Range Trees

Lumi Halitjaha

12/11/2017

Range searching problems are problems that, typically, have the following form: *Let S be a set of points in $\mathbb{R}^d$, how do we pre-process the set S so that we can quickly report and count the points inside a specific query region?* [3] Common data structures used to solve this type of problems are Sequential Scans, Projections, Cells, k-d trees, k-ranges, and Range trees. [1, p. 398] In this write-up I will discuss Range trees(usually referred to as *Orthogonal Range Trees*)

Range trees are balanced binary trees in which each node has an additional structure attached to it. [5] They have been independently invented by D. E. Willard, J. L. Bentley, D. T. Lee and C. K. Wong, and G. S. Lueker. [6] [1, p. 404] More formally, *a range tree is a static structure that supports d-dimensional orthogonal range queries in a set of d-dimensional points.* [2] Range trees are defined recursively. The recursion is on the dimensions of the tree, that is, the range tree of n dimensions is defined in terms of the $(n-1)$-dimensional range tree. [1, p. 404]

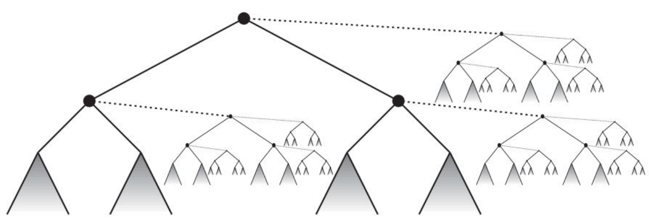

Figure 1: An example of a 3-dimensional range tree: Each node has an associated 2-dimensional tree, in which each node has an associated 1-dimensional tree. [2, p. 183]

To build a Range tree with d dimensions, we first build a 1-dimensional tree by inserting each of the n first coordinate points. We attach to each node a $(d-1)$-dimensional range tree data structure to store the other coordinates(dimensions) of the point, thus making the cost of building the tree $O(n(\log n)^d)$. [2, p. 183] As can be seen in figure 1, each tree is represented as a 1-dimensional tree of n nodes, with $d-1$-dimensional trees attached to each of these nodes, so we need $O(n)$ time to build the one dimensional tree first, and then attach the $(d-1)$-dimensional trees to each node. Thus, the space complexity of a d-dimensional tree is $O(n(\log n)^{d-1})$. [2, p. 183]

Let us now consider the query time for range trees. In a 1-dimensional binary tree, the cost of looking up a node is $O(\lg n)$. [4, p. 292]. To find the coordinates of a point in a range tree, however, we need to find all the different coordinates of the point. So it takes us $O(\lg n)$ time to find the first coordinate, then we search for the other dimensions(coordinates) of the point in the $(d-1)$-dimensional tree extending from each node, thus for each coordinate, we need to search a different one dimensional tree. Since the range searching problem requires us to output the points in an interval, it takes $O(1+k)$ time to list all the nodes belonging to the canonical interval decomposition. Therefore, the running time is $O((\log n)^d + k)$ if we are asked to output k points. [2, p. 183] Can we query faster? Yes. Since the 1-dimensional range tree is just a binary search tree, we can't do better than $\Omega(n \log n)$, because we are limited by the comparison-based model for one dimensional searching. [4, p. 292] [2, p. 184] But we can improve the query time of a 2-dimensional range tree using the technique of fractional cascading which reduces the time complexity from $O(\log n)^2 + k)$ to $O(\log n + k)$. [2, p. 184] One major advantage of range trees is that they achieve the best worst-case search time compared to the above-mentioned data structures used for range searching. [1] A disadvantage of this data-structure, though, is the high pre-processing and storage costs. [1, p. 404] For small files that need to be searched few times, range trees are inferior to Sequential scan. In addition, *k-d* trees perform better than range trees in practice since a range tree is practical mainly when it has up to 3 dimensions. [1, p. 405]

Nevertheless, Range trees have numerous applications. In many problems in computational geometry, range trees are used for reducing the time complexity of the solutions. [5] Below are two problems that can be solved using range trees:

1. Given N segments in a $2-D$ plane, parallel with the x and y-axis, we want to determine the number of intersections between them. This problem can be solved using a sweep-line list(a collection of segments that intersect the sweep line) which is implemented via a range tree. [5]

2. Assume a health-care company is introducing a health plan for all the users who are between 50 and 70 years old, and earn less than $100K$, and needs to have access to this category of users. We can solve this problem using a two-dimensional range tree whose dimensions are the users' incomes and their age. Using a $2-D$ range tree we can find in a very short time the users that fall into this range.

References

[1] J. L. Bentley and J. H. Friedman. Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409, Dec. 1979.

[2] P. Brass. *Advanced data structures*, volume 1. Cambridge University Press, Cambridge MA, 2008.

[3] B. Chazelle, J. E. Goodman, and R. Pollack. *Advances in Discrete and Computational Geometry: Proceedings of the 1996 AMS-IMS-SIAM Joint Summer Research Conference, Discrete and Computational Geometry–Ten Years Later, July 14-18, 1996, Mount Holyoke College*, volume 223. American Mathematical Soc., 1999.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, Cambridge MA, 1990.

[5] A.-G. Sturzu and C.-A. BOIANGIU. Range tree applications in computational geometry. In *Proc. 18th WSEAS International Conference on Applied Mathematics*, pages 10–12.

[6] D. E. Willard. The super-b-tree algorithm. Technical report, HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB, 1979.