Treap

**来自"NOCOW"**

跳转到: [导航](http://www.nocow.cn/index.php/Treap#column-one), [搜索](http://www.nocow.cn/index.php/Treap#searchInput)

**Treap**，就是有另一个随机数满足[堆](http://www.nocow.cn/index.php/%E5%A0%86" \o "堆)的性质的[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)，其结构相当于以随机顺序插入的二叉搜索树。其基本操作的期望复杂度为O(log *n*)。

其特点是实现简单，效率高于[伸展树](http://www.nocow.cn/index.php/%E4%BC%B8%E5%B1%95%E6%A0%91)并且支持大部分基本功能，性价比很高。

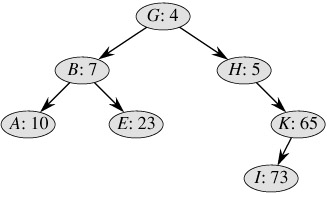
|  |
| --- |
| **目录**  [[隐藏](javascript:toggleToc())]   * [1 前言](http://www.nocow.cn/index.php/Treap#.E5.89.8D.E8.A8.80) * [2 介绍](http://www.nocow.cn/index.php/Treap#.E4.BB.8B.E7.BB.8D) * [3 操作](http://www.nocow.cn/index.php/Treap#.E6.93.8D.E4.BD.9C)   + [3.1 插入](http://www.nocow.cn/index.php/Treap#.E6.8F.92.E5.85.A5)   + [3.2 删除](http://www.nocow.cn/index.php/Treap#.E5.88.A0.E9.99.A4)   + [3.3 查找](http://www.nocow.cn/index.php/Treap#.E6.9F.A5.E6.89.BE)   + [3.4 分离](http://www.nocow.cn/index.php/Treap#.E5.88.86.E7.A6.BB)   + [3.5 合并](http://www.nocow.cn/index.php/Treap#.E5.90.88.E5.B9.B6) * [4 参考程序](http://www.nocow.cn/index.php/Treap#.E5.8F.82.E8.80.83.E7.A8.8B.E5.BA.8F) * [5 算法分析](http://www.nocow.cn/index.php/Treap#.E7.AE.97.E6.B3.95.E5.88.86.E6.9E.90) * [6 评价](http://www.nocow.cn/index.php/Treap#.E8.AF.84.E4.BB.B7)   + [6.1 与其他结构的比较](http://www.nocow.cn/index.php/Treap#.E4.B8.8E.E5.85.B6.E4.BB.96.E7.BB.93.E6.9E.84.E7.9A.84.E6.AF.94.E8.BE.83) * [7 引用](http://www.nocow.cn/index.php/Treap#.E5.BC.95.E7.94.A8) * [8 外部链接](http://www.nocow.cn/index.php/Treap#.E5.A4.96.E9.83.A8.E9.93.BE.E6.8E.A5) |

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=1)] 前言

我们可以看到，如果一个[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)节点插入的顺序是随机的，这样我们得到的[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)大多数情况下是平衡的，即使存在一些极端情况，但是这种情况发生的概率很小，所以我们可以这样建立一棵[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)，而不必要像AVL那样旋转，可以证明随机顺序建立的[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91" \o "二叉搜索树)在期望高度是O(log *n*)，但是某些时候我们并不能得知所有的待插入节点，打乱以后再插入。所以我们需要一种规则来实现这种想法，并且不必要所有节点。也就是说节点是顺序输入的，我们实现这一点可以用Treap。

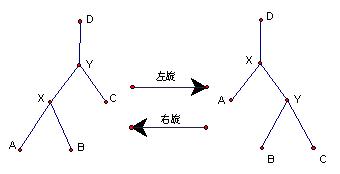
[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=2)] 介绍

Treap=Tree+Heap。Treap本身是一棵[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91" \o "二叉搜索树)，它的左子树和右子树也分别是一个Treap，和一般的[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91" \o "二叉搜索树)不同的是，Treap记录一个额外的数据，就是优先级。Treap在以关键码构成[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91" \o "二叉搜索树)的同时，还按优先级来满足[堆](http://www.nocow.cn/index.php/%E5%A0%86)的性质(在这里我们假设节点的优先级小于该节点的孩子的优先级)。但是这里要注意的是Treap和二叉堆有一点不同，就是二叉堆必须是完全二叉树，而Treap可以并不一定是。（网友注：这些图来自算法导论）

[](http://www.nocow.cn/index.php/Image:Treap01.png)

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=3)] 操作

Treap维护堆性质的方法用到了旋转，这里先简单地介绍一下。Treap只需要两种旋转，这样编程复杂度比[Splay](http://www.nocow.cn/index.php/Splay" \o "Splay)等就要小一些，这正是Treap的特色之一。

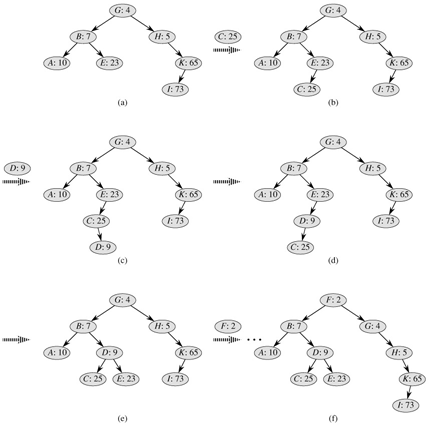
旋转是这样的： [](http://www.nocow.cn/index.php/Image:Treap_Rotate.jpg)给节点随机分配.

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=4)] **插入**

给节点随机分配一个优先级，先和[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)的插入一样，先把要插入的点插入到一个叶子上，然后跟维护堆一样，如果当前节点的优先级比根大就旋转，如果当前节点是根的左儿子就右旋如果当前节点是根的右儿子就左旋。

我们如果把插入写成递归形式的话，只需要在递归调用完成后判断是否满足堆性质，如果不满足就旋转，实现非常容易。

由于旋转是O(1)的，最多进行h次(h是树的高度),插入的复杂度是O(*h*)的，在期望情况下h=O(log *n*)，所以它的期望复杂度是O(log *n*)。

[](http://www.nocow.cn/index.php/Image:Treap02.png)"如果当前节点的优先级比根大就旋转，如果当前节点是根的左儿子就右旋如果当前节点是根的右儿子就左旋。" (好像上面的图是最小堆，而不是最大堆)

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=5)] **删除**

有了旋转的操作之后，Treap的删除比[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91" \o "二叉搜索树)还要简单。因为Treap满足堆性质，所以我们只需要把要删除的节点旋转到叶节点上，然后直接删除就可以了。具体的方法就是每次找到优先级最大的儿子，向与其相反的方向旋转，直到那个节点被旋转到了叶节点，然后直接删除。

删除最多进行O(*h*)次旋转，期望复杂度是O(log *n*)。。

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=6)] **查找**

和一般的[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)一样，但是由于Treap的随机化结构，可以证明Treap中查找的期望复杂度是O(log *n*)。

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=7)] **分离**

要把一个Treap按大小分成两个Treap，只要在需要分开的位置加一个虚拟节点，然后旋至根节点删除，左右两个子树就是得出的两个Treap了。根据[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91" \o "二叉搜索树)的性质，这时左子树的所有节点都小于右子树的节点。

时间相当于一次插入操作的复杂度，也就是O(log *n*)。

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=8)] **合并**

合并是指把两个平衡树（或者其他的有序表）合并成一个平衡树（有序表），其中第一个树（表）的所有节点都**必须小于或等于**第二个树（表）中的所有节点，这也是上面的分离操作的结果所满足的条件。

Treap的合并操作的过程和分离相反，只要加一个虚拟的根，把两棵树分别作为左右子树，然后把根删除就可以了。

时间复杂度和删除一样，也是期望O(log *n*)。

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=9)] 参考程序

[Treap\_Pascal](http://www.nocow.cn/index.php/Treap_Pascal)

[Treap\_C](http://www.nocow.cn/index.php/Treap_C)

[Treap\_C++](http://www.nocow.cn/index.php/Treap_C%2B%2B)

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=10)] 算法分析

首先我们注意到[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)有一个特性，就是每个子树的形态在优先级唯一确定的情况下都是唯一的，不受其他因素影响，也就是说，左子树的形态与树中大于根节点的值无关，右子树亦然。

这是因为Treap满足堆的性质，Treap的根节点是优先级最大的那个节点，考虑它的左子树，树根也是子树里面最大的一点，右子树亦然。所以Treap相当于先把所有节点按照优先级排序，然后插入，实质上就相当于以随机顺序建立的[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)，只不过它并不需要一次读入所有数据，可以一个一个地插入。而当这个随机顺序确定的时候，这个树是唯一的。

因此在给定优先级的情况下，只要是用符合要求的操作，通过任何方式得出的Treap都是一样的，所以不改变优先级的情况下，特殊的操作不会造成Treap结构的退化。而改变优先级可能会使Treap变得不够随机以致退化。

证明随机建立[二叉搜索树](http://www.nocow.cn/index.php/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91)的E[h]=O(log *n*)大家可以参见CLRS P265 12.4 Randomly built binary search trees，这里略去。

如果有E[h]=O(log *n*)我们就证明了，Treap插入的期望复杂度是O(log *n*)。

Treap的其它操作的期望复杂度同样是O(log *n*)。

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=11)] 评价

[[编辑](http://www.nocow.cn/index.php?title=Treap&action=edit&section=12)] **与其他结构的比较**

* [AVL树](http://www.nocow.cn/index.php/AVL%E6%A0%91)（基于长度保持平衡的二叉查找树）
* [伸展树](http://www.nocow.cn/index.php/%E4%BC%B8%E5%B1%95%E6%A0%91)（Splay Tree，一种基于查找次序优化的二叉查找树，适用于查询操作比较集中的查询集合）
* [线段树](http://www.nocow.cn/index.php/%E7%BA%BF%E6%AE%B5%E6%A0%91)（可以保存线段，往往用于统计、图形等题目）
* [红黑树](http://www.nocow.cn/index.php/%E7%BA%A2%E9%BB%91%E6%A0%91)(红黑树编程更理性一些，但编程复杂度更高)
* [SBT](http://www.nocow.cn/index.php/SBT) (SBT比TREAP的速度更快，但编程复杂度更高)