



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

[Interaction](#)  
[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

[Tools](#)  
[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)

[Print/export](#)  
[Create a book](#)  
[Download as PDF](#)  
[Printable version](#)

[Languages](#)  
[Deutsch](#)  
[فارسی](#)

[Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [More](#)

# Range tree

From Wikipedia, the free encyclopedia

In [computer science](#), a **range tree** is an [ordered tree data structure](#) to hold a list of points. It allows all points within a given range to be [reported](#) efficiently, and is typically used in two or higher dimensions. Range trees were introduced by [Jon Louis Bentley](#) in 1979.<sup>[1]</sup> Similar data structures were discovered independently by Lueker,<sup>[2]</sup> Lee and Wong,<sup>[3]</sup> and Willard.<sup>[4]</sup> The range tree is an alternative to the *k*-*d* tree. Compared to *k*-*d* trees, range trees offer faster query times of (in [Big O notation](#))  $O(\log^d n + k)$  but worse storage of  $O(n \log^{d-1} n)$ , where *n* is the number of points stored in the tree, *d* is the dimension of each point and *k* is the number of points reported by a given query.

[Bernard Chazelle](#) improved this to query time  $O(\log^{d-1} n + k)$  and space complexity

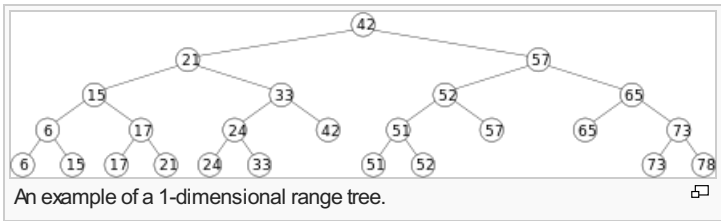
$$O\left(n \left(\frac{\log n}{\log \log n}\right)^{d-1}\right)$$
<sup>[5][6]</sup>

## Contents

- 1 Description
- 2 Operations
  - 2.1 Construction
  - 2.2 Range queries
- 3 See also
- 4 References
- 5 External links

## Description

A range tree on a set of 1-dimensional points is a balanced [binary search tree](#) on those points. The points stored in the tree are stored in the leaves of the tree; each internal node stores the largest value contained in its left subtree. A range tree on a set of points in *d*-dimensions is a [recursively defined](#) multi-level [binary search tree](#). Each level of the data structure is a binary search tree on one of the *d*-dimensions. The first level is a binary search tree on the first of the *d*-coordinates. Each vertex *v* of this tree contains an associated structure that is a (*d*−1)-dimensional range tree on the last (*d*−1)-coordinates of the points stored in the subtree of *v*.



## Operations

### Construction

A 1-dimensional range tree on a set of *n* points is a binary search tree, which can be constructed in  $O(n \log n)$  time. Range trees in higher dimensions are constructed recursively by constructing a balanced binary search tree on the first coordinate of the points, and then, for each vertex *v* in this tree, constructing a (*d*−1)-dimensional range tree on the points contained in the subtree of *v*. Constructing a range tree this way would require  $O(n \log^d n)$  time.

This can be improved by noticing that a range tree on a set 2-dimensional points can be constructed in  $O(n \log n)$  time.<sup>[7]</sup> Let *S* be a set of *n* 2-dimensional points. If *S* contains only one point, return a leaf containing that point. Otherwise, construct the associated structure of *S*, a 1-dimensional range tree on the *y*-coordinates of the points in *S*. Let *x<sub>m</sub>* be the median *x*-coordinate of the points. Let *S<sub>L</sub>* be the set of points with *x*-coordinate less than or equal to *x<sub>m</sub>* and let *S<sub>R</sub>* be the set of points with *x*-coordinate greater than *x<sub>m</sub>*. Recursively construct *v<sub>L</sub>*, a 2-dimensional range tree on *S<sub>L</sub>*, and *v<sub>R</sub>*, a 2-dimensional range tree on *S<sub>R</sub>*. Create a vertex *v* with left-child

$v_L$  and right-child  $v_R$ . If we sort the points by their  $y$ -coordinates at the start of the algorithm, and maintain this ordering when splitting the points by their  $x$ -coordinate, we can construct the associated structures of each subtree in linear time. This reduces the time to construct a 2-dimensional range tree to  $O(n \log n)$ , which also reduces the time to construct a  $d$ -dimensional range tree to  $O(n \log^{d-1} n)$ .

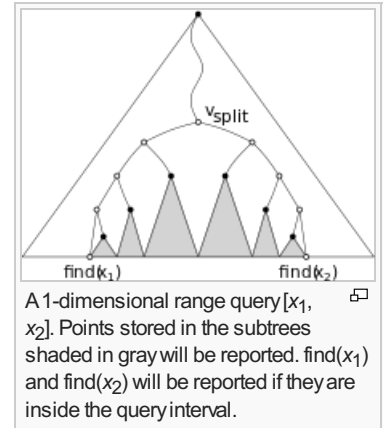
## Range queries [\[edit\]](#)

Range trees can be used to find the set of points that lie inside a given interval. To report the points that lie in the interval  $[x_1, x_2]$ , we start by searching for  $x_1$  and  $x_2$ . At some vertex in the tree, the search paths to  $x_1$  and  $x_2$  will diverge. Let  $v_{\text{split}}$  be the last vertex that these two search paths have in common. Continue searching for  $x_1$  in the range tree. For every vertex  $v$  in the search path from  $v_{\text{split}}$  to  $x_1$ , if the value stored at  $v$  is greater than  $x_1$ , report every point in the right-subtree of  $v$ . If  $v$  is a leaf, report the value stored at  $v$  if it is inside the query interval.

Similarly, reporting all of the points stored in the left-subtrees of the vertices with values less than  $x_2$  along the search path from  $v_{\text{split}}$  to  $x_2$ , and report the leaf of this path if it lies within the query interval.

Since the range tree is a balanced binary tree, the search paths to  $x_1$  and  $x_2$  have length  $O(\log n)$ . Reporting all of the points stored in the subtree of a vertex can be done in linear time using any [tree traversal](#) algorithm. It follows that the time to perform a range query is  $O(\log n + k)$ , where  $k$  is the number of points in the query interval.







Range queries in  $d$ -dimensions are similar. Instead of reporting all of the points stored in the subtrees of the search paths, perform a  $(d-1)$ -dimensional range query on the associated structure of each subtree. Eventually, a 1-dimensional range query will be performed and the correct points will be reported. Since a  $d$ -dimensional query consists of  $O(\log n)$   $(d-1)$ -dimensional range queries, it follows that the time required to perform a  $d$ -dimensional range query is  $O(\log^d n + k)$ , where  $k$  is the number of points in the query interval. This can be reduced to  $O(\log^{d-1} n + k)$  using the technique of [fractional cascading](#).<sup>[2][4][7]</sup>



See also [\[edit\]](#)

- $k$ -d tree
- Segment tree

## References [\[edit\]](#)

1. <sup>A</sup> Bentley, J. L. (1979). "Decomposable searching problems". *Information Processing Letters* **8** (5): 244–251. doi:[10.1016/0020-0190\(79\)90117-0](https://doi.org/10.1016/0020-0190(79)90117-0) 
2. <sup>A</sup> <sup>b</sup> Lueker, G. S. (1978). "A data structure for orthogonal range queries". *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*. pp. 28–21. doi:[10.1109/SFCS.1978.1](https://doi.org/10.1109/SFCS.1978.1) 
3. <sup>A</sup> Lee, D. T.; Wong, C. K. (1980). "Quintary trees: A file structure for multidimensional database systems". *ACM Transactions on Database Systems* **5** (3): 339. doi:[10.1145/320613.320618](https://doi.org/10.1145/320613.320618) 
4. <sup>A</sup> <sup>b</sup> Willard, Dan E. *The super-b-tree algorithm* (Technical report). Cambridge, MA: Aiken Computer Lab, Harvard University. TR-03-79.
5. <sup>A</sup> Chazelle, Bernard (1990). "Lower Bounds for Orthogonal Range Searching: I. The Reporting Case"  (PDF). *ACM* **37**: 200–212.
6. <sup>A</sup> Chazelle, Bernard (1990). "Lower Bounds for Orthogonal Range Searching: II. The Arithmetic Model"  (PDF). *ACM* **37**: 439–463.
7. <sup>A</sup> <sup>a</sup> <sup>b</sup> de Berg, Mark; Cheong, Otfried; van Kreveld, Marc; Overmars, Mark (2008). *Computational Geometry*. doi:[10.1007/978-3-540-77974-2](https://doi.org/10.1007/978-3-540-77974-2)  ISBN 978-3-540-77973-5.

## External links [[edit](#)]

- [Range and Segment Trees](#) in [CGAL](#), the Computational Geometry Algorithms Library.
- [Lecture 8: Range Trees](#), Marc van Kreveld.

| v · t · e  | Tree data structures   | [hide] |
|--|--|--------|
| <b>Search trees</b><br>(dynamic sets/associative arrays) | 2–3 · 2–3–4 · AA · (a,b) · AVL · B · B+ · B* · B <sup>x</sup> · (Optimal) Binary search · Dancing · HTree · Interval · Order statistic · (Left-leaning) Red-black · Scapegoat · Splay · T · Treap · UB · Weight-balanced |        |
| <b>Heaps</b>   | Binary · Binomial · Fibonacci · Leftist · Pairing · Skew · Van Emde Boas   |        |
| <b>Tries</b>   | Hash · Radix · Suffix · Ternary search · X-fast · Y-fast   |        |

|  |  |
|--|--|
| <b>Spatial data partitioning trees</b> | <a href="#">BK</a> · <a href="#">BSP</a> · <a href="#">Cartesian</a> · <a href="#">Hilbert R</a> · <a href="#">k-d (implicit k-d)</a> · <a href="#">M</a> · <a href="#">Metric</a> · <a href="#">MMP</a> · <a href="#">Octree</a> · <a href="#">Priority R</a> · <a href="#">Quad</a> · <a href="#">R</a> · <a href="#">R+</a> · <a href="#">R*</a> · <a href="#">Segment</a> · <a href="#">VP</a> · <a href="#">X</a>                                     |
| <b>Other trees</b>                     | <a href="#">Cover</a> · <a href="#">Exponential</a> · <a href="#">Fenwick</a> · <a href="#">Finger</a> · <a href="#">Fusion</a> · <a href="#">Hash calendar</a> · <a href="#">iDistance</a> · <a href="#">K-ary</a> · <a href="#">Left-child right-sibling</a> · <a href="#">Link/cut</a> · <a href="#">Log-structured merge</a> · <a href="#">Merkle</a> · <a href="#">PQ</a> · <b><a href="#">Range</a></b> · <a href="#">SPQR</a> · <a href="#">Top</a> |

Categories: [Trees \(data structures\)](#) | [Geometric data structures](#)

This page was last modified on 29 August 2015, at 07:23.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#)
[About Wikipedia](#)
[Disclaimers](#)
[Contact Wikipedia](#)
[Developers](#)
[Mobile view](#)

