4. Augmented AVL Tree

2018年10月2日 19:28

Rank an element x of a set $S \in \mathbb{Z}$ has rank r IFF there are exactly $r-1$ elements of S less than x

$rank_S(x) = |\{ y \in S \mid y < x \}| + 1$

Augmenting AVL Tree with Operations

$rank(v)$ return the rank of $v \in S$ given its pointer

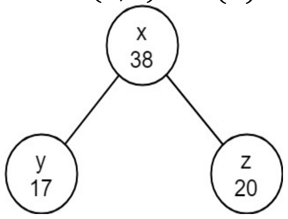
$select(r)$ return the key of rank r

Implementation

Adding a field $size(v)$ #nodes in subtrees rooted at x.

Note $v.size = v.lchild.size + v.rchild.size + 1$

$select(v, r) \in O(h)$



Example

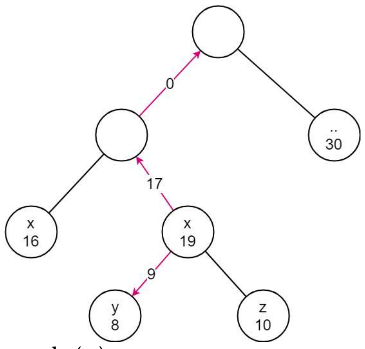
$select(x, 8) \rightarrow select(y, 8)$

$select(x, 18) \rightarrow x$

$select(x, 30) \rightarrow select(z, 12)$

```
select(v, r):
    if r = v. left. size+1
        return v
    if r < v. left. size +1
        return select(v. left, r)
    else
        return select(v. right, r - (x. left. size))
```

$rank(x) \in O(h)$



```
rank (x):
    if x is root:
        return x. left. size + 1
    if x is left child of its parent:
        return rank(x. parent)
    else
        return rank(x. parent) + x. parent. left. size +1
```

$\boldsymbol{insert(v) \in O(h)}$

Do a BST insert, and along the tracing process, add 1 to size of each node to update it
Update balance factor by rotate, then update the size, note that updating size take constant time (updating the root and only its children)

Notice when augmenting a ADT

Not too many auxiliary fields should change when inserting and deleting
Should be able to compute auxiliary field from fields of the children + keys