Amortized Analysis of Balanced Quadtrees

Huck Bennett *   Chee Yap †

October 16, 2013

Abstract

Quadtrees are a well-known data structure for representing geometric data in the plane. A quadtree is called *balanced* if any two adjacent leaf boxes differ by at most one in height. In this paper, we analyze quadtrees which maintain balance as an invariant with each split operation, called a balanced split. Our main result shows that the balanced split operation has an amortized cost of $O(1)$ time.

1 Introduction

Quadtrees [dBCvKO08, Sam90] are a well-known data structure for representing geometric data in two dimensions. Although the term "quadtree" is often overloaded to encompass many generalizations [Moo95], here we only consider the basic case corresponding to an aligned subdivision of a square. In this case there exists a natural one-to-one correspondence between quadtree nodes and boxes in the underlying subdivision. We refer the reader to Chapter 14 in [dBCvKO08] whose quadtree nomenclature we follow.

Two boxes (or associated nodes in a quadtree) are *adjacent* if the boxes share an edge, but have disjoint interiors. Two boxes that are adjacent are called *neighbors*.

We follow [dBCvKO08] in calling a quadtree *balanced* if any two adjacent leaf boxes differ by at most one in height. The motivation for balanced quadtrees comes from multiple domains, including good mesh generation [dBCvKO08] and motion planning [WCY13].

One important motivation is to maintain pointers to adjacent boxes to ensure efficient neighbor queries. Since a box can have $\Theta(n)$ neighbors in a tree of size n, we must not maintain explicit pointers to each neighbor. We associate 4 pointers with each node which point to adjacent boxes $u.D$ ($D \in \{N, S, E, W\}$), one in each of the 4 compass directions. Box $u.D$ has depth at most $depth(u)$, and shares the D-edge of u; it is uniquely determined if we require its depth to be maximum, subject to these properties. With such pointers, we can easily list all the neighbors of a box in $O(1)$ time per neighbor.

Besides neighbor queries, our quadtrees are dynamic and support the `split` operation at any specified leaf u. After splitting, the box at u is divided into four congruent subboxes which become children of u (u is no longer a leaf).

This leads to the following result:

Lemma 1. *In the worst case a sequence of n splits, starting from the trivial quadtree of one node, has complexity $\Omega(n \log n)$.*

This result says that the amortized cost of splits is $\Omega(\log n)$. It raises the question: is it possible to ensure amortized $O(1)$ time for splits? This paper gives a positive answer, provided we maintain balance after each split.

This paper makes two primary contributions:

- It introduces a quadtree variant that maintains neighbor pointers with each box, and maintains balance as an invariant between splits. This allows for performing the `neighbor_query` operation in worst-case $O(1)$ time.
- It shows that maintaining balance with each split requires amortized $O(1)$ additional splitting operations.

We also discuss our implementation of the data structure and its applications in motion planning.

1.1 Related Work

A recent paper [LSS13] defines a slightly different model for balanced quadtrees, and claims that it is

*Department of Computer Science, New York University. hbennett@cs.nyu.edu

†Department of Computer Science, New York University. yap@cs.nyu.edu

	Balanced	Standard
`neighbor_query`	$O(1)$	$O(d)$
`bsplit/split`	Amortized $O(1)$	$O(1)$
`balance`	(Maintained as invariant)	$O((d+1)n)$
`point_query`	$O(d)$	$O(d)$
Space used	$O(n)$	$O(n)$

Table 1: Comparison of the balanced quadtrees described in this paper with standard quadtrees. All costs are worst-case except for splitting balanced quadtrees. We achieve improvements to the `neighbor_query` and `balance` operations at the cost of `split` requiring amortized rather than worst-case $O(1)$ time.

possible to maintain balance in this model in *worst-case* $O(1)$ time per split. We discuss this claim and present a family of examples that show that a class of related local balancing algorithms cannot ensure balance in worst-case $O(1)$ time in our model.

2 Main Results

Table 1 compares the cost of standard operations on quadtrees. We use n to denote the number of nodes in and d the depth of a quadtree T.

The following well-known theorem says that an arbitrary quadtree can be balanced using $O(n)$ splits and $O((d+1)n$ time:

Theorem 1 (Theorem 14.4 in [dBCvKO08], Theorem 3 in [Moo95])**.** *Let T be a quadtree with n nodes. Then the balanced version of T has $O(n)$ nodes and can be constructed in $O((d+1)n)$ time.*

We analyze the case where we balance after each split instead of performing an arbitrary number of splits before balancing. Let a *balanced split* operation `bsplit`(B) be a `split` of B followed by a `balance` of the resulting tree.

Intuitively a single splitting operation does not unbalance a quadtree much, so only a few additional splits should be required to rebalance a tree after one split. To show this formally one might try applying the analysis given by Theorem 1 to a sequence of balanced splits `bsplit`$(B_1), \ldots,$ `bsplit`(B_n). However that analysis only gives a worst-case linear time bound of $O(i)$ for balancing after the ith split in a sequence `split`$(B_1), \ldots,$ `split`(B_n) where B_1 is the root box. It implies that a sequence of balanced splits `bsplit`$(B_1), \ldots,$ `bsplit`(B_n) takes $\sum_{i=1}^{n} O(i) = O(n^2)$ time, or an amortized bound of $O(n)$ per `bsplit`. Our main result is Theorem 2 which says that we can get a much better amortized bound by maintaining balance as an invariant:

Theorem 2. *The total cost of any sequence of n balanced splits, starting from the trivial quadtree of one node, is $O(n)$.*

Proof Outline: The proof requires the analysis of what we call a *forcing chain*, $B_1 \to B_2 \to \cdots \to B_k$, of boxes where $depth(B_{i+1}) = depth(B_i) - 1$. In this case a split of B_1 will force the entire chain to split. We show that `bsplit`(B) causes at most two such chain splits for any box B. The charging of these chain splits is subtle: in the worst case, we only have to pay directly for the splits of B_1, B_k and some suitably identified B_i $(1 < i < k)$. The remaining splits can be charged to nodes in the quadtree.

Remark 1. Theorem 2 is stronger than Theorem 1 and implies it as a corollary.

References

[dBCvKO08] Mark de Berg, Otfried Cheong, Mark van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer, Third edition, 2008.

[LSS13] Maarten Löffler, Joseph A. Simons, and Darren Strash. Dynamic planar point location with sub-logarithmic local updates. In Frank Dehne, Roberto Solis-Oba, and Jörg-Rüdiger Sack, editors, *WADS*, volume 8037 of *Lecture Notes in Computer Science*, pages 499–511. Springer, 2013.

[Moo95] Doug Moore. The cost of balancing generalized quadtrees. In *Symposium on Solid Modeling and Applications*, pages 305–312, 1995.

[Sam90] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[WCY13] Cong Wang, Yi-Jen Chiang, and Chee Yap. On soft predicates in subdivision motion planning. In *Proceedings of the twenty-ninth annual Symposium on Computational Geometry*, SoCG '13, pages 349–358, New York, NY, USA, 2013. ACM.