**[Splay Tree – 伸展树](http://www.fcicq.net/wp/?p=107" \o "Permanent link to Splay Tree – 伸展树)**

August 9, 2006 8:36 · Filed under [SEO](http://www.fcicq.net/wp/?cat=18), [技术杂谈](http://www.fcicq.net/wp/?cat=16)

假设想要对一个二叉查找树执行一系列的查找操作。为了使整个查找时间更小，被查频率高的那些条目就应当经常处于靠近树根的位置。于是想到设计一个简单方法，在每次查找之后对树进行重构，把被查找的条目搬移到离树根近一些的地方。splay tree应运而生。splay tree是一种自调整形式的二叉查找树，它会沿着从某个节点到树根之间的路径，通过一系列的旋转把这个节点搬移到树根去。

**重构方法**

1、单旋：在查找完位于节点x中的条目i之后，旋转链接x和其父节点的边。（除非x就是树根）  
2、搬移至树根：在查找完位于节点x中的条目i之后，旋转链接x和其父节点的边，然后重复这个操作直至x成为树根。  
splay tree的重构方法和搬移至树根的方法相似，它也会沿着查找路径做自底向上的旋转，将被查找条目移至树根。但不同的是，它的旋转是成对进行的，顺序取决于查找路径的结构。为了在节点x处对树进行splay操作，我们需要重复下面的步骤，直至x成为树根为止：  
1、第一种情况：如果x的父节点p(x)是树根，则旋转连接x和p(x)的边。（这种情况是最后一步）  
2、第二种情况：如果p(x)不是树根，而且x和p(x)本身都是左孩子或者都是右孩子，则先旋转连接p(x)和x的祖父节点g(x)的边，然后再旋转连接x和p(x)的边。  
3、第三种情况：如果p(x)不是树根，而且x是左孩子，p(x)是右孩子，或者相反，则先旋转连接x和p(x)的边，再旋转连接x和新的p(x)的边。  
在节点x处进行splay操作的时间是和查找x所需的时间成比例的。splay操作不单是把x搬移到了树根，而且还把查找路径上的每个节点的深度都大致减掉了一半。

**splay tree支持的操作**

1、access(i,t):如果i在树t中，则返回指向它的指针，否则返回空指针。为了实现access(i,t)，可以从树t的根部向下查找i。如果查找操作遇到了一个含有i的节点x，就在x处进行splay操作，并返回指向x的指针，访问结束。如果遇到了空指针，表示i不在树中，此时就在最后一个非空节点处进行splay操作，然后返回空指针。如果树是空的，将忽略掉splay操作。  
2、insert(i,t):将条目i插入树t中（假设其尚不存在）。为了实现insert(i,t)，首先执行split(i,t)，然后把t换成一个由新的包含有i的根节点组成的树，这个根节点的左右子树分别是split返回的树t1和t2。  
3、delete(i,t):从树t中删除条目i（假设其已经存在）。为了实现delete(i,t)，首先执行access(i,t)，然后把t换成其左子树和右子树join之后的新树。  
4、join(t1,t2):将树t1和t2合并成一棵树，其中包含之前两棵树的所有条目，并返回合并之后的树。这个操作假设t1中的所有条目都小于t2中的条目，操作完成之后会销毁t1和t2。为了实现join(t1,t2)，首先访问t1中最大的条目i。访问结束之后，t1的根节点中包含的就是i，它的右孩子显然为空。于是把t2作为这个根节点的右子树并返回完成之后的新树即可实现join操作。  
5、split(i,t):构建并返回两棵树t1和t2，其中t1包含t中所有小于等于i的条目，t2包含t中所有大于i的条目。操作完成之后销毁t。为了实现split(i,t)，首先执行access(i,t)，然后根据新根节点中的值是大于还是小于等于i来切断这个根节点的左链接或右链接，并返回形成的两棵树。

**Splay Tree的优势所在**

由于Splay Tree仅仅是不断调整，并没有引入额外的标记，因而树结构与标准BST没有任何不同，从空间角度来看，它比Treap、Red-Black Tree、AVL要高效得多。因为结构不变，因此只要是通过左旋和右旋进行的操作对Splay Tree性质都没有丝毫影响，因而它也提供了BST中最丰富的功能，包括快速的拆分和合并（这里指的是将原树拆分成两棵子树，其中一棵子树所有节点都比另一子树小，以及它的逆过程），并且实现极为便捷。这一点是其它结构较难实现的。其时间效率也相当稳定，和Treap基本相当。