可并优先队列 之 左偏树

(2006-08-07 21:42:37)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载](javascript:;)

|  |  |
| --- | --- |
|  | 分类：[我的OI天堂](http://blog.sina.com.cn/s/articlelist_1233261541_2_1.html" \t "_blank) |

   NOI前一直忽略了leftist tree，回来后研究了一下，发现这东西还真是好用。

可并优先队列，顾名思义，就是可以合并的优先队列。

堆是一种很优秀的数据结构，编程简单，效率高，在很多场合都有其用武之地。但是一旦涉及到合并，堆的弱点就体现出来了——因为堆的设计初衷不是为了方便合并而构造的，所以其合并效率非常低下，是O(n)级别的。如果碰到合并次数比较多的话，堆的优点就几乎全部被抵消了。

左偏树（leftist tree）是堆的一种变体。为了方便介绍左偏树，先给出一些定义。

左偏树的左右子树都是左偏树。

一个节点是**外节点**，当且仅当该节点的左子树或右子树为空。

一个节点的距离**dis(p)**表示节点p到其子孙中最近外节点的距离。如果一个节点本身就是外节点，则dis(p)=0。

左偏性质：对于一棵左偏树，有dis(leftchild(root))>=dis(rightchild(ro

ot))。

由此我们可以得出：**左偏树是具有左偏性质的堆有序二叉树**。

左偏树满足以下两条性质：

1.     dis(p)=dis(rightchild(p))+1；

2.     一棵n个节点的左偏树的距离至多为?/ΣΠΑΝ>λογ(ν+1)?/ΣΠΑΝ>−1

由此我们就可以得出左偏树基本操作的算法了。

基本操作：

**合并：**当其中某棵树为空时，返回另一棵树。否则取两棵树的根节点中较小者为新树的根，合并新根的右子树和剩下的另一棵树。可以看出这是一个递归过程，复杂度是O(logn)。

**插入：**把一个节点看作一棵左偏树，并入原树即可。复杂度O(logn)。

**删除：**删掉根节点后合并左右子树。复杂度O(logn)。

**取最小值：**直接取树根即可。复杂度O(1)。

**构建一棵左偏树：**有两种算法。

朴素算法：把所有节点依次插入一棵左偏树。复杂度O(nlogn)。

队列算法：把所有节点放入一个队列，每次取队首的两棵左偏树合并后放入队尾，直到队列中只剩下一棵树为止。容易证明，这样操作的复杂度为O(n)。