A NOTE ON DYNAMIC RANGE SEARCHING

by H. Edelsbrunner

Technical University of Graz, Institut fuer
Informationsverarbeitung, Steyrergasse 17, A-8010 Graz, Austria.

This work was partially carried out while the author visited the Department of Computer Science at the University of British Columbia, Vancouver, Canada.

Abstract.

The planar range searching problem asks for the enumeration of all points of a given finite set that lie inside an orthogonal rectangle. In dynamic environments, insertions and deletions of points have to be supported as well.

This note presents an improvement of dynamic planar range searching achieving logarithmic query time and $O(\log^2 n)$ time for insertions and deletions. This improvement carries over to higher dimensions by standard techniques.

1. Introduction.

Recently, McCreight [6] introduced a surprisingly efficient data structure supporting a special kind of 'range queries'. He called the structure a treap, since it features a combination of a binary search tree and a heap.

A range query asks for all points of a given finite set which lie inside an orthogonal rectangle, i.e. a rectangle with edges

parallel to the coordinate-axes. For simplicity, we consider only the planar case.

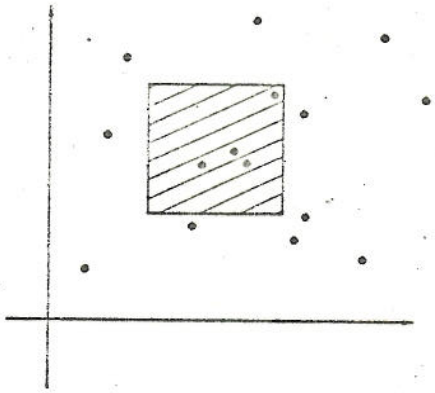

Figure 1-1. Instance of a range query.

McCreight's treap stores n planar points in O(n) space and supports dynamic range searching restricted to rectangles unbounded in one fixed direction with O(logn+t) query time and O(logn) update time (where t denotes the number of detected points) in the worst case.

So far, the best worst-case solution for dynamic range searching (with general orthogonal query rectangles) is due to a modification of the dynamic range tree developed independently by Willard [9] and Lueker [4]. Their tree answers range queries in $O(\log^2 n+t)$ time and requires O(nlogn) space and $O(\log^2 n)$ time for each update. Overmars, Leeuwen [8] improved the time bound needed for deletions to O(logn).

In the next section, we present as main result of this note an improvement of the query time for range searching. Notice that all results stated carry over to higher dimensions by standard techniques described e.g. in Edelsbrunner [2]. Each additional dimension implies an additional factor logn in query time, space, insertion time, and deletion time. Hence, our improvement also implies an improvement beyond two dimensions.

2. Main Result.

As mentioned above the best solution, so far, for dynamic range searching answers queries in $O(\log^2 n+t)$ time and requires $O(n\log n)$ space, $O(\log^2 n)$ time to insert a new point, and $O(\log n)$ time to delete a point. As main result of this note, the time required to answer a query is improved by simultaneously increasing the time required to delete a point.

Theorem: Dynamic range searching with n points in the plane can be solved with query time $O(\log n+t)$, space $O(n\log n)$, and insertion and deletion time $O(\log^2 n)$, where t denotes the number of points reported. All bounds state worst-case complexities.

Proof: The above assertion is proved by explicitely exhibiting the data structure achieving these bounds. It consists of two components, a range tree as first component and a number of treaps attached to the nodes of the range tree as second component. This

is our motivation to term the structure the RT-tree.

Let us first describe the static shape of the RT-tree. The dynamic behaviour is then achieved by replacing the optimally balanced trees by certain kinds of balanced trees.

The first component of the RT-tree is an optimally balanced binary tree which stores the points of S sorted with respect to the x-coordinates in its leaves. Each inner node contains some information to guide searches down the tree. Additionally, each inner node (except for the root) is augmented with a treap storing all points which appear in the leaves of the subtree rooted at the inner node.

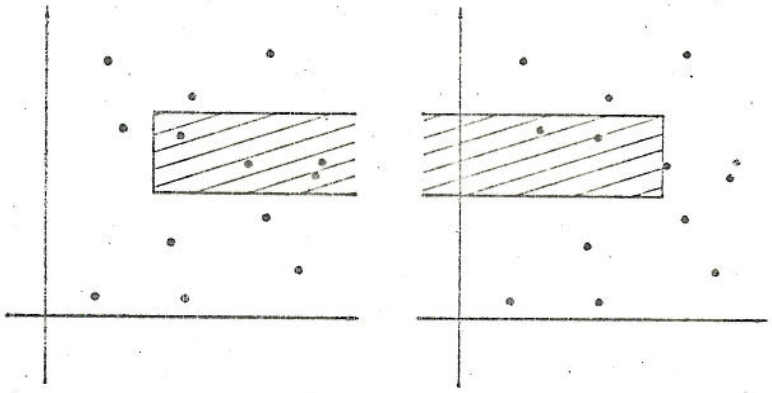

Figure 2-1. Queries supported by left son and right son treaps.

Let k be the left son of some node in the tree. Then the attached treap stores the points in k's subtree such that range queries with to the right unbounded ranges are accommodated.

If k is the right son of some node then the treap is built to accommodate range queries with to the left unbounded ranges.

The space required to store n points is, of course, proportional to nlogn.

A range query with general query rectangle can be answered as follows:

Start at the root of the RT-tree. If only one subtree of the current node has points with x-coordinates in the x-projection of the query rectangle then examine this subtree recursively - otherwise involve two appropriate queries in the treaps associated with the two sons of the node.

The path in the binary tree of the first component consists of no more than logn nodes and the examination of two treaps costs at most time O(logn+t). Consequently, the RT-tree answers range queries in time O(logn+t).

The dynamic behaviour stated in the theorem is achieved by replacing the static treaps by dynamic ones, and replacing the optimally balanced binary tree by a weight balanced binary tree. McCreight showed that half-balanced binary trees due to Olivie [7] yield dynamic treaps requiring time O(logn) for an update. ditionally, Willard [10] developed a strategy to update an augmented weight balanced tree in time O(wlogn), where w denotes the worst-case update time of the attached structures. Consequently, the dynamic RT-tree, composed of the balanced trees mentioned above, can be maintained in time $O(\log^2 n)$ for each update without increasing the asymptotic bounds for space and

answering a query. This completes the proof.

As mentioned in the introduction, the improvement carries over by standard techniques to three and higher dimensions. (Consult Edelsbrunner [2] for details.) More specifically, d-dimensional range searching can be solved by $R^{d-1}T$-trees, which are multidimensional generalization of the RT-trees. Queries are answered in time $O(\log^{d-1} n+t)$, $O(n\log^{d-1} n)$ space is needed, and each insertion or deletion is accomplished in time $O(\log^{d} n)$.

3. Discussion.

In this note we presented an improvement of dynamic planar range searching. Logarithmic time for answering queries is shown to suffice together with $O(\log^2 n)$ update time.

As an interesting detail note that the presented structure cannot be used to determine the number of points inside a query rectangle efficiently. The strategy of combining the enumeration of a point with the check: "Has the 'next' point to be reported as well?" makes this impossible. A similar trick was recently exploited for the design of a linear space structure which supports interval intersection searching in logarithmic time, see Edelsbrunner [1] and McCreight [5].

A challenging question is to provide lower bound arguments for dynamic range searching. Although Fredman [3] developed lower bound arguments for dynamic range searching with answers in a

semigroup, his results seem not to apply in their strength to the enumeration version dealt with in this note.

References.

[1] Edelsbrunner,H. A New Approach to Rectangle Intersection. Submitted for publication.

[2] Edelsbrunner,H. Dynamic Data Structures for Orthogonal Intersection Queries. Technical University of Graz, Institut fuer Informationsverarbeitung (1980), Report F59.

[3] Fredman,M.L. A Lower Bound on the Complexity of Orthogonal Range Queries. To appear in Journal of the ACM.

[4] Lueker,G.S. A Transformation for Adding Range Restriction Capability to Dynamic Data Structures for Decomposable Searching Problems. University of California, Irvine, Department of Information and Computer Science (1979), Report 129.

[5] McCreight,E.M. Efficient Algorithms for Enumerating Intersecting Intervals and Rectangles. XEROX Palo Alto Research Center (1980), Report CSL-80-9.

[6] McCreight,E.M. In preparation.

[7] Olivie,H. A Study of Balanced Binary Trees and Balanced One-Two Trees. University of Antwerp (1980), Ph.D.Thesis.

[8] Overmars,M.H.; Leeuwen,J.v. Dynamization of Decomposable Searching Problems Yielding Good Worst-Case Bounds. University of Utrecht, Department of Computer Science (1980), Report RUU-CS-80-6.

[9] Willard,D.E. New Data Structures for Orthogonal Queries. To appear in Communications of the ACM.

[10] Willard,D.E. An Introduction to Super-B-Trees. University of Iowa, Department of Computer Science (1979), manuscript.