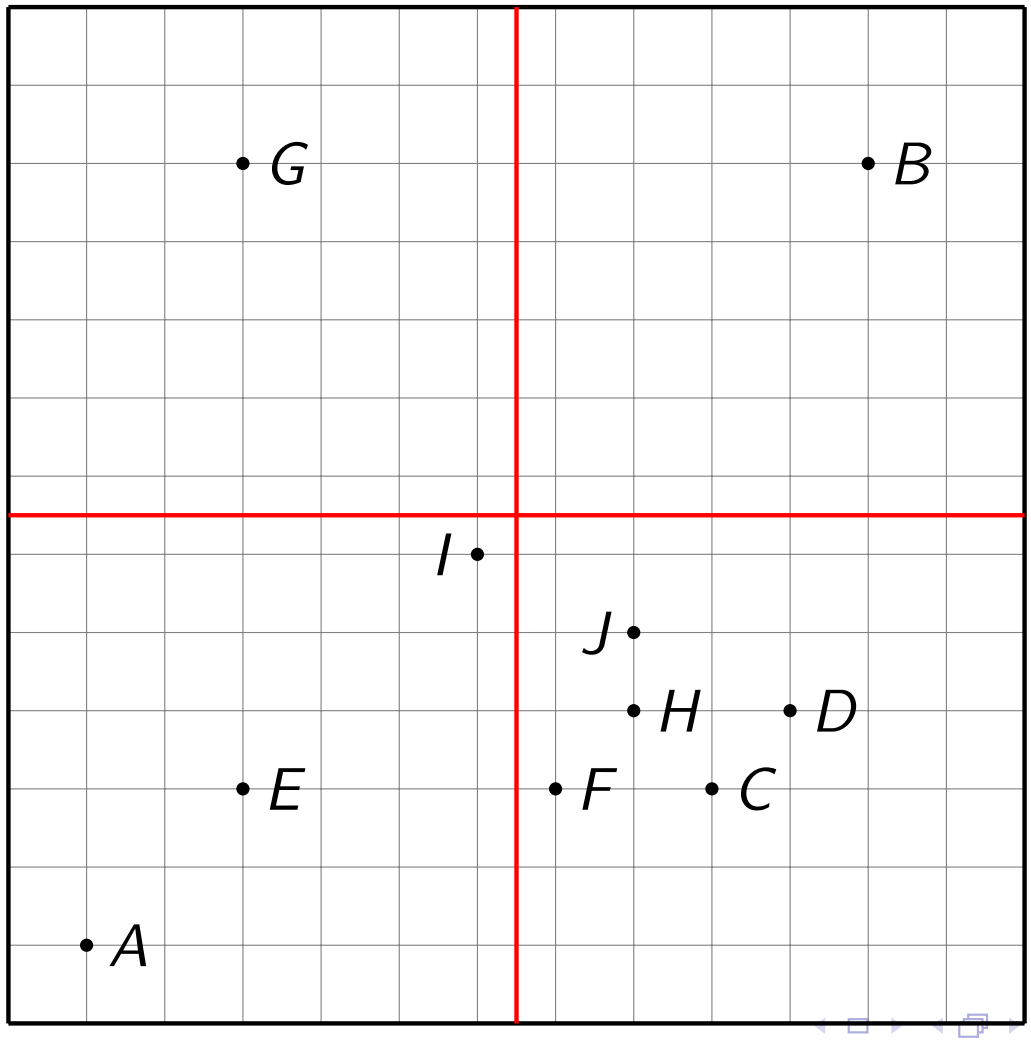
The Quad-tree is agnostic of point density, so we simply bisect the length and the width to create 4 children.



After the quad-split, we are able to add F, G, H, and I to the Quad tree without overflow.



Overflow! Adding Point J, means that we will have 5 points in the Southeast quadrant which is higher than 4. Therefore we have to do a quad-split on just the SE quadrant.



Again, we bisect length and width to separate the SE quadrant. Lastly, we have no more points to add and all boxes have less than 4 points. **DONE! :)**

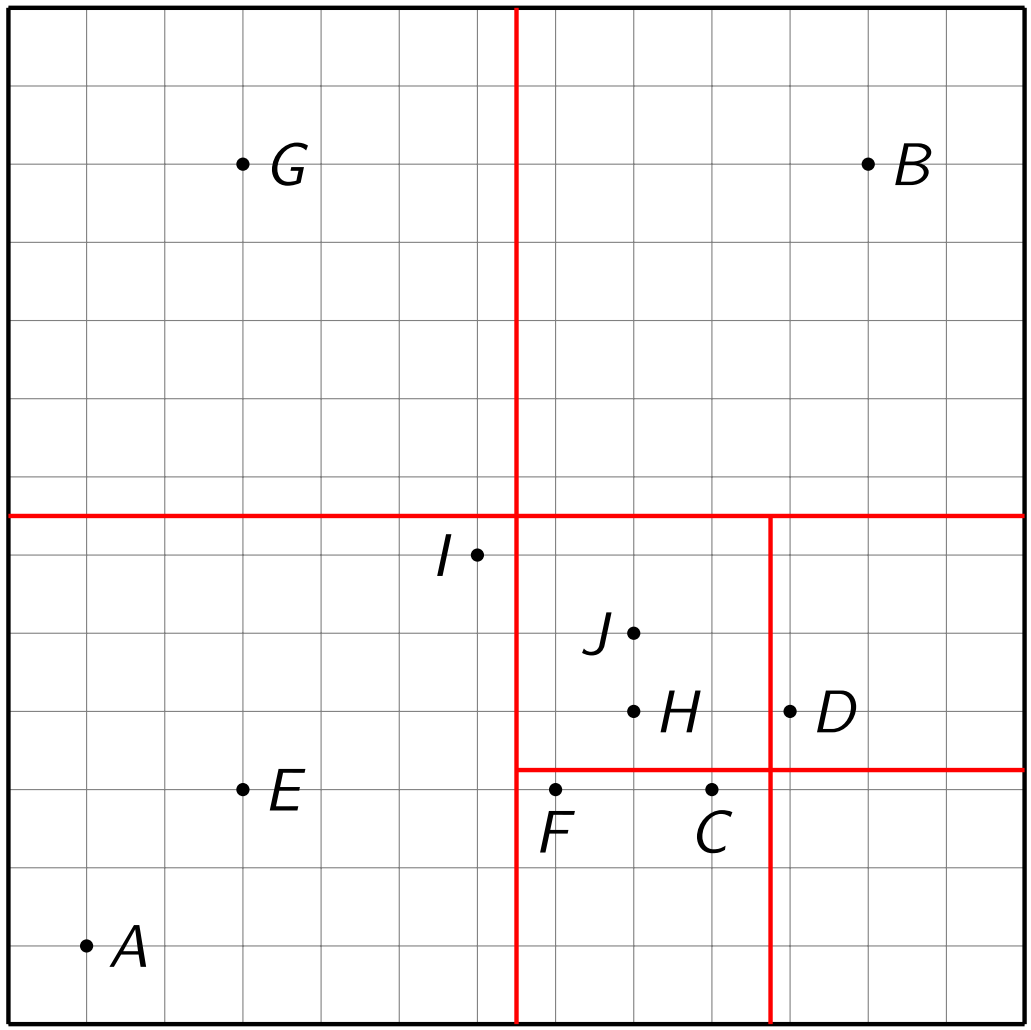


Table of Contents

Assignment 4-1: Quad Trees

Assignment 4-2: kD-Trees

Assignment 4-4: R Trees

kd-Tree Set-up

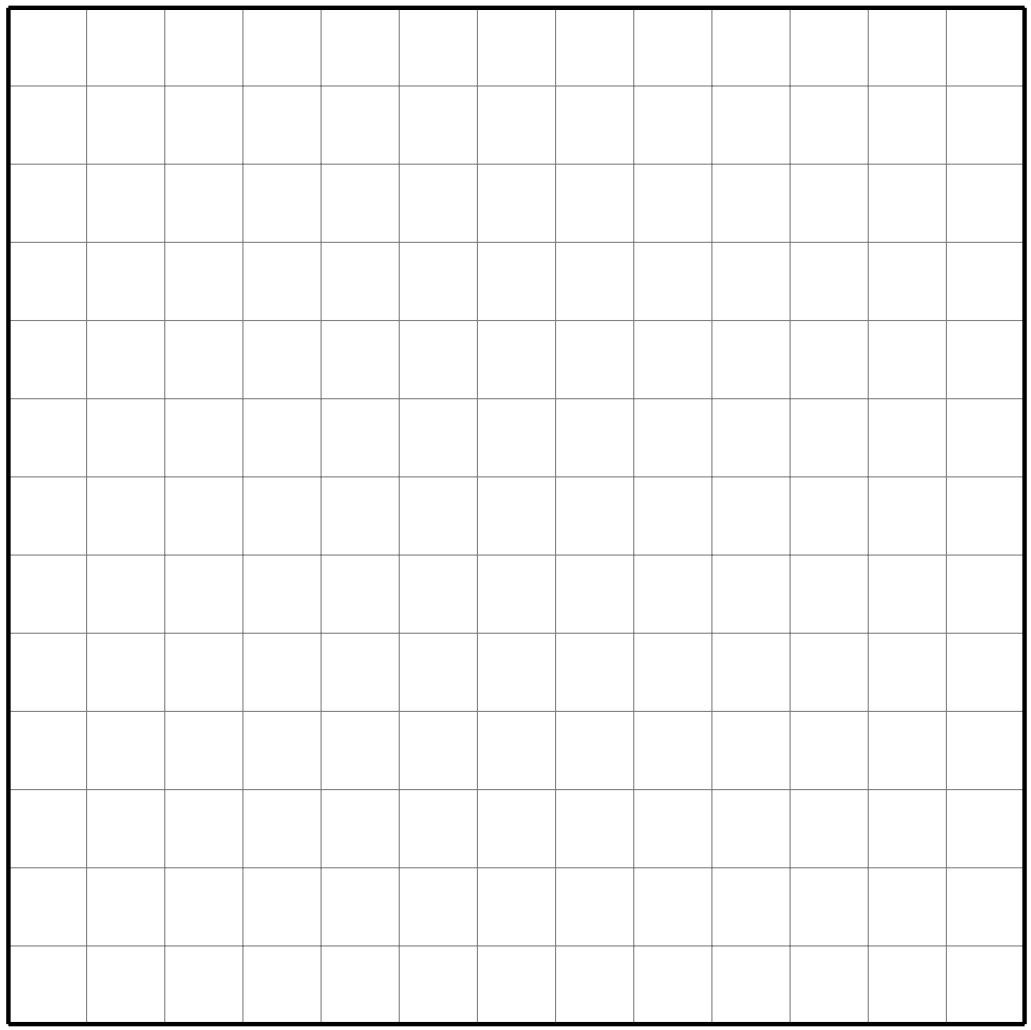
Insert the following points (in this order) into an initially empy Quad-Tree:

$$A = (0, 0), B = (10, 10), C = (8, 2), D = (9, 3), E = (2, 2),$$
$$F = (6, 2), G = (2, 10), H = (7, 3), I = (5, 5), J = (7, 4)$$

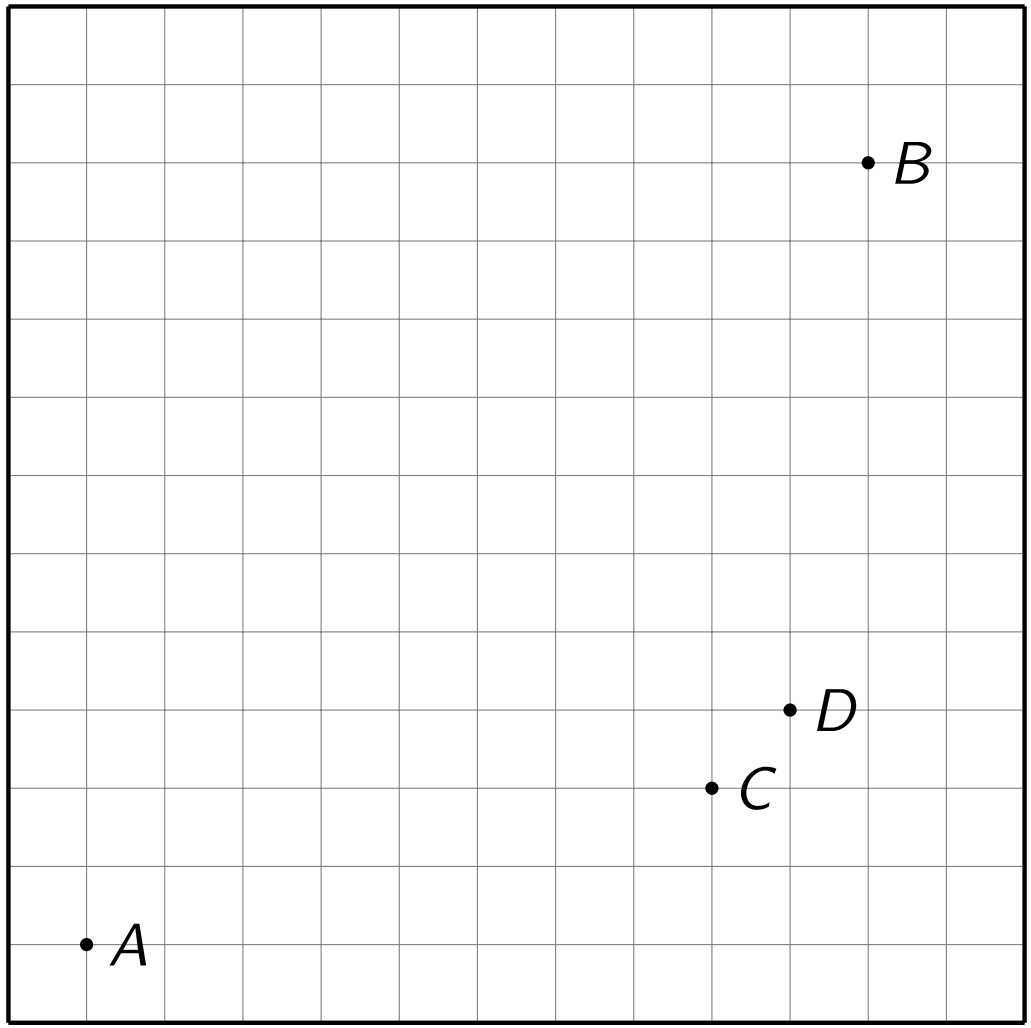
The maximum page capacity of this kd-Tree is 4.

- Draw the current kd-tree after each split.

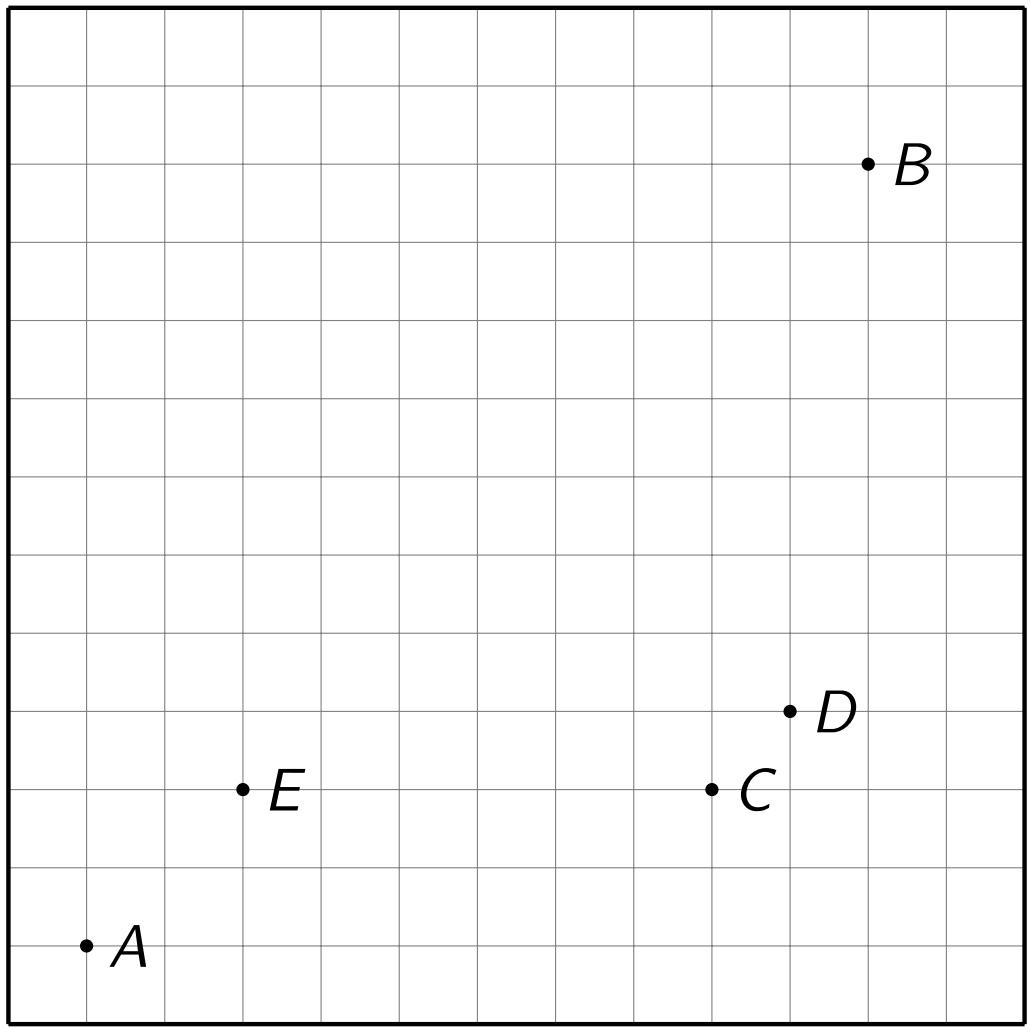
The initially empty kd-tree has one entry that covers the entire data space, by definition. We decided to make it $[-1, 12] \times [-1, 12]$.



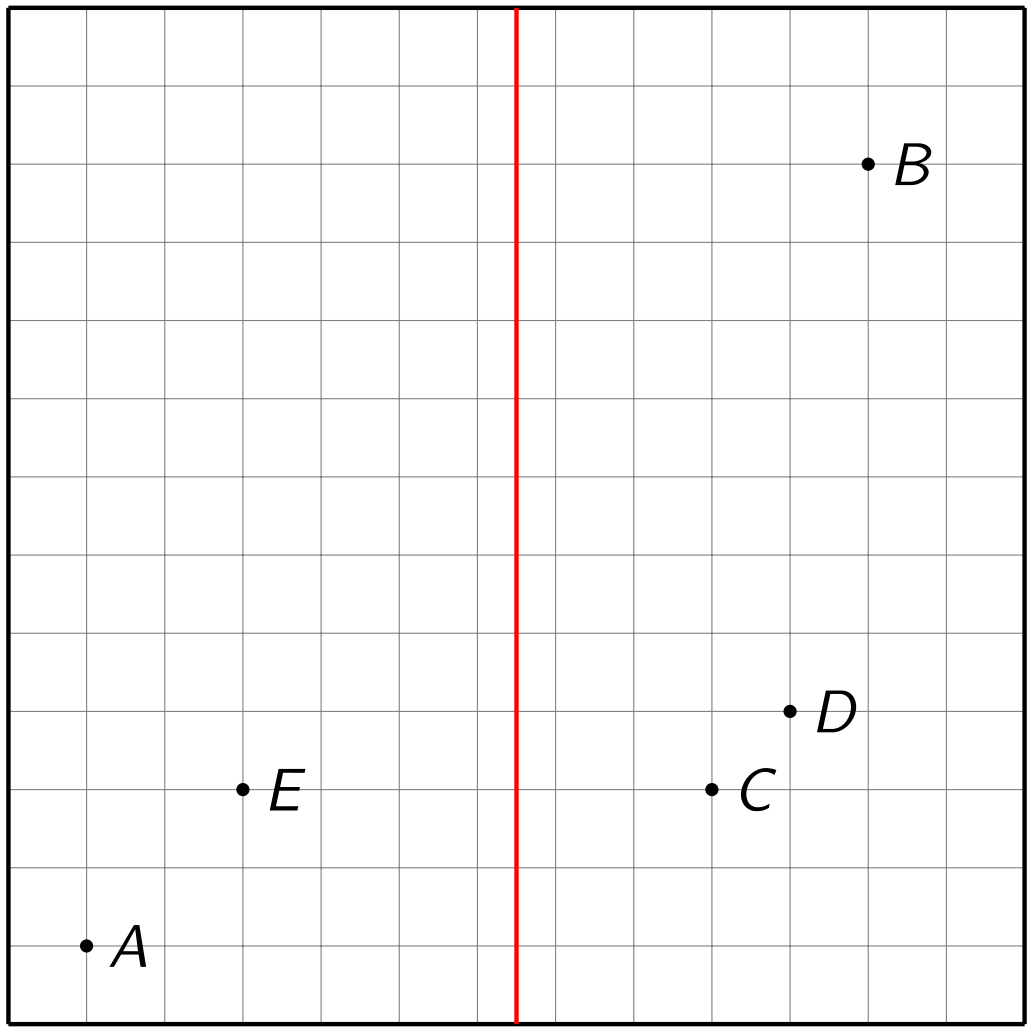
Since the page capacity is 4, we are able to add A, B, C, and D with no hiccups.



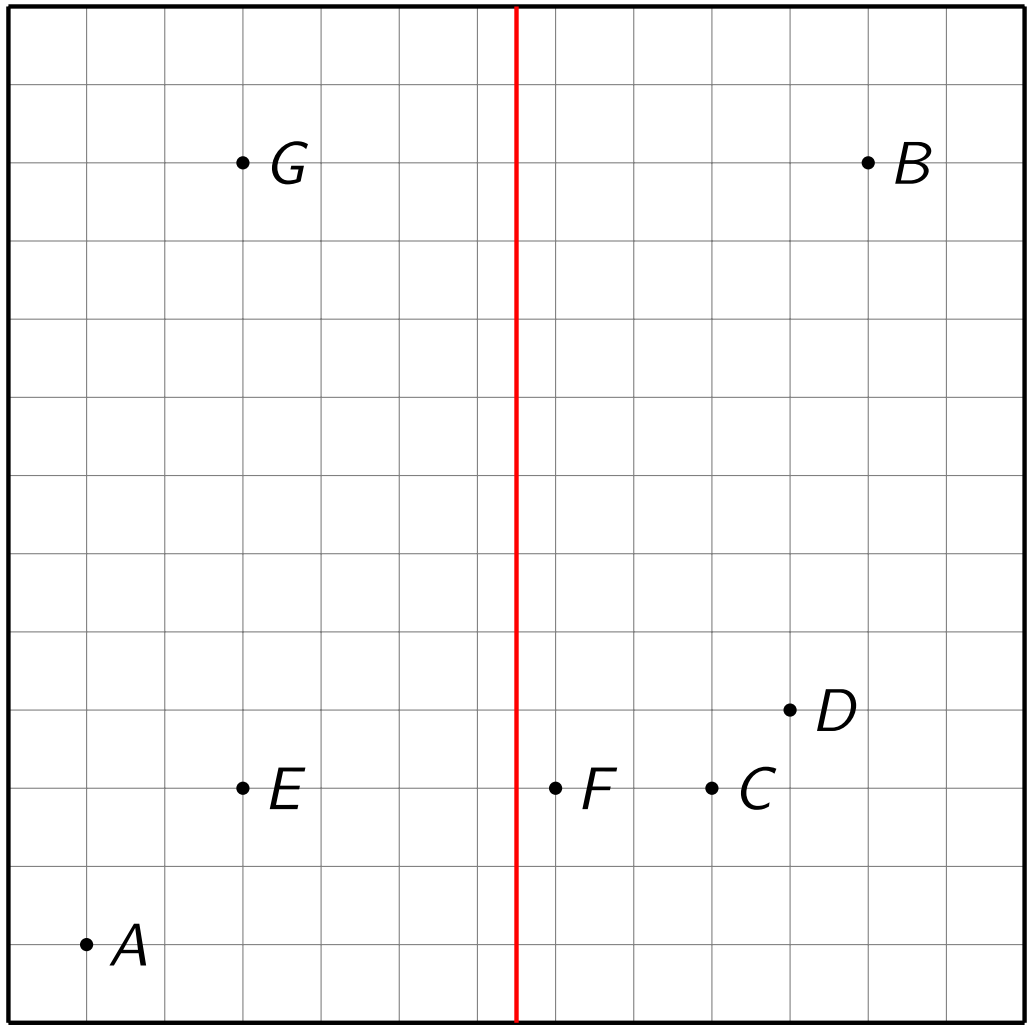
Overflow! Adding Point E, means that we will have 5 points in the data space, which is higher than 4. Since depth(Point E) % 2 = 0, we will perform a horizontal split.



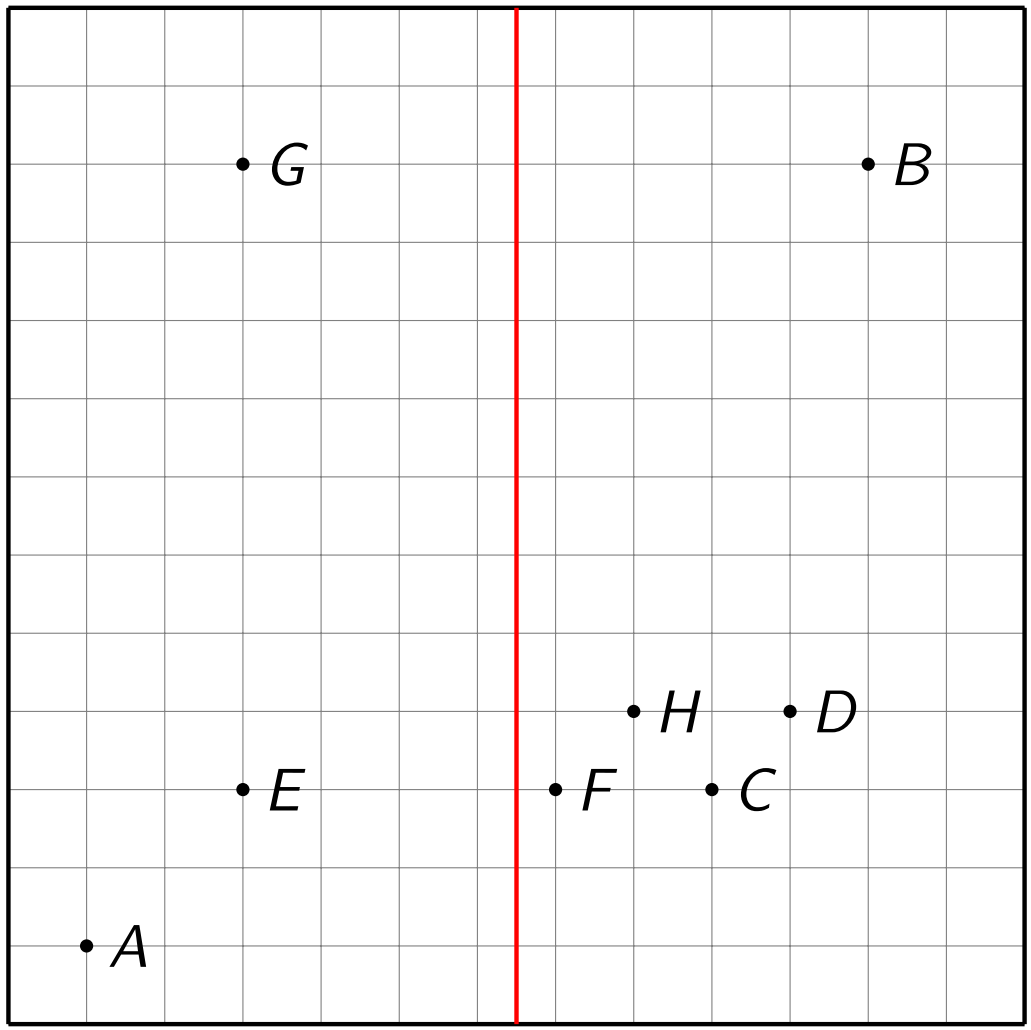
We need to make sure that the vertical line separates the points around the median, so there are 2 points on one side, 3 on the other.



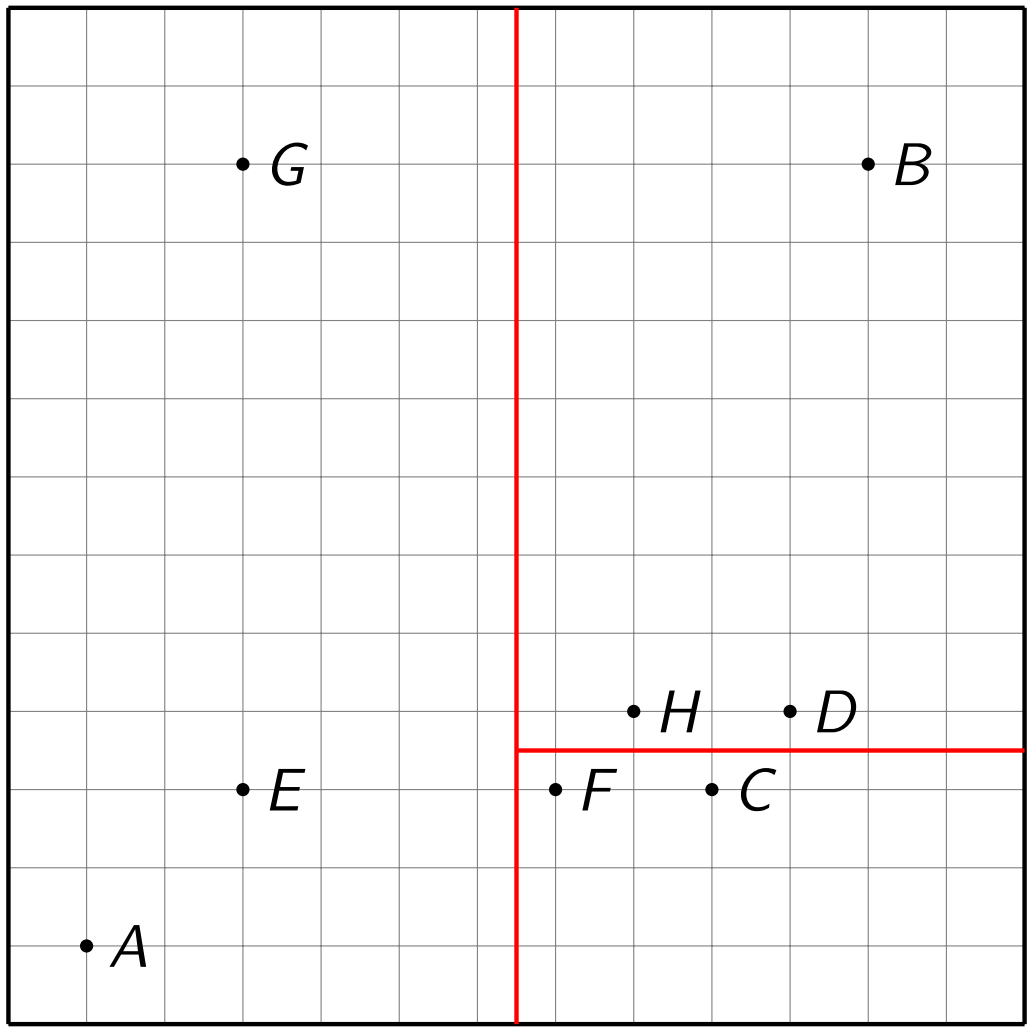
We can add Points F and G to the kd-tree, and both nodes still have $\leqslant 4$ points.



Overflow! Adding Point H, means that we will have 5 points in theright node, which is higher than 4. Since depth(Point H) % 2 = 1, we will perform a vertical split.



We need to make sure that the horizontal line separates the points around the median, so there are 2 points on one side, 3 on the other.



We add Points I and J to the kd-Tree. Lastly, we have no more points to add and all nodes have $\leqslant 4$ points.

DONE! :)

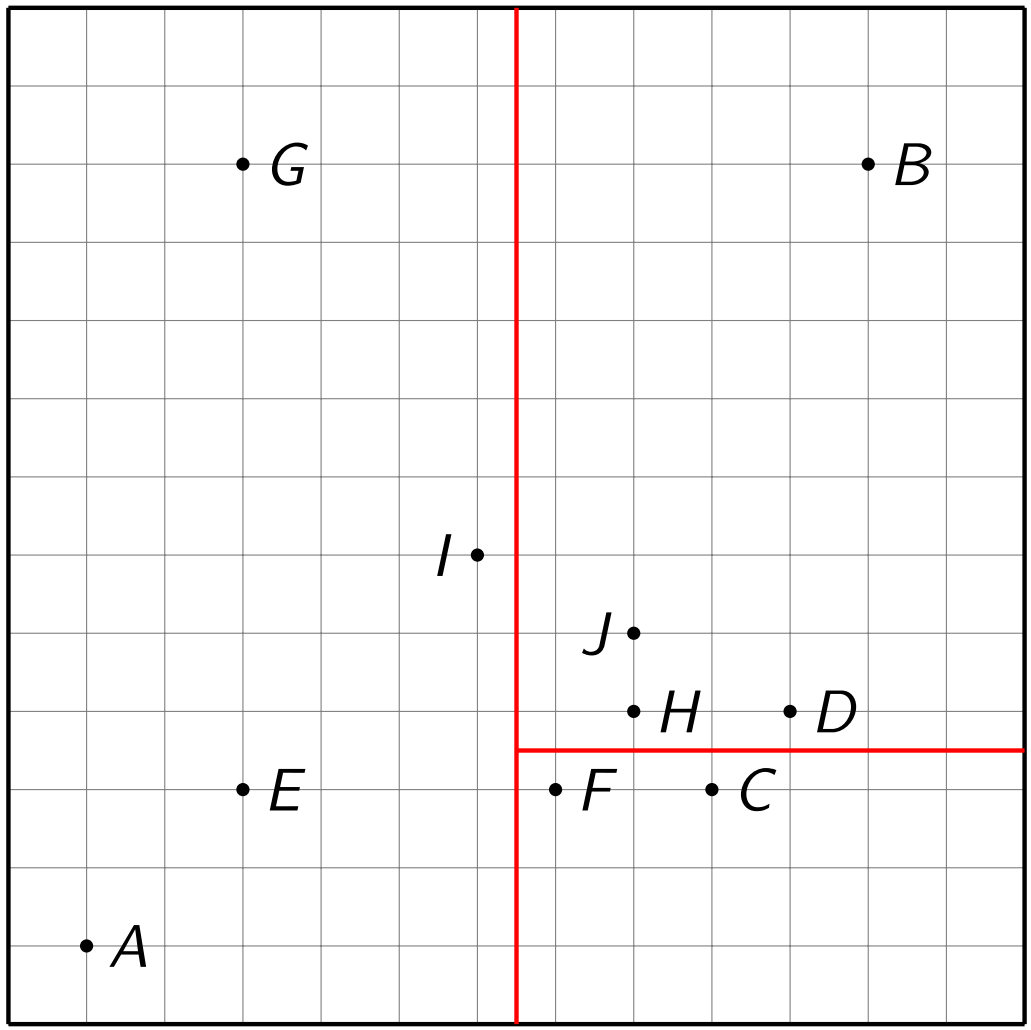


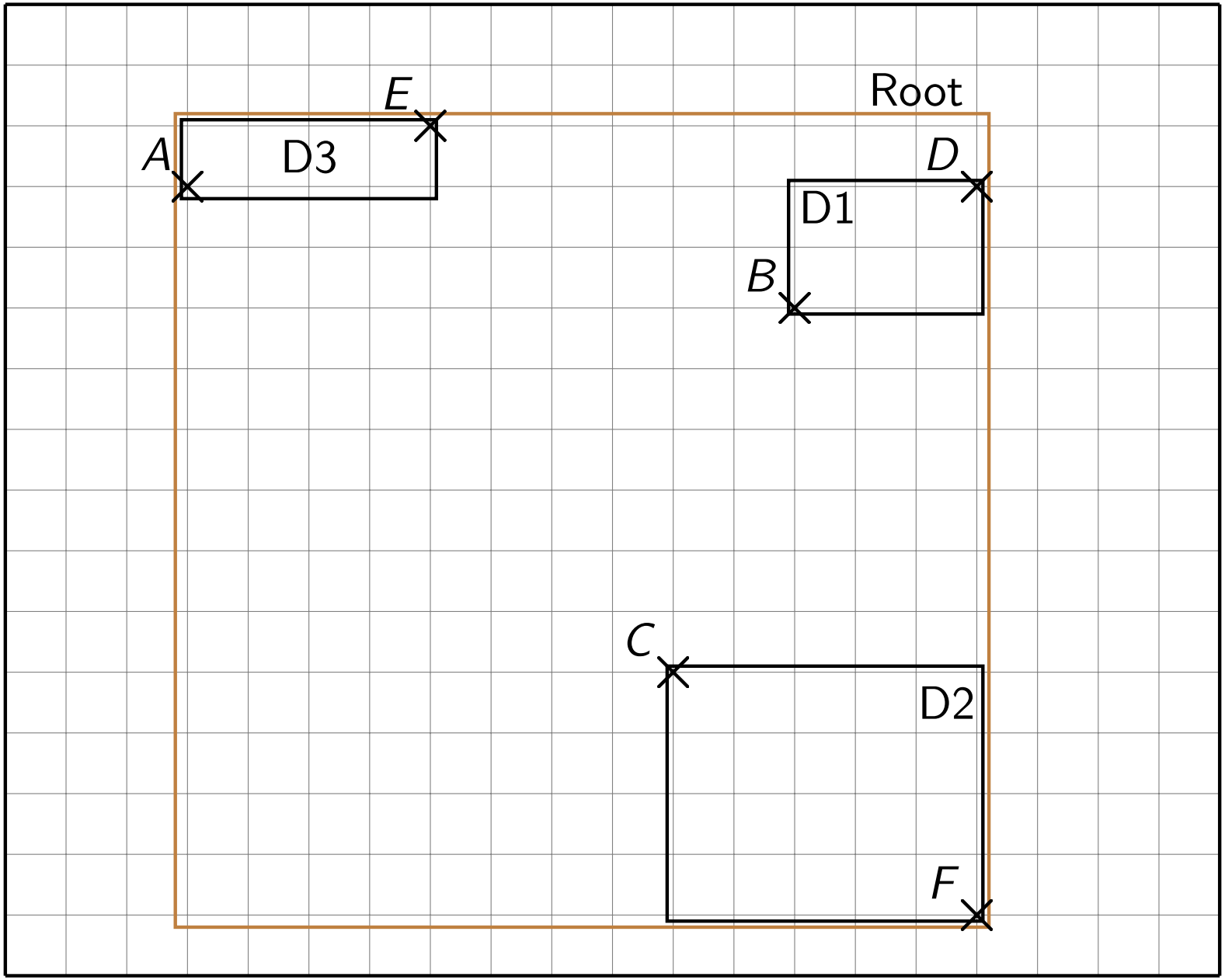
Table of Contents

Assignment 4-1: Quad Trees

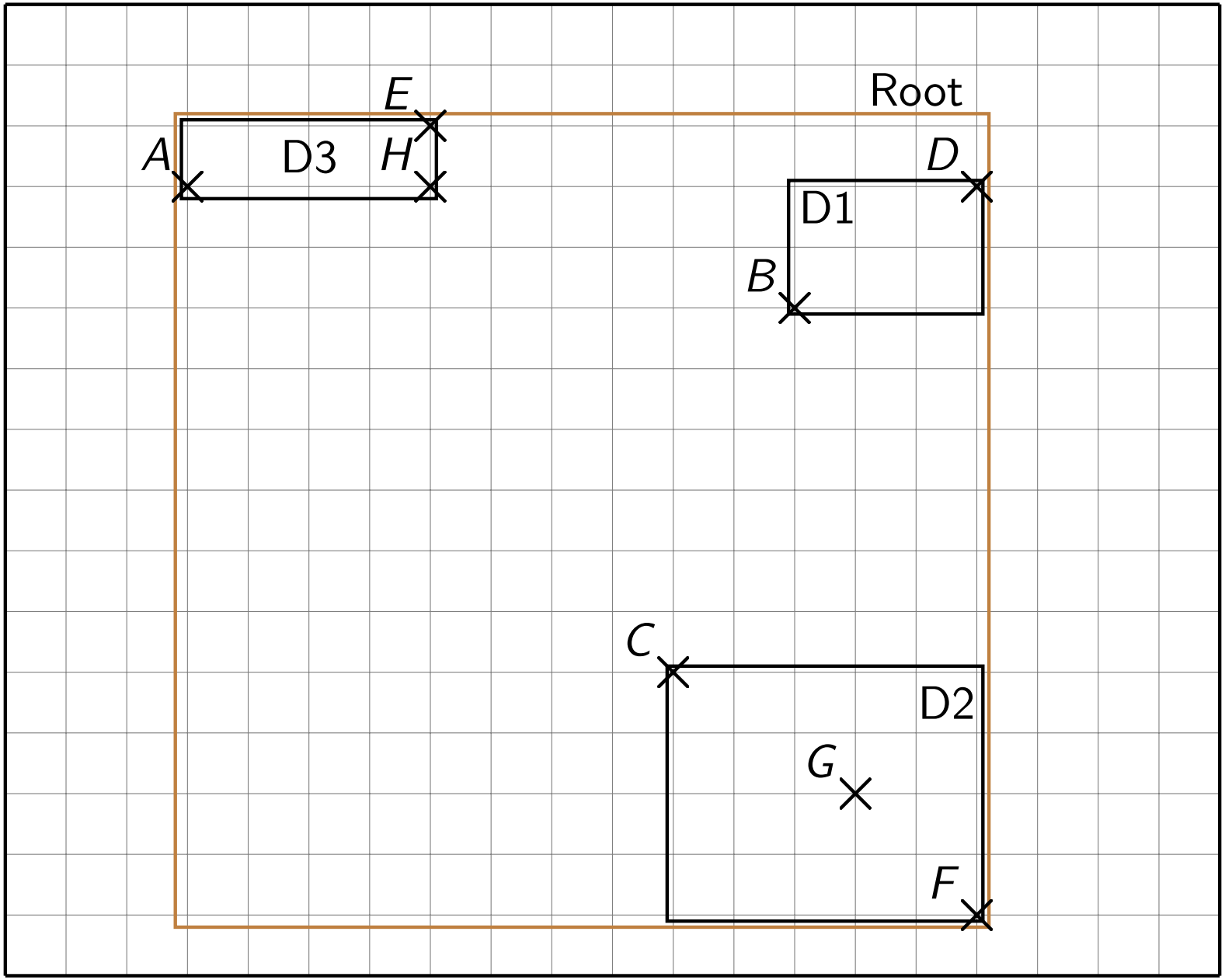
Assignment 4-2: kD-Trees

Assignment 4-4: R Trees

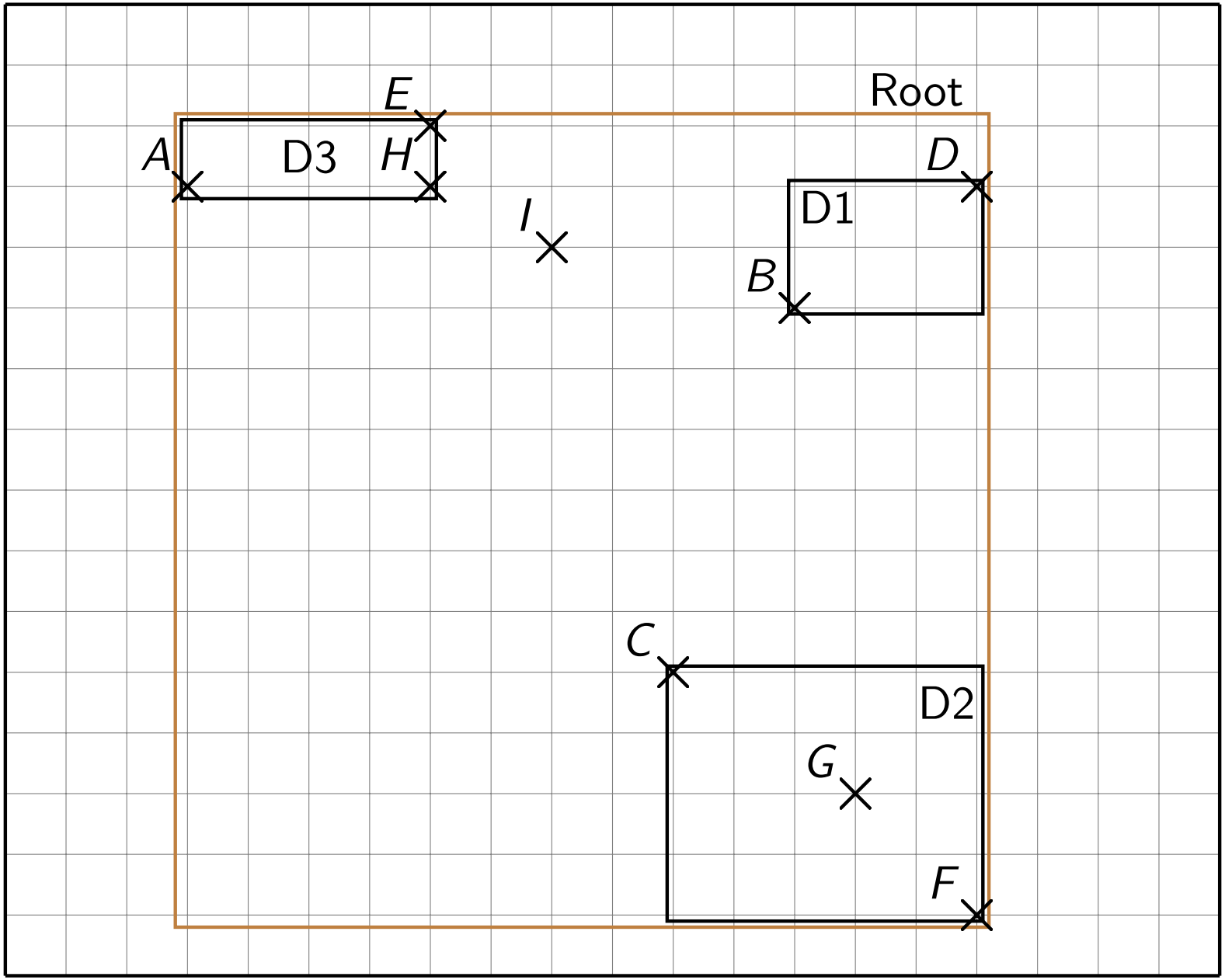
This is the R-tree into which we are inserting points.



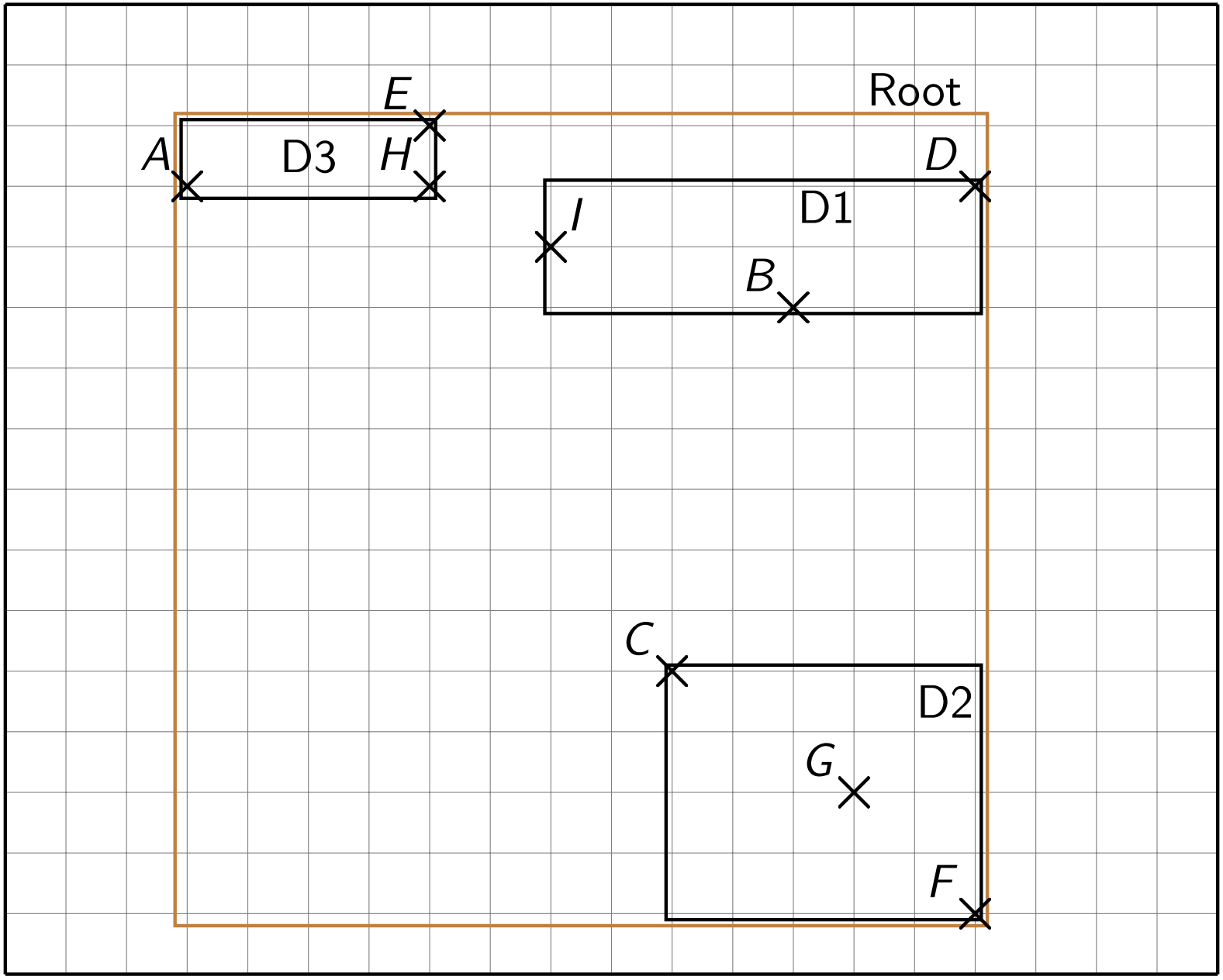
We are able to add Points G and H without overflow.



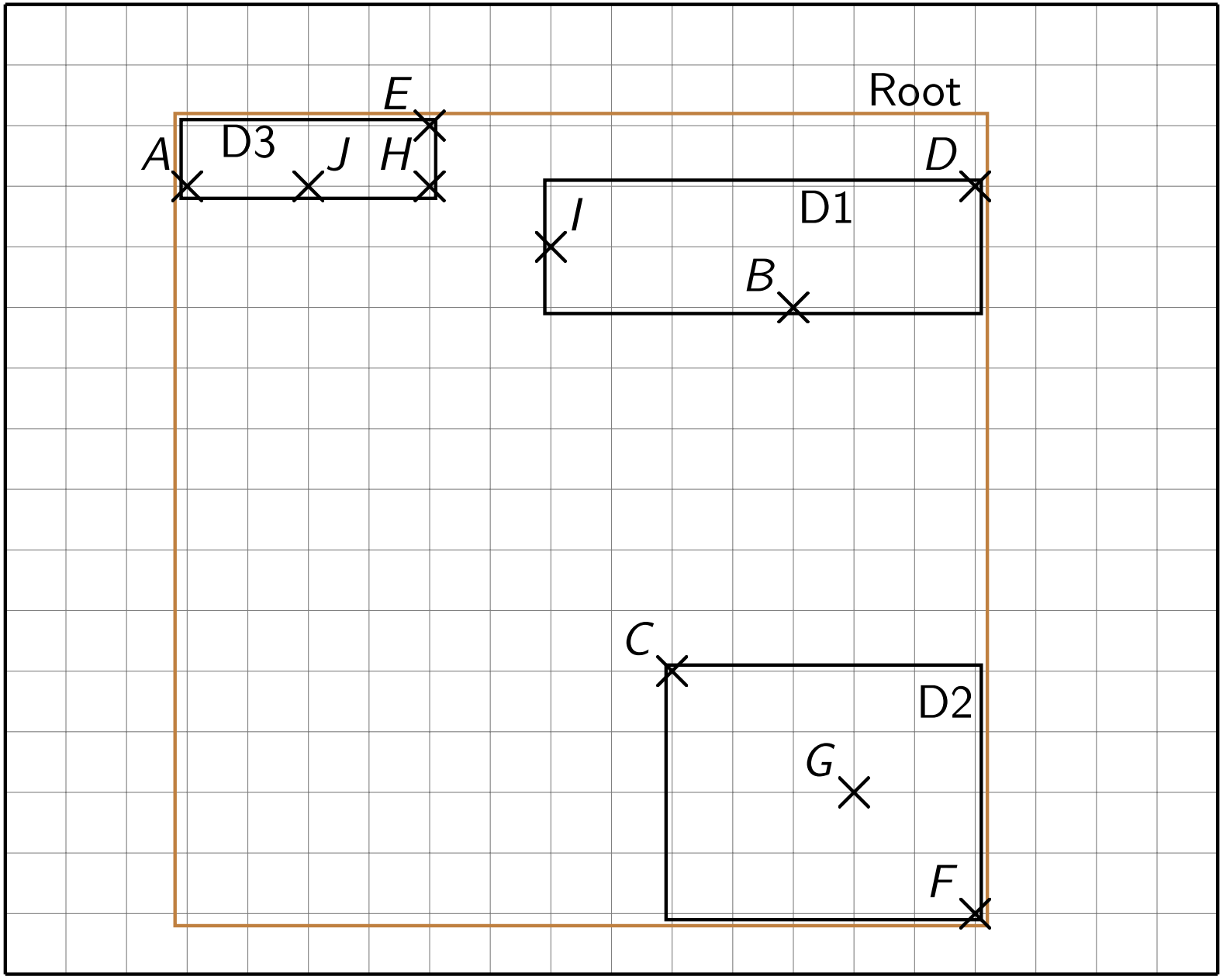
We add Point I, which isn't in a node.



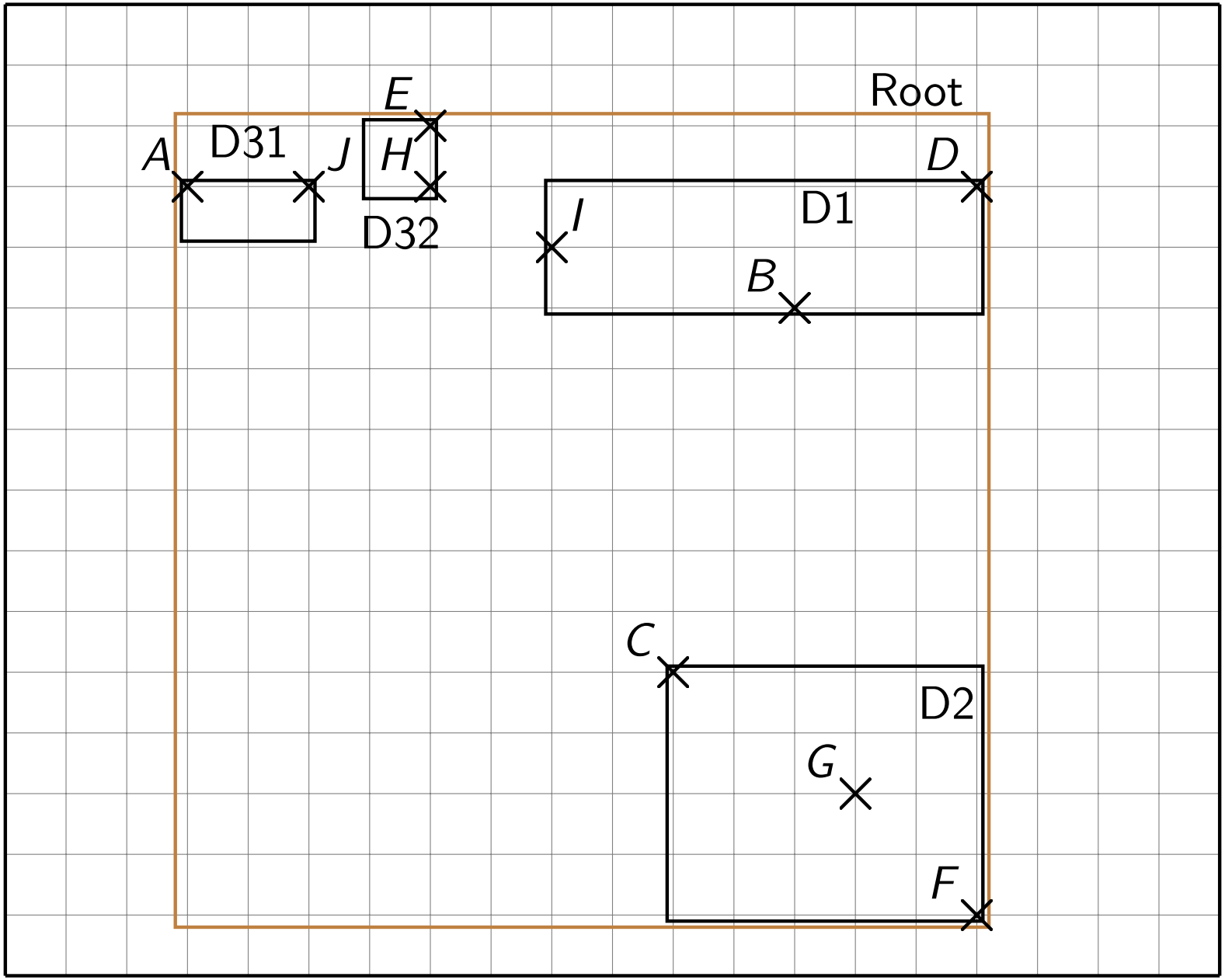
Expanding D1, D3 both yield an increase of 8, we pick D1.



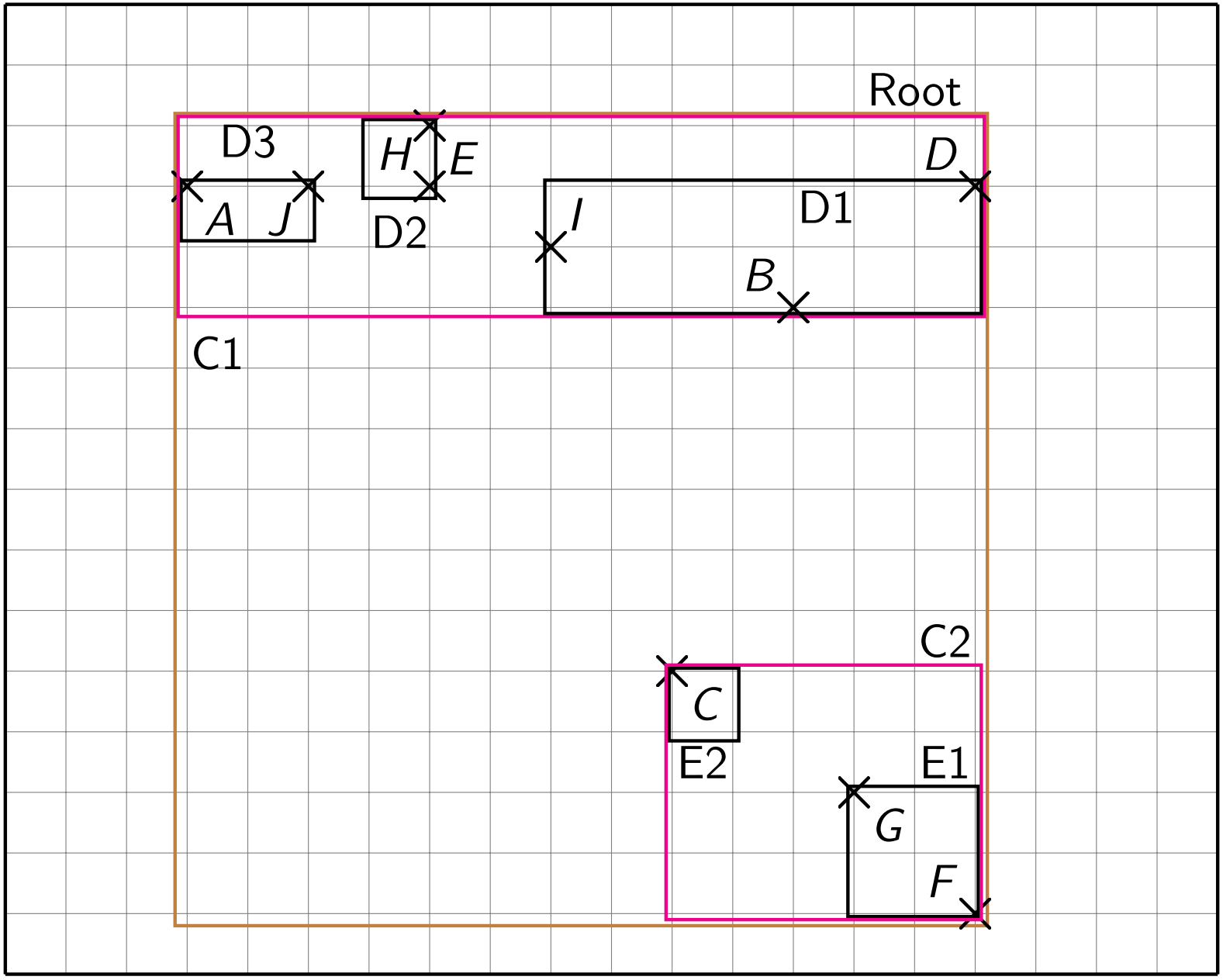
Overflow! Adding Point J, D3 is overfilled, so we split it.



Overflow! We have overflowed the Root with 4 children



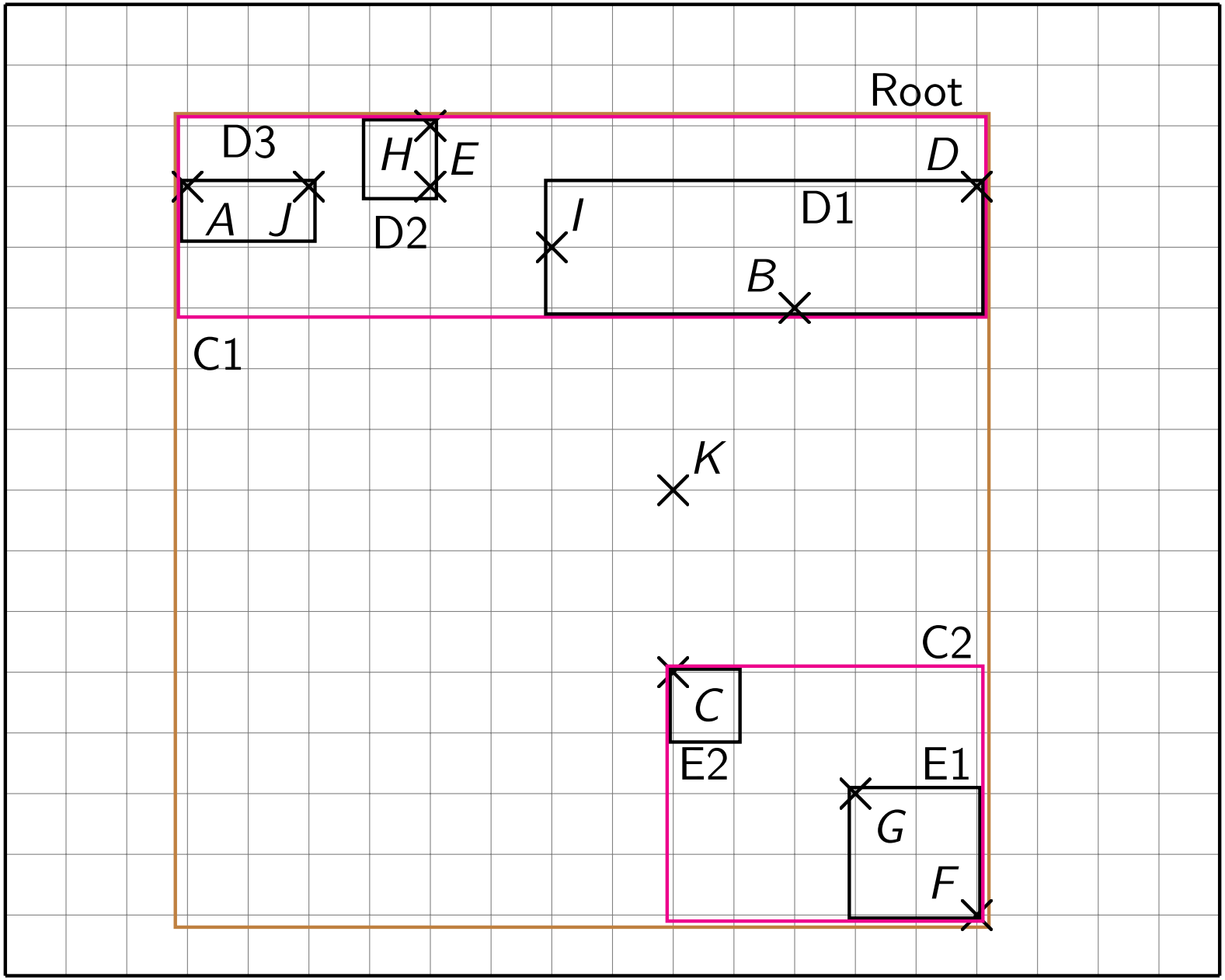
We group the 4 nodes into 2 clusters, fixing the overflow.



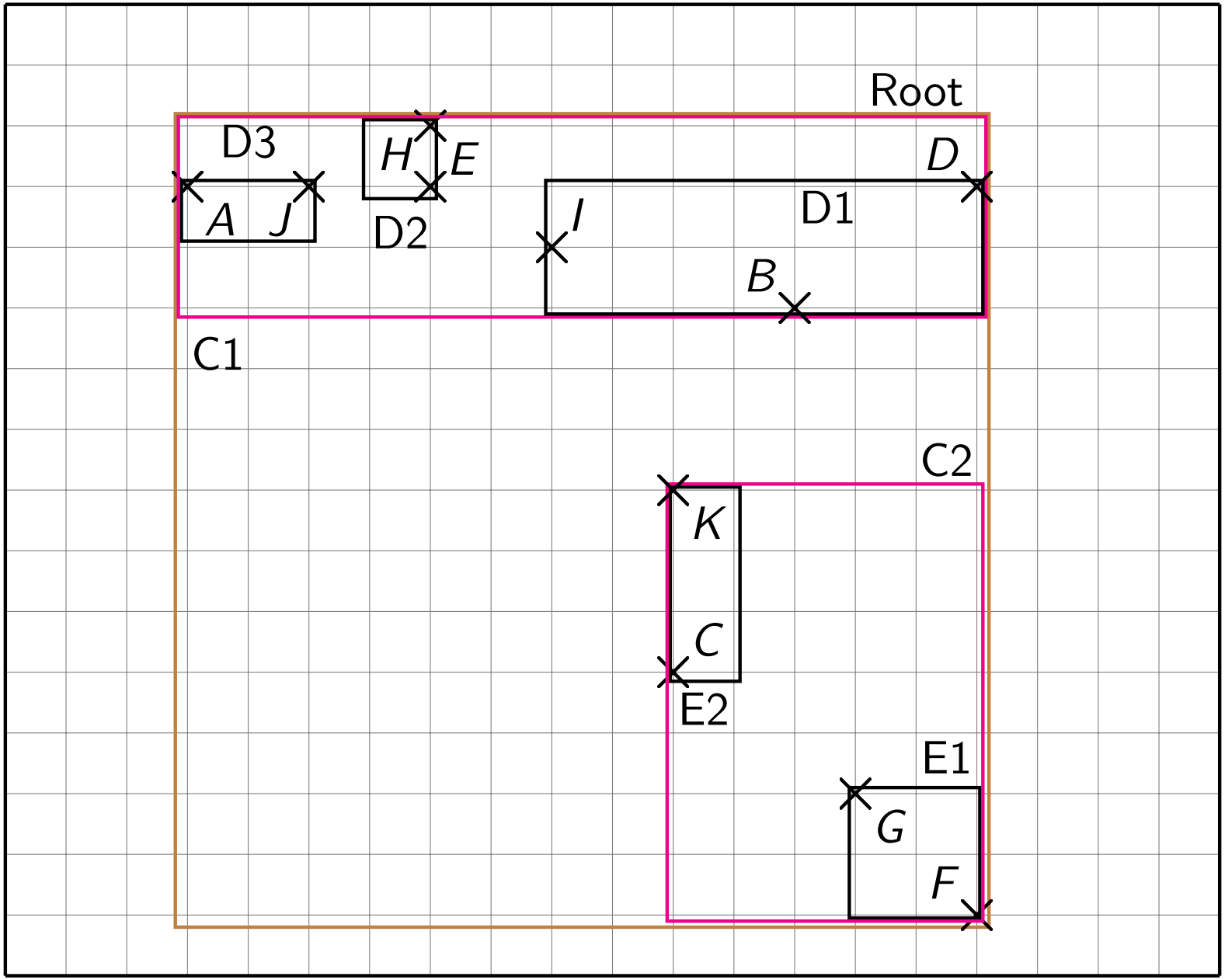
Conceptual Explanation

Since we overflowed the root, we needed to find a way to combine the pages into 3 or less clusters. Additionally, we need all the points to be at the same level so they are all balanced. We decided to create C1 and C2 which both have at most 3 pages in them. There were two ways to combine the pages, but the way that was proposed above has no overlap and minimized the area of the magenta pages.
By doing what we did, we have balanced the tree, minimized overlap and total area. Also we have solved the overflows. Neither the root is overflowed with 3 children, nor are C1 or C2 overflowed, 3 and 2 respectively. Lastly, Each page has at least 1 entry in it.

We add Point K, which isn't in a node.



We expand C2 and E2 to add Point K to a node.



Conclusion

We expanded C2 and E2 as it minimized the area added. Since, we have no more points to add, we should double check the overflow at each level. All black pages have at least 1 and at most 3 points in them. Additionally, the magenta pages have at least 1 and at most 3 pages in them. Lastly, the Root has 2 pages within. All the points are 2 levels deep – in black which are in magenta, which are in the root. We have always minimized overlap, added area and lastly, considered which is "more square" in a locally greedy fashion. **DONE! :)**