Quadtrees

Yuping(Allan) Lu
CS 594 - Graph Theory
April 23, 2014

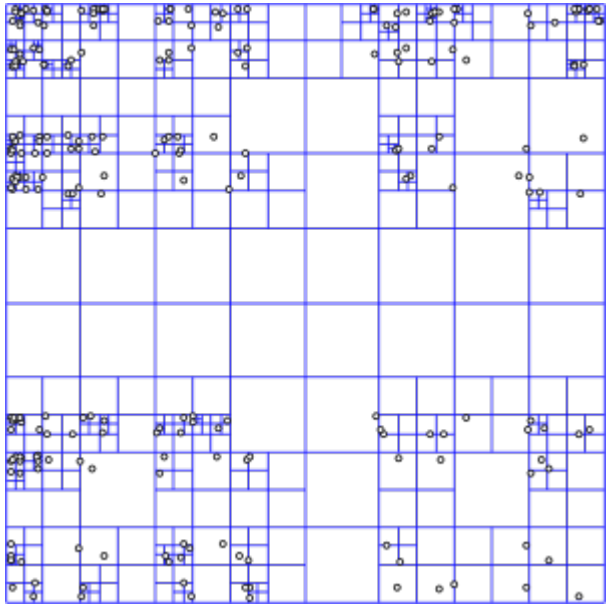
Agenda

1. Definitions
2. History
3. Examples
4. Applications
5. Open Problems
6. References
7. Homework

Definitions

A **quadtree** is a tree data structure in which each internal node has exactly four children.

In a quadtree, each node represents a bounding box covering some part of the space being indexed, with the root node covering the entire area.



A representation of how a quadtree divides an indexed area. Source: Wikipedia[4]

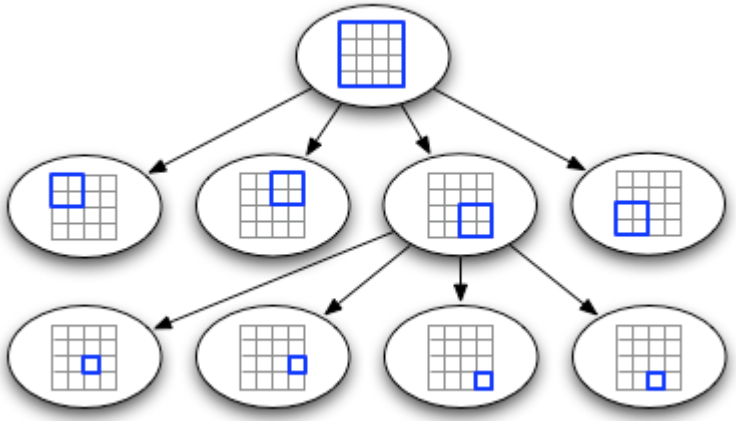
Definitions

Inserting data into a quadtree is simple:

Starting at the root, determine which quadrant your point occupies.

Recurse to that node and repeat, until you find a leaf node.

Then, add your point to that node's list of points. If the list exceeds some pre-determined maximum number of elements, split the node, and move the points into the correct subnodes.



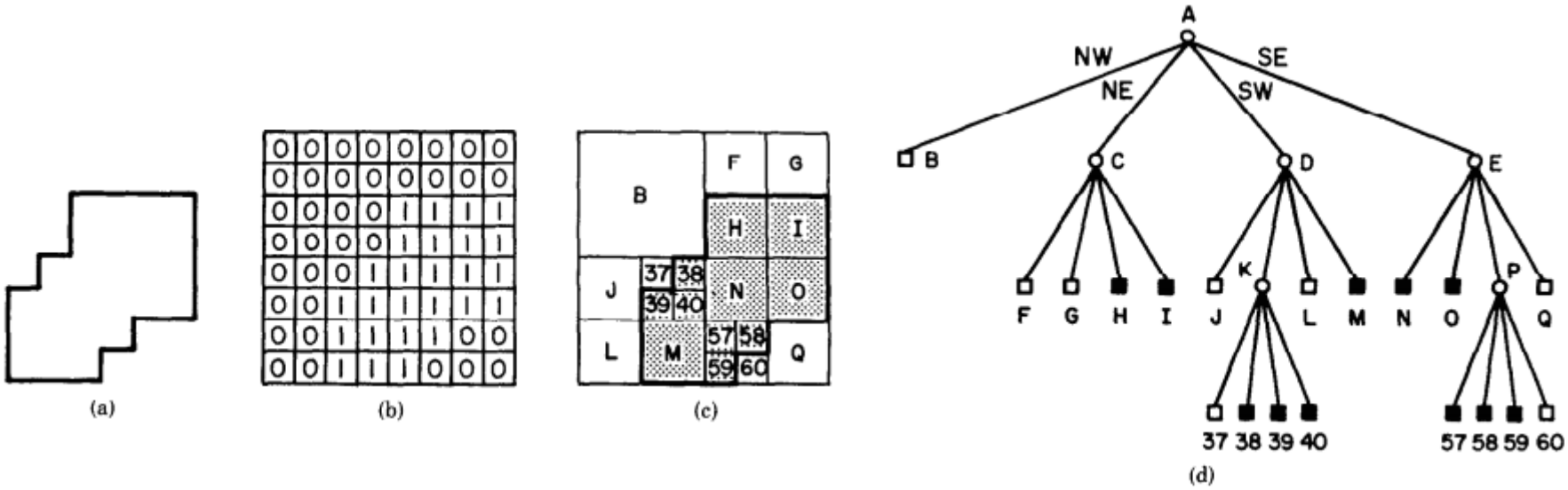
A representation of how a quadtree is structured internally. Source: [2]

Definitions

To query a quadtree, starting at the root, examine each child node, and check if it intersects the area being queried for.

If it does, recurse into that child node. Whenever you encounter a leaf node, examine each entry to see if it intersects with the query area, and return it if it does.

Definitions

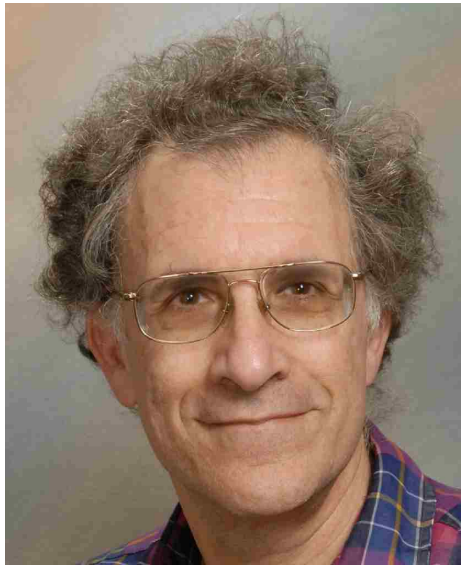


A region, its binary array, its maximal blocks, and the corresponding quadtree. (a) Region. (b) Binary array. (c) Block decomposition of the region (a). Blocks in the region are shaded. (d) Quadtree representation of the blocks in (c). Source: [3]

History

Quadtree was named by Raphael Finkel and J.L. Bentley in 1974. "Quad Trees: A Data Structure for Retrieval on Composite Keys".[1]

In their algorithm, straight forward insertion yields O(nlogn) performance with worst case: $O(n^2)$



Raphael Finkel

J.L. Bentley

History

In 1985, Hanan Samet and Robert E. Webber proposed a PM (polygonal map) quadtree.[3]

Use of the PM quadtree to perform point location, dynamic line insertion, and map overlay is discussed in [3].

Together with Aluru [6], they give an extensive overview of the various types of quadtrees and their applications

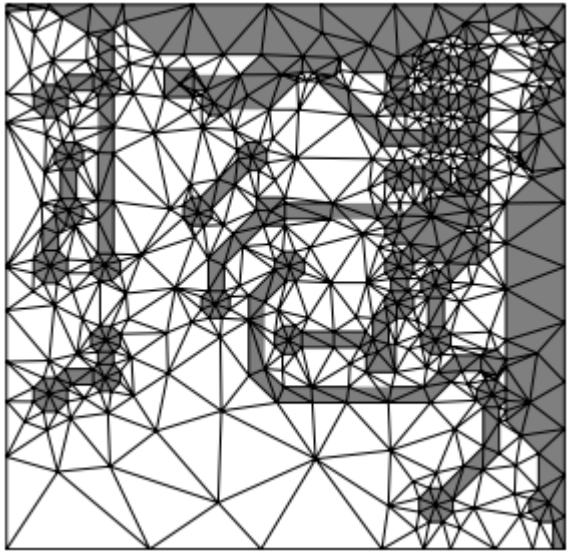
Hanan Samet

Robert E. Webber

History

Almost all electrical devices, from shavers and telephones to televisions and computers, contain some electronic circuitry to control their functioning. This raises the problem of mesh generation. Mark de Berg et al.[5] proposed the Non-Uniform Mesh Generation algorithm based on quadtrees in 2000.

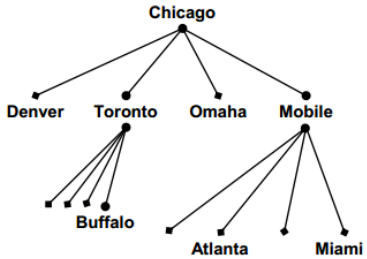
Eppstein et al. [7] developed a quadtree variant that has not only linear size, but also allows insertions, deletions, and search operations in O(logn)time in 2005.



Source: [5]

Examples--Point Quadtree

Two-dimensional binary search tree; First point is the root; The shape of the tree depends on the order of insertion



Chicago	(x_0,y_0)
Mobile	(x_1,y_1)
Toronto	(x_2,y_2)
Buffalo	(x_3,y_3)
Denver	(x_4,y_4)
Omaha	(x_5,y_5)
Atlanta	(x_6,y_6)
Miami	(x_7,y_7)

Source: [11]

Examples--PR Quadtree

A quadtree for a set of points P in a square $Q=[x_{1q}:x_{2q}] \times [y_{1q}:y_{2q}]$ is:

– If $|P| \leq 1$, then the quadtree is a single leaf

– Otherwise, let Q_{NE}, Q_{NW}, Q_{SE} and Q_{SW} are the four quadrants[5]

$$x_{mid} = (x_{1q} + x_{2q})/2 \quad , \quad y_{mid} = (y_{1q} + y_{2q})/2$$

$$P_{NE} = \{p \in P : p_x > x_{mid} \wedge p_y > y_{mid}\}$$

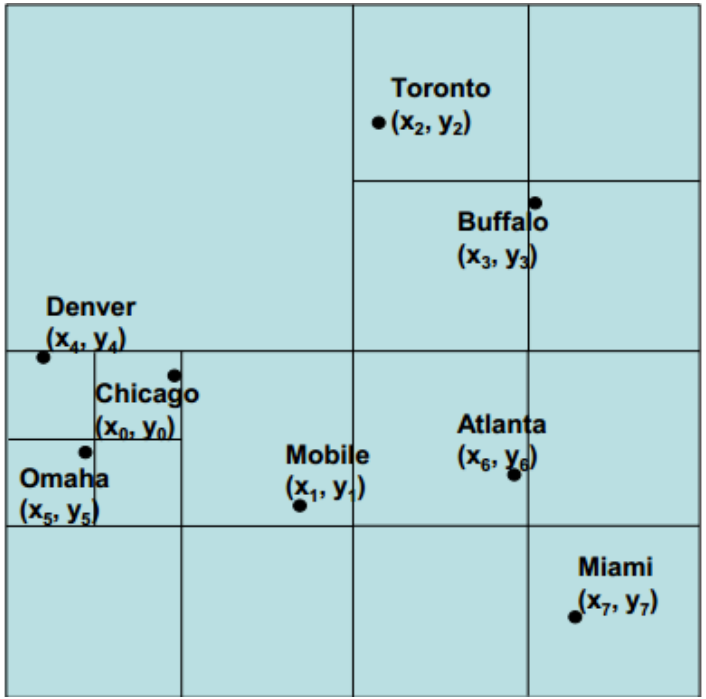
$$P_{NW} = \{p \in P : p_x \leq x_{mid} \wedge p_y > y_{mid}\}$$

$$P_{SW} = \{p \in P : p_x \leq x_{mid} \wedge p_y \geq y_{mid}\}$$

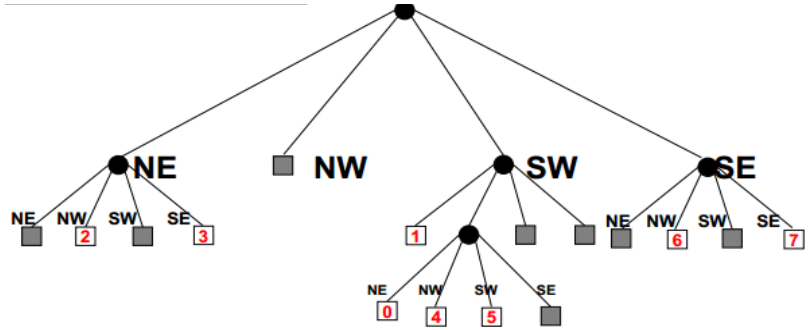
$$P_{SE} = \{p \in P : p_x > x_{mid} \wedge p_y \leq y_{mid}\}$$

The quadtree consists of root node v; Q is stored at v; Q(v) denotes the square stored at v

Examples--PR Quadtree



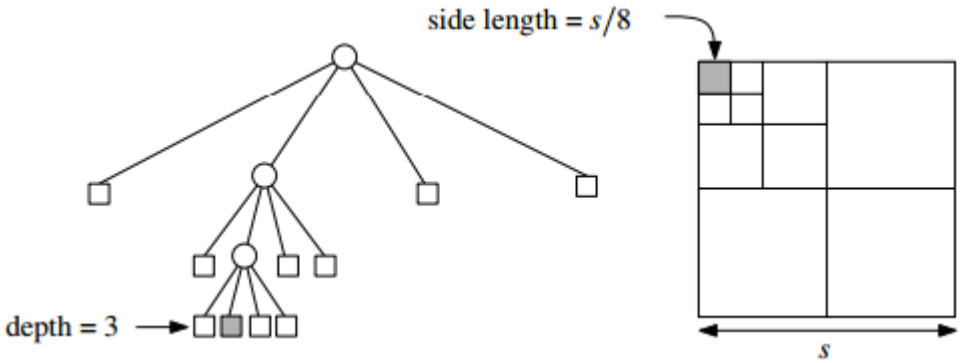
0	Chicago	(x_0,y_0)
1	Mobile	(x_1,y_1)
2	Toronto	(x_2,y_2)
3	Buffalo	(x_3,y_3)
4	Denver	(x_4,y_4)
5	Omaha	(x_5,y_5)
6	Atlanta	(x_6,y_6)
7	Miami	(x_7,y_7)



Source: [11]

Theorems

Lemma 1: The depth of a quadtree for a set P of points in the plane is at most log(s/c) + 3/2, where c is the smallest distance between any two points in P and s is the side length of the initial square that contains P. [5]



A node at depth i corresponds to a square of side length $s/2^i$

Theorems

Lemma 2: A quadtree of depth d storing a set of n points has O((d+1)n) nodes and can be constructed in O((d+1)n)time. [5]

An operation on quadtrees that is often needed is neighbor finding: given a node v and a direction—north, east, south, or west—find a node v’ such that Q(v’)is adjacent to Q(v) in the given direction.

Theorems

Neighbor Finding: North of v?

If v is SE or SW child

– Then its north neighbor is NE or NW child of its parent

If v is NE or NW child

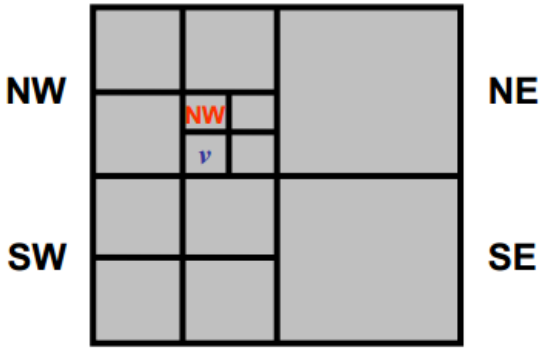
– Then find the north neighbor u of the parent of v

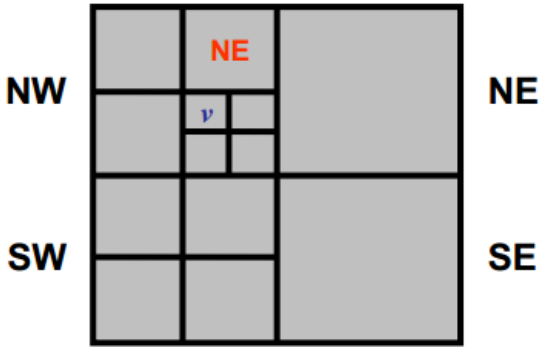
• If u is internal node

– Then the north neighbor is a child of u

• If u is a leaf

– Then the north neighbor is u



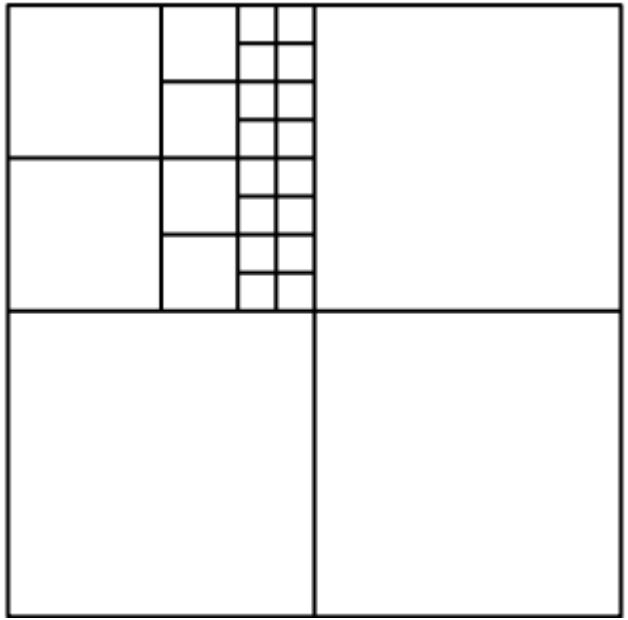


Theorems

Theorem: Let T be a quadtree of depth d. The neighbor of a given node v in T in a given direction, as defined above, can be found in O(d+1)time. [5]

Examples--Balanced Quadtree

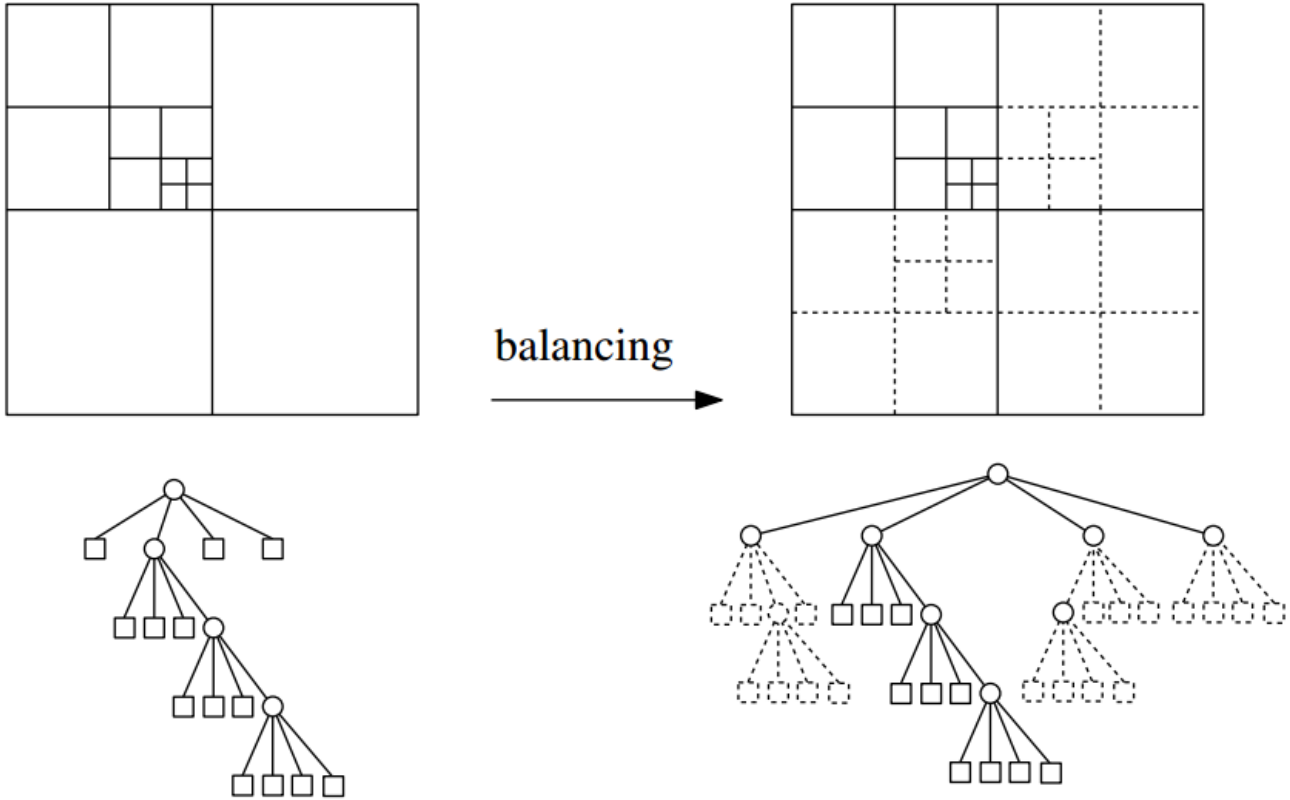
A quadtree is balanced if any two neighboring nodes differ at most 1 in depth [12]



an unbalanced quadtree subdivision

Balancing a Quadtree

Add nodes.



Complexity of a balanced Quadtree

Theorem: Let T be a quadtree with m nodes. Then the balanced version of T has O(m) nodes and can be constructed in O((d + 1)m) time.[12]

Applications

1. Image representation
2. Spatial indexing
3. Efficient collision detection in two dimensions
4. Solution of multidimensional fields (computational fluid dynamics, electromagnetism)
5. Conway's Game of Life simulation program.
6. State estimation[4]

Collision Detection

Collision detection is an essential part of most video games. Both in 2D and 3D games, detecting when two objects have collided is important as poor collision detection can lead to some very interesting results. [9]

Collision detection is also a very expensive operation. Let's say there are 100 objects that need to be checked for collision. Comparing each pair of objects requires 10,000 operations - that's a lot of checks!

Collision Detection

Two objects that are at opposite ends of the screen can not possibly collide, so there is no need to check for a collision between them. We can use quadtree to reduce the number of checks.

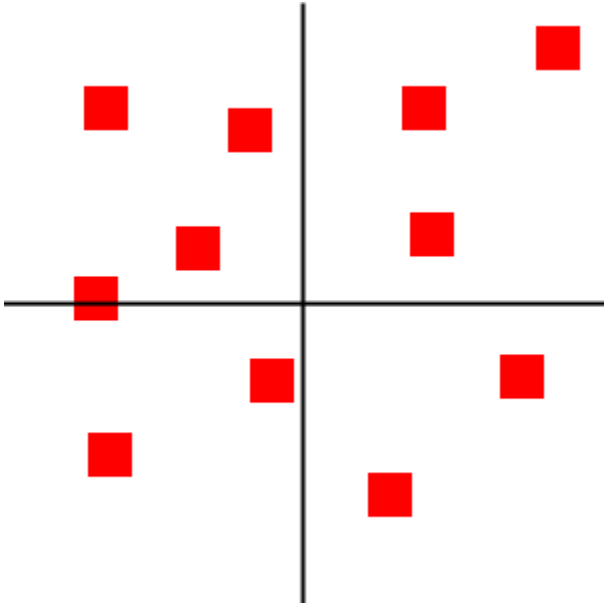
A quadtree starts as a single node.

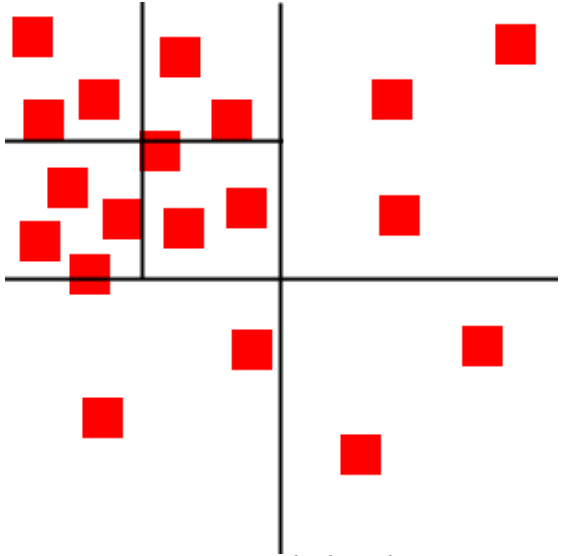
Objects added to the quadtree are

added to the single node.

Collision Detection

When more objects are added to the quadtree, it will eventually split into four subnodes. Each object will then be put into one of these subnodes according to where it lies in the 2D space. Any object that cannot fully fit inside a node's boundary will be placed in the parent node.

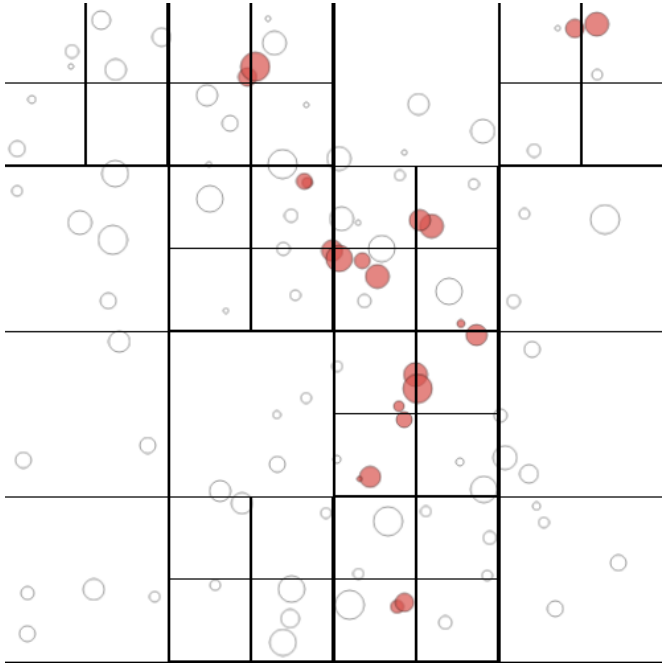




Each subnode can continue subdividing as more objects are added.

Collision Detection

As you can see, each node only contains a few objects. We know then that, for instance, the objects in the top-left node cannot be colliding with the objects in the bottom-right node, so we don't need to run an expensive collision detection algorithm between such pairs.



Source: [8]

Implementation of the operations

Actual objects stored inside the leaves, not inner nodes.[10]

1. **Insert an object into the quadtree:**

Check if the object intersects the current node. If so, recurse. If you've reached the leaf level, insert the object into the collection.

2. **Delete an object from the quadtree:**

Execute the exact same steps as if inserting the object, but when you've reached the leaf level delete it from the collection.

3. **Test if an object intersects any object inside the quadtree:**

Execute the exact same steps as if inserting the object, but when you've reached the leaf level check for collision with all the objects in the collection.

4. **Test for all collisions between all objects inside the quadtree:**

For every object in the quadtree execute the single object collision test.

5. **Update the quadtree:**

Delete all objects from the quadtree whose position has been modified and insert them again.

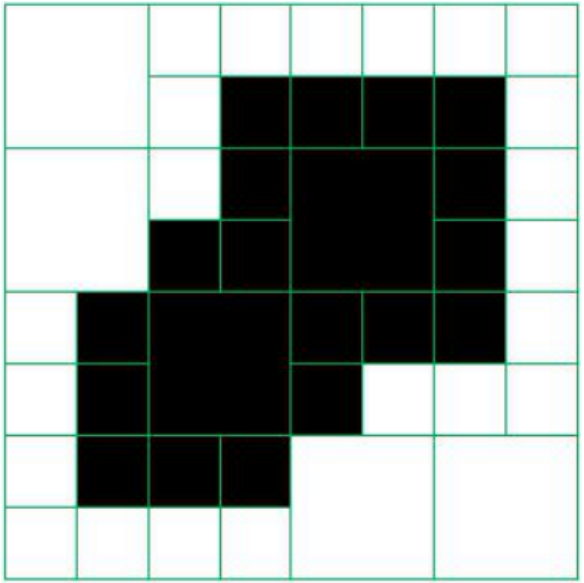
Open Problems

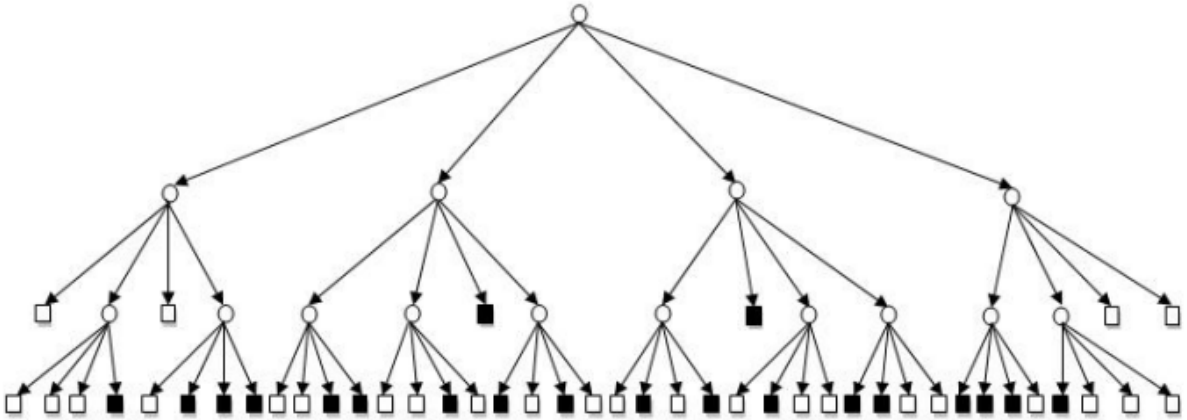
A good choice of quadtree root node improves substantially both the quadtree representation and the final image compression.[13]

In 2011, Xiang Yin et al. proposed an algorithm CORN (Choosing an Optimal Root Node). It reduces space greatly if compared with standard method based on quadtree concept.

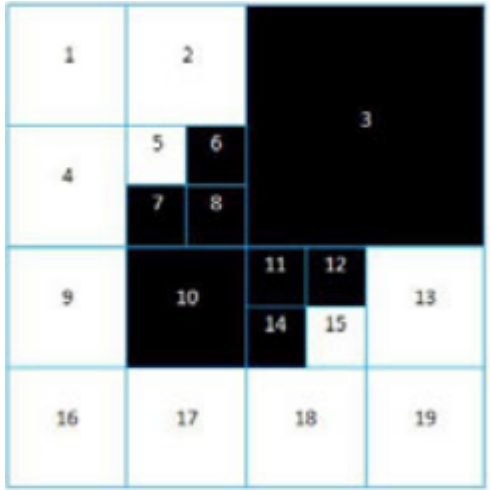
Traditionally, the root node is always placed in the center of the chosen image area or the image to be considered.

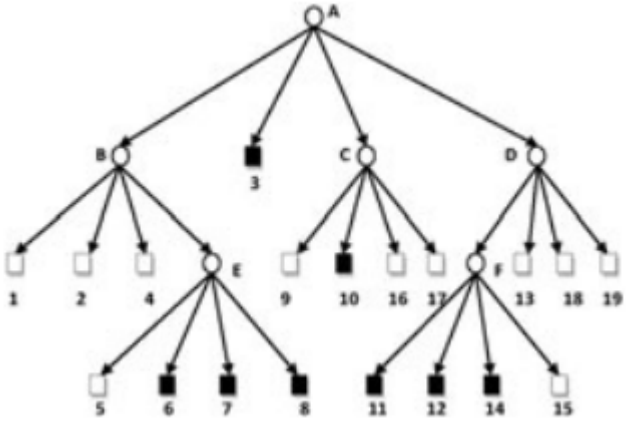
Open Problems





Open Problems





In 2012, they improved the CORN algorithm and named it ACORN, and tests have shown that ACORN can reduce space by 30–40% when compared with CORN. [13]

References

[1] Raphael Finkel and J.L. Bentley (1974). "Quad Trees: A Data Structure for Retrieval on Composite Keys". Acta Informatica 4 (1): 1–9. doi:10.1007/BF00288933.

[2] Damn Cool Algorithms: Spatial indexing with Quadtrees and Hilbert Curves. http://blog.notdot.net/2009/11/Damn-Cool-Algorithms-Spatial-indexing-with-Quadtrees-and-Hilbert-Curves

[3] Samet, Hanan; Webber, Robert (July 1985). "Storing a Collection of Polygons Using Quadtrees". Retrieved 23 March 2012.

[4] Quadtree https://en.wikipedia.org/wiki/Quadtree

[5] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf (2000). Computational Geometry (2nd revised ed.). Springer-Verlag. ISBN 3-540-65620-0. Chapter 14: Quadtrees: pp. 291–306.

[6] S. Aluru. Quadtrees and octrees. In D. Metha and S. Sahni, editors, Handbook of Data Structures and Applications, chapter 19. Chapman & Hall/CRC, 2005.

[7] D. Eppstein, M. Goodrich, and J. Sun. The skip quadtree: A simple dynamic data structure for multidimensional data. In Proc. 21st ACM Sympos. Comput. Geom., pages 296–205, 2005.

References

[8] JavaScript QuadTree Implementation http://www.mikechambers.com/blog/2011/03/21/javascript-quadtree-implementation/

[9] Quick Tip: Use Quadtrees to Detect Likely Collisions in 2D Space http://gamedevelopment.tutsplus.com/tutorials/quick-tip-use-quadtrees-to-detect-likely-collisions-in-2d-space--gamedev-374

[10] Quadtree for 2D collision detection http://stackoverflow.com/questions/4981866/quadtree-for-2d-collision-detection

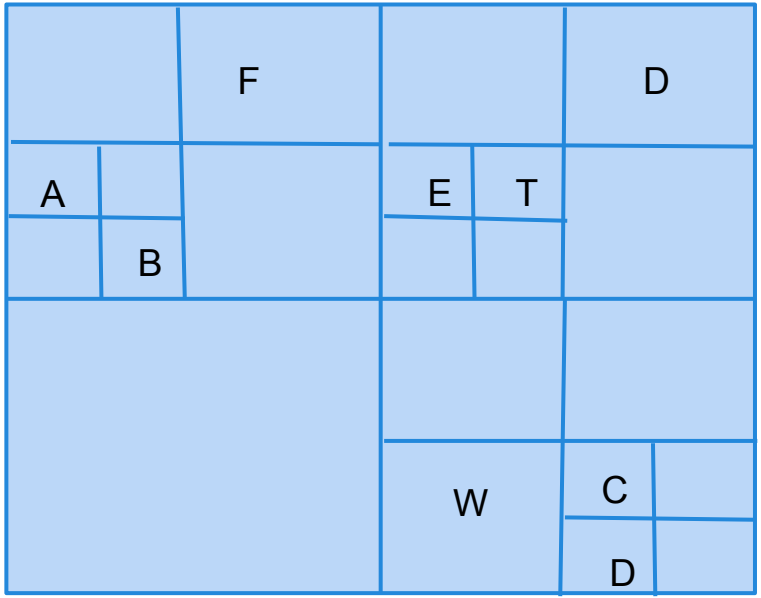
[11] Implementing Quadtree http://www.scs.gmu.edu/~fcamelli/academics/csi703_students_only/csi703_quadtree.pdf

[12] Geometric Algorithms, Lecture 3: Quadtrees http://www.win.tue.nl/~kbuchin/teaching/2IL55/slides/03quadtrees.pdf

[13] Xiang Yin, Ryszard Janicki. Optimization of Quadtree Representation and Compression. Rough Sets and Current Trends in Computing, Lecture Notes in Computer Science Volume 7413, 2012, pp 198-205

Homework

1. Draw the PR Quadtree of the following graph.



Homework

2. In 1975, Bentley developed a binary structure, the k-d tree. The k-d-tree is a binary tree. Each node contains a point which divides a single axis into two parts and thus the entire space into two sub-spaces.

(a) Does a quadtree for a set of n points always have O(n) nodes? Why or why not?

(b) Does a k-d tree for a set of n points always have O(n) nodes? Why or why not?