

# 替罪羊树

## 算法简介

一种平衡树，思维简单，码量少，速度还行，而且没有旋转操作。

## 算法思想

首先替罪羊树的插入类似普通的二叉搜索树，删除操作就是打上删除标记。

但只有这样显然不能保证替罪羊树的平衡度。

替罪羊树的核心操作是重构操作，当树的不平衡度大于一个范围时就考虑对一棵子树进行重构。

重构方法为中序遍历子树，将没有打上删除标记的结点加入序列。得到一个有序序列，然后用类似线段树建树的方法重新建树。

重构操作虽然单次复杂度为  $O(n)$  但可以证明均摊复杂度为  $O(\log n)$  证明方法自行百度。

问题在于何时考虑重构。

考虑维护每个结点所在子树的未被删除的结点个数  $\text{cnt}$  和结点总数  $\text{tot}$

引入一个平衡因子  $\alpha$  值，当  $\alpha \text{last} \text{cnt} \lt \max(\text{cnt}_{\text{lson}}, \text{cnt}_{\text{rson}})$  时考虑重构。

$\alpha$  过大将导致树的平衡度较差，查询效率低  $\alpha$  过小将导致树的重构次数过多，插入、删除效率低。

因此  $\alpha$  值通常会设置成  $0.7 \sim 0.8$  可以根据题目要求自行调整。

同时，如果一棵树上被删除的无效结点过多，也会影响查找效率，所以也需要重构。

这里设置为当  $\alpha \text{last} \text{tot} \lt \text{cnt}$  时考虑重构。

## 代码模板

[洛谷p3369](#)



From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:%E6%9B%BF%E7%BD%AA%E7%BE%8A%E6%A0%91&rev=1592877254](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E6%9B%BF%E7%BD%AA%E7%BE%8A%E6%A0%91&rev=1592877254)

Last update: 2020/06/23 09:54

