2017-07-28

AAA-Tree

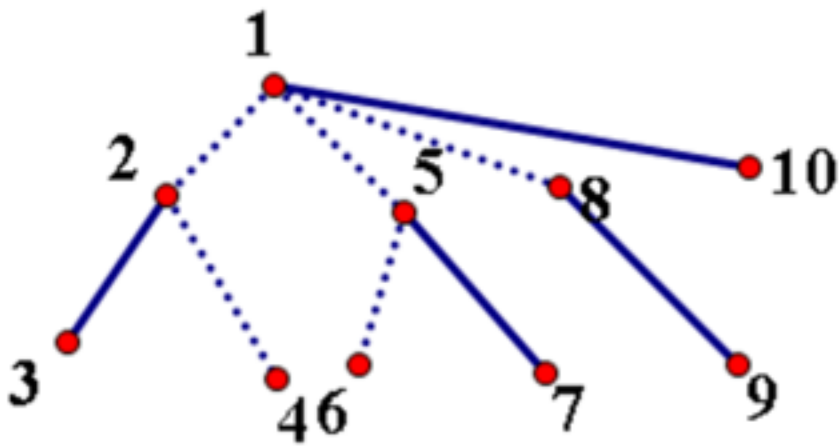
刘予希

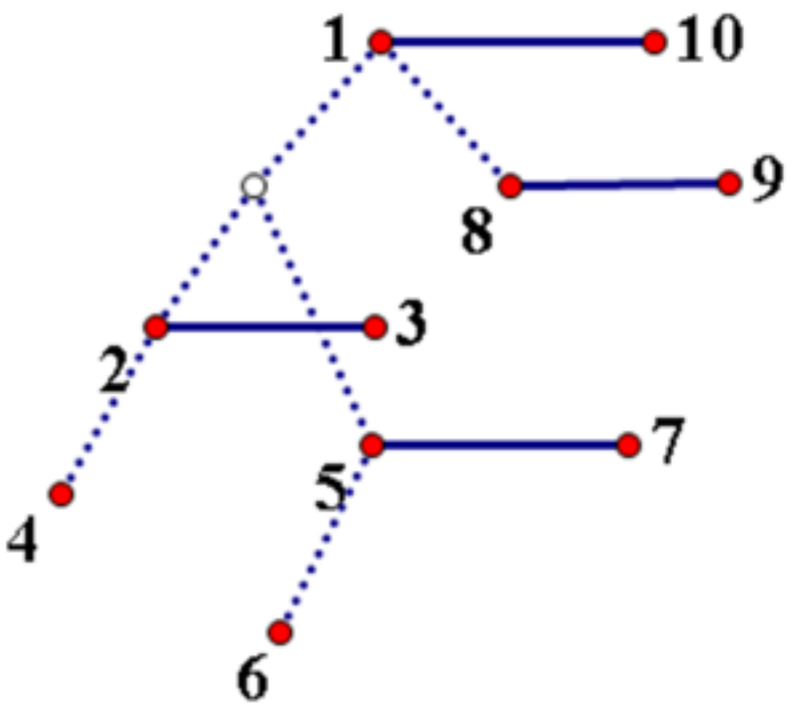
0 目录

1 基本思想

AAA-tree的名字来源于黄志翱的2014年集训队论文《浅谈动态树的相关问题及简单拓展》，可看作是一种LCT的拓展应用，在Tarjan和Werneck的论文中也被称作Self-Adjusting Top Trees，也简称作top tree。

AAA-tree的基本思想是内层虚边splay，外层AAA树结点（类似于我们的LCT）。换句话说，也就是在一般LCT的基础上，对于每一个节点，将其所有虚儿子（在树上通过虚边连接的儿子节点）用一颗splay（伸展树）来单独维护。这样做的好处，也就是AAA树的一个重要功能，就是下传子树操作的标记时，通过这棵维护虚儿子的splay来下传给虚儿子所在的子树，这要就得到了维护子树以及查询子树的方法。





2 基本操作

2.1 内层splay（对应的为无实际意义的内部点，也称白点）

内层splay需要做到因虚边实边的变化导致splay中节点的增删，同时更新内层splay维护的一个AAA节点所有虚儿子的data信息以及tag信息。主要的操作有：

2.2 外层AAA节点（对应的为原有的树的节点，外部点）

与link-cut-tree类似

3 具体操作

3.0 sone1

此次实现AAA tree是具体到 bzoj 3153 Sone1这道题目上的。

在此大致描述一下题目：一棵100000规模的树，进行100000规模的操作，该树每个节点有点权，需支持以下操作：子树赋值，换根，链赋值，子树最小值，子树最大值，子树加值，链加值，链最小值，链最大值，换父亲，链和，子树和

3.1 四个孩子

c[0]和c[1]就是外部节点splay中的左右儿子

c[2]和c[3]就是内部节点splay中的左右儿子

3.2 标记

定义了一个tag结构体，因为标记只有整体加以及整体赋值，所以可以使用非常传统的标记方法: ax + b 来表示。

对于每一个节点x，需要维护一下标记：

rev : 链翻转标记

ctag : 链修改标记

ttag : 子树修改标记（不包括链上）

rev标记和ctag标记下传方法与link-cut-tree一致

ttag下传方法为：

如果是在实链中的下传，直接下传到ttag, 不修改信息（因为不包括链嘛）

如果是在虚边下传到内部点，下传到ttag并修改。

如故事在虚边下传到外部点，下传到ttag和ctag并进行修改。

当然，对于tag已经提前做好了tag直接合并的operator，以及通过tag更新data的工具函数，以方便后面使用。

3.3 数据信息

数据信息也就是sum，min，max，size，简单的维护，并有相应的合并工具函数。

对于每个节点x，需要维护的一些值：

in : 这个点是否是内部点，白点，无意义的点

val : 点权

以及三个重要的数据信息：

csum : 链上信息和（这里的和为我们定义的加法（也就是合并操作）对应的和）

tsum : 子树信息和（不包括链上）

asum : 所有信息和

以下为统计方法，也就是对应的update操作：

csum[x] = val[x] + csum[c[x][0]] + csum[c[x][1]]

tsum[x] = tsum[c[x][0]] + tsum[c[x][1]] + asum[c[x][2]]+ asum[c[x][3]]

asum[x]= csum[x] + tsum[x]

3.4 splay相关

此处不做过点阐述，主要是因为有内外splay之分，必须要处理好。

3.4 垃圾回收

注意到我们的内部点，也就是白点，至多有N个，所以我们需要垃圾回收，非常简单，在del操作中，我们把不需要的多余的白点记录到一个数据结构（数组）中，当程序再次申请新的内部点时，可以释放这些白点来重新利用。

3.5 add(x, y)

Add(x,y)操作就是从x点连一条虚边到y，使得x是y的父亲。这就等价于在x的虚边splay中插入y这个叶子节点，必要的话要新建内部白点。

3.6 del(x)

Del(x)操作就是把x点和它父亲之间连着的虚边断开。这就等价于在x的父亲的虚边splay中删除x这个叶子节点，必要的话要删除某些已经没用的内部点。

3.7 access(x)

Access(x)操作就是把x到根路径上的所有边都变成实边，并把x向它所有儿子的边都变成虚边。

考虑普通Link-Cut Tree的Access过程：

```
void access(int x){
        for(int y = 0; x; y = x, x = f[x]){
                splay(x);
                c[x][1] = y;
                up(x);
        }
}
```

每一步都是实边虚边的转化，有了Add和Del操作，可以很自然的改写成：

```
   int y = 0;
   for (; x; y = x, x = getfa(x)) {
      splay(x, 0);
      del(y);
      add(x, c[x][1]);
      setson(x, 1, y);
      update(x);
   }
   return y;
```

此处的getfa()取得是真正的非内部点父亲。

3.8 传统操作

lca(虚实交错的地方)

root(找根)

makeroot

link(有点特殊，在lct的基础上，调用add函数)

cut

3.9 链操作

先将x暂时作为根，再access(y)，将y到根x的路径打通，进行链上操作即可，询问也类似。谨记，事先记下真正的根，在操作完后，还原原来的根。

3.10 子树操作

方便起见首先Access(x)，这样x向它的孩子连着的肯定都是虚边，x的子树部分就是x的虚边Splay以及自己本身。于是在虚儿子splay上打上标记，再改改当前这个节点的值即可。询问子树的话，将两部分答案合并即可。

4 时间复杂度

在不严谨的分析下，这个做法的时间复杂度至多不会超过O(sqr(log n))。在tarjan的论文中证明了这样做的复杂度在均摊意义下单次操作是O(log n)的，有常数为96（97）的因子。

5 正确性测试

通过bzoj3153的测试

7 参考资料

1 国家集训队论文 黄志翱《浅谈动态树的相关问题及简单拓展》

2 iamzky http://blog.csdn.net/iamzky/article/details/43494481

3 claris http://www.cnblogs.com/clrs97/p/4403244.html

4 xzj, yzl, zfg AAA-Tree report