ELSEVIER

Information Processing Letters 49 (1994) 263-267

Information Processing Letters

Analysis of the worst case space complexity of a PR quadtree

Sriram V. Pemmaraju, Clifford A. Shaffer *

Department of Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061, USA

(Communicated by F. Dehne; received 26 April 1993)

Abstract

We demonstrate that a resolution-r PR quadtree containing n points has, in the worst case, at most $8n(r - [\log_4(n/2)]) + 8n/3 - 1/3$ nodes. This captures the fact that as n tends towards 4^r, the number of nodes in a PR quadtree quickly approaches $O(n)$. This is a more precise estimation of the worst case space requirement of a PR quadtree then has been attempted before.

Key words: Data structures; Spacial data structures; Worst case analysis; PR quadtree; Point data

1. Introduction

Hierarchical data structures, and the quadtree in particular are coming into widespread use in applications requiring the organization of large amounts of spatial data. Samet [8,9] provides a wide survey of these data structures and their applications. Examples of applications are geographic information systems, computer graphics, image processing, fluid dynamics, star cluster simulations, and robotics. However, despite the popularity of these data structures and the large number of algorithms that involve them, their time and space analysis has proved notoriously difficult. The few theoretical results that have been obtained for quadtrees do make them attractive. Hunter and Steiglitz [3] show that the space requirement for a region quadtree is $O(P)$ where P is the total perimeter length for the region represented, a finding confirmed in [1]. Samet [6], and Samet and Shaffer [7] show that the amortized cost of neighbor finding in any quadtree is O(1). In turn, Webber [11], Shaffer and Stout [10], and Lattanzi and Shaffer [5] have applied these results to show that several algorithms involving these data structures run in linear time. However, for most applications, nontrivial space and time bounds are difficult to obtain.

In this paper we present an analysis of the worst case space requirements for a data structure known as the *PR quadtree*. The PR quadtree is of more than theoretical interest as it is gaining widespread use for efficient fluid dynamics and star cluster simulations [2,4]. We believe that this analysis will also be applicable to other quadtree data structures.

The *PR quadtree* [8] is a hierarchical, variable resolution data structure based on the recursive partitioning of a bounded planar region into

* Corresponding author.

equal-sized quadrants. More precisely, consider a unit square with n points in it. If $n \geqslant 2$, split the square into four equal squares by drawing a vertical line and a horizontal line through the center. Each square so obtained is recursively split into four equal-sized squares if and only if it contains more than one point. This process terminates when every square that has not been further split contains at most one point. Therefore, at the termination of this process, among the squares that have not been split, there are some that contain no points and some that contain one point. Call a square that contains no points an *empty square* and call a square that contains one point a *full square*. The recursive decomposition of the unit square described above can be represented by a PR quadtree. A PR quadtree is a rooted tree in which each node represents a square created during the course of recursively decomposing the unit square. The root of the PR quadtree represents the unit square. If a node v represents a square S that contains two or more points, then v has four children, where each child represents a distinct square that was obtained by splitting S.

Note that each node in the PR quadtree has exactly four children or no children. Those nodes that have no children (leaves) represent full squares or empty squares. A node that represents a full square is called a *full node*, while a node that represents an empty square is called an *empty node*. The rest of the nodes of the PR quadtree are called *internal nodes* and they represent squares that contain two or more points and have been recursively split. The depth of the PR quadtree is the same as the depth of the recursion used to decompose the unit square. The depth of a PR quadtree is called its *resolution* and we call a PR quadtree of depth r, a *resolution-r* PR quadtree. This is typical of the use of PR quadtrees since data points usually have coordinates of fixed resolution. Note that a resolution-r PR quadtree can have at most $(4^{r+1} - 1)/3$ nodes and the resolution of a PR quadtree places a lower bound of 2^{-r} on the side length of squares obtained during the course of the recursive decomposition. Also note that a PR quadtree of resolution-r contains at most 4^r points.

The space requirement of a resolution-r PR quadtree is simply the number of nodes in it. But, the number of nodes in a resolution-r PR quadtree depends on the arrangement of the n points in the unit square. For example, if $n = 2$, the number of nodes in a resolution-r PR quadtree could be as small as 5, if the two points were placed far apart, or could be as large as $4r + 1$, if the two points are placed as close to each other as possible. In the former case, the PR quadtree contains one internal node, two empty nodes, and two full nodes, while in the latter case the PR quadtree contains r internal nodes, two full nodes, and $3r - 1$ empty nodes. Our goal then, is to determine a tight worst case upper bound on the space requirement of a resolution-r PR quadtree, that stores n points. We achieve this goal by constructing a *worst case* resolution-r PR quadtree storing n points i.e., a resolution-r PR quadtree storing n points whose size is the largest among all such PR quadtrees.

The analysis in the next section shows that the worst case space requirement of a resolution-r PR quadtree storing n points is

$$8n\left(r - \left\lceil \log_4\left(\frac{n}{2}\right)\right\rceil\right) + \frac{8n}{3} - \frac{1}{3}.$$

While a worst case space requirement analysis of a PR quadtree does not exist in the literature, it has been generally stated without qualification that the worst case space requirement of a resolution-r PR quadtree storing n points is $O(rn)$, where r is at least $\log_4 n$. This statement is imprecise, especially in the light of the fact that as n tends to 4^r, the space requirement of the PR quadtree must become linear. From our result, it is easy to determine how the worst case space requirement of a PR quadtree varies, as n varies with respect to r.

2. Analysis

The analysis is in two parts. In the first part, a procedure called **create_tree** is presented. This procedure constructs a worst case resolution-r PR quadtree. In the second part, the number of nodes in the tree constructed by **create_tree** is determined. Let the number of points inserted into T by **create_tree** be n. First, we assume that $n = 2 \cdot 4^k$. This assumption simplifies technical details in the analysis, while keeping the concepts intact. Subsequently, we drop this assumption and extend our analysis to an arbitrary n.

Having established the correspondence between nodes in the PR quadtree and the squares obtained by the recursive decomposition of the unit square, we shall use the two interchangeably. In particular, we shall talk about inserting a point into a node, while actually meaning that a point is being inserted into the square that the node corresponds to. A node in the PR quadtree is at level l if its distance from the root is l. Note that a node at level l represents a square of side length 2^{-l}.

```
create_tree
T is initialized to be an empty node;
Repeat 4^k times
    Insert two new points as close to each other as
    possible into an empty node at the smallest
    level in T;
    Recursively decompose the node until the new
    nodes obtained contain at most one point;
```

The intuition behind the above procedure is that a node at the smallest level (i.e., the largest square), when recursively decomposed, can generate the greatest number of new nodes. Points are placed as close to each other as possible so as to ensure that the depth of the recursive decomposition is as large as possible. Clearly, for any PR quadtree that stores n points, there exists a PR quadtree, at least as large, that stores n points and all its full nodes appear at level r. Thus we need to consider only those trees with all full nodes at level r. It is also easy to check that the PR quadtree constructed by **create_tree** is a largest quadtree among all those that store n points and have all their full nodes at level r. These two observations lead to the fact that T is a worst case, resolution-r PR quadtree that stores n points. The next two lemmas establish properties of T that will be used in counting the number of nodes in T.

Lemma 1. *For each l, $0 \leqslant l \leqslant k$, there are 4^l internal nodes in T at level l.*

Proof. The proof is by induction on the number of levels. Assume that for some l, $0 \leqslant l \leqslant k$, $2 \cdot 4^l$ points have been inserted into T according to **create_tree**. The induction hypothesis is that, at this stage, for all j, $0 \leqslant j \leqslant l$ there are 4^j internal nodes in T at level j. The claim is trivially true for $l = 0$. Assume that the claim is true for some l, $0 \leqslant l < k$. Consider the nodes at level l. Each of these nodes contains two points as close to each other as possible. Hence, each of these nodes has four children, three of them empty, and one internal. So there are a total of $3 \cdot 4^l$ empty nodes at level $l+1$ and the subsequent $3 \cdot 4^l$ pairs of points that are inserted by **create_tree** are devoted to forcing these empty nodes to recursively decompose. Therefore with the insertion of $2 \cdot 3 \cdot 4^l$ additional points all the empty nodes in level $l+1$ are converted into internal nodes. Therefore after inserting a total of $2 \cdot 4^l + 2 \cdot 3 \cdot 4^l = 2 \cdot 4^{l+1}$ points according to **create_tree**, there are 4^{l+1} internal nodes in level $l+1$. □

Lemma 2. *Each node at level k is the root of a subtree that contains $4(r-k)+1$ nodes.*

Proof. Consider a node v at level k. The subtree rooted at v has height $r-k$ and at each level l, $k < l < r$, there are three empty nodes and one internal node. At level r there are two empty nodes and two full nodes. Therefore v has $4 \cdot (r-k)$ descendents yielding a total of $4(r-k)+1$ nodes in the subtree rooted at v. □

Theorem 3. *Suppose $n = 2 \cdot 4^k$, for some k, $0 \leqslant k < r$. Let T be a resolution-r PR quadtree that stores n points and has the maximum number of nodes. Then the number of nodes in T is*

$$2n(r - \log_4 n) + \frac{5n}{3} - \frac{1}{3}.$$

Proof. According to Lemma 1 the number of nodes in T at level k or less is $\sum_{l=0}^{k} 4^l$. According to Lemma 2 the number of nodes in T at level

$k+1$ or more is $4^k \cdot 4(r-k)$. Therefore the total number of nodes is

$$= \sum_{l=0}^{k} 4^l + 4^k \cdot 4(r-k)$$

$$= \frac{4^{k+1}-1}{3} + 4^{k+1}(r-k).$$

Simplification yields

$$4^{k+1}(r-k) + \frac{4^{k+1}}{3} - \frac{1}{3}.$$

Substituting $n = 2 \cdot 4^k$ and $k = \log_4(n/2)$ we get

$$= 2n\left(r - \log_4\left(\frac{n}{2}\right)\right) + \frac{2n}{3} - \frac{1}{3}$$

$$= 2n(r - \log_4 n) + \frac{5n}{3} - \frac{1}{3}. \quad \square$$

We now drop the assumption that $n = 2 \cdot 4^k$ and extend Theorem 3.

Theorem 4. *A resolution-r PR quadtree that stores n points, contains at most*

$$8n\left(r - \left\lceil \log_4\left(\frac{n}{2}\right)\right\rceil\right) + \frac{8n}{3} - \frac{1}{3}$$

nodes.

Proof. There exists a k, $1 \leqslant k \leqslant r$, such that $2 \cdot 4^{k-1} \leqslant n \leqslant 2 \cdot 4^k$. From the proof of Theorem 3, it follows that the number of nodes in the PR quadtree T created by procedure **create_tree** is at most

$$4^{k+1}(r-k) + \frac{4^{k+1}}{3} - \frac{1}{3}.$$

Using $k = \lceil \log_4(n/2) \rceil$ and $2 \cdot 4^{k-1} \leqslant n$ we obtain that T contains at most

$$8n\left(r - \left\lceil \log_4\left(\frac{n}{2}\right)\right\rceil\right) + \frac{8n}{3} - \frac{1}{3}$$

nodes. $\square$

For practical applications, we are interested in the worst case space complexity of the PR quadtree as n varies with respect to r. The following two corollaries are immediate from the above theorem.

Corollary 5. *if $n = c \cdot 4^r$, where $0 \leqslant c \leqslant 1$, then T has at most*

$$8n\left(\frac{1}{3} - \left\lceil \log_4\left(\frac{c}{2}\right)\right\rceil\right) - \frac{1}{3}$$

nodes in it.

Corollary 6. *If $n = (4^r)^c$, where $0 \leqslant c \leqslant 1$, then T has at most*

$$8n\left(r(1-c) + \tfrac{4}{3}\right) - \tfrac{1}{3}$$

nodes in it.

The first corollary confirms the fact that if n is a constant fraction of 4^r, then the worst case space requirement of a PR quadtree is $O(n)$. The second corollary shows that if n is a fractional power of 4^r then the worst case space requirement of a PR quadtree can be as bad as $O(n \cdot \log n)$.

Finally, note that the above analysis can be easily extended to PR trees of higher dimensions.

Corollary 7. *A PR tree of dimension d, resolution r, and storing n points has at most*

$$2^{d+1}n\left(r - \left\lceil \log_{2^d}\frac{n}{2}\right\rceil\right) + \frac{2^{d+1}n}{3} - \frac{1}{3}$$

nodes.

3. Acknowledgement

We thank Lenwood S. Heath for his helpful comments.

4. References

[1] F.W. Burton, V.J. Kollias and J.G. Kollias, Expected and worst-case storage requirements for quadtrees, *Pattern Recognition Lett.* **3** (1985) 131–135.

[2] L.F. Greengard, The rapid evaluation of potential fields in particle systems, Ph.D. Dissertation, Dept. of Computer Science, Yale University, 1987.

[3] G. Hunter and K. Steiglitz, Operations on images using quadtrees, *IEEE Trans. Pattern Analysis and Machine Intelligence* **1** (1979) 145–153.

[4] P. Hut and S. McMillin, eds., The use of supercomputers in stellar dynamics, in: *Proc. Workshop held at the Institute for Advanced Study*, Princeton, NJ, June 1984.

[5] M. Lattanzi and C.A. Shaffer, An optimal boundary to quadtree conversion algorithm, *Computer Vision, Graphics, and Image Processing* **53** (1991) 303–311.

[6] H. Samet, Neighbor finding techniques for images represented by quadtrees, *Computer Graphics Image Processing* **18** (1982) 37–57.

[7] H. Samet and C.A. Shaffer, A model for the analysis of neighbor finding in pointer-based quadtrees, *IEEE Trans. Pattern Analysis and Machine Intelligence* **7** (1985) 717–720.

[8] H. Samet, *Applications of Spatial Data Structures; Computer Graphics, Image Processing, and GIS* (Addison-Wesley, Reading, MA, 1989).

[9] H. Samet, *The Design and Analysis of Spatial Data Structures* (Addison-Wesley, Reading, MA, 1990).

[10] C.A. Shaffer and Q.F. Stout, Linear time distance transforms for quadtrees, *Computer Vision, Graphics, and Image Processing: Image Understanding* **54** (1991) 215–223.

[11] R.E. Webber, Analysis of quadtree algorithms, Ph.D. Dissertation, TR-1376, Computer Science Dept., University of Maryland, College Park, MD, 1984.