Order statistic tree

From Wikipedia, the free encyclopedia

In computer science, an **order statistic tree** is a variant of the binary search tree (or more generally, a B-tree[1]) that supports two additional operations beyond insertion, lookup and deletion:

- Select(*i*) — find the *i*'th smallest element stored in the tree
- Rank(*x*) – find the rank of element *x* in the tree, i.e. its index in the sorted list of elements of the tree

Both operations can be performed in $O(\log n)$ time in the average case; when a self-balancing tree is used as the base data structure, this bound also applies in the worst case.

To turn a regular search tree into an order statistic tree, the nodes of the tree need to store one additional value, which is the size of the subtree rooted at that node (i.e., the number of nodes below it). All operations that modify the tree must adjust this information to preserve the invariant that

```
size[x] = size[left[x]] + size[right[x]] + 1
```

where `size[nil] = 0` by definition. Select can then be implemented as[2]:342

```
function Select(t, i)
    // Returns the i'th element (zero-indexed) of the elements in t
    r ← size[left[t]]
    if i = r
        return key[t]
    else if i < r
        return Select(left[t], i)
    else
        return Select(right[t], i - (r + 1))
```

Rank can be implemented as[3]:342

```
function Rank(T, x)
    // Returns the position of x (one-indexed) in the linear sorted list of elements
of the tree T
    r ← size[left[x]] + 1
    y ← x
    while y ≠ T.root
         if y = right[y.p]
              r ← r + size[left[y.p]] + 1
         y ← y.p
    return r
```

Order-statistic trees can be further amended with bookkeeping information to maintain balance (e.g., tree height can be added to get an order statistic AVL tree, or a color bit to get a red-black order statistic tree). Alternatively, the size field can be used in conjunction with a weight-balancing scheme at no additional storage cost.[4]

Another way to implement an order statistic tree is an implicit data structure derived from the min-max heap.[5]

References [edit]

1. ^ "Counted B-Trees". 11 December 2004. Retrieved 18 January 2014.
2. ^ Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001) [1990]. *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill. ISBN 0-262-03293-7.
3. ^ Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009) [1990]. *Introduction to Algorithms* (3rd ed.). MIT Press and McGraw-Hill. ISBN 0-262-03384-4.
4. ^ Roura, Salvador (2001). *A new method for balancing binary search trees*. ICALP. Lecture Notes in Computer Science **2076**. pp. 469–480. doi:10.1007/3-540-48224-5_39. ISBN 978-3-540-42287-7.
5. ^ Atkinson, M. D.; Sack, J.-R.; Santoro, N.; Strothotte, T. (1986). "Min-Max Heaps and Generalized Priority

Queues" (PDF). *Communications of the ACM*.

External links [edit]

- Order statistic tree on PineWiki, Yale University.
- The Python package blist uses order statistic B-trees to implement lists with fast insertion at arbitrary positions.

v · t · e	Tree data structures [hide]
Search trees (dynamic sets/associative arrays)	2–3 · 2–3–4 · AA · (a,b) · AVL · B · B+ · B* · B^x · (Optimal) Binary search · Dancing · HTree · Interval · **Order statistic** · (Left-leaning) Red-black · Scapegoat · Splay · T · Treap · UB · Weight-balanced
Heaps	Binary · Binomial · Fibonacci · Leftist · Pairing · Skew · Van Emde Boas
Tries	Hash · Radix · Suffix · Ternary search · X-fast · Y-fast
Spatial data partitioning trees	BK · BSP · Cartesian · Hilbert R · *k*-d (implicit *k*-d) · M · Metric · MVP · Octree · Priority R · Quad · R · R+ · R* · Segment · VP · X
Other trees	Cover · Exponential · Fenwick · Finger · Fusion · Hash calendar · iDistance · K-ary · Left-child right-sibling · Link/cut · Log-structured merge · Merkle · PQ · Range · SPQR · Top

This algorithms or data structures-related article is a stub. You can help Wikipedia by expanding it.

Categories: Search trees | Selection algorithms | Algorithms and data structures stubs | Computer science stubs